計算アルゴリズム論レポート課題

161003 今村秀明

June 29, 2017

1 Introduction

本レポートでは、示された課題のうち課題番号 1 番に回答する。

この問題は、2 次元正方形領域上のポアソン方程式を差分法で定式化し、作られた連立方程式を CG 法で解くというものである。問題にする 2 次元正方形領域としては $[0,1] \times [0,1]$ を用い、解くべきポアソン方程式は以下を用いる。

$$u_{xx} + u_{yy} = -8\pi^2 \sin 2\pi x \cos 2\pi y$$

境界条件は、全 Dirichlet 条件として以下を用いる。

$$u(x,0) = u(x,1) = \sin 2\pi x$$
$$u(0,y) = u(1,y) = 0$$

この方程式の厳密解は、$u(x,y) = \sin 2\pi x \cos 2\pi y$ である。

次章以降、この方程式を如何にして離散化し、連立方程式を解いたか説明し、その誤差や収束の速さについての分析を行う。

2 Method

まず、解くべきポアソン方程式を差分法で定式化し離散化方程式を得た。その手順について説明する。

まず、2 次元正方形領域を格子状に区切る。格子は $n \times n$ に区切られたとする。すると、格子上の点は $(x_i, y_j) = (\frac{i}{n}, \frac{j}{n})$ という形をしている。($0 \leq i, j \leq n$ である。)

次に、格子上の点に未知数 $u_{i,j}$ を置く。これが格子上の点における真の解 $u(x_i, y_j)$ の近似値である。今、境界上の点は全て Dirichlet 条件を仮定しているので、未知数 $u_{i,j}$ は境界上の点には定義されていないことに注意する。

その後、微分の値を差分で近似し、2 次精度の中心差分で下式を得た。

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{\Delta x^2} = \frac{\partial^2 u}{\partial x^2} + O(\Delta x^4)$$

ただし、$\Delta x = \Delta y = \frac{1}{n}$ である。左辺の格子状上の点における真の解 $u(x_i, y_j)$ を全て未知数 $u_{i.j}$ で置き換えることにより、2 次微分 $\frac{\partial^2 u}{\partial x^2}$ の近似値を得る。y の 2 次微分についても同様に近似する。

そして、解くべき方程式の微分を差分で置き換えると以下の離散化方程式を得る。

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = f_{i,j}$$

$$\Leftrightarrow -4u_{i,j} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} = \frac{f_{i,j}}{n^2}$$

ただし、解くべき方程式の右辺の 2 変数関数を $f(x, y)$ とし、その格子上の点 (x_i, y_j) における値を $f_{i.j}$ としている。また、$\Delta x = \Delta y = \frac{1}{n}$ を用いた。この方程式において、添字 i, j は $1 \leq i, j \leq n-1$ の範囲にある。$i = 0, n, j = 0, n$ の時、定義されていない未知数が登場してしまうが、その時は境界条件を用いて値を設定することにする。

この離散化方程式を行列とベクトルを用いて表示すると、係数行列は $(n-1)^2 \times (n-1)^2$ の対称広義優対角行列になる。得られた連立方程式は共役勾配法（CG 法）で解く。以下が CG 法のアルゴリズムである。

Algorithm 1 Calculate x such that $Ax = b$

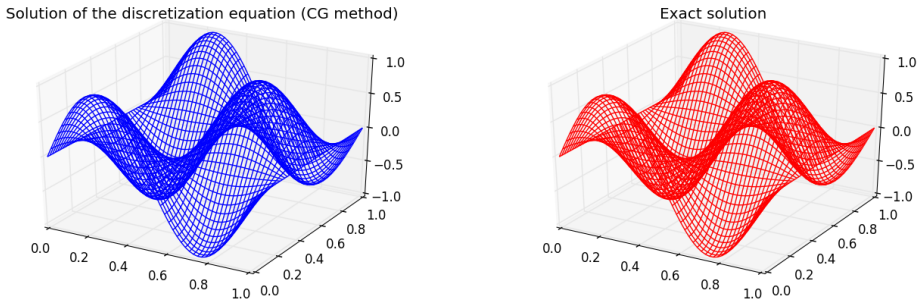
Input: $A : (n-1)^2 \times (n-1)^2\ matrix, b : (n-1)^2\ dimensional\ vector, \epsilon$
Output: $x : (n-1)^2\ dimensional\ vector\ such\ that\ Ax \fallingdotseq b$

