1. Решение краевых задач в MatLab

1. Алгоритм решения краевой задачи. На основе предыдущего, приходим к следующей последовательности действий для нахождения приближенного решения исходной задачи:

1) определение геометрии области (области Ω и участков ее границы Γ и Γ_N);

2) определение коэффициентов и правых частей уравнения и краевых условий;

3) построение триангуляции области; в частности, определение матриц связности элементов t и граничных ребер e;

4) формирование матрицы A и вектора F;

5) учет краевых условий: определение индексов узлов i_N и i_D; формирование H и G; построение A_0 и F_0;

6) решение системы уравнений $A_0 u_0 = F_0$; образование вектора узловых параметров u решения u_h по u_0;

7) обработка решения u_h (например, построение графика u_h).

Отметим, что способ задания области на 1-ом шаге важен только для второго и третьего шага, поскольку он должен обеспечить в удобном виде нужную информацию для алгоритма триангуляции области. Выбор способа кодировки триангуляции является важным решением, поскольку он непосредственно влияет на программную реализацию шагов 4, 5, 7.

Задача построения подходящих для МКЭ триангуляций областей является непростой и самостоятельной задачей, важной не только для метода конечных элементов.

2. pde toolbox. Набор программ для решения задач из некоторой предметной области в MatLab называется toolbox. Для решения разнообразных краевых задач для уравнений в частных производных в двумерных областях (как стационарных, так и нестационарных),

имеется Partial Differential Equation Toolbox (pde-toolbox). Он включает графическое приложение pdetool, используя который пользователь в удобной форме может решать конкретные задачи в плоских областях.

Все функции pde-toolbox доступны также для вызова из программы пользователя, так что их можно использовать при решении самых разнообразных задач. Pde-toolbox содержит средства для исследования и решения краевых задач для дифференциальных уравнений в частных производных второго порядка. В нем используется метод конечных элементов на основе линейных треугольных конечных элементов, который был описан выше.

Функции pde-toolbox и графический интерфейс пользователя могут быть использованы для математического моделирования применительно к широкому классу инженерных и научных приложений, включая плоские статические задачи теории упругости, расчеты электромагнитных устройств, задачи тепломассопереноса, диффузии и т.д.

Доступ к графическому приложению pdetool осуществляется через командную строку Matlab. В ней необходимо набрать команду pdetool. Далее, в открывшемся окне мы увидим интерфейс приложения, меню которой позволяет задать геометрию области, определить решаемую задачу и т.д. (рис. 1).

3. Алгоритм решения задачи в среде pdetool. Для решения каждой задачи необходимо пройти следующие шаги.

1) определение геометрии области (области Ω и участков ее границы Γ и Γ_N);
2) определение краевых условий;
3) определение коэффициентов и правых частей уравнения;
4) построение триангуляции области;
5) решение задачи;
6) построение графиков решения.

Для выполнения этих шагов имеются соответствующие пункты меню, а также быстрые кнопки, расположенные под ними.

4. Описание пунктов меню *pdetool.* Отметим сразу, что в конце второй линии меню в списке указано Generic Scalar (общее ска-

