# FFTを用いたポアソン方程式の高速解法

M2010MM026 蓑島　伸明
指導教員　杉浦　洋

## 1　はじめに

FFT を用いた Poisson 方程式の高速解法と，そのGPU(Graphics Processing Unit) への効果的な実装法について研究する．GPU は画像処理に特化して開発されたプロセッサである．そのため，GPU は高い並列演算処理能力を持ち，大量のデータを扱う場合，CPU を用いて行うより効率的に処理することができる．

本研究ではポアソン方程式を差分法で離散化した線形方程式の各次元を離散型フーリエ変換し，係数行列を対角行列，又は三重対角行列に変換して解く．

処理系として NVIDIA 社の GPU を用いる．また，NVIDIA 社が汎用コンピューティングを行うために開発した CUDA を用いてプログラムを作成する．本研究では，2 種類のポアソン解法について高速化を目指す．主に実行時間及び精度について調べ，各種アルゴリズムの比較を行う．物理的次元は 2 次元について考える．

## 2　行列方程式

領域 $D = [0,1]\times[0,1]$ の 2 次元ポアソン方程式のディリクレ型境界値問題

$$\begin{cases} u_{xx}(x,y)+u_{yy}(x,y)=f(x,y),(x,y)\in D, \\ u(x,y)=g(x,y),(x,y)\in\partial D \end{cases} \tag{1}$$

の数値解法を考える．
領域 $D$ を $x$ 方向に $m+1$ 等分して $y$ 方向に $n+1$ 等分する．ただし，$m+1, n+1$ は 2 の巾乗とする．格子点 $x_i, y_j$ は

$$x_i = i\Delta x \ \ (0\le i\le m+1),\ \ \Delta x=\frac{1}{m+1}, \tag{2}$$

$$y_j = j\Delta y \ \ (0\le j\le n+1),\ \ \Delta y=\frac{1}{n+1} \tag{3}$$

となる．ここで格子点 $(x_i, y_j)$ 上の $u$ の近似値を $u_{ij}\cong u(x_i,y_j)$ とする．このうち，内点上での値 $u_{ij}(1\le i\le m, 1\le j\le n)$ が未知で未知数の数は $mn$ である．境界点上での値

$$u_{0,j}=g(0,y_j), u_{m+1,j}=g(1,y_j) \tag{4}$$

$$u_{i,0}=g(x_i,0), u_{i,n+1}=g(x_i,1) \tag{5}$$

は,
境界条件から得る．偏微分係数の中心差分近似は

$$u_{xx}(x_i,y_j)\cong u_{ij}^{(x)}=\frac{u_{i-1,j}-2u_{ij}+u_{i+1,j}}{\Delta x^2}, \tag{6}$$

$$u_{yy}(x_i,y_j)\cong u_{ij}^{(y)}=\frac{u_{i,j-1}-2u_{ij}+u_{i,j+1}}{\Delta y^2} \tag{7}$$

$$(1\le i\le m, 1\le j\le n)$$

となる．これより，ポアソン方程式の中心 5 点差分近似

$$u_{ij}^{(x)}+u_{ij}^{(y)}=f_{ij}=f(x_i,y_i) \tag{8}$$
$$(1\le i\le m, 1\le j\le n)$$

が得られる．内点における各量を要素にした行列とその列ベクトル，行ベクトル表示を

$$\boldsymbol{u}_j=\begin{pmatrix} u_{1j}\\ u_{2j}\\ \vdots\\ u_{mj}\end{pmatrix}(1\le j\le n), \tag{9}$$

$$\boldsymbol{v}_j=\begin{pmatrix} u_{i1} & u_{i2} & \dots & u_{in}\end{pmatrix}(1\le i\le m), \tag{10}$$

$$U=\begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n}\\ u_{21} & u_{22} & \dots & u_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ u_{m1} & u_{m2} & \dots & u_{mn}\end{pmatrix}$$

$$=\begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n\end{pmatrix}=\begin{pmatrix} \boldsymbol{v}_1\\ \boldsymbol{v}_2\\ \vdots\\ \boldsymbol{v}_m\end{pmatrix}, \tag{11}$$

$$U^{(x)}=\begin{pmatrix} u_{11}^{(x)} & u_{12}^{(x)} & \dots & u_{1n}^{(x)}\\ u_{21}^{(x)} & u_{22}^{(x)} & \dots & u_{2n}^{(x)}\\ \vdots & \vdots & \ddots & \vdots\\ u_{m1}^{(x)} & u_{m2}^{(x)} & \dots & u_{mn}^{(x)}\end{pmatrix}, \tag{12}$$

