応用数値解析特論 第 8 回

～発展系の数値解析 (1)～

かつらだ まさし
桂田 祐史

http://nalab.mind.meiji.ac.jp/~mk/ana2021/

2021 年 11 月 15 日

目次

本日の内容

- 前回の授業中、図 5 をどのように描いたか質問されたので、スライド PDF に説明「図 5 をどのように描いたか」を書いておいた。
- FreeFem++の入出力は C++ 風のストリーム入出力である。簡単な説明を付録で用意した。
- 発展系の有限要素解析を説明するため、熱方程式に対する差分法を駆け足で解説する。
- それから、ようやく有限要素法で解く話になる。サンプル・プログラムを提示して解説する。あまり深い話は出来ない。

8 発展系の有限要素解析

8.1 準備 — 1 次元熱方程式の初期値境界値問題に対する差分法

時間の経過に伴い変化する系 (発展系) のシミュレーションを考える。

要点は、空間方向は有限要素近似し、時間方向は差分近似する、というもの。

そのため、差分法について大急ぎで説明する。

差分法について、より詳しい解説は、例えば桂田 [1], [2] の第 1 章を見よ。

次の熱方程式の初期値境界値を例題として取り上げる。

$$
\begin{aligned}
&\text{(1a)} && u_t(x,t) = u_{xx}(x,t) \quad ((x,t) \in (0,1) \times (0,\infty)), \\
&\text{(1b)} && u(0,t) = \alpha \quad (t \in (0,\infty)), \\
&\text{(1c)} && u_x(1,t) = \beta \quad (t \in (0,\infty)), \\
&\text{(1d)} && u(x,0) = u_0(x) \quad (x \in [0,1]).
\end{aligned}
$$

8.1.1 格子点

未知関数 u の定義域 $[0,1] \times [0,\infty)$ を “格子” に分割する。

具体的には、$N \in \mathbb{N}$, $\Delta t > 0$ として、

$$\Delta x := \frac{1}{N}, \quad x_i = i\Delta x \quad (i = 0, 1, \cdots, N),$$
$$t_n = n\Delta t \quad (n = 0, 1, \cdots),$$
$$u_i^n = u(x_i, t_n)$$

とおく。

Δt, Δx を**刻み幅** (stepsize), (x_i, t_n) を**格子点**と呼ぶ。

u は連続変数 x, t の関数であるが、それを求めることはあきらめて、u_i^n を求めることを目標にする。

8.1.2 差分近似の公式

f が C^2 級のとき (2a) と (2b) が、f が C^3 級のとき (2c) が、f が C^4 級のとき (2d) が成り立つ。

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \quad (h \to 0), \tag{2a}$$

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h) \quad (h \to 0), \tag{2b}$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (h \to 0), \tag{2c}$$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \quad (h \to 0). \tag{2d}$$

右辺の第 1 項をそれぞれ、前進差分商、後退差分商、1 階中心差分商、2 階中心差分商と呼ぶ。

左辺の導関数を右辺の第 1 項で近似することを、それぞれ前進差分近似、後退差分近似、1 階中心差分近似、2 階中心差分近似と呼ぶ。

8.1.3 熱方程式に対する差分方程式の導出

前のスライドに述べたことから

$$\frac{\partial u}{\partial t}(x_i, t_n) = \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t) \quad (\Delta t \to 0), \tag{3}$$

$$\frac{\partial u}{\partial t}(x_i, t_n) = \frac{u_i^{n} - u_i^{n-1}}{\Delta t} + O(\Delta t) \quad (\Delta t \to 0), \tag{4}$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_n) = \frac{u_{i-1}^{n} - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta x^2) \quad (\Delta x \to 0). \tag{5}$$

$u_t = u_{xx}$ が成り立つので、

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{u_{i-1}^{n} - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta t + \Delta x^2), \tag{6a}$$

$$\frac{u_i^{n} - u_i^{n-1}}{\Delta t} - \frac{u_{i-1}^{n} - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta t + \Delta x^2). \tag{6b}$$

