Электромагнитная совместимость:

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ЭЛЕКТРОСТАТИКИ

С.П. Куксенко

6 МЕТОД КОНЕЧНЫХ ЭЛЕМЕНТОВ

6.1 Конечные элементы

Математическая трактовка метода конечных элементов (МКЭ, FEM) была предложена Р. Курантом в 1943 г. Первоначально метод применялся при решении задач строительной механики. Для решения электромагнитных задач он начал использоваться с 1968 г. при анализе волноводов, электрических машин, полупроводниковых приборов, микрополосковых линий, электромагнитного излучения биологическими объектами и т. д. По сравнению с МКР и МоМ МКЭ является более мощным и универсальным численным методом, подходящим для решения задач, связанных со сложной геометрией и неоднородными средами. В то же время МКЭ считается сложным с точки зрения его концепции и реализации в программном коде. Методическая общность метода позволяет строить на его основе универсальные компьютерные программы для решения широкого круга задач. При этом программы, разработанные для решения задач из одной предметной области, могут успешно применяться в других областях с незначительными модификациями или без таковых.

Использование МКЭ в общем виде состоит из следующих этапов.

– Дискретизация области решения на конечное число подобластей (конечных элементов) и выбор базисных функций. В рассматриваемой области решения фиксируется конечное число точек, называемых узлами или узловыми точками. Область определения непрерывной величины разбивается на конечное число элементов. Эти элементы имеют общие узловые точки и в совокупности аппроксимируют форму области решения.

– Формирование уравнений для каждого конечного элемента. На данном этапе формируются локальные матрицы, непрерывную величину аппроксимируют на каждом элементе полиномом (функцией элемента), который определяется с помощью узловых значений этой величины. Для каждого элемента подбирают свой

полином таким образом, чтобы сохранить непрерывность величины вдоль границ элемента.

– Сбор всех элементов в области решения в конечно-элементную сетку (ансамблирование). Результаты аппроксимации подставляются в уравнения Максвелла или производные от них с учетом граничных условий. В результате формируется общая СЛАУ.

– Решение общей СЛАУ.

– Вычисление интересующих величин из полученного вектора-решения СЛАУ.

Дискретизация области решения заключается в ее разбиении на подобласти, называемые конечными элементами (КЭ). На рисунке 6.1 показаны некоторые типовые конечные элементы для одномерных, двухмерных и трехмерных задач.

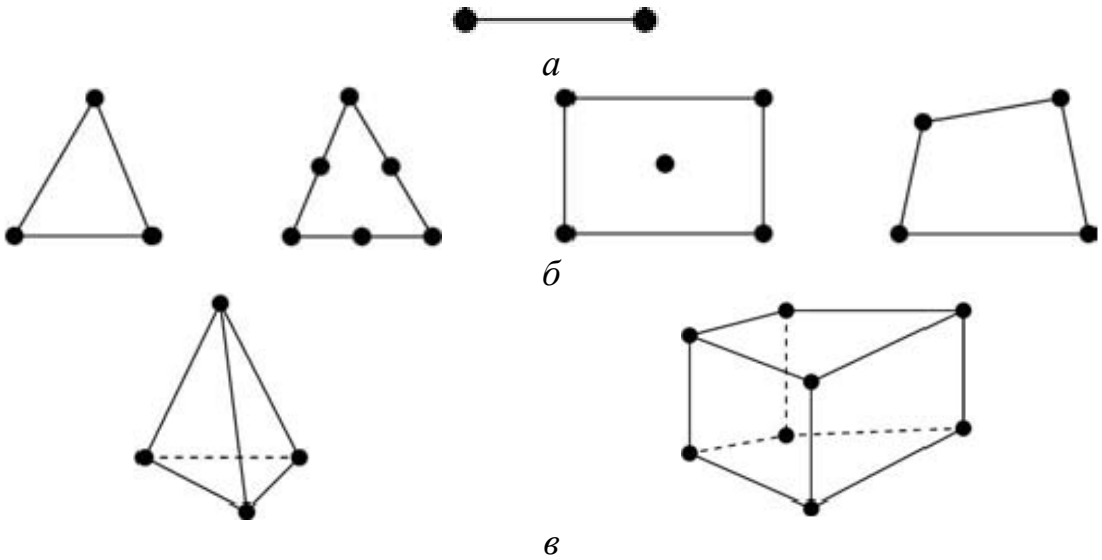


Рисунок 6.1 – Типовые конечные элементы для одномерных (*а*), двухмерных (*б*) и трехмерных (*в*) задач

Самыми распространенными функциями, которые используются для аппроксимации непрерывной функции дискретной моделью, являются полиномы. Они строятся на множестве кусочно-непрерывных функций, определенных на конечном числе подобластей. Порядок полинома в каждом узле элемента зависит от числа используемых данных непрерывной функции. Классификация КЭ может быть выполнена в соответствии с порядком полиномиальных функций этих элементов. Рассматриваются три группы элементов: симплекс-, комплекс- и мультиплекс-элементы.

Первой группе соответствуют полиномы, содержащие константу и линейные члены, в которых число коэффициентов на единицу больше размерности координатного пространства.

Комплекс-элементы являются расширением класса симплекс-элементов. В них могут присутствовать не только линейные слагаемые, но и члены более высоких порядков. Формы комплекс-элементов такие же, как и у симплекс-элементов, но имеют дополнительные граничные узлы и могут иметь внутренние узлы. Главное различие между симплекс- и комплекс-элементами состоит в том, что число узлов у последних больше размерности координатного пространства, увеличенной на единицу.

Мультиплекс-элементы также содержат члены высокого порядка, но границы элементов должны быть параллельны координатным осям, что необходимо для достижения непрерывности при переходе от одного элемента к другому. Это требование достаточно строгое, поэтому мультиплекс-элементы используют в редких случаях.

Рассмотрим одномерный симплекс-элемент, представляющий собой отрезок с двумя узлами (рисунок 6.2).

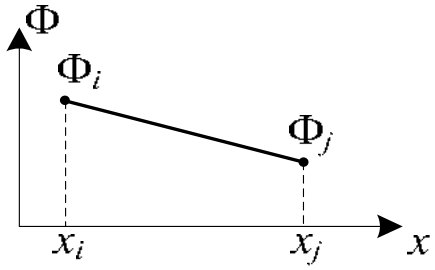


Рисунок 6.2 – Одномерный конечный элемент
с узлами (двухузловой симплекс-элемент)

Обозначим эти узлы индексами i и j, а узловые значения Φ_i и Φ_j соответственно. Начало координат расположено вне рассматриваемого КЭ. Полиномиальная функция имеет вид

$$\Phi = a + bx . \qquad (6.1)$$

Используя условия в узловых точках, можно найти коэффициенты a и b ($x_j - x_i = L$) из системы уравнений

$$\Phi_i = a + bx_i,$$
$$\Phi_j = a + bx_j.$$

Тогда

$$a = \frac{\Phi_i x_j - \Phi_j x_i}{L}, \quad b = \frac{\Phi_j - \Phi_i}{L}. \tag{6.2}$$

Подставив коэффициенты (6.2) в уравнение (6.1), получим

$$\Phi = \frac{x_j - x}{L}\Phi_i + \frac{x - x_i}{L}\Phi_j. \tag{6.3}$$

Линейные функции в равенстве (6.3) называются функциями формы или интерполяционными функциями. Каждая функция формы должна быть дополнена нижним индексом для обозначения узла, к которому она относится. Обозначим $\alpha_i = (x_j - x)/L$ и $\alpha_j = (x_i - x)/L$. Тогда выражение (6.3) примет вид

$$\Phi = \alpha_i\Phi_i + \alpha_j\Phi_j = \begin{pmatrix}\alpha_i & \alpha_j\end{pmatrix}\begin{pmatrix}\Phi_i \\ \Phi_j\end{pmatrix}. \tag{6.4}$$

Очевидно, что функция α_i равна единице в *i*-м узле и равна нулю в *j*-м, а функция α_j наоборот.

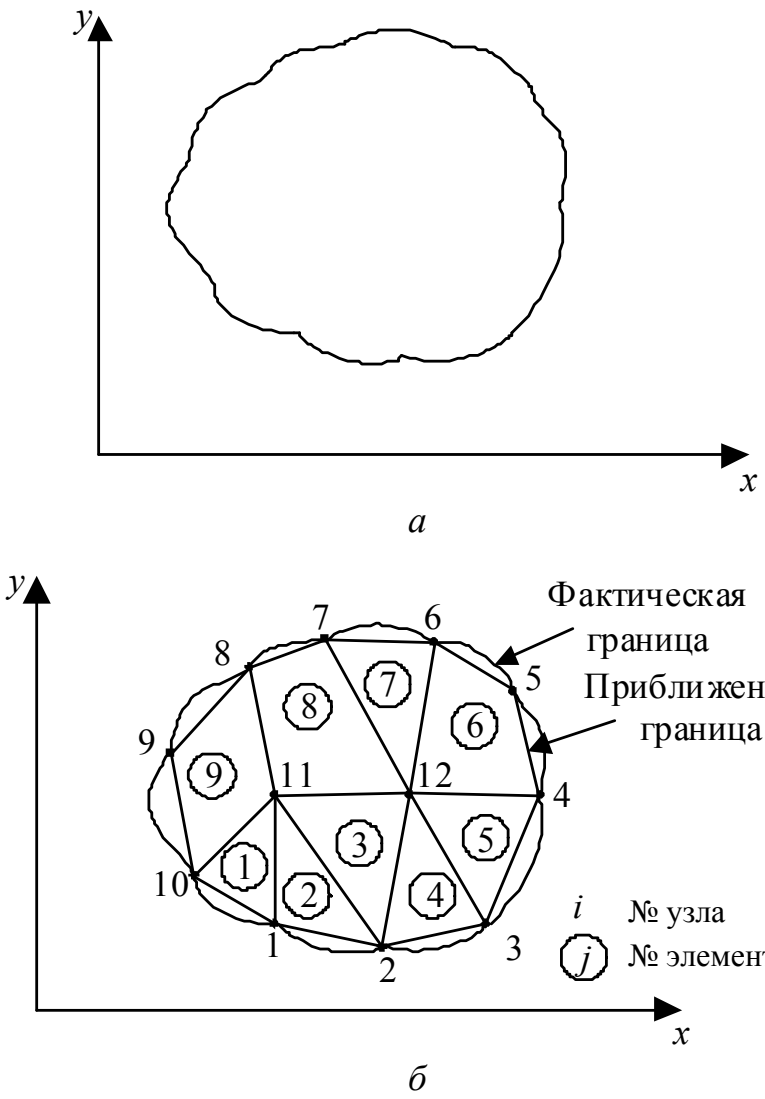
6.2 Решение двухмерного уравнения Лапласа

6.2.1 Дискретизация области

Найдем распределение потенциала $\Phi(x, y)$, соответствующее уравнению Лапласа $\nabla^2\Phi = 0$, в двухмерной области (рисунок 6.3, *а*).

Разделим область решения на несколько КЭ (рисунок 6.3, *б*). В данном случае это девять неперекрывающихся элементов: элементы 6, 8 и 9 – четырехузловые четырехугольники; остальные элементы – трехузловые треугольники. Для удобства вычислений предпочтительно разбивать всю область на элементы одного и того же типа. Так, если разбить четырехугольники на два треугольника каждый, то получим 12 треугольных элементов. Разбиение

областей несложной формы может выполняться вручную, а сложной – с помощью автоматической генерации сетки.



а

б

Рисунок 6.3 – Область решения (*а*)
и ее дискретизация конечными элементами (*б*)

Найдем приближение для потенциала Φ_e внутри элемента e, а затем свяжем распределение потенциала во всех элементах с его непрерывностью на межэлементных границах. Приближенное решение для всей области определяется как

$$\Phi(x, y) = \sum_{e=1}^{N} \Phi_e(x, y), \qquad (6.5)$$

где N – число КЭ, на которое разбита расчетная область.

Для приближенного вычисления потенциала Φ_e внутри элемента используем полиномиальные базисные функции. Для треугольного КЭ

$$\Phi_e(x, y) = a + bx + cy, \tag{6.6}$$

а для четырехугольного КЭ

$$\Phi_e(x, y) = a + bx + cy + dxy. \tag{6.7}$$

Тогда расчетная область, в которой ищется решение, является пространством кусочно-полиномиальных функций.

Таким образом, для приближенного решения должны быть определены коэффициенты a, b, c и d. Потенциал Φ_e отличен от нуля в пределах элемента e и равен нулю вне его. Поскольку треугольные элементы по сравнению с четырехугольными лучше подходят для описания изогнутых границ, то далее используем только их. Линейное изменение потенциала внутри элемента предполагает равномерность электрического поля внутри него, т. е.

$$\mathbf{E}_e = -\nabla\Phi_e = -(b\mathbf{i} + c\mathbf{j}). \tag{6.8}$$

6.2.2 Формирование уравнений отдельного конечного элемента

Рассмотрим типовой треугольный КЭ e (рисунок 6.4). Потенциалы Φ_{e1}, Φ_{e2} и Φ_{e3} в узлах 1, 2 и 3 соответственно получаются с помощью уравнения (6.6):

$$\begin{pmatrix} \Phi_{e1} \\ \Phi_{e2} \\ \Phi_{e3} \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}. \tag{6.9}$$

Коэффициенты a, b и c определяются из СЛАУ (6.9), например, с помощью обратной матрицы:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}^{-1} \begin{pmatrix} \Phi_{e1} \\ \Phi_{e2} \\ \Phi_{e3} \end{pmatrix}. \tag{6.10}$$

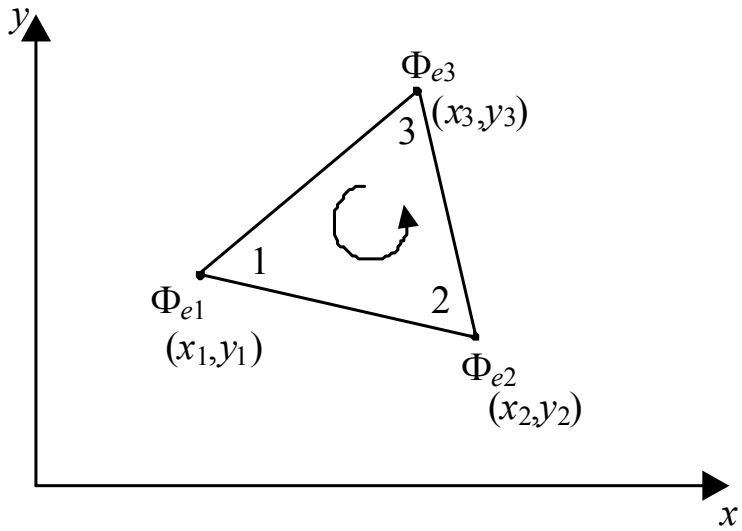


Рисунок 6.4 – Типовой треугольный конечный элемент
(нумерация локальных узлов против часовой стрелки)

Подставив уравнение (6.10) в (6.6) и воспользовавшись обращением матрицы на основе союзной матрицы, получим

$$\Phi_e = \begin{pmatrix} 1 & x & y \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & x & y \end{pmatrix} \frac{1}{2S} \begin{pmatrix} x_2 y_3 - x_3 y_2 & x_3 y_1 - x_1 y_3 & x_1 y_2 - x_2 y_1 \\ y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{pmatrix} \begin{pmatrix} \Phi_{e1} \\ \Phi_{e2} \\ \Phi_{e3} \end{pmatrix}$$

или

$$\Phi_e = \sum_{i=1}^{3} \alpha_i (x, y) \Phi_e^i, \qquad (6.11)$$

где Φ_e^i – потенциал в i-м узле;

$$\alpha_1 = \frac{1}{2S} \left[\left(x_2 y_3 - x_3 y_2 \right) + \left(y_2 - y_3 \right) x + \left(x_3 - x_2 \right) y \right]; \qquad (6.12)$$

$$\alpha_2 = \frac{1}{2S} \left[\left(x_3 y_1 - x_1 y_3 \right) + \left(y_3 - y_1 \right) x + \left(x_1 - x_3 \right) y \right]; \qquad (6.13)$$

$$\alpha_3 = \frac{1}{2S} \left[\left(x_1 y_2 - x_2 y_1 \right) + \left(y_1 - y_2 \right) x + \left(x_2 - x_1 \right) y \right]; \qquad (6.14)$$

S – площадь элемента e, определяемая из уравнения [46]

$$2S = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix} = \left[(x_2 y_3 - x_3 y_2) + (x_3 y_1 - x_1 y_3) + (x_1 y_2 - x_2 y_1) \right],$$

или

$$S = \frac{1}{2}\left[(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \right]. \qquad (6.15)$$

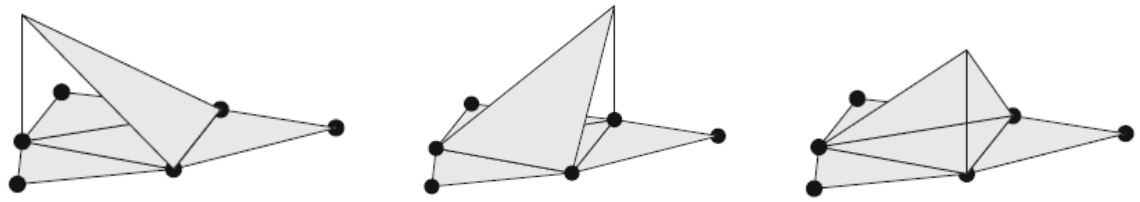
Значение величины S положительно, если узлы пронумерованы против часовой стрелки (начиная с любого узла), как показано стрелкой на рисунке 6.4. Заметим, что выражение (6.11) дает потенциал в любой точке внутри элемента с координатами (x, y) при условии, что потенциалы в его вершинах известны. Это является одним из отличий от конечно-разностного подхода, где потенциал известен только в узлах сетки.