$$U^{(y)}=\begin{pmatrix} u_{11}^{(y)} & u_{12}^{(y)} & \dots & u_{1n}^{(y)}\\ u_{21}^{(y)} & u_{22}^{(y)} & \dots & u_{2n}^{(y)}\\ \vdots & \vdots & \ddots & \vdots\\ u_{m1}^{(y)} & u_{m2}^{(y)} & \dots & u_{mn}^{(y)}\end{pmatrix}, \tag{13}$$

$$F=\begin{pmatrix} f_{11} & f_{12} & \dots & f_{1n}\\ f_{21} & f_{22} & \dots & f_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ f_{m1} & f_{m2} & \dots & f_{mn}\end{pmatrix}, \tag{14}$$

とする．式 (8) より

$$U^{(x)}+U^{(y)}=F \tag{15}$$

である．式 (6) をベクトル表示すると

$$\boldsymbol{u}_j^{(x)}=\begin{pmatrix} u_{1j}^{(x)}\\ \vdots\\ u_{mj}^{(x)}\end{pmatrix}=\alpha T_m\begin{pmatrix} u_{1j}\\ \vdots\\ u_{mj}\end{pmatrix}+\alpha\begin{pmatrix} u_{0j}\\ \mathbf{0}\\ u_{m+1,j}\end{pmatrix}$$

$$=\alpha T_m\boldsymbol{u}_j+\alpha\boldsymbol{c}_j^{(x)}\ \ (1\le j\le n),\alpha=\frac{1}{\Delta x^2} \tag{16}$$

ここで $T_m$ は対角要素が $-2$ で副対角要素が 1 の三重対角行列

$$T_m = \begin{pmatrix} -2 & 1 & & \mathbf{0} \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ \mathbf{0} & & 1 & -2 \end{pmatrix} \in R^{m\times m} \tag{17}$$

である．式 (16) を行列表示すると

$$U^{(x)} = \alpha T_m U + \alpha C^{(x)} \tag{18}$$

となる．ここで，$C^{(x)}$ は

$$C^{(x)} = \begin{pmatrix} u_{01} & u_{02} & \dots & u_{0n} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ u_{m+11} & u_{m+12} & \dots & u_{m+1n} \end{pmatrix} \tag{19}$$

である．同様に式 (7) をベクトル表示すると，

$$(\boldsymbol{v}_i^{(y)})^T = \begin{pmatrix} u_{i1}^{(y)} \\ \vdots \\ u_{in}^{(y)} \end{pmatrix} = \beta T_n \begin{pmatrix} u_{i1} \\ \vdots \\ u_{in} \end{pmatrix} + \beta \begin{pmatrix} u_{i0} \\ \mathbf{0} \\ u_{i,n+1} \end{pmatrix}$$

$$= \beta T_n \boldsymbol{v}_i^T + \beta(\boldsymbol{c}_i^{(y)})^T \quad (1 \le i \le m), \beta = \frac{1}{\Delta y^2}. \tag{20}$$

よって，$\boldsymbol{v}_i^{(y)}$ は

$$\boldsymbol{v}_i^{(y)} = \beta \boldsymbol{v}_i T_n + \beta \boldsymbol{c}_i^{(y)} (1 \le i \le m) \tag{21}$$

となり，行列表示すると

$$U^{(y)} = \beta U T_n + \beta C^{(y)}. \tag{22}$$

ただし，$C^{(y)}$ は

$$C^{(y)} = \begin{pmatrix} u_{10} & 0 & \dots & 0 & u_{1n+1} \\ u_{20} & 0 & \dots & 0 & u_{2n+1} \\ \vdots & \vdots & \ddots & \vdots & \\ u_{m0} & 0 & \dots & 0 & u_{mn+1} \end{pmatrix} \tag{23}$$

である．$U^{(x)}, U^{(y)}$ を式 (15) に代入して，未知行列 $U$ に関する行列方程式

$$\alpha T_m U + \beta U T_n = F - \alpha C^{(x)} - \beta C^{(y)} \tag{24}$$

を得る．ここで $B = F - \alpha C^{(x)} - \beta C^{(y)}$ とおくと

$$\alpha T_m U + \beta U T_n = B \tag{25}$$

が得られる [3].