この 2 つの式を参考に、次のスライドで差分方程式を立てる。

8.1.3 熱方程式に対する差分方程式の導出

(a) 前進 Euler 法

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2}.$$

これを書き直すと (ただし $\lambda := \Delta t/\Delta x^2$ とおく)

$$U_i^{n+1} = (1-2\lambda)U_i^n + \lambda\left(U_{i-1}^n + U_{i+1}^n\right) \quad (1 \le i \le N-1,\ n = 0, 1, 2, \cdots).$$

(b) 後退 Euler 法

$$\frac{U_i^n - U_i^{n-1}}{\Delta t} = \frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} \quad (1 \le i \le N-1,\ n = 1, 2, \cdots).$$

これを書き直すと

$$(1+2\lambda)U_i^{n+1} - \lambda\left(U_{i-1}^{n+1} + U_{i+1}^{n+1}\right) = U_i^n \quad (1 \le i \le N-1,\ n = 0, 1, 2, \cdots).$$

(c) θ 法　これは (a) と (b) を “混ぜた” ものである。$0 \le \theta \le 1$ を満たす θ を固定して

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = (1-\theta)\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} + \theta\frac{U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}}{\Delta x^2}$$

これを書き直すと

$$(7) \quad (1+2\theta\lambda)U_i^{n+1} - \theta\lambda\left(U_{i-1}^{n+1} + U_{i+1}^{n+1}\right) = \left[1 - 2(1-\theta)\lambda\right]U_i^n + (1-\theta)\lambda\left(U_{i-1}^n + U_{i+1}^n\right).$$

$\theta = 0$ のとき *(a)* の前進 *Euler* 法、$\theta = 1$ のとき *(b)* の後退 *Euler* 法と一致する。そこで以下では *(c)* θ 法の式のみ書く。$\theta = 1/2$ の場合は *Crank-Nicolson* 法と呼ばれる。

8.1.4 境界条件に対する差分方程式　Dirichlet 境界条件

$u(0,t)=\alpha$ であるから、次の方程式を課すのが自然であろう。

(8)
$$U_0^{n+1}=\alpha \quad (n=1,2,\dots).$$

$i=1$ の場合の (7)、つまり

$$(1+2\theta\lambda)U_1^{n+1}-\theta\lambda\left(U_0^{n+1}+U_2^{n+1}\right)=(1-2(1-\theta)\lambda)U_1^n+(1-\theta)\lambda\left(U_0^n+U_2^n\right)$$

に (8) を代入して、U_0^{n+1} を消去し、移項すると

(9)
$$(1+2\theta\lambda)U_1^{n+1}-\theta\lambda U_2^{n+1}=(1-2(1-\theta)\lambda)U_1^n+(1-\theta)\lambda\left(U_0^n+U_2^n\right)+\theta\lambda\alpha.$$

8.1.4 境界条件に対する差分方程式　Neumann 境界条件

$u_x(1, t_n) = \beta$ の近似としては、後退差分近似を用いた

$$\frac{U_N^{n+1} - U_{N-1}^{n+1}}{\Delta x} = \beta$$

が浮かぶが、この場合の誤差は $O(\Delta x)$ で精度が低い。

番号 i が $N+1$ である仮想格子点 (x_{N+1}, t_{n+1}) を導入すると、

$$\frac{U_{N+1}^{n+1} - U_{N-1}^{n+1}}{2\Delta x} = \beta \tag{10}$$

という 1 階中心差分近似ができる。この場合の誤差は $O(\Delta x^2)$ であり、後退差分近似よりも精度が高い。

このままでは方程式が不足するので、(7) が $i = N$ の場合にも成立すると仮定する。

$$(1+2\theta\lambda)U_N^{n+1} - \theta\lambda\left(U_{N-1}^{n+1} + U_{N+1}^{n+1}\right) = (1-2(1-\theta)\lambda)U_N^n + (1-\theta)\lambda\left(U_{N-1}^n + U_{N+1}^n\right).$$

(10) から得られる $U_{N+1}^{n+1} = 2\beta\Delta x + U_{N-1}^{n+1}$, $U_{N+1}^n = 2\beta\Delta x + U_{N-1}^n$ を代入すると

$$(1+2\theta\lambda)U_N^{n+1} - 2\theta\lambda U_{N-1}^{n+1} - 2\theta\lambda\beta\Delta x = [1-2(1-\theta)\lambda]\,U_N^n + 2(1-\theta)\lambda U_{N-1}^n + 2(1-\theta)\lambda\beta\Delta x.$$

