θ 法のプログラムを作って実験

桂田 祐史

2023 年 6 月 27 日, 2024 年 6 月 19 日

対応する講義の資料 [1] を公開している。

1 はじめに

(内部での熱発生を考慮した) 熱方程式の初期値境界値問題を考えよう。

Ω は $\mathbb{R}^2$ の有界領域で、その境界 Γ は区分的に十分滑らかであるとする。また Γ_1, Γ_2 は条件

$$\Gamma = \overline{\Gamma}_1 \cup \overline{\Gamma}_2, \quad \Gamma_1 \cap \Gamma_2 = \emptyset, \quad \Gamma_1 \neq \emptyset$$

を満たすとする。$f\colon \Omega \to \mathbb{R}$, $g_1\colon \Gamma_1 \to \mathbb{R}$, $g_2\colon \Gamma_1 \to \mathbb{R}$ が与えられたとき、次の方程式を満たす $u = u(x,t)$ を求めよ。

$$u_t(x,t) = \triangle u(x,t) + f(x) \quad ((x,t) \in \Omega \times (0,\infty)), \tag{1a}$$

$$u(x,t) = g_1(x) \quad ((x,t) \in \Gamma_1 \times (0,\infty)), \tag{1b}$$

$$\frac{\partial u}{\partial n}(x,t) = g_2(x) \quad ((x,t) \in \Gamma_2 \times (0,\infty)), \tag{1c}$$

$$u(x,0) = u_0(x) \quad (x \in \overline{\Omega}) \tag{1d}$$

この問題に対して、時間微分を θ 法で離散化したプログラムを作って、安定性の条件を調べてみよう。

θ 法では、$0 \le \theta \le 1$ を満たす θ を取って、微分方程式 $\dfrac{\partial u}{\partial t}(\cdot, t_n) = \triangle u(\cdot, t_n) + f$ を

$$\frac{1}{\Delta t}\left(u^{n+1} - u^n\right) = \triangle u^{n+\theta} + f \tag{2}$$

で置き換えるわけである。ここで

$$u^n := u(\cdot, t_n), \quad t_n := n\Delta t, \tag{3}$$

$$u^{n+\theta} := \theta u^{n+1} + (1-\theta)u^n = \theta u(\cdot, t_{n+1}) + (1-\theta)u(\cdot, t_n). \tag{4}$$

(4) を (2) に代入して

$$\frac{1}{\Delta t}\left(u^{n+1} - u^n\right) = \theta\triangle u^{n+1} + (1-\theta)\triangle u^n + f. \tag{5}$$

これから弱形式を導く。

後退 Euler 法については、弱形式とサンプル・プログラム `heatB.edp` を与えておいた (Ω, Γ_1, Γ_2 は Poisson 方程式のプログラム `poisson-kikuchi.edp` と同じに選んである)。

θ 法については、弱形式はあるが、プログラムがない。このような場合にプログラムを作るというのは、よくあるタイプの作業である。

heatB.edp を叩き台にして heatT.edp を作る

```
curl -O https://m-katsurada.sakura.ne.jp/program/fem/heatB.edp
cp heatB.edp heatT.edp
```

以下 heatT.edp を書き換えていこう。

2 後退 Euler 法のコードを読む

時間について θ 法で差分近似した方程式は

$$\frac{1}{\Delta t}\left(u^{n+1}-u^n\right)=\triangle u^{n+1}+f. \tag{6}$$

これに対する弱形式は以下の通り。

$$\left(u^{n+1},v\right)-(u^n,v)+\Delta t\langle u^{n+1},v\rangle-\Delta t(f,v)-\Delta t[g_2,v]=0\quad(v\in X). \tag{7}$$

(内積の定義の復習: $(u,v)=\displaystyle\int_\Omega uv\ dx$, $\langle u,v\rangle=\displaystyle\int_\Omega \nabla u\cdot\nabla v\ dx$, $\nabla u\cdot\nabla v=u_xv_x+u_yv_y$, $[g_2,v]=\displaystyle\int_{\Gamma_2} g_2v\ d\sigma$, $d\sigma$ は線要素)

FreeFem++ で対応するコードは

heatB.edp 内の problem

```
problem heat(u,v,init=n)=
  int2d(Th)(u*v)-int2d(Th)(uold*v)
 +int2d(Th)(tau*(dx(u)*dx(v)+dy(u)*dy(v)))
 -int2d(Th)(tau*f*v)-int1d(Th,2,3)(tau*g2*v)
 +on(1,4,u=g1);
```

(ラベル 1, 4 が Γ_1 を, ラベル 2, 3 が Γ_2 を構成している。)