## 3　行列 $T_n$ の固有値解析

加法定理より

$$\sin\frac{(k-l)l\pi}{n+1} + \sin\frac{(k+1)l\pi}{n+1} = 2\cos\frac{l\pi}{n+1}\sin\frac{kl\pi}{n+1}$$

$$(1 \le k \le n, 1 \le l \le n). \tag{26}$$

ここで $s_{kl} = \sin\dfrac{kl\pi}{n+1}, c_l = 2\cos\dfrac{l\pi}{n+1}$ とおくと

$$s_{k-1,l} + s_{k+1,l} = c_l s_{kl} \quad (1 \le k \le n) \tag{27}$$

となる．$k = 0, n$ のとき，$s_{01} = s_{n+1,l} = 0$ ゆえ，

$$\begin{cases} s_{2l} = c_l s_{1l} & (k = 1), \\ s_{k-1,l} + s_{k+1,l} = c_l s_{kl} & (2 \le k \le n-1), \\ s_{n-1,l} = c_l s_{n,l} & (k = n) \end{cases}$$

となり，これを行列表示すると

$$B_n \boldsymbol{s}_l = \begin{pmatrix} 0 & 1 & & 0 \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & 0 \end{pmatrix} \begin{pmatrix} s_{1l} \\ s_{2l} \\ \vdots \\ s_{nl} \end{pmatrix}$$

$$= c_l \begin{pmatrix} s_{1l} \\ s_{2l} \\ \vdots \\ s_{nl} \end{pmatrix} = c_l \boldsymbol{s}_l \quad (1 \le k \le n). \tag{28}$$

よって，$c_l, \boldsymbol{s}_l (1 \le l \le n)$ は $B_n$ の固有値と固有ベクトルであることが分かる．ここで，
$S_n = (\boldsymbol{s}_1, \boldsymbol{s}_2, \dots, \boldsymbol{s}_n), C_n = \mathrm{diag}(c_1, c_2, \dots, c_n)$ とおくと，$B_n$ は

$$B_n = S_n C_n S_n^{-1} \tag{29}$$

と分解される．$T_n = -2I_n + B_n$ ゆえ，固有値は $d_l^{(n)} = -2 + c_l (1 \le l \le n)$, 対応する固有ベクトルは $\boldsymbol{s}_l$ である．ゆえに，

$$T_n = S_n D_n S_n^{-1} \tag{30}$$

となる．ここで，$D_n = \mathrm{diag}(d_1^{(n)}, d_2^{(n)}, \dots, d_n^{(n)})$ である．
また，$d_l^{(n)}$ は

$$d_l^{(n)} = -2(1 - \cos\frac{l\pi}{n+1})$$
$$= -4\sin^2\frac{l\pi}{2(n+1)} \quad (1 \le l \le n) \tag{31}$$

ここで変換 $\boldsymbol{y} = S_n \boldsymbol{x}$ とすると，

$$y_l = \sum_{k=1}^{n} x_k \sin\frac{\pi kl}{n+1} \quad (1 \le l \le n) \tag{32}$$

となる．これは離散型 sine 逆変換である．また，変換 $\boldsymbol{x} = S_n^{-1}\boldsymbol{y}$ は

$$x_l = \frac{2}{n+1}\sum_{k=1}^{n} y_k \sin\frac{\pi kl}{n+1} \qquad (1 \le l \le n) \quad (33)$$

であり，これは離散型 sine 変換である．これらは，$n+1$ が 2 の巾のとき，Cooley-Tukey の FFT により，$n\log_2 n + \mathrm{O}(n)$ の乗算数で計算できる．式 (33) より，

$$S_n^{-1} = \frac{2}{n+1}S_n \quad (34)$$

ここで，$S$ をスケーリングしたものを

$$\tilde{S}_n = \sqrt{\frac{2}{n+1}}S_n \quad (35)$$

とすると，

$$\begin{aligned}\tilde{S}_n^{-1} &= \sqrt{\frac{n+1}{2}}S_n^{-1} = \sqrt{\frac{n+1}{2}}\cdot\frac{2}{n+1}S_n \\ &= \sqrt{\frac{2}{n+1}}S_n = \tilde{S}_n. \end{aligned} \quad (36)$$

よって式 (30) より，

$$\begin{aligned}T_n &= \sqrt{\frac{n+1}{2}}\cdot\sqrt{\frac{2}{n+1}}S_n D_n S_n^{-1} \\ &= \tilde{S}_n D_n \tilde{S}_n^{-1} = \tilde{S}_n D_n \tilde{S}_n \end{aligned} \quad (37)$$

