応用複素関数 第 10 回

～ ポテンシャル問題 (3) ～

かつらだ まさし
桂田 祐史

2021 年 6 月 22 日

目次

本日の内容・連絡事項

- ポテンシャル問題の数値解法として有力な**基本解の方法**を解説する。
  - 電磁気学の簡単な知識 (クーロンの法則など) が必要である。
  - Laplacian の基本解は、基礎的な概念であるので、偏微分方程式の入門書の多くで説明されている (例えば、桂田の講義ノート [1] の §3.5)。
  - 基本解の方法については、桂田 [2] というノートがある (このスライドに書いてあることと大差ない)。

(有限要素法、FreeFem++ は汎用性が高く、紹介する価値が高いが、ポテンシャル問題に限れば最善の方法というわけでなく、基本解を活用する方法にも触れておきたい、関数論の立場ではこちらがより重要である。)

4.8 ポテンシャル問題の数値解法 (2) 基本解の方法
4.8.1 $-\triangle$ の基本解

次の関数 E は、$-\triangle$ の**基本解** (fundamental solution) と呼ばれる。

$$(1) \qquad E(x) := \begin{cases} -\dfrac{1}{2\pi}\log|x| & (2\text{ 次元の場合, } x \in \mathbb{R}^2 \setminus \{0\}) \\ \dfrac{1}{4\pi}\dfrac{1}{|x|} & (3\text{ 次元の場合, } x \in \mathbb{R}^3 \setminus \{0\}). \end{cases}$$

一体何者か？

数学的な解答　次を満たす。ここで δ は **Dirac のデルタ超関数**。

$$(2) \qquad -\triangle E = \delta \qquad (\triangle \overset{\text{def.}}{=} \sum_{j=1}^{n} \frac{\partial^2}{\partial x_j^2}).$$

物理的な解答 (解釈)　E は原点にある単位点電荷の作る電場のポテンシャル (電位) である。

(Cf. 密度 ρ で分布する場合のポテンシャル u は $-\triangle u = \rho$ を満たす。)

4.8.1 $-\triangle$ の基本解

なぜ基本解は重要か？重ね合わせることで “任意” の電荷分布 ρ のポテンシャルが得られる。

定理 (のようなもの)　Poisson 方程式の特解

$\Omega \subset \mathbb{R}^n$, $\rho\colon \Omega \to \mathbb{R}$ が与えられたとき

$$u(x) := \int_\Omega E(x-y)\rho(y)\,dy \quad (x \in \Omega) \tag{3}$$

とおくと次式が成り立つ。

$$-\triangle u = \rho \quad (\text{in } \Omega). \tag{4}$$

E には特異性があるので、(4) を証明するのは少し難しい (神保 [3] など)。

物理的には次のように納得できる。

微小体積 dy に存在する電荷は $\rho(y)\,dy$ で、それが作る電場のポテンシャルは (基本解を平行移動したものの電荷量倍で) $E(x-y)\rho(y)\,dy$. それを Ω 全体でトータルした u がポテンシャルになる。実際、$\boldsymbol{E} := -\operatorname{grad} u$ は電場で、Maxwell の方程式の 1 つ $\operatorname{div} \boldsymbol{E} = \rho$ から、$-\operatorname{div}\operatorname{grad} u = \rho$ が得られる。すなわち (4) が成り立つ。

ポテンシャル問題の数値解法 (近似解法) への応用「**基本解の方法** (the method of fundamental solutions)」を紹介する。

Dirichlet 境界値問題を考えよう (Neumann 境界値問題でも同様)。

$$\triangle u = 0 \quad (\text{in } \Omega) \tag{5}$$

$$u = g \quad (\text{on } \partial\Omega). \tag{6}$$

ここで Ω は $\mathbb{R}^n$ ($n = 2$ or $n = 3$) の領域である。

Ω の外部に、Ω を取り囲むように、有限個の点 $y_1, \cdots, y_N$ を取り、各 y_k に電荷量 Q_k の電荷を置く。

4.8.2 アルゴリズム (電荷の作る電場の電位でポテンシャルを近似)

それらの電荷の作る電場のポテンシャルは

$$u^{(N)}(x) := \sum_{k=1}^{N} Q_k E(x - y_k). \tag{7}$$

$\triangle E = 0$ (in $\mathbb{R}^n \setminus \{0\}$) であるから、$Q_k$ の取り方によらず

$\triangle u^{(N)}(x) = 0 \quad (x \in \mathbb{R}^n \setminus \{y_1, \cdots, y_N\})$. 特に $\triangle u^{(N)} = 0$ (in Ω).

