数値計算/シミュレーションハンドブック

Ver. 1.0.0

勉強会参加メンバー

2016 年 8 月 25 日

目次

記号のリスト

d　次元

h_t　時間のステップ幅

h_x　空間のステップ幅

第 1 章

本稿に関して

1.1 本稿の目的

しばしば耳にする「数値計算」や「数値シミュレーション」,「仮想実験」などは一度は少し踏み込んだところまで触っておきたいもの．そのような簡単な理由から，分野問わず幅広い知見から上記項目をまとめていくことを目的にしている．特に，専門的になりすぎず初学者がとっつきやすいノート (ハンドブックと名付けた理由) になればと思っている．ネット上にはたくさんの知識が転がっている．特に，同じような目的からか wiki なりまとめなりのサイトも増え始めている．これらネットに転がっている知識もこのノートに集約できるなら最高の幸せでる．いつの日かこのノートのおかげで数値計算に入り込むことができたという人が現れてくれたら嬉しいな．なお，本稿のデザインは及びから拝借している．この場を持って感謝申し上げる．

1.2 ルール

短いコードなら pdf に書き込み可.

1.3 TEX

索引は index で可能.

1.4 事前準備

第 2 章

線型方程式

2.1 線形方程式の解法の種類

線型方程式

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (\text{簡潔に } \boldsymbol{Ax} = \boldsymbol{b}) \tag{2.1.1}$$

を計算機を使って解くことを考える.

1. 直接法
   - i. Gauss の消去法
   - ii. LU 分解
2. 反復法
   - 定常法
     - i. Jacobi 法
     - ii. Gauss-Seidel 法
     - iii. SOR 法 (Successive Over-Relaxation: 逐次加速緩和法)
   - 非定常法
     - i. Krykov 部分空間法
     - ii. CG (Conjugate Gradient: 共役勾配法)
     - iii. BiCGSTAB (Bi-Conjugate Gradient Stabilized)
     - iv. GMRES (Generalized Minimal Residual)

本章では，これらの解法アルゴリズムの紹介をする.

2.2 直接法

2.2.1 Gauss の消去法

2.2.2 LU 分解

2.3 反復法

直接法では，計算量が多い上に，そもそも計算ができない場合がある．そこで反復法を用いる．反復法は，初期値 $\boldsymbol{x}^{(0)}$ から始めて繰り返し計算を行うことで解へ収束させていく手法である.

2.3.1 Jacobi 法

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right). \tag{2.3.1}$$

2.3.2 Gauss-Seidel 法

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right). \tag{2.3.2}$$

2.3.3 SOR 法

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)} \right) \tag{2.3.3}$$

2.3.4 Krylov 部分空間法

第 3 章

常微分方程式の数値計算

偏微分方程式の数値計算において

3.1 問題設定

常微分方程式

$$\frac{du(t)}{dt} = f(t, u(t)) \quad (t > 0)$$

について考えよう.

3.1.1 Runge-Kutta 法

第 4 章

偏微分方程式の差分法

もっとも基礎的な数値スキームである有限差分法 (Finite Difference Method; FDM) を用いて，いくつかの有名な線形偏微分方程式を計算する．

4.1 Poisson 方程式

線形の Poisson 方程式は

$$-\Delta u(x) = f(x) \quad (x \in \Omega) \tag{4.1.1}$$

と記述される．

4.1.1 1 次元の問題

まず，1 次元の場合 ($\Omega = (0,1)$) について考えてみる．一般の区間への拡張は容易である．

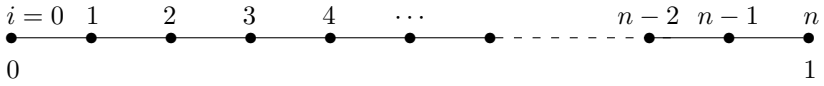

図 4.1　空間分割のイメージ (n 等分割した場合; $h = 1/n$)

(4.1.1) は2 階の中心差分を通して

$$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f_i \quad (i = 1, \cdots, n-1),$$