Как и ранее, α_i – функции формы, которые обладают следующими свойствами:

$$\alpha_i = \begin{cases} 1, & i = j, \\ 0, & i \neq j; \end{cases} \qquad (6.16)$$

$$\sum_{i=1}^{3} \alpha_i(x, y) = 1. \qquad (6.17)$$

Функции формы α_1, α_2 и α_3 показаны на рисунке 6.5.



а *б* *в*

Рисунок 6.5 – Функции формы треугольного элемента (также показан смежный элемент): *а* – α_1; *б* – α_2; *в* – α_3

В соответствии с принципом минимума потенциальной энергии распределение потенциала в рассматриваемой области должно быть таким, чтобы минимизировать запасенную энергию

$$W_e = \frac{1}{2}\int \varepsilon |\nabla \Phi_e|^2 d\Omega, \tag{6.18}$$

где интегрирование ведется по всей двухмерной области решения Ω. Физически функционал W_e – это погонная энергия, запасенная в элементе e.

Принцип минимума энергии математически эквивалентен уравнению Лапласа $\nabla^2 \Phi = 0$. Распределение потенциала, удовлетворяющее уравнению Лапласа, соответствует минимальной запасенной энергии, а значение потенциала Φ_e, минимизирующее функционал (6.18), удовлетворяет уравнению Лапласа. Поэтому возможны два варианта решения краевых задач теории поля.

Из уравнения (6.11) получим

$$\nabla \Phi_e = \sum_{i=1}^{3} \Phi_{ei} \nabla \alpha_i. \tag{6.19}$$

Подстановка выражения (6.19) в (6.18) дает

$$W_e = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3} \varepsilon \Phi_{ei} \left[\int \nabla \alpha_i \cdot \nabla \alpha_j d\Omega\right] \Phi_{ei}. \tag{6.20}$$

Если обозначить выражение в скобках через

$$c_{ij}^{(e)} = \int \nabla \alpha_i \cdot \nabla \alpha_j d\Omega, \tag{6.21}$$

то уравнение (6.20) в матричной форме примет вид

$$W_e = \frac{1}{2} \varepsilon \mathbf{\Phi}_e^t \mathbf{C}^{(e)} \mathbf{\Phi}_e, \tag{6.22}$$

где верхний индекс t обозначает транспонирование матрицы;

$$\mathbf{\Phi}_e = \begin{bmatrix} \Phi_{e1} \\ \Phi_{e2} \\ \Phi_{e3} \end{bmatrix}; \tag{6.23}$$

$$\mathbf{C}^{(e)} = \begin{bmatrix} c_{11}^{(e)} & c_{12}^{(e)} & c_{13}^{(e)} \\ c_{21}^{(e)} & c_{22}^{(e)} & c_{23}^{(e)} \\ c_{31}^{(e)} & c_{32}^{(e)} & c_{33}^{(e)} \end{bmatrix}. \tag{6.24}$$

Матрица $\mathbf{C}^{(e)}$ называется матрицей коэффициентов. Элемент $c_{ij}^{(e)}$ этой матрицы соответствует связи между узлами *i* и *j*, а его значение вычисляется по формуле (6.21) с помощью выражений (6.12)–(6.14). Например,

$$c_{12}^{(e)} = \int \nabla\alpha_1 \cdot \nabla\alpha_2 d\Omega =$$

$$= \int \left[\frac{1}{2S}\left(\left(y_2 - y_3\right) + \left(x_3 - x_2\right)\right) \cdot \frac{1}{2S}\left(\left(y_3 - y_1\right) + \left(x_1 - x_3\right)\right) \right] d\Omega =$$

$$= \frac{1}{4S^2}\left[\left(y_2 - y_3\right)\left(y_3 - y_1\right) + \left(x_3 - x_2\right)\left(x_1 - x_3\right)\right] \int d\Omega =$$

$$= \frac{1}{4S}\left[\left(y_2 - y_3\right)\left(y_3 - y_1\right) + \left(x_3 - x_2\right)\left(x_1 - x_3\right)\right]. \qquad (6.25)$$

Аналогично получим

$$c_{13}^{(e)} = \frac{1}{4S}\left[\left(y_2 - y_3\right)\left(y_1 - y_2\right) + \left(x_3 - x_2\right)\left(x_2 - x_1\right)\right]; \qquad (6.26)$$

$$c_{23}^{(e)} = \frac{1}{4S}\left[\left(y_3 - y_1\right)\left(y_1 - y_2\right) + \left(x_1 - x_3\right)\left(x_2 - x_1\right)\right]; \qquad (6.27)$$

$$c_{11}^{(e)} = \frac{1}{4S}\left[\left(y_2 - y_3\right)^2 + \left(x_3 - x_2\right)^2\right]; \qquad (6.28)$$

$$c_{22}^{(e)} = \frac{1}{4S}\left[\left(y_3 - y_1\right)^2 + \left(x_1 - x_3\right)^2\right]; \qquad (6.29)$$

$$c_{33}^{(e)} = \frac{1}{4S}\left[\left(y_1 - y_2\right)^2 + \left(x_2 - x_1\right)^2\right]. \qquad (6.30)$$

При этом

$$c_{21}^{(e)} = c_{12}^{(e)}, c_{31}^{(e)} = c_{13}^{(e)}, c_{32}^{(e)} = c_{23}^{(e)}. \qquad (6.31)$$

Для примера рассмотрим один конечный элемент (рисунок 6.6). На языке Octave этот элемент описывается массивом координат своих узлов [–0.5 0.0 0.6; 0.5 –0.2 0.4].

По выражению (6.15) получим площадь треугольника

$$S = \frac{1}{2}\left[(0.0 + 0.5)(0.4 - 0.5) - (0.6 + 0.5)(-0.2 - 0.5)\right] = 0.36.$$

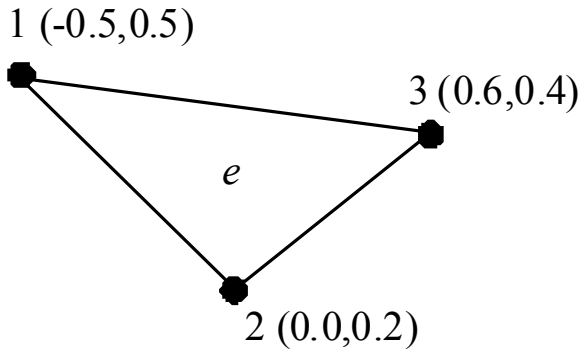


Рисунок 6.6 – Конечный элемент e
с локальной нумерацией узлов

Далее вычислим элементы матрицы коэффициентов $\mathbf{C}^{(e)}$:

$$c_{11}^{(e)} = \frac{1}{4 \cdot 0.36}\left[\left(-0.2 - 0.4\right)^2 + \left(0.6 - 0.0\right)^2 \right] = 0.5;$$

$$c_{12}^{(e)} = \frac{1}{4 \cdot 0.36}\left[\left(-0.2 - 0.4\right)\left(0.4 - 0.5\right) + \left(0.6 - 0.0\right)\left(-0.5 - 0.6\right) \right] = -0.4167;$$

$$c_{13}^{(e)} = \frac{1}{4 \cdot 0.36}\left[\left(-0.2 - 0.4\right)\left(0.5 + 0.2\right) + \left(0.6 - 0.0\right)\left(0.0 + 0.5\right) \right] = -0.0833;$$

$$c_{22}^{(e)} = \frac{1}{4S}\left[\left(y_3 - y_1\right)^2 + \left(x_1 - x_3\right)^2 \right] =$$

$$= \frac{1}{4 \cdot 0.36}\left[\left(0.4 - 0.5\right)^2 + \left(-0.5 - 0.6\right)^2 \right] = 0.8472;$$

$$c_{33}^{(e)} = \frac{1}{4S}\left[\left(y_1 - y_2\right)^2 + \left(x_2 - x_1\right)^2 \right].$$

6.2.3 Ансамблирование

После рассмотрения типовых элементов следующим шагом является объединение отдельных элементов в конечно-элементную сетку в области решения. Данный процесс называется ансамблированием или просто сборкой. С математической точки зрения ансамблирование состоит в объединении матриц коэффициентов отдельных элементов в одну глобальную матрицу. Полная энергия совокупности многих элементов в общем случае равна сумме энергий отдельных элементов:

$$W = \sum_{e=1}^{N} W_e = \frac{1}{2}\varepsilon \mathbf{\Phi}^t \mathbf{C} \mathbf{\Phi}, \tag{6.32}$$

где

$$\mathbf{\Phi} = \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \dots \\ \Phi_n \end{pmatrix}; \tag{6.33}$$

n – общее число узлов; N – общее число КЭ. Матрица **C** является глобальной и состоит из матриц коэффициентов отдельных элементов.

При получении уравнения (6.32) предполагалось, что вся расчетная область Ω однородна, т. е. величина ε постоянна. Дискретизация неоднородной области должна выполняться так, чтобы каждый из конечных элементов был однородным (рисунок 6.7). При этом в уравнении (6.22) от элемента к элементу меняется значение $\varepsilon = \varepsilon_r \varepsilon_0$.

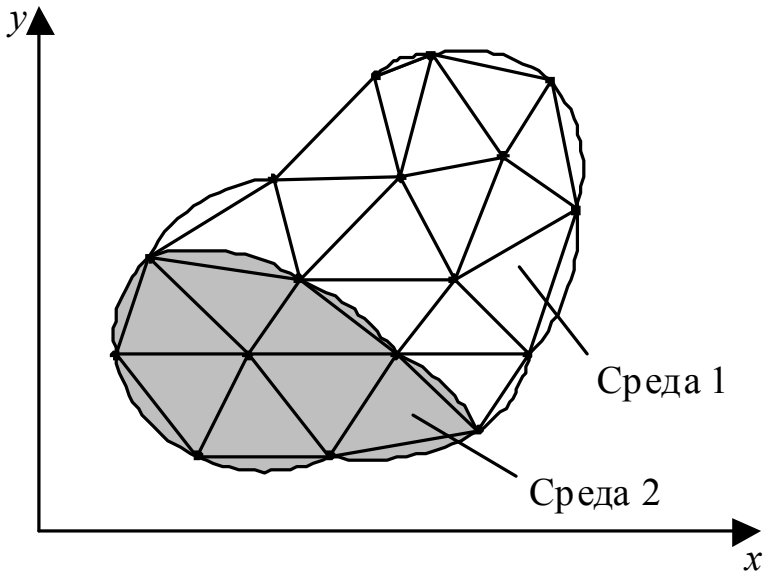


Рисунок 6.7 – Дискретизация неоднородной области решения

Проиллюстрируем процесс ансамблирования. Рассмотрим сетку, состоящую из трех КЭ (рисунок 6.8). Особого внимания заслуживает нумерация узлов сетки. Так, нумерация узлов 1–5

называется глобальной, тогда как нумерация 1-2-3 внутри треугольников называется локальной. Она соответствует нумерации узлов элемента на рисунке 6.4. Например, для элемента 3 на рисунке 6.8 глобальная нумерация 3-5-4 соответствует локальной нумерации 1-2-3, поскольку локальная нумерация должна выполняться последовательно против часовой стрелки, начиная с любого узла элемента. Для этого элемента можно выбрать и нумерацию 4-3-5 вместо 3-5-4, что также соответствует нумерации 1-2-3 для элемента на рисунке 6.4.

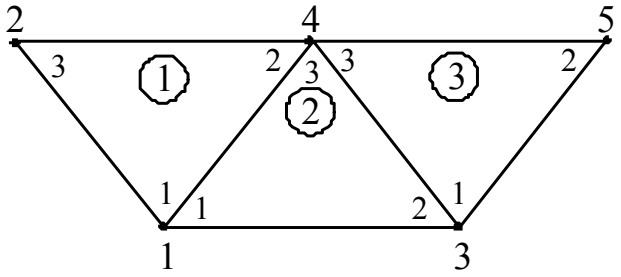


Рисунок 6.8 – Ансамблирование трех элементов

Нумерация на рисунке 6.8 не является уникальной, но важно то, что какая бы нумерация не использовалась, глобальная матрица коэффициентов остается неизменной. Так, задавшись определенной нумерацией, получим матрицу

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix}, \qquad (6.34)$$

которая имеет размер 5×5 по числу глобальных узлов ($n = 5$). Значения элементов c_{ij} вычисляются исходя из того, что распределение потенциала должно быть непрерывным на смежных границах КЭ. Вклад в значение элемента c_{ij} вносят все элементы, содержащие узлы i и j. Например, элементы 1 и 2 содержат общий глобальный узел 1, поэтому

$$c_{11} = c_{11}^{(1)} + c_{11}^{(2)}. \qquad (6.35)$$

Узел 2 соответствует только элементу 1, тогда

$$c_{22} = c_{33}^{(1)}, \tag{6.36}$$

а узел 4 – элементам 1, 2 и 3, а значит

$$c_{44} = c_{22}^{(1)} + c_{33}^{(2)} + c_{33}^{(3)}. \tag{6.37}$$

Узлы 1 и 4 принадлежат элементам 1 и 2, поэтому

$$c_{14} = c_{41} = c_{12}^{(1)} + c_{13}^{(2)}, \tag{6.38}$$

а поскольку между узлами 2 и 3 нет прямой связи, то

$$c_{23} = c_{32} = 0. \tag{6.39}$$

Аналогично вычисляются остальные элементы глобальной матрицы и в результате она принимает вид

$$\mathbf{C} = \begin{bmatrix} c_{11}^{(1)} + c_{11}^{(2)} & c_{13}^{(1)} & c_{12}^{(2)} & c_{12}^{(1)} + c_{13}^{(2)} & 0 \\ c_{31}^{(1)} & c_{33}^{(1)} & 0 & c_{32}^{(1)} & 0 \\ c_{21}^{(2)} & 0 & c_{22}^{(2)} + c_{11}^{(3)} & c_{23}^{(2)} + c_{13}^{(3)} & c_{12}^{(3)} \\ c_{21}^{(1)} + c_{31}^{(2)} & c_{23}^{(1)} & c_{32}^{(2)} + c_{31}^{(3)} & c_{22}^{(1)} + c_{33}^{(2)} + c_{33}^{(3)} & c_{32}^{(3)} \\ 0 & 0 & c_{21}^{(3)} & c_{23}^{(3)} & c_{22}^{(3)} \end{bmatrix}. \tag{6.40}$$

Таким образом, значения матрицы коэффициентов суммируются в узлах, являющихся общими для разных элементов (в рассматриваемом случае 1, 4 и 3, 4). Видно, что глобальная матрица содержит 27 ненулевых элементов (по 9 от каждого из трех элементов). Также она обладает следующими свойствами:

– она симметрична, как и матрицы коэффициентов элементов;

– поскольку есть элементы $c_{ij} = 0$ (между узлами i и j нет прямой связи), то при их большом числе она становится разреженной. Элементы матрицы также группируются (она становится ленточной или блочно-диагональной), если узлы надлежащим образом пронумерованы. Это можно показать, используя уравнения (6.25)–(6.30) и тот факт, что

$$\sum_{i=1}^{3} c_{ij}^{(e)} = \sum_{j=1}^{3} c_{ij}^{(e)} = 0;$$

– она сингулярная. Это можно показать, используя выражение (6.24).

6.2.4 Решение результирующего матричного уравнения

Используя понятия вариационного исчисления, можно показать, что уравнение Лапласа выполняется, когда полная энергия в области решения Ω минимальна. Таким образом, требуется, чтобы частные производные W по отношению к величине потенциала в каждом узле были равными нулю:

$$\frac{\partial W}{\partial \Phi_1} = \frac{\partial W}{\partial \Phi_2} = ... = \frac{\partial W}{\partial \Phi_n} = 0$$

или

$$\frac{\partial W}{\partial \Phi_k} = 0,\ k = 1,\ 2,\ ...,\ n. \qquad (6.41)$$

Например, получить $\partial W/\partial \Phi_1 = 0$ для сетки конечных элементов на рисунке 6.8 можно, подставив матрицу (6.34) в уравнение (6.32) и взяв частную производную W по Φ_1. Тогда

$$2\Phi_1 c_{11} + \Phi_2 c_{12} + \Phi_3 c_{13} + \Phi_4 c_{14} +$$

$$+ \Phi_5 c_{15} + \Phi_2 c_{21} + \Phi_3 c_{31} + \Phi_4 c_{41} + \Phi_5 c_{51} = 0,$$

что с учетом симметрии матрицы **C** дает

$$0 = \Phi_1 c_{11} + \Phi_2 c_{12} + \Phi_3 c_{13} + \Phi_4 c_{14} + \Phi_5 c_{15}. \qquad (6.42)$$

В общем виде при $\partial W/\partial \Phi_k = 0$ получим

$$0 = \sum_{i=1}^{n} \Phi_k c_{ik}, \qquad (6.43)$$

где n – число узлов сетки. Записав уравнение (6.43) для всех узлов $k = 1, 2, \ldots, n$, получим СЛАУ, из которой найдем решение $\mathbf{\Phi}^T = [\Phi_1, \Phi_2, \ldots, \Phi_n]$. Для этого можно воспользоваться теми же методами, которые применяются при решении конечно-разностных уравнений.