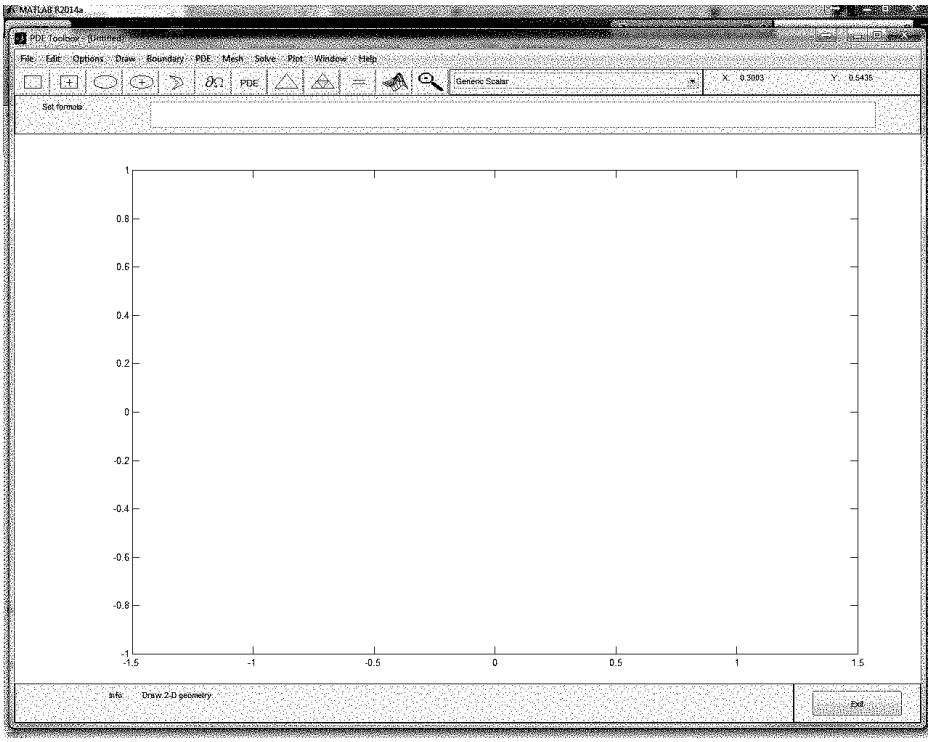

Рис. 1. Внешний вид GUI pdetool в MatLab.

лярное уравнение в частных производных того же вида, что мы рассмотрели выше). Из списка можно выбрать ту задачу, которую мы хотим решить (если она есть в списке; например, — уравнение теплопроводности — Heat Trnsfer).

Дадим краткое описание пунктов меню.

1) **Draw**: определение геометрии области. В *pdetool* область задается теоретико-множественным объединением (+), вычитанием (-) и пересечением (*) трех основных типов подобластей: прямоугольник, эллипс, многоугольник. Каждой подобласти присваивается своя метка, а область определяется соответствующей формулой.

Для примера рассмотрим, как получить область, изображенную на левом рис. 2: она состоит из трех подобластей, отмеченных на рисунке цифрами 1, 2, 3 (метками). [1] Области с метками 1, 3 — прямоугольники одинакового размера; область 2 — квадрат; из области 1 удален круг радиуса 0.25.

Первоначально, нажмем быструю кнопку в виде прямоугольни-

[1]Предполагается, что в этих подобластях коэффициенты дифференциального уравнения определяются разными формулами.

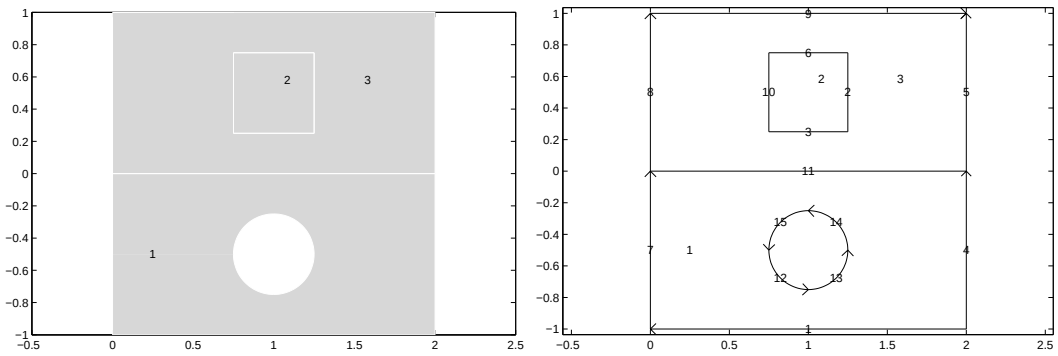

Рис. 2. Пример области (слева) и части его границы (отрезки со стрелками).

ка и, удерживая нажатой левую клавишу мыши, указываем в любое место осей координат и протягиваем мышку так, чтобы образовался прямоуглльник. Он получит метку R1. Дважды кликаем на эту метку и вводим в появившемся окне размеры прямоугольника, указывая координаты его левого-нижнего угла (0,-1), длину (2) и высоту (1). Если на экране эта подобласть не помещается, в меню Options/Axes Limits выбираем (ставим галочки) auto как по оси x, так и по оси y. Эти шаги завершают построение первой подобласти R1. Повторяем эти операции и строим вторую подобласть — квадрат R2, указывая его координаты левого-нижнего угла (0.75,0.25), длину (0.5) и высоту (0.5). Еще раз повторяем эти операции и наносим на оси третью подобласть R3 в виде прямоугольника, указывая его координаты левого-нижнего угла (0,0), длину (2) и высоту (1). Наконец, аналогично, только используя быструю кнопку в виде эллипса, строим круг E1, задавая центр эллипса (1,-0.5) и его A и B полуоси, равные 0.25. Остается в окошке Set Formula (над осями) ввести формулу

$$(R1 + R2 + R3) - E1$$

Эти операции завершают определение геометрии области. Выбирая меню PDE/PDE Mode, можно посмотреть на построенную область. Рекомендуется сохранить в .m файле проделанную работу, пользуясь File/Save as.

