

応用数値解析特論 第9回

～プログラミング言語 FreeFem++ (2), 発展問題 (1)～

かつらだ まさし
桂田 祐史

<http://nalab.mind.meiji.ac.jp/~mk/ana/>

2020年11月23日

1 本日の内容

2 FreeFem++言語 (続き)

- 有限要素法のための機能 (続き)
 - 弱形式を定義して解く
- 参考プログラム — 有限要素解の誤差を見る

3 発展系の有限要素解析

- 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
 - 格子点
 - 差分近似の公式
 - 熱方程式に対する差分方程式の導出
 - 境界条件に対する差分方程式
 - 差分方程式の行列・ベクトル表記
 - 差分スキームの安定性
 - 大まかなまとめ
 - 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

本日の内容

- 前回に引き続き言語としての FreeFem++ の説明、それと発展系のシミュレーション、それと付録の3本立て。
- FreeFem++ の説明は、今回は弱形式の定義周辺。おまけとして、厳密解が分かっている Poisson 方程式の問題を解いて、誤差の減衰の様子を見てみる。
- 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。
(差分法に詳しい人はスライドを斜め読みして構わない。)
- FreeFem++ の入出力は C++ 風のストリーム入出力である。C++ を知らない・慣れていない人向けに簡単な説明を用意した。

8.3 有限要素法のための機能 前回まで

border, mesh, fespace を簡単に説明した。

(蛇足?) 低レベル・プログラミング用のメモ mesh Th, fespace Vh(Th,P1) を定義して、Vh 型の u があるとする。

- 三角形要素の総数 Th.nt, 接点の総数 Th.nv, 領域の面積 Th.area
- i 番目の節点 ($0 \leq i \leq \text{Th.nt} - 1$) は Th(i)
その x 座標, y 座標, ラベルはそれぞれ Th(i).x, Th(i).y, Th(i).label
- k 番目の三角形 ($0 \leq k \leq \text{Th.nt} - 1$) は Th[k]
Th[k] の面積は Th[k].area である。
Th[k][j] という式は、Th[k] の局所節点番号 j の節点の全体節点番号を表す。
Th[k][j] の x 座標, y 座標はそれぞれ Th[k][j].x, Th[k][j].y
- 点 (x, y) を含む三角形の番号は Th(x,y).nuTriangle
- u の節点での値を集めた配列は u[] で表す。
u[].n (u.n でも同じ) は Th.nv と同じである。
 i 番目の節点での値 (授業中の式で $u^i = \hat{u}(P_i)$) は u[](i)
- u は補間多項式でもあり、 (x, y) での値は u(x,y) で得られる。

8.3.3 弱形式を定義して解く

いよいよ弱形式を定義する方法の説明である。大きく分けて3通りある。

- Ⓐ solve — 弱形式を与えると同時にそれを解く (弱解を求める)。
- Ⓑ problem — 弱形式を与えて問題を解く関数を定義する。
- Ⓒ varf, matrix — 弱形式を与えて連立1次方程式を作る。

これまで説明して来た次の Poisson 方程式の境界値問題を元に説明する。

$$(1a) \quad -\Delta u(x, y) = f(x, y) \quad ((x, y) \in \Omega)$$

$$(1b) \quad u(x, y) = g_1(x, y) \quad ((x, y) \in \Gamma_1)$$

$$(1c) \quad \frac{\partial u}{\partial \mathbf{n}}(x, y) = g_2(x, y) \quad ((x, y) \in \Gamma_2).$$

弱解 u は X_{g_1} に属し、次の弱形式を満たすものである。

$$(2) \quad \langle u, v \rangle = (f, v) + [g_2, v] \quad (v \in X).$$

ただし

$$(3) \quad X_{g_1} := \{w \mid w = g_1 \text{ on } \Gamma_1\}, \quad X := \{v \mid v = 0 \text{ on } \Gamma_1\}.$$