Сначала используем итерационный метод. Например, предположим, что узел 1 на рисунке 6.8 является свободным (значение потенциала в нем неизвестно). Из уравнения (6.42) получим

$$\Phi_1 = -\frac{1}{c_{11}} \sum_{i=2}^{5} \Phi_i c_{1i}. \qquad (6.44)$$

Таким образом, в общем случае для сетки, состоящей из n узлов, в k-м узле имеем

$$\Phi_k = -\frac{1}{c_{kk}} \sum_{i=1, i\neq k}^{n} \Phi_i c_{ki} \qquad (6.45)$$

где узел k является свободным. Если между узлами k и i нет прямой связи, то $c_{ki} = 0$. Следовательно, только узлы, которые имеют прямую связь с узлом k, вносят вклад в Φ_k согласно выражению (6.45). Уравнение (6.45) можно применить итерационно ко всем свободным узлам. Итерационный процесс начинается с задания значений потенциалов в граничных узлах. Хорошим начальным приближением для свободных узлов является их равенство нулю или среднему значению потенциала:

$$\Phi_{\text{ср}} = 0.5(\Phi_{\min} + \Phi_{\max}), \qquad (6.46)$$

где $\Phi_{\min}$ и $\Phi_{\max}$ – минимальное и максимальное значения потенциала в граничных узлах. При этих начальных значениях потенциалы в свободных узлах вычисляются с использованием выражения (6.45). В конце первой итерации, когда для всех свободных узлов вычислены новые значения, они становятся старыми значениями для второй итерации. Процедура повторяется до тех пор, пока разница между значениями, полученными на текущей и предыдущей итерациях, не будет достаточно малой (с требуемой точностью).

Теперь рассмотрим метод решения ленточных систем. Если все свободные узлы пронумерованы первыми, а граничные узлы – последними, то уравнение (6.22) может быть записано в виде

$$W = \frac{1}{2}\varepsilon \begin{bmatrix} \boldsymbol{\Phi}_f & \boldsymbol{\Phi}_p \end{bmatrix} \begin{bmatrix} \mathbf{C}_{ff} & \mathbf{C}_{fp} \\ \mathbf{C}_{pf} & \mathbf{C}_{pp} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_f \\ \boldsymbol{\Phi}_p \end{bmatrix}, \qquad (6.47)$$

где индексы f и p соответствуют узлам со свободными и граничными значениями потенциала. Поскольку матрица $\mathbf{\Phi}_p$ постоянна, то дифференцировать уравнение (6.47) согласно (6.41) нужно только по $\mathbf{\Phi}_f$. В результате получим

$$\frac{\partial W}{\mathbf{\Phi}_f} = \begin{bmatrix} \mathbf{C}_{ff} & \mathbf{C}_{fp} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_f \\ \mathbf{\Phi}_p \end{bmatrix} = 0$$

или

$$\mathbf{C}_{ff}\mathbf{\Phi}_f = -\mathbf{C}_{fp}\mathbf{\Phi}_p. \tag{6.48}$$

СЛАУ (6.48) запишем в матричном виде:

$$\mathbf{A\Phi} = \mathbf{B}, \tag{6.49}$$

где $\mathbf{\Phi} = \mathbf{\Phi}_f$; $\mathbf{A} = \mathbf{C}_{ff}$; $\mathbf{B} = -\mathbf{C}_{fp}\mathbf{\Phi}_p$. Так как матрица $\mathbf{A}$ является невырожденной, то потенциал в свободных узлах можно найти, используя выражение (6.49), с помощью подходящего метода решения СЛАУ.

Ранее при описании МКР было показано, что на границах области решения или на линиях симметрии (в случае их использования) иногда необходимо накладывать условие Неймана ($\partial\Phi/\partial n = 0$). Предположим, что область решения симметрична вдоль оси y (рисунок 6.9).

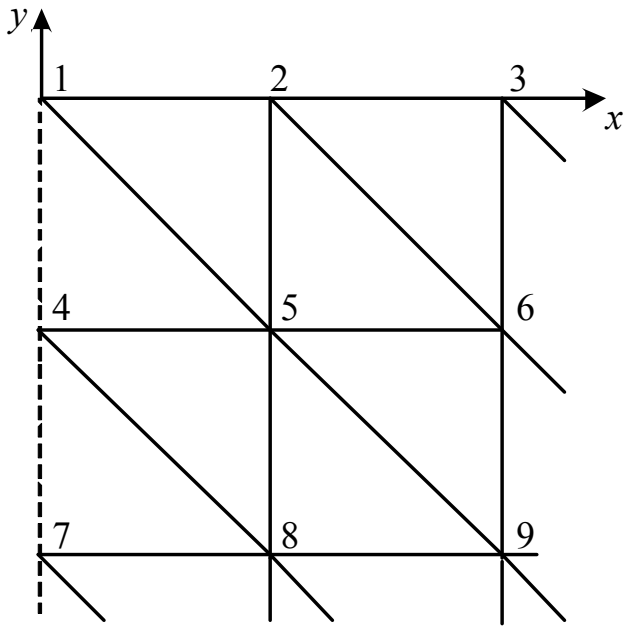


Рисунок 6.9 – Область решения, симметричная вдоль оси y

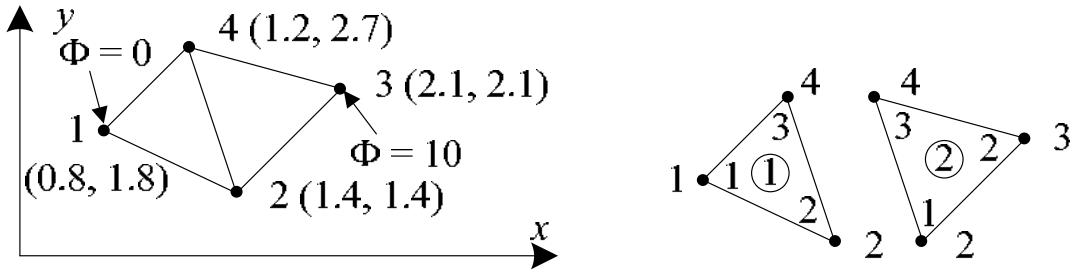
Тогда для выполнения условия Неймана необходимо положить

$$\Phi_1 = \Phi_2,\ \Phi_4 = \Phi_5,\ \Phi_7 = \Phi_8. \qquad (6.50)$$

Отметим, что согласно уравнению (6.18) решение ищется в двухмерной области и соответствует уравнению Лапласа $\nabla^2\Phi = 0$. Рассмотренные выше основные понятия распространяются также на конечно-элементный анализ задач, связанных с решением уравнения Пуассона и волнового уравнения.

Пример 6.1

Найти потенциалы внутри заданной сетки (рисунок 6.10), используя МКЭ.



а *б*

Рисунок 6.10 – Конечно-элементная сетка (*а*), локальная и глобальная нумерация ее узлов (*б*)

Решение

Матрицы коэффициентов для элементов можно вычислить с использованием уравнений (6.25)–(6.31). Однако вычисления можно упростить, если положить

$$\begin{aligned} P_1 &= (y_2 - y_3),\ P_2 = (y_3 - y_1),\ P_3 = (y_1 - y_2), \\ Q_1 &= (x_3 - x_2),\ Q_2 = (x_1 - x_3),\ Q_3 = (x_2 - x_1), \end{aligned} \qquad (6.51)$$

где P_i и Q_i $(i = 1, 2, 3)$ – номера локальных узлов. Тогда каждый элемент матрицы вычисляется с помощью выражения

$$c_{ij}^{(e)} = \frac{1}{4S}\left(P_i P_j + Q_i Q_j\right), \qquad (6.52)$$

где $S = 0.5(P_2 Q_3 - P_3 Q_2)$. Формула (6.52) более удобна для использования по сравнению с уравнениями (6.25)–(6.31). Применив

выражение (6.52) для элемента 1, содержащего глобальные узлы 1-2-4, соответствующие локальным узлам 1-2-3 (рисунок 6.10, *б*), получим

$$P_1 = (1.4 - 2.7) = -1.3;\ P_2 = (2.7 - 1.8) = 0.9;\ P_3 = (1.8 - 1.4) = 0.4;$$

$$Q_1 = (1.2 - 1.4) = -0.2;\ Q_2 = (0.8 - 1.2) = -0.4;$$

$$Q_3 = (1.4 - 0.8) = 0.6;\ S = 0.5(0.54 + 0.16) = 0.35.$$

Подстановка вычисленных значений в формулу (6.52) дает

$$\mathbf{C}^{(1)} = \begin{bmatrix} 1.2357 & -0.7786 & -0.4571 \\ -0.7786 & 0.6929 & 0.0857 \\ -0.4571 & 0.0857 & 0.3714 \end{bmatrix}. \qquad (6.53)$$

Аналогично для элемента 2, содержащего глобальные узлы 2-3-4, соответствующие локальным узлам 1-2-3 (см. рисунок 6.10, *б*), получим

$$P_1 = -0.6;\ P_2 = 1.3;\ P_3 = -0.7;$$

$$Q_1 = -0.9;\ Q_2 = 0{,}2;\ Q_3 = 0.7;$$

$$S = 0.5(0.91 + 0.14) = 0.525.$$

Тогда

$$\mathbf{C}^{(2)} = \begin{bmatrix} 0.5571 & -0.4571 & -0.1 \\ -0.4571 & 0.8238 & -0.3667 \\ -0.1 & -0.3667 & 0.4667 \end{bmatrix}. \qquad (6.54)$$

Важно, что при вычислении элементов матрицы отдельного КЭ необходимо придерживаться локальной нумерации, а глобальную нумерацию следует использовать только при формировании глобальной матрицы. В результате вычислим элементы глобальной матрицы:

$$\mathbf{C}_{22} = \mathbf{C}_{22}^{(1)} + \mathbf{C}_{11}^{(2)} = 0.6929 + 0.5571 = 1.25;$$

$$\mathbf{C}_{24} = \mathbf{C}_{23}^{(1)} + \mathbf{C}_{13}^{(2)} = 0.0857 - 0.1 = -0.0143;$$

$$\mathbf{C}_{44} = \mathbf{C}_{33}^{(1)} + \mathbf{C}_{33}^{(2)} = 0.3714 + 0.4667 = 0.8381;$$

$$\mathbf{C}_{21} = \mathbf{C}_{21}^{(1)} = -0.7786;$$

$$\mathbf{C}_{23} = \mathbf{C}_{12}^{(2)} = -0.4571;$$

$$\mathbf{C}_{41} = \mathbf{C}_{31}^{(1)} = -0.4571$$

$$\mathbf{C}_{43} = \mathbf{C}_{32}^{(2)} = -0.3667.$$

Теперь найдем глобальную матрицу:

$$\mathbf{C} = \begin{pmatrix} c_{11}^{(1)} & c_{12}^{(1)} & 0 & c_{13}^{(1)} \\ c_{21}^{(1)} & c_{22}^{(1)} + c_{11}^{(2)} & c_{12}^{(2)} & c_{23}^{(1)} + c_{12}^{(2)} \\ 0 & c_{21}^{(2)} & c_{22}^{(2)} & c_{23}^{(2)} \\ c_{31}^{(1)} & c_{32}^{(1)} + c_{31}^{(2)} & c_{32}^{(2)} & c_{33}^{(1)} + c_{33}^{(2)} \end{pmatrix} =$$

$$= \begin{pmatrix} 1.2357 & -0.7786 & 0 & -0.4571 \\ -0.7786 & 1.25 & -0.4571 & -0.0143 \\ 0 & -0.4571 & 0.8238 & -0.3667 \\ -0.4571 & -0.0143 & -0.3667 & 0.8381 \end{pmatrix}. \qquad (6.55)$$

Видно, что $\sum_{i=1}^{4} c_{ij} = \sum_{j=1}^{4} c_{ij} = 0$, это говорит о правильности вычисления элементов матрицы. Применив уравнение (6.45) к свободным узлам 2 и 4, получим

$$\Phi_2 = -\frac{1}{c_{22}}\left(\Phi_1 c_{12} + \Phi_3 c_{32} + \Phi_4 c_{42}\right) = -\frac{1}{1,25}\left(-4,571 - 0,0143\Phi_4\right),$$

$$\Phi_4 = -\frac{1}{c_{44}}\left(\Phi_1 c_{14} + \Phi_2 c_{24} + \Phi_3 c_{34}\right) = -\frac{1}{0,8381}\left(-0,143\Phi_2 - 3,667\right). \qquad (6.56)$$

Используя нулевое начальное приближение и выражения (6.56), найдем после первой итерации $\Phi_2 = 3.6568$, $\Phi_4 = 4.4378$, а после второй – $\Phi_2 = 3.7075$, $\Phi_4 = 4.4386$.

Итерационная схема вычисления обычно имеет быструю сходимость и предпочтительна при большом числе узлов. Как только значения потенциалов в узлах становятся известны, потенциал в любой точке сетки можно определить с помощью уравнения (6.11).

Пример 6.2

Решить уравнение Лапласа для двухмерной области, изображенной на рисунке 6.11, *а*.

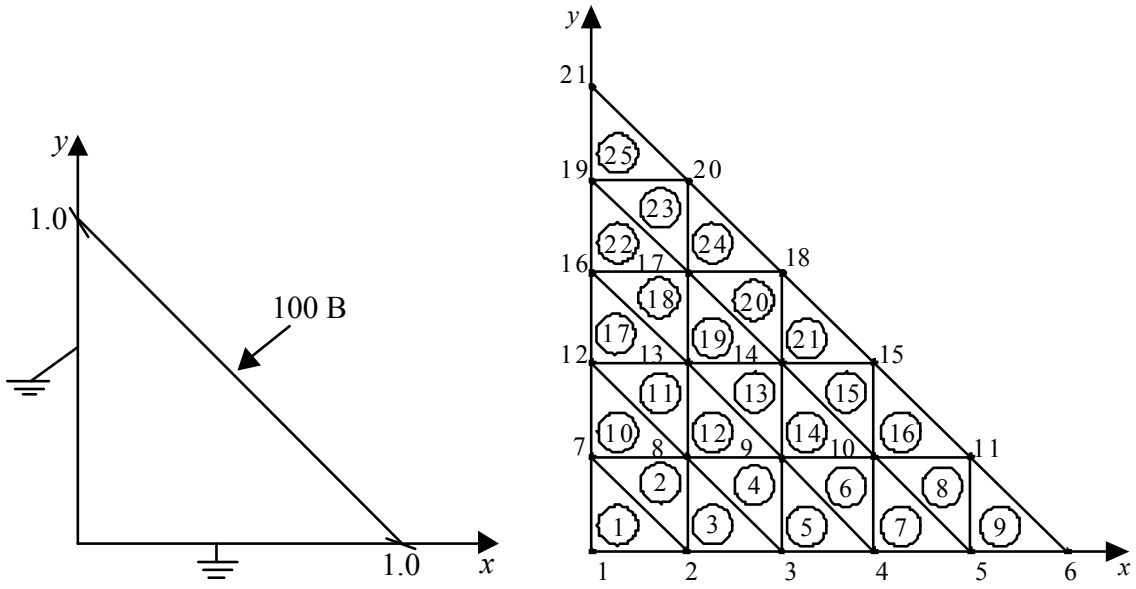


Рисунок 6.11 – Двухмерная область решения (*а*)
и конечно-элементная сетка (*б*)

Решение

На этапе 1 необходимо определить начальные данные для выполнения вычислений. Отметим, что это единственный этап, который зависит от геометрии задачи. Здесь определяется число конечных элементов, узлов, граничных узлов, значения потенциала в свободных узлах, координаты x и y всех узлов и список, идентифицирующий узлы, принадлежащие каждому элементу в порядке локальной нумерации 1-2-3. Для решения задачи разделим расчетную область на 25 треугольных КЭ с числом глобальных узлов 21 (рисунок 6.11, *б*). В результате получим три набора данных: координаты, отношения элемент-узел и заданные потенциалы на граничных узлах (таблицы 6.1–6.3).

На этапе 2 в соответствии с п. 6.2.2 и п. 6.2.3 выполняется вычисление значений элементов матриц $\mathbf{C}^{(e)}$ для каждого КЭ, а также их ансамблирование в глобальную матрицу $\mathbf{C}$.

На этапе 3 формируется список свободных узлов с использованием списка значений потенциалов в граничных узлах. Далее ко

всем свободным узлам итерационно применяется уравнение (6.45). В данном примере для сходимости требуется порядка 50 итераций, поскольку задействовано только 6 узлов.

На этапе 4 выполняется вывод результатов вычислений (таблица 6.4).