後は Q_k をうまく選んで、境界条件 (6) $u = g$ (on $\partial\Omega$) を近似的に満たすようにする。

一つのやり方として、$\partial\Omega$ 上に N 個の点 $x_1, \cdots, x_N$ を取って

$$u^{(N)}(x_j) = g(x_j) \quad (j = 1, \cdots, N). \tag{8}$$

これで Q_k $(k = 1, \cdots, N)$ が定まることはすぐ分かる (次のスライド)。

非常に素朴な感じがするが、とてもうまく行くことが多い。

4.8.2 アルゴリズム (電荷の作る電場の電位でポテンシャルを近似)

(8) は次の連立 1 次方程式と同値である。

$$\begin{pmatrix} E(x_1-y_1) & E(x_1-y_2) & \cdots & E(x_1-y_N) \\ E(x_2-y_1) & E(x_2-y_2) & & E(x_2-y_N) \\ \vdots & & \ddots & \vdots \\ E(x_N-y_1) & E(x_N-y_2) & \cdots & E(x_N-y_N) \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_N \end{pmatrix} = \begin{pmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_N) \end{pmatrix}.$$

Gauss の消去法などを用いて、Q_k $(k = 1, \cdots, N)$ が求められる。

(いわゆる密行列であるが、それほど大きな N は必要ないので、難しくない。)

4.8.3 基本解の方法の特徴

1. ある ρ $(0 < \rho < 1)$, C が存在して
$$\left\| u - u^{(N)} \right\| \leq C\rho^N \quad (\|\cdot\| \text{ は適当なノルム})$$
が成り立つ (**誤差の指数関数的減少**, 次のスライドで数値例を示す)。しばしば、**高精度の解が非常に少ない計算量で得られる**ことが期待できる。
Cf. 差分法, 有限要素法では、典型的な場合に $\left\| u - u^{(N)} \right\| \leq \frac{C}{N^2}$.
2. $u^{(N)}$ は調和関数である。特に $\operatorname{grad} u^{(N)}$ の計算が簡単:
$$\operatorname{grad} u^{(N)}(x) = -\frac{1}{2\pi} \sum_{k=1}^{N} Q_k \frac{x - y_k}{|x - y_k|^2} \quad (2 \text{ 次元の場合}).$$
(例えばポテンシャル流の計算を思い浮かべると、超便利と分かる。)
しかも $\left\| \operatorname{grad} u - \operatorname{grad} u^{(N)} \right\|$ も指数関数的に減少する。
Cf. 差分法や有限要素法では、微分が難しかったり、精度が下がったりする。
3. 理論的な基礎づけは、差分法、有限要素法と比べて不十分である。
4. 同次方程式にしか適用できない、具体的な基本解が必要 → 汎用性は低い。

汎用性低いが、使えるときは、差分法・有限要素法に性能で勝る場合が多い。

4.8.3 基本解の方法の特徴　数値例

Ω が円盤 $\left\{\boldsymbol{x} \in \mathbb{R}^2 \,\middle|\, |\boldsymbol{x}| < 1\right\}$ の場合に、原点中心半径 $R = 2$ の円周上に一様に電荷点 y_k を配置した場合の近似解の精度を示す。

左が $g(\boldsymbol{x}) = \mathrm{Re}\,[(x+iy)^m]$, 右が $g(\boldsymbol{x}) = \log|\boldsymbol{x} - \boldsymbol{p}|$ $(\boldsymbol{p} = (p, 0))$ の場合。