$x_0 = n\ dimensional\ all\ 0\ vector$
$i = 0$
$p_0 = r_0 = b - Ax_0$
while $\frac{||Ax_i - b||}{||b||} > \epsilon$ **do**
  $\alpha_i = \frac{r_i^T r_i}{p_i^T A p_i}$
  $r_{i+1} = r_i - \alpha_i A p_i$
  $x_{i+1} = x_i + \alpha p_i$
  $\beta_i = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$
  $p_{i+1} = r_{i+1} + \beta_i p_i$
  $i = i + 1$
end while
return x_i

以上の内容を python で実装した。以下がソースコードや出力結果を含む github ページへのリンクである。

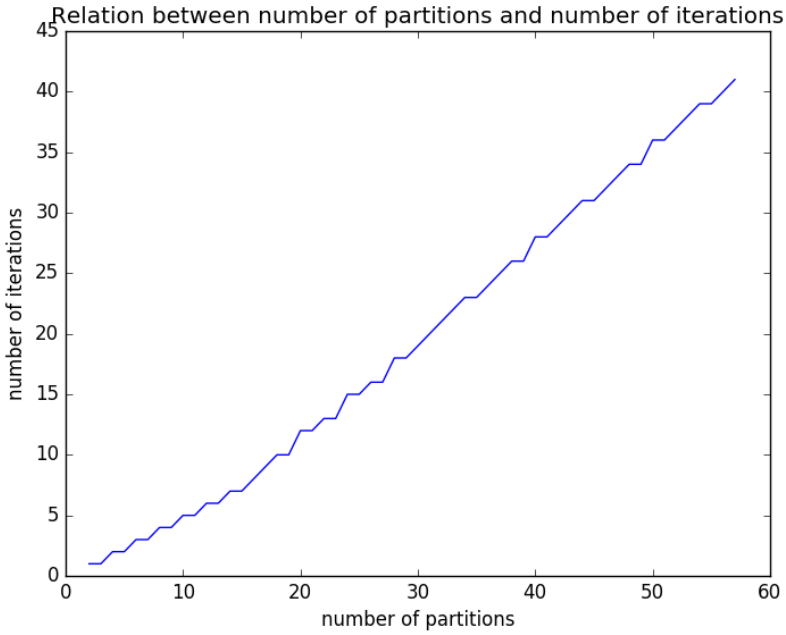
3 Result

まず、離散化方程式を解いて得られた近似解と、真の解のプロットを以下に示す。ともに、刻み幅は $\frac{1}{50}$ である。図のように、近似解と真の解は肉眼ではほとんど一致している。

