計算物理学および演習

偏微分方程式の数値解法 2

畝山多加志

1　多次元境界値問題

前回は 1 次元の境界値問題の数値スキームについて考えたが、実際には空間は 3 次元であるため、物理で表れる問題の多くは 1 次元ではなく 3 次元の境界値問題となっている。1 次元の場合と 2 次元以上の多次元の問題では空間の方向が複数存在するために問題の性質が大きく異なってくる。そのため、多次元境界値問題は 1 次元の境界値問題とは少し違う考え方をせねばならない部分がある。

1.1　2 次元空間での差分化・離散化

ここでは例として 2 次元空間内の Laplace-Poisson 方程式の場合を考える。(3 次元、あるいはそれ以上でも本質的にはあまり大きく変わらない。) このとき、Laplace-Poisson 方程式

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = \phi(x,y) \tag{1}$$

を境界条件

$$u(0,y) = u(L_x,y) = u(x,0) = u(x,L_y) = 0 \tag{2}$$

のもとで解くことを考える。簡単のため境界条件は $x = 0, L_x$ および $y = 0, L_y$ で値がゼロとなることとしたが、より一般的な境界条件を課しても以下で説明するスキームはほとんど同じように使うことができる。

1 次元空間のときと同様にまず離散化を行うが、空間は x, y の 2 つの座標で表現されるため、x, y を両方とも差分化する必要がある。

$$x_j = \Delta_x j \qquad (j = 0, 1, 2, \ldots, N_x) \tag{3}$$
$$y_k = \Delta_y k \qquad (k = 0, 1, 2, \ldots, N_y) \tag{4}$$

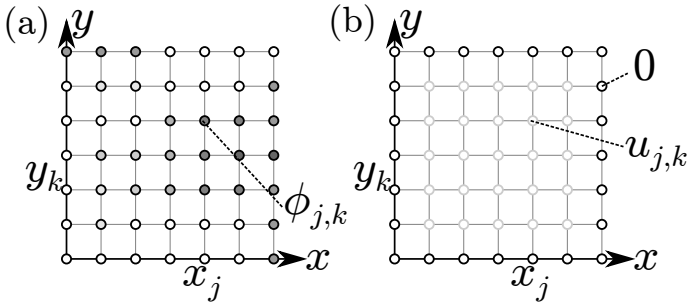

図 1: 2 次元境界値問題の離散化のイメージ。x, y 方向それぞれを離散化する。(a) 関数 $\phi(x,y)$ は離散化して $\phi_{j,k}$ (丸の色が値に対応する) で表現できる。(b) $u(x,y)$ も同様に離散化する。境界条件から、$u_{j,k}$ の端の部分は全て 0 になる。残りの部分の $u_{j,k}$ (灰色の線の丸) を求める。

Δ_x, Δ_y は x, y 方向それぞれの刻みであり、Δ_x と Δ_y は一致していなくても良い。1 次元空間のときと同様に、これらは

$$\Delta_x N_x = L_x, \qquad \Delta_y N_y = L_y \tag{5}$$

を満たすように決められているものとする。$u(x, y)$ および $\phi(x, y)$ はこれらの離散化した点に対して

$$u_{j,k} = u(x_j, y_k), \qquad \phi_{j,k} = \phi(x_j, y_k) \tag{6}$$

と離散化してやればよい (図 1)。

続いて微分の差分化を行う。x, y 方向それぞれの偏微分に対して前回導入した中心差分を適用してやれば、

$$\left.\frac{\partial^2 u(x,y)}{\partial x^2}\right|_{x_j, y_k} \approx \frac{u_{j+1,k} - 2u_{j,k} + u_{j-1,k}}{\Delta_x^2} \tag{7}$$

$$\left.\frac{\partial^2 u(x,y)}{\partial y^2}\right|_{x_j, y_k} \approx \frac{u_{j,k+1} - 2u_{j,k} + u_{j,k-1}}{\Delta_y^2} \tag{8}$$

