ポテンシャル問題の数値計算 (工事中)

桂田 祐史

2017 年 6 月 20 日, 2017 年 7 月 24 日

http://nalab.mind.meiji.ac.jp/~mk/complex2/

現象数理学科の学生で、この文書に載っているプログラムの動かし方が分からない人は、相談に応じます。

目 次

1 はじめに

Laplace 方程式 $\triangle u = 0$ の境界値問題を**ポテンシャル問題**という。正則関数の実部・虚部は調和関数 (ラプラス方程式の解) であるため、関数論のあちこちの重要な場面でポテンシャル問題が登場する。

(2 次元渦無し非圧縮流の速度ポテンシャル ϕ は、Laplace 方程式の Neumann 境界値問題

$$\triangle \phi = 0 \quad \text{in } \Omega, \qquad \frac{\partial \phi}{\partial n} = \boldsymbol{v} \cdot \boldsymbol{n} \quad \text{on } \partial\Omega$$

の解である、ということを既に紹介したが、関数論の中で有数の重要な定理である Riemann の写像定理に現れる等角写像を求めるためにも、ポテンシャル問題が現れる。)

ここでは少し一般化した Poisson 方程式の境界値問題

$$-\triangle u = f \quad \text{in } \Omega, \tag{1}$$

$$u = g_1 \quad \text{in } \Gamma_1, \tag{2}$$

$$\frac{\partial u}{\partial n} = g_2 \quad \text{in } \Gamma_2 \tag{3}$$

を考える。ここで Ω は平面内の領域で、Γ_1 と Γ_2 はその境界 $\partial\Omega$ を分解したものである ($\partial\Omega = \Gamma_1 \cup \Gamma_2$, $\Gamma_1 \cap \Gamma_2 = \emptyset$ が成り立つ)。n は Γ_2 上の点における、Ω の外向き単位法線ベクトルである。f, g_1, g_2 は与えられた関数である。

実は、この問題は非常に筋の良い問題であり、様々な数値計算法が適用出来る。ここでは、(1) 差分法, (2) 有限要素法, (3) 基本解の方法を紹介する。

差分法、有限要素法は、偏微分方程式に対する数値解法の、二大スタンダードと言えるもので、そういう有名な方法を紹介出来るのは有意義と考えられる。基本解の方法は、微分作用素の簡単な基本解が分かっているという、Laplace 方程式の特徴を最大限に生かす方法で、Laplace 方程式の解法としては特に優れていると言える。

2 Poisson 方程式に対する差分法

有名な差分法を紹介しよう。簡単のために領域は区間 (1 次元では線分、2 次元では長方形) とする、

2.1 基本的な考え方

差分法 (finite difference method, FDM) では、次の 2 つの考え方を用いる。

- 微分方程式に含まれる導関数を差分商で置き換えた差分方程式の解を近似解に採用する。
- 領域を格子に区切って、格子点上の値を求めることを目標にする。

常微分方程式の初期値問題に対する Euler 法, Runge-Kutta 法などを学んだことがあれば、理解しやすいであろう。

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \quad (h \to 0). \tag{4}$$

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h) \quad (h \to 0). \tag{5}$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (h \to 0). \tag{6}$$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \quad (h \to 0). \tag{7}$$

$$f'''(x) = \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3} + O(h^2) \quad (h \to 0). \tag{8}$$

$$f''''(x) = \frac{f(x+2h) - 4f(x+h) + 6f(x,y) - 4f(x-h) + f(x-2h)}{h^4} + O(h^2) \quad (h \to 0). \tag{9}$$

多変数関数の偏導関数はこれらを適当に組み合わせて近似する。例えば

$$\triangle u(x,y) = \frac{u(x+h_x,y) - 2u(x,y) + u(x-h_x,y)}{h_x^2} + \frac{u(x,y+h_y) - 2u(x,y) + u(x,y-h_y)}{h_y^2} + O(h_x^2 + h_y^2).$$

2.2 1 次元でイントロ

(6 月 14 日に説明した。)
$\Omega = (0,1)$, $\Gamma_1 = \{0\}$, $\Gamma_2 = \{1\}$ の場合、(1), (34), (3) は、次のように書き換えられる。

$$-u''(x) = f(x) \quad (x \in (0,1)), \tag{10}$$

$$u(0) = \alpha, \tag{11}$$

$$u'(1) = \beta. \tag{12}$$

2.2.1 差分方程式

$$h = \Delta x = \frac{1}{N}, \quad x_i = ih \quad (i = 0, 1, \cdots, N, N+1),$$

$$-\frac{u_{i-1}-2u_i+u_{i+1}}{h^2}=f(x_i)\quad(i=1,2,\cdots,N), \tag{13}$$

$$u_0=\alpha, \tag{14}$$

$$\frac{u_{N+1}-u_{N-1}}{2h}=\beta \tag{15}$$

結果として次の連立 1 次方程式が得られる:

$$\begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -2 & 2 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{N-1} \\ U_N \end{pmatrix} = h^2 \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-1} \\ f_N \end{pmatrix} + \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ 2\beta h \end{pmatrix}.$$

あるいは係数行列を対称化して (両辺の最後の成分に $1/2$ をかけて)

$$\begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 1 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{N-1} \\ U_N \end{pmatrix} = h^2 \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-1} \\ \frac{1}{2}f_N \end{pmatrix} + \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta h \end{pmatrix}.$$

この係数行列は既約優対角行列なので、正則である。ゆえに連立 1 次方程式は一意的な解を持つ。また、この係数行列は、対角線とその両隣り以外は 0 とになっている、いわゆる三重対角行列であり (連立 1 次方程式自体は三項方程式と呼ばれる)、Gauss の消去法で効率的に ($O(N)$ の計算量で) 解くことが出来る。

2.2.2 C 言語のプログラム例

```
/*
 * poisson1d.c --- 1 次元 Poisson 方程式
 *
 *   - u''(x)=f(x)      (0<x<1)
 *   u(0)=α
 *   u'(1)=β
 *
 * を差分法で解くプログラム。cglsc コマンドでコンパイル&実行出来る。
 *
 * u(x)=(x-1/3)^2 の場合、 α=1/9, β=4/3  --- 2 次関数なので厳密に解ける
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <glsc.h>

#define MAXN (100);

// 三重対角行列の Gauss の消去法による LU 分解
void trilu1(int, double *, double *, double *);
// LU 分解を用いて三項方程式を解く
void trisol1(int, double *, double *, double *, double *);

// Poisson 方程式の右辺
double f(double x)
{
```

```
  return - 2.0;
}

// 厳密解
double solution(double x)
{
  return (x - 1.0 / 3) * (x - 1.0 / 3);
}

int main(void)
{
  double alpha, beta;
  int i, n;
  double *x, *al, *ad, *au, *u;
  double h, h2;
  alpha = 1.0 / 9.0;
  beta = 4.0 / 3.0;

  printf("n="); scanf("%d", &n);

  x = malloc(sizeof(double) * (n + 1));
  al = malloc(sizeof(double) * (n + 1));
  ad = malloc(sizeof(double) * (n + 1));
  au = malloc(sizeof(double) * (n + 1));
  u = malloc(sizeof(double) * (n + 1));
  if (ad == NULL || al == NULL || au == NULL || u == NULL) {
    fprintf(stderr, "メモリの割り当てに失敗しました。\n");
    exit(1);
  }
  h = 1.0 / n;
  for (i = 0; i <= n; i++)
    x[i] = i * h;
  h2 = h * h;
  for (i = 1; i <= n; i++) {
    ad[i] = 2;
    au[i] = al[i] = -1;
    u[i] = h2 * f(i * h);
  }
  /* 第 N 成分は 1/2 倍する */
  ad[n] = 1;
  u[n] *= 0.5;
  /* 非同次境界条件 */
  u[1] += alpha;
  u[n] += beta * h;
  /* 連立 1 次方程式を解く */
  trilu1(n, al, ad, au);
  trisol1(n, al, ad, au, u);

  /* グラフの準備 */
  u[0] = alpha;
  g_init("POISSON1D", 170.0, 170.0);
  g_device(G_BOTH);
  g_def_scale(0,
      -0.2, 1.2, -0.2, 1.2,
      10.0, 10.0, 150.0, 150.0);
  g_sel_scale(0);
  g_move(-0.2, 0.0); g_plot(1.2, 0.0);
  g_move(0.0, -0.2); g_plot(0.0, 1.2);

  /* 厳密解を描く */
  g_move(x[0], solution(x[0]));
  for (i = 1; i <= n; i++)
    g_plot(x[i], solution(x[i]));
```

```
  printf("クリックして下さい\n");
  g_sleep(-1.0);
  /* 差分解を赤く描く */
  g_line_color(G_RED);
  g_move(x[0], u[0]);
  for (i = 1; i <= n; i++)
    g_plot(x[i], u[i]);

  printf("x=1 での厳密解 %f, 差分解 %f, 誤差 %e\n",
         solution(1.0), u[n], fabs(solution(1.0) - u[n]));
  g_sleep(-1.0);
  g_term();

  return 0;
}

/* 3 項方程式 (係数行列が三重対角である連立 1 次方程式のこと) Ax=b を解く
 *
 *   入力
 *     n: 未知数の個数
 *     al,ad,au: 連立 1 次方程式の係数行列
 *       (al: 対角線の下側 i.e. 下三角部分 (lower part)
 *        ad: 対角線       i.e. 対角部分 (diagonal part)
 *        au: 対角線の上側 i.e. 上三角部分 (upper part)
 *        つまり
 *
 *         ad[1] au[1]   0   ................. 0
 *         al[2] ad[2] au[2]   0  ............. 0
 *           0   al[3] ad[3] au[3]  0 ......... 0
 *                          ...................
 *                          al[n-1] ad[n-1] au[n-1]
 *                            0     al[n]   ad[n]

 *        al[i] = A_{i,i-1}, ad[i] = A_{i,i}, au[i] = A_{i,i+1},
 *        al[1], au[n] は意味がない)
 *
 *     b: 連立 1 次方程式の右辺の既知ベクトル
 *        (添字は 1 から。i.e. b[1],b[2],...,b[n] にデータが入っている。)
 *   出力
 *     al,ad,au: 入力した係数行列を LU 分解したもの
 *     b: 連立 1 次方程式の解
 *   能書き
 *     一度 call すると係数行列を LU 分解したものが返されるので、
 *     以後は同じ係数行列に関する連立 1 次方程式を解くために、
 *     関数 trisol1() が使える。
 *   注意
 *     Gauss の消去法を用いているが、ピボットの選択等はしていな
 *     いので、ピボットの選択をしていないので、係数行列が正定値である
 *     などの適切な条件がない場合は結果が保証できない。
 */

void trid1(int n, double *al, double *ad, double *au, double *b)
{
  trilu1(n,al,ad,au);
  trisol1(n,al,ad,au,b);
}

/* 三重対角行列の LU 分解 (pivoting なし) */
void trilu1(int n, double *al, double *ad, double *au)
{
  int i;
  /* 前進消去 (forward elimination) */
  for (i = 1; i < n; i++) {
```

```
    al[i + 1] /= ad[i];
    ad[i + 1] -= au[i] * al[i + 1];
  }
}

/* LU 分解済みの三重対角行列を係数に持つ 3 項方程式を解く */
void trisol1(int n, double *al, double *ad, double *au, double *b)
{
  int i;
  /* 前進消去 (forward elimination) */
  for (i = 1; i < n; i++) b[i + 1] -= b[i] * al[i + 1];
  /* 後退代入 (backward substitution) */
  b[n] /= ad[n];
  for (i = n - 1; i >= 1; i--) b[i] = (b[i] - au[i] * b[i + 1]) / ad[i];
}
```