Таблица 6.1 – Координаты узлов (рисунок 6.11)

Узел	*x*	*y*	Узел	*x*	*y*
1	0	0	12	0	0.4
2	0.2	0	13	0.2	0.4
3	0.4	0	14	0.4	0.4
4	0.6	0	15	0.6	0.4
5	0.8	0	16	0	0.6
6	1	0	17	0.2	0.6
7	0	0.2	18	0.4	0.6
8	0.2	0.2	19	0	0.8
9	0.4	0.2	20	0.2	0.8
10	0.6	0.2	21	0	1
11	0.8	0.2			

Таблица 6.2 – Соответствие между элементами и узлами (рисунок 6.11)

Элемент	Локальный	Узел	Номер	Элемент	Локальный	Узел	Номер
	1	2	3		1	2	3
1	1	2	7	14	9	10	14
2	2	8	7	15	10	15	14
3	2	3	8	16	10	11	15
4	3	9	8	17	12	13	16
5	3	4	9	18	13	17	16
6	4	10	9	19	13	14	17
7	4	5	10	20	14	18	17
8	5	11	10	21	14	15	18
9	5	6	11	22	16	17	19
10	7	8	12	23	17	20	19
11	8	13	12	24	17	18	20
12	8	9	13	25	19	20	21
13	9	14	13				

Таблица 6.3 – Значения потенциала в граничных узлах (рисунок 6.11)

Узел	Значение потенциала	Узел	Значение потенциала
1	0	18	100
2	0	20	100
3	0	21	50
4	0	19	0
5	0	16	0
6	50	12	0
11	100	7	0
15	100		

Таблица 6.4 – Выходные данные для задачи из примера 6.2 (число узлов 21, число КЭ 25, граничных узлов 15)

Узел	*X*	*Y*	Потенциал
1	0	0	0
2	0.2	0	0
3	0.4	0	0
4	0.6	0	0
5	0.8	0	0
6	1	0	50
7	0	0.2	0
8	0.2	0.2	18.182
9	0.4	0.2	36.364
10	0.6	0.2	59.091
11	0.8	0.2	100
12	0	0.4	0
13	0.2	0.4	36.364
14	0.4	0.4	68.182
15	0.6	0.4	100
16	0	0.6	0
17	0.2	0.6	59.091
18	0.4	0.6	100
19	0	0.8	0
20	0.2	0.8	100
21	0	1	50

При использовании МКР получены следующие значения: $\Phi_8 = 15.41$; $\Phi_9 = 26.74$; $\Phi_{10} = 56.69$; $\Phi_{13} = 34.88$; $\Phi_{14} = 65.41$;

$\Phi_{17} = 58.72$ [32]. Несмотря на то что эти значения достаточно точные, в данной задаче можно добиться повышения точности методом конечных элементов путем деления области решения на большее число КЭ или за счет использования элементов более высокого порядка. Как упоминалось ранее, МКЭ имеет два основных преимущества по сравнению с МКР. Во-первых, при использовании МКР значения потенциала могут быть получены только в узлах сетки области решения, а при использовании МКЭ – в любой точке области решения. Во-вторых, МКЭ лучше подходит для решения задач со сложной геометрией.

6.3 Решение уравнения Пуассона

Рассмотрим особенности решения двухмерного уравнения Пуассона

$$\nabla^2 \Phi = -\frac{\rho}{\varepsilon}. \tag{6.57}$$

При его решении выполняются те же этапы, что и при решении уравнения Лапласа. Главное отличие заключается только в необходимости учета члена в правой части уравнения. Поэтому отметим лишь основные различия в этапах при решении этих уравнений.

После того как область решения разбита на треугольные КЭ, необходимо аппроксимировать распределение потенциала $\Phi_e(x, y)$ и поверхностной плотности заряда ρ_e линейными комбинациями интерполяционного многочлена α_i на каждом КЭ, т. е.

$$\Phi_e = \sum_{i=1}^{3} \Phi_{ei} \alpha_i(x, y), \tag{6.58}$$

$$\rho_e = \sum_{i=1}^{3} \rho_{ei} \alpha_i(x, y). \tag{6.59}$$

Коэффициенты Φ_{ei} и ρ_{ei} соответствуют значениям в i-м узле e-го элемента. Значения ρ_{ei} известны, поскольку $\rho(x, y)$ известно, а значения Φ_{ei} должны быть вычислены.

Функционал, описывающийся уравнением (6.57), ассоциируется с уравнением Эйлера

$$F(\Phi_e) = \frac{1}{2}\int_{\Omega}\left(\varepsilon \mid \nabla\Phi_e \mid^2 -2\rho_e\Phi_e\right)d\Omega, \tag{6.60}$$

где $F(\Phi_e)$ представляет собой погонную энергию, запасенную внутри e-го элемента. Первый член под знаком интеграла $\frac{1}{2}\mathbf{D}\cdot\mathbf{E} = \frac{1}{2}\varepsilon|\nabla\Phi_e|^2$ – это плотность энергии в электростатической системе, а второй член $\rho_e\Phi_e d\Omega$ – работа, совершаемая при перемещении заряда $\rho_e d\Omega$ в место, где находится потенциал Φ_e. После подстановки выражений (6.58) и (6.59) в уравнение (6.60) получим

$$F(\Phi_e) = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\varepsilon\Phi_{ei}\left[\int\nabla\alpha_i\cdot\nabla\alpha_j d\Omega\right]\Phi_{ej} - \sum_{i=1}^{3}\sum_{j=1}^{3}\Phi_{ei}\left[\int\alpha_i\alpha_j d\Omega\right]\rho_{ej}$$

или в матричной форме

$$F(\Phi_e) = \frac{1}{2}\varepsilon\mathbf{\Phi}_e^T\mathbf{C}^{(e)}\mathbf{\Phi}_e - \mathbf{\Phi}_e^T\mathbf{T}^{(e)}\boldsymbol{\rho}_e, \tag{6.61}$$

где

$$c_{ij}^{(e)} = \int\nabla\alpha_i\cdot\nabla\alpha_j d\Omega, \tag{6.62}$$

что аналогично уравнению (6.25), а

$$t_{ij}^{(e)} = \int\alpha_i\alpha_j d\Omega. \tag{6.63}$$

Можно показать, что

$$t_{ij}^{(e)} = \begin{cases} S/12, & \text{если} \quad i \neq j, \\ S/6, & \text{если} \quad i = j, \end{cases} \tag{6.64}$$

где S – площадь треугольного элемента.

Уравнение (6.61) может быть применено к каждому элементу в области решения. Тогда получим дискретный аналог функционала для всей области решения (с N элементами и n узлами) в виде суммы функционалов для отдельных элементов:

$$F(\Phi) = \sum_{e=1}^{N}F(\Phi_e) = \frac{1}{2}\varepsilon\mathbf{\Phi}^T\mathbf{C}\mathbf{\Phi} - \mathbf{\Phi}^T\mathbf{T}\boldsymbol{\rho}. \tag{6.65}$$

В данном уравнении матрица-строка **Φ** состоит из значений Φ_{ei}, а матрица-строка **ρ** – из n значений исходной функции ρ в узлах сетки.

Результирующие уравнения могут быть решены итерационно или с помощью метода для ленточных матриц, как показано в п. 6.2.4. Далее используем итерационный подход. Рассмотрим область решений на рисунке 6.8, имеющую пять узлов (n = 5). С помощью уравнения (6.65) получим

$$F(\Phi)=\frac{1}{2}\varepsilon\begin{pmatrix}\Phi_1 & \Phi_2 & \dots & \Phi_5\end{pmatrix}\begin{pmatrix}c_{11} & c_{12} & \dots & c_{15}\\ c_{21} & c_{22} & \dots & c_{25}\\ \dots & \dots & \dots & \dots\\ c_{51} & c_{52} & \dots & c_{55}\end{pmatrix}\begin{pmatrix}\Phi_1\\ \Phi_2\\ \dots\\ \Phi_5\end{pmatrix}-$$

$$-\begin{pmatrix}\Phi_1 & \Phi_2 & \dots & \Phi_5\end{pmatrix}\begin{pmatrix}t_{11} & t_{12} & \dots & t_{15}\\ t_{21} & t_{22} & \dots & t_{25}\\ \dots & \dots & \dots & \dots\\ t_{51} & t_{52} & \dots & t_{55}\end{pmatrix}\begin{pmatrix}\rho_1\\ \rho_2\\ \dots\\ \rho_5\end{pmatrix}. \qquad (6.66)$$

Минимизируем энергию:

$$\frac{\partial F}{\partial \Phi_k}=0,\quad k=1,2,...,n. \qquad (6.67)$$

Для $\frac{\partial F}{\partial \Phi_1}=0$ из уравнения (6.66) найдем

$$\frac{\partial F}{\partial \Phi_1}=\varepsilon\left[\Phi_1 c_{11}+\Phi_2 c_{21}+...+\Phi_5 c_{51}\right]-\left[t_{11}\rho_1+t_{12}\rho_2+...+t_{51}\rho_5\right]=0,$$

$$\Phi_1=-\frac{1}{c_{11}}\sum_{i=2}^{5}\Phi_i c_{i1}+\frac{1}{\varepsilon c_{11}}\sum_{i=1}^{5}t_{i1}\rho_i. \qquad (6.68)$$

Таким образом, в общем случае для сетки, содержащей n узлов, имеем

$$\Phi_k=-\frac{1}{c_{kk}}\sum_{i=1,i\neq k}^{n}\Phi_i c_{ik}+\frac{1}{\varepsilon c_{kk}}\sum_{i=1}^{n}t_{ik}\rho_i, \qquad (6.69)$$

где k считается свободным узлом.

Для получения решения, как и при решении уравнения Лапласа, необходимо итерационно применить уравнение (6.69) ко всем свободным узлам до достижения сходимости, зафиксировав потенциалы в граничных узлах и первоначально установив потенциалы в свободных узлах (например, равными нулю).

Теперь рассмотрим процесс решения на основе системы с ленточной матрицей. Для этого необходимо пронумеровать сначала свободные узлы, а затем граничные. Тогда уравнение (6.65) можно записать в виде

$$F(\Phi)=\frac{1}{2}\varepsilon\begin{bmatrix}\boldsymbol{\Phi}_f & \boldsymbol{\Phi}_p\end{bmatrix}\begin{bmatrix}\mathbf{C}_{ff} & \mathbf{C}_{fp}\\ \mathbf{C}_{pf} & \mathbf{C}_{pp}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Phi}_f\\ \boldsymbol{\Phi}_p\end{bmatrix}-$$

$$-\begin{bmatrix}\boldsymbol{\Phi}_f & \boldsymbol{\Phi}_p\end{bmatrix}\begin{bmatrix}\mathbf{T}_{ff} & \mathbf{T}_{fp}\\ \mathbf{T}_{pf} & \mathbf{T}_{pp}\end{bmatrix}\begin{bmatrix}\boldsymbol{\rho}_f\\ \boldsymbol{\rho}_p\end{bmatrix}. \qquad (6.70)$$

Минимизируя $F(\Phi)$ по Φ_f, получим

$$\frac{\partial F(\Phi)}{\partial \boldsymbol{\Phi}_f}=0=\varepsilon\left(\mathbf{C}_{ff}\boldsymbol{\Phi}_f+\mathbf{C}_{fp}\boldsymbol{\Phi}_p\right)-\left(\mathbf{T}_{ff}\boldsymbol{\rho}_f+\mathbf{T}_{fp}\boldsymbol{\rho}_p\right)$$

или

$$\mathbf{C}_{ff}\boldsymbol{\Phi}_f=-\mathbf{C}_{fp}\boldsymbol{\Phi}_p+\frac{1}{\varepsilon}\mathbf{T}_{ff}\boldsymbol{\rho}_f+\frac{1}{\varepsilon}\mathbf{T}_{fp}\boldsymbol{\rho}_p. \qquad (6.71)$$

Данное выражение в матричной форме имеет вид

$$\mathbf{A\Phi}=\mathbf{B},$$

где $\mathbf{A}=\mathbf{C}_{ff}$, $\boldsymbol{\Phi}=\boldsymbol{\Phi}_f$; $\mathbf{B}$ – правая часть выражения (6.71). Полученную СЛАУ можно решить относительно $\boldsymbol{\Phi}$ любым подходящим методом. Следует отметить, что различия между уравнениями (6.45), (6.48) и (6.69), (6.71) незначительны.

6.4 Решение уравнения Гельмгольца

Для демонстрации универсальности МКЭ рассмотрим решение одномерного уравнения Гельмгольца

$$-\frac{d}{dx}\left(\alpha\frac{df}{dx}\right)+\beta f=s,\ \ a<x<b, \qquad (6.72)$$

$$f(a) = f_a, \tag{6.73}$$

$$f(b) = f_b, \tag{6.74}$$

где функция $f(x)$ – искомое решение; $\alpha = \alpha(x)$ и $\beta = \beta(x)$ – функции, определяющие свойства материала; $s = s(x)$ – параметры источника.

Найдем функцию $f(x)$ на интервале $a < x < b$. Для ясности положим $a = -2$, $b = 5$ и разделим ось x на 7 одинаковых подынтервалов. Пронумеровав узлы подынтервала, получим их координаты $x_i = i - 3$, $i = 1, 2, \ldots, 8$. Далее, введем узловые кусочно-линейные базисные функции $\varphi_i(x)$, которые линейны на каждом подынтервале. Они равны единицам в i-х узлах и нулям в остальных узлах (рисунок 6.12).

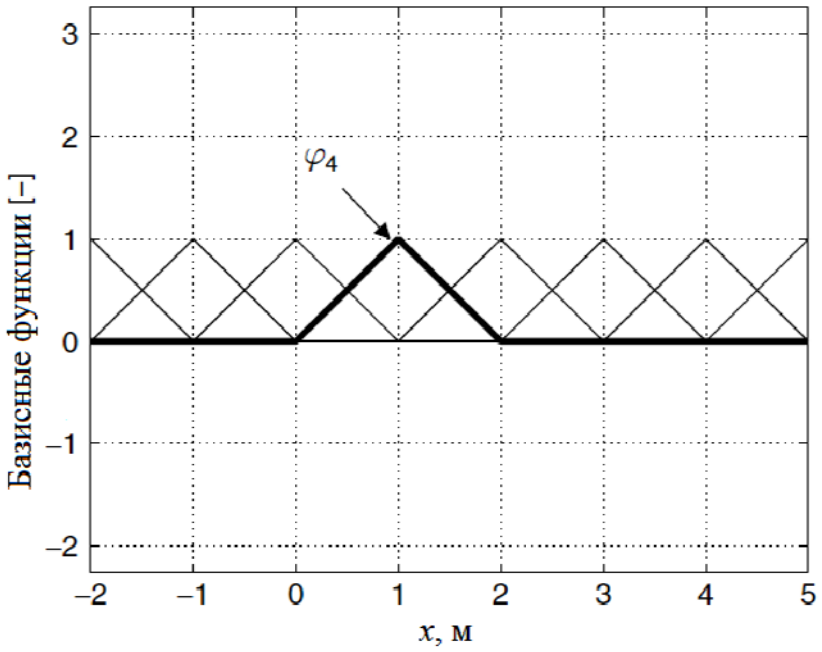


Рисунок 6.12 – Одномерные кусочно-линейные базисные функции

Будем искать приближенное решение с помощью разложения по базисным функциям:

$$f(x) = \sum_{i=1}^{8} f_i \varphi_i(x). \tag{6.75}$$

Заметим, что $f(x_i) = f_i$. Так как $f(a) = f_a$ и $f(b) = f_b$ известны, то положим $f_1 = f_a$ и $f_8 = f_b$.

Следуя методу Галёркина, выберем тестовые функции $w_i(x) = \varphi_i(x)$, где $i = 2, 3, \ldots, 7$ (конечные точки исключены, поскольку соответствующие значения функции известны). Умножим невязку уравнения (6.72) на тестовые функции $w_i(x)$ и результат проинтегрируем от $x = a$ до $x = b$. Будем интегрировать по частям, чтобы освободиться от одной из производных от f по w_i. В результате получим слабую форму (weak form) записи исходной задачи, которая представляет собой взвешенное значение невязки:

$$\int_a^b \left(\alpha w_i' f' + \beta w_i f - \omega_i s\right) ds = 0. \qquad (6.76)$$

Тогда слагаемое $\alpha w_i' f'$ обнуляется, поскольку $w_i(a) = w_i(b) = 0$.

Подставив выражение (6.75) в (6.5) и выбрав $w_2(x) = \varphi_2(x)$, получим уравнение, содержащее 6 неизвестных коэффициентов f_j для внутренних узлов x_j, $j = 2, 3, \ldots, 7$. Аналогично, выбрав $w_3(x) = \varphi_3(x)$, получим второе уравнение с шестью неизвестными и т. д. В результате получим СЛАУ вида **Az** = **b**, где

$$a_{ij} = \int_a^b \left(\alpha \varphi_i \varphi_j' + \beta \varphi_i \varphi_j\right) dx, \qquad (6.77)$$

$$z_j = f_j, \qquad (6.78)$$

$$b_i = \int_a^b \varphi_i s dx. \qquad (6.79)$$

Здесь $i = 2, 3, \ldots, 7$ (по числу уравнений); $j = 1, 2, \ldots, 8$ (по числу коэффициентов). Матрица **A** состоит из 6 строк и 8 столбцов, а векторы **z** и **b** – из 8 и 6 строк соответственно. Значения коэффициентов f_1 и f_8 известны благодаря граничным условиям, поэтому слагаемые, содержащие их, могут быть перенесены в правую часть СЛАУ. Тогда

$$\begin{pmatrix} a_{22} & a_{23} & \ldots & a_{27} \\ a_{32} & a_{33} & \cdots & a_{37} \\ \vdots & \vdots & \ddots & \vdots \\ a_{72} & a_{73} & \cdots & a_{77} \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ \vdots \\ f_7 \end{pmatrix} = \begin{pmatrix} b_2 \\ b_3 \\ \vdots \\ b_7 \end{pmatrix} - \begin{pmatrix} a_{21} f_1 + a_{28} f_8 \\ a_{31} f_1 + a_{38} f_8 \\ \vdots \\ a_{71} f_1 + a_{78} f_8 \end{pmatrix}.$$

Поскольку значения функции в конечных точках известны, можно не использовать соответствующие весовые функции (что и было сделано). При этом матрица СЛАУ является квадратной (число уравнений совпадает с числом неизвестных), разреженной (поскольку использованные базисные функции описывают только связь ближайших элементов) и симметричной (поскольку оператор Гельмгольца является самосопряженным и использован метод Галёркина).