とできる。

1.2 2 次元境界値問題の解法

以上で離散化・差分化を一通り終えたので、2 次元の偏微分方程式を解くための方法を考える。中心差分を適用すれば

$$\frac{u_{j+1,k} - 2u_{j,k} + u_{j-1,k}}{\Delta_x^2} + \frac{u_{j,k+1} - 2u_{j,k} + u_{j,k+1}}{\Delta_y^2} = \phi_{j,k} \tag{9}$$

となるから、変形して

$$u_{j+1,k} - 2u_{j,k} + u_{j-1,k} + \alpha(u_{j,k+1} - 2u_{j,k} + u_{j,k+1}) = \Delta_x^2 \phi_{j,k} \tag{10}$$

ただし、$\alpha = \Delta_x^2/\Delta_y^2$ とする。1 次元境界値問題の場合、この方程式を行列を使って表現することで直接法のスキームを求めた。多次元でも同様に行列を使って表現してみる。まず、ベクトル $\boldsymbol{u}, \boldsymbol{b}$ を

$$\boldsymbol{u} = \begin{bmatrix} u_{1,1} \\ \vdots \\ u_{N_x-1,1} \\ u_{1,2} \\ \vdots \\ u_{N_x-1,2} \\ \vdots \\ u_{N_x-1,N_y-1} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} \Delta_x^2\phi_{1,1} \\ \vdots \\ \Delta_x^2\phi_{N_x-1,1} \\ \Delta_x^2\phi_{1,2} \\ \vdots \\ \Delta_x^2\phi_{N_x-1,2} \\ \vdots \\ \Delta_x^2\phi_{N_x-1,N_y-1} \end{bmatrix} \tag{11}$$

とする。方程式は線形なので、1 次元と同様に 行列 $\boldsymbol{A}$ を導入して

$$\boldsymbol{A} \cdot \boldsymbol{u} = \boldsymbol{b} \tag{12}$$

とできる。ただし、行列 $\boldsymbol{A}$ は 1 次元の場合とは異なり三重対角行列ではなくなってしまい、対角要素から離れた部分にも値を持つような行列となる。

$$\boldsymbol{A} = \begin{bmatrix} -2-2\alpha & 1 & 0 & \dots & \alpha & 0 & 0 & \dots & 0 \\ 1 & -2-2\alpha & 1 & \dots & 0 & \alpha & 0 & \dots & 0 \\ 0 & 1 & -2-2\alpha & \dots & 0 & 0 & \alpha & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha & 0 & 0 & \dots & -2-2\alpha & 1 & 0 & \dots & \alpha \\ 0 & \alpha & 0 & \dots & 1 & -2-2\alpha & 1 & \dots & 0 \\ 0 & 0 & \alpha & \dots & 0 & 1 & -2-2\alpha & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha & 0 & 0 & \dots & \alpha & 0 & 0 & \dots & -2-2\alpha \end{bmatrix} \tag{13}$$

三重対角行列ではないため、Thomas 法は適用できない。さらに、行列 $\boldsymbol{A}$ は $(N_x-1)(N_y-1)\times(N_x-1)(N_y-1)$ というサイズになるため、Gauss の消去法をベースにして $\boldsymbol{A}^{-1}\cdot\boldsymbol{b}$ を求めるのはあまりに効率が悪い。3 次元ではさらに状況は悪くなる。そのため、2 次元以上の多次元の場合には Thomas 法のようなスキームはほとんど使われない。

一方、1 次元の場合の反復法は行列 $\boldsymbol{A}$ が三重対角行列であるという性質を使っていなかった。これは多次元の場合でも同様である。従って、反復法は 2 次元以上の多次元の場合にも特に問題なく適用できる。今の問題では、解が満たすべき方程式は

$$u_{j,k} = \frac{u_{j+1,k}+u_{j-1,k}+\alpha u_{j,k+1}+\alpha u_{j,k-1}-\Delta_x^2\phi_{j,k}}{2+2\alpha} \tag{14}$$