2.3 2 次元の場合 (結果だけ紹介)

長方形領域 $\Omega = (0, W) \times (0, H)$ の場合は、1 次元と大筋で同様にして扱える。簡単のため Dirichlet 境界値問題、すなわち

$$\mathit{\Gamma}_1 = \partial\Omega, \quad \mathit{\Gamma}_2 = \emptyset \quad (\text{Neumann 境界が空集合})$$

の場合に説明する。この場合は、(1), (34), (3) は、

$$-\triangle u = f \quad (\text{in } \Omega), \quad u = g_1 \quad (\text{on } \partial\Omega)$$

と書ける。

2.3.1 差分方程式

$N_x, N_y \in \mathbb{N}$ に対して、

$$h_x = \Delta x := \frac{W}{N_x}, \quad h_y = \Delta y := \frac{H}{N_x},$$
$$x_i = i\Delta x \quad (0 \le i \le N_x), \quad y_j = j\Delta y \quad (0 \le j \le N_y)$$

によって格子点 (x_i, y_j) を定める。

領域内部にある格子点のインデックスの集合を

$$\omega := \{(i, j) \mid 1 \le i \le m,\ 1 \le j \le n\}, \quad m := N_x - 1, \quad n := N_y - 1,$$

領域の境界上にある格子点のインデックスの集合を

$$\gamma := \{(i, 0) \mid 0 \le i \le N_x\} \cup \{(i, N_y) \mid 0 \le i \le N_x\} \cup \{(0, j) \mid 0 \le j \le N_y\} \cup \{(N_x, j) \mid 0 \le j \le N_y\}$$

とおく。個数を調べておこう。

$$\sharp\omega = (N_x - 1)(N_y - 1) = mn, \quad \sharp\gamma = 2(N_x + N_y), \quad \sharp(\omega \cup \gamma) = (N_x + 1)(N_y + 1).$$

$$-\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{\Delta x^2} - \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{\Delta y^2} = f(x_i, y_j) \quad ((i, j) \in \omega), \tag{16}$$

$$U_{i,j} = g_1(x_i, y_j) \quad ((i, j) \in \gamma), \tag{17}$$

という差分方程式の解 U_{ij} を $u(x_i, y_j)$ の近似値とするのが良いと期待できる。

これは $(N_x+1)(N_y+1)$ 個の未知数 $U_{i,j}$ $((i,j) \in \omega \cup \gamma)$ についての、$(N_x+1)(N_y+1)$ 個の 1 次方程式である。$U_{i,j}$ $((i,j) \in \gamma)$ は (17) から分かるので、それを (16) に代入して消去すると、$(N_x-1)(N_y-1)$ 個の未知数 $U_{i,j}$ $((i,j) \in \omega)$ についての、

$$N := (N_x-1)(N_y-1) \tag{18}$$

個の 1 次方程式が得られる。

未知数の個数と方程式の個数が等しいので、正方行列を係数とする連立 1 次方程式の形に表せるはずである。実際にそうするためには、$U_{i,j}$ を並べて 1 つのベクトルにする必要がある。

以下このことを実行するが、自分でプログラムを書く必要が生じるまで、読む必要はないであろう。

2.3.2 差分方程式 (連立 1 次方程式) の行列、ベクトル表現

簡単のため、同次 Dirichlet 境界条件 $(g_1 \equiv 0)$ の場合に説明する。

$$\boldsymbol{U} = \begin{pmatrix} U_{11} \\ U_{21} \\ \vdots \\ U_{N_x-1,1} \\ \hline U_{12} \\ U_{22} \\ \vdots \\ U_{N_x-1,2} \\ \hline \vdots \\ \hline U_{1,N_y-1} \\ U_{2,N_y-1} \\ \vdots \\ U_{N_x-1,N_y-1} \end{pmatrix}, \quad \boldsymbol{F} = \begin{pmatrix} f(x_1,y_1) \\ f(x_2,y_1) \\ \vdots \\ f(x_{N_x-1},y_1) \\ \hline f(x_1,y_2) \\ f(x_2,y_2) \\ \vdots \\ f(x_{N_x-1},y_2) \\ \hline \vdots \\ \hline f(x_1,y_{N_y-1}) \\ f(x_2,y_{N_y-1}) \\ \vdots \\ f(x_{N_x-1},y_{N_y-1}) \end{pmatrix}$$

とおくと、差分方程式から次のような連立 1 次方程式が得られる:

$$A\boldsymbol{U} = \boldsymbol{F},$$

$$A := \left(I_{N_y-1} \otimes \frac{1}{h_x^2}(2I_{N_x-1} - J_{N_x-1}) + \frac{1}{h_y^2}(2I_{N_y-1} - J_{N_y-1}) \otimes I_{N_x-1} \right). \tag{19}$$

ここで $\otimes$ は行列のテンソル積を表す記号であり、I_k は k 次の単位行列、J_k は次の形の k 次正方行列であるとする。

$$J_k = \begin{pmatrix} 0 & 1 & & & 0 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ 0 & & & 1 & 0 \end{pmatrix},$$

(詳しくは桂田 [1] を見よ。)

プログラムを書くときのために、$\boldsymbol{U}$, $\boldsymbol{F}$ の成分 U_ℓ, F_ℓ を式で表しておく。

$$U_\ell = U_{i,j}, \quad F_\ell = f(x_i, y_j), \tag{20}$$

$$\ell = i + (j-1)(N_x-1). \tag{21}$$

つまり、領域内部の格子点 (x_i, y_j) を 1 次元的に並べて番号をつけた[1]、ということである。並べ方は一通りではなく、

$$\ell = j + (i-1)(N_y - 1) \tag{22}$$

というものも良く使われる。(21) を row first, (22) を column first と呼んで区別する。次に紹介する MATLAB プログラムでは、(21) を採用してある。

2.3.3 MATLAB プログラム

MATLAB[2] は定評のあるシミュレーション・ソフトウェアである (桂田 [2])。

明治大学ではライセンス契約をしているので、学生は申請すれば使えるようになる (MATLAB TAH ライセンス[3])。

(もちろん研究テーマによるが) 卒業研究等で有効に利用できる可能性があり、学ぶ価値は高い。この授業で MATLAB の解説をする余裕はないが、ここでは既に MATLAB を知っている人向けにサンプル・プログラムを紹介しておく。(なお、熱方程式を解くための MATLAB プログラムについては、桂田 [3], [4] を見よ。)

(19) の係数行列 A は次のプログラムで計算出来る。

poisson_coef.m

```
function A=poisson_coef(W, H, nx, ny)
% 長方形領域 (0,W) × (0,H) における Poisson 方程式の Dirichlet 境界値問題
% Laplacian を差分近似した行列を求める。
% 長方形を nx × ny 個の格子に分割して差分近似する。
% MATLAB では
%  (1) 行列は Fotran と同様の column first であり、
%  (2) mesh(), contour() による「行列描画」は Z(j,i) と添字の順が普通と逆なので、
% l=i+(j-1)*(nx-1) と row first となるように 1 次元的番号付けする
  hx=W/nx;
  hy=H/ny;
  m=nx-1;
  n=ny-1;
  ex=ones(nx,1);
  ey=ones(ny,1);
  Lx=spdiags([-ex,2*ex,-ex],-1:1,m,m)/(hx*hx);
  Ly=spdiags([-ey,2*ey,-ey],-1:1,n,n)/(hy*hy);
  A=kron(speye(m,m),Ly)+kron(Lx,speye(n,n));
```

$$W = 3, \quad H = 2, \quad f \equiv 1, \quad N_x = 30, \quad N_y = 20$$

の場合、次のプログラムで計算出来る。

[1] $(i,j) \mapsto \ell$ の逆変換は、ℓ を $N_x - 1$ で割った余りと商を $i-1$, $j-1$ とすれば良い。

[2] https://jp.mathworks.com/

[3] http://www.meiji.ac.jp/isc/matlab-tah/

poisson2d_f1.m

```
% 長方形領域 (0,W) × (0,H) で Poisson 方程式の同次 Dirichlet 境界値問題を解く
W=3.0;
H=2.0;
nx=30;
ny=20;
m=nx-1;
n=ny-1;
% 連立 1 次方程式を作成する。
A=poisson_coef(W, H, nx, ny);
% 描画用のメッシュ・グリッドを用意
x=linspace(0,W,nx+1); % x=[x_0,x_1,...,x_nx]
y=linspace(0,H,ny+1); % y=[y_0,y_1,...,y_ny]
[X,Y]=meshgrid(x,y); % これについては meshgrid() の説明を見よ。
% f ≡ 1 の場合
F=ones(m*n,1); % ここを頑張ると一般の非同次項の問題が解ける。
%
u=zeros(ny+1,nx+1);
u(2:ny,2:nx)=reshape(A\F,ny-1,nx-1);
%
% グラフの鳥瞰図
clf
colormap hsv
subplot(1,2,1);
mesh(X,Y,u);
colorbar
% 等高線
subplot(1,2,2);
contour(X,Y,u);
%
disp('図を保存する');
print -dpdf poisson2d.pdf % 利用できるフォーマットは doc print で分かる
```

問 1. プログラム poisson2d_f1.m は、$f \equiv 1$ の場合に問題を解くプログラムであるが、一般の f の場合に問題を解くプログラムを作成せよ。

以下の図は、$f(x,y) = -2(x^2 - 3x + y^2 - 2y)$ の場合 (厳密解は $u(x,y) = x(3-x)y(2-y)$) の解を可視化したものである。

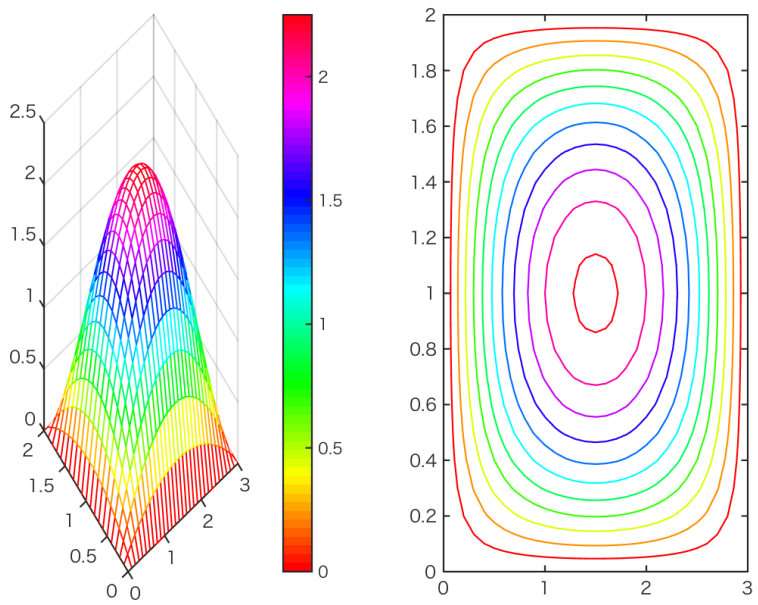

図 1: $-\triangle u = -2(x^2 - 3x + y^2 - 2y)$

問 2. プログラム `poisson2d_f1.m` は、同次 Dirichlet 境界条件 ($u = 0$ on $\partial\Omega$) の場合に問題を解くプログラムであるが、非同次 Dirichlet 境界条件 ($u = g_1$ on $\partial\Omega$) の場合に問題を解くプログラムを作成せよ。

2.4 区間でない領域における差分法

(準備中)

2.5 結び

- 差分方程式の導出は比較的分かりやすい (他人が作った差分方程式が、元の微分方程式の近似になっていることは分かりやすい) が、差分解の収束や、差分スキームの安定性などの証明は、もとの偏微分方程式の性質の証明と関連がある (結局勉強をサボるのは難しい)。
- 多次元の場合、境界が曲がっている領域を扱うには工夫が必要になる。

3 Poisson 方程式に対する有限要素法

この節の狙い: 有限要素法の話は、基本的な部分を説明するだけで、セメスターの科目になってしまうようなボリュームがあり、とても短時間で紹介出来るものではないが、FreeFem++ (これはぜひ紹介したい) のプログラムをただの呪文にしてしまわないためには、弱形式にそれなりのしっかりした背景 (弱解の方法) があることを知って欲しい、と考えた。

弱解の方法は、現代の偏微分方程式論になくてはならないもので、学ぶ価値が高い、ということもある。

3.1 ポテンシャル問題の解の存在証明、弱解の方法、Ritz-Galerkin 法

(ここはお話です。急ぐ時はすっ飛ばしても大丈夫。)

G. F. B. Riemann (1826–1866) が、今では Riemann の写像定理と呼ばれる定理を発表した際 (1851 年) に、Laplace 方程式の Dirichlet 境界値問題

$$\triangle u = 0 \quad (\text{in } \Omega), \quad u = g_1 \quad (\text{on } \partial\Omega)$$

の解 u が存在すること証明する必要性が生じた。彼はそれを「**Dirichlet の原理**[4]」を用いて "証明" した。

Riemann による Laplace 方程式の Dirichle 境界値問題の議論のあらすじ　境界条件 $u = g_1$ (on $\partial\Omega$) を満たす u のうちで

$$J[u] = \iint_\Omega \left(u_x^2 + u_y^2\right) dx\, dy \quad (\text{この } J \text{ を Dirichlet 積分と呼ぶ})$$

を最小にするものは、$\triangle u = 0$ を満たす。実際、v を $v = 0$ (on $\partial\Omega$) を満たす任意の関数とするとき、

$$f(t) := J[u + tv] \quad (t \in \mathbb{R})$$

は $t = 0$ で最小となる (なぜならば $f(t) = J[u + tv] \geq J[u] = f(0)$)。ところで

$$f(t) = J[u] + 2t \iint_\Omega (u_x v_x + u_y v_y)\, dx\, dy + t^2 \iint_\Omega \left(v_x^2 + v_y^2\right) dx\, dy$$

であるから、f は 2 次関数であり、$t = 0$ で最小となるためには

$$\iint_\Omega (u_x v_x + u_y v_y)\, dx\, dy = 0$$

が必要十分である。Green の積分公式[5]して

$$\iint_\Omega (u_{xx} + u_{vv}) v \; dx\, dy = 0.$$

これが任意の v について成り立つことから、$\triangle u = u_{xx} + u_{yy} = 0$.