Граничные условия (6.73) и (6.74) задают значение функции $f(x)$ на границах. При других типах граничных условий может быть задана производная функции или линейная комбинация из функции и ее производной. На любой границе, например слева $x = a$, могут быть применены условия стандартных типов

$$f(a) = p \tag{6.80}$$

или

$$f'(a) + \gamma f(a) = q. \tag{6.81}$$

Уравнение (6.80) соответствует граничному условию Дирихле. При $\gamma = 0$ уравнение (6.81) – это граничное условие Неймана, а при $\gamma \neq 0$ – смешанное. В случае использования условий Неймана или смешанного функция $f(a)$ является дополнительной неизвестной. При этом формируется еще одно уравнение с помощью тестовой функции $w_1(x) = \varphi_1(x)$.

Теперь расширим модельную задачу (6.72) до двух измерений. Функция f, как и ранее, является скалярной:

$$-\nabla \cdot (\alpha \nabla f) + \beta f = s \text{ в области } \Omega, \tag{6.82}$$

$$f = p \text{ на границе } L_1, \tag{6.83}$$

$$\overline{\mathbf{n}} \cdot (\alpha \nabla f) + \gamma f = q \text{ на границе } L_2. \tag{6.84}$$

Граница области решения Ω имеет две части – L_1 и L_2, с разными заданными на них граничными условиями.

В качестве примера рассмотрим задачу вычисления сопротивления между левым и нижним краями проводящей пластины (рисунок 6.13). Тогда функция f – электростатический потенциал, α – проводимость, $\beta = 0$, $s = 0$. Потенциал вдоль границы, показанной на рисунке жирной линией, установим равным 10 В, т. е.

используется граничное условие Дирихле $f = 10$. Вдоль границы, показанной на рисунке жирной пунктирной линией, установим потенциал 0 В. Оставшуюся часть границы будем считать изоляцией. На ней используем граничное условие Неймана $\overline{\mathbf{n}} \cdot \nabla f = 0$, что означает равенство нулю величины потока через границу.

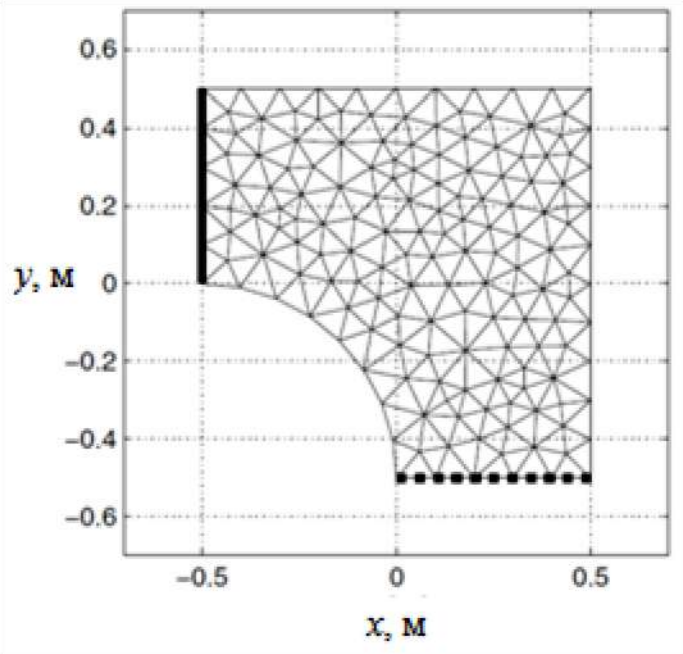


Рисунок 6.13 – Двухмерная проводящая пластина

Умножим уравнение (6.82) на тестовые функции w_i и выполним интегрирование по области Ω:

$$\int_{\Omega} w_i \left[-\nabla \cdot (\alpha \nabla f) + \beta f \right] d\Omega = \int_{\Omega} w_i s d\Omega.$$

Далее интегрируем по частям:

$$\nabla \cdot [w_i (\alpha \nabla f)] = \alpha \nabla w_i \cdot \nabla f + w_i \nabla \cdot (\alpha \nabla f). \qquad (6.85)$$

По теореме Гаусса

$$\int_{\Omega} \nabla \cdot F d\Omega = \int_{L_1 + L_2} \overline{\mathbf{n}} \cdot F dl,$$

где $F = w_i \alpha \nabla f$. В результате получим слабую форму записи для (6.82)

$$\int_{\Omega} \left(\alpha \nabla w_i \cdot \nabla f + \beta w_i f \right) d\Omega - \int_{L_2} w_i (q - \gamma f)\, dl = \int_{\Omega} w_i s d\Omega, \qquad (6.86)$$

где использовано граничное условие (6.84). Интеграл по той части границы, где решение известно (L_1), не учитывается, поскольку тестовые функции на ней обращаются в нуль. Отметим, что дифференциальное уравнение (6.86) включает в себя сведения о граничных условиях.

Узлы сетки пометим целыми числами i. Они расположены в точках r_i, $i = 1, 2, \ldots, N_n$. Конечные элементы представляют собой треугольники. Выберем КЛБФ $\varphi_i(r)$, которые линейны внутри каждого треугольника, т. е. $\varphi_i(r_i) = 1$, $\varphi_i(r_j) = 0$ при $i \neq j$. Существует по одной такой функции, связанной с каждым узлом. В качестве примера на рисунке 6.14 показаны две таких функции.

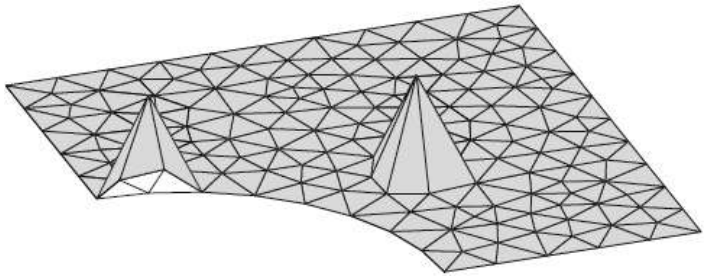


Рисунок 6.14 – Иллюстрация двух узловых базисных функций (одна на границе и одна внутри области решения)

Найдем приближенное решение $f(r)$ с помощью разложения по базисным функциям:

$$f(r) = \sum_{j=1}^{N_n} f_j \varphi_j(r). \tag{6.87}$$

Подставив выражение (6.87) в уравнение слабой формы (6.86) и используя метод Галёркина, т. е. $w_i(r) = \varphi_i(r)$, найдем неизвестные значения функции f в узлах. Это дает СЛАУ вида $\mathbf{Az} = \mathbf{b}$, элементы которой вычисляются как

$$a_{ij} = \int_{\Omega} \left(\alpha \nabla \varphi_i \cdot \nabla \varphi_j + \beta \varphi_i \varphi_j \right) d\Omega + \int_{L_2} \gamma \varphi_i \varphi_j dl, \tag{6.88}$$

$$z_j = f_j, \tag{6.89}$$

$$b_i = \int_{\Omega} \varphi_i s d\Omega + \int_{L_2} \varphi_i q dl. \tag{6.90}$$

Здесь индекс j нужен для обхода всех узлов, а i – только тех, в которых значение f неизвестно (кроме границы L_1, где задано условие Дирихле). Переупорядочим переменные, чтобы известным значениям f соответствовал вектор $\mathbf{z}_e$, а неизвестным – $\mathbf{z}_n$. Аналогично разбивается матрица **A**. В результате СЛАУ преобразуется к виду

$$\begin{pmatrix}\mathbf{A}_e & \mathbf{A}_n\end{pmatrix}\begin{pmatrix}\mathbf{z}_e \\ \mathbf{z}_n\end{pmatrix} = \mathbf{A}_e\mathbf{z}_e + \mathbf{A}_n\mathbf{z}_n = \mathbf{b},$$

где значения $\mathbf{A}_n$ и $\mathbf{b} - \mathbf{A}_e\mathbf{z}_e$ полностью известны. Поэтому СЛАУ можно переписать в виде

$$\mathbf{A}_n\mathbf{z}_n = \mathbf{b} - \mathbf{A}_e\mathbf{z}_e.$$

Процедура нахождения неизвестных подробно описана в подразделе 6.3. В итоге процесс решения сведен к работе с матрицами и векторами, что более предпочтительно для двухмерных и трехмерных задач.

Теперь вернемся к примеру по вычислению сопротивления металлической пластины. Толщину пластины обозначим h. Результирующее распределение электростатического потенциала показано на рисунке 6.15. Зная это распределение, сопротивление пластины можно вычислить двумя способами.

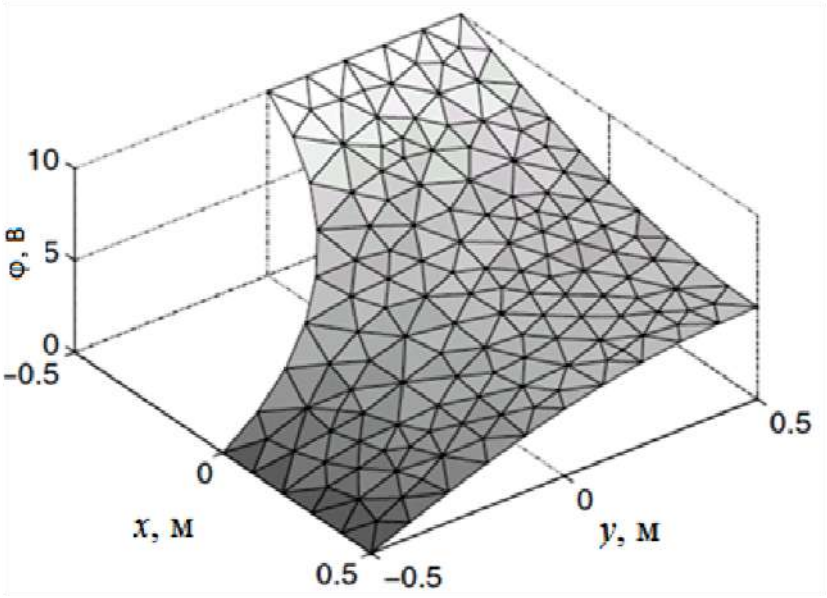


Рисунок 6.15 – Распределение потенциала
на проводящей пластине

Первый способ – интегрирование нормальной составляющей плотности тока $\mathbf{J} = -\sigma\nabla\Phi$ по поперечному сечению пластины для получения величины полного тока, протекающего через пластину:

$$I = \int_{z=0}^{h} \int_{x=0}^{0,5} \sigma \left. \frac{\partial \Phi}{\partial y} \right|_{y=-0.5} dx. \qquad (6.91)$$

После этого сопротивление вычисляется с помощью закона Ома $R = U/I$, где $U = \Delta\Phi = 10$ В.

Второй способ – вычисление общей рассеиваемой пластиной мощности (аналогично вычислению емкости)

$$P = \int_V \mathbf{J} \cdot \mathbf{E} dV = \int_V \sigma |\nabla\Phi|^2 dV = h\mathbf{z}^t \mathbf{A}\mathbf{z} = h\mathbf{z}^t \mathbf{b}, \qquad (6.92)$$

а затем сопротивления $R = U^2 / P$.

6.5 Особенности построения сетки

Этап подготовки данных для решения задач электростатики является одной из основных трудностей, возникающих при конечно-элементном анализе. Поэтому вычислительно эффективные программы должны иметь универсальные средства генерации сетки. Это не только сокращает время, затрачиваемое на подготовку данных, но и устраняет возможные ошибки, возникающие при ручной обработке. Комбинирование автоматической генерации сетки с компьютерной графикой особенно важно, так как результирующую сетку можно оценить визуально. Поскольку электромагнитные задачи часто основаны на простой прямоугольной геометрии области решения, то рассмотрим особенности создания сетки для прямоугольных областей.

Пусть прямоугольная область решения имеет размер $a \times b$ (рисунок 6.16). Задачей является разделение области на прямоугольные элементы, каждый из которых затем делится на два треугольных элемента. Пусть n_x и n_y – число делений по осям x и y соответственно. Тогда общее число элементов и узлов определяется как

$$n_e = 2n_x n_y, \; n_d = (n_x + 1)(n_y + 1). \qquad (6.93)$$

Для хранения глобальных координат (x, y) каждого узла требуется массив, содержащий значения Δx_i, $i = 1, 2, \ldots, n_x$ и Δy_j, $j = 1, 2, \ldots, n_y$, которые соответствуют расстояниям между узлами по осям x и y. Если выбрать порядок нумерации узлов слева направо и снизу вверх, то первый узел является началом координат (0, 0), следующий узел – $x + \Delta x_1$, $y = 0$, затем $x + \Delta x_2$, $y = 0$ и т. д., пока все Δx_i не будут пройдены. Переходя на вторую горизонтальную строку, необходимо повторить процесс, т. е. сначала $x = 0$ и $y + \Delta y_1$, затем инкрементировать x до тех пор, пока все Δx_i не будут пройдены. Процесс продолжается до тех пор, пока не будут получены координаты последнего узла $(n_x + 1)(n_y + 1)$.

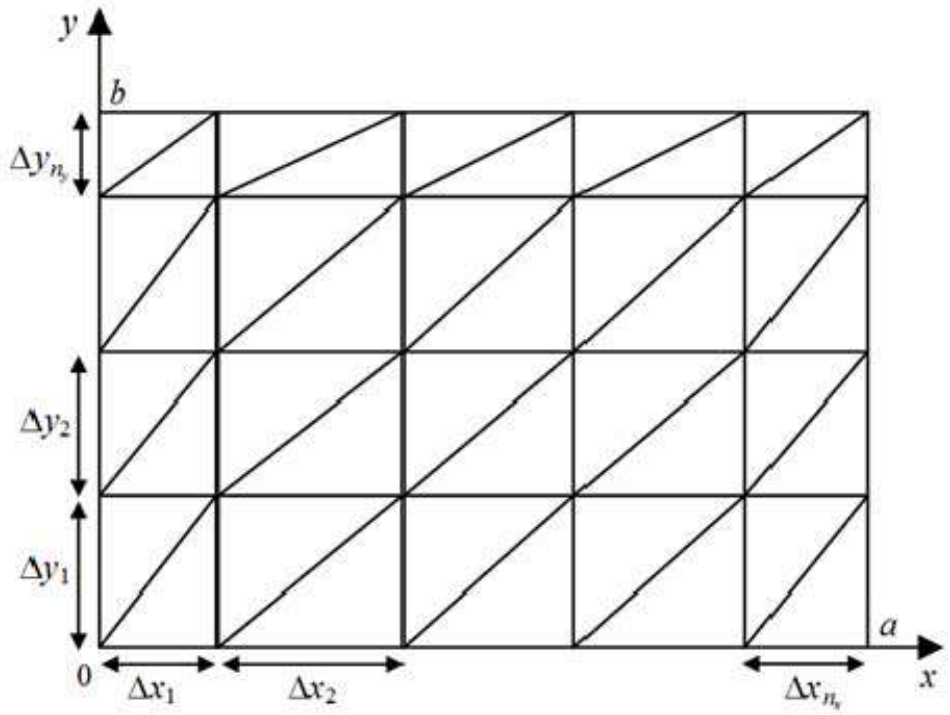


Рисунок 6.16 – Дискретизация прямоугольной области неравномерной сеткой

Представленная процедура позволяет генерировать равномерные и неравномерные сетки. Так, сетка считается равномерной, если все Δx_i равны между собой, а все Δy_j – между собой. В противном случае сетка считается неравномерной. Использование неравномерной сетки предпочтительней, если заранее известно, что интересующий параметр быстро меняется в некоторых частях об-

ласти решения. Это позволяет концентрировать малые элементы в областях, где данный параметр изменяется быстро, поскольку именно такие области часто представляют наибольший интерес для решения. При отсутствии указанных данных может быть использована равномерная сетка, для которой

$$\Delta x_1 = \Delta x_2 = \ldots = h_x,\ \Delta y_1 = \Delta y_2 = \ldots = h_y, \qquad (6.94)$$

где $h_x = a/n_x$; $h_x = b/n_y$.

Предположим, что на всей границе значение потенциала известно, тогда число граничных узлов оценивается как

$$n_p = 2(n_x + n_y). \qquad (6.95)$$

Простым способом формирования списка узлов является их перечисление последовательно на нижней, правой, верхней и левой (против часовой стрелки) границах прямоугольной области решения.

6.6 Математическая модель вычисления емкостной матрицы многопроводной линии передачи

Поскольку для вычисления погонных параметров линии передачи требуется решение уравнения Лапласа, то при вычислении емкостной матрицы линии передачи справедлива описанная в подразделе 6.2 последовательность действий. Так, область решения дискретизируется на трехузловые треугольники. Искомыми величинами являются потенциалы в свободных узлах. Данные узлы используются для вычисления полной энергии структуры и затем ее погонной емкости. Для одиночной линии емкость определяется по формуле

$$C = 2\frac{W_e}{\Phi^2}.$$

Для МПЛП полная энергия вычисляется столько раз, сколько имеется проводников N_{COND} в линии. В результате формируется емкостная матрица. Ее диагональные элементы вычисляются как

$$c_{ii} = 2W_{ii},\ \ i = 1, 2, \ldots, N_{\mathrm{COND}}, \qquad (6.96)$$

где W_{ii} – энергия, вычисленная при установлении потенциала 1 В на i-м проводнике и 0 В на остальных. Внедиагональные элементы емкостной матрицы вычисляются как

$$c_{ij} = 2W_{ii} - 0.5(c_{ii} + c_{jj}), \tag{6.97}$$

где W_{ii} – энергия, вычисленная при задании потенциала 1 В на i-м и j-м проводниках.