と書ける。1 次元の場合と同様にして適当な $u_{j,k}$ の値が与えられたとして、$u_{j,k}$ を解に近い新しい値 $u^*_{j,k}$ に更新するという処理を繰り返せば良い。Jacobi 法を適用すれば

$$u^*_{j,k} = \frac{u_{j+1,k}+u_{j-1,k}+\alpha u_{j,k+1}+\alpha u_{j,k-1}-\Delta_x^2\phi_{j,k}}{2+2\alpha} \tag{15}$$

である。更新は誤差が許容限界 ϵ より小さい値に収まるまで、すなわち

$$\max_{j,k}|u^*_{j,k}-u_{j,k}| \le \epsilon \tag{16}$$

を満たすまで更新を繰り返せばよい。

Gauss-Seidel 法や SOR 法も 1 次元の場合と同様に適用できる。

$$u^*_{j,k} = \frac{u_{j+1,k}+u^*_{j-1,k}+\alpha u_{j,k+1}+\alpha u^*_{j,k-1}-\Delta_x^2\phi_{j,k}}{2+2\alpha} \tag{17}$$

あるいは緩和パラメータ ω $(0<\omega<2)$ を導入して

$$u^*_{j,k} = \omega\frac{u_{j+1,k}+u^*_{j-1,k}+\alpha u_{j,k+1}+\alpha u^*_{j,k-1}-\Delta_x^2\phi_{j,k}}{2+2\alpha}+(1-\omega)u_{j,k} \tag{18}$$

とすればよい。

ところで、これらの方法では $u^*_{j,k}$ を求める際、既に更新された値を使うため、j,k の値は

$$(j,k) = (1,1),(2,1),(3,1),\dots(N_x-1,1),(1,2),\dots,(N_x-1,2),\dots,(N_x-1,N_y-1) \tag{19}$$

のような順番で取っていく。(j,k は逆でもよいが、値は「順に」増えていくとする。) 実はこのような順番で点を取るよりも互い違いに点を取るような形としたほうが効率が上がることが知られている。離散化した位置に対してちょうどチェスの盤のように互い違いに白と黒で色を付け、最初に白だけ、続いて黒だけ、という形で j,k の値を取るとよい。N_x,N_y が偶数として、j,k を上記の順の代わりに

$$\begin{aligned}(j,k) =&(1,1),(3,1),(5,1)\dots(N_x-1,1),(2,2),(4,2),\dots,(N_x-1,3),\dots,(N_x-1,N_y-1)\\ &(2,1),(4,1),(6,1)\dots(N_x-2,2),(1,2),(3,2),\dots,(N_x-2,4),\dots,(N_x-2,N_y-2)\end{aligned} \tag{20}$$

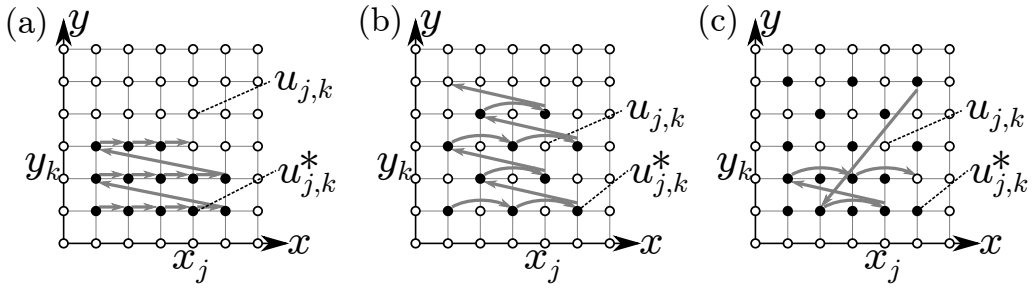

図 2: Gauss-Seidel 法と SOR 法における赤黒選択。(a) 通常は $u_{j,k}$ を順に更新していく。白い丸が $u_{j,k}$、黒い丸が更新した $u^*_{j,k}$。灰色の矢印が更新の順番を表す。(b) 赤黒選択を用いる場合、1 つ飛ばしに更新をしていく。(c) 最後まで進んだら一度戻って、更新していなかった点を更新していく。このとき、周囲の点は全て更新済みとなっている。