以上の議論から、$J[u]$ を最小にするような u を見出せば問題が解決することが分かる。J は常に $J \geq 0$ を満たすので、J が下に有界でありることは明らかで、従って J の下限が存在する。(この下限は最小値であるから)、最小値を与える u が存在する、と議論したのだが、Weierstrass は「下限は最小値である」ことに疑義を示した (「数学解析」を学んだ人は、いかにも Weierstrass がツッコミそうなところと思うかも)。■

残念ながら若くして亡くなった Riemann は、Weierstrass の批判に答えることが出来なかった。この論法による完全な証明は、約 50 年後 (1900 年頃) の D. Hilbert まで持ち越された。そのやり方は、Laplace 方程式以外の多くの微分方程式に対しても拡張され、今では「**弱解の方法**」と呼ばれる。

---

[4] なんでも、Dirihlet 先生の講義に出て来たのだとか。

[5] Green の積分公式とは、$\displaystyle\iint_\Omega \triangle uv \; dx\, dy = \int_{\partial\Omega} \frac{\partial u}{\partial n} v \; d\sigma - \iint_\Omega \nabla u \cdot \nabla v \; dx\, dy$ と言うもの。これは発散定理 $\displaystyle\int_\Omega \operatorname{div} \boldsymbol{f} \; dx\, dy = \int_{\partial\Omega} \boldsymbol{f} \cdot \boldsymbol{n} \; d\sigma$ に、$\boldsymbol{f} = v\nabla u$ を代入すれば得られる。$\operatorname{div}(v\nabla u) = \nabla u \cdot \nabla v + v \triangle u$, $\nabla u \cdot \boldsymbol{n} = \frac{\partial u}{\partial n}$ であることに注意する。

本当は、Dirichlet の原理は、C. F. Gauss (1777–1855) がルーツで、物理学の世界ではすでに知られていた考え方で、それを Riemann が純粋数学に応用した、という見方をする人もいる。

弱解の方法は、数値計算とも相性がよく、そこに基礎を置く W. Ritz による Ritz の方法は 1909 年に発表され次第、重要な地位を占めている。この **Ritz-Galerkin 法**は有限要素法の基礎ともなっている。

(差分法の基礎を、導関数の差分商への置き換え＋領域の格子への分割とまとめるのを真似ると、有限要素法の基礎は、Ritz-Galerkin 法＋領域の有限要素への分割、とまとめるのが良いだろうか。)

3.2　1 次元でイントロ

有限要素法を使うためには、最低限弱形式の導出が出来ないといけない。

簡単のため、まず 1 次元版で論じる (多次元でも本質的な違いはない)。

(P)

$$-u''(x) = f(x) \quad (x \in (0,1)), \tag{23}$$

$$u(0) = \alpha, \tag{24}$$

$$u'(1) = \beta \tag{25}$$

上の境界値問題の解は、次の 2 つの問題 (W), (V) の解でもある。

まず弱形式 (weak from)、あるいは弱定式化 (weak formulation) した問題 (W) を述べよう。

(W)

Find $u \in X_{g_1}$ s.t.

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x)dx + \beta v(1) \quad (v \in X).$$

ここで X_{g_1} と X は

$$X := \left\{ v \in H^1(0,1) \mid v(0) = 0 \right\}, \quad X_{g_1} := \left\{ v \in H^1(0,1) \mid v(0) = \alpha \right\} \tag{26}$$

で定義される。ただし $H^1(0,1)$ は 1 階の Sobolev 空間である。

(P) の解が (W) を満たすこと　u が (P) を満たすとする。任意の $v \in X$ を (23) にかけて、$[0,1]$ で積分し、部分積分すると、

$$-\left[u'(x)v(x)\right]_0^1 + \int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx.$$

X の定義から $v(0) = 0$, また (25) が成り立つので、

$$\left[u'(x)v(x)\right]_0^1 = u'(1)v(1) - u'(0)v(0) = \beta v(1).$$

ゆえに

$$\int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx + \beta v(1).$$

すなわち u は問題 (W) の解である。 ■

問 3. 逆に問題 (W) の解は、C^2 級であれば、(P) の解でもあることを示せ。

次に変分問題[6](variational problem) にしたものを述べる。

(V)

Find $u \in X_{g_1}$ s.t.

$$J[u] = \min_{w \in X_{g_1}} J[w].$$

ただし

$$J[u] := \frac{1}{2}\int_0^1 |u'(x)|^2\,dx - \int_0^1 f(x)v(x)\,dx - \beta v(1).$$

(W) と (V) は同値な問題であり、常に一意的な解を持つことが比較的容易に分かる。

問 4. (W) と (V) が同値な問題であることを示せ。(ヒント:

$$J[u+tv] - J[u] = t\left[\iint_0^1 (u'(x)v'(x) - f(x)v(x))\,dx\,dy - \beta v(1)\right] + \frac{t^2}{2}\iint_\Omega (v_x^2 + v_y^2)dx\,dy$$

が成り立つことが簡単な計算で確認できる。)

問題 (V) の解 (それは (W) の解でもある) が C^2 級であることを認めると、(P), (W), (V) は互いに同値な問題ということになる。(W) $\Rightarrow$ (P) は、Dirichlet 原理の一般化である (Laplace 方程式の Dirichlet 境界値問題の場合、この J は Dirichlet 積分 (の 1/2 倍) に他ならない。)。

そこで問題 (P) を解く代わりに、(W) あるいは (V) を解くことを目指す。

通常、変分法は、変分問題を解くために、それと同値な微分方程式の問題を導き、そちらを解くことで変分問題の解を得るのが普通であるが、ここでは逆に微分方程式の問題を解くために、それを変分問題に書き換え、それを直接解く、という手順の議論をしている。これは、変分法の**直接法**と呼ばれるものになっている。

$\{x_i\}_{i=0}^N$ を

$$0 = x_0 < x_1 < \cdots < x_N = 1$$

を満たす数列として、

$$\widetilde{X} := \{v \in C([0,1]) \mid v \text{ は小区間 } [x_i - 1, x_i] \text{ では 1 次多項式と一致 }\},$$

$$\hat{X} := \left\{v \in \widetilde{X} \;\middle|\; v(0) = 0\right\}, \quad \hat{X}_{g_1} := \left\{v \in \widetilde{X} \;\middle|\; v(0) = \alpha\right\}$$

とおくとき、X を $\hat{X}$ で、X_{g_1} を $\hat{X}_{g_1}$ で置き換えた問題を考える。$\widetilde{X}$ の要素を**区分 1 次多項式**と呼ぶ。

次の 2 つの問題は同値であり、常に一意的な解 $\hat{u}$ を持つ。それを近似解として採用する。

($\widehat{\text{W}}$)

Find $\hat{u} \in \hat{X}_{g_1}$ s.t.

$$\int_0^1 \hat{u}'(x)v'(x)dx = \int_0^1 f(x)v(x)dx + \beta v(1) \quad (v \in \hat{X}).$$

[6]変分法を知らない人のために説明: 汎函数 (関数を変数とする関数のこと) の最小値問題を変分問題と呼ぶ。ここでは、$J\colon X_{g_1} \to \mathbb{R}$ が汎函数で、u は J の最小値を与える点となっている。

($\widehat{\mathrm{V}}$)

Find $\hat{u} \in \hat{X}_{g_1}$ s.t.

$$J[\hat{u}] = \min_{w \in \hat{X}_{g_1}} J[w].$$

ϕ_i を、$\phi_i \in \widetilde{X}$,

$$\widetilde{\phi}_i(x_j) = \delta_{ij}$$

を満たすものとする (この条件で ϕ_i は一意的に定まる)。任意の $\hat{u} \in X_{g_1}$ は、

$$\hat{u}(x) = \alpha\phi_0(x) + \sum_{i=1}^{N} a_i\phi_i(x)$$

の形に一意的に表現出来る。係数 $a_1, \cdots, a_N$ を定めれば良いが、u が (W) (あるいは (V)) を満たすことは、$a_1, \ldots, a_N$ がある連立 1 次方程式の解であることと同値であることが分かる。

実は $\{x_i\}$ が $[0,1]$ の N 等分点であるとき、有限要素解 $\hat{u}$ の x_i での値は、差分解 U_i と一致する。もちろん、いつもそうなるわけではない (もしそうならば、2 つの方法を考える意味がない)。

有限要素法には以下の利点がある。

- 弱形式の議論を済ませてあれば、有限要素解の厳密解への収束の議論は簡単になる。
- 多次元問題の場合に、長方形領域以外でも、それほど苦労なく解析が可能である。
- プログラムの自動生成がしやすい。

3.3 2 次元の場合—

3.3.1 弱形式

部分積分を、その一般化である Green の積分公式に置き換えるだけで、後は 1 次元とほぼ同様の議論が可能である。その結果、次のような弱形式が得られる。

Find $u \in X_{g_1}$ s.t.

$$\int_\Omega \operatorname{grad} u \cdot \operatorname{grad} v \, dx = \int_\Omega fv \, dx + \int_{\Gamma_2} g_2 v \, ds \quad (v \in X). \tag{27}$$

ここで

$$X_{g_1} := \left\{ w \in H^1(\Omega) \mid w = g_1 \quad \text{on } \Gamma_1 \right\},$$
$$X := \left\{ w \in H^1(\Omega) \mid w = 0 \quad \text{on } \Gamma_1 \right\}.$$

念のため: $\operatorname{grad} u \cdot \operatorname{grad} v = u_x v_x + u_y v_y$.

3.3.2 FreeFem++ プログラム (その 1)

有限要素法は、弱解の方法を原理とする数値計算法である。それはプログラム作成のかなりの部分を自動化出来るため、専用のソフトウェアがいくつか開発されている。

その 1 つである、FreeFem++[7] は、パリ第 6 大学 J. L. Lions 研究所の Frédéric Hecht, Oliver Pironneau, A. Le Hyaric, 広島国際学院大学の大塚厚二氏らが開発した、2 次元, 3 次元問題を

[7] http://www.freefem.org/ff++/

有限要素法で解くための、 一種の PSE (problem solving environment) である。ソースコード、マニュアル (約 400 ページ, 幸い英文)、主なプラットホーム (Windows, Mac, Linux) 向けの 実行形式パッケージが公開されている。

細かいことは、以前書いた桂田 [5] という紹介文を見てもらうことにする。

有限要素法の定番教科書の一つである菊地 [6] に載っている Poisson 方程式の例題

$$-\triangle u = f \quad \text{in } \Omega, \tag{28}$$

$$u = g_1 \quad \text{in } \Gamma_1, \tag{29}$$

$$\frac{\partial u}{\partial n} = g_2 \quad \text{in } \Gamma_2 \tag{30}$$

(ただし、$\Omega = (0,1) \times (0,1)$, $\Gamma_1 = \{0\} \times [0,1] \cup [0,1] \times \{0\}$, $\Gamma_2 = \{1\} \times (0,1] \cup (0,1] \times \{1\}$, $f = 1$, $g_1 = 0$, $g_2 = 0$) を FreeFEM++ を用いて解くとどうなるか。

次のようなプログラムで解ける。

Poisson2.edp

```
// Poisson2.edp
int Gamma1=1, Gamma2=2;
border Gamma10(t=0,1) { x=0; y=1-t; label=Gamma1; }
border Gamma11(t=0,1) { x=t; y=0; label=Gamma1; }
border Gamma20(t=0,1) { x=1; y=t; label=Gamma2; }
border Gamma21(t=0,1) { x=1-t; y=1; label=Gamma2; }
int m=10;
mesh Th = buildmesh(Gamma10(m)+Gamma11(m)+Gamma20(m)+Gamma21(m));
plot(Th, wait=1,ps="Th.eps");
savemesh(Th,"Th.msh"); // optional
fespace Vh(Th,P1);
Vh u,v;
func f=1;
func g1=0;
func g2=0;
solve Poisson(u,v) =
   int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))
  -int2d(Th)(f*v)
  -int1d(Th,Gamma2)(g2*v)
  +on(Gamma1,u=g1); // on(Gamma10,Gamma11,u=g1)
plot(u,ps="contour.eps");
```

プログラムはテキスト・エディター (現象数理学科 Mac では、mi, テキスト・エディット, emacs など) で作成し、ターミナルから、

こんなふうにして実行

```
FreeFem++ Poisson2.edp
```

とタイプして実行できる。return キーを打つごとに次の図に移り、最後は esc キーを打って終了する。

3.3.3 FreeFem++ プログラム (その 2)

速度ポテンシャル ϕ に対する Laplace 方程式の Neumann 境界値問題で、$\partial\Omega$ 上で $\boldsymbol{v}$ が与えられたとき

$$\triangle \phi = 0 \quad (\text{in } \Omega) \tag{31}$$

$$\frac{\partial \phi}{\partial \boldsymbol{n}} = \boldsymbol{v} \cdot \boldsymbol{n} \quad (\text{on } \partial\Omega) \tag{32}$$

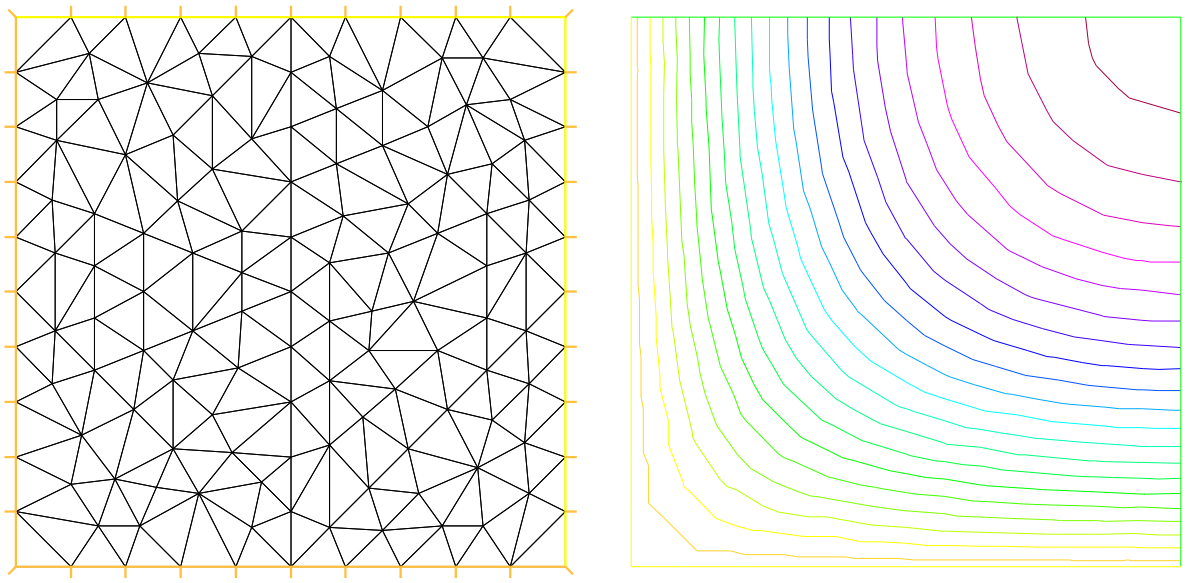

図 2: Poisson2.edp の出力 — 要素分割と解の等高線

を満たす ϕ を求めよ、というもの。

これは、$\Gamma_1 = \emptyset$, $\Gamma_2 = \partial\Omega$, $f = 0$, $g_2 = \boldsymbol{v} \cdot \boldsymbol{n}$ の場合に相当する。

特に $\Omega = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 < 1\}$ のときは、$(x, y) \in \partial\Omega$ において $\boldsymbol{n} = \begin{pmatrix} x \\ y \end{pmatrix}$ であるから、$\boldsymbol{v} \equiv \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ とすると

$$g_2 = \boldsymbol{v} \cdot \boldsymbol{n} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = x + 2y.$$

速度ポテンシャルを求める solveLaplace.edp

```
// solveLaplace.edp
border Gamma2(t=0,2*pi) { x=cos(t); y=sin(t); }
int m=40;
mesh Th = buildmesh(Gamma2(m));
plot(Th, wait=1, ps="Th.eps");
savemesh(Th,"Th.msh"); // optional
fespace Vh(Th,P1);
Vh u,v;
func f=0;
func g1=0; // no use
func g2=x+2*y;
solve Poisson(u,v) =
   int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))
  -int2d(Th)(f*v)
  -int1d(Th,Gamma2)(g2*v) // +on(Gamma1,u=g1)
    ;
plot(u,ps="equipotential.eps");
```

(実は、上の弱形式は解の一意性がないので、このプログラムは少し危ういところがある。)

4　等角写像の数値計算

関数論における重要な等角写像の数値計算に対する 1 つのアルゴリズムを紹介する。

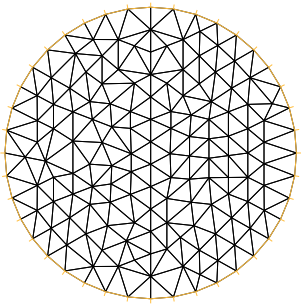

図 3: Ω の三角形分割

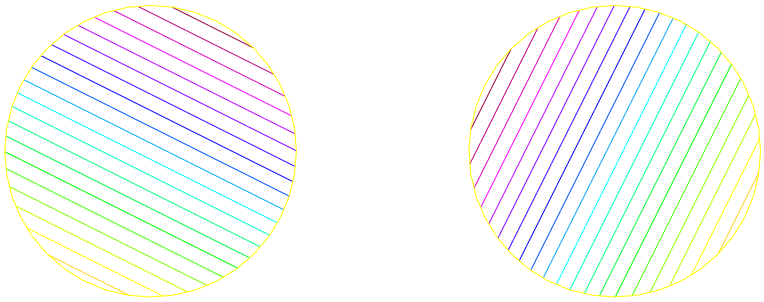

図 4: 一様流の等ポテンシャル線と流線

4.1 Riemann の写像定理

(講義でそれなりに説明するつもり…その要点をこちらに写したい)

U, V を $\mathbb{C}$ の領域とする。$\varphi\colon U \to V$ が**双正則**であるとは、φ が正則でかつ全単射で、φ^{-1} も正則であることをいう。

Ω を $\mathbb{C}$ の単連結領域で、$\mathbb{C}$ とは異なるものとするとき、双正則写像

$$\varphi\colon \Omega \to D_1 = D(0;1)$$

が存在する (**Riemann の写像定理**, 1851 年)。

この φ のことを領域 Ω の**等角写像**, あるいは**写像関数**と呼ぶ[8]。

問題となる領域の等角写像はしばしば役に立つ。そのため、その計算方法は重要視され、古くから研究されてきた。多角形領域の場合の Schwarz-Christoffel mapping などは、時間の関係で、「複素関数」、「応用複素関数」ではスルーしているが、複素関数論の定番のメニューと言える。

4.2 単連結領域の場合の等角写像の正規化条件

Ω を $\mathbb{C}$ の単連結領域で、$\mathbb{C}$ とは異なるものとする。Riemann の写像定理により、Ω の等角写像

$$\varphi\colon \Omega \to D_1 = D(0;1)$$

が存在する。この写像は一意的には定まらないが、次が成り立つ。

[8]関数論において等角写像とは、いたるところ $\varphi' \neq 0$ を満たす正則関数のことを指す。一対一の等角写像 $\varphi\colon U \to \mathbb{C}$ があるとき、終域を $V := \varphi(U)$ で置き換えた、$\tilde{\varphi}\colon z \mapsto \varphi(z) \in V$ は双正則である ($\tilde{\varphi}'(z) \neq 0$ が簡単に導かれる)。しばしば、等角写像という言葉を、双正則な関数の意味に使うことがある。d

