熱方程式の差分法

目次

1　はじめに

本レポートでは拡散方程式の数値解法である差分法について述べる。なお、表および実験で用いたプログラムは付録に記す。

2　拡散方程式

本レポートでは、1 次元空間内での拡散現象を考える。拡散する微粒子の濃度を $C(x,t)$, 微粒子の流れを $J(x,t)$ とすれば、粒子数の保存から

$$\frac{\partial}{\partial t}C(x,t) = -\frac{\partial}{\partial x}J(x,t) \tag{1}$$

流れ $J(x,t)$ が濃度勾配に比例すると仮定すれば

$$J(x,t) = -D\frac{\partial}{\partial x}C(x,t) \tag{2}$$

式 (1) に代入して、$D = 1$ とすれば拡散方程式

$$\frac{\partial}{\partial t}u(x,t) = \frac{\partial^2}{\partial x^2}u(x,t) \tag{3}$$

$$\tag{4}$$

を得る。

2.1　境界条件

拡散方程式を解くためには、初期条件に加えて境界 $x = 0, L$ における境界条件を考える必要がある。本レポートでは、Dirichlet 境界条件および Neumann 境界条件を用いる。各境界条件の詳細を以下に示す。

Dirichlet 境界条件

$$u(0,t) = u_L$$
$$u(L,t) = u_R$$

Neumann 境界条件

$$\frac{\partial u}{\partial x}(0,t) = J_{\mathrm{L}}$$
$$\frac{\partial u}{\partial x}(L,t) = J_{\mathrm{R}}$$

3　問題 1

拡散方程式 (3) を、初期条件

$$u_0(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(x-5)^2} \tag{5}$$

のもとで数値的に解く。時間刻み幅 Δt、空間刻み幅 Δx とする。また、$L = N\Delta x$,$L = 10$ とする。このとき、離散化された時空間上での時刻 n の平均濃度を

$$u_j^n := \frac{1}{\Delta x}\int_{(j-1)\Delta x}^{j\Delta x} u(y, n\Delta t)dy \tag{6}$$

で表す。
計算手法には以下の 4 つを用いる。

- オイラー陽解法、Dirichlet 境界条件
- オイラー陽解法、Neumann 境界条件
- クランク・ニコルソン法、Dirichlet 境界条件
- クランク・ニコルソン法、Neumann 境界条件

3.1 各アルゴリズムの詳細

オイラー陽解法およびクランク・ニコルソン法の詳細について述べる。以下では $c := \frac{\Delta t}{\Delta x^2}$ とする。

3.1.1 オイラー陽解法

オイラー陽解法は、最も単純な差分方程式による数値解法である。以下で差分方程式を導出する。

3.1.2 差分方程式の導出

式 (1) を $[n\Delta t, (n+1)\Delta t]$ で積分すると

$$u(x, (n+1)\Delta t) - u(x, n\Delta t) = -\int_{n\Delta t}^{(n+1)\Delta t} \frac{\partial}{\partial x} J(x, s)ds \tag{7}$$

この右辺の積分を時刻 $n\Delta t$ での値を用いて近似すると

$$u(x, (n+1)\Delta t) - u(x, n\Delta t) \simeq -\frac{1}{\Delta x}\frac{\partial}{\partial x} J(x, n\Delta t) \tag{8}$$

よって離散化した時空間上では

$$u_j^{n+1} - u_j^n \tag{9}$$

$$\simeq \frac{1}{\Delta x}\int_{(j-1)\Delta x}^{j\Delta x} \Delta t\frac{\partial}{\partial y} J(y, n\Delta t)dy \tag{10}$$

$$= -\frac{\Delta t}{\Delta x}[J(j\Delta x, n\Delta t) - J((j-1)\Delta x, n\Delta t)] \tag{11}$$

ここで式 (2) から

$$J(j\Delta x, n\Delta t) = -\frac{\partial u}{\partial x}(j\Delta x, n\Delta t) \tag{12}$$

$$\simeq -\frac{u_{j+1}^n - u_j^n}{\Delta x} \tag{13}$$

なので、結局

$$u_j^{n+1} - u_j^n \simeq c(u_{j-1}^n - 2u_j^n + u_{j+1}^n) \tag{14}$$

を得る。この差分方程式を利用するのがオイラー陽解法である。

3.2 クランク・ニコルソン法

クランク・ニコルソン法はオイラー陽解法よりも精度の高いