

Exercises

The exercises in this chapter are concerned with boundary value problems defined on a rectangular domain

$$\Omega = \{(x, y) \in R^2 | x_0 \leq x \leq x_0 + L_1, y_0 \leq y \leq y_0 + L_2\} \quad (2.49)$$

which is illustrated in Figure 2.5 for the case $(x_0, y_0) = (0, 0)$. The finite element meshes to be used are of the type shown in the same Figure. Note that the mesh in the figure may be viewed as a 4×3 array of smaller rectangles, each consisting of two triangular elements.

Matlab arguments

<code>x0, y0:</code>	The coordinates (x_0, y_0) of the lower left point of the domain.
<code>L1:</code>	Length of the domain (L_1). Interval $x \in [x_0, x_0 + L_1]$.
<code>L2:</code>	Height of the domain (L_2). Interval $y \in [y_0, y_0 + L_2]$
<code>noelms1:</code>	Number of rectangles in the x -direction. (<code>noelms1</code> = 4 in Figure 2.5).
<code>noelms2:</code>	Number of rectangles in the y -direction. (<code>noelms2</code> = 3 in Figure 2.5).
<code>noelms:</code>	Number of elements. (<code>noelms</code> = 24 in Figure 2.5).
<code>nonodes:</code>	Number of nodes. (<code>nonodes</code> = 20 in Figure 2.5).
<code>VX(1:nonodes):</code>	The x -coordinates of the nodes.
<code>VY(1:nonodes):</code>	The y -coordinates of the nodes.
<code>EToV(1:noelms,1:3):</code>	The Element To Vertice connectivity table. (See Exercise 2.1).
<code>delta:</code>	The area of an element.
<code>abc(1:3,1:3):</code>	Coefficients of the local basis functions. (See Exercise 2.2).
<code>lam1, lam2:</code>	Constant values of the functions $\lambda_1(x, y)$ and $\lambda_2(x, y)$.
<code>qt(1:nonodes):</code>	Values of $\tilde{q}(x, y)$ at the nodes.
<code>A(1:nonodes,1:nonodes):</code>	The global matrix, in Matlab's sparse format.
<code>b(1:nonodes):</code>	The global right-hand-side vector.
<code>nobnodes:</code>	The number of nodes on Γ_2 . (See Exercise 2.4).
<code>bnodes(1:nobnodes):</code>	The global node numbers of nodes on Γ_2 .
<code>f(1:nobnodes):</code>	The values of $f(x, y)$ at the nodes on Γ_2 .
<code>nobeds:</code>	The number of element edges on Γ_1 . (See Exercise 2.6).
<code>beds(1:nobeds,1:2):</code>	The element edges on Γ_1 .
<code>q(1:nobeds):</code>	The values of $q(x, y)$ at the midpoints of the element edges on Γ_1 .

N.B. In all test cases that require values of `lam1` and `lam2`, put `lam1=lam2=1`.

Exercise 2.1

Let a finite element mesh be imposed on domain (2.49) of the type shown in Figure 2.5.

- a) Write a Matlab routine that computes the arrays `VX(1:nonodes)` and `VY(1:nonodes)`, the x and y coordinates of the mesh nodes. As illustrated in Figure 2.5, the ordering of the nodes should be column-wise, beginning at the left boundary and running downward in each column.

Head:

```
function [VX,VY] = xy(x0,y0,L1,L2,noelms1,noelms2)
```

Test case:

$(x_0, y_0) = (-2.5, -4.8)$, $L_1 = 7.6$, $L_2 = 5.9$, `noelms1 = 4`, `noelms2 = 3`.

- b) Write a Matlab routine that constructs an Element-To-Vertice array `EToV(1:noelms,1:3)` defining how mesh nodes are connected to define the element of the triangular mesh. `EToV(n,r)` is defined to be the global node number of the node in element e_n with local number r . For the mesh in Figure 2.5, for example, the 4'th row of `EToV` should consist of the values $\text{EToV}(4,1) = 3$, $\text{EToV}(4,2) = 7$ and $\text{EToV}(4,3) = 2$.