命題 4.1 Ω を $\mathbb{C}$ とは異なる $\mathbb{C}$ の単連結領域で、z_0 を Ω の任意の点とする。このとき

$$\varphi(z_0) = 0, \quad \varphi'(z_0) > 0 \tag{33}$$

という条件を満たす双正則写像 $\varphi\colon \Omega \to D_1$ は、存在すれば一意的である。

証明　φ_1, φ_2 がその条件を満たすとする。$\psi := \varphi_2 \circ \varphi_1^{-1}$ とおくと、$\psi\colon D_1 \to D_1$ は双正則写像である。そして、

$$\psi(0) = \varphi_2\left(\varphi_1^{-1}(0)\right) = \varphi_2(z_0) = 0$$

であるから、

$$(\exists \varepsilon \in \mathbb{C} : |\varepsilon| = 1)(\forall z \in D_1) \quad \psi(z) = \varepsilon \frac{z - 0}{1 - \overline{0}z} = \varepsilon z.$$

ところが、

$$\varepsilon = \psi'(0) = \varphi_2'(z_0)\frac{1}{\varphi_1'(z_0)} > 0.$$

であるから、$\varepsilon = 1$. ゆえに $\psi(z) = z$. ゆえに $\varphi_2 = \varphi_1$. ■

4.3　Jordan 領域の等角写像を求めるアルゴリズム

典型的な単連結領域に **Jordan 領域**がある。

Jordan 曲線定理

$\mathbb{C}$ 内の Jordan 閉曲線 C に対して、C の “囲む” 領域 Ω が定まり、Ω は有界かつ単連結で、その境界は C の像に等しい。

この定理で存在が保証される領域を、C の定める Jordan 領域と呼ぶ。Jordan 領域 Ω は単連結領域であるから、Riemann の写像定理によって、Ω の等角写像 φ が存在するが、以下に示すように、φ は、あるポテンシャル問題を解くことによって求めることが出来る。

このとき、$\varphi\colon \Omega \to D_1$ が双正則写像で、(33) を満たすとしよう。Ω の閉包から閉円盤への同相写像 $\widetilde{\varphi}\colon \overline{\Omega} \to \overline{D_1}$ に拡張できる (Carathéodory の定理)。以下 $\widetilde{\varphi}$ のことも φ と書くことにする。

関数 $\dfrac{\varphi(z)}{z - z_0}$ は Ω で正則であり、かつ 0 という値を取らない。Ω が単連結であるから、$\log \dfrac{\varphi(z)}{z - z_0}$ の一価正則な分枝が取れる。その実部、虚部をそれぞれ u, v とおく:

$$u(z) + iv(z) := \log \frac{\varphi(z)}{z - z_0}.$$

両辺の実部を取ると、

$$u(z) = \log \frac{|\varphi(z)|}{|z - z_0|}.$$

$z \in \partial\Omega$ のとき、$\varphi(z) \in \partial D_1$, すなわち $|\varphi(z)| = 1$ であるから、

$$u(z) = -\log|z - z_0| \quad (z \in \partial\Omega). \tag{34}$$

一方

$$\triangle u(z) = 0 \quad (z \in \Omega). \tag{35}$$

(35), (34) は、Laplace 方程式の Dirichlet 境界値問題である。これを解いて u を求め、v を u の共役調和関数で、$v(z_0) = 0$ を満たすものとすると、φ は次のように求まる。

$$\varphi(z) = (z - z_0) \exp\left[u(z) + iv(z)\right].$$

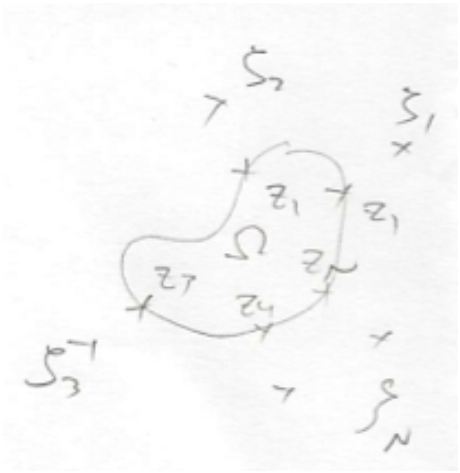

図 5:

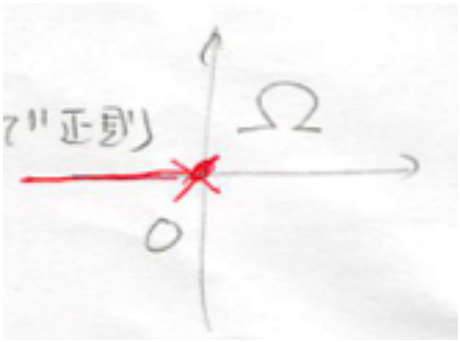

図 6:

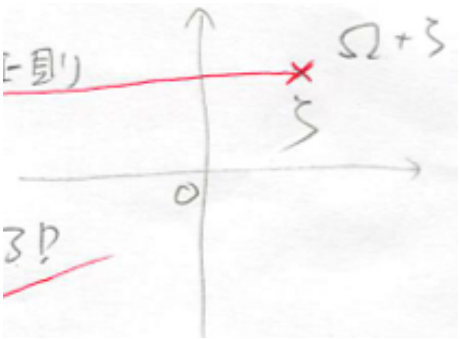

図 7:

4.4 (細かい話) 多重連結領域の場合

Ω が単連結でないときは？D_1 は単連結であるから、$\varphi\colon \Omega \to D_1$ は双正則ではありえない。

$\mathbb{C}$ の領域 Ω は、$\widehat{\mathbb{C}} \setminus \Omega$ が $n-1$ 個の連結成分からなるとき、n 重連結領域であるという。例えば $\mathbb{C} \setminus \{0\}$ や $\mathbb{C} \setminus \overline{D_1}$ は二重連結、$\mathbb{C} \setminus \{0, 1\}$ は三重連結である。(単連結は、1 連結に相当して、補集合 $\widehat{\mathbb{C}} \setminus \Omega$ は 1 個の連結成分からなる— つまり連結である。)

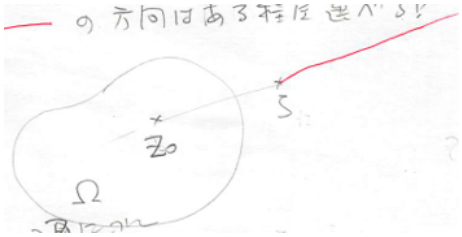

図 8:

Ω が2重連結領域である場合、1 より小さい r と、Ω から円環領域 $A(0;r,1)=\{w\in\mathbb{C}\mid r<|w|<1\}$ の上への双正則写像 $\varphi\colon\Omega\to A(0;r,1)$ が存在する。

三重連結以上の場合も、調べられている。

5 Laplace 方程式に対する基本解の方法

5.1 準備

5.1.1 基本解とは

3 次元の場合 $E(\boldsymbol{x})=\dfrac{1}{4\pi}\dfrac{1}{|\boldsymbol{x}|}$, 2 次元の場合 $E(\boldsymbol{x})=-\dfrac{1}{2\pi}\log|\boldsymbol{x}|$ を $-\triangle$ の**基本解** (the fundamental solutionn) と呼ぶ。

E は原点以外では調和関数であることは簡単な計算で分かる:

$$\triangle E(x)=0\quad(x\in\mathbb{R}^n\setminus\{0\}).$$

さらに、超関数の言葉で言うと (原点まで込めて)

$$-\triangle E=\delta \tag{36}$$

が成り立つ。ここで δ は Dirac のデルタ関数である。

物理的な解釈: 原点に置かれた単位点電荷の作る電場のポテンシャルが $E(x)$ である。

5.1.2 Poisson 方程式の特解

$$U(x):=\int_\Omega E(x-y)f(y)dy$$

とおくと、

$$-\triangle U=f$$

が成り立つ (証明は結構難しい)。つまり U は Poisson 方程式の特解である。

$v:=u-U$ とおくと、$\triangle v=0$ が成り立つので、$f=0$ の場合の問題が一般に解ければ良いことになる。(実際には U を計算することは難しいことが多く、数値計算向きではないかもしれない。)

5.1.3 Green の積分公式 (続き)

Green の第 2 積分公式

$$\int_{\partial\Omega}\left(v\frac{\partial u}{\partial n}-u\frac{\partial v}{\partial n}\right)d\sigma=\int_{\Omega}(v\triangle u-u\triangle v)\,d\boldsymbol{x}$$