整理して

$$(1+2\theta\lambda)U_N^{n+1} - 2\theta\lambda U_{N-1}^{n+1} = [1-2(1-\theta)\lambda]\,U_N^n + 2(1-\theta)\lambda U_{N-1}^n + 2\lambda\beta\Delta x. \tag{11}$$

8.1.5 差分方程式の行列・ベクトル表記

$2 \leq i \leq N-2$ に対する (7), (9), (11) は次のようにまとめられる。

$$\begin{bmatrix} 1+2\theta\lambda & -\theta\lambda & & & \\ -\theta\lambda & 1+2\theta\lambda & -\theta\lambda & & \\ & \ddots & \ddots & \ddots & \\ & & -\theta\lambda & 1+2\theta\lambda & -\theta\lambda \\ & & & -2\theta\lambda & 1+2\theta\lambda \end{bmatrix} \begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ \vdots \\ U_{N-1}^{n+1} \\ U_N^{n+1} \end{bmatrix}$$

$$= \begin{bmatrix} [1-2(1-\theta)\lambda]U_1^n + (1-\theta)\lambda\left(U_0^n + U_2^n\right) \\ \vdots \\ (1-2(1-\theta)\lambda)\,U_i^n + (1-\theta)\lambda\left(U_{i-1}^n + U_{i+1}^n\right) \\ \vdots \\ (1-2(1-\theta)\lambda)\,U_N^n + 2(1-\theta)\lambda U_{N-1} \end{bmatrix} + \begin{bmatrix} \theta\lambda\alpha \\ 0 \\ \vdots \\ 0 \\ 2\beta\lambda\Delta x \end{bmatrix}.$$

この右辺の第 1 項は ($U_0^n = \alpha$ に注意して) 次のように表せる。

$$\begin{bmatrix} 1-2(1-\theta)\lambda & (1-\theta)\lambda & & & \\ (1-\theta)\lambda & 1-2(1-\theta)\lambda & (1-\theta)\lambda & & \\ & \ddots & \ddots & \ddots & \\ & & (1-\theta)\lambda & 1-2(1-\theta)\lambda & (1-\theta)\lambda \\ & & & 2(1-\theta)\lambda & 1-2(1-\theta)\lambda \end{bmatrix} \begin{bmatrix} U_1^n \\ U_2^n \\ \vdots \\ U_{N-1}^n \\ U_N^n \end{bmatrix}.$$

8.1.6 差分スキームの安定性 (あらっぽい説明)

簡単のため、境界条件を同次 Dirichlet 境界条件 $u(0,t) = u(1,t) = 0$ とする。$\boldsymbol{U}^n = (U_i^n)$ とする。あるノルム $\|\cdot\|$ について

$$\sup_n \|\boldsymbol{U}^n\| \leq \text{初期値・境界値から定まる量}$$

が成り立つとき、**差分スキームは安定**である、という。

最大値ノルム $\|\boldsymbol{x}\| = \max_i |x_i|$ については、次の定理から安定性の条件が分かる。

定理 8.1 (離散最大値原理)

つねに $\max_{\substack{0\leq i\leq N\\ 0\leq n\leq J}} U_i^n = \max\left\{ \max_{0\leq i\leq N} U_i^0, \max_{0\leq n\leq J} U_0^n, \max_{0\leq n\leq J} U_N^n \right\}$ が成り立つには、

$\theta = 1$ または $(0 \leq \theta < 1$ かつ $\lambda \leq \frac{1}{2(1-\theta)})$ が必要十分。

例えば桂田 [1] を見よ。$\theta = 0$ (陽解法) の場合は、$\lambda \leq \frac{1}{2}$ であることに注意する (これは有名であろう)。

8.1.6 差分スキームの安定性 (あらっぽい説明)