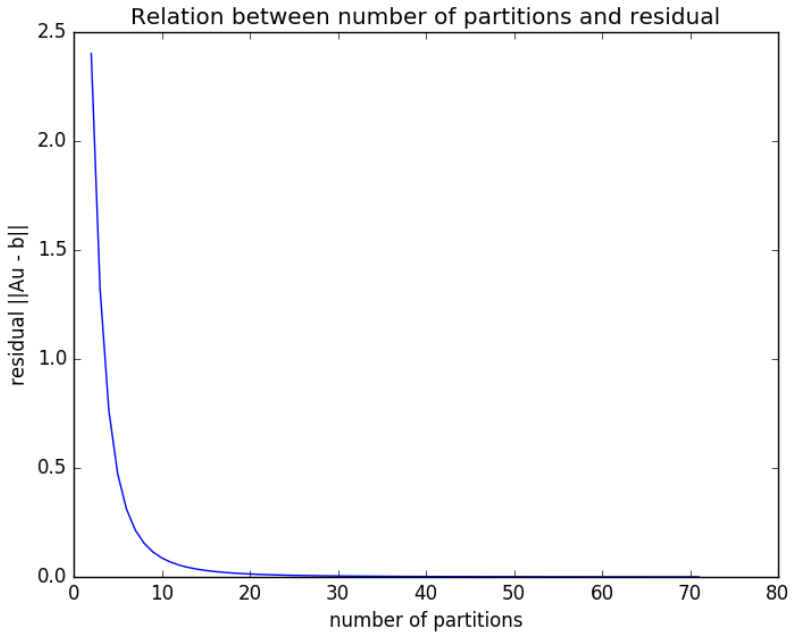


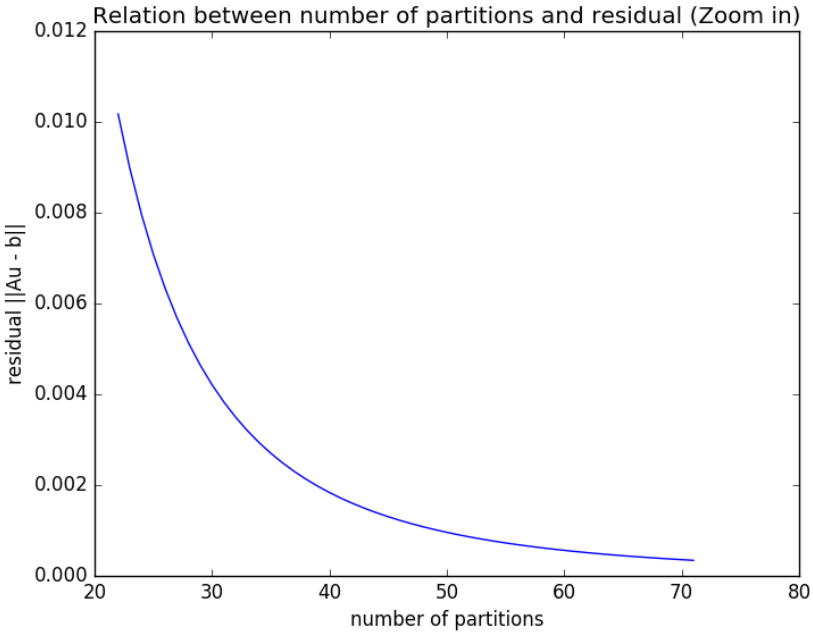
それでは以下、順に課題で与えられた観点から手法の評価を行う。

まず、差分法の刻み幅と CG 法の反復回数との関係を示す。以下が出力画像である。刻み幅は $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, ..., \frac{1}{n}, ...$ と変化させた。横軸は刻み幅の逆数である。縦軸が反復回数である。この図から、CG 法の反復回数は刻み幅の逆数に対して線形に増加していくことがわかる。



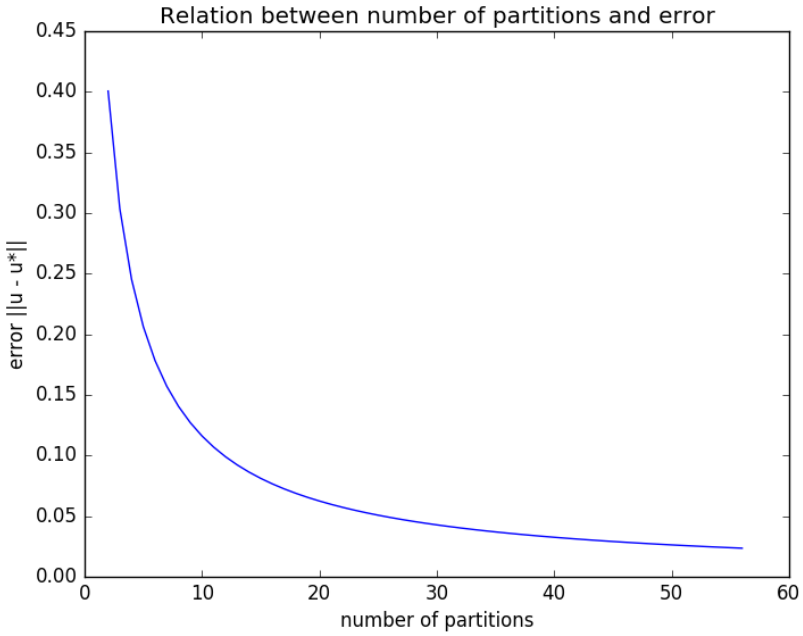
次に、離散化の精度を上げた時に離散化方程式に厳密解を入れた時の残差が減少していくことを示すプロットを以下に示す。ここで、離散化の精度を上げるとは、離散化の際に 2 次元正方形領域を裁断するが、その刻み幅を細くすることである。横軸は刻み幅の逆数、縦軸は残差である。この図から、残差は刻み幅の逆数に対して二次関数的に減少していく様子が見て取れる。