Контрольные вопросы и задания

1. Назовите этапы решения задач при использовании МКЭ.
2. В чем различие между комплекс-элементом и симплекс-элементом?
3. К матрице СЛАУ какого вида приводит использование МКЭ?
4. Опишите особенности вычисления емкостной матрицы МПЛП методом конечных элементов.
5. Для чего применяется ансамблирование в МКЭ?
6. Назовите особенности нумерации узлов конечно-элементной сетки.
7. Разработать программу на языке Octave для решения задачи из примера 6.2.
8. Разработать программу на языке Octave для вычисления методом конечных элементов емкости коаксиальной структуры, изображенной на рисунке 3.16, при $a = b = 1$ см, $c = d = 2$ см.

ЗАКЛЮЧЕНИЕ

Вследствие массового распространения различных технических средств в различных областях человеческой деятельности выполнение требований ЭМС становится все более сложным, а число аспектов, принимаемых во внимание, стремительно увеличивается. Поэтому проблема обеспечения ЭМС тесно связана с тем, что составляет обширную область радиотехники, электроники и электротехники. Для минимизации как финансовых затрат, так и затрат времени на проектирование технических средств с учетом требований ЭМС целесообразно использовать математическое моделирование, в основе которого лежат численные методы.

В данном пособии показана актуальность применения математического моделирования при решении задач ЭМС в части электростатики. Обсуждаются общие вопросы, связанные с интегральными и дифференциальными уравнениями, особенности использования численных методов, а также способы повышения точности вычислений и экономии машинных ресурсов. Приведены примеры решения тестовых задач. Помимо этого, подробно рассмотрены математические модели для анализа линий передачи (без потерь), широко используемые на практике при решении задач ЭМС. В первую очередь пособие предназначено для будущих специалистов в области радиоэлектроники и информационных технологий.

Автор не ставил перед собой цель всесторонне осветить все численные методы, применяемые при решении задач электростатики, а заострил внимание только на наиболее широко используемых: методах конечных разностей, моментов и конечных элементов. Приведенные примеры и программы на языке GNU Octave, а также задания нацелены на изучение особенностей этих методов. Тем не менее, остались нерассмотренными представляющие практический интерес методы матрицы линий передачи, эквивалентной схемы из частичных элементов, метод прямых, Монте-Карло и др. Однако, по мнению автора, математическая основа численных методов, рассмотренная в пособии, позволит заинтересованному читателю самостоятельно изучить особенности этих

методов. Следует заметить, что до сих пор не существует универсального численного метода, поэтому разработка новых и, по всей видимости, гибридных методов станет одним из направлений дальнейших исследований.

В основе упомянутых численных методов лежит замена непрерывных функций их дискретными аналогами, что часто сводит задачу к решению линейных алгебраических систем, поэтому значительная часть пособия посвящена рассмотрению методов их решения. Эта часть вычислительной линейной алгебры, несмотря на давнюю историю, остается бурно развивающейся и заслуживает пристального рассмотрения в виде отдельного и объемного пособия. Однако автор предпринял попытку кратко, но емко осветить все ее аспекты, чтобы читателю было легче в дальнейшем изучении этой удивительной и увлекательной проблемы – решение СЛАУ, которой он сам увлекся, еще обучаясь в университете. Необходимо отметить, что не существует универсального метода решения СЛАУ, обеспечивающего требуемую точность при минимальных затратах времени и машинной памяти. Поэтому поиски продолжаются. Так, одним из передовых направлений является аппроксимация матрицы СЛАУ матрицами меньшего ранга для сокращения объема требуемой машинной памяти, например с помощью адаптивной перекрестной аппроксимации.

Таким образом, актуальна разработка новых численных методов и читателю пособия найдется место для приложения своих усилий в этом направлении исследований.

Литература

1. Уилльямс Т. ЭМС для разработчиков / Т. Уилльямс. – М.: Технологии, 2003. – 540 с.
2. Неганов В. А. Электродинамические методы проектирования устройств СВЧ и антенн: учеб. пособие для вузов / В. А. Неганов, Е. И. Нефедов, Г. П. Яровой ; под ред. В. А. Неганова. – М.: Радио и связь, 2002. – 415 с.
3. Банков С. Е. История САПР СВЧ (1950–2010) / С. Е. Банков, А. А. Курушин. – LAP LAMBERT Academic Publishing, 2016. – 100 с.
4. Григорьев А. Д. Методы вычислительной электродинамики / А. Д. Григорьев. – М.: Физматлит, 2013. – 430 с.
5. Altair FEKO [Электронный ресурс]. – Режим доступа: www.altair.com.
6. Нефедов Е. И. Полосковые линии передачи / Е. И. Нефедов, А. Т. Фиалковский. – М.: Наука, 1980. – 312 с.
7. Garg R. Analytical and computational methods in electromagnetic / R. Garg. – Norood: Artech House, 2008. – 528 p.
8. Paul C. R. Transmission lines in digital systems for EMC practitioners / C. R. Paul. – Hoboken, New Jersey: John Wiley & Sons, 2012. – 270 p.
9. Краснов М. И. Интегральные уравнения. Задачи и примеры с подробными решениями / М. И. Краснов, А. И. Киселев, Г. И. Макаренко. –3-е изд., испр. – М.: Едиториал УРСС, 2003. – 192 с.
10. Теоретические основы электротехники: учеб. для вузов. В 3 т. Т. 3 / К. С. Демирчян [и др.]. – 4-е изд. – СПб.: Питер, 2003. – 364 с.
11. Jackson J. D. Classical electrodynamics / J. D. Jackson. – NY: John Wiley & Sons, 1962. – 641 p.
12. Ramo S. Fields and waves in communication electronics / S. Ramo, J. R. Whinnery, T. van Duzer. – 3nd ed. – Hoboken, New Jersey: John Wiley & Sons, 1994. – 844 p.

13. Вольман В. И. Справочник по расчету и конструированию СВЧ полосковых устройств / В. И. Вольман. – М.: Радио и связь, 1982. – 328 с.

14. Ховратович В. С. Параметры многопроводных передающих линий / В. С. Ховратович // Радиотехника и электроника. – 1975. – № 3. – С. 469–473.

15. Djordjevic A. R. Time-domain response of multiconductor transmission lines / A. R. Djordjevic, T. K. Sarkar, R. F. Harrington // Proceedings of the IEEE. – 1987. – Vol. 75, no 6. – P. 743–764.

16. Matthaei G. L. Approximate calculation of the high-frequency resistance matrix for multiple coupled lines / G. L. Matthaei, G. C. Chinn // IEEE Microwave Symposium Digest. – 1992. – P. 1353–1354.

17. Куксенко С. П. Итерационные методы решения системы линейных алгебраических уравнений с плотной матрицей / С. П. Куксенко, Т. Р. Газизов. – Томск: Томский государственный университет, 2007. – 208 с.

18. Фаддеев Д. К. Вычислительные методы линейной алгебры / Д. К. Фаддеев, В. Н. Фаддеева. – М.: Физматгиз, 1963. – 734 с.

19. Olyslager F. Numerical and experimental study of the shielding effectiveness of a metallic enclosure / F. Olyslager, E. Lermans, D. de Zutter // IEEE Transactions on electromagnetic compatibility. – 1999. – Vol. 41, no 3. – P. 202–213.

20. Голуб Д. Матричные вычисления / Д. Голуб, Ч. Ван Лоун. – М.: Мир, 1999. – 548 с.

21. Канахер Д. Численные методы и программное обеспечение / Д. Канахер, К. Моулер , С. Нэш. – М.: Мир, 1999. – 576 с.

22. Куксенко С. П. Методы оптимального проектирования линейных антенн и полосковых структур с учетом электромагнитной совместимости : дис. … д-ра техн. наук / С. П. Куксенко. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2019. – 435 с.

23. Evans D. J. The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices /

D. J. Evans // Journal of the institute of mathematics and its applications. – 1968. – Vol. 4. – P. 295–314.

24. Preconditioners for adaptive integral method implementation / W.-B. Ewe [et al.] // IEEE Transactions on antennas and propagation. – 2005. – Vol. 53, no 7. – P. 2346–2350.

25. Alleon G. Sparse approximate inverse preconditioning for dense linear systems arising in computation of electromagnetic / G. Alleon, M. Benzi, L. Giraud // Numerical algorithms. – 1997. – Vol. 16. – P. 1–15.

26. Solution of dense systems of linear equations arising from integral equation formulations / K. Forsman [et al.] // IEEE Transactions antennas and propagation. – 1995. – Vol. 37, no 6. – P. 96–100.

27. On short recurrence Krylov type methods for linear systems with many right-hand sides / S. Rashedi [et al.] // Journal of computational and applied mathematics. – 2016. – Vol. 300. – P. 18–29.

28. Knoll D. A. Newton-Krylov methods for low-Mach-number compresible combustion / D. A. Knoll, P. R. McHugh, D. E. Keyes // AIAA Journal. – 1996. – Vol. 34, no 5. – P. 961–967.

29. Tebbens J. D. Efficient preconditioning of sequences of nonsymmetric linear systems / J. D. Tebbens, M. Tuma // SIAM Journal on scientific computing. – 2007. – Vol. 29, no 5. – P. 1918–1941.

30. Jolivet P. Block iterative methods and recycling for improved scalability of linear solvers / P. Jolivet, P. H. Tournier // Proceedings of the International conference for high performance computing, networking, storage and analysis. – 2016. – P. 1–15.

31. Multipreconditioned GMRES for shifted systems / T. Bakhos [et al.] // SIAM Journal on scientific computing. – 2017. – Vol. 39, no 5. – P. 222–247.

32. Sadiku M. N. O. Numerical techniques in electromagnetic / M. N. O. Sadiku. – 3-rd edition. – CRC Press LLC, 2009. – 710 p.

33. Пантелеев А. В. Вариационное исчисление в примерах и задачах: учеб. пособие / А. В. Пантелеев. – М.: Изд-во МАИ, 2000. – 228 с.

34. Гельфанд И. М. Вариационное исчисление / И. М. Гельфанд, С. В. Фомин. – М.: Гос. изд-во физ.-мат. лит., 1961. – 227 с.

35. Михлин С. Г. Вариационные методы в математической физике / С. Г. Михлин. – М.: Наука, 1970. – 512 с.

36. Харрингтон Р. Ф. Применение матричных методов к задачам теории поля / Р. Ф. Харрингтон // Труды института инженеров по электронике и радиотехнике. – 1967. – № 2. – С. 5–19.

37. Harrington R. F. Field computation by moment methods / R. F. Harrington. – USA, NY: Macmillan, 1968. – 240 p.

38. Crandall S. H. Engineering analysis / S. H. Crandall. – New York: McGraw-Hill, 1956. – 151 p.

39. Davidson D. B. Computational electromagnetics for RF and microwave engineering / D. B. Davidson. – Cambridge: University Press, 2011. – 505 p.

40. Канторович А. В. Функциональный анализ в нормированных пространствах / А. В. Канторович, Г. П. Акилов. – М.: Физматлит, 1959. – 684 с.

41. Kryloff N. M. Les méthodes de résolution approchee des problèmes de la physique methematique / N. M. Kryloff . – Paris: Gauthier-Villars, 1931. – 71 p.

42. Кравчук М. Ф. О методе Крылова в теории приближенного интегрирования дифференциальных уравнений / М. Ф. Кравчук // Труды физ.-мат. отдел. ВУАН. – 1926. – Vol. 5, no 2. – P. 12–33. (на украинском).

43. Gibson W. C. The method of moments in electromagnetic / W. C. Gibson. – Boca Raton: Chapman &Hall/CRC, 2008. – 272 p.

44. Makarov S. N. Antenna and EM modeling with MATLAB / S. N. Makarov. – New York: John Wiley & Sons, 2002. – 288 p.

45. Газизов Т. Р. Уменьшение искажений электрических сигналов в межсоединениях / Т. Р. Газизов ; под ред. Н. Д. Малютина. – Томск: НТЛ, 2003. – 212 с.

46. Привалов И. И. Аналитическая геометрия / И. И. Привалов. – М.: Наука, 1966. – 272 с.

Приложение А
(справочное)
Программирование в GNU Octave

GNU Octave – это свободно распространяемый язык программирования высокого уровня, ориентированный на проведение численных расчетов и обладающий богатым инструментарием для решения различных задач.

А.1 Основы синтаксиса

После запуска Octave пользователю доступно окно интерпретатора (рисунок А.1), которое содержит:

1) главное меню, предназначенное для получения доступа ко всем возможным командам и интерфейсам;

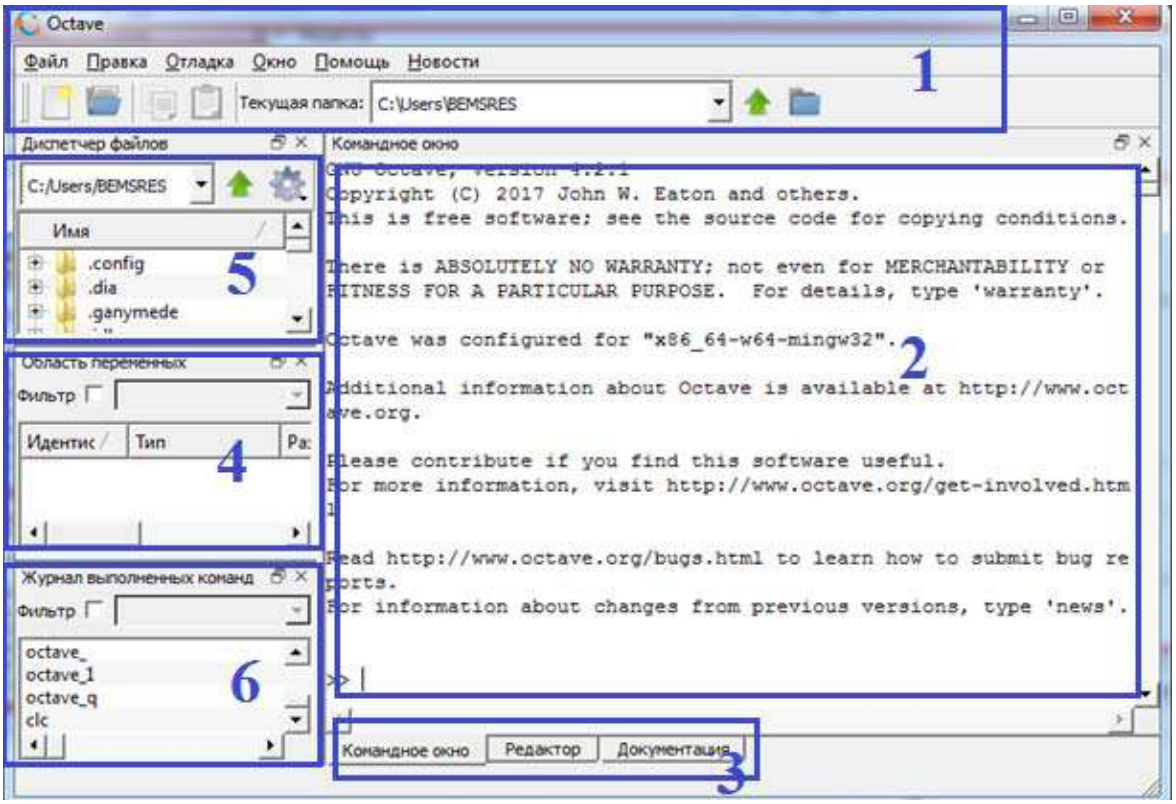


Рисунок А.1 – Графическое окно Octave

2) окно ввода и вывода информации (командное окно/ редактор) для внесения необходимых изменений в программы, созданные или импортированные пользователем. Здесь под программой

понимается текстовый файл, содержащий команды для среды Octave с расширением .m;

3) командное окно/редактор, где пользователь может вводить как отдельные команды языка Octave, так и группы команд. Группы команд удобно объединять в программы (окно Редактор). Содержащиеся в файлах команды последовательно передаются на исполнение, а результат выводится во вкладке *Командное окно*;

4) область переменных, предназначенную для контроля значений и типов данных переменных;

5) файловый менеджер;

6) журнал выполненных команд.

Возможны два варианта решения любой задачи в Octave:

– терминальный режим через вкладку *Командное окно*. В этом режиме в окно интерпретатора последовательно вводятся отдельные команды;

– программный режим через вкладку *Редактор*. В этом режиме создается текстовый файл с расширением .m, в котором хранятся последовательно выполняемые команды Octave. Затем программа запускается на выполнение.