差分解に対して

$$\boldsymbol{U}^{n+1} = R\boldsymbol{U}^n$$

のような行列 R が存在することが示される。$\boldsymbol{U}^n = R^n\boldsymbol{U}^0$ が成り立つ。

ノルムとして

$$\|\boldsymbol{x}\| = \left(\frac{1}{N}\sum_i |x_i|^2\right)^{1/2}$$

を採用した場合は、行列 R のスペクトル半径 (固有値の絶対値の最大値) で安定性の判定ができる。

$$\frac{1}{2} \le \theta \le 1 \quad \text{または} \quad (0 \le \theta < \frac{1}{2} \wedge 0 < \lambda \le \frac{1}{2(1-2\theta)})$$

であれば、R のスペクトル半径が 1 より小さいことが保証され、差分スキームの安定性が導かれる (例えば桂田 [3] を見よ。)。

8.1.7 大まかなまとめ

前進 Euler 法、後退 Euler 法、θ 法の差分方程式は、

$$\frac{\partial u}{\partial t}(\cdot, t_n) = \triangle u(\cdot, t_n)$$

を、それぞれ

$$\frac{u^{n+1} - u^n}{\Delta t} = \triangle u^n,$$
$$\frac{u^n - u^{n-1}}{\Delta t} = \triangle u^n \quad \text{すなわち} \quad \frac{u^{n+1} - u^n}{\Delta t} = \triangle u^{n+1}$$
$$\frac{u^{n+1} - u^n}{\Delta t} = \triangle[(1-\theta)u^n + \theta u^{n+1}]$$

と時刻について差分近似して、さらに空間についても差分近似して得られる、とみなせる (ただし、$u^n = u(\cdot, t_n)$ とおいた)。

以下では、我々は、時刻について差分近似して、空間については有限要素近似して近似方程式を作ることにする。

8.1.8 熱方程式の初期値境界値問題 (Dirichlet 境界条件) の差分法プログラム

差分法の安定性の話をする際に、従来サンプル・プログラムは C+GLSC で記述したものを紹介していたが、そういう環境を持っていない学生がいたので、以前冗談半分に FreeFem++ 言語で差分法によるプログラムを書いてみた。紹介しておく。