を利用して、次の Green の第 3 積分公式を得る (証明の詳細は略するが、関数論の Cauchy の積分公式の証明のように、x を中心とする球を除いた領域で Green の公式を適用してから、球の半径を 0 に近づける。詳しくは桂田 [7] の §3.5 を見よ。)。

$$-\int_{\Omega}E(x-y)\triangle u(y)d\boldsymbol{y}+\int_{\partial\Omega}E(x-y)\frac{\partial u}{\partial n_y}(y)d\sigma_y-\int_{\partial\Omega}u(y)\frac{\partial}{\partial n_y}E(x-y)d\sigma_y=\begin{cases}u(x) & (x\in\Omega)\\ \frac{1}{2}u(x) & (x\in\partial\Omega)\\ 0 & (x\in\mathbb{R}^m\setminus\overline{\Omega}).\end{cases}$$

($x\in\partial\Omega$ の場合は左辺第 3 項は主値積分である。)

(a) u が $\triangle u=0$ を満たすならば、$x\in\Omega$ に対して、

$$u(x)=\int_{\partial\Omega}E(x-y)\frac{\partial u}{\partial n_y}(y)d\sigma_y-\int_{\partial\Omega}u(y)\frac{\partial}{\partial n_y}E(x-y)d\sigma_y.$$

すなわち、u が調和関数であるとき、$u, \partial u/\partial n$ の $\partial\Omega$ での値が分かれば、$u(x)$ の値がこの式で求まることになる (正則関数の Cauchy の積分公式に似ていて、使いでのある公式)。境界条件から半分は分かっているので、もう半分求めれば良いことになる。

以下は細かい話になるが: 例えば $\Gamma_N=\emptyset$ のとき、

$$\frac{1}{2}g_1(x)=\int_{\partial\Omega}E(x-y)\frac{\partial u}{\partial n_y}(y)d\sigma_y-\int_{\partial\Omega}g_1(y)\frac{\partial}{\partial n_y}E(x-y)d\sigma_y.$$

これから $\partial\Omega$ 上で $\partial u/\partial n$ を求めることが出来る。

(b) u が $\partial\Omega$ の近傍で 0 ならば、$x\in\Omega$ に対して、

$$-\int_{\Omega}E(x-y)\triangle u(y)d\boldsymbol{y}=u(x).$$

この事実を超関数解釈すると $-\triangle E(x-\cdot)=\delta(\cdot-x)$ となる。

5.2 基本解の方法

簡単のため、$\Gamma_N=\emptyset$ の場合の Laplace 方程式の境界値問題

$$\begin{aligned}\triangle u(x)&=0\quad(x\in\Omega),\\ u(x)&=g_1(x)\quad(x\in\Gamma_D=\partial\Omega)\end{aligned}$$

を取り上げる。

Ω が滑らかな境界を持つ有界領域の場合に、この境界値問題が一意的な解を持つことは知られている。

Ω が (円盤とか、長方形とか) 特別な形をしている場合に、解 u を求める公式はいくつか知られているが、ここでは多くの場合に使える数値解法を紹介する。一見素朴であるが、多くの場合に良好な近似解を得ることが出来る。

Ω の外部に Ω を「囲むように」点 $\{y_k\}_{k=1}^N$ を取り、

$$u^{(N)}(x) = \sum_{k=1}^{N} Q_k E(x - y_k)$$

とおく (もちろん、E は Laplacian の基本解である)。ここで $Q_1, Q_2, \ldots, Q_N$ は未定係数である。これらが何であっても

$$\triangle u^{(N)}(x) = 0 \quad (x \in \mathbb{R}^n \setminus \{y_1, y_2, \ldots, y_N\})$$

が成り立つ。もし $u^{(N)}(x) = g_1(x)$ $(x \in \partial\Omega)$ が成り立てば $u^{(N)}$ は解である。さすがにそんな都合の良いことはめったにおこらないが、多くの場合、境界 $\partial\Omega$ 上で選んだ点 $x_1, \ldots, x_N$ に対して

$$u^{(N)}(x_j) = g_1(x_j) \quad (j = 1, 2, \cdots, N)$$

を満たすように Q_k を決めることが出来る。このとき

$$u^{(N)} \fallingdotseq u$$

が成り立つことが期待できる。この近似解法を、the method of fundamental solutions (**基本解の方法**, fundamental solution method), あるいは代用電荷法 (**charge simulation method**) と呼ぶ[9]。

次のような利点がある。

- しばしば**誤差の指数関数的減少**

  $$(\exists C \in \mathbb{R})(\exists \rho \in (0,1))(\forall N \in \mathbb{N}) \quad \left\| u - u^{(N)} \right\| \leq C\rho^N$$

  が成り立つ (この場合は、差分法や有限要素法と比較して、高精度の解が少ない計算量で得られる)。

- $u^{(N)}$ 自身が調和関数であり、特に

  $$\operatorname{grad} u^{(N)}(x) = \begin{cases} -\dfrac{1}{2\pi} \displaystyle\sum_{j=1}^{N} Q_j \frac{x - y_j}{|x - y_j|^2} & (2 \text{ 次元の場合}) \\ -\dfrac{1}{4\pi} \displaystyle\sum_{j=1}^{N} Q_j \frac{x - y_j}{|x - y_j|^3} & (3 \text{ 次元の場合}) \end{cases}$$

  のように grad が直接計算できる (ポテンシャルの grad が必要な場合が多いので、非常に便利である)。

基本解の方法以外に、基本解を利用する方法として、**境界要素法** (boundary element method, BEM) がある。

5.3 近似等角写像を求めるための天野の方法

天野要は、§4.3 で述べた等角写像の求め方と、基本解の方法を組み合わせた、効率的なアルゴリズムを提唱した ([8])。それを解説する。

§4.3 で導入した記号を用いる。

[9] その理由は、$y_1, \ldots, y_N$ それぞれに電荷量 $Q_1, \ldots, Q_N$ の電荷を置いたとき、それら電荷の作る電場のポテンシャルが $u^{(N)}$ である、という事実に基づく。

u の近似 $u^{(N)}$ を基本解の方法で求めよう。$N \in \mathbb{N}$ に対して、$\{\zeta_k\}_{k=1}^N$ を「Ω を取り囲むように」 $\mathbb{C} \setminus \overline{\Omega}$ から選び、

$$u^{(N)}(z) := \sum_{k=1}^{N} Q_k \log|z - \zeta_k| \tag{37}$$

とおく。ここで Q_k は未知の実定数である $(k = 1, \ldots, N)$。$\{z_j\}_{j=1}^N$ を $\partial\Omega$ から選び、collocation equation

$$u^{(N)}(z_j) = -\log|z_j - z_0| \quad (j = 1, \ldots, N) \tag{38}$$

で $\{Q_k\}$ を定める。

天下りになるが、

$$f^{(N)}(z) := Q_0 + \sum_{k=1}^{N} Q_k \operatorname{Log} \frac{z - \zeta_k}{z_0 - \zeta_k}, \quad Q_0 := \sum_{k=1}^{N} Q_k \log|z_0 - \zeta_k| \tag{39}$$

とおく。ここで Log は主値を表すとする ($\mathbb{C} \setminus (-\infty, 0]$ を定義域とする)。

$$\operatorname{Re} f^{(N)}(z) = \sum_{k=1}^{N} Q_k \log|z_0 - \zeta_k| + \sum_{k=1}^{N} Q_k \log\left|\frac{z - \zeta_k}{z_0 - \zeta_k}\right| = \sum_{k=1}^{N} Q_k \log|z - \zeta_k| = u^{(N)}(z)$$

であり、

$$f^{(N)}(z_0) = Q_0 + \sum_{k=1}^{N} Q_k \operatorname{Log} \frac{z_0 - \zeta_k}{z_0 - \zeta_k} = Q_0 + \sum_{k=1}^{N} 0 = Q_0 \in \mathbb{R}.$$

言い換えると $\operatorname{Im} f^{(N)}(z_0) = 0$. この $f^{(N)}$ は、$f = u + iv$ の良い近似であると考えられる。

天野のアルゴリズム

(1) (37), (38) で $\{Q_k\}$ を定める。

(2) (39) で $f^{(N)}$ を定める。

(3) $\varphi^{(N)}(z) := (z - z_0) \exp f^{(N)}(z)$ で定義される $\varphi^{(N)}$ を、等角写像 $\varphi\colon \Omega \to D_1$ の近似として採用する。

5.4 Eigen ライブラリィを用いた Jordan 領域の等角写像の計算プログラム

個人的には、この手の計算には MATLAB を使うのが好みであるが、ここでは、C++ でベクトル、行列に関する演算をするために便利なクラス・ライブラリィである、**Eigen**[10] を利用してみる (なお、C++ では、複素数を使うのも簡単である)。

WWW サイトから、`eigen-eigen-5a0156e40feb.tar.gz` のような名前のソース・ファイル一式を入手して、以下のようにインストールする。

[10] `http://eigen.tuxfamily.org/`

MacBook のターミナルで次のようにインストール

```
tar xzf eigen-eigen-5a0156e40feb.tar.gz
cd eigen-eigen-5a0156e40feb
ls
cp -pr Eigen ~/include
```

(アーカイブ･ファイルに含まれていた、Eigen というディレクトリィを、~/include の下にコピーしている。これは、現象数理学科 Mac を想定している。普通は、/usr/local/include などにコピーするのかも。)