このコードが弱形式 (7) の実現であることを読み取ろう。

注意 2.1 FreeFem++ の文法は制限が強く、自分で弱形式をコーディングしようとすると、色々な文法エラーに遭遇する。

- `int2d(Th)(u*v)-int2d(Th)(uold*v)` の部分を `int2d(Th)((u-uold)*v)` や `int2d(Th)(u*v-uold*v)` に置き換えることは出来ない。
- 一方で `int2d(Th)(u*v)` と `int2d(Th)(tau*(dx(u)*dx(v)+dy(u)*dy(v)))` を `int2d(Th)(u*v+tau*(dx(u)*dx(v)+dy(u)*dy(v)))` のように 1 つにまとめることはできる。
- 数`*int2d(Th)(...)` という式が扱えないので、`int2d(Th)(tau*(dx(u)*dx(v)+dy(u)*dy(v)))` の `tau` を `tau*int2d(Th)((dx(u)*dx(v)+dy(u)*dy(v)))` のようにくくり出すことは出来ない。

積分は複数の `int2d()()` にバラす、積分の前にある定数は被積分関数に入れてしまう、という方針が良いだろう (そう割り切ってからエラーに遭遇するのが減った)。■

3　θ 法のコードを作る

θ 法の場合の弱形式は次のようになる。

$$\frac{1}{\Delta t}\left(u^{n+1}-u^n, v\right) + \langle u^{n+\theta}, v\rangle - (f,v) - [g_2, v] = 0, \quad u^{n+\theta} := \theta u^{n+1} + (1-\theta)u^n. \tag{8}$$

すなわち

$$\left(u^{n+1}, v\right) - (u^n, v) + \Delta t\theta\langle u^{n+1}, v\rangle + \Delta t(1-\theta)\langle u^n, v\rangle - \Delta t(f,v) - \Delta t[g_2, v] = 0. \tag{9}$$

(7) と (9) を見比べると、$\Delta t\langle u^{n+1}, v\rangle$ が $\Delta t\theta\langle u^{n+1}, v\rangle + \Delta t(1-\theta)\langle u^n, v\rangle$ に変わっていることが分かる。

次のコードは弱形式 (9) を実現したものである。

コード 1

```
problem heat(u,v,init=i)=
  int2d(Th)(u*v)-int2d(Th)(uold*v)
 +int2d(Th)(theta*tau*(dx(u)*dx(v)+dy(u)*dy(v)))
 +int2d(Th)((1-theta)*tau*(dx(uold)*dx(v)+dy(uold)*dy(v)))
 -int2d(Th)(tau*f*v)-int1d(Th,2,3)(tau*g2*v)
 +on(1,4,u=g1);
```

もちろん、ばらす方は問題なく、次のようなコードでも良い。

コード 2

```
problem heat(u,v,init=i)=
  int2d(Th)(u*v)-int2d(Th)(uold*v)
 +int2d(Th)(theta*tau*(dx(u)*dx(v)))
 +int2d(Th)(theta*tau*(dy(u)*dy(v)))
 +int2d(Th)((1-theta)*tau*(dx(uold)*dx(v)))
 +int2d(Th)((1-theta)*tau*(dy(uold)*dy(v)))
 -int2d(Th)(tau*f*v)-int1d(Th,2,3)(tau*g2*v)
 +on(1,4,u=g1);
```

4　実験してみよう

1 次元熱方程式を差分法で解いた。以下のことが分かった。

- $\theta = 1$ (後退 Euler 法) であれば、最大値ノルムの意味で無条件に安定であった。
- ノルムによっては
  - $\theta \geq 1/2$ であれば無条件安定になった。
  - $0 < \theta < 1/2$ のとき、時間刻みが十分小さくない ($\Delta t \leq \dfrac{(\Delta x)^2}{2(1-2\theta)}$) と不安定になる。

`heatT.edp` で $\theta = 1, 1/2, 1/4$ の場合に、時間刻みを大きくしても安定性が保たれるか調べる。

このようにパラメーターを頻繁に変更する場合は、プログラム中でパラメーターの値を設定するのではなく、実行時に入力するのが良い場合がある。以下は、実行時にパラメーターをキーボード入力するためにどうすれば良いかの説明である。

キーボード入力が出来るように改造する

```
real Tmax=10, tau=0.01, t, theta=1; // theta を加える。=1 の時は実は後退 Euler
// 次のコメントアウトしている 2 行をアンコメントすると、m, tau, theta が実行時に入力できる
// cout << "m dt theta: "; cin >> m >> tau >> theta;
// cout << "m=" << m << ", tau=" << tau << ", theta=" << theta << endl;
mesh Th=square(m,m);
```

参考文献

[1] 桂田祐史：応用数値解析特論 2024 年度第 7 回, https://m-katsurada.sakura.ne.jp/ana2024/ANA07_0604_handout.pdf, 特に https://m-katsurada.sakura.ne.jp/lecture/ouyousuuchikaisekitokuron-2024/ANA07_0604_handout.pdf#page=27 に課題が載っている。