ただし境界条件が $u(0,t) = u(1,t) = 0$ $(t \in (0,\infty))$ の場合のプログラムである。

heat1d-e-freefem.edp を入手して実行

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/program/fem/heat1d-e-freefem.edp
FreeFem++ heat1d-e-freefem.edp
```

(最初に λ の値を入力する。初期値のグラフを描いて一時停止する。ウィンドウ内で [enter] キーを打って再開。遅いので途中で中断したくなるかも。)

λ が 0.5 の場合、0.51 の場合を比べてみることを勧める。

このプログラムをこの節の例題 (1a), (1b), (1c), (1d) を解くように書き換えた人がいたら、プログラムを下さい。

8.2 熱方程式に対する有限要素法

8.2.1 例題

熱方程式 (内部で熱が発生する or 熱が吸収される) の初期値境界値問題

$$u_t(x,t) = \triangle u(x,t) + f(x) \quad ((x,t) \in \Omega \times (0,\infty)), \tag{12a}$$

$$u(x,t) = g_1(x) \quad ((x,t) \in \Gamma_1 \times (0,\infty)), \tag{12b}$$

$$\frac{\partial u}{\partial \boldsymbol{n}}(x,t) = g_2(x) \quad ((x,t) \in \Gamma_2 \times (0,\infty)), \tag{12c}$$

$$u(x,0) = u_0(x) \quad (x \in \overline{\Omega}) \tag{12d}$$

を考える。

多くの設定は、これまで扱ってきた Poisson 方程式の境界値問題に準じる。

(ほぼ復習) Ω は $\mathbb{R}^2$ の有界領域で、その境界 $\Gamma := \partial\Omega$ は $\Gamma = \Gamma_1 \cup \Gamma_2$, $\Gamma_1 \cap \Gamma_2 = \emptyset$ と分割されているとする。$\boldsymbol{n}$ は Γ_2 上の点 x における外向き単位法線ベクトルである。また $u\colon \overline{\Omega} \times [0,\infty) \to \mathbb{R}$ は未知関数であり、それ以外の $u_0\colon \overline{\Omega} \to \mathbb{R}$, $f\colon \Omega \to \mathbb{R}$, $g_1\colon \Gamma_1 \to \mathbb{R}$, $g_2\colon \Gamma_2 \to \mathbb{R}$ は既知関数とする。

$f = f(x,t)$, $g_1 = g_1(x,t)$, $g_2 = g_2(x,t)$ と時間依存している場合の問題を解くのも難しくない。

8.2.2 解法の方針

時間微分については差分法で近似し、空間微分については有限要素法で近似する。つまり前節の最後に書いたように、まず

$$\frac{u^{n+1}-u^n}{\Delta t} = \triangle u^{n+1} + f \quad \text{(後退 Euler 法の場合)}, \tag{13a}$$

あるいは

$$\frac{u^{n+1}-u^n}{\Delta t} = \triangle[(1-\theta)u^n + \theta u^{n+1}] + f \quad (\theta \text{ 法の場合}) \tag{13b}$$

と時刻について差分近似してから、各時刻 t_{n+1} で (12b), (12c) と合わせて、Ω における境界値問題とみなす。すなわち、境界条件は

$$u^{n+1} = g_1 \quad (\text{on } \Gamma_1), \tag{14a}$$

$$\frac{\partial u^{n+1}}{\partial \boldsymbol{n}} = g_2 \quad (\text{on } \Gamma_2). \tag{14b}$$

(u^n を既知とすると、$-\triangle u^{n+1} + cu^{n+1} = F$ という形をしていて、Poisson 方程式ではないが、ほぼ同様に解くことが出来る)。

以下、内積の記号 $\langle\cdot,\cdot\rangle$, $(\cdot,\cdot)$, $[\cdot,\cdot]$ や、関数空間 $\hat{X}_{g_1}$, $\hat{X}$ は Poisson 方程式のときのものを利用する。

8.2.3 熱方程式に対する前進 Euler 法　余談

一応、時刻についての導関数 $\partial u/\partial t$ を前進差分近似した、前進 Euler 法についても述べておく。

弱形式は

$$\left(\frac{u^{n+1}-u^n}{\Delta t}, v\right) - \langle u^n, v\rangle - (f, v) - [g_2, v] \quad (v \in \hat{X}). \tag{15}$$

すなわち

$$\left(u^{n+1}, v\right) - (u^n, v) + \Delta t\langle u^n, v\rangle - \Delta t(f, v) - \Delta t[g_2, v] = 0 \quad (v \in \hat{X}) \tag{16}$$

となる。

これは私が不勉強なのかもしれないが、この方法を使うプログラムは見たことがない。差分法の場合と違って陽解法でないので (つまり u^{n+1} を求めるのに、結局は連立 1 次方程式を解く必要がある) メリットがないからだろうか (と考えている)。安定性を調べるのは意味があるので、数値実験してみても良いだろう。

8.2.4 熱方程式に対する後退 Euler 法

まず後退 Euler 法のプログラムを紹介しよう。弱形式は

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

すなわち

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

サンプル・プログラムを用意してある。

ターミナルではこうして入手

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/program/fem/heatB.edp
```

次の次のスライドに載せてある (Δt を tau (τ) という変数名にしてあるのに注意)。

ループの制御変数を i として、problem に `,init=i` と書き足すのがミソ。最初は i が 0 であるので init は false、それ以降は $\mathtt{i} \neq 0$ であるので init は true, と指示するのが工夫 (そうしないと毎ステップで行列を再構成してしまう)。連立 1 次方程式の係数行列が時刻 (したがって n) に依らないことに注意しよう。

8.2.4 熱方程式に対する後退 Euler 法　heatB.edp

```
// 菊地文雄，有限要素法概説，サイエンス社の Poisson 方程式の問題の非定常版（簡略版）
int i,m=10;
real Tmax=1, tau=0.01, t=0;
func f=1;
func g1=0;
func g2=0;
func u0=sin(pi*x)*sin(pi*y);
mesh Th=square(m,m);
plot(Th,wait=true);
fespace Vh(Th,P1);
Vh u=u0,uold,v;
problem heat(u,v,init=i)=
  int2d(Th)(u*v+tau*(dx(u)*dx(v)+dy(u)*dy(v)))
 -int2d(Th)(tau*f*v)
 -int1d(Th,2,3)(tau*g2*v)
 -int2d(Th)(uold*v)
 +on(1,4,u=g1);
for (i=0;i<Tmax/tau;i++) {
  uold=u;
  heat;
  t=(i+1)*tau;
  plot(u,wait=0);
}
plot(u,wait=1);
```