8.3.3 弱形式を定義して解く (a) solve を利用

Poisson 方程式の境界値問題を解くサンプル・プログラムでは、(a) を用いた。

solve で弱形式を定義して解く

```
solve Poisson(u,v)=  
    int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))-int2d(Th)(f*v)-int1d(Th,2,3)(g2*v)  
    +on(1,4,u=g1);
```

次はどの方法でも共通である。

- `dx()`, `dy()` はそれぞれ x , y での微分を表す。
高階の微分は `dxx()`, `dxy()`, `dyy()` のようにする。
- `int2d(Th)` は考えている領域全体の積分 (重積分) を表す。
また `int1d(Th,2,3)` は境界のうち、ラベルが 2,3 である部分 (正方形の右と上) の積分 (境界積分、今の場合は線積分) を表す。
- `+on(1,4,u=g1)` は境界のうち、ラベルが 1,4 である部分 (正方形の下と左) で、 $u = g_1$ という Dirichlet 境界条件を課すことを表す (`+on(1,u=g1)+on(4,u=g1)` と分けて書くことも可能)。
ベクトル値関数の場合は、`+on(1,u1=g1,u2=g2)` のように複数の方程式を書くこともできる。

以下、この問題の場合に、(b), (c) がどうなるか示す。

8.3.3 弱形式を定義して解く (b) problem を利用

(b) problem を利用する方法では、次のようになる。

problem で弱形式を定義して解く

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

この問題の場合は、`solve` と比べての利点は特に感じられないかもしれないが、時間発展の問題では、同じ形の弱形式を何度も解く必要が生じるので、有効である。

8.3.3 弱形式を定義して解く (c) varf, matrix を利用

varf, matrix を利用

```
real Tgv=1.0e+30; // tgv と小文字でも可
varf a(u,v)=
  int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))
  +on(1,4,u=g1);
matrix A=a(Vh,Vh,tgv=Tgv,solver=CG);
varf l(UNUSED,v)=
  int2d(Th)(f*v)+int1d(Th,2,3)(g2*v)
  +on(1,4,UNUSED=0);
Vh F;
F[]=l(0,Vh,tgv=Tgv);
u[]=A^-1*F[];
```

あらすじは、連立 1 次方程式 $Au = f$ の A, f を別々に計算して、 $A^{-1}f$ を計算することで u を得る、ということである (詳細は、実は現時点で把握していないので省略する)。

tgv (terrible great value) は以前説明した。

solver= は連立 1 次方程式の解法を指定する。

CG **CG 法** (共役勾配法) (反復法, 正値対称行列 (spd) 用)

GMRES **GMRES 法** (反復法, 一般の正則行列用)

UMFPACK **UMFPACK** を利用 (直接法, 一般の正則行列用)

sparsesolver ダイナミック・リンクで**外部のソルバー**を呼ぶ

8.4 参考プログラム — 有限要素解の誤差を見る

有限要素解の収束、誤差の減衰は本科目の最後に説明する予定であるが、見ておこう。
真の解 u , 有限要素解 \hat{u} について

$$\|u - \hat{u}\|_{L^2} = \left(\iint_{\Omega} |u(x, y) - \hat{u}(x, y)|^2 dx dy \right)^{1/2}$$

を L^2 誤差、

$$\|u - \hat{u}\|_{H^1} = \left(\|u - \hat{u}\|_{L^2}^2 + \|u_x - \hat{u}_x\|_{L^2}^2 + \|u_y - \hat{u}_y\|_{L^2}^2 \right)^{1/2}$$