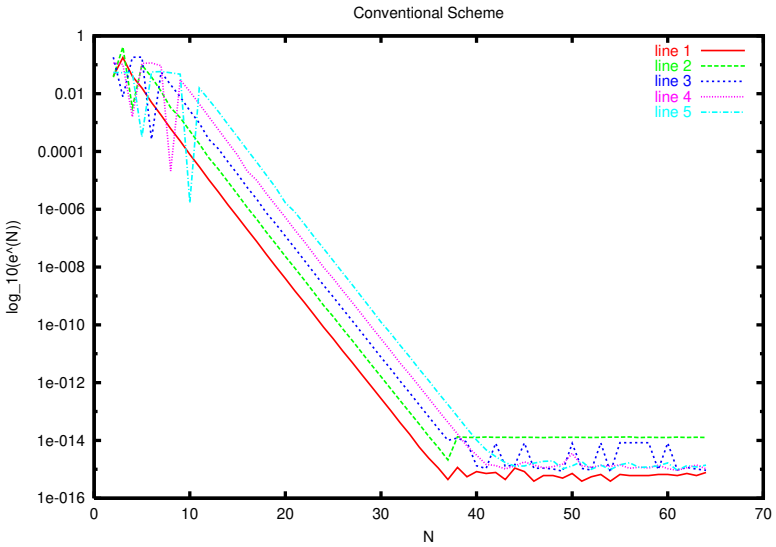

図 1: $m = 1, \cdots, 5$

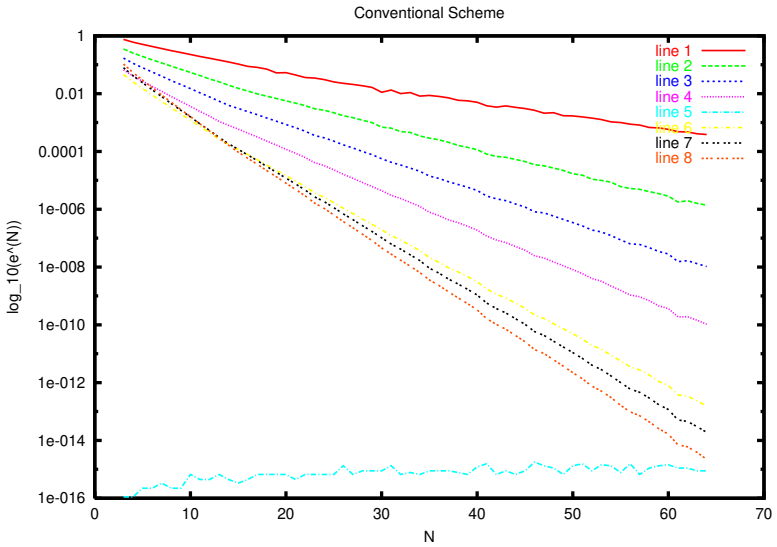

図 2: $p = 1.2, 1.4, \cdots, 2.6$. p が大きい (特異点が遠い) ほど速く減衰

誤差の減少は非常に速い！

片側対数目盛で、直線上にのることから、誤差が指数関数的に減少している。減少の速さについては研究されていて、ある程度まで説明可能である。

4.8.4 数値等角写像に対する天野の方法

天野要は、§4.4 で述べた等角写像の求め方 (定理 8.6) と、基本解の方法を組み合わせた、数値等角写像 (領域の写像関数を数値的に求めること) の効率的なアルゴリズムを提唱した (天野 [4])。それを解説する。

§4.4 で導入した記号を用いる。

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、

$$u^{(N)}(z) := \sum_{k=1}^{N} Q_k \log|z - \zeta_k| \tag{9}$$

とおく。ここで Q_k $(k = 1, \ldots, N)$ は未知の実定数である。

$\{z_j\}_{j=1}^N$ を $\partial\Omega$ から選び、連立 1 次方程式

$$u^{(N)}(z_j) = -\log|z_j - z_0| \quad (j = 1, \ldots, N) \tag{10}$$

を解いて Q_k $(k = 1, \ldots, N)$ が求められる。

4.8.4 数値等角写像に対する天野の方法

$u^{(N)}$ の共役調和関数 $v^{(N)}$ を求めたい ($u^{(N)}$ を実部に持つ正則関数を求めたい)。天下りになるが、

$$f^{(N)}(z) := Q_0 + \sum_{k=1}^{N} Q_k \operatorname{Log} \frac{z-\zeta_k}{z_0-\zeta_k}, \qquad Q_0 := \sum_{k=1}^{N} Q_k \log|z_0-\zeta_k| \tag{11}$$