と離散化される．ただし，$f_i = f(x_i)$ である．また，これは

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{bmatrix}, \quad \boldsymbol{u}_b = \begin{bmatrix} -u_0 \\ 0 \\ \vdots \\ 0 \\ -u_n \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} h^2 f_1 \\ h^2 f_2 \\ \vdots \\ h^2 f_{n-1} \end{bmatrix} \tag{4.1.2}$$

と定義することにより，線型方程式

$$\boldsymbol{A}\boldsymbol{u} + \boldsymbol{u}_b = \boldsymbol{f} \tag{4.1.3}$$

に書き換えられる．ここで，$\boldsymbol{u}_b$ は境界条件から決まる．

周期境界

周期境界条件

$$u(0) = u(1) \tag{4.1.4}$$

を仮定する．

Dirichlet 問題

Dirichlet 境界値

$$u(0) = a, \quad u(1) = b \quad (a, b \in \mathbb{R}) \tag{4.1.5}$$

を仮定すれば，$u_0 = a$ かつ $u_n = b$ である．したがって，(4.1.3) は

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}_D$$

へと簡約化される．なお，

$$\boldsymbol{f}_D = \begin{bmatrix} h^2 f_1 + a \\ h^2 f_2 \\ \vdots \\ h^2 f_{n-2} \\ h^2 f_{n-1} + b \end{bmatrix}$$

である．

Neumann 問題

Neumann 境界値

$$u'(0) = a, \quad u'(1) = b \quad (a, b \in \mathbb{R}) \tag{4.1.6}$$

を仮定してみる．これらを

$$\frac{u_1 - u_0}{h} = a, \quad \frac{u_n - u_{n-1}}{h} = b$$

と近似すれば，$u_0 = u_1 - ah$ と $u_n = u_{n-1} - bh$ であるので，(4.1.3) は

$$\boldsymbol{A}_N \boldsymbol{u} = \boldsymbol{f}_N$$

と簡約化される．ただし，

$$\boldsymbol{A}_N = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}, \quad \boldsymbol{f}_N = \begin{bmatrix} h^2 f_1 - ah \\ h^2 f_2 \\ \vdots \\ h^2 f_{n-2} \\ h^2 f_{n-1} - bh \end{bmatrix} \tag{4.1.7}$$

である．

4.1.2 2 次元の問題

次に，2 次元の場合を考えてみる．領域は簡単のため，正方形[*1]($\Omega = (0,1) \times (0,1)$) とする．長方形などへの一般化は容易である．

(4.1.1) は，より具体的には

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

[*1] 矩型 (くけい) 領域の厳密な定義はなんでしょうかね？

と書けている．x 方向と y 方向それぞれに2 階の中心差分を適用すれば

$$-\frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{h_x^2}-\frac{u_{i,j-1}-2u_{i,j}+u_{i,j+1}}{h_y^2}=f_{i,j}$$