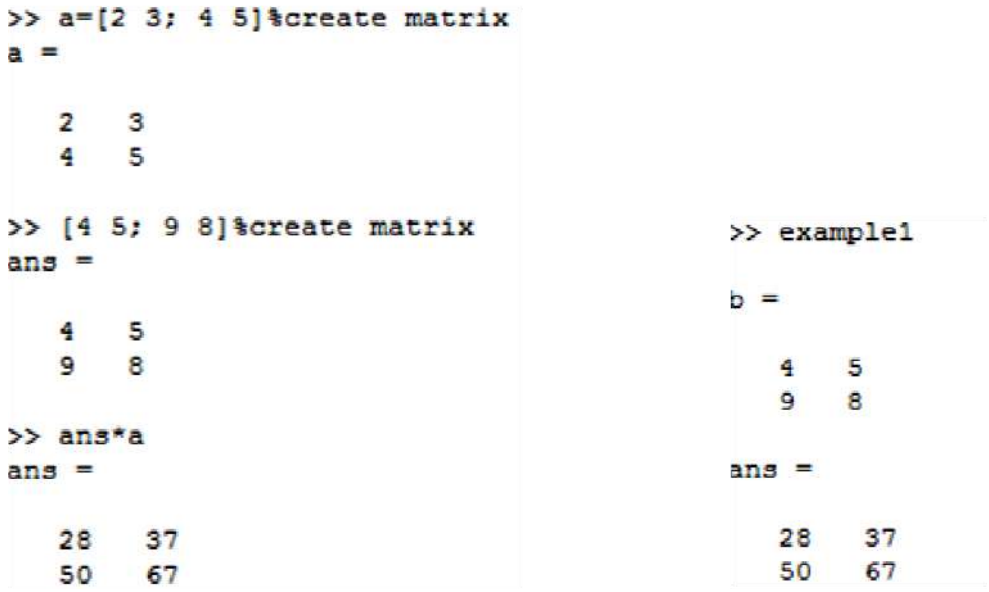
На рисунке А.2, *а* показан пример создания и перемножения двух матриц в терминальном режиме. Для подтверждения ввода и запуска вычислений используется нажатие на клавишу *Enter*. В переменной *ans* хранится результат последней операции. Причем полученное значение можно использовать в последующих вычислениях.

При работе в программном режиме нужно перейти во вкладку *Редактор* и последовательно ввести команды (здесь и далее для облегчения восприятия примеры команд будут показаны курсивом):

```
a=[2 3; 4 5];
b=[4 5; 9 8]
b*a
```

После этого следует сохранить файл, например, с именем example1.m в нужной директории рабочей станции и нажать F5. Перейдя назад во вкладку *Командное окно*, можно увидеть результат вычисления (рисунок А.2, *б*). Если строка заканчивается символом «;», результаты на экран не выводятся. Текст в строке,

следующий за символом «%», является комментарием и интерпретатором не обрабатывается.



а *б*

Рисунок А.2 – Перемножение двух матриц:
а – терминальный режим; *б* – программный режим

Рассмотрим основные моменты синтаксиса языка Octave, используя терминальный режим. Основной рабочей структурой данных в Octave является матрица. Поэтому необходимо помнить, что любому числу соответствуют матрицы размером 1×1. Продемонстрируем это с помощью операции присваивания «=», которая передает значение выражения в левой части равенства переменной, стоящей справа. Так, если написать в командном окне строку

>>a=5

то переменной *a* будет присвоено значение 5. Чтобы узнать размер этой переменной, воспользуемся командой

>> size(a)
ans =
1 1

Видно, что размер матрицы *a* в данном случае 1×1.

Имя переменной не должно совпадать с именами встроенных системных процедур, функций и переменных. Octave различает прописные и строчные буквы в именах переменных, т. е. ABC, abc, Abc, aBc и т. д. – это имена разных переменных.

Кроме переменной *ans*, в Octave существуют и другие системные переменные, которые можно использовать в математических выражениях:

– i, j – мнимая единица;

– pi – число π;

– *e* –экспонента, 2,71828183;

– realmin – наименьшее число с плавающей точкой (2.2251e-308);

– realmax – наибольшее число с плавающей точкой (1.7977e+308);

– inf – машинный символ бесконечности;

– NaN – неопределенный результат.

Как было показано выше, возможно выполнение операции присвоения сразу ко всем элементам матрицы, например

```
>> B=[1,2,3]
B =
   1 2 3
```

Эта команда генерирует матрицу размерности 1×3 (вектор-строка). При введении данных через «;» будет сформирован вектор-столбец

```
>> C=[4;5;6]
C =
   4
   5
   6
```

Аналогичным образом генерируются матрицы любой размерности.

Если значения некоторой величины меняются с равным шагом, то оператор присваивания может использоваться совместно с оператором «:», определяющим пределы изменения величины. Значение шага задается пользователем. По умолчанию шаг равен единице. Например:

```
>> E=1:4:17 %начальное значение : шаг : конечное значение
E =
   1 5 9 13 17.
```

Оператор «:» имеет еще одно применение – он может определять совокупность строк или столбцов в матрице:

```
>> D=zeros(4,4)
```

```
D =
     0   0   0   0
     0   0   0   0
     0   0   0   0
     0   0   0   0
>> D(1,:)=1
D =
     1   1   1   1
     0   0   0   0
     0   0   0   0
     0   0   0   0
```

Здесь с помощью оператора «:» произошло присвоение значения 1 всем элементам первой строки матрицы *D*.

Для очистки командного окна можно воспользоваться командой *clc*, а для удаления из памяти всех переменных или конкретной переменной – командами *clear* и *clear имя переменной* соответственно. Простейшие арифметические операции в Octave выполняют с помощью следующих операторов:

– сложение «+»;

– вычитание «–»;

– умножение «*»;

– деление слева направо «/»;

– деление справа налево «\»;

– возведение в степень «^».

Если вычисляемое выражение слишком длинное, то перед нажатием клавиши ENTER следует ввести троеточие. Это будет означать продолжение текущей (командной) строки.

```
>> 1+2+...
4+5
ans =  12
```

Допускается переназначение не только значений переменных, но и их типов. Например, вещественной переменной можно присвоить символьное значение, при этом автоматически произойдет смена ее типа. Но интерпретатор команд отслеживает только согласование размерности переменных в выражении. Так, умножение символьной строки на матрицу или сложение двух матриц разной размерности вызовет ошибку.

```
>> A=[3 4; 7 5];
>> A='line';
```

```
>> B=[1 2; 3 4];
>> A*B
error: operator *: nonconformant arguments (op1 is 1x4, op2 is 2x2)
```

Для удобства пользователя большинство арифметических операций можно выполнить поэлементно. Для этого используется знак «.» в дополнение к требуемому оператору.

```
>> A=[1,2;3,4];
>> B=A*A
B =
    7 10
    15 22
>> C=A.*A
C =
    1 4
    9 16
```

В первом случае матрица **A** была умножена сама на себя, а во втором произошло поэлементное возведение в квадрат каждого ее элемента.

Числовые значения результатов могут быть представлены с плавающей или с фиксированной точкой. Для представления чисел с плавающей точкой используется экспоненциальная форма записи $mE \pm p$, где m – мантисса (целое или дробное число с десятичной точкой), p – порядок (целое число).

```
>> 5.49e-2
ans =  0.054900
>> 5.49E-2
ans =  0.054900
>> 5.49.*1E-2
ans =  0.054900
>> 5.49.*10^-2
ans =  0.054900
>> 5.49*10^-2
ans =  0.054900
```

Рассмотрим ввод числа:

```
>> 0.987654321
ans =  0.98765
```

Видно, что количество знаков в дробной части числа при вводе больше, чем при выводе. Это связано с тем, что вывод результата вычислений определяется предварительно заданным форматом представления чисел. В Octave предусмотрено несколько форматов чисел. Приведем некоторые из них:

– Short – краткая запись, применяется по умолчанию;

– Long – длинная запись;

– Short E (Short e) – краткая запись в формате с плавающей точкой;

– Long E (Long e) – длинная запись в формате с плавающей точкой.

Продемонстрируем их использование на примере числа π.

```
>> format short
>> pi
ans =  3.1416
>> format long
>> pi
ans =  3.14159265358979
>> format short E
>> pi
ans =   3.1416E+000
>> format long E
>> pi
ans =   3.14159265358979E+000
```

Имеются также тригонометрические функции, экспонента и натуральный логарифм.

```
>> x=pi/2
>> x =  1.5708
>> sin(x)
ans =  1
>> tan(x)
ans =   1.6331e+016
>> exp(x)
ans =  4.8105
>> log(x)
ans =  0.45158
```

Кроме того, имеется ряд других полезных функций:

– fix(x) – округление числа *x* до ближайшего целого в сторону нуля;

– floor(x) – округление числа *x* до ближайшего целого в сторону отрицательной бесконечности;

– ceil(x) – округление числа *x* до ближайшего целого в сторону положительной бесконечности;

– round(x) – округление числа *x* до ближайшего целого;

– rem(x, y) – вычисление остатка от деления *x* на *y*;

– sign(x) – выдает 0, если x=0, –1 при $x < 0$ и 1 при $x > 0$;

– sqrt(x) – корень квадратный из числа x;

– abs(x) – модуль числа x;

– log10(x) – десятичный логарифм от числа x;

– log2(x) – логарифм по основанию два от числа x;

– pow2(x) – возведение двойки в степень x;

– gcd(x, y) – наибольший общий делитель чисел x и y;

– lcm(x, y) – наименьшее общее кратное чисел x и y;

– rats(x) – представление числа x в виде рациональной дроби.

Для обозначения мнимой единицы используются i и/или j. Ввод комплексного числа производится в формате *действительная часть* + *i*мнимая часть*. К комплексным числам применимы элементарные арифметические операции +, –, *, \, /, ^, а также специальные функции:

– real(z) – выдает действительную часть комплексного аргумента z;

– imag(z) – выдает мнимую часть комплексного аргумента z;

– angle(z) – вычисляет значение аргумента комплексного числа z в радианах от $-\pi$ до π;

– conj(z) – выдает число комплексно сопряженное z.

Рассмотрим операции отношения, предназначенные для выполнения сравнения двух операндов и определения истинности выражения. Результатом операции отношения является логическое значение (1 или «истина» и 0 или «ложь»). Предусмотрены следующие операции отношения:

– меньше «<»;

– больше «>»;

– равно «==»;

– не равно «~=»;

– меньше или равно «<=»;

– больше или равно «>=».

В Octave существует возможность представления логических выражений в виде логических операторов и логических операций (таблица А.1).

Таблица А.1 – Виды логических выражений

Тип выражения	Выражение	Логический оператор	Логическая операция
Логическое «и»	A and B	and(A, B)	A & B
Логическое «или»	A or B	or(A, B)	A | B
Исключающее «или»	A xor B	xor(A,B)	
Отрицание	not A	not (A)	~ A

Для визуализации входных данных и полученных результатов Octave располагает обширными библиотеками графических построений. Возможно построение как двухмерных, так и трехмерных графиков. Основной функцией для построения двухмерных графиков служит функция plot, у которой несколько вариантов вызова:

– plot(X,Y) – построение зависимости $y(x)$, где значения y и x берутся из матриц **Y** и **X**, как правило, одномерных;

– plot(X1,Y1,..., Xn,Yn) – одновременное построение нескольких функциональных зависимостей;

– plot(X1,Y1,LineSpec1,..., Xn,Yn,LineSpecn) – наиболее полный вариант вызова функции, где LineSpec – это шаблон, с помощью которого определяется цвет линии, ее толщина, вид маркеров и другие параметры (таблица А.2).

```
>> x=0:0.1:2*pi;
>> plot(x,sin(x),'Color','m','LineStyle','-.','LineWidth',4);
```

Таблица А.2 – Параметры функции plot для задания шаблона линии

Параметр	Возможные значения
Color' – цвет линии	'y' – желтый; 'm' – магента (пурпурный); 'c' – циан (зелено-голубой); 'r' – красный; 'g' – зеленый; 'b' – голубой; 'k' – черный; [r g b] – цвет в формате *RGB* (параметры *r*, *g*, *b* лежат в пределах от 0 до 1)
LineStyle' – вид линии	'-' – сплошная; '- -' – двойная сплошная; ':' – пунктирная; '-.' – штрихпунктирная; 'none' – без линии (только маркеры)
'LineWidth' – ширина линии	Положительные значения с шагом 0.5
Marker' – вид маркера	'o' – круговой; '+' – знак «+»; 's' – квадрат; 'p' – пятиугольник; '^' – ориентированный вверх треугольник

Если последовательно использовать еще одну команду plot, то после ее выполнения график новой функции будет нарисован поверх предыдущего. Чтобы это не происходило, надо настроить графическое окно (таблица А.3).

Таблица А.3 – Функции для задания параметров работы графического окна

Функция	Варианты использования
figure – функция запроса графического окна. После применения все изображения будут перенаправляться в запрошенное окно	figure() – вызывает графическое окно, номер которому будет присвоен автоматически (возможен вызов в виде N=figure(), в этом случае переменной *N* будет присвоен номер графического окна) figure(N) – будет создано новое окно с номером *N*, если такого окна не существовало, в противном случаев в него будет перенаправлен графический вывод
hold – функция, включающая и отключающая сохранение в графическом окне предыдущих графиков	hold on – новые графики будут изображаться совместно с предыдущими hold off – новые графики будут изображаться поверх предыдущих, затирая их
grid – функция, включающая и отключающая отображение линий сетки	grid on – включает отображение линий сетки grid off – выключает отображение линий сетки
xlabel, ylabel – функции подписывания осей	xlabel('НАЗВАНИЕ ОСИ X') ylabel('НАЗВАНИЕ ОСИ Y')
legend – функция нанесения на график легенды (сопровождающих и поясняющих надписей)	legend(' СТРОКА 1', ' СТРОКА 2', ...)

Следующий пример демонстрирует построение зависимостей sin(*x*) и cos(*x*) на одном графике.

```
>> x=0:0.1:2*pi;
>> figure(1);
>> hold on;
>> plot(x,sin(x),'Color','m','LineStyle','-.','LineWidth',4)
>> plot(x,cos(x),'Color','g','LineStyle','--','LineWidth',3)
>> legend('sin(x)','cos(x)')
>> grid on;
```

```
>> xlabel('x');
>> ylabel('sin(x)');
```

В Octave есть возможность строить несколько координатных осей в графическом окне и выводить на каждую из них разные графики. Для этого используется функция *subplot(row, col, cur)*. Параметры *row* и *col* определяют требуемое число графиков по вертикали и горизонтали соответственно, а *cur* – номер текущего графика. Повторное обращение к этой функции с теми же значениями *row* и *col* позволяет изменять номер текущего графика и переключаться между графиками.

Для построения графиков поверхностей используется несколько функций. Сначала формируется прямоугольная сетка, содержащая координаты узловых точек, с помощью функции meshgrid():

```
>> [x y]=meshgrid(-2:2,-1:1)
x =
    -2  -1   0   1   2
    -2  -1   0   1   2
    -2  -1   0   1   2
y =
    -1  -1  -1  -1  -1
     0   0   0   0   0
     1   1   1   1   1
```

Далее используется функция mesh() для построения «каркасного» графика. Рассмотрим это на примере функции $z(x, y) = 4x^2 - 2\sin^2 y$.

```
>> z=4*x.^2-2*sin(y).^2
z =
    14.58385   2.58385  -1.41615   2.58385  14.58385
    16.00000   4.00000   0.00000   4.00000  16.00000
    14.58385   2.58385  -1.41615   2.58385  14.58385
>> mesh(x, y, z)
```

Еще одной функцией для построения трехмерных графиков является surf(**x**, **y**, **z**) или surf(**x**, **y**, **z**, *C*), где **x** и **y** – векторы-строки, определяющие значения координат узлов; **z** – матрица с размерностью, равной произведению размерностей матриц **x** и **y**, задающая значения координат узлов по оси *z* для соответствующих пар *x* и *y*. Параметр *C* определяет способ отображения результата (цвет, режим отображения кромок и т. д.).

Перечисленные способы построения как двухмерных, так и трехмерных графиков не являются единственно возможными. О других способах можно узнать из документации Octave.