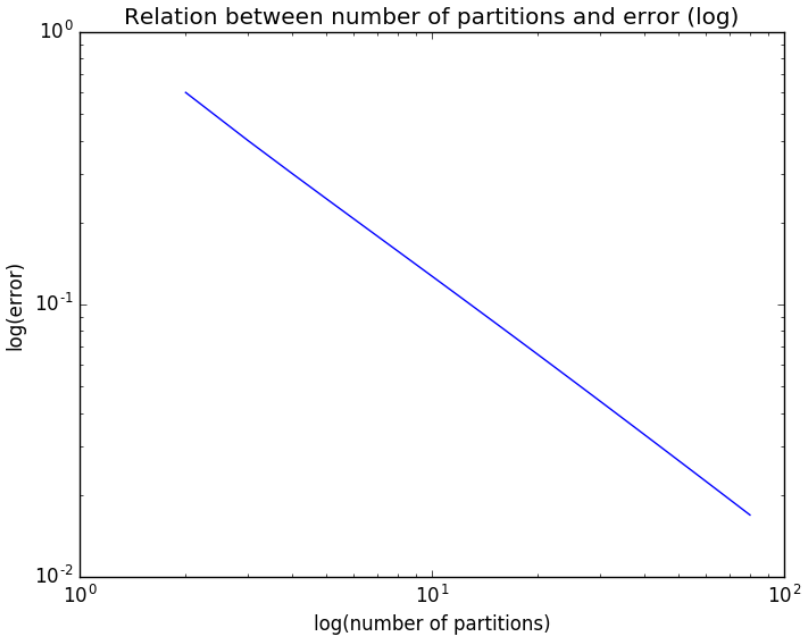


次に、離散化の精度を上げた時に離散化方程式を解いた近似解が厳密解に近づくことを

示す。近似解が厳密解に近づくことを示すために、これらの差のノルムを縦軸に取った。横軸は刻み幅の逆数である。この図から、差のノルムは刻み幅の逆数に対して二次関数的に減少していく様子が見て取れる。



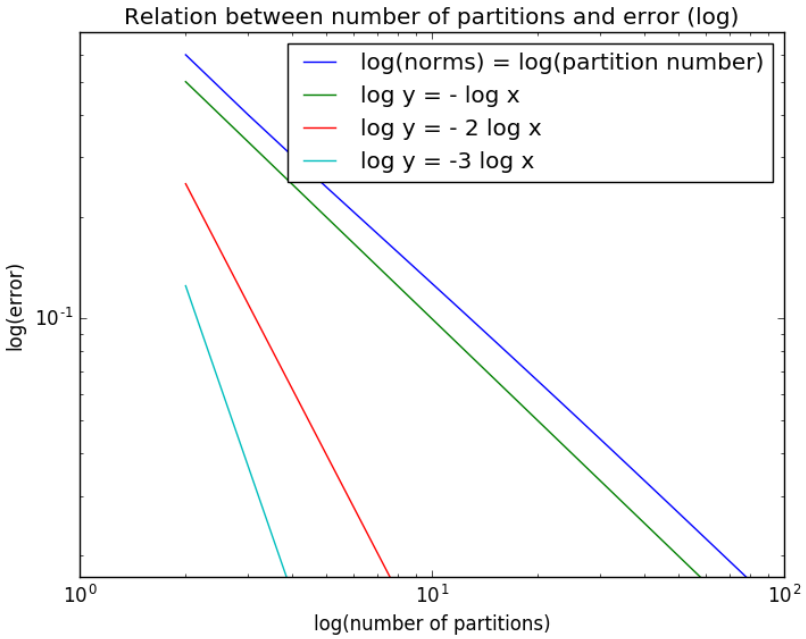
最後に、収束の速さを実験的に示すためのプロットを示す。以下は、両対数グラフで横軸が刻み幅の逆数、縦軸が誤差ノルムである。図は直線となっている。このプロットから。収束の速さを求める考察は次章で行う。



4 Discussion

2 つの観点から考察を行う。第一の観点は、収束の速さについてである。これは前章の最後に示したグラフに基づく議論である。第二の観点は、作成したプログラムの高速化手法についてである。これは今回考えている離散化連立方程式の係数行列が、刻み幅を小さくしていくと大規模疎行列になることに基づく議論である。

まず、第一の観点について述べよう。前章の最後に示した両対数グラフについて、三つの関数 $\log y = -\log x$、$\log y = -2\log x$、$\log y = -3\log x$ を同時にプロットしたものを下図に示す。



この図から、刻み幅の逆数の対数と誤差のノルムの対数の間には比例係数 -1 で比例の関係にあることがわかる。すると、誤差のノルムが刻み幅の逆数に反比例する。これは、以下の簡単な計算により確かめることができる。