と離散化できる．

4.2 熱方程式

線形の熱方程式は

$$\partial_t u(t,x)-c\Delta u(t,x)=f(t,x)\quad (t>0,\ x\in\Omega) \tag{4.2.1}$$

と記述される．ここで，$c>0$ は拡散係数 (または粘性係数) と呼ばれる (今回は) 定数である．

4.2.1　1 次元の問題

1 次元の場合 ($\Omega=(0,1)$) を考える．

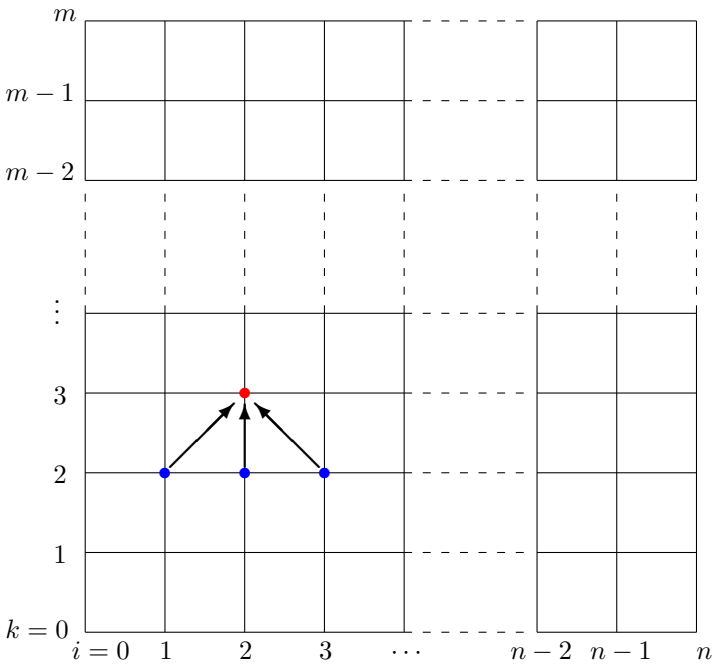

図 4.2　時空間分割のイメージ (空間方向に n 等分割，時間方向に m 等分割した場合; $h_x=1/n$, $h_t=t_{max}/m$)

まず，空間に関して2 階の中心差分を適用すれば

$$\frac{du_i(t)}{dt}-c\frac{u_{i-1}(t)+2u_i(t)-u_{i+1}(t)}{h_x^2}=f_i(t)\quad (t>0,\ i=1,\cdots,n-1) \tag{4.2.2}$$

を得る．[*2]さらに，前進差分を適用することで

$$\begin{aligned}&\frac{u_i^{k+1}-u_i^k}{h_t}-c\frac{u_{i-1}^k-2u_i^k+u_{i+1}^k}{h_x^2}=f_i^k\\ \Leftrightarrow\ &u_i^{k+1}=\lambda u_{i-1}^k+(1-2\lambda)u_i^k+\lambda u_{i+1}^k+h_t f_i^k\quad (i=1,\cdots,n-1,\ k=0,\cdots,m-1)\end{aligned} \tag{4.2.3}$$

[*2] このような離散化を (空間方向への) 半離散化と呼ぶ．対して，空間方向・時間方向ともに離散化することを全離散化と呼ぶこともある．

が導出される．ここで，$\lambda := ch_t/h_x^2$ である．もし

$$\boldsymbol{A} = \begin{bmatrix} 1-2\lambda & \lambda & & & \\ \lambda & 1-2\lambda & \lambda & & \\ & \ddots & \ddots & \ddots & \\ & & \lambda & 1-2\lambda & \lambda \\ & & & \lambda & 1-2\lambda \end{bmatrix}, \quad \boldsymbol{u}^k = \begin{bmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_{n-1}^k \end{bmatrix}, \quad \boldsymbol{u}_b^k = \begin{bmatrix} \lambda u_0^k \\ 0 \\ \vdots \\ 0 \\ \lambda u_n^k \end{bmatrix}, \quad \boldsymbol{f}^k = \begin{bmatrix} h_t f_1^k \\ h_t f_2^k \\ \vdots \\ h_t f_{n-1}^k \end{bmatrix} \tag{4.2.4}$$

を導入するならば，(4.2.3) は

$$\boldsymbol{u}^{k+1} = \boldsymbol{A}\boldsymbol{u}^k + \boldsymbol{u}_b^k + \boldsymbol{f}^k \quad (k = 0, \cdots, m-1)$$

と書くこともできる．初期条件が

$$u(0, x) = g(x) \quad (0 \le x \le 1) \tag{4.2.5}$$

と与えられたとすれば，$u_i^0 = g_i$ すなわち $\boldsymbol{u}^0 = \boldsymbol{g}$ であるので，結局

$$\begin{cases} \boldsymbol{u}^{k+1} = \boldsymbol{A}\boldsymbol{u}^k + \boldsymbol{u}_b^k + \boldsymbol{f}^k \quad (k = 0, \cdots, m-1) \\ \boldsymbol{u}^0 = \boldsymbol{g} \end{cases} \tag{4.2.6}$$