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)。

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

である。さらに

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

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。

4.8.4 数値等角写像に対する天野の方法

以上をまとめると、次のアルゴリズムが得られる。

(再掲 9) $$u^{(N)}(z) := \sum_{k=1}^{N} Q_k \log|z-\zeta_k|,$$

(再掲 10) $$u^{(N)}(z_j) = -\log|z_j - z_0| \quad (j=1,\dots,N),$$

(再掲 11) $$f^{(N)}(z) := Q_0 + \sum_{k=1}^{N} Q_k \operatorname{Log}\frac{z-\zeta_k}{z_0-\zeta_k}, \quad Q_0 := \sum_{k=1}^{N} Q_k \log|z_0-\zeta_k|$$

天野のアルゴリズム

0. $\{\zeta_k\}_{k=1}^N \subset \mathbb{C}\setminus\overline{\Omega}$, $\{z_j\}_{j=1}^N \subset \partial\Omega$ を適当に選ぶ。
1. (9), (10) で $\{Q_k\}$ を求める。
2. (11) 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。

4.8.5 Jordan 領域の等角写像の計算プログラム

以下の C++プログラム `conformalmap.cpp` では

$$\Omega = D_1 \overset{\text{def.}}{=} \{z \in \mathbb{C} \mid |z| < 1\}, \quad z_0 = \frac{1}{2}$$