また，式 (36) より，

$$\tilde{S}_n^2 = \tilde{S}_n\tilde{S}_n^{-1} = I_n \quad (38)$$

である．

## 4 FFT アルゴリズム

3 節の結果を用いて，行列方程式 (25) を変形し，係数行列が対角行列又は三重対角行列の線形方程式に帰着させる．

### 4.1 Semi FFT アルゴリズム

式 (25) の $\alpha T_m U + \beta U T_n = B$ に
式 (39) の $T_n = \tilde{S}_n D_n \tilde{S}_n$ を代入すると

$$\alpha T_m U + \beta U \tilde{S}_n D_n \tilde{S}_n = B. \quad (39)$$

右から $\tilde{S}_n$ を掛けると，式 (39) より

$$\alpha T_m U\tilde{S}_n + \beta U\tilde{S}_n D_n = B\tilde{S}_n \quad (40)$$

ここで，

$$V = U\tilde{S}_n, A = B\tilde{S}_n \quad (41)$$

とすると

$$\alpha T_m V + \beta V D_n = A \quad (42)$$

これを，第 $i$ 列ベクトル $\boldsymbol{v}_j, \boldsymbol{a}_j$ について書くと

$$\alpha T_m \boldsymbol{v}_j + \beta d_j^{(n)}\boldsymbol{v}_j = \boldsymbol{a}_j, \quad (43)$$

すなわち

$$(\alpha T_m + \beta d_j^{(n)})\boldsymbol{v}_j = \boldsymbol{a}_j \qquad (1 \le j \le n). \quad (44)$$

係数行列 $\alpha T_m + \beta d_j^{(n)}$ は 3 重対角行列になる．これをガウス消去法で解く．計算量は各 $j$ について $\mathrm{O}(m)$，全体で $\mathrm{O}(mn)$ である．
最後に解 $U$ を

$$U = V\tilde{S}_n \quad (45)$$

で計算する．

- Semi FFT アルゴリズムの計算量

1. 式 (41) の A の計算 $B\tilde{S}_n$
   $mn\log_2 n$ flops
2. 式 (44) の V の計算
   $5mn$ flops
3. 式 (45) で U の計算 $V\tilde{S}_n$
   $mn\log_2 n$ flops

であり，合計を $C_S$ とすると，

$$C_S = 2mn\log_2 n + 5mn \ \ \text{flops} \quad (46)$$

である．

### 4.2 Full FFT アルゴリズム

式 (25) の $\alpha T_m U + \beta U T_n = B$ に，
$T_m = \tilde{S}_m D_m \tilde{S}_m, T_n = \tilde{S}_n D_n \tilde{S}_n$ を代入すると，

$$\alpha\tilde{S}_m D_m \tilde{S}_m U + \beta U\tilde{S}_n D_n \tilde{S}_n = B \quad (47)$$

両辺に左から $\tilde{S}_m$，右から $\tilde{S}_n$ をかけると，

$$\alpha D_m \tilde{S}_m U \tilde{S}_n + \beta\tilde{S}_m U\tilde{S}_n D_n = \tilde{S}_m B\tilde{S}_n \quad (48)$$

となる．ここで，

$$V = \tilde{S}_m U\tilde{S}_n = (v_{ij}), A = \tilde{S}_m B\tilde{S}_n = (a_{ij}) \quad (49)$$

とおくと，

$$\alpha D_m V + \beta V D_n = A \quad (50)$$

これを要素ごとにかくと

$$\alpha d_i^{(m)} v_{ij} + \beta v_{ij} d_j^{(n)} = a_{ij} \quad (51)$$

すなわち，
$V = (v_{ij})$ として，式 (51) で求めた $a_{ij}$ を用いて，$v_{ij}$ を求める．

$$v_{ij} = \frac{a_{ij}}{\alpha d_i^{(m)} + \beta d_j^{(n)}} \quad (52)$$

最後に解 $U$ を

$$U = \tilde{S}_m V\tilde{S}_n \quad (53)$$

で計算する．計算量は $2mn\log_2 mn + \mathrm{O}(mn)$ flops になる．

- Full FFT アルゴリズムの計算量

1. 式 (49) の A の計算 $\tilde{S}_n B \tilde{S}_n$
   $mn \log_2 n + mn \log_2 m = mn \log_2 mn$ flops
2. 式 (52) の V の計算
   $mn$ flops
3. 式 (53) で U の計算 $\tilde{S}_n V \tilde{S}_n$
   $mn \log_2 mn$ flops