(脱線になるけれど、Eigen を用いた、Runge-Kutta 法のサンプル・プログラム http://nalab.mind.meiji.ac.jp/~mk/complex2/ball-bound.cpp を紹介しておく。コンパイルの仕方、実行の仕方は、注釈に書いてある。)

$\Omega = D_1 = D(0;1)$, $z_0 = 1/2$ の場合を解いてみよう。つまり双正則な

$$\varphi\colon \Omega \to D_1$$

で、

$$\varphi(z_0) = 0, \quad \varphi'(z_0) > 0$$

を満たすものを求める。この場合は、1 次分数変換

$$\varphi(z) = \frac{z - z_0}{1 - \overline{z_0}\, z}$$

が解であることが分かっている。

conformalmap.cpp[11] — 例えばターミナルで

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/complex2/conformalmap.cpp
```

としてダウンロード出来る。

```
/*
 * conformalmap.cpp --- 等角写像の数値計算例（基本解の方法の天野要氏バージョン）
 *
 * g++ -I/opt/X11/include -I/usr/local/include conformalmap.cpp
 *   -L/usr/local/lib -lglscd -L/opt/X11/lib -lX11
 *  これは /opt/X11 に X 関係のファイルが、
 *         /usr/local に Eigen, GLSC 関係のファイルが
 *  あると仮定している。
 *
 *   現象数理学科 Mac では、GLSC は ~/include, ~/lib にインストールして
 * あるだろうから、次のようにするのかな。
 * g++ -I/opt/X11/include -I ~/include conformalmap.cpp -L ~/lib -lglscd -L/opt/X11/lib -lX11
 * その場合は、Eigen も ~/include にインストールすると良い。
 * まあ、適宜やって下さい。
 */

#include <iostream>
#include <complex>
#include <Eigen/Dense>

extern "C" {
#include <glsc.h>
```

[11] http://nalab.mind.meiji.ac.jp/~mk/complex2/conformalmap.cpp

```
};

using namespace std;
using namespace Eigen;

// MatrixXd は大きさが実行時に指定できて、成分が double の行列
// d(double) の代わりに i(int), f(float), cf, cd (complex<double>) も OK

// φ(z0)=0, φ'(z0)>0 を満たす等角写像
complex<double> phi(complex<double> z,
    complex<double> z0)
{
  complex<double> w;
  w = (z-z0)/(1.0-conj(z0)*z);
  return w;
}

// 天野の方法による等角写像
complex<double> f(complex<double> z,
  complex<double> z0,
  int N,
  double Q0,
  VectorXd Q, VectorXcd zeta)
{
  int k;
  complex<double> s;
  s = Q0;
  for (k = 0; k < N; k++)
    s += Q(k) * log((z-zeta(k))/(z0-zeta(k)));
  return (z-z0) * exp(s);
}

int main(void)
{
  int i, j, k, N, m;
  double theta, pi, dt, R, Q0, r, dr;
  // 虚数単位と原点に移る点 z0=1/2
  complex<double> I(0,1), z0(0.6,0.0), zp, w, w2;

  R = 2;
  N = 80;
  VectorXcd zeta(N), z(N);
  VectorXd b(N), Q(N);
  MatrixXd a(N,N);

  pi = 4.0 * atan(1.0);
  dt = 2 * pi / N;
  // 円周 |z|=2 上の等分点
  for (k = 0; k < N; k++)
    zeta(k) = R * exp(I * (k * dt));
  //  cout << "zeta=" << zeta << endl;
  // 単位円周 |z|=1 上の等分点
  for (j = 0; j < N; j++)
    z(j) = exp(I * (j * dt));

  // 係数行列の準備
  for (j = 0; j < N; j++)
    for (k = 0; k < N; k++) {
      a(j,k) = log(abs(z(j)-zeta(k)));
    }
  // 右辺
  for (j = 0; j < N; j++) {
    b(j) = - log(abs(z(j)-z0));
```

```
  }
  // 電荷を求める
  Q = a.partialPivLu().solve(b);

  // Q0 を求める
  Q0 = 0;
  for (k = 0; k < N; k++) {
    Q0 += Q(k) * log(abs(z0-zeta(k)));
  }

  g_init((char *)"GRAPH", 150.0, 150.0);
  g_device(G_BOTH);
  g_def_scale(0, - 1.2, 1.2, -1.2, 1.2, 10.0, 10.0, 140.0, 140.0);
  g_sel_scale(0);

  m = 20;
  // 同心円の像
  dr = 1.0 / m;
  dt = 2 * pi / (4 * m);
  for (i = 1; i <= m; i++) {
    r = i * dr;
    for (j = 0; j <= 4 * m; j++) {
      zp = r * exp(I * (j * dt));
      w=f(zp,z0,N,Q0,Q,zeta);
      if (j == 0)
        g_move(w.real(), w.imag());
      else
        g_plot(w.real(), w.imag());
    }
  }
  // 放射線の像
  for (j = 0; j <= 4 * m; j++) {
    theta = j * dt;
    for (i = 0; i <= m; i++) {
      r = i * dr;
      zp = r * exp(I * theta);
      w=f(zp,z0,N,Q0,Q,zeta);
      if (i == 0)
        g_move(w.real(), w.imag());
      else
        g_plot(w.real(), w.imag());
    }
  }
  g_sleep(-1.0);
  g_term();
  return 0;
}
```

こんな風にコンパイル

```
g++ -I/opt/X11/include -I ~/include conformalmap.cpp -L ~/lib -lglscd -L/opt/X11/lib -lX11
```

(GLSC を利用していて、現象数理学科 Mac では、`~/include`, `~/lib` にインストールされていることを想定している。上の手順で、Eigen も `~/include` にインストールしておいた。)

参考文献

[1] 桂田祐史：Poisson 方程式に対する差分法, http://nalab.mind.meiji.ac.jp/~mk/labo/text/poisson.pdf (2000 年?〜).

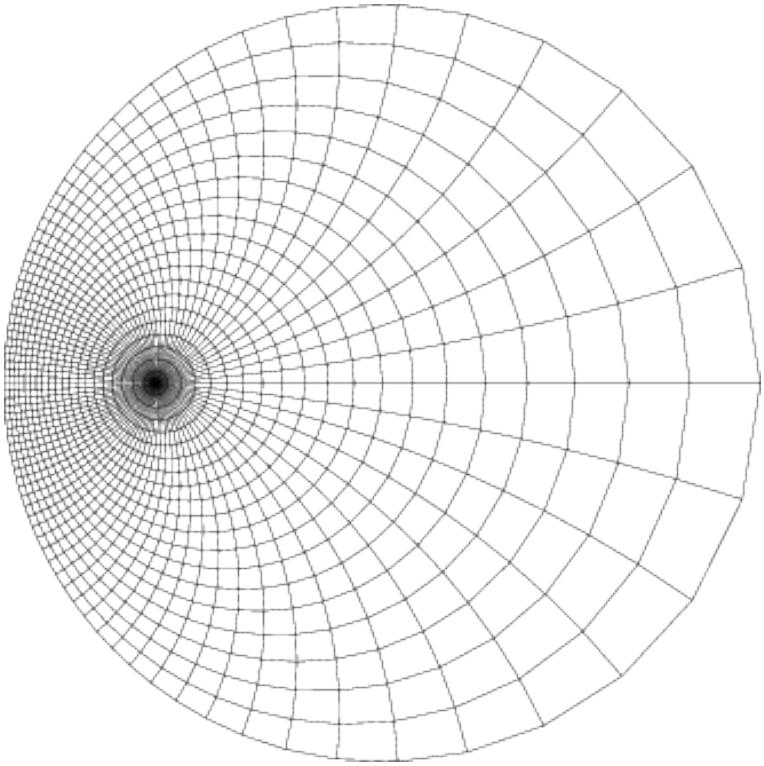

図 9: $w = \dfrac{z - z_0}{1 - \overline{z_0} z}$, $z_0 = 0.6$ による z 平面の同心円、放射線の像を描いた。

[2] 桂田祐史：MATLAB 使い方入門, http://nalab.mind.meiji.ac.jp/~mk/labo/text/matlab.pdf (HTML 版もある) (2005 年〜).

[3] 桂田祐史：長方形領域における熱方程式に対する差分法 — MATLAB を使って数値計算—, http://nalab.mind.meiji.ac.jp/~mk/labo/text/heat2d.pdf (2015 年〜).

[4] 桂田祐史：Neumann 境界条件下の熱方程式に対する差分法 — MATLAB を使って数値計算 —, http://nalab.mind.meiji.ac.jp/~mk/labo/text/heat2n.pdf (2015 年〜).

[5] 桂田祐史：FreeFEM++ の紹介, http://nalab.mind.meiji.ac.jp/~mk/labo/text/welcome-to-freefem-2012/ (2007〜).

[6] 菊地文雄：有限要素法概説, サイエンス社 (1980), 新訂版 1999.

[7] 桂田祐史：微分方程式 2 講義ノート (旧「応用解析 II」), http://nalab.mind.meiji.ac.jp/~mk/lecture/pde/pde2013.pdf (1997 年〜).

[8] 天野 要（かなめ）：代用電荷法に基づく等角写像の数値計算法, 情報処理学会論文誌, Vol. Vol. 28, No. 27, pp. 697–704 (1987).