А.2 Работа с матрицами

Элементы вектора-строки отделяют пробелами или запятыми, а всю конструкцию заключают в квадратные скобки. Вектор-столбец можно задать, если элементы отделять друг от друга точкой с запятой. Для обращения к элементу вектора надо указать его порядковый номер. Нумерация элементов начинается с единицы.

```
>> a=[2 -3 5 6 -1 0 7 -9]
a =
     2 -3 5 6 -1 0 7 -9
>> b=[-1,0,1]
b =
     -1 0 1
>> c=[-pi;-pi/2;0;pi/2;pi]
c =
     -3.14159
     -1.57080
     0.00000
     1.57080
     3.14159
>> b(3)
ans = 1
```

Ввод элементов матрицы, как и в случае с векторами, осуществляется в квадратных скобках. При этом элементы строки отделяются друг от друга пробелом или запятой, а строки разделяются между собой точкой с запятой. Обратиться к элементу матрицы можно, указав номера строки и столбца, на пересечении которых расположен интересующий элемент.

```
>> Matr=[0 1 2 3;4 5 6 7]
Matr =
     0 1 2 3
     4 5 6 7
>> Matr(2,3)
ans = 6
>> Matr(1,1)=pi; Matr(2,4)=-pi;
>> Matr
Matr =
```

```
    3.1416 1.0000 2.0000 3.0000
    4.0000 5.0000 6.0000 -3.1416
```

Матрицы и векторы можно формировать, составляя их из ранее заданных матриц и векторов, используя конкатенацию.

```
>> a=[-3 0 2];b=[3 2 -1];c=[5 -2 0];
>> M=[a b c]
M = -3 0 2 3 2 -1 5 -2 0
>> N=[a;b;c]
N =
    -3 0 2
    3 2 -1
    5 -2 0
>> Matrica=[N N N]
Matrica =
    -3 0 2 -3 0 2 -3 0 2
    3 2 -1 3 2 -1 3 2 -1
    5 -2 0 5 -2 0 5 -2 0
```

Важную роль при работе с матрицами играют знак двоеточия «:» (задание диапазона индексов) и оператор [] (удаление строк и столбцов из матрицы).

```
>>Tabl=[-1.2 3.4 0.8; 0.9 -0.1 1.1; 7.6 -4.5 5.6; 9.0 1.3 -8.5];
>> Tabl(:,3)
ans =
    0.80000
    1.10000
    5.60000
    -8.50000
>> Tabl(1,:)
ans = -1.20000 3.40000 0.80000
>> Matr=Tabl(2:3,1:2)
Matr =
     0.90000 -0.10000
     7.60000 -4.50000
>> Tabl(3:4,2:3)=Matr
Tabl =
    -1.20000 3.40000 0.80000
    0.90000 -0.10000 1.10000
    7.60000 0.90000 -0.10000
    9.00000 7.60000 -4.50000
>> Tabl(:,2)=[]
Tabl =
    -1.20000 0.80000
    0.90000 1.10000
```

```
    7.60000 -0.10000
    9.00000 -4.50000
>> Vector=Matr(:)
Vector =
    0.90000
    7.60000
    -0.10000
    -4.50000
>> V=Vector(1:3)
V =
    0.90000
    7.60000
    -0.10000
>> V(2)=[]
V =
    0.90000
    -0.10000
```

Рассмотрим действия над векторами, предусмотренные в Octave. Сложение и вычитание (знаки «+» и «–») возможно только для векторов одного типа, то есть суммировать/вычитать можно либо векторы-столбцы, либо векторы-строки одинаковой длины. Знак апострофа «'» применяется для транспонирования. Умножение вектора на число осуществляется с помощью знака «*». Знак деления «/» применяют для того, чтобы разделить элементы вектора на число.

```
>> a=[2 4 6];b=[1 3 5];
>> c=a+b
c =
    3 7 11
>> a'
ans =
    2
    4
    6
>>> z=2*a+a/4
z =
    4.5000 9.0000 13.5000
```

Умножение вектора на вектор выполняется также с помощью знака «*». Эта операция применима только к векторам одинаковой длины, причем один из них должен быть вектором-столбцом, а второй – вектором-строкой.

```
>> a=[2 4 6]; b=[1 3 5];
>> a*b'
ans = 44
>> a'*b
ans =
     2 6 10
     4 12 20
     6 18 30
>> a*b
error: operator *: nonconformant arguments (op1 is 1x3, op2 is 1x3)
```

Кроме перечисленных действий с векторами, возможно их поэлементное преобразование. Например, если к некоторому вектору применить математическую функцию, то в результате получим новый вектор того же размера и структуры, но с преобразованными в соответствии с заданной функцией элементами.

```
>> x=[-pi/2,-pi/3,-pi/4,0,pi/4,pi/3,pi/2]
x =
     -1.5708 -1.0472 -0.7854 0.0000 0.7854 1.0472 1.5708
>> y=sin(2*x)+cos(2*x)
y =
     -1.0000 -1.36603 -1.0000 1.0000 1.0000 0.36603 -1.0000
```

Поэлементное умножение векторов выполняется при помощи оператора «.*». В результате генерируется вектор, каждый элемент которого равен произведению соответствующих элементов исходных векторов. Поэлементное деление осуществляется оператором «./». В результате получается вектор, каждый элемент которого является частным от деления соответствующих элементов первого и второго векторов. Для обратного поэлементного деления используется совокупность знаков «.\». Поэлементное возведение в степень элементов вектора выполняет оператор «.^».

```
>> a=[2 4 6];b=[1 3 5];
>> a.*b
ans = 2 12 30
>> a=[2 4 6];b=[1 3 5];
>> a./b
ans =
     2.0000 1.3333 1.2000
>> a.\b
ans =
     0.50000 0.75000 0.83333
```

Далее рассмотрим действия над матрицами, которые по сути совпадают с операциями над векторами, но есть принципиальные отличия. При сложении «+» и вычитании «–» важно помнить, что матрицы должны быть одной размерности.

При умножении матриц «*» число столбцов первой матрицы должно совпадать с числом строк второй матрицы. При умножении матрицы на число результатом будет матрица, каждый элемент которой помножен на заданное число. Операция транспонирования «'» меняет местами строки и столбцы заданной матрицы.

Возведение матрицы в степень «^» эквивалентно ее умножению на себя указанное число раз. При этом показатель степени может быть как положительным, так и отрицательным. Матрица в степени минус один называется обратной к данной. Возведение в отрицательную и целочисленную степень соответствует умножению обратной матрицы на себя указанное число раз. Для поэлементного преобразования матриц могут применяться операции, описанные для векторов.

```
>> A=[1 2 3; 4 5 6; 7 8 9]; B=[-1 -2 -3; -4 -5 -6; -7 -8 -9];
>> (2*A+1/4*B')^2-A*B^(-1)
ans =
    83.875   93.125  104.375
   219.250  244.062  272.875
   356.625  395.000  457.375
```

Операторы «/» и «\» используются для деления матриц слева направо и справа налево соответственно. Так, деление матриц *B/A* соответствует выражению $\mathbf{BA}^{-1}$ и, как правило, используется при решении матричных уравнений вида $\mathbf{XA} = \mathbf{B}$. Соответственно деление *A\B* эквивалентно $\mathbf{A}^{-1}\mathbf{B}$ и применяется для решения СЛАУ вида $\mathbf{AX} = \mathbf{B}$.

```
>> A=[1 2;1 1];
>> b=[7;6];
>> x=A\b
x =
   5
   1
```

В Octave имеются специальные функции, которые можно разделить на 3 группы: функции для работы с векторами; функции

для работы с матрицами; функции, реализующие алгоритмы решения задач линейной алгебры.

Приведем наиболее часто используемые функции для векторов:

– length(x) – определение длины вектора **x**;

– prod(x) – вычисление произведения элементов вектора **x**;

– sum(x) – вычисление суммы элементов вектора **x**;

– diff(x) – генерация вектора длиной на единицу меньше, чем у вектора **x**, каждый элемент которого есть разность между двумя соседними элементами вектора **x**;

– diag(x, k) – генерация квадратной матрицы с элементами вектора **x** на главной (если *k* не задано) или *k*-й диагонали;

– min(x) – поиск минимального элемента вектора **x**;

– max(x) – поиск максимального элемента вектора **x**;

– mean(x) – вычисление среднего арифметического элементов вектора;

– dot(x1,x2) – вычисление скалярного произведения двух векторов;

– cross(x1,x2) – вычисление векторного произведения векторов;

– sort(x) – сортировка массива **x** по возрастанию (сортировка по убыванию – sort(-x)).

Наиболее часто используемые функции для матриц:

– eye(n , m) – генерация единичной матрицы размерности $n \times m$;

– ones(n , m) – генерация матрицы, состоящей из единиц, размерности $n \times m$;

– zeros(n, m) – генерация матрицы, состоящей из нулей, размерности $n \times m$;

– diag(A, k) – генерация вектора-столбца из элементов, расположенных на главной (если *k* не задано) или *k*-й диагонали матрицы **A**;

– rand(n, m) – генерация матрицы размерности $n \times m$, элементы которой случайные числа, распределенные по равномерному закону;

– randn(n, m) – генерация матрицы размерности $n \times m$, элементы которой случайные числа, распределенные по нормальному закону;

– linspace(a, b, n) – генерация массива из n элементов, равномерно распределенных между значениями a и b;

– repmat(A, n, m) – генерация матрицы, состоящей из $n \times m$ копий матрицы **A**;

– reshape(A, m, n) – генерация матрицы размерности $m \times n$ из матрицы **A** путем последовательной выборки по столбцам;

– cat(n, A, B) – конкатенация матриц **A** и **B** (при $n = 1$ конкатенация по столбцам, а при $n = 2$ – по строкам);

– rot90(A, k) – поворот матрицы **A** на $90k$ градусов, где k – целое число;

– tril(A, k) – генерация нижнетреугольной матрицы из матрицы **A** начиная с главной (если k не задано) или k-й диагонали;

– triu(A , k) – генерация верхнетреугольной матрицы из матрицы **A** начиная с главной (если k не задано) или k-й диагонали;

– size(A) – определение числа строк и столбцов матрицы **A**;

– prod(A, k) – генерация вектора-столбца ($k = 1$) или вектора-строки ($k = 2$), каждый элемент которого является произведением элементов соответствующего столбца или строки матрицы **A**;

– sum(A, k) – генерация вектора-столбца ($k = 1$) или вектора-строки ($k = 2$), каждый элемент которого является суммой элементов соответствующего столбца или строки матрицы **A**;

– diff(A) – генерация матрицы размерности $(n - 1) \times m$, элементы которой являются разностями между элементами соседних строк матрицы **A** размерности $n \times m$;

– min(A) – генерация вектора-строки, элементы которого являются наименьшими элементами из соответствующих столбцов матрицы **A**;

– max(A) – генерация вектора-строки, элементы которого являются наибольшими элементами из соответствующих столбцов матрицы **A**;

– mean(A, k) – генерация вектора-столбца ($k = 1$) или вектора-строки ($k = 2$), каждый элемент которого является средним ариф-

метическим значением элементов соответствующего столбца или строки матрицы **A**;

– sort(A) – генерация из матрицы **A** матрицы той же размерности, каждый столбец которой упорядочен по возрастанию.

Рассмотрим наиболее часто используемые функции, реализующие численные алгоритмы решения задач линейной алгебры:

det(M) – вычисление определителя квадратной матрицы **M**;

trace(M) – вычисление суммы элементов на главной диагонали матрицы **M**;

– norm(A, p) – вычисление нормы матрицы **A**, где p = 1, 2, 'inf', 'fro';

– cond(A, p) – вычисление числа обусловленности матрицы **A** по норме p, где p = 1, 2, 'inf', 'fro';

– rcond(A) – вычисление величины, обратной значению числа обусловленности матрицы **A** относительно первой нормы (если полученная величина близка к единице, то матрица считается хорошо обусловленной, а если к нулю, то плохо обусловленной);

– inv(A) – генерация матрицы, обратной к **A**;

– eig(A) – вычисление собственных значений столбцов матрицы **A**;

– chol(A, str) – генерация верхнетреугольной (если str = 'upper') или нижнетреугольной (str = 'lower') матрицы **L**, полученной разложением по Холецкому для положительно определенной и симметричной матрицы **A**;

– lu(A) – LU-разложение матрицы **A**, результат является объединением матриц **L** и **U**;

– qr(A) – QR–разложение матрицы **A**, результат является объединением матриц **Q** и **R**;

– svd(A) – генерация вектора-столбца, состоящего из сингулярных чисел матрицы **A**.

A.3 Основы программирования

Рассмотренные выше группы команд представляют собой простейшие программы Octave. Если такая программа хранится в файле с расширением .m, то для ее выполнения достаточно в командной строке Octave ввести имя этого файла (без расширения).

Очевидно, что часто требуется разработка более сложных программ, содержащих функции как составные элементы, поэтому возникает необходимость в использования циклов, условий и прочих атрибутов программирования. Далее рассмотрим основные операторы языка Octave и примеры их использования.

Даже при разработке простейших программ возникает необходимость ввода исходных данных и вывода результатов. Если для вывода результатов на экран можно просто не ставить «;» в конце нужной строки, то для ввода данных при разработке программы, реализующей диалоговый режим, следует использовать функцию input ('подсказка'). Если в тексте программы встречается эта функция, то ее выполнение приостанавливается и на экран выводится текст подсказки, а Octave переходит в режим ожидания ввода. После того как пользователь введет с клавиатуры требуемое значение и нажмет клавишу *Enter*, это значение будет присвоено переменной, указанной слева от знака присваивания.

```
>> d=input('Enter value ')
Enter value 5
d =  5
```

В Octave, как и в большинстве других языков программирования, одним из основных операторов, предназначенных для ветвления, является условный оператор *if-else*. Рассмотрим его простую и расширенную версии. Простой условный оператор работает следующим образом. Если некое условие истинно, то выполняется *блок кода 1*, содержащий один или несколько операторов, а если условие ложно, то выполняется *блок кода 2*.

```
if условие
    блок кода 1
else
    блок кода 2
end
```

Расширенная версия условного оператора используется для организации сложных проверок. Например, если *условие 1* истинно, то выполняется *блок кода 1*, иначе проверяется *условие 2*, если оно истинно, то выполняется *блок кода 2* и т. д. Если ни одно из условий по веткам *elseif* не является истинным, то выполняются операторы по ветке *else*.

if условие 1
    блок кода 1
elseif условие 2
    блок кода 2
elseif условие 3
    блок кода 3
…
elseif условие n
    блок кода n
else
    блок кода m
end

Еще одним способом организации сложных ветвлений является оператор *switch*. Так, если значение контролируемого *параметра* равно *значению 1*, то выполняется *блок кода 1*, иначе, если *параметр* равен *значению 2*, то выполняется *блок кода 2*, и т. д. Если значение *параметра* не совпадает ни с одним из значений в группах *case*, то выполняется *блок кода*, следующий за ключевым словом *otherwise*.

swith параметр
case значение 1
    блок кода 1
case значение 2
    блок кода 2
case значение 3
    блок кода 3
…
otherwise
    блок кода m
end

Для организации повторяющихся действий предусмотрены циклы *while* и *for*. Цикл с предусловием *while* работает следующим образом. Если *выражение* (переменная или логическое выражение) истинно, то выполняется *блок кода*, находящийся после слова *while*, иначе цикл игнорируется и управление передается оператору, следующему за телом цикла. Перед каждой итерацией цикла происходит проверка *выражения*.

while выражение
    блок кода
end

Цикл *for* в общем случае является циклом с заранее известным числом итераций. Выполнение цикла начинается с присвоения контролируемому параметру цикла начального значения, после чего следует проверка, не превышает ли оно конечное значение. Если результат утвердительный, то выполняется *блок кода* в теле цикла, иначе цикл прерывается и управление передается следующему за телом цикла оператору. По окончании итерации значение параметра изменяется на значение шага и следует его повторная проверка.

```
for параметр = начальное значение:шаг:конечное значение
    блок кода
end
```

Если значение шага цикла равно 1, то можно использовать сокращенную запись.

```
for параметр = начальное значение:конечное значение
    операторы
end
```

Иногда при выполнении цикла надо прервать итерации, например при выполнении какого-то условия. Поэтому нужны операторы, которые принудительно изменяют порядок выполнения команд за счет передачи управления. Для этого внутри циклов используются операторы *break* и *continue*. Оператор *break* осуществляет немедленный выход из цикла и управление передается следующему за циклом оператору. Оператор *continue* начинает новую итерацию цикла, даже если предыдущая не была завершена.

В Octave файлы с расширением .m могут быть оформлены как отдельные функции. В этом случае имя функции должно совпадать с именем файла, в котором она хранится. Например, функция с именем *example* должна храниться в файле example.m. Функции имеют определенную структуру. Так, первая строка функции – это заголовок вида

$$function\ [y_1,y_2,...y_n]=name\ function(x_1,x_2,...,x_m),$$

где *name function* – имя функции; x_1, x_2, …, x_m – список ее входных параметров; y_1, y_2, …, y_n – список выходных параметров. После заголовка следуют требуемые операторы. Для обозначения конца функции используется ключевое слово *end*.

```
function [y1,y2,...yn]=name_function(x1,x2,...,xm)
```

```
    оператор1;
    оператор2;
    ...
    оператор n;
end
```

В файле с расширением .m, кроме основной функции, могут находиться так называемые подфункции, которые сами является функциями, но доступны они только внутри этого файла.

```
%Начало основной функции из  m-файла, имя которой должно совпадать
% с именем файла, в котором она хранится
function [y1,y2,...yn]=name function(x1,x2,...,xm)
% Среди операторов основной функции могут быть операторы
% вызова подфункций f1, f2, f3, ...,fl
    оператор1;
    оператор2;
    ...
    оператор n;
end; % здесь заканчивается основная функция

function [y1,y2,...yn]=f1(x1,x2,...,xm) % начало первой подфункции
    операторы
end % конец первой подфункции
...
function [y1,y2,...yn]=fn(x1,x2,...,xm) % начало n-й подфункции
    операторы
end % конец n-й подфункции
```

В Octave есть возможность передавать как входной параметр имя функции, что существенно расширяет рамки программирования. Для этого используется функция *feval*, аргументами которой являются строка с именем вызываемой функции (встроенной или определенной пользователем) и параметры вызываемой функции, разделенные запятой.

```
>> a=[2;4];
>> feval('sum',a)
ans =
   6
```

Встроенные функции *tic* и *toc* используются для измерения времени работы любого блока кода, расположенного между ними.

```
tic
    блок кода
toc
```

Иногда необходимо просмотреть информацию по всем переменным или по какой-либо определенной переменной. Это можно сделать через область переменных (см. рисунок А.1) или воспользоваться командами who (выводит список переменных) и whos (выводит список переменных с указанием их размера и объема занимаемой памяти).

```
>> A=[1 2; 3 4];
>> b='line';
>> c=5;
>> d=1+i;
>> who
Variables in the current scope:
A  b  c  d
>> whos
Variables in the current scope:
  Attr      Name       Size                Bytes      Class
  ====      ====       ====                =====      =====
            A          2x2                    32      double
            b          1x4                     4      char
            c          1x1                     8      double
  c         d          1x1                    16      double
Total is 10 elements using 60 bytes
>> whos d
Variables in the current scope:
  Attr      Name       Size                Bytes      Class
  ====      ====       ====                =====      =====
  c         d          1x1                    16      double
Total is 1 element using 16 bytes
```

Оглавление

Учебное издание

Куксенко Сергей Петрович

ЭЛЕКТРОМАГНИТНАЯ СОВМЕСТИМОСТЬ: ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ЭЛЕКТРОСТАТИКИ

Учебное пособие

Подписано в печать 18.11.20. Формат 60x84/16.
Усл. печ. л. 15,58. Тираж 100 экз. Заказ № 259.

Томский государственный университет
систем управления и радиоэлектроники.
634050, г. Томск, пр. Ленина, 40.
Тел. (3822) 533018.