Contents

1. Linear Elliptic PDEs

1.1. **Definition.** Linear elliptic PDEs (partial differential equations) are those of the form

$$-\mathsf{tr}\,(M(x)\nabla\nabla u(x)) + b(x)\cdot\nabla u + c(x)u(x) = f(x), \quad x\in\Omega\subset\mathbb{R}^n \tag{1}$$

where ∇u and $\nabla\nabla u$ denote the gradient and the Hessian of the function u:

$$\nabla u := \begin{bmatrix} \frac{\partial u}{\partial x_1} \\ \vdots \\ \frac{\partial u}{\partial x_n} \end{bmatrix}, \quad \nabla\nabla u := \begin{bmatrix} \frac{\partial^2 u}{\partial x_1^2} & \cdots & \frac{\partial^2 u}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 u}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 u}{\partial x_n^2} \end{bmatrix}.$$

The $n\times n$ matrix function $M(x)$ is continuously differentiable such that for all $x\in\Omega$ $M(x) = M(x)^\top$ and

$$x^\top M(x) x \geq \alpha\|x\|_2^2 \text{ for some } \alpha > 0, \tag{2}$$

$b(x)$ is a continuously differentiable vector function, $c(x)$ is continuous scalar function, and $f(x)$ is a given scalar function. Eq. (2) is called the *uniform ellipticity condition*. Note

that $\mathsf{tr}\,(M(x)\nabla\nabla u(x))$ is the sum of all entries of the entry-wise matrix product:

$$\mathsf{tr}\,(M(x)\nabla\nabla u(x)) \equiv \sum_{i,j=1}^{n} M_{ij}(x)\frac{\partial^2 u}{\partial x_i \partial x_j}.$$

Typically, the models leading to equations of the form (1) imply some boundary conditions. The task to solve an equation supplemented only with boundary conditions is called a *Boundary Value Problem* (BVP).

We will assume that the domain Ω is connected and bounded. We will consider three types of boundary conditions: Dirichlet, Neumann, and periodic. Dirichlet conditions are:

$$u(x) = g(x), \quad x \in \partial\Omega, \tag{3}$$

Neumann conditions are

$$\frac{du(x)}{d\nu} \equiv \nabla u \cdot \nu = g(x), \quad x \in \partial\Omega, \tag{4}$$

where ν is the unit outer normal to the boundary $\partial\Omega$. Periodic conditions are often imposed when one or more components of x are angles. Let the first component of x be an angle ranging between 0 and 2π: $0 \le x_1 \le 2\pi$. Then the periodic boundary conditions are given by

$$u(0, x_2, \ldots, x_n) = u(2\pi, x_2, \ldots, x_n), \quad \tfrac{\partial}{\partial x_1}u(0, x_2, \ldots, x_n) = \tfrac{\partial}{\partial x_1}u(2\pi, x_2, \ldots, x_n). \tag{5}$$

If the boundary $\partial\Omega$ consists of several disjoint pieces then different types of boundary conditions can be prescribed on each connected component of the boundary.

1.2. **Where linear elliptic PDEs come from.**

1.2.1. *Stationary heat distribution.* The stationary heat (or temperature) distribution in the domain Ω with a heat conductance coefficient equal to 1 is given by the solution of *the Poisson equation*

$$-\Delta u = f(x), \tag{6}$$

where the symbol Δ denotes the Laplacian

$$\Delta u := \frac{\partial^2 u}{\partial x_1^2} + \ldots + \frac{\partial^2 u}{\partial x_n^2},$$

$f(x)$ is a given function representing the heat source, i.e., amount of heat supplied per unit time in a small neighborhood of x divided by the volume of the neighborhood. The term *stationary* or, *steady-state*, means that the distribution does not change with time.

If $f(x) \equiv 0$ then Eq. (6) becomes *Laplace's equation*

$$\Delta u = 0. \tag{7}$$

If the domain Ω is insulated, i.e., there is no heat flux through its boundary, then one should pick the homogeneous Neumann boundary conditions

$$\frac{du(x)}{d\nu} \equiv \nabla u \cdot \nu = 0, \quad x \in \partial\Omega. \tag{8}$$

If the temperature distribution on the boundary of Ω is enforced to be $g(x)$ then one should pick the Dirichlet boundary condition (3).

If the heat conductance coefficient $\kappa(x)$ is variable, then the stationary heat distribution is the solution to the equation

$$\nabla \cdot (\kappa(x)\nabla u) + f(x) = 0. \tag{9}$$

Eq. (9) can be rewritten as

$$\nabla\kappa(x) \cdot \nabla u + \kappa(x)\Delta u + f(x) = 0. \tag{10}$$

1.2.2. *Electrostatics.* The electric potential $\phi(x)$ and the electric field $\mathbf{E}$ are related via

$$\mathbf{E} = -\nabla\phi.$$

The first Maxwell equation is Gauss's law, $\nabla \cdot \mathbf{E} = 4\pi\rho(x)$, where $\rho(x)$ is the density of the electric charges. Rewriting it in terms of the electric potential one obtains the Poisson equation

$$-\Delta\phi = 4\pi\rho(x). \tag{11}$$

If Eq. (11) is solved in an infinite domain without boundary, its solution is nonunique. For example, $-\nabla^2\phi = 0$, $x \in \mathbb{R}^2$ has infinitely many linearly independent solutions, e.g., $\phi_1(x) = x_1^2 - x_2^2$, $\phi_2(x) = \cos(x_1)e^{x_2}$, etc. Typically, one seeks solutions bounded at infinity. This condition still determines the function $\phi(x)$ up to an additive constant.

1.2.3. *Electrodynamics.* Let us consider a continuous medium Ω with conductivity $\sigma(x)$, where the unit voltage is applied between the given surfaces (or curves) ∂A and ∂B. Then the electric potential ϕ can be defined so that

$$\phi(\partial A) = 1 \quad \text{and} \quad \phi(\partial B) = 0.$$

According to Kirchhoff's generalization of Ohm's law, the density of electric current in the region $\Omega_{AB} := \Omega\backslash(A \cup B)$ is given by

$$j(x) = -\sigma(x)\nabla\phi(x). \tag{12}$$

According to the generalization of Ampere's law,

$$\nabla \cdot j(x) = -\frac{\partial p}{\partial t}, \tag{13}$$

where $p(t)$ is the density of electric charges in the medium. We assume that the electric current is stationary, hence the density of the electric charges is independent of time. Then

$$\nabla \cdot j(x) = 0. \tag{14}$$

Plugging Eq. (12) into Eq. (14) we obtain

$$\nabla \cdot (\sigma(x)\nabla\phi(x)) \equiv \nabla\sigma(x) \cdot \nabla\phi(x) + \sigma(x)\Delta\phi(x) = 0. \tag{15}$$

The boundary conditions on ∂A and ∂B are given by

$$\phi(\partial A) = 1, \quad \phi(\partial B) = 0. \tag{16}$$

The Neumann boundary conditions $\nabla\phi \cdot \nu$ on $\partial\Omega$ mean that no charges escape from Ω, i.e., the current at $\partial\Omega$ is tangent to it.

1.2.4. *Stochastic processes.* Consider a particle evolving according to a *stochastic differential equation* (SDE) of the form

$$dX_t = b(X_t)dt + \sigma(X_t)dW_t, \quad X_0 = x, \quad x \in \Omega \subset \mathbb{R}^n, \tag{17}$$

where b is a continuously differentiable vector function, σ is a continuously differentiable matrix function, and dW_t is the increment of the standard Brownian motion. Let $f(X_t)$ be a function. The family of transfer operators P_t, $t \geq 0$, is defined by

$$(P_t f)(x) = E[f(X_t) \mid X_0 = x]. \tag{18}$$

Now we can define a function

$$u(x,t) := (P_t f)(x) \equiv E[f(X_t) \mid X_0 = x]. \tag{19}$$

Then its time evolution is given by the *backward Kolmogorov equation*

$$\frac{\partial u}{\partial t} = \lim_{s\to 0} \frac{(P_{t+s}f)(x) - (P_t f)(x)}{s} =: LP_t f \equiv Lu. \tag{20}$$

The derivative of the transfer operator in Eq. (20) is called its *infinitesimal generator*. Applying Ito's formula to SDE (17) one can calculate L explicitly:

$$L = b(x) \cdot \nabla + \frac{1}{2}\mathsf{tr}\left(\Sigma(x)\nabla\nabla\right), \quad \Sigma(x) := \sigma(x)\sigma(x)^\top. \tag{21}$$

Now we consider some particular choices of the function $f(X_t)$.

- Let $A \subset \mathbb{R}^n$ and $B \subset \mathbb{R}^n$ be some regions. The *committor function* $q(x)$ (a.k.a. the equilibrium potential or the capacitor) is defined as the probability that the process starting at the point x will first reach B rather than A. Let us derive a boundary-value problem for the committor. It is clear that $q(\partial A) = 0$ and $q(\partial B) = 1$. For $x \in (A \cup B)^c$ it is clear that the committor does not depend on time explicitly, i.e.,

  $$\frac{\partial q}{\partial t} = 0.$$

  Therefore, using Eqs. (20) and (21) we obtain the boundary-value problem for it:

  $$b(x) \cdot \nabla q + \frac{1}{2}\mathsf{tr}\left(\Sigma(x)\nabla\nabla q\right) = 0, \quad x \in (A \cup B)^c, \quad q(\partial A) = 0, \quad q(\partial B) = 1.$$

- Let $A \subset \mathbb{R}^n$ be some region. The *first passage time* to A, τ_A, is defined as
$$\tau_A = \inf\{t \geq 0 \mid X_t \in A\}.$$
Let $u(x,t)$ be the expected first passage time to A for the process X_t starting at x and evolving according to SDE (17), i.e.,
$$u(x,t) = E\left[\tau_A \mid X_t = x\right].$$
We note that
$$\frac{\partial}{\partial t}u(x,t) = 1 \quad \text{and} \quad u(\partial A) = 0.$$
Plugging this into Eq. (20) and using Eq. (21) and taking care of the re-definition of $u(x,t)$ (compare it with Eq. (19)) one can derive the following boundary value problem for $u(x,t)$:

$$b(x)\cdot\nabla u + \frac{1}{2}\mathsf{tr}\left(\Sigma(x)\nabla\nabla u\right) = -1, \quad x \notin A, \quad u(\partial A) = 0. \tag{22}$$

- If $X_t \in \Omega \subset \mathbb{R}^n$, then the homogeneous Neumann boundary condition of $\partial\Omega$
$$\frac{\partial u}{\partial \nu} = 0 \text{ for } x \in \partial\Omega$$
means that the particle is reflected from the boundary $\partial\Omega$ whenever it reaches it.

1.3. **Strong and weak maximum principles.** It is important to have a clue about how solutions of elliptic equations behave. The most important result that is easy to check by visual inspection of the plotted solution is the strong maximum principle. For brevity, we will denote the linear elliptic operator by L:

$$Lu := -\mathsf{tr}\left(\Sigma(x)\nabla\nabla u\right) + b(x)\cdot\nabla u + c(x)u. \tag{23}$$

We chose the sign "$-$" in front of the first term in L following the common practice. It makes the corresponding matrix obtained via a spatial discretization of L symmetric positive definite provided that the discretization is fine enough. We will get to it later.

We assume that M, b, and c are as described in Section 1.1. We assume that the function u is defined in a connected, open and bounded domain $\Omega \subset \mathbb{R}^n$ and on its boundary $\partial\Omega$, and $u \in C^2(\Omega) \cap C(\partial\Omega)$.

Theorem 1. The Strong Maximum Principle

(1) *Let $c(x) \equiv 0$ in Ω.*
   (a) *Let $Lu \leq 0$ in Ω and attains its maximum over $\bar{\Omega}$ in its interior point, then u is constant within $\bar{\Omega}$.*
   (b) *Let $Lu \geq 0$ in Ω and attains its minimum over $\bar{\Omega}$ in its interior point, then u is constant within $\bar{\Omega}$.*

(2) *Let $c(x) \geq 0$ in Ω.*
   (a) *Let $Lu \leq 0$ in Ω and attains its nonnegative maximum over $\bar{\Omega}$ in its interior point, then u is constant within $\bar{\Omega}$.*
   (b) *Let $Lu \geq 0$ in Ω and attains its nonpositive minimum over $\bar{\Omega}$ in its interior point, then u is constant within $\bar{\Omega}$.*

The proof of this theorem is found e.g. in [4].

Theorem 1 implies the *weak maximum principle.*

Theorem 2. (1) *Let* $c(x) \equiv 0$ *in* Ω.

(a) *Let* $Lu \leq 0$ in Ω. *Then*

$$\max_{\bar{\Omega}} u = \max_{\partial\Omega} u.$$

(b) *Let* $Lu \geq 0$ in Ω. *Then*

$$\min_{\bar{\Omega}} u = \min_{\partial\Omega} u.$$

(c) *If* $Lu = 0$ *in* Ω *then*

$$\max_{\bar{\Omega}} |u| = \max_{\partial\Omega} |u|.$$

(2) *Let* $c(x) \geq 0$ *in* Ω.

(a) *Let* $Lu \leq 0$ in Ω. *Then*

$$\max_{\bar{\Omega}} u \leq \max_{\partial\Omega} \max\{u, 0\}.$$

(b) *Let* $Lu \geq 0$ in Ω. *Then*

$$\min_{\bar{\Omega}} u \geq \min_{\partial\Omega} \min\{u, 0\}.$$

We will prove the first claim of the weak maximum principle.

Proof. **Case** $Lu < 0$**.**

First, we will consider the case with a stronger assumption: $Lu < 0$ in Ω. Let x_0 be an interior maximum. Then $\nabla u(x_0) = 0$ and $\nabla\nabla u(x_0)$ is nonpositive definite. Since $M(x_0)$ is symmetric positive definite, it can be decomposed as $M(x_0) = ODO^\top$ where D is a diagonal matrix with positive entries and O is an orthogonal matrix, i.e., $O^\top = O^{-1}$. Next, we recall the cyclic property of trace. Therefore,

$$\begin{aligned}\mathsf{tr}\left(M(x_0)\nabla\nabla u(x_0)\right) &= \mathsf{tr}\left(M(x_0)^{1/2}\nabla\nabla u(x_0)M(x_0)^{1/2}\right)\\ &= \mathsf{tr}\left(D^{1/2}O^\top\nabla\nabla u(x_0)OD^{1/2}\right).\end{aligned}$$

The matrix $O^\top\nabla\nabla u(x_0)O$ is symmetric nonnegative definite. Hence its trace i.e., the sum of its diagonal entries, which is the sum of eigenvalues of $O^\top\nabla\nabla u(x_0)O$ is nonpositive. The trace of $D^{1/2}O^\top\nabla\nabla u(x_0)OD^{1/2}$ is the the linear combination of diagonal entries of $O^\top\nabla\nabla u(x_0)O$ with coefficients being the diagonal entries of D.

Remark The fact that $M(x_0)^{1/2}\nabla\nabla u(x_0)M(x_0)^{1/2}$ is nonpositive definite follows immediately from Sylvester's law of inertia.

Hence

$$\mathsf{tr}\left(M(x_0)\nabla\nabla u(x_0)\right) = \mathsf{tr}\left(D^{1/2}O^\top\nabla\nabla u(x_0)OD^{1/2}\right) \leq 0.$$

Therefore,

$$Lu(x_0) = -\mathsf{tr}\left(M(x_0)\nabla\nabla u(x_0)\right) + b\cdot\nabla u(x_0) = -\mathsf{tr}\left(M(x_0)\nabla\nabla u(x_0)\right) \geq 0.$$

On the other hand, by assumption, $Lu(x_0) < 0$ in Ω. This contradicts to the established fact that $Lu(x_0) \geq 0$. Hence maximum cannot be achieved at an interior point of Ω if $Lu < 0$ in Ω.

Case $Lu \geq 0$**.** To carry out a proof for this case, we will define a family of slightly modified functions $u^\epsilon(x)$ such that $Lu^\epsilon > 0$, use the previously proven case, and then take a limit $\epsilon \to 0$. We define the following family of functions u:

$$u^\epsilon(x) := u(x) + \epsilon e^{\lambda x_1}, \tag{24}$$

where x_1 is the first component of x and $\lambda > 0$ will be selected below. Next, we calculate Lu^ϵ

$$Lu^\epsilon = Lu + \epsilon L(e^{\lambda x_1}) \leq \epsilon e^{\lambda x_1}\left[-\lambda^2 M_{11} + \lambda b_1\right], \tag{25}$$

where $b_1(x)$ is the first component of $b(x)$ and $M_{11}(x)$ is the $(1,1)$-entry of $M(x)$. Since M is symmetric positive definite and obeys the strong ellipticity condition (2), $M_{11} \geq \alpha$. Therefore,

$$Lu^\epsilon \leq \epsilon e^{\lambda x_1}\left[-\lambda^2\alpha + \lambda \max_{\bar\Omega}|b_1|\right] < 0 \quad x\in\Omega \quad \text{if} \quad \lambda > \frac{1}{\alpha}\max_{\bar\Omega}|b_1|. \tag{26}$$

By our proof above,

$$\max_{\bar\Omega} u^\epsilon = \max_{\partial\Omega} u^\epsilon.$$

Letting $\epsilon \to 0$ we find that

$$\max_{\bar\Omega} u = \max_{\partial\Omega} u.$$

□

2. Finite difference approach. A model problem

We start with a simple model problem, the Poisson equation in 2D in a unit square $\Omega = \{0 \leq x \leq 1, 0 \leq y \leq 1\}$ with Dirichlet boundary conditions:

$$u_{xx} + u_{yy} - f(x,y) = 0, \quad (x,y)\in\Omega, \tag{27}$$

$$u = 0, \quad (x,y)\in\partial\Omega. \tag{28}$$

2.1. Numerical solution.

Let us introduce a mesh with step $h = 1/J$ and approximate Eq. (27) with the central difference scheme with a 5-point stencil (Fig. 1):

$$\frac{1}{h^2}\left(U_N + U_S + U_W + U_E - 4U_P\right) = f_P. \tag{29}$$

Each mesh point is naturally indexed with two indices (i,j) where $i,j \in \{0,1,\ldots,J\}$, i.e., $J+1$ points in each direction in total. I prefer to index them in the same order as the matrix entries are indexed (Fig. 1). The function u is known to be 0 at all boundary points of the mesh: $u_{0j} = u_{i0} = u_{Jj} = u_{iJ} = 0$, $i,j \in \{0,1,\ldots,J\}$. Therefore, we need to find u_{ij}, $1 \leq i,j \leq J-1$. To do it, we set up a linear system for u_{ij} of the form $Au = f$.

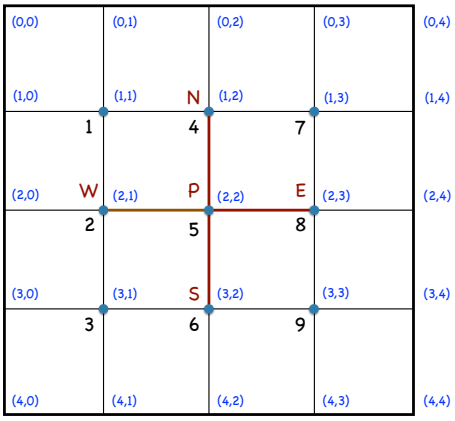


FIGURE 1. An illustration for the 5-point stencil and indexing of mesh points.

The matrix A will be $(J-1)^2 \times (J-1)^2$. We start with casting u_{ij}, $1 \le i, j \le J-1$ into a vector column-wise as shown in Fig. 1. This indexing is consistent with Matlab's operator (:) and its inverse operator `reshape` illustrated in the following example.

```
>> a = [1 4 7; 2 5 8; 3 6 9]

a =

     1     4     7
     2     5     8
     3     6     9

>> a1 = a(:)

a1 =

     1
     2
     3
     4
     5
     6
     7
     8
     9

>> a2 = reshape(a1,3,3)
```

```
a2 =

     1     4     7
     2     5     8
     3     6     9
```

For the example in Fig. 1 we obtain the following linear system.

$$Au = \frac{1}{h^2}$$

	1	2	3	4	5	6	7	8	9		
1	-4	1		1						u_1	f_1
2	1	-4	1		1					u_2	f_2
3		1	-4			1				u_3	f_3
4	1			-4	1		1			u_4	f_4
5		1		1	-4	1		1		u_5	f_5
6			1		1	-4			1	u_6	f_6
7				1			-4	1		u_7	f_7
8					1		1	-4	1	u_8	f_8
9						1		1	-4	u_9	f_9

From this example, it is easy to catch the block structure of the matrix A, where each block is $(J-1)\times(J-1)$:

$$A = \frac{1}{h^2}\begin{bmatrix} T & I & & \\ I & T & I & \\ & \ddots & \ddots & \ddots \\ & & I & T \end{bmatrix}, \quad \text{where} \quad T = \begin{bmatrix} -4 & 1 & & \\ 1 & -4 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -4 \end{bmatrix}, \tag{30}$$

and I is the $(J-1)\times(J-1)$ identity matrix.

Matlab has two types of matrices: full, where all entries are kept in memory, and sparse, where only nonzero entries and their indices are kept in memory. Each row of A in Eq. (30) has at most 5 nonzero entries, hence it is worthwhile to set it up as sparse. Note that if you set it up as full, Matlab will be able to solve your system for at most $J = 150$ or so depending on your computer. It can handle much larger values of J if you set up A as sparse. Look how A is set up in the code below. The command `kron(A,B)` gives the Kronecker product of matrices A and B. The code below solves Eq. (27) with $f(x,y) = \sin(2\pi x)\sin(2\pi y)$ for $J = 2^k$, $k \in \{3,4,\dots,10\}$. The exact solution is $u_{exact} = -f(x,y)/(8\pi^2)$. At the end, it finds the constant C and the power q for the error estimate of the form: $er = Ch^q$: $C = 0.04274$, $q = 2.0045$. The plots of the solution and the error distribution for $J = 1024$ are shown in Figs. 2(a,b). The plot of the maximal norm error versus h in the log-log scale is shown in fig. 2(c).

```
function poisson()
for k = 1 : 8
    n = 2^(k + 2) + 1;
    n2 = n - 2;
    t = linspace(0,1,n);
```

```
    [x,y] = meshgrid(t,t);
    f = sin(2*pi*x).*sin(2*pi*y);
    f1 = f(2 : n - 1,2 : n - 1);
    f_aux = f1(:);
    h(k) = 1/(n - 1);
    u = zeros(n);

    % Set up the matrix A
    I = speye(n2); % n2-by-n2 sparse identity matrix
    e = ones(n2,1);
    T = spdiags([e -4*e e],[-1:1],n2,n2); % set up sparse T as in Eq. 27
    S = spdiags([e e],[-1 1],n2,n2); % set up sparse S with ones along
    % the first sub- and super-diagonal
    A = (kron(I,T) + kron(S,I))/h(k)^2; % kron is the Kronecker product

    % Solve the linear system
    u_aux = A\f_aux;
    u(2:n-1,2:n-1) = reshape(u_aux,n2,n2);

    u_exact = -f/(8*pi^2);

    er(k) = max(max(abs(u - u_exact)));
end

% plot the solution
figure(1);
clf; hold on; grid;
ma = max(max(u));
mi = min(min(u));
contourf(x,y,u,linspace(mi,ma,20));

% plot the error
figure(2);
clf; hold on; grid;
imagesc(t,t,u - u_exact);

% find C and q for the error estimate of the form er = C*h^q
figure(3);
clf; hold on; grid;
plot(h,er,'.','Markersize',20);
plot(h,er);
set(gca,'XScale','log','YScale','log');
p = polyfit(log(h),log(er),1);
```

```
fprintf('Error(h) = (%d)*h^(%d)\n',exp(p(2)),p(1));
end
```

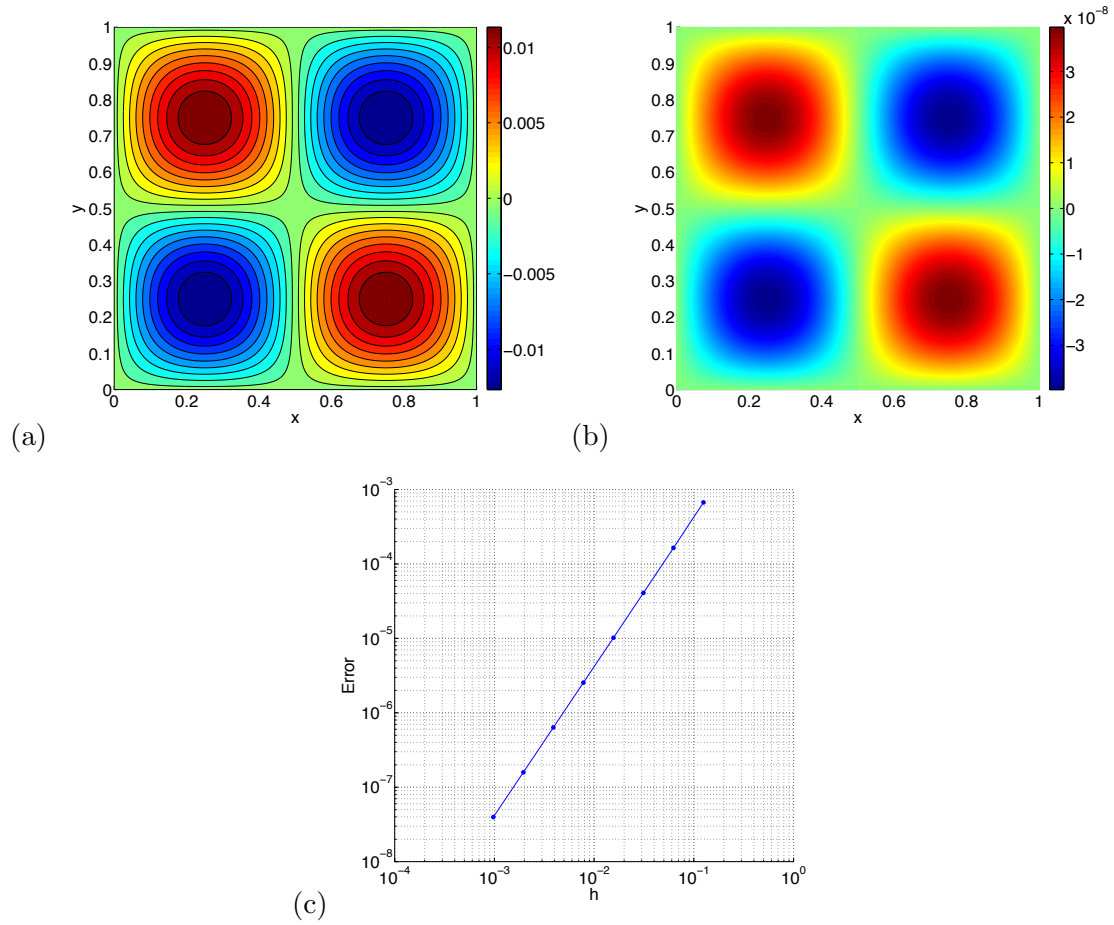


FIGURE 2. Illustration to the numerical solution of Eq. (27) with $f(x, y) = \sin(2\pi x)\sin(2\pi y)$. (a) The numerical solution on 1025×1025 mesh. (b) The error distribution of the numerical solution on 1025×1025 mesh. (c) The maximal norm error $\max_{i,j} |u_{ij} - u_{ij,exact}|$ versus h.

2.2. **Error Analysis.** In this section, we will denote the exact solution by u and the numerical solution by U. Substituting the exact solution u into the finite difference Eq. (29) and Taylor expanding u at the point P we obtain the *truncation error*

$$T_p := \frac{1}{h^2}(u_N + u_S + u_W + u_E - 4u_P) - f_P = \frac{h^2}{12}(u_{xxxx} + u_{yyyy})_P + o(h^2) \tag{31}$$

Now we define the discrete differential operator

$$(L_h U)_P \equiv L_h U_P := \frac{1}{h^2}\left(U_N + U_S + U_W + U_E - 4U_P\right) \tag{32}$$

at all interior mesh points P. The set of all mesh points will be denoted by Ω_h. Then for the numerical and the exact solutions we have:

$$L_h U_P - f_P = 0, \tag{33}$$

$$L_h u_P - f_P = T_P. \tag{34}$$

Subtracting Eq. (34) from Eq. (33) we obtain the equation for the error $e_P := U_P - u_P$:

$$L_h e_P = -T_P. \tag{35}$$

Note that e_P is zero at all boundary points.

To obtain a bound on e_P we will use a *comparison function* and a *discrete maximum principle*. The comparison function is chosen to be quadratic so that the discrete differential operator is exact on it (see Eq. (31)) and constant at every point:

$$\Phi_P := (x_P - \tfrac{1}{2})^2 + (y_P - \tfrac{1}{2})^2. \tag{36}$$

Then

$$L_h \Phi_P = 4. \tag{37}$$

The discrete maximum principle for the operator L_h defined in Eq. (32) is given in the following lemma.

Lemma 1. A Discrete Maximum Principle

Let L_h be the discrete differential operator defined in Eq. (32).

(1) *Let ψ_P be a mesh function defined on Ω_h such that*

$$L_h \psi_P \geq 0 \quad \textit{for all} \quad P \in \Omega_h. \tag{38}$$

Then

$$\max_{P \in \Omega_h} \psi_P = \max_{P \in \partial\Omega_h} \psi_P. \tag{39}$$

(2) *Let ψ_P be a mesh function defined on Ω_h such that*

$$L_h \psi_P \leq 0 \quad \textit{for all} \quad P \in \Omega_h. \tag{40}$$

Then

$$\min_{P \in \Omega_h} \psi_P = \min_{P \in \partial\Omega_h} \psi_P. \tag{41}$$

Proof. We prove Statement (1). Statement (2) is proven similarly. Assume that P is an interior point of Ω_h and $\psi_P > \max\{\psi_N, \psi_S, \psi_W, \psi_E\}$. Then

$$L_h \psi_P = h^{-2}(\psi_N + \psi_S + \psi_W + \psi_E - 4\psi_P) < 0,$$

which is in a contradiction with Eq. (38). Hence, either every interior point has at least one nearest neighbor with a greater value of ψ, or $\psi_P = \psi_N = \psi_S = \psi_E = \psi_W$. This implies Eq. (39). □

Now we define a function

$$\psi_P := e_P + \tfrac{1}{4}T\Phi_P, \quad \text{where} \quad T := \max_{p\in\Omega_h} |T_P|.$$

Then ψ satisfies Eq. (38):

$$L_h\psi_P = -T_P + T \geq 0 \quad \text{for all} \quad P \in \Omega_h.$$

By Lemma 1 part (1), the maximum of ψ_h is achieved on the mesh boundary $\partial\Omega_h$. Since $e_P = 0$ on $\partial\Omega_h$, this maximum is equal to

$$\max_{P\in\partial\Omega_h} \tfrac{1}{4}T\Psi_P = \tfrac{1}{8}T.$$

Hence

$$e_P \leq e_P + \tfrac{1}{4}T\Phi_P \equiv \psi_P \leq \tfrac{1}{8}T. \tag{42}$$

Similarly, we define a function

$$\psi_P = e_P - \tfrac{1}{4}T\Phi_P, \quad \text{where} \quad T := \max_{p\in\Omega_h} |T_P|.$$

Then ψ satisfies Eq. (40):

$$L_h\psi_P = -T_P - T \leq 0 \quad \text{for all} \quad P \in \Omega_h.$$

By Lemma 1 part (2), the minimum of ψ_h is achieved on the mesh boundary $\partial\Omega_h$. Since $e_P = 0$ on $\partial\Omega_h$, this minimum is equal to

$$\min_{P\in\partial\Omega_h} -\tfrac{1}{4}T\Psi_P = -\tfrac{1}{8}T.$$

Hence

$$e_P \geq e_P - \tfrac{1}{4}T\Phi_P \equiv \psi_P \geq -\tfrac{1}{8}T. \tag{43}$$

Combining Eqs. (42) and (43) and decoding T we get the following error bound:

$$\boxed{|e_P| \leq \frac{h^2}{96}\left(\max_{(x,y)\in\Omega} |u_{xxxx}| + \max_{(x,y)\in\Omega} |u_{yyyy}|\right).} \tag{44}$$

Eq. (44) shows that the error is proportional to h^2 and the constant is largely determined by fourth derivatives of the solution.

3. Handling different types of boundary conditions and non-constant coefficients

3.1. Homogeneous Neumann and Nonhomogeneus Dirichlet boundary conditions. Consider the BVP for Laplace's equation

$$u_{xx} + u_{yy} = 0, \quad (x, y) \in \Omega = (0, 1) \times (0, 1), \tag{45}$$

$$u(0, y) = 1, \quad u(1, y) = 0, \tag{46}$$

$$u_y(x, 0) = u_y(x, 1) = 0. \tag{47}$$

Exercise Show that the solution of the BVP (45)-(47) is $u(x, y) = 1 - x$.

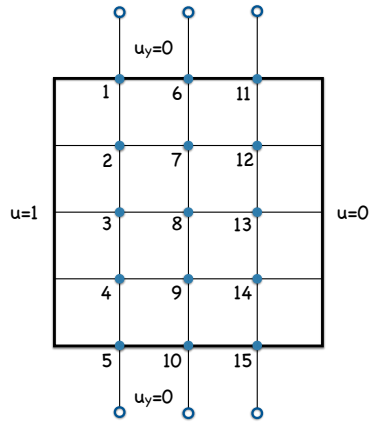


FIGURE 3. The method of "ghost" points for handling the homogeneous Neumann boundary conditions.

Set up a mesh as shown in Fig. 3. The numbered points marked by shaded circles are those where we need to compute the solution. The points marked by transparent circles are the so called "ghost points". We do not add them to the mesh but imagine them in order to write the finite difference equations for the top and bottom boundary mesh points. For the boundary mesh points lying on the top and on the bottom (points 1, 6, 11, and 5, 10, 15) Northern and Southern neighbors are absent respectively. We imagine ghost points in places of these absent neighbors and write

$$\frac{1}{h^2}\left(u_{N,i}+u_{S,i}+u_{E,i}+u_{W,i}-4u_{P,i}\right)=0, \quad i=1,6,11,5,10,15.$$

For points 1, 6, and 11, $u_{N,i}$s are the ghost points. The BCs

$$u_y(x,0)=0\approx\tfrac{1}{2h}(u_{S,i}-u_{N,i})$$

imply that $u_{N,i}\approx u_{S,i}$. Motivated by that we set $u_{N,i}=u_{S,i}$ and get

$$\frac{1}{h^2}\left(2u_{S,i}+u_{E,i}+u_{W,i}-4u_{P,i}\right)=0, \quad i=1,6,11.$$

Furthermore, for point 1, $u_{W,1}=1$. Hence

$$\frac{1}{h^2}\left(2u_{S,1}+1+u_{E,1}-4u_{P,1}\right)=0,$$

which is equivalent to

$$\frac{1}{h^2}\left(2u_{S,1}+u_{E,1}-4u_{P,1}\right)=-\frac{1}{h^2}.$$

For point 11, $u_{E,11}=0$. Hence

$$\frac{1}{h^2}\left(2u_{S,11}+u_{W,11}-4u_{P,11}\right)=0.$$

Equations for points 5, 10, 15 are derived in a similar manner.

Exercise Write explicitly the system of linear algebraic equations of the form $Au = f$ for this problem (see HW1, Problem 1(a)).

3.2. **Example: finding the electric potential on a plate with varying conductivity.** Imagine a conducting plate $[0,1] \times [0,1]$ with conductivity distribution

$$a(x,y) = 2.1 + \sin(2\pi x) + \cos(3\pi y)$$

(see Fig. 4(a)) bent into a cylinder around the y-axis. The unit voltage is applied between

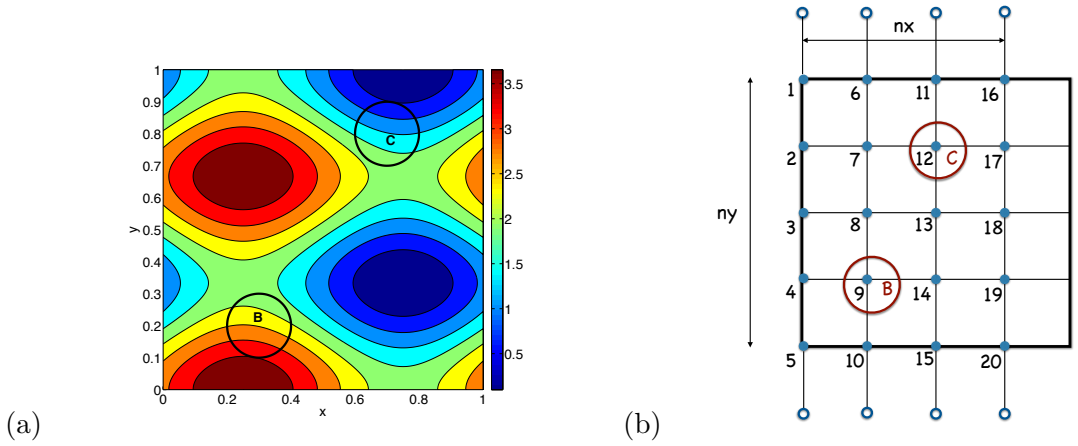


FIGURE 4. (a): The conductivity $a(x,y)$ of the conducting plate. (b): The mesh.

the regions $B := \{(x,y) \mid (x-0.3)^2 + (y-0.2)^2 \le 0.1^2\}$ and $C := \{(x,y) \mid (x-0.7)^2 + (y-0.8)^2 \le 0.1^2\}$. We can assume that the potential in B is 0, while it is 1 in C. We need to find the distributions of the electric potential $u(x,y)$. The electric current is defined according to Ohm's law

$$j(x,y) := -a(x,y)\nabla u(x,y).$$

We pose the following BVP. According to Ampere's law,

$$\nabla \cdot (a(x,y)\nabla u) = 0. \tag{48}$$

The boundary conditions are periodic in x, i.e.

$$u(0,y) = u(1,y), \quad u_x(0,y) = u_x(1,y). \tag{49}$$

Also, since there is no current through the top and bottom boundaries of the plate, we impose the homogeneous Neumann on them:

$$u_y(x,0) = u_y(x,1) = 0. \tag{50}$$

Furthermore, the applied voltage can be accounted for by the following boundary conditions:

$$u = 0, \quad (x,y) \in B, \quad u = 1, \quad (x,y) \in C. \tag{51}$$

Now we discuss how one can solve the BVP given by Eqs. (48)-(51) in Matlab. First we set up an $ny \times nx$ mesh with steps and hy and hx and number the points with unknown values of u column-wise. For now, we ignore the regions B and C. We include the points on the left, top, and bottom boundaries and do not include those on the right (see Fig. 4(b)). We discretize Eq. (48) using the following finite difference scheme:

$$\frac{a_s \frac{u_S - u_P}{hy} - a_n \frac{u_P - u_N}{hy}}{hy} + \frac{a_e \frac{u_E - u_P}{hx} - a_w \frac{u_P - u_W}{hx}}{hx} = 0, \tag{52}$$

where a_n, a_s, a_e, and a_w are the values of a at the mid-mesh points as shown in Fig. 5. It is convenient to approximate these values as

$$a_n = \tfrac{1}{2}(a_P + a_N),\ a_s = \tfrac{1}{2}(a_P + a_S),\ a_e = \tfrac{1}{2}(a_P + a_E),\ a_w = \tfrac{1}{2}(a_P + a_W).$$

Reorganizing terms in Eq. (52) we get:

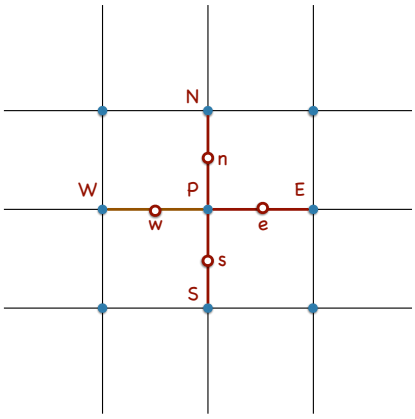


FIGURE 5. The stencil for the difference scheme (52).

$$c_S u_S + c_N u_N + c_E u_E + c_W u_W + c_P u_P = 0. \tag{53}$$

where

$$c_S := \frac{a_s}{hy^2},\ c_N := \frac{a_n}{hy^2},\ c_E := \frac{a_e}{hx^2},\ c_W := \frac{a_w}{hx^2},$$
$$c_P := -\left(\frac{a_n + a_s}{hy^2} + \frac{a_e + a_w}{hx^2}\right).$$

Now we start writing a matlab code for solving Eqs. (48)-(51) on $n \times n$ mesh. First we set up the mesh.

```
n = 201;
xaux = linspace(0,1,n);
[x,y] = meshgrid(xaux,xaux);
```

```
u = zeros(n);
ny = n;
nx = n - 1;
hx = 1/nx;
hy = 1/(ny - 1);
aaux = 2.1 + sin(2*pi*x) + cos(3*pi*y);
a = aaux(:,1 : nx);
nxy = nx*ny;
```

Now we set up the matrices of coefficients for the difference scheme (53) and convert them into column vectors.

```
cN = 0.5*(a + circshift(a,[1 0]))/(hy^2);
cS = 0.5*(a + circshift(a,[-1 0]))/(hy^2);
cW = 0.5*(a + circshift(a,[0 1]))/(hx^2);
cE = 0.5*(a + circshift(a,[0 -1]))/(hx^2);
cP = -(cE + cW + cN + cS);
% Conversion to column vectors
cn = cN(:); cs = cS(:); cw = cW(:); ce = cE(:); cp = cP(:);
```

As in the example in Section 2.1, we first write out the matrix A of the linear system to be solved on a small mesh. This allows us to understand its sparsity pattern and extend it for any mesh. We get that A consists of $nx \times nx$ blocks each of which is $ny \times ny$:

$$A = \begin{bmatrix} T_1 & E_1 & 0 & \dots & 0 & W_1 \\ W_2 & T_2 & E_2 & \dots & 0 & 0 \\ & \ddots & \ddots & \ddots & & \\ & \dots & 0 & W_{nx-1} & T_{nx-1} & E_{nx-1} \\ E_{nx} & 0 & \dots & 0 & W_{nx} & T_{nx} \end{bmatrix},$$

where T_k are tridiagonal matrices of the form

$$T_k = \begin{bmatrix} c_{P,k_1} & 2c_{S,k_1} & & & \\ c_{N,k_2} & c_{P,k_2} & c_{S,k_2} & & \\ \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & c_{N,k_{ny-1}} & c_{P,k_{ny-1}} & c_{S,k_{ny-1}} & \\ & & 2c_{N,k_{ny}} & c_{P,k_{ny}} & \end{bmatrix},$$

$k_i = (k-1) \cdot ny + i$, $i = 1, \dots, ny$, and $C_{W,k}$ and $C_{E,k}$ are diagonal matrices

$$E_k = \mathrm{diag}\{c_{E,k_1}, \dots, c_{E,k_{ny}}\}, \quad W_k = \mathrm{diag}\{c_{W,k_1}, \dots, c_{W,k_{ny}}\}.$$

To set up the matrix A we will use the commands `circshift` and `spdiags`. The following example demonstrate how they work.

```
>> a = [1 2 3 4]'
```

```
a =

     1
     2
     3
     4

>> circshift(a,[1 0])

ans =

     4
     1
     2
     3

>> circshift(a,[-1 0])

ans =

     2
     3
     4
     1

>> b=spdiags([a a a a a],[-2 : 2],4, 4)

b =

   (1,1)        1
   (2,1)        1
   (3,1)        1
   (1,2)        2
   (2,2)        2
   (3,2)        2
   (4,2)        2
   (1,3)        3
   (2,3)        3
   (3,3)        3
   (4,3)        3
   (2,4)        4
   (3,4)        4
   (4,4)        4
```

```
>> bfull = full(b)

b =

     1     2     3     0
     1     2     3     4
     1     2     3     4
     0     2     3     4
```

We see that `spdiags` places the same indices of the diagonal vectors in the same column. We will need to set up A with coefficients of the same indices in each row. We can do it with the aid of `circshift`:

```
>> b=spdiags([circshift(a,[-2,0]),circshift(a,[-1,0]),a,...
              circshift(a,[1,0]),circshift(a,[2,0])],[-2 : 2],4, 4)

b =

   (1,1)        1
   (2,1)        2
   (3,1)        3
   (1,2)        1
   (2,2)        2
   (3,2)        3
   (4,2)        4
   (1,3)        1
   (2,3)        2
   (3,3)        3
   (4,3)        4
   (2,4)        2
   (3,4)        3
   (4,4)        4

>> b = full(b)

b =

     1     1     1     0
     2     2     2     2
     3     3     3     3
     0     4     4     4
```

Now we are ready to set up the matrix A.

```
A = spdiags([circshift(cw,[-ny 0]),circshift(cn,[-1 0]),cp,...
    circshift(cs,[1 0]),circshift(ce,[ny 0])],[-ny,-1,0,1,ny],nxy,nxy);
% Take care of the Neumann BCs: u_y(x,0) = u_y(x,1) = 0
for j = 1 : nx - 1
    A(ny*j,ny*j - 1) = 2*A(ny*j,ny*j - 1) ;
    A(ny*j,ny*j + 1) = 0;
    A(ny*j + 1,ny*j + 2) = 2*A(ny*j + 1,ny*j + 2);
    A(ny*j + 1,ny*j) = 0;
end
A(1,2) = 2*A(1,2);
A(nxy,nxy - 1) = 2*A(nxy,nxy - 1) ;
% Take care of the periodic BCs: u(0,y) = u(1,y), u_x(0,y) = u_x(1,y);
A1 = spdiags(circshift(ce,[-nxy + ny, 0]),-nxy + ny,nxy,nxy);
A2 = spdiags(circshift(cw,[nxy - ny, 0]),nxy - ny,nxy,nxy);
A = A + A1 + A2;
```

Now it remains to take care of the regions B and C. We will split the set of indices of the mesh points into three subsets: `iB` and `iC` are the indices of the mesh points lying in B and C respectively, and `inBC` are the indices of the rest of the mesh points.

```
% Regions B and C are balls centered at (x1,y1) and (x2,y2) of radii r1 and
% r2 respectively
x1 = 0.3;
y1 = 0.2;
r1 = 0.1;
x2 = 0.7;
y2 = 0.8;
r2 = 0.1;
iB = find((x - x1).^2 + (y - y1).^2 < r1^2);
iC = find((x - x2).^2 + (y - y2).^2 < r2^2);

% Set the RHS due to the BC's at C
uc = zeros(nxy,1);
uc(iC) = 1;
rhs = -A*uc;
% Remove rows and columns corresponding to iBC
iBC = [iB;iC];
inBC = [1 : nxy]';
inBC(iBC) = [];
A(:,iBC) = [];
A(iBC,:) = [];
rhs(iBC) = [];
```

In the rest of the code we solve the system `Au=rhs`, place the solution into the original square domain $[0,1]^2$, and visualize the conductivity $a(x,y)$, the solution, i.e., the electric

potential $u(x, y)$, and the absolute value and the direction of the electric current $j(x, y) = a(x, y)\nabla u(x, y)$.

```
% Solution
uaux = A\rhs;
ucol = zeros(nxy,1);
ucol(inBC) = uaux;
ucol(iC) = 1;
ucol(iB) = 0;
u1 = reshape(ucol,ny,nx);
u = zeros(ny,nx + 1);
u(:,1 : nx) = u1;
u(:,nx + 1) = u1(:,1);

% Plot the conductivity, the solution, and the current;
figure(1);
clf;
hold on;
ma = max(max(aaux));
mi = min(min(aaux));
contourf(xaux,xaux,aaux,linspace(mi,ma,10));
set(gca,'DataAspectRatio',[1,1,1],'Fontsize',24);
xlabel('x','Fontsize',24);
ylabel('y','Fontsize',24);
%
figure(2);
clf;
hold on;
contourf(xaux,xaux,u,[0:0.1:1]);
t = linspace(0,2*pi,100);
plot(x1 + r1*cos(t),y1 + r1*sin(t),'w','Linewidth',2);
plot(x2 + r2*cos(t),y2 + r2*sin(t),'w','Linewidth',2);
set(gca,'DataAspectRatio',[1,1,1],'Fontsize',24);
xlabel('x','Fontsize',24);
ylabel('y','Fontsize',24);
%
figure(3);
clf;
hold on;
curx = zeros(ny,nx + 1);
cury = zeros(ny,nx + 1);
curx(:,1 : nx) = a.*(0.5*(circshift(u1,[0 -1]) - circshift(u1,[0 1]))/hx);
cury(:,1 : nx) = a.*(0.5*(circshift(u1,[-1 0]) - circshift(u1,[1 0]))/hy);
```

```
curx(:,nx + 1) = curx(:,1);
cury(:,nx + 1) = cury(:,1);
cury(1,:) = 0;
cury(ny,:) = 0;
curx = -curx;
cury = -cury;
ecur = sqrt(curx.^2 + cury.^2);
ecmax = max(max(ecur));
contourf(xaux,xaux,ecur,linspace(0,ecmax,10));
plot(x1 + r1*cos(t),y1 + r1*sin(t),'w','Linewidth',2);
plot(x2 + r2*cos(t),y2 + r2*sin(t),'w','Linewidth',2);
ix = [1 : 8 : nx];
iy = [1 : 8 : ny];
ac = sqrt(curx(iy,ix).^2 + cury(iy,ix).^2 + 1e-12);
quiver(x(iy,ix),y(iy,ix),curx(iy,ix)./ac,cury(iy,ix)./ac,'color','w')
contour(xaux,xaux,aaux,linspace(mi,ma,10),'color','k');
set(gca,'DataAspectRatio',[1,1,1],'Fontsize',24);
xlabel('x','Fontsize',24);
ylabel('y','Fontsize',24);
```

The results are shown in Fig. 6. Note that while the solution $u(x, y)$ looks smooth, there

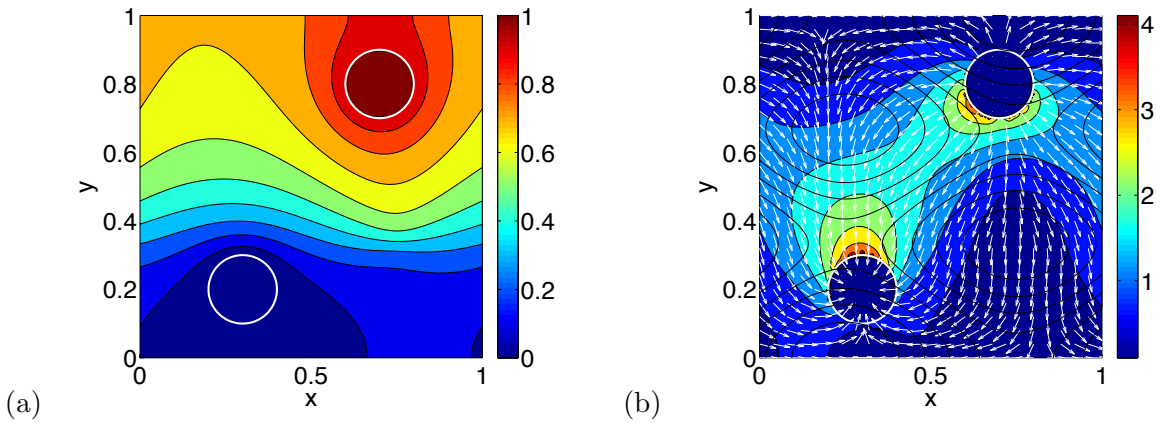


FIGURE 6. (a): The computed electric potential $u(x, y)$. (b): The colored shows the absolute value of the electric current, while the arrows indicate its direction.

are visible artifacts near the boundaries of B and C. They are due to the fact that we approximated the boundaries of B and C by non smooth curves. To cure this problem, one can set up a finite difference scheme for curvy boundaries as is explained in [3] or use the *finite element method* which handles curvy boundaries more naturally.

4. Finite Element Method (FEM)

4.1. **An illustration of the general idea of the FEM on a model problem.** A very good introduction into the FEM and its implementation is given in [7]. In these notes, I will provide some comments for [7]. We consider a model BVP for the Poisson equation in a bounded domain $\Omega \subset \mathbb{R}^2$ with the boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$. We assume that Γ_D is some subset of connected components of $\partial\Omega$, and so is Γ_N.

$$-\Delta u = f(x,y),\ (x,y) \in \Omega, \tag{54}$$

$$u = \tilde{u}_D,\ (x,y) \in \Gamma_D, \tag{55}$$

$$\frac{\partial u}{\partial n} = g,\ (x,y) \in \Gamma_N, \tag{56}$$

where n is the unit outer normal to the boundary Γ_N (see Fig. 7). The first step in the

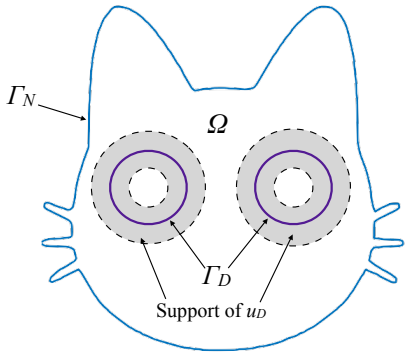


Figure 7. The boundary Γ_N is the outer contour of the cat's head. The boundary Γ_D consists of two connected components, the purple circles (the cat's eyes). The shaded areas around the eyes constitute the support of the function u_D. The domain Ω is the region between the outer contour Γ_N and the eye contours Γ_D.

FEM is to reduce the BVP for a PDE to an integral equation (Eq. (4) at the end of Section 2 in [7]). Let us understand how it is done. First, we will get rid of the non-homogeneous Dirichlet boundary conditions Eq. (55). We define a function u_D such that $u_D = \tilde{u}_D$ on Γ_D, u_D is twice continuously differentiable, and u_D is zero outside a small neighborhood $\mathcal{U}(\Gamma_D)$ of Γ_D, such that $\mathcal{U}(\Gamma_D) \cap \Gamma_N = \emptyset$. For example, one can construct u_D using linear interpolation and mollification (for mollification, see e.g. [4] Appendix C.4). Then, we do a variable change:

$$v := u - u_D. \tag{57}$$

The BVP for v becomes

$$-\Delta v = f(x,y) + \Delta u_D,\ (x,y) \in \Omega, \tag{58}$$

$$v = 0,\ (x,y) \in \Gamma_D, \tag{59}$$

$$\frac{\partial v}{\partial n} = g,\ (x,y) \in \Gamma_N, \tag{60}$$

Let v be a solution of the BVP (58)-(60). Let us multiply Eq. (58) by an arbitrary twice continuously differentiable in Ω function w such that $w = 0$ on Γ_D, and integrate the result over Ω. The result is:

$$-\int_\Omega w\Delta v dx = \int_\Omega w f dx + \int_\Omega w\Delta u_D dx. \tag{61}$$

In order to get rid of the 2nd derivatives in Eq. (61) and incorporate boundary conditions we use the First Green's Identity: for any region $\Omega \subset \mathbb{R}^n$ with a piecewise smooth boundary, and any functions $\phi, \psi : \Omega \to \mathbb{R}^n$, such that ϕ is $C^2(\Omega)$ (twice continuously differentiable in Ω) and ψ is $C^1(\Omega)$ (continuously differentiable in Ω),

$$\int_\Omega (\psi\Delta\phi + \nabla\psi\cdot\nabla\phi)dx = \int_{\partial\Omega} \psi\frac{\partial\phi}{\partial n} ds. \tag{62}$$

Eq. (62) is equivalent to Eq. (63):

$$\int_\Omega \nabla\psi\cdot\nabla\phi dx = -\int_\Omega \psi\Delta\phi dx + \int_{\partial\Omega} \psi\frac{\partial\phi}{\partial n} ds. \tag{63}$$

Replacing ψ with w and ϕ with v in Eq. (63) and using Eqs. (61) and (59) we get

$$\int_\Omega \nabla w\cdot\nabla v dx = \int_\Omega w\Delta u_D dx + \int_\Omega w f dx + \int_{\Gamma_N} w g ds + \int_{\Gamma_D} w\frac{\partial v}{\partial n} ds. \tag{64}$$

The last term in Eq. (64) vanishes due to our assumption that $w = 0$ on Γ_D. By our construction, u_D is identically zero outside a small neighborhood of Γ_D, hence $\frac{\partial u_D}{\partial n} = 0$ on Γ_N. Therefore, plugging in u_D for ϕ and w for ψ in Eq. (63) we get

$$\int_\Omega w\Delta u_D dx = -\int_\Omega \nabla w\cdot\nabla u_D dx. \tag{65}$$

Combining Eqs. (64) and (65) we obtain the integral equation we wanted,

$$\boxed{\int_\Omega \nabla w\cdot\nabla v dx = -\int_\Omega \nabla w\cdot\nabla u_D dx + \int_\Omega w f dx + \int_{\Gamma_N} w g ds,} \tag{66}$$

that holds for any solution v of the BVP (58)-(60) and any function $w \in C^1(\Omega)$ satisfying $w = 0$ on Γ_D.

Next, we want to solve the integral equation (66). It is hard to find its exact solution. The central idea of the FEM is to use Galerkin's trick: instead of solving Eq. (66) exactly, find its approximate solution in some finite dimensional subspace spanned by a basis consisting of piece-wise linear functions $\{\eta_j(x,y)\}_{j=1}^N$ with compact support (i.e., zero outside some compact set). This idea poses immediate problem: piecewise-linear functions are not

differentiable, which means that the gradient of $v(x,y) = \sum_j c_j \eta_j(x,y)$ will not be well-defined on some subset of Ω. To get around this difficulty, we need to broaden the concept of the derivative by introducing so-called weak derivative. In [7], the functions v and w are assumed to be in the Sobolev space $H_D^1(\Omega)$ specially for for this reason. We discuss what is a weak derivative in Section 4.1.1.

4.1.1. *Weak derivatives and Sobolev spaces.*

Example 1 Consider the piecewise linear function

$$f(x) = \max\{0, 1 - |x|\}, \quad -\infty < x < \infty.$$

Its graph is shown in Fig. 8 (Top). The classic derivative of $f(x)$ is undefined at 0 and ± 1. In addition, consider the function

$$\phi(x) = \begin{cases} 1, & -1 < x < 0, \\ -1, & 0 < x < 1, \\ 0, & \text{otherwise} \end{cases}.$$

Its graph is shown in Fig. 8 (Bottom). Note that $\phi(x)$ is the derivative of $f(x)$ on $\mathbb{R}\backslash\{-1, 0, 1\}$, i.e., everywhere except for a measure zero subset of points.

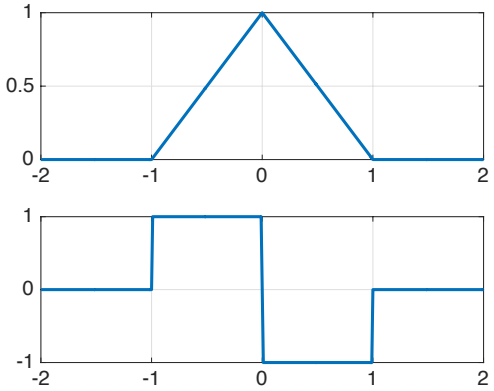


FIGURE 8. An illustration for Example 1.Top: the piecewise function $f(x) = \max\{0, 1 - |x|\}$. Bottom: the piecewise continuous function $\phi(x)$ which is a weak derivative of $f(x)$.

We want to define weak derivatives so that the function ϕ is a weak derivative of $f(x)$ in Example 1. More generally, we want ϕ to be a weak partial derivative of $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ if f is piecewise differentiable, and ϕ is its classic partial derivative everywhere in Ω except for maybe a subset of measure zero. The construction of the definition of weak derivatives involves so called test functions, i.e., functions that you can multiply by any integrable

function and still obtain an integrable function, and that you can differentiate as many times as you wish. These considerations motivate the following definition of a test function.

Definition 1. *Let $\Omega \subset \mathbb{R}^n$ be an open set. A function $\phi : \Omega \to \mathbb{R}$ is a test function if $\phi \in C^\infty(\Omega)$ (i.e., is infinitely differentiable), and has a compact support in Ω, i.e., ϕ is nonzero only within some the interior of some compact set lying within Ω. The set where $\phi \neq 0$ is called the support of ϕ and is denoted by* $\mathrm{supp}(\phi)$. *Note that if Ω has a boundary, then $\phi = 0$ on $\partial\Omega$.*

Let $f(x)$ be continuously differentiable in Ω. Then for any test function $\phi(x)$ the integration by parts formula gives

$$\int_\Omega \frac{\partial f(x)}{\partial x_i}\phi(x)dx = -\int_\Omega f(x)\frac{\partial \phi(x)}{\partial x_i}dx.$$

The identity above is used to define the weak partial derivative.

Definition 2. *The function $g(x)$ is the weak partial derivative of the function $f(x)$ with respect to the variable x_i if for any test function ϕ we have*

$$\int_\Omega g(x)\phi(x)dx = -\int_\Omega f(x)\frac{\partial \phi(x)}{\partial x_i}dx. \tag{67}$$

We will abuse notations and denote weak derivatives the same as we denote the classic ones. In view of Definition 2, a weak gradient of f, that we denote simply by ∇f, satisfies

$$\int_\Omega \nabla f(x)\phi(x)dx = -\int_\Omega f(x)\nabla\phi(x)dx. \tag{68}$$

The functional space $H^1(\Omega)$ is an instance of Sobolev spaces (see e.g. Wikipedia).

$$H^1(\Omega) := \left\{ f : \Omega \to \mathbb{R} \mid \int_\Omega |f|^2 dx < \infty,\ \int_\Omega |\nabla f|^2 dx < \infty \right\}, \tag{69}$$

where ∇f is a weak gradient of f. $H^1(\Omega)$ is a normed space:

$$\|f\|_{H^1} = \left[\int_\Omega |f|^2 dx + \int_\Omega |\nabla f|^2 dx\right]^{1/2}. \tag{70}$$

Furthermore, H^1 is equipped with the following inner product:

$$(f,g)_{H^1} = \int_\Omega fg dx + \int_\Omega \nabla f \cdot \nabla g dx. \tag{71}$$

Note that, in particular, H^1 contains all continuous and piecewise continuously differentiable functions $\Omega \to \mathbb{R}$.

The space $H^1_D(\Omega)$ consists of all functions of $H^1(\Omega)$ that vanish on the curve Γ_D.

4.2. **Mesh generation.** Suppose we are given a set of points $\{x_i\}_{i=1}^N$ and we want to triangulate it. For the stiffness matrix to be well-conditioned, it is important to define triangulation that avoids small angles in the triangles. A Delaunay triangulation (named after the 1934 work of Boris Delaunay) of the set of points $\{x_i\}_{i=1}^N$ is such a triangulation that none of the points $\{x_i\}_{i=1}^N$ lies in ant circumcircle of any triangle. Delaunay triangulations maximize the minimal angle in the triangles. The Wikipedia article Delaunay triangulation contains a comprehensive overview of algorithms for obtaining Delaunay triangulations. Matlab offers the command `delaunay` for generating a Delaunay triangulation of a given set of points, and a number of commands for its visualization, for example `triplot`. Here is an example of their use.

```
 y = linspace(0,1,10);
 e = ones(1,10);
x = rand(100,2); % inner points
% concatenate the sets of inner and boundary points
z = [x;[y',0*e'];[e',y'];[y',e'];[0*e',y']];
zu = unique(z,'rows'); % remove repeating points
tri = delaunay(zu(:,1),zu(:,2));
% the rows tri are the triplets of indices of vertices of the triangles
 figure;
triplot(tri,zu(:,1),zu(:,2)); % visualize the triangulation
daspect([1 1 1]); % set equal scales in the x- and y- axes
```

The obtained triangulation is shown in Fig. 9.

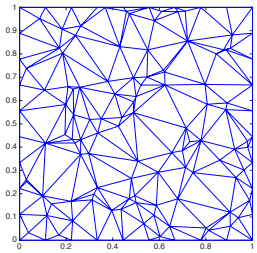


FIGURE 9. The delaunay triangulation for a random set of 100 points uniformly distributed in $[0,1]^2$ and 36 uniformly distributed boundary points produced by the Matlab script Section 4.2.

Thus, given a set of points, one can readily find a Delaunay triangulation. A harder question is how to pick the set of points for a given problem to be solved using FEM. For the general outline of the Delaunay triangulation and Matlab software overview, see the presentation by John Burkardt, CS, FSU.

Mesh generation packages.

(1) The `distmesh` package for 2D and 3D mesh generation developed by P.-O. Persson and G. Strang is available at http://persson.berkeley.edu/distmesh/. Their algorithm is described in the paper P.-O. Persson, G. Strang, A Simple Mesh Generator in MATLAB. SIAM Review, Volume 46 (2), pp. 329-345, June 2004. The code is written in Matlab. The advantage of this algorithm is its simplicity. The user can adjust it for his/her needs. Its shortcomings are that it requires an analytic function (desirably the signed distance function) whose zero-level set is the boundary of the region and that it is slow.
(2) The package `mesh2d` developed by Darren Engwirda is available on GitHub: https://github.com/dengwirda/mesh2d. This package is also written in Matlab. It is more complicated than `distmesh`, works with arbitrary contours, and is less transparent than `distmesh`.
(3) The package `Gmsh` developed by Christophe Geuzaine and Jean-Francois Remacle is suitable for using in C++, Python, or Julia: https://gmsh.info. Its shortcoming is that it is harder to use than `distmesh` and `mesh2d`. Its advantage is that it is a fast algorithm. The resulting mesh in 3D can be of rather poor quality.
(4) The package `tetgen` written in ANSI C++ is a 3D tetrahedral mesh generator developed by Hans Si https://wias-berlin.de/software/index.jsp?id=TetGen&lang=1. This is a fast mesh-generating algorithm, easier-to-use than `Gmsh`. The generated mesh is of moderate quality.

4.3. **Quadrature for FEM.** See the note by Shaozhong Deng, Dept. of Math. and Stat., UNCC.

4.4. **A simple basic FEM code with an example.** A simple 50-line matlab FEM routine supplemented with an example is written by Jochen Alberty, Carsten Carstensen and Stefan A. Funken. Their software is available at http://www.ii.uib.no/ sasha/I263/fem2d/. Change `.dat` to `.txt` extensions in the data files if you are a Mac user.

An example of use of this package an a triangular mesh generation using `mesh2d` is given in the code `myFEMcat.m`. Some simplifications were applied to data structure of the 50-line routine. All necessary functions except for `mesh2d` and the functions called by it are combined into this code. This code solves Laplace's equation $-\nabla^2 u = 0$ on the cat's head shaped domain with homogeneous Neumann boundary conditions on the outer boundary and Dirichlet conditions at the eyes: $u = 0$ at the left eye and $u = 1$ at the right eye. The mesh is generated in the function `triangulate_cat` that calls a sequence of functions from the `MESH2D` package by Darren Engwirda. The solution is shown in Fig. 10.

```
function MyFEMCat()
% % make mesh
[coordinates,elements3,c1,c2,c3] = triangulate_cat();
% coordinates is a N-by-2 array with coordinated of the mesh points
% elements3 is a Ntriag-by-3 array of indices of the triangular elements
```

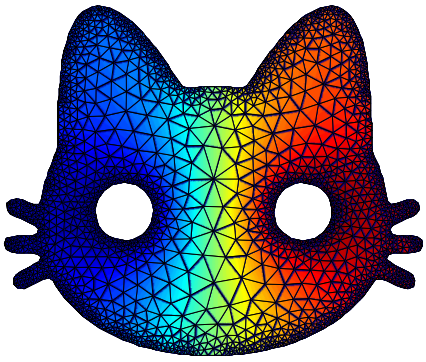


FIGURE 10. The solution Laplace's equation $-\nabla^2 u = 0$ on the cat's head shaped domain with homogeneous Neumann boundary conditions on the outer boundary and Dirichlet conditions at the eyes: $u = 0$ at the left eye and $u = 1$ at the right eye.

```
%%
% Find boundary points with homogeneous Neumann BCs
% The number of rows in neumann is the number of boundary intervals with
% Neumann BCs

% Each row of neumann contains indices of endpoints of the corresponding
% boundary interval
lc1 = length(c1)
k = 0;
for i = 1 : lc1
    j = find(coordinates(:,1) == c1(i,1) & coordinates(:,2) == c1(i,2));
    if ~isempty(j)
        k = k + 1;
        ind(k) = j;
    end
end
neumann = [ind',circshift(ind',[-1, 0])];

% Find boundary points with Dirichlet BCs
% dirichlet is a column vector of indices of points with Dirichlet BCs
lc2 = length(c2);
```

```
lc3 = length(c3);
k = 0;
for i = 1 : lc2
    j = find(coordinates(:,1) == c2(i,1) & coordinates(:,2) == c2(i,2));
    if ~isempty(j)
        k = k + 1;
        ind2(k) = j;
    end
end
dirichlet = ind2';
for i = 1 : lc3
    j = find(coordinates(:,1) == c3(i,1) & coordinates(:,2) == c3(i,2));
    if ~isempty(j)
        k = k + 1;
        dirichlet(k) = j;
    end
end

% call FEM
fem2d_2(coordinates,elements3,neumann,dirichlet);
end

%% fem2d_2
function fem2d_2(coordinates,elements3,neumann,dirichlet)
% mesh points with unknown values of u
FreeNodes = setdiff(1:size(coordinates,1),dirichlet);
A = sparse(size(coordinates,1),size(coordinates,1));
b = sparse(size(coordinates,1),1);

%% Assembly
%% The Stiffness matrix
for j = 1:size(elements3,1) % for all triangles
  A(elements3(j,:),elements3(j,:)) = A(elements3(j,:),elements3(j,:)) ...
      + stima3_2(coordinates(elements3(j,:),:));
  % stima3_2 computes M = 0.5*|T_j|*G*G';
end

%% The Right-hand side, i.e., the load vector
% Volume Forces
for j = 1:size(elements3,1)
  b(elements3(j,:)) = 0;  % for the case where f = 0
end
```

```
% Neumann conditions
for j = 1 : size(neumann,1)
  b(neumann(j,:)) = b(neumann(j,:)) + norm(coordinates(neumann(j,1),:)- ...
      coordinates(neumann(j,2),:)) * myg(sum(coordinates(neumann(j,:),:))/2)/2;
end

% Dirichlet conditions
u = sparse(size(coordinates,1),1);
u(dirichlet) = myu_d(coordinates(dirichlet,:));
b = b - A * u;

% Computation of the solution
u(FreeNodes) = A(FreeNodes,FreeNodes) \ b(FreeNodes);

% graphic representation
trisurf(elements3,coordinates(:,1),coordinates(:,2),full(u)','facecolor','interp')
hold on
axis ij
end

%%
function DirichletBoundaryValue = myu_d(x)
xmin = min(x(:,1));
xmax = max(x(:,1));
midx = 0.5*(xmin + xmax);
DirichletBoundaryValue =  0.5 * (sign(x(:,1) - midx) + 1);
end

%%
function Stress = myg(x)
Stress = zeros(size(x,1),1);
end

%%
function M = stima3_2(vertices)
d = size(vertices,2);
G = [ones(1,d+1);vertices'] \ [zeros(1,d);eye(d)];
M = det([ones(1,d+1);vertices']) * G * G' / prod(1:d);
end

%%
function [vert,tria,c1,c2,c3] = triangulate_cat()
c = imread('cat.png');
```

```
cc = sum(c,3);
h = contour(cc,[1 1]);
% extract contours of face and eyes
c1 = h(:,2:6:1682)';  % face
c2 = h(:,2884:6:3080)'; % eye
c3 = h(:,3082:6:3278)'; % another eye
lc1 = length(c1);
lc2 = length(c2);
lc3 = length(c3);
% connect boundary points
a1 = [1 : lc1]';
e1 = [a1, circshift(a1,[-1 0])]
a2 = [1 : lc2]';
e2 = [lc1 + a2, lc1 + circshift(a2,[-1 0])];
a3 = [1 : lc3]';
e3 = [lc1 + lc2 + a3, lc1 + lc2 + circshift(a3,[-1 0])];
node = [c1; c2; c3];
edge = [e1; e2; e3];


olfs.dhdx = +0.15;

[vlfs,tlfs, ...
hlfs] = lfshfn2(node,edge, ...
                []  ,olfs) ;

[slfs] = idxtri2(vlfs,tlfs) ;

hfun = @trihfn2;

[vert,edge,tria,tnum] = refine2(node,edge,[],[],hfun , ...
                vlfs,tlfs,slfs,hlfs);


figure;
patch('faces',tria(:,1:3),'vertices',vert, ...
    'facecolor','w', ...
    'edgecolor',[.2,.2,.2]) ;
hold on; axis image off;
axis ij

[vert,edge,tria,tnum] = smooth2(vert,edge,tria,tnum);
figure;
```

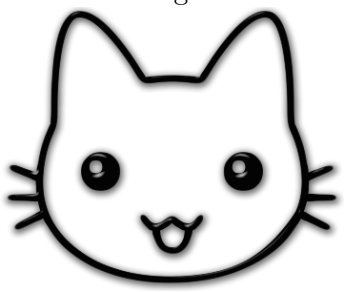
```
patch('faces',tria(:,1:3),'vertices',vert, ...
    'facecolor','w', ...
    'edgecolor',[.2,.2,.2]) ;
hold on; axis image off;
axis ij
end

end
```

4.4.1. *Extraction of contours.* The piece of the code `MyFEMCat.m`

```
c = imread('cat.png');
cc = sum(c,3);
h = contour(cc,[1 1]);
% extract contours of face and eyes
c1 = h(:,2:6:1682)';  % face
c2 = h(:,2884:6:3080)'; % eye
c3 = h(:,3082:6:3278)'; % another eye
```

reads the image of a cat that I have downloaded from the internet



and extracts three contours from it. In this Section, I will explain how I determined the numbers inside`h`. The variable `c` has size $m \times n \times 3$ where $m \times n$ is the number of pixels, and 3 is the number of channels. First, I add sum the data from all channels to obtain `cc` an $m \times n$ image, i.e., a matrix with nonnegative entries. Next, I extract contour `h` which is the level set corresponding to the value 1. The variable `h` has the following structure. Suppose the level set corresponding to the given value X consists of p disjoint connected components, and the ith component contains q_i points. Then Then `h` is a matrix of two rows and $p+\sum_i q_i$ columns where the group of columns corresponding to the ith component is preceded by the column

$$\begin{matrix} X \\ q_i \end{matrix}$$

Hence, we can identify the contours of interest of the image `cc` using the following sequence of commands:

```
ind = find(h(1,:) == 1); % 1 is the level set value X
lh = size(h,2); % the number of columns in h
lind = length(ind); % the number of connected components in the level set
```

```
ind(lind + 1) = lh + 1; % add one more value for the for-loop below
figure;
hold on;
col = jet(lind); % make different colors for different components
 for k = 1 : lind
  f = h(:,ind(k) + 1 : ind(k+1) - 1);
plot(f(1,:),f(2,:),'Linewidth',2,'color',col(k,:));
end
```

Then look at the produced image, find the contours corresponding to the outer face and the eyes (contours 1, 3, and 4). They are given by the columns of `h ind(i) + 1 : ind(i + 1) - 1` where `i = 1,3,4`. The contours contain much more points than sufficient, so I rarefy them by taking every 6th point.

4.5. **Complete FEM packages.** A well-known FEM package is FEniCS available at http://fenicsproject.org.

Shawn Walker (Louisiana State University) developed a FEM package FELICITY written in MATLAB/C++.

D. J. Silvester, H. C. Elman (UMD, CS), and A. Ramage, developed the package IFISS (Matlab).

4.6. **Error analysis.** The content of this section is a digest of the error analysis for FEM found in [8]. We will conduct error analysis for the following model BVP:

$$-\nabla \cdot (a(x)\nabla u) = f(x), \quad x \in \Omega \subset \mathbb{R}^2, \quad u(\partial\Omega) = 0, \tag{72}$$

where $a(x)$ is continuously differentiable in $\Omega \cup \partial\Omega$ and $a(x) \geq a_0 > 0$ for all $x \in \Omega$. Assume that the region Ω is bounded. Multiplying Eq. (72) by $v \in C^1(\Omega)$, $v(\partial\Omega) = 0$, and using the First Green's Identity Eq. (62) we obtain an equivalent integral equation problem: find $u \in C^2(\Omega)$, $u(\partial\Omega) = 0$, such that for all $v \in C^1(\Omega)$, $v(\partial\Omega) = 0$, the following equation holds:

$$\int_\Omega a(x)\nabla u \cdot \nabla v dx = \int_\Omega f(x) v dx. \tag{73}$$

Next we weaken the smoothness requirements: we want $u, v \in H_0^1(\Omega)$:

$$H_0^1(\Omega) = \{w \in H^1(\Omega) \mid w(\partial\Omega) = 0\}.$$

The variational formulation for Eq. (72) is [3]

$$I(v) := \int_\Omega \left(\frac{1}{2} a(x) |\nabla v|^2 - f(x) v \right) dx \longrightarrow \min, \quad v(\partial\Omega) = 0. \tag{74}$$

Theorem 3. *u is a solution of Eq. (73) if and only if u is a minimizer of $I(u)$ among all $v \in H_0^1(\Omega)$.*

Proof. Let $u \in H_0^1(\Omega)$ be the minimizer of $I(u)$. Let $v \in H_0^1(\Omega)$ be any function. Then

$$\begin{aligned} I(u+v) &= \int_\Omega \left(\frac{1}{2} a(x) |\nabla u + \nabla v|^2 - f(x)(u+v) \right) dx \\ &= \int_\Omega \left(\frac{1}{2} a(x) |\nabla u|^2 - f(x)u + [a(x)\nabla u \cdot \nabla v - f(x)v] + \frac{1}{2} a(x) |\nabla v|^2 \right) dx \\ &= I(u) + \int_\Omega [a(x)\nabla u \cdot \nabla v - f(x)v]\, dx + \int_\Omega \frac{1}{2} a(x) |\nabla v|^2 dx. \end{aligned}$$

Clearly, since u is the minimizer of I, the integral of the term in square brackets must be 0 for all $v \in H_0^1(\Omega)$. Indeed, assume v is small in the sense of the $H^1(\Omega)$ norm. Suppose

$$\int_\Omega [a(x)\nabla u \cdot \nabla v - f(x)v]\, dx \neq 0. \tag{75}$$

If it is positive, we change v to $-v$ made hence make it negative. Since $\|v\|_{H_0^1(\Omega)}$ is small, the integral $\int_\Omega \frac{1}{2} a(x) |\nabla v|^2 dx$ is much smaller in absolute value than the one in Eq. (75). Hence $I(u+v) < I(u)$ which contradicts to the fact that u is a minimizer of I.

Now, let $u \in H_0^1(\Omega)$ be the solution to (73) for all $v \in H_0^1(\Omega)$. Then clearly $I(u) < I(u+v)$ unless $\nabla v = 0$ almost everywhere. Hence u is the minimizer of (74).

Therefore, Eq. (74) is equivalent to the problem of finding $u \in H_0^1(\Omega)$ such that for all $v \in H_0^1(\Omega)$ Eq. (73) holds. □

Now, let us fix a triangulation $\mathcal{K} = \{\Delta_k\}$, where Δ_k's are triangles. The diameter $\mathrm{diam}\Delta$ of triangle Δ is the length of its longest side. Let h be the maximal diameter of triangles in the triangulation, i.e.,

$$h = \max_{\Delta_k \in \mathcal{K}} \mathrm{diam}\Delta_k.$$

Note that one can refine the mesh decreasing h by the factor of 2 by dividing each triangle Δ_k into 4 triangles with three intervals connecting the midpoints of each side as shown in Fig. 11.

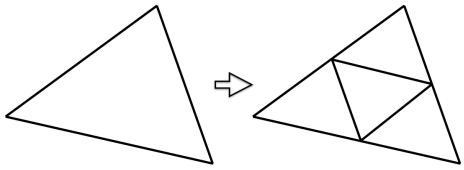


FIGURE 11. Refinement of a triangular mesh.

For the fixed triangulation K, we define the set of basis functions $\{\phi_i\}_{i \in I}$, such that each ϕ_i is continuous, linear within each triangle Δ_k, and $\phi_i(x_i) = 1$ and $\phi_i(x_j) = 0$, $j \neq i$, where the set of nodes x_i is the set of vertices of triangles in $\mathcal{K}$. Since the boundary conditions are homogeneous, it suffices to consider only those ϕ_i such that the corresponding nodes x_i lie in the interior of Ω. Let us renumber the nodes so that the inner nodes are the first N nodes $x_1, \ldots x_N$.

Recall definitions of norm and inner product. We define the norm $\|\cdot\|_a$ and the inner product $(\cdot,\cdot)_a$ associated with the function $a(x)$ and the region Ω:

$$(v,w)_a := \int_\Omega a(x)\nabla v\cdot\nabla w dx, \quad \|v\|_a = \left(\int_\Omega a(x)|\nabla v|^2 dx\right)^{1/2}. \tag{76}$$

The usual Legendre inner product will be denoted by $\langle\cdot,\cdot\rangle$:

$$\langle f,g\rangle := \int_\Omega f(x)v dx.$$

Then Eq. (73) can be rewritten as

$$(u,v)_a = \langle f,v\rangle \quad \text{for all} \quad v\in H_0^1(\Omega). \tag{77}$$

Let us consider the N-dimensional vector space S_h of continuous and piece-wise linear functions $\chi(x)$, $x\in\Omega$, such that each $\chi(x)$ is a linear combination of basis functions ϕ_i, i.e.,

$$S_h := \left\{\chi:\Omega\to\mathbb{R} \mid \chi(x)=\sum_{i=1}^N \chi_i\phi_i\right\}. \tag{78}$$

Since all basis functions ϕ_i, $1\le i\le N$, are zero on $\partial\Omega$, bounded, and piecewise continuously differentiable, all of them also belong to $H_0^1(\Omega)$, i.e., $S_h\subset H_0^1(\Omega)$. Therefore, we have

$$(u,\phi_j)_a = \langle f,\phi_j\rangle \quad \text{for all basis functions} \quad \phi_j\in S_h. \tag{79}$$

The finite element solution $u_h=\sum_{i=1}^N U_i\phi_i$ of problem (73) is found from the solution of the linear system

$$\left(\sum_{i=1}^N U_i\phi_i,\phi_j\right)_a = \langle f,\phi_j\rangle, \quad 1\le j\le N, \tag{80}$$

which is equivalent to $AU=b$ where

$$A_{ij}=(\phi_i,\phi_j)_a, \quad b_j=\langle f,\phi_j\rangle.$$

Subtracting Eq. (80) from Eq. (79) we get

$$(u-u_h,\phi_j)_a = 0, \quad 1\le j\le N. \tag{81}$$

Now we are ready to prove that the FEM solution u_h is the best approximation of the exact solution u of Eq. (77) among all functions in S_h.

Theorem 4.

$$\|u-u_h\|_a = \min_{\chi\in S_h}\|u-\chi\|_a. \tag{82}$$

Proof. Take an arbitrary $\chi\in S_h$. Introduce $e:=\chi-u_h$. Obviously, $e\in S_h$. Then

$$\begin{aligned}\|u-\chi\|_a^2 &= (u-u_h-e,u-u_h-e)_a = \|u-u_h\|_a^2 - 2(u-u_h,e)_a + \|e\|_a^2\\ &= \|u-u_h\|_a^2 + \|e\|_a^2 \ge \|u-u_h\|_a^2.\end{aligned}$$

Here we have used the fact that $(u - u_h, e)_a = 0$. Indeed, $e = \chi - u_h \in S_h$, hence $e = \sum_{i=1}^{N} e_i \phi_i$, and the result follows from Eq. (81). The equality takes place if and only if $e = 0$, i.e., $\chi \equiv u_h$. □

Next, we would like to see how the error $u - u_h$ decays with h. The error bounds for the finite element solution u_h are given in terms of the Sobolev norm $\|u\|_{H^2(\Omega)}$. For a 2D domain Ω, this norm is defined by:

$$\|v\|_{H^2(\Omega)} := \left[\int_\Omega \left(v^2 + v_x^2 + v_y^2 + v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2\right) dx\right]^{1/2}. \tag{83}$$

The following theorem establishes an error bound for the gradient of the linear interpolant $I_h u$.

Theorem 5.

$$\|\nabla u - \nabla u_h\| \le Ch\|u\|_{H^2(\Omega)}. \tag{84}$$

Proof. First we observe that since

$$0 < a_0 \le a(x) \le a_1 := \max_{x \in \Omega \cup \partial\Omega} a(x) < \infty$$

(as Ω is bounded), for any function $v \in H_0^1(\Omega)$ we have

$$a_0 \int_\Omega |\nabla v|^2 dx \le \|v\|_a^2 \le a_1 \int_\Omega |\nabla v|^2 dx.$$

Therefore, we can write the following chain of inequalities:

$$\|\nabla u - \nabla u_h\| \le a_0^{-1/2} \|u - u_h\|_a = C \min_{\chi \in S_h} \|u - \chi\|_a \le \frac{a_1^{1/2}}{a_0^{1/2}} \min_{\chi \in S_h} \left[\int_\Omega |\nabla u - \nabla \chi|^2 dx\right]^{1/2}.$$

Taking $I_h u$ for χ we get:

$$\|\nabla u - \nabla u_h\| \le \frac{a_1^{1/2}}{a_0^{1/2}} \left[\int_\Omega |\nabla u - \nabla I_h u|^2 dx\right]^{1/2}.$$

Using the result from interpolation theory that

$$\|\nabla u - \nabla I_h u\| = \left[\int_\Omega |\nabla u - \nabla I_h u|^2 dx\right]^{1/2} \le \tilde{C} h \|u\|_{H^2(\Omega)}, \tag{85}$$

we obtain that

$$\|\nabla u - \nabla u_h\| \le Ch\|u\|_{H^2(\Omega)} \quad \text{for some constant} \quad C.$$

□

To establish further results it is necessary to assume that the domain Ω is convex. In this case, one can prove that for some constant C independent of f,

$$\|u\|_{H^2(\Omega)} \le C\|f\|. \tag{86}$$

If Ω is nonconvex, then the solution u will generally have such singularities at the corners of $\partial\Omega$ that will make (86) invalid, and this will result in a lower order of convergence. Note that (84) still holds for nonconvex Ω.

Theorem 6. *Let Ω be convex. Then*

$$\|u - u_h\| \le Ch^2 \|u\|_{H^2(\Omega)}. \tag{87}$$

Proof. We consider the auxiliary problem of finding $\psi \in H_0^1(\Omega)$ such that

$$(\psi, v)_a = \langle u - u_h, v\rangle \quad \forall v \in H_0^1(\Omega). \tag{88}$$

By (86) we have

$$\|\psi\|_{H^2(\Omega)} \le C\|u - u_h\|.$$

Let us take $v = u - u_h$ in (88). Then

$$\|u - u_h\|^2 = (\psi, u - u_h)_a = (\psi - I_h\psi, u - u_h)_a. \tag{89}$$

The last equality follows from the fact that $I_h\psi \in S_h$ and for all $\phi \in S_h$ we have

$$(u - u_h, \phi) = 0$$

by definition of the finite element solution u_h. We continue (89):

$$\begin{aligned}\|u - u_h\|^2 = (\psi - I_h\psi, u - u_h)_a &\le C\|\nabla u - \nabla u_u\|\|\nabla\psi - \nabla I_h\psi\| \\ &\le C_1 h\|\nabla u - \nabla u_u\|\|\psi\|_{H^2(\Omega)} \le C_2 h\|\nabla u - \nabla u_h\|\|u - u_h\|.\end{aligned}$$

Canceling $\|u - u_h\|$ we get:

$$\|u - u_h\| \le C_2 h\|\nabla u - \nabla u_h\|. \tag{90}$$

Finally, using (85) we obtain:

$$\|u - u_h\| \le C_3 h^2 \|u\|_{H^2(\Omega)}$$

as desired. □

References

[1] Grigorios Pavliotis, Stochastic processes and Applications, Diffusion Processes, the Fokker-Planck, and Langevin Equations, Springer, 2014

[2] Randall J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations, SIAM 2007 (available online via the UMD library).

[3] K. W. Morton and D. F. Mayers, Numerical Solution of Partial Differential Equations, Second Edition, Cambridge University Press, 2005

[4] Lawrence C. Evans, Partial Differential Equations, AMS, Providence, RI, 1998

[5] John Burkardt, Meshing for the Finite Element Method (Presentation)

[6] Shaozhong Deng, Quadrature Formulas in Two Dimensions

[7] Jochen Alberty, Carsten Carstensen and Stefan A. Funken, Remarks around 50 lines of Matlab: short finite element implementation, Numerical Algorithms **20** (1999) 117137

[8] S. Larsson and V. Thomee, Partial Differential Equations with Numerical Methods, Springer-Verlag Berlin Heidelberg, 2003, 2009 (soft cover)

[9] Wikipedia