2) **Boundary**: задание краевых условий. Перейдем в меню Boundary/Boundary Mode. Мы увидим внутренние границы области (выделены серым) и части его границы (красные отрезки со стрелками). На каждой такой части или на всех частях можно задать свое краевое условие (первого или третьего рода). Чтобы задать краевое

условие на некоторой части границы, достаточно дважды кликнуть на нее мышкой и в появившемся окне выбрать тип краевого условия: первого рода — Dirichlet, или третьего рода — Newmann и задать соответствующие функции ($h = 1$, $r = u^D$ или $q = q_N$, $g = g_N$). Если на нескольких (всех) участках границы задано одно и то же условие, то удерживая клавишу ctrl выбираем эти участки (они окрашиваются черным цветом) и выбирая меню Boundary/Specify Boundary Conditions, задаем аналогично краевое условие. При вводе функции записываются в виде допустимых выражений MatLab, включая различные функции, в терминах переменных x и y.

3) **PDE**: задание коэффициентов уравнения. Перейдем в меню PDE/PDE Mode. Мы увидим подобласти. Дважды кликая на выбранную подобласть мышкой, в появившемся окне выбраем тип уравнения и задаем его коэффициенты. На верхней строчке окна указывается вид уравнения, при этом штрих указывает на производную по времени.

4) **Mesh**: триангуляции области. В меню Mesh/Parametrs можно задать параметры сетки, например, величину шага сетки h (maximum edge size — максимальный размер треугольников). Если он не задан, то программа выбирает этот параметр самостоятельно. Построить триангуляцию можно нажав быструю кнопку в виде треугольника. Нажав быструю кнопку в виде четырех треугольников, можно получить новую триангуляцию, когда каждый треугольник делится на четыре равные части. Полезно после этого выбрать меню Mesh/Jiggle Mesh, чтобы несколько сгладить сетку узлов.

5) **Solve**: решение задачи. Нажимая быструю кнопку в виде знака =, можно решить стационарную краевую задачу. При решении нестационарных задач в меню Solve/Parametrs дополнительно указывается моменты времени, в которых определяется решение задачи, а также функции, определяющие начальные условия. Например, задавая в окне Time вектор 0:0.1:1, мы указываем, что решение задачи надо определять в моменты времени 0, 0.1, 0.2, ..., 1.0. При решении задач на собственные значение — здесь указывается интервал, содержащий интересующие нас собственные числа.

6) **Plot**: построение графиков решения. Меню Plot/Parametrs или

быстрая кнопка в виде графика, позволяет получить графическое представление решения. Тип графика указывается в первом столбце: contour — строятся линии уровня решения, количество которых задается в окне Contour plot levels; arrows — строится векторное поле градиента или потока решения в виде стрелок; Height (3D plot) — строится трехмерный график решения. При решении нестационарных задач выбрав Animation, можно решение показать в виде мультфильма. Выбор пункта Show mesh позволяет нанести триангуляцию на график решения.

5. Решение модельной задачи с использованием функций pde-toolbox. Рассмотрим кратко, как, используя функции pde-toolbox, можно решить следующую краевую задачу в единичном круге Ω:

$$-\triangle u(x) = 1, \quad x \in \Omega, \quad u(x) = 0, \quad x \in \Gamma. \tag{0.1}$$

Она соответствует рассматриваемой нами задаче (**??**), если принять $c \equiv 1$, $a = u^D \equiv 0$, $f \equiv 1$, $\Gamma_N = \emptyset$ и имеет точное решение

$$u(x) = -(x_1^2 + x_2^2 - 1)/4.$$

Приведем набор операторов, который позволяет решить эту задачу, не используя графический интерфейс. Результаты выполнения представленных ниже команд приведены на рис. 3. Из результатов вычислений следует, что максимальная погрешность приближенного решения в равномерной норме (поточечно) не превосходит $5 \cdot 10^{-4}$, тогда как максимум решения равен 0.25. Следовательно, относительная погрешность приближенного решения не превосходит 0.2 процента.

```
g  ='circleg';       % 1) задание геометрии области
bc ='circleb1';      % 2) задание краевых условий
c=1; a=0; f=1;       % 2) задание коэффициентов уравнения

[p,e,t]=initmesh(g,'Hmax',0.1); % 3) построение триангуляции
u=-(p(1,:).^2+p(2,:).^2-1)/4;   % точное решение u=-(x_1^2+x_2^2)/4

uh=assempde(bc,p,e,t,c,a,f);    % 4) сборка и решение системы МКЭ

figure;                         % 5) график триангуляции
pdemesh(p,e,t), axis equal xlabel('x_1'), ylabel('x_2')
title(['Number of mesh points np=' int2str(size(p,2))...
       ',  triangles nt=' int2str(size(t,2))])
figure;                         % 6) график погрешности
pdesurf(p,t,u'-uh) colormap('hsv') xlabel('x_1'), ylabel('x_2'),
```