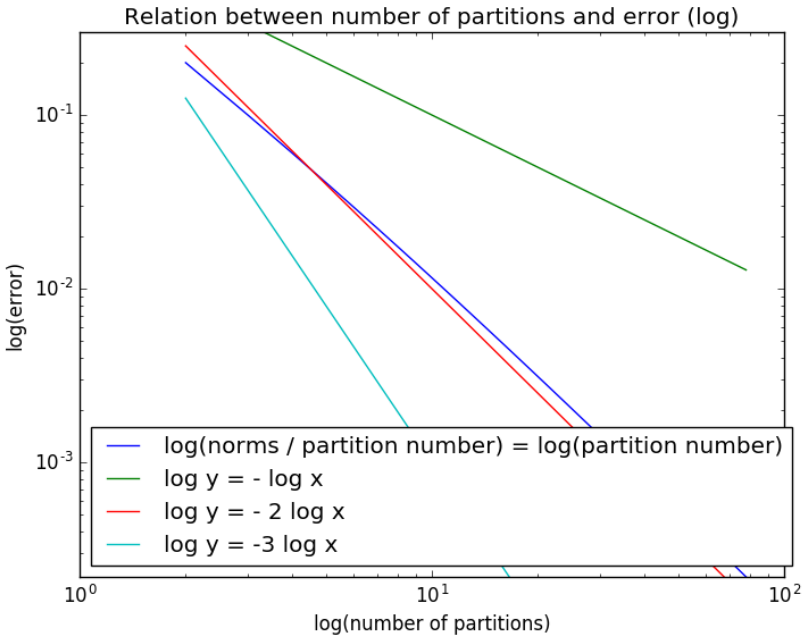
$$\begin{aligned} & \log y = a \log x + const \\ \Leftrightarrow \quad & \log y = \log(const \times x^a) \\ \Leftrightarrow \quad & y = const \times x^a \end{aligned}$$

すなわち、$-a$ が収束次数に他ならない。これは、授業で扱った真の収束次数 2 と一致しない。

問題はどこにあるだろうか。考えうるのは、刻み幅を大きくしていった時、離散化方程式の係数行列や解ベクトルの次元は 2 乗に比例して大きくなっていく。そのため、異なる次元数のベクトルのノルムをそのまま比較している。これは、誤差の評価という視点からは適当でないと思われる。そこで、誤差ベクトルのノルムを直接比較するのではなく、誤差ノルムをベクトルの次元数のルートで割った以下の量を比較することにした。

$$\frac{||u - u^*||}{\sqrt{dim\, u}}$$

このような量を用いれば、次元の増大に対するノルムの増分を打ち消すことができる。（ノルムに対するこの修正はベクトルの”正規化”に似ている。）そうして、再度プロットを行ったところ下図を得た。



この図を見ればわかるように、今度は刻み幅の逆数の対数と誤差のノルムの対数の間には比例係数 -2 で比例の関係にあることになり、真の収束次数 2 と見事に対応する。

次に、第二の観点について述べる。考えている離散化連立方程式の係数行列は大規模疎行列である。したがって、それに準じた手法で CG 法を設計すれば高速なプログラムが得られると期待できる。しかし、今回作成したプログラムは係数行列にそのような仮定を置かず、一般的な行列・ベクトル積を利用している。この点について考慮を行い、高速化する手法について検討する。

まず、大規模な疎行列と言ってもその形によって効率は変わりうる。今回考えている行列は、五重対角行列の亜種とも言える形をしている。各行には高々 5 つの要素しかない。そのため、$n \times n$ 行列・n 次元ベクトル積は次ページに示すようなアルゴリズムで凡そ $5n$ 回の乗算で計算できる。

このアルゴリズムにおける non-zero indices は高々 5 つしかないため、行列・ベクトル積は高速に実行できる。

5 Conclusion

今回のレポートでは、2 次元正方形領域上のポアソン方程式を差分法で定式化し、得られた離散化連立方程式を CG 法で解いた。結果をグラフの形で出力し、幾つかの関数とに比較により収束の速さを実験的に求めた。その後、大規模疎行列の行列・ベクトル積について高速化の手法について検討した。

Algorithm 2 Calculate Ax

Input: $A : n \times n\ matrix, x : n\ dimensional\ vector$
Output: $Ax : n\ dimensional\ vector$

```
y = n dimensional all 0 vector
for i = 1 to n do
    non-zero indices = indices of non-zero component of row i vector of A
    for j in non-zero indices do
        y_i += A[i][j] * x[j]
    end for
end for
return y
```