が得られる．

Cauchy-Dirichlet **問題**

Dirichlet 境界条件

$$u(t, 0) = a(t), \quad u(t, 1) = b(t) \quad (t \ge 0) \tag{4.2.7}$$

を仮定する．ただし，$a, b : [0, \infty) \to \mathbb{R}$ は両立条件として $g(0) = a(0)$ と $g(1) = b(0)$ を満たすと仮定する．(4.2.7) から $u_0^k = a^k$ と $u_n^k = b^k$ を得るので，(4.2.6) は

$$\begin{cases} \boldsymbol{u}^{k+1} = \boldsymbol{A}\boldsymbol{u}^k + \boldsymbol{f}_D^k \quad (k = 0, \cdots, m-1) \\ \boldsymbol{u}^0 = \boldsymbol{g} \end{cases} \tag{4.2.8}$$

となる．ここで，

$$\boldsymbol{f}_D^k = \begin{bmatrix} h_t f_1^k + \lambda a^k \\ h_t f_2^k \\ \vdots \\ h_t f_{n-2}^k \\ h_t f_{n-1}^k + \lambda b^k \end{bmatrix} \tag{4.2.9}$$

Cauchy-Neumann **問題**

Neumann 境界条件

$$u_x(t, 0) = a(t), \quad u_x(t, 1) = b(t) \quad (t \ge 0) \tag{4.2.10}$$

の場合は，

$$\frac{u_1^k - u_0^k}{h_x} = a^k, \quad \frac{u_n^k - u_{n-1}^k}{h_x} = b^k$$

と近似すれば，$u_0^k = u_1^k - a^k h_x$ と $u_{n+1}^k = u_n^k - b^k h_x$ であるので，(4.2.6) は

$$\begin{cases} \boldsymbol{u}^{k+1} = \boldsymbol{A}_N\boldsymbol{u}^k + \boldsymbol{f}_N^k \quad (k = 0, \cdots, m-1) \\ \boldsymbol{u}^0 = \boldsymbol{g} \end{cases} \tag{4.2.11}$$

となる．ここで，

$$\boldsymbol{A}_N = \begin{bmatrix} 1-\lambda & \lambda & & & \\ \lambda & 1-2\lambda & \lambda & & \\ & \ddots & \ddots & \ddots & \\ & & \lambda & 1-2\lambda & \lambda \\ & & & \lambda & 1-\lambda \end{bmatrix}, \quad \boldsymbol{f}_N^k = \begin{bmatrix} h_t f_1^k - h_t a^k \\ h_t f_2^k \\ \vdots \\ h_t f_{n-1}^k \\ h_t f_n^k - h_t b^k \end{bmatrix} \tag{4.2.12}$$

である．

4.2.2 2 次元の問題

4.2.3 問題

- 他のスキームを用いた場合はどうなるか．
- 拡散係数が定数でない場合，すなわち

$$u_t(t,x) - \mathrm{div}\,(c(t,x)\nabla u(t,x)) = f(t,x) \quad (t>0,\ x\in\Omega)$$

の場合はどうなるか．

4.3 波動方程式

線形の波動方程式は

$$u_{tt}(t,x) - c^2\Delta u(t,x) = f(t,x) \quad (t>0,\ x\in\Omega) \tag{4.3.1}$$

と記述される．

4.3.1 1 次元の問題

1 次元 ($\Omega=(0,1)$) の場合について考えてみる．空間方向に2 階の中心差分を適用すると

$$\frac{d^2u_i(t)}{dt^2} - c^2\frac{u_{i-1}(t)-2u_i(t)+u_{i+1}(t)}{h_x^2} = f_i(t) \quad (t>0,\ i=1,\cdots,n-1)$$

と半離散化される．さらに，時間方向にも2 階の中心差分を適用すると

$$\begin{aligned}&\frac{u_i^{k-1}-2u_i^k+u_i^{k+1}}{h_t^2} - c^2\frac{u_{i-1}^k-2u_i^k+u_{i+1}^k}{h_x^2} = f_i^k\\ \Leftrightarrow\ & u_i^{k+1} = \lambda u_{i-1}^k + 2(1-\lambda)u_i^k + \lambda u_{i+1}^k - u_i^{k-1} + h_t^2 f_i^k \quad (i=1,\cdots,n-1,\ k=1,\cdots,m-1)\end{aligned} \tag{4.3.2}$$