のような順番で取っていく。$(j,k) = (N_x - 1, N_y - 1)$ までは周囲の点はまったく更新されていないが、$(j,k) = (2,1)$ 以降は周囲の点は全て既に更新済みとなる。Gauss-Seidel 法にこの順番を適用してやれば、

$$\begin{aligned} u^*_{j,k} &= \frac{u_{j+1,k} + u_{j-1,k} + \alpha u_{j,k+1} + \alpha u_{j,k-1} - \Delta_x^2 \phi_{j,k}}{2 + 2\alpha} \\ &\qquad ((j,k) = (1,1), (3,1), \ldots, (N_x - 1, N_y - 1)) \\ u^*_{j,k} &= \frac{u^*_{j+1,k} + u^*_{j-1,k} + \alpha u^*_{j,k+1} + \alpha u^*_{j,k-1} - \Delta_x^2 \phi_{j,k}}{2 + 2\alpha} \\ &\qquad ((j,k) = (2,1), (4,1), \ldots, (N_x - 2, N_y - 1)) \end{aligned} \tag{21}$$

のようになる。後半では周囲の値はすべて前半で更新されており、更新後の値を使うことができる。このスキームは先ほどのチェス盤の色分けに見立てて赤黒選択などと呼ばれる [1]。プログラムを組む際に j,k の繰り返し部分の実装が少し面倒になるが、手間に見合うだけ効率が良くなる。より次元が大きくなっても同様の選択が可能であり、計算の効率化が期待できる。(赤黒選択自体は 1 次元でも適用できるが、多次元問題で特に効率がよくなるのでここで示した。)

2 スペクトル法

偏微分方程式を解析的に解く場合、固有関数を使うと便利なことが多い。Fourier 変換や量子力学でよく用いられる各種直交多項式などは微分演算子に対する固有関数であり、うまく使うと偏微分微分方程式を非常に見通しの良い形に変形することができる。偏微分方程式を数値的に解く場合にも固有関数の方法は有用である。

先ほど考えた 2 次元 Laplace-Poisson 方程式を例として考える。$u(x,y)$ は Fourier 級数展開を使って

$$u(x,y) = \sum_{n,m=1}^{\infty} \tilde{u}_{n,m} \sin\left(\frac{n\pi x}{L_x}\right) \sin\left(\frac{m\pi y}{L_y}\right) \tag{22}$$

とできる (図 3 参照)。(境界条件より、cos を含む項は全てゼロとなる。) ただし、

$$\tilde{u}_{n,m} = \frac{4}{L_x L_y} \int_0^{L_x} dx \int_0^{L_y} dy\, u(x,y) \sin\left(\frac{n\pi x}{L_x}\right) \sin\left(\frac{m\pi y}{L_y}\right) \tag{23}$$

である。$u(x,y)$ に対する Laplace-Poisson 方程式は $\tilde{u}_{n,m}$ に対する方程式として書き換える

[1] チェス盤の色分けなら「白黒」とするのが自然そうだが、なぜか「白黒」ではなく「赤黒」と呼ばれることが多いようである。(チェス盤の色分けは赤黒のものがあるようである。)

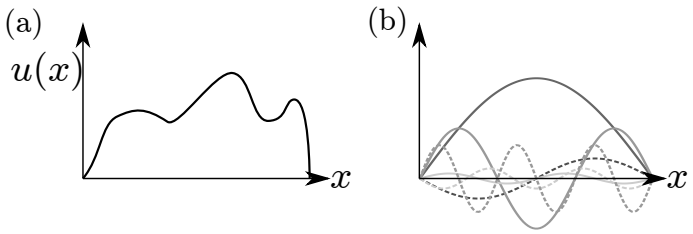

図 3: Fourier 級数を使った関数の離散化のイメージ。(a) 1 次元の関数 $u(x)$ が与えられたとき、何らかの形で離散化して有限のデータにしなければ計算機で取り扱えない。(a) 空間を離散化する代わりに、$u(x)$ を Fourier 級数に展開する (色の違う線がそれぞれ異なる周波数の成分)。有限個の Fourier 級数は計算機で取り扱うことができる離散的なデータである。