Head:

```
function EToV = conelmtab(noelms1,noelms2)
```

Test case:

`noelms1 = 4`, `noelms2 = 3` (See Figure 2.5).

Exercise 2.2

For a given finite element mesh we need to be able to compute the geometric information from the mesh data tables when generating the system of equations defining our discrete finite element method.

- a) Write a Matlab routine that computes, for a given element e_n , the quantities Δ and \tilde{a}_i , \tilde{b}_i , \tilde{c}_i , $i = 1, 2, 3$ defined by (2.1) and (2.3), respectively. Test output using $n = 9$.

The contents of array `abc` should be arranged as follows:

$$\begin{aligned} \text{abc}(1,1) &= \tilde{a}_1, & \text{abc}(1,2) &= \tilde{b}_1, & \text{abc}(1,3) &= \tilde{c}_1 \\ \text{abc}(2,1) &= \tilde{a}_2, & \text{abc}(2,2) &= \tilde{b}_2, & \text{abc}(2,3) &= \tilde{c}_2 \\ \text{abc}(3,1) &= \tilde{a}_3, & \text{abc}(3,2) &= \tilde{b}_3, & \text{abc}(3,3) &= \tilde{c}_3 \end{aligned}$$

Head:

```
function [delta,abc] = basfun(n,VX,VY,EToV)
```

- b) Write a Matlab routine that computes, for a given element e_n and edge number k , the outer normal vector components n_1 and n_2 as defined by (2.6). The edge number is defined from the `EToV` table, e.g. edge 1 is defined for element n by the vertices `EToV(n,1:2)`, edge two by `EToV(n,2:3)` and edge three by `EToV(n,[3 1])`. Test output using $n = 9$.

Head:

```
function [n1,n2] = outernormal(n,k,VX,VY,EToV)
```

Test case:

$(x_0, y_0) = (-2.5, -4.8)$, $L_1 = 7.6$, $L_2 = 5.9$, $\text{noelms1} = 4$, $\text{noelms2} = 3$.

Exercise 2.3

- Write a Matlab routine that implements the assembly process described by Algorithm 4 but stores *all* nonzeros of the global matrix A , not just those in the upper triangle. (This makes it possible to use later Matlab's very efficient equation solver command $\mathbf{u} = \mathbf{A}\backslash\mathbf{b}$). Assume that $\lambda_1(x, y)$ and $\lambda_2(x, y)$ are constants (`lam1` and `lam2`) and that $\tilde{q}(x, y)$ is defined at the mesh nodes by array `qt(1:nonodes)`. Take the value of \tilde{q} in (2.28) to be the average of $\tilde{q}(x, y)$ at the three nodes of the element.
- For each test case use the Matlab command `[B,d] = spdiags(A)` to print `A`. In case 2 print also `b`.

Head:

```
function [A,b] = assembly(VX,VY,EToV,lambda1,lambda2,qt)
```

Array `A(1:nonodes,1:nonodes)` should contain the assembled matrix in Matlab's sparse format, and array `b(1:nonodes)` should contain the assembled right-hand-side vector.

Test cases:

- $(x_0, y_0) = (0, 0)$, $L_1 = 1$, $L_2 = 1$, $\text{noelms1} = 4$, $\text{noelms2} = 3$, $\tilde{q}(x, y) = 0$.
- $(x_0, y_0) = (-2.5, -4.8)$, $L_1 = 7.6$, $L_2 = 5.9$, $\text{noelms1} = 4$, $\text{noelms2} = 3$,
 $\tilde{q}(x, y) = -6x + 2y - 2$.

Exercise 2.4

- Write a Matlab routine based on Algorithm 6, i.e. in a version that does *not* exploit symmetry, that updates arrays A and b so as to impose the Dirichlet boundary condition

$$u = f(x, y), \quad (x, y) \in \Gamma_2$$

where Γ_2 is part, or all, of the boundary Γ . (The case where Γ_2 is only part of Γ arises in problem (2.32), (2.33)). The number of nodes on Γ_2 is `nobnodes`. The global node numbers of the nodes on Γ_2 are given by array `bnodes(1:nobnodes)`, and the values of $f(x, y)$ at these nodes are given by array `f(1:nobnodes)`.