を得る．ここで，$\lambda := ch_t^2/h_x^2$．もし

$$\boldsymbol{A} = \begin{bmatrix} 2(1-\lambda) & \lambda & & & \\ \lambda & 2(1-\lambda) & \lambda & & \\ & \ddots & \ddots & \ddots & \\ & & \lambda & 2(1-\lambda) & \lambda \\ & & & \lambda & 2(1-\lambda) \end{bmatrix}, \quad \boldsymbol{u}^k = \begin{bmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_{n-1}^k \end{bmatrix}, \quad \boldsymbol{u}_b^k = \begin{bmatrix} \lambda u_0^k \\ 0 \\ \vdots \\ 0 \\ \lambda u_n^k \end{bmatrix}, \quad \boldsymbol{f}^k = \begin{bmatrix} h_t^2 f_1^k \\ h_t^2 f_2^k \\ \vdots \\ h_t^2 f_{n-1}^k \end{bmatrix} \tag{4.3.3}$$

を導入するならば，(4.3.2) は

$$\boldsymbol{u}^{k+1} = \boldsymbol{A}\boldsymbol{u}^k + I\boldsymbol{u}^{k-1} + \boldsymbol{u}_b^k + \boldsymbol{f}^k \quad (k=1,\cdots,m-1)$$

と行列表現できる．ここで，$I := \mathrm{diag}\,[1,\cdots,1]$ である．初期条件として

$$u(0,x) = g_1(x), \quad u_t(0,x) = g_2(x) \quad (0\le x\le 1) \tag{4.3.4}$$

を仮定するなら，$u_i^0 = (g_1)_i$ と，$(u_i^1-u_i^0)/h_t = (g_2)_i$ から $u_i^1 = u_i^0 + h_t(g_2)_i = (g_1)_i + h_t(g_2)_i$ と求まるので，結局

$$\begin{cases} \boldsymbol{u}^{k+1} = \boldsymbol{A}\boldsymbol{u}^k + I\boldsymbol{u}^{k-1} + \boldsymbol{u}_b^k + \boldsymbol{f}^k & (k=1,\cdots,m-1) \\ \boldsymbol{u}^0 = \boldsymbol{g}_1, \quad \boldsymbol{u}^1 = \boldsymbol{g}_1 + h_t\boldsymbol{g}_2 \end{cases} \tag{4.3.5}$$

を解くことになる．

Cauchy-Dirichlet **問題**

Dirichlet 境界条件 (4.2.7) を仮定する．

Cauchy-Neumann **問題**

4.3.2 2 次元の問題

4.3.3 問題

- 拡散係数が定数でない場合，すなわち

$$u_{tt}(t,x) - \operatorname{div}\left(c(t,x)\nabla u(t,x)\right) = f(t,x) \quad (t>0,\ x\in\Omega)$$

  の場合はどうなるか．

4.4 移流方程式

線形の (スカラー) 移流方程式[*3]は

$$\partial_t u(t,x) + \nu\cdot\nabla u(t,x) = f(t,x) \quad (t>0,\ x\in\Omega) \tag{4.4.1}$$

によって記述される．ここで，$u:\Omega\to\mathbb{R}$ は未知関数，$\nu\in\mathbb{R}^d$ はゼロでない定数ベクトルであり，$f=f(t,x):(0,\infty)\times\Omega\to\mathbb{R}$ は既知の関数ある．もし $\Omega=\mathbb{R}^d$ ならば，解は

$$u(t,x) = u(0,x-t\nu) + \int_0^t f(s,x+(s-t)\nu)ds \quad (x\in\mathbb{R}^d, t\geq 0)$$

とかける ([1, Section 2.1.2])

4.4.1 離散化

4.4.2 移流拡散方程式

前々節と合わせることで，線形の移流拡散方程式[*4]

$$u_t(t,x) - c\Delta u(t,x) + \nu\cdot\nabla u(t,x) = f(t,x) \quad (t>0,\ x\in\Omega) \tag{4.4.2}$$