ことができる。すなわち、上記の Fourier 級数展開を Laplace-Poisson 方程式に代入すれば

$$\begin{aligned}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\sum_{n,m=1}^{\infty}\tilde{u}_{n,m}\sin\left(\frac{n\pi x}{L_x}\right)\sin\left(\frac{m\pi y}{L_y}\right)&=\phi(x,y)\\ -\sum_{n,m=1}^{\infty}\left[\left(\frac{n\pi}{L_x}\right)^2+\left(\frac{m\pi}{L_y}\right)^2\right]\tilde{u}_{n,m}\sin\left(\frac{n\pi x}{L_x}\right)\sin\left(\frac{m\pi y}{L_y}\right)&=\phi(x,y)\end{aligned} \tag{24}$$

となるから、右辺を Fourier 級数展開して左辺と右辺を比較すれば

$$-\left[\left(\frac{n\pi}{L_x}\right)^2+\left(\frac{m\pi}{L_y}\right)^2\right]\tilde{u}_{n,m}=\tilde{\phi}_{n,m} \tag{25}$$

が得られる。ただし、

$$\tilde{\phi}_{n,m}=\frac{4}{L_xL_y}\int_0^{L_x}dx\int_0^{L_y}dy\,\phi(x,y)\sin\left(\frac{n\pi x}{L_x}\right)\sin\left(\frac{m\pi y}{L_y}\right) \tag{26}$$

とした。$\tilde{u}_{n,m}$ に対する方程式はもとの Laplace-Poisson 方程式と等価だが、もはや偏微分方程式ではなく単なる代数方程式であり、解は容易に求められる。

$$\tilde{u}_{n,m}=-\frac{\tilde{\phi}_{n,m}}{(n\pi/L_x)^2+(m\pi/L_y)^2} \tag{27}$$

あとは $\tilde{u}_{n,m}$ から $u(x,y)$ を求めればよい。Fourier 級数展開では一般に無限個の項が表れるが、計算機で扱える量は有限なので適当な項までの和 (n,m についてそれぞれ N_x, N_y までとする) で打ちきってやれば

$$u(x,y)\approx\sum_{n=1}^{N_x}\sum_{m=1}^{N_y}\left[-\frac{\tilde{\phi}_{n,m}}{(n\pi/L_x)^2+(m\pi/L_y)^2}\right]\sin\left(\frac{n\pi x}{L_x}\right)\sin\left(\frac{m\pi y}{L_y}\right) \tag{28}$$

となる。

ここで偏微分方程式が代数方程式になったのは Fourier 級数展開を行った際に現れる三角関数の性質による。すなわち、

$$\frac{\partial}{\partial x}\sin(qx)=q\cos(qx),\qquad \frac{\partial^2}{\partial x^2}\sin(qx)=-q^2\sin(qx) \tag{29}$$

であるから (q は定数)、微分演算子 $\partial^2/\partial x^2$ が微分ではなく定数 $-q^2$ をかけることと一緒になっているのである。一般に、線形微分演算子[2] $\mathcal{L}$ に対して

$$\mathcal{L}\psi_k(\boldsymbol{r})=\lambda_k\psi_k(\boldsymbol{r})\qquad(k=0,1,2,\dots) \tag{30}$$

[2]微分演算子 $\mathcal{L}$ を関数 f,g の和に対して作用させる際、$\mathcal{L}(f+g)=\mathcal{L}f+\mathcal{L}g$ なる関係が成立するとき、$\mathcal{L}$ は線形微分演算子と呼ばれる。物理の問題に現れる多くの微分演算子は線形微分演算子である。