メモ: solve と problem にかかわるパラメーター

FreeFem++ ドキュメンテーションの §3.3.13 (Hecht [4]) から。

- `solver=` には LU, CG, Crout, Cholesky, GMRES, sparsesolver, UMFPACK が指定できる (最初の 5 つはアルゴリズムの名前, 説明省略)。
  デフォールトでは sparsesolver で、それは他の sparsesolver が定義されていなければ UMFPACK に等しい。それは他の直接法のソルバーが使えない場合は LU にセットされる。
  行列のメモリーへの格納の仕方は、solver により決まる。LU の場合は非対称なスカイライン (説明省略)、Crout の場合は対称なスカイライン、Cholesky の場合は正値対称なスカイライン、CG の場合は正値対称な疎行列、その他 (GMRES, sparsesolver, UMFPACK) では疎行列。
- `init=`論理型の式
  `false` または 0 のとき、行列が再構成 (reconstruct) される、とある。初期化されているかどうか、という意味か？
- `eps=`実数型の式
  反復法の停止則を指定する。
  $\varepsilon < 0$ の場合は $\|Ax - b\| < |\varepsilon|$, $\varepsilon > 0$ の場合は $\|Ax - b\| < \frac{|\varepsilon|}{\|Ax_0 - b\|}$
  (と書いてあるけれど、$\frac{\|Ax-b\|}{\|Ax_0-b\|} < |\varepsilon|$ の間違いではないかな？)

(連立 1 次方程式のアルゴリズムを学んだことがないと、少し分かりにくいかもしれない…)

8.2.5 熱方程式に対する θ 法

$$u^{n+\theta} := \theta u^{n+1} + (1-\theta)u^n$$

とおいて

$$\frac{u^{n+1}-u^n}{\Delta t} = \triangle u^{n+\theta} + f \quad (\text{in } \Omega), \tag{17a}$$

$$u^{n+1} = g_1 \quad (\text{on } \Gamma_1), \tag{17b}$$

$$\frac{\partial u^{n+1}}{\partial n} = g_2 \quad (\text{on } \Gamma_2). \tag{17c}$$

(もし f や g_2 が時間依存する場合は、f や g_2 は $t = t_{n+1}$ のときの値を用いる。)

弱形式は

$$\frac{1}{\Delta t}\left(u^{n+1}-u^n, v\right) + \langle u^{n+\theta}, v\rangle - (f,v) - [g_2, v] = 0. \tag{18}$$

すなわち

$$\left(u^{n+1}, v\right) - (u^n, v) + \Delta t\theta\langle u^{n+1}, v\rangle + \Delta t(1-\theta)\langle u^n, v\rangle - \Delta t(f,v) - \Delta t[g_2, v] = 0. \tag{19}$$

記憶用には (18) の方が短くて便利、プログラムを書くには (19) の方が便利であろう。

8.2.6 実習課題

1. 2 つのプログラム (`poisson-kikuchi-square.edp`, `heatB.edp`) を入手＆実行し、熱方程式版の最終結果 ($t = \mathrm{T}_{\max}$) が、Poisson 方程式の結果とほぼ同じであることを確認せよ。
2. θ 法のプログラム `heatT.edp` を作成せよ。
3. ある程度分割を細かくして、`init=` の指定の効果を調べよ。
   (指定しないと遅く、指定しないで `solver=CG` とすると少し速くなるが、CG 法にせず直接法の系統で `init=` を指定した方が速い (ようである)。)
   実行時間は `time` コマンドで計測できる (`time FreeFem++ heatB.edp`)。
4. 安定性について実験的に調べよ。(長方形領域における差分法では、$1/2 \leq \theta \leq 1$ の場合は無条件安定、$0 \leq \theta < 1/2$ の場合は $0 < \lambda \leq \dfrac{1}{2(1-2\theta)}$ が安定のための必要十分条件であった。ただし $\lambda = \dfrac{\Delta t}{\Delta x^2} + \dfrac{\Delta t}{\Delta y^2}$.
   … 有限要素法の場合は、このような簡単な判定条件は得られないが、θ が 1 に近い時、0 に近い時、Δt を変えて、安定に計算出来るかどうか試してみる。)
5. (もし出来れば) 厳密解が分かる問題を選び、誤差を調べよ。
6. 自分が選んだ問題 (領域などを変える) で数値実験してみよ。

8.2.7 その他

時間発展問題に有限要素法を適用したときの理論的解析について、日本語で読めるテキストはほとんどない (Poisson 方程式の解析より一段以上高度である)。

齊藤 [5] は貴重である。(齊藤 [6] というのもある)。

A. FreeFem++の real データの入出力の書式指定

- 何も指定しないと C 言語の %g 相当の出力になる。
- `cout.precision(`n`);` とすると、以下小数点以下の桁数は n になる。

```
cout.precision(15);
cout << "pi=" << pi << endl;
```

- 幅を指定するには `<< setw(`桁数`)` とする (これは毎回必要)。

```
cout << "pi=" << setw(20) << pi << endl;
```

- `cout.fixed;` とすると、以下固定小数点数形式 (C 言語の %f 相当) になる。

```
cout.fixed;
cout << "NA=" << NA << endl;
```

- `cout.scientific;` とすると、以下指数形式 (C 言語の %e 相当) になる。

```
cout.scientific;
cout << "pi=" << pi << endl;
```

- `cout.default;` とすると、以下デフォールト (C 言語の%g) に戻る。

A. FreeFem++の real データの入出力の書式指定　例

```
// testfloat.edp
real NA = 6.022e+23;
// デフォールト %g に相当
cout << "pi=" << pi << ", NA=" << NA << ", pi*NA=" << pi * NA << endl << endl;
// 幅を 20 に指定 %20g に相当
cout << "pi=" << setw(20) << pi << ", NA=" << setw(20) << NA
     << ", pi*NA=" << setw(20) << pi * NA << endl << endl;
// 小数点以下の桁数を 15 に指定 %20.15g に相当？
cout.precision(15);
cout << "pi=" << setw(20) << pi << ", NA=" << setw(20) << NA
     << ", pi*NA=" << setw(20) << pi * NA << endl << endl;
// 固定小数点数形式 %.15f に相当
cout.fixed;
cout << "pi=" << pi << ", NA=" << NA << ", pi*NA=" << pi * NA << endl << endl;
// %20.15f に相当
cout << "pi=" << setw(20) << pi << ", NA=" << setw(20) << NA
     << ", pi*NA=" << setw(20) << pi * NA << endl << endl;
// 指数形式 %20.15e に相当
cout.scientific;
cout << "pi=" << setw(20) << pi << ", NA=" << setw(20) << NA
       << ", pi*Na=" << setw(20) << pi * NA << endl << endl;
// %g 形式に戻す %.15g に相当
cout.default;
cout << "pi=" << pi << ", NA=" << NA << ", pi*Na=" << pi * NA << endl;
```

B. (おまけ) C++のストリーム入出力

前回述べたように、FreeFem++ の入出力は、C++の**ストリーム入出力**の機能に良く似ている (似ているけれど同じではない。同じにすれば良いのに。)。

ここでは C++ のストリーム入出力機能の大まかな説明を行う。

B.1 標準入力 cin, 標準出力 cout, 標準エラー出力 cerr

C++のソース・プログラムで次のようにしてあることを仮定する。

```
#include <iostream> // C の <stdio.h> に相当するような定番
#include <iomanip>  // setprecision() 等に必要
using namespace std;// こうしないと std::cin, std::cout, std::cerr とする必要
```

通常は、標準入力は端末のキーボードからの入力、標準出力は端末 (ターミナル) の画面への文字出力、標準エラー出力も端末の画面への文字出力、に結びつけられている (入出力のリダイレクトで、指定したファイルに結びつけることもできる)。

とりあえず、C 言語のプログラムの printf() を使う代わりに cout << 式, scanf() を使う代わりに cin >> 変数名 を使う、と覚える。

```
double a, b;
cout << "Hello, world" << endl; // endl は改行 \n である。
cout << "Please input two numbers: ";
cin >> a >> b;
cout << "a+b=" << a + b << ", a-b=" << a - b << ", a*b=" << a * b
     << ", a/b=" << a / b << endl;
```

B.2 数値の書式指定

C 言語の `printf()` での書式指定 `"%4d"`, `"%7.2f"`, `"%20.15e"`, `"%25.15g"` は、C++ で使うのはあきらめることを勧める。

- 幅の指定は `<< setw(桁数)` で行う。これは次のフィールドにしか影響しない (必要ならば毎回指定する)。
- 浮動小数点数の小数点以下の桁数の指定は `<< setprecision(桁数)` で行う。
- 浮動小数点数で固定小数点形式での出力の指定は、`<< fixed` で行う。
  (C 言語の `%f` に相当)
- 浮動小数点数で指数形式での出力の指定は、`<< scientific` で行う。
  (C 言語の `%e` に相当)
- 浮動小数点数でデフォールト形式での出力の指定は、`<< defaultfloat` で行う。
  (C 言語の `%g` に相当)

B.2 数値の書式指定

```
// testfloat.cpp --- ナンセンスな計算（円周率とアボガドロ数の積）
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

int main(void)
{
  double pi, NA;
  pi = 4.0 * atan(1.0);
  NA = 6.022e+23;
  cout << setprecision(15); //double は 10 進 16 桁弱なので。cout.precision(15); も可
  cout << fixed;
  cout << "π=" << setw(20) << pi << ", NA=" << setw(20) << NA
       << ", π Na=" << setw(20) << pi * NA << endl; // %20.15f 相当
  cout << scientific;
  cout << "π=" << setw(24) << pi << ", NA=" << setw(24) << NA
       << ", π Na=" << setw(24) << pi * NA << endl; // %24.15e 相当
  cout << defaultfloat;
  cout << "π=" << setw(20) << pi << ", NA=" << setw(20) << NA
       << ", π Na=" << setw(20) << pi * NA << endl; // %20.15g 相当
}
```

(実行してみると分かるが、意外と難しい…)

B.3 外部ファイルとの入出力

```
// sintable.cpp --- 0 度から 90 度までの sin の表 sin.txt を作る
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath>
using namespace std;

int main(void)
{
  int n=90;
  double x, dx, pi;
  pi = 4.0 * atan(1.0);
  dx = (pi / 2) / n;
  {
    ofstream out("sin.txt");
    out << fixed << setprecision(15); // %.15f
    for (int i = 0; i <= n; i++)
      out << setw(3) << i << setw(20) << sin(i * dx) << endl;
  } // ブロックを抜けるとファイルがクローズされる
  return 0;
}
```

参考文献

[1] 桂田祐史：発展系の数値解析,
http://nalab.mind.meiji.ac.jp/~mk/labo/text/heat-fdm-0.pdf (1997 年〜).

[2] 桂田祐史：熱方程式に対する差分法 I — 区間における熱方程式 —,
http://nalab.mind.meiji.ac.jp/~mk/labo/text/heat-fdm-1.pdf (1998 年〜).

[3] 桂田祐史：発展系の数値解析の続き,
http://nalab.mind.meiji.ac.jp/~mk/labo/text/heat-fdm-0-add.pdf (1997 年〜).

[4] Hecht, F.: Freefem++,
https://doc.freefem.org/pdf/FreeFEM-documentation.pdf, 以前は
http://www3.freefem.org/ff++/ftp/freefem++doc.pdf にあった。

[5] 齊藤宣一：熱方程式に対する有限要素法と誤差解析, 東大数理科学研究科の「応用数理特別講義 III」(2004 June) の講義ノートの縮小版。今は公開していないみたい。(2006 年 3 月 29 日).

[6] 齊藤宣一：発展方程式の数値解析 — 最大値原理, 解析半群と有限要素法,
http://www.infsup.jp/saito/materials/110827tsukuba1.pdf (2011/8/27).