の計算は可能である．

4.4.3 問題

すぐに思い浮かぶ問題としては例えば次がある．

- Neumann 境界条件を課した場合はどうなるか．
- (4.4.1) において移流係数が定数でない場合[*5]，すなわち

$$u_t(t,x) + \nabla(\nu(t,x)u(t,x)) = f(t,x) \quad (t>0,\ x\in\Omega)$$

  の場合はどうなるか．ただし，ν は既知のスカラー値関数である．

[*3] advection equation
[*4] advection diffusion equation
[*5] Liouville 方程式と呼ばれることがあるようだ．

- (4.4.2) において移流係数または/及び拡散係数が定数でない場合[*6]，すなわち

$$u_t(t,x) - \mathrm{div}\,(c(t,x)\nabla u(t,x)) + \nabla(\nu(t,x)u(t,x)) = f(t,x) \quad (t>0,\ x\in\Omega)$$

の場合はどうなるか．ただし，ν は既知のスカラー値関数である．

4.5 アニメーション

計算結果の可視化にはいくつか方法があるようだが，ここではアニメーションによる可視化について説明する．

4.5.1 C/C++ から gnuplot を呼び出す

C/C++ でプログラムを組んでいる場合に，ソースコード内に gnuplot を呼び出すコマンドを記述しておけば，いちいち実行とは別に gnuplot を開いて実行データのプロットを行う手間が省ける．

Linux での方法

Linux で作成中の場合，popen 関数[*7]で gnuplot を呼び出すことができる．

ソースコード 4.1　C++ のコード内から gnuplot を呼び出し $\sin x$ を描く

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  int main()
6  {
7          FILE *fp = popen("gnuplot", "w");
8          if (fp == NULL){
9                  return 1;
10         }
11         fputs("set␣mouse\n", fp);
12         fputs("plot␣sin(x)\n", fp);
13         fflush(fp);
14         cin.get();
15         pclose(fp);
16         return 0;
17 }
```

1-3 行目: iostream は読み込みや書き出しを行う関数が入っているライブラリ．C 言語での stdio.h に対応している．また，cstdio は FILE ポインタや fputs 関数などが入っているライブラリである．3 行目は名前空間の宣言であるが無視して構わない．

7 行目: FILE 型で fp というファイルを用意する．そこに，popen 関数を用いて gnuplot と書き込む ("w")．記号 *はポインタを表すが別の記事を参照せよ．

8-17 行目: もし fp が確保できないならそこでお終い．確保できるなら，fputs 関数で ( ) 内の文字列を fp に書き出す．一つ目はマウスで操作できるようにする宣言文で，二つ目は $\sin x$ をプロットする宣言文である．fflush 関数により FILE ポインタ fp のバッファに格納されているデータを吐き出させる．最後に pclose で fp を閉じれば良い．

Windows での場合

Windows (Visual C++) を使う場合は，マウスが自動的に有効になっているので宣言する必要はない．また，popen 関数と pclos 関数の代わりに_popen 関数と_pclose 関数を使う．

[*6] Fokker-Plank 方程式と呼ばれる．

[*7] 外部コマンドをプログラム内で使用できる一つの関数．終わりには pclose 関数で閉じないといけない．

ソースコード 4.2　Visual C++ のコード内から gnuplot を呼び出し $\sin x$ を描く

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  int main()
6  {
7          FILE *fp = _popen("pgnuplot.exe", "w");
8          if (fp == NULL){
9                  return 1;
10         }else{
11                 fputs("plot␣sin(x)\n", fp);
12                 fflush(fp);
13                 cin.get();
14                 _pclose(fp);
15                 return 0;
16         }
17 }
```

参考文献

[1] L. C. Evans, Partial Differential Equations, in the series Graduate studies in mathematics, Second Edition v. 19, American Math. Society, 2010.

第 5 章

偏微分方程式の有限要素法

第 6 章

その他の偏微分方程式

第 7 章

モンテカルロシミュレーション

第 8 章

セルオートマトン