を H^1 誤差と呼ぶ。

分割を細かくしたとき、どのように減衰するか、厳密解が分かる問題で調べてみよう。

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/program/fem/poisson-mixedBC-mk.edp
FreeFem++ poisson-mixedBC-mk.edp
FreeFem++ poisson-mixedBC-mk.edp 2
FreeFem++ poisson-mixedBC-mk.edp 4
...
FreeFem++ poisson-mixedBC-mk.edp 64
```

正方形の辺を $20 \cdot 2^m$ ($m = 0, 1, \dots, 6$) 分割したときの誤差を近似計算する。

8.4 参考プログラム — 有限要素解の誤差を見る

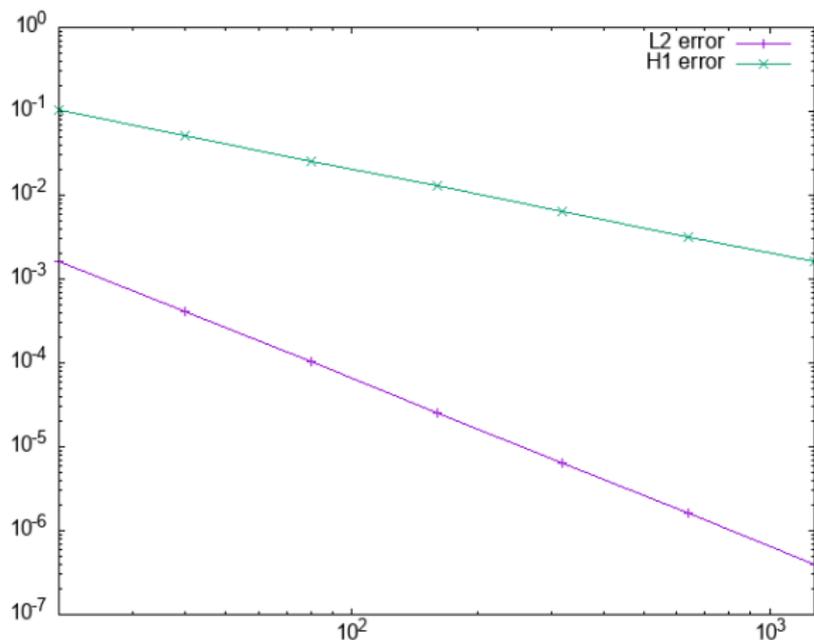


図 1: 1 辺を $n = 20, 40, 80, \dots, 1280$ 分割したときの誤差 (横軸 n)

L^2 誤差は $O(n^{-2})$, H^1 誤差は $O(n^{-1})$ となっている (ように見える)。

9 発展系の有限要素解析

9.1 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。既に慣れている人は、スライド PDF を斜め読みして「今週は短くて良かった」で構わない。

差分法について、より詳しい解説は、例えば桂田 [1] と [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。

$$(4a) \quad u_t(x, t) = u_{xx}(x, t) \quad ((x, t) \in (0, 1) \times (0, \infty)),$$

$$(4b) \quad u(0, t) = \alpha \quad (t \in (0, \infty)),$$

$$(4c) \quad u_x(1, t) = \beta \quad (t \in (0, \infty)),$$

$$(4d) \quad u(x, 0) = u_0(x) \quad (x \in [0, 1]).$$

9.1.1 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、

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

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。

9.1.2 差分近似の公式

f が C^2 級するとき (5a) と (5b) が、 f が C^3 級するとき (5c) が、 f が C^4 級するとき (5d) が成り立つ。

$$(5a) \quad f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \quad (h \rightarrow 0),$$

$$(5b) \quad f'(x) = \frac{f(x) - f(x-h)}{h} + O(h) \quad (h \rightarrow 0),$$

$$(5c) \quad f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (h \rightarrow 0),$$

$$(5d) \quad f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \quad (h \rightarrow 0).$$

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。

9.1.3 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

$$(6) \quad \frac{\partial u}{\partial t}(x_i, t_n) = \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t),$$

$$(7) \quad \frac{\partial u}{\partial t}(x_i, t_n) = \frac{u_i^n - u_i^{n-1}}{\Delta t} + O(\Delta t),$$

$$(8) \quad \frac{\partial^2 u}{\partial x^2}(x_i, t_n) = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta x^2).$$

$u_t = u_{xx}$ が成り立つので、

$$(9a) \quad \frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta t + \Delta x^2),$$

$$(9b) \quad \frac{u_i^n - u_i^{n-1}}{\Delta t} - \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta t + \Delta x^2).$$

この2つの式を参考に、次のスライドの差分方程式を立てる。

9.1.3 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)

$$U_i^{n+1} = (1 - 2\lambda)U_i^n + \lambda(U_{i-1}^n + U_{i+1}^n) \quad (1 \leq i \leq N - 1, n = 0, 1, 2, \dots).$$

(b) 後退 Euler 法

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

これを書き直すと

$$(1 + 2\lambda)U_i^{n+1} - \lambda(U_{i-1}^{n+1} + U_{i+1}^{n+1}) = U_i^n \quad (1 \leq i \leq N - 1, n = 0, 1, 2, \dots).$$

(c) θ 法 これは (a) と (b) を “混ぜた” ものである。 $0 \leq \theta \leq 1$ 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

(10)