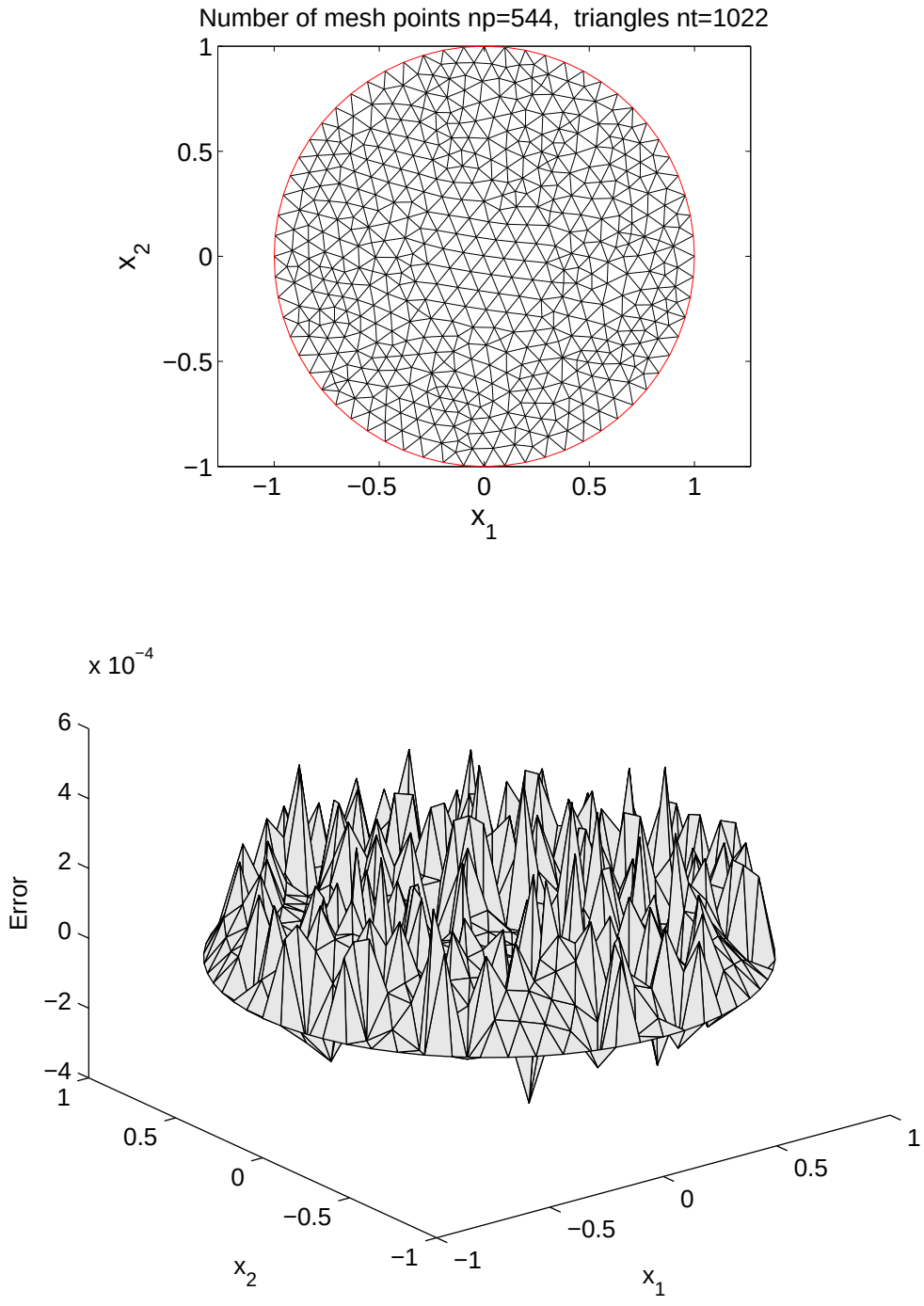

Рис. 3. Триангуляция области и погрешность решения задачи (0.1) методом конечных элементов.

```
zlabel('Error')
```

Ипользованные здесь функции circleg, circleb1, initmesh, assempde, pdemesh, pdesurf входят в состав pde-toolbox.