- The test cases in this exercise continue those of Exercise 2.3. Call `dirbc` after `assembly`. Use command `[B,d] = spdiags(A)` to print `A`, and print also `b`.

Head:

```
function [A,b] = dirbc(bnodes,f,A,b)
```

Test cases: Let $\Gamma_2 = \Gamma$.

1. $(x_0, y_0) = (0, 0)$, $L_1 = 1$, $L_2 = 1$, `noelms1 = noelms2 = 4`,
 $\tilde{q}(x, y) = 0$, $f(x, y) = 1$.
2. $(x_0, y_0) = (-2.5, -4.8)$, $L_1 = 7.6$, $L_2 = 5.9$, `noelms1 = 4`, `noelms2 = 3`,
 $\tilde{q}(x, y) = -6x + 2y - 2$, $f(x, y) = x^3 - x^2y + y^2 - 1$.

Exercise 2.5

- a) Use the Matlab routines of the previous exercises to write a program that solves the boundary value problem

$$u_{xx} + u_{yy} = -\tilde{q}(x, y), \quad (x, y) \in \Omega \quad (2.50)$$

$$u = f(x, y), \quad (x, y) \in \Gamma \quad (2.51)$$

where Ω is the rectangular domain given by (2.49) and Γ is its boundary.

An easy but effective way to test the program is the following:

- Choose a simple function $u(x, y)$.
- Determine analytically the functions $\tilde{q}(x, y)$ and $f(x, y)$ that make $u(x, y)$ the solution of problem (2.50), (2.51); i.e., define

$$\tilde{q}(x, y) = -(u_{xx} + u_{yy}), \quad (x, y) \in \bar{\Omega}, \quad f(x, y) = u(x, y), \quad (x, y) \in \Gamma$$

- Solve problem (2.50), (2.51) using the program.
- Compute the error in the finite element solution. A suitable measure of the error is

$$E = \max_{1 \leq j \leq M} |\hat{u}_j - u(x_j, y_j)| \quad (2.52)$$

where \hat{u}_j is the finite element solution at node (x_j, y_j) and M is the number of nodes ($M = \text{nonodes}$).

- Produce a *convergence plot* where measured errors versus mesh size are plotted. Make sure to employ sufficiently small mesh sizes to clearly illustrate the *asymptotic* behavior of the error. It is a good idea to include a line in the plot which clearly shows the *expected* (from theory) behavior of the method tested.

Test cases:

1. $u(x, y) = x^3 - x^2y + y^2 - 1$,
 $(x_0, y_0) = (-2.5, -4.8)$, $L_1 = 7.6$, $L_2 = 5.9$, `noelms1 = 4`, `noelms2 = 3`.
a) Print the solution values \hat{u}_j and the error E .
2. $u(x, y) = x^2y^2$,
 $(x_0, y_0) = (-2.5, -4.8)$, $L_1 = 7.6$, $L_2 = 5.9$,
`noelms1 = noelms2 = 2^p`, $p = 1, 2, 3, \dots$
a) Compute E for each value of p .
b) E is a function of h , where h is the largest element edge length in the mesh. On the basis of the results from a), describe this function for small values of h .

Exercise 2.6

- a) Write a Matlab routine that constructs an array for holding information about boundary edges (`beds`) of the triangular mesh. (HINT: the use of a distance function `fd` can make the job of finding nodes that belong to a boundary implicitly defined by the distance function very easy. Below it is included in the argument list as an optional argument. `top` is a tolerance to be used with the signed distance function to tune the selection of nodes.)

Head:

```
function [beds] = ConstructBeds(VX,VY,EToV,tol [,fd])
```

- b) Write a Matlab routine that updates the global right-hand-side vector stored in array `b` so as to impose the Neumann boundary condition

$$\lambda_1 u_x n_1 + \lambda_2 u_y n_2 = -q(x, y), \quad (x, y) \in \Gamma_1$$

The computation required is given in Algorithm 8.

Head:

```
function b = neubc(VX,VY,EToV,beds,q,b)
```

`VX`, `VY` and `EToV` are the same as before. Introduce `nobeds` as the number of element edges on Γ_1 . Then an array `beds(1:nobeds,1:2)` is a list of these element edges. Each row (p) of `beds` identifies an element edge as follows: $n = \text{beds}(p,1)$ is the number of the element that contains the edge. $r = \text{beds}(p,2)$ is the local node number of the first node on the edge. (See Section 2.9 for the definition of "first node".)

The computation requires also the second node on the edge. If the local number of the second node is denoted s , then

$$r=1 \implies s=2, \quad r=2 \implies s=3, \quad r=3 \implies s=1$$

assuming the three nodes are ordered counterclockwise.

Array `q(1:nobeds)` contains the values of the function $q(x, y)$ at the midpoints of the element edges.

Exercise 2.7

Let Γ_1 denote the left and bottom edges of the rectangular domain (2.49), and let Γ_2 denote the right and top edges.

- a) Use the Matlab routines of the previous exercises to write a program that solves the boundary value problem

$$u_{xx} + u_{yy} = -\tilde{q}(x, y), \quad (x, y) \in \Omega$$

$$u_n = -q(x, y), \quad (x, y) \in \Gamma_1, \quad u = f(x, y), \quad (x, y) \in \Gamma_2$$

where u_n is the outward-directed normal derivative of u . Since we have here $\lambda_1(x, y) = \lambda_2(x, y) = 1$, the boundary condition on Γ_1 is the Neumann condition (2.32). Note that $u_n = -u_x$ on the left edge of the boundary and $u_n = -u_y$ on the bottom edge.

Test cases

These test cases are based on the testing procedure described in Exercise 2.5. The functions to be derived analytically from the given solution u are now \tilde{q} , q and f . The error E is defined in (A.40).

1. $u(x, y) = 3x + 5y - 7,$
 $(x_0, y_0) = (-2.5, -4.8), \ L_1 = 7.6, \ L_2 = 5.9, \ \text{noelms1} = 4, \ \text{noelms2} = 3.$
 a) Print the solution values \hat{u}_j in the 2-D array format, where the solution array is allocated as $\mathbf{u} = \text{zeros}(\text{noelms2+1}, \text{noelms1+1}).$
 Print also the error $E.$

2. $u(x, y) = \sin(x) \sin(y),$
 $(x_0, y_0) = (-2.5, -4.8), \ L_1 = 7.6, \ L_2 = 5.9, \ \text{noelms1} = \text{noelms2} = 32.$
 a) Print $E.$

Exercise 2.8

The Neumann boundary condition is often used to exploit symmetry in the solution of a boundary value problem. Consider, for example, the problem

$$u_{xx} + u_{yy} = -2\pi^2 \cos(\pi x) \cos(\pi y), \quad -1 \leq x, y \leq 1 \quad (2.53)$$

$$u(x, y) = \cos(\pi x) \cos(\pi y), \quad (x, y) \in \Gamma \quad (2.54)$$

where Γ is the boundary of the domain in (2.53). The domain as well as the function on the right-hand side of (2.53) are symmetric about both the x and y axes. The solution $u(x, y)$ inherits these symmetry properties, so it is sufficient to determine $u(x, y)$ on a quarter of the domain.

- a) Formulate a boundary value problem for $u(x, y)$ on the subdomain $0 \leq x, y \leq 1.$
- b) Use your Matlab software to solve this problem. Let $\text{noelms1} = \text{noelms2} = 3$ and print the solution in a 2-D format.
- c) Use your Matlab software to solve the original problem, (2.53), (2.54). Let $\text{noelms1} = \text{noelms2} = 6$, and print the solution in a 2-D format.
- d) Both programs compute an approximation to the value $u(0, 0).$ Compute these approximations for the following meshes:

Program b): $\text{noelms1} = \text{noelms2} = 2^p, p = 1, 2, \dots, 6.$

Program c): $\text{noelms1} = \text{noelms2} = 2^{p+1}, p = 1, 2, \dots, 6.$

Note that for the same value of p , the element dimensions in the two cases are identical; i.e., the two meshes have the same fineness.