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

$\theta = 0$ とすると (a) の前進 Euler 法、 $\theta = 1$ とすると (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。

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

$$(11) \quad U_0^{n+1} = \alpha \quad (n = 1, 2, \dots).$$

$i = 1$ の場合の (10)、つまり

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

に (11) を代入して、 U_0^{n+1} を消去し、移項すると

$$(12) \quad (1 + 2\theta\lambda)U_1^{n+1} - \theta\lambda U_2^{n+1} = (1 - 2(1 - \theta)\lambda)U_1^n + (1 - \theta)\lambda(U_0^n + U_2^n) + \theta\lambda\alpha.$$

$u_x(1, t_n) = \beta$ 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、

$$(13) \quad \frac{U_{N+1}^{n+1} - U_{N-1}^{n+1}}{2\Delta x} = \beta$$

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。

このままでは差分方程式が不足するので、(10) が $i = N$ の場合に成立すると仮定する。

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

(13) 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

$$(14) \quad (1 + 2\theta\lambda)U_N^{n+1} - 2\theta\lambda U_{N-1}^{n+1} = [1 - 2(1 - \theta)\lambda]U_N^n + 2(1 - \theta)\lambda U_{N-1}^n + 2\lambda\beta\Delta x.$$

9.1.5 差分方程式の行列・ベクトル表記

$2 \leq i \leq N-2$ に対する (10), (12), (14) は次のようにまとめられる。

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

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。

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

9.1.6 差分スキームの安定性

(工事中…次回に回します)

- $\theta = 0$ の場合。つねに

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

$0 < \lambda \leq \frac{1}{2}$ が必要十分。

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

- $1/2 \leq \theta \leq 1$ か、 $0 < \theta < 1/2$ かつ $0 < \lambda \leq \frac{1}{2(1-2\theta)}$ であれば、...

9.1.7 大まかなまとめ

前進 Euler 法、後退 Euler 法、 θ 法の差分方程式は、

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

を、それぞれ

$$\frac{u^{n+1} - u^n}{\Delta t} = \Delta u^n,$$

$$\frac{u^n - u^{n-1}}{\Delta t} = \Delta u^n \quad \text{すなわち} \quad \frac{u^{n+1} - u^n}{\Delta t} = \Delta u^{n+1}$$

$$\frac{u^{n+1} - u^n}{\Delta t} = \Delta[(1 - \theta)u^n + \theta u^{n+1}]$$

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)。

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。

9.1.8 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。

heat1d-e-freefem.edp

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

このプログラムをこの節の例題 (4a), (4b), (4c), (4d) 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。

A 付録: 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。

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。

A.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、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;
```

A.2 数値の書式指定

`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ルト形式での出力の指定は、`<< defaultfloat` で行う。
(C 言語の `%g` に相当)

A.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 << 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…)

A.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～).