であり，合計を $C_F$ とすると，

$$C_F = 2mn \log_2 mn + mn \ \text{flops} \tag{54}$$

である．以上の 2 つのアルゴリズムの計算量を比較すると，

$$C_F - C_S = 2mn \log_2 m - 4mn \geq 0 (\log_2 m \geq 2) \tag{55}$$

となる．よって式 (54) をみたすとき，計算量では Full FFT のほうが多くなる．これを GPU 上で実装する場合，計算量及び計算時間はどちらが優れているかを考えていく．

## 5 CUDA

CUDA(Compute Unified Architecture) は NVIDIA 社が GPU を汎用コンピューティングに活用するために C 言語をベースとして開発したプログラミング環境である．CUDA プログラミングには多くのメモリがあり，実際の計算ではレジスタや共有メモリ，定数メモリなどの高速なメモリを用いて計算を行う．レジスタはスレッドごとに確保するメモリであり，SM 内のレジスタに確保される．共有メモリはブロック内ののスレッドで共有されるメモリである．定数メモリは専用キャッシュを持ち，全スレッドから読み取り可能である．しかし，書き込みは CPU 側からのみ可能である [1].

## 6 実験

本研究の実験で用いる 2 次元ポアソン方程式を，

$$\begin{cases} u_{xx}(x,y) + u_{yy}(x,y) = (x^2+y^2)e^{xy}, (x,y) \in D, \\ u(x,y) = e^{xy}, (x,y) \in \partial D \end{cases} \tag{56}$$

とする．解は $u(x,y) = e^{xy}$ となる．

GPU 上での Full FFT と Semi FFT の単精度における計算時間，精度の比較を行う．実験では，複素フーリエ変換を用いた．

### 6.1 2 つのアルゴリズムの比較

計算時間及び精度を示す．参考として，マルチグリッド法による池田 [2] の結果を併記する． Full のほうが Semi より速いという結果となった．Full は FFT を 4 回，Semi では 2 回用いている．Semi の三重対角のガウス消去は完全に並列化することが難しく多くの時間がかかる．その点，Full のほうは単純な要素ごとの割り算だけなので，並列処理しやすい．Semi のガウス消去は，並列処理にはあまり向いてなく，予想以上に計算時間がかかり，Full の計算時間に負けてしまったと考えられる．

表 1 Full FFT アルゴリズムの計算時間と誤差

| | Full FFT | |
|---|---|---|
| 格子サイズ | 計算時間 | 最大絶対誤差 |
| $64 \times 64$ | 0.000275 | $3.3 \times 10^{-6}$ |
| $128 \times 128$ | 0.000884 | $3.33 \times 10^{-6}$ |
| $256 \times 256$ | 0.005154 | $1.16 \times 10^{-5}$ |

表 2 Semi FFT アルゴリズムの計算時間と誤差

| | Semi FFT | |
|---|---|---|
| 格子サイズ | 計算時間 | 最大絶対誤差 |
| $64 \times 64$ | 0.000588 | $4.7 \times 10^{-5}$ |
| $128 \times 128$ | 0.001294 | $1.99 \times 10^{-4}$ |
| $256 \times 256$ | 0.005707 | $6.64 \times 10^{-4}$ |

FFT，及び Semi におけるガウス消去の部分を高速化することにより高速に処理できる．次にマルチグリッド法との比較を行う．実験結果は以下である．計算時間に関しては，Full,Semi ともにマルチグリッド法より高速処理することができた．今回，用いた解法は並列処理に向いていると言える．

表 3 マルチグリッド法の計算時間と誤差

| | マルチグリッド法 | |
|---|---|---|
| 格子サイズ | 計算時間 | 精度 |
| $64 \times 64$ | 0.0035012 | $3.70 \times 10^{-6}$ |
| $128 \times 128$ | 0.0070329 | $4.89 \times 10^{-6}$ |
| $256 \times 256$ | 0.0069289 | $1.03 \times 10^{-5}$ |

## 7 おわりに

本研究では，FFT を用いたポアソン方程式を実装し計算を高速化することを目指した．実験結果として，マルチグリッドによる実装より良い結果を得ることができた．ただ，本研究に用いた 2 つのアルゴリズムはまだ高速化する余地がある．アルゴリズム内の FFT，及びガウス消去を高速化することで計算時間を減らすことができると考えられる．

## 参考文献

[1] NVIDIA：CUDAZone, `http://www.nvidia.co.jp/object/cuda_home_new_jp.html`.

[2] 池田達哉：GPU を用いたマルチグリッド法の実装, 南山大学数理情報研究科数理情報専攻修士論文 (2011).

[3] Charles Van Loan,Computational Frameworks for Fast Fourier Transform Society for Industrial and Applied Mathematics,1992.