を満たすような関数 $\psi_k(\boldsymbol{r})$ を微分演算子 $\mathcal{L}$ の固有関数と呼び、λ_k を固有値と呼ぶ[3]。Laplace-Poisson 方程式のような偏微分方程式の境界値問題は

$$\mathcal{L}u(\boldsymbol{r}) = \phi(\boldsymbol{r}) \tag{31}$$

のような形で表現できる ($\mathcal{L} = \nabla^2$) ので、$u(\boldsymbol{r})$ そのものの代わりに固有関数を使った展開形

$$u(\boldsymbol{r}) = \sum_k \tilde{u}_k \psi_k(\boldsymbol{r}), \qquad \phi(\boldsymbol{r}) = \sum_k \tilde{\phi}_k \psi_k(\boldsymbol{r}) \tag{32}$$

を使うことで

$$\lambda_k \tilde{u}_k = \tilde{\phi}_k \tag{33}$$

ようにして微分演算子を固有値に置き換えて代数方程式にすることができるのである。Fourier 級数をはじめとする固有関数展開を使い、偏微分方程式を代数方程式に変換して解く方法はスペクトル法と呼ばれる。スペクトル法は直接法の一種であり、多次元の場合でも微分演算子に対する適切な固有関数が使えれば現実的な計算量で数値的に解を求められるスキームである。

スペクトル法は式を見ただけでは単純で必要な計算も少なそうに見えるかも知れないが、実際には固有関数展開の計算と展開係数から関数を求める計算が必要なので、場合によっては非常に計算が大変になる。Fourier 級数展開の場合、展開の各項に表れる係数 $\tilde{\phi}_{n,m}$ を求めるには x, y についての積分を求めなければならない。積分は計算機で直接扱えないので、離散化した場を足し合わせることで近似的に計算することになる。今の場合であれば例えば

$$\tilde{\phi}_{n,m} \approx \frac{4}{L_x L_y} \sum_{j=0}^{N_x} \sum_{k=0}^{N_y} \phi_{j,i} \sin\left(\frac{n\pi x \Delta_x}{L_x}\right) \sin\left(\frac{m\pi y \Delta_y}{L_y}\right) \Delta_x \Delta_y \tag{34}$$

などとすれば良いのだが、これでは 1 つの $\tilde{\phi}_{n,m}$ を求めるのに $(1+N_x)(1+N_y)$ 回の計算が必要となってしまう。精度を上げるには使う n, m の値を増やす必要があり、そのためには相当な回数の計算が必要である。結果として、このような方法で展開係数を求めるのであれば反復法のほうが良いという結論になってしまう。

実際には上記のような方法で展開係数を計算する必要はない。Fourier 級数を求めるためのスキームとして高速 Fourier 変換 (FFT) と呼ばれるスキームがあり、FFT を使えば効率的に展開係数を求めることができる[4]。また、展開係数から関数を求めるのは FFT を逆方向に適用すれば可能であるので (逆 FFT)、やはり効率的に計算を行うことができる。FFT 以外にも固有関数に対して類似の高速変換スキームが適用できることがあり、スペクトル法では基本的にそれらの高速変換スキームを使うのが基本である。ただし、FFT のような高速変換は通常、境界の形が単純な場合にしか適用できない。複雑な境界のもとでの境界値問題の場合、スペクトル法は有効ではなく、現実的な選択肢は反復法のみとなる。

[3] k の値が離散ではなく連続変数を取る場合もある。ここでは k が離散的な値を取ると仮定し、その値を $0, 1, 2, \ldots$ としておく。(連続的な値の場合でも形式的な扱いはあまり変わらない。)

[4] 1 次元の場合、離散化して N 点にした場に対して定義に従って直接的に Fourier 級数を求めると $O(N^2)$ の計算が必要である。FFT では必要な計算は $O(N \log N)$ であり、N が大きい場合には直接的な計算と比べて著しく計算が少なくてすむ。FFT は再帰計算と呼ばれる処理を使う少し高度なアルゴリズムであり、実装は少し複雑になるのだが、非常に有用なアルゴリズムである。