の場合の Ω の写像関数、すなわち双正則な $\varphi\colon \Omega \to D_1$ で

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

を満たすものを求める。この場合、実は次の 1 次分数変換が解である。

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

プログラム入手 — ターミナルで次を実行

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

コンパイル — ターミナルで次のコマンド (1 行) を実行

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

4.8.5 Jordan 領域の等角写像の計算プログラム

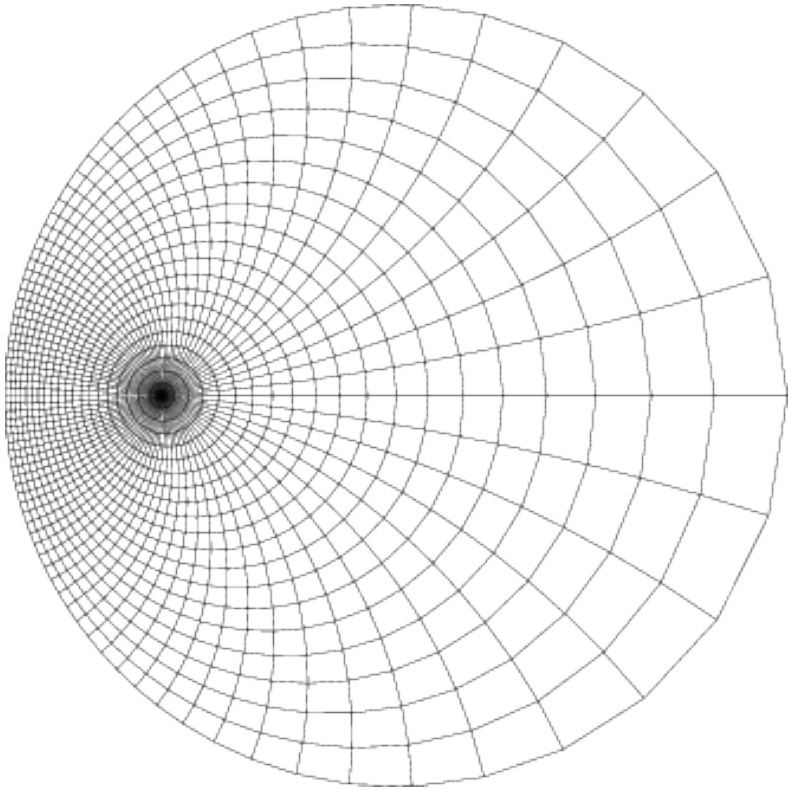

図 3: $z_0 = 0.6$ の場合. $w = \varphi(z)$ による z 平面の原点中心の同心円、原点を始点とする半直線の像を描いた。

4.8.5 Jordan 領域の等角写像の計算プログラム　おわび

`conformalmap.cpp` では、次のライブラリィを用いている。

- Eigen: ベクトル・行列演算をするために便利な C++クラス・ライブラリィ
- GLSC: C&C++&Fortran 用のグラフィックス・ライブラリィ
  注: 2021 年度の 3 年生の現象数理学科 Mac には、GLSC がインストールされていないので、実行するのに手間がかかる (参考プログラムという扱いになる)。

一応、付録でこれらのインストール方法を説明しておくが、手こずる可能性が大きいので (特に GLSC)、サンプル・プログラムの実行を強くは勧めない。

他のプログラミング言語 (MATLAB, Julia, Python) でサンプル・プログラムを書き直すべきでしたが、時間切れとなりました。

補足: Eigen と GLSC のインストール

Eigen のサイト から、eigen-3.3.9.zip のような zip ファイルを入手して、ターミナルで以下のようなコマンドを入力すればインストールできる。

```
unzip eigen-3.3.9.zip
cd eigen-3.3.9
sudo mkdir -p /usr/local/include
tar cf - Eigen | (cd /usr/local/include; sudo tar xzf -)
```

もちろんバージョン番号 3.3.9 は適当に修正する。sudo するのでパスワードを尋ねられる。

/usr/local/include にコピーしたので、コンパイルするとき、-I/usr/local/include というオプションをつけることになる。

補足: GLSC のインストール (慣れない人には大変かも)

以下の 3 つのファイルを入手する (curl -O でできる)。

(a) http://www602.math.ryukoku.ac.jp/~nakano/software/math/glsc-3.5.a.tar.Z

(b) http://nalab.mind.meiji.ac.jp/~mk/daishin/glsc-3.5+a.patch

(c) http://nalab.mind.meiji.ac.jp/~mk/program/graphics/glsc-3.5+mk.patch20201229

ライブラリィをコンパイルしてインストール

```
tar xzf glsc-3.5.a.tar.Z
cd glsc-3.5.a
patch -p1 < ../glsc-3.5+a.patch
patch -p1 < ../glsc-3.5+mk.patch20201229
```

ここで必要があるならば、 Makefile の FC を gfortran にセットする。

```
make>&make.log
sudo mkdir -p /usr/local/bin /usr/local/include /usr/local/lib
sudo make install >& make-install.log
```

補足: サンプル・プログラムのコンパイル

この補足に書いてある仕方で GLSC をインストールした場合、コンパイルの仕方が本文とは少し違うので説明しておく。

コンパイル — ターミナルで次のコマンド (1 行) を実行

```
c++ -I/usr/local/include -I/opt/X11/include conformalmap.cpp \
 -L/usr/local/lib -lglscd -L/opt/X11/lib -lX11
```

行末の \ は行が継続することを意味していて、実際には 1 つのコマンドである。

- Eigen も GLSC も、ヘッダーファイルを `/usr/local/include` にコピーしたので、`-I/usr/local/include` を指定してある。
- GLSC のライブラリィは、`/usr/local/lib` にコピーしたので、`-L/usr/local/include -lglscd` を指定してある。
- `-I/opt/X11/include`, `-L/opt/X11/lib -lX11` は、X Window System (GLSC から呼び出す) を使うために指定する。

参考文献

[1] 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〜).

[2] 桂田祐史：ポテンシャル問題の数値計算, http://nalab.mind.meiji.ac.jp/~mk/complex2/potential.pdf (2017).

[3] 神保秀一：偏微分方程式入門, 共立出版 (2006).

[4] 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. 28, No. 27, pp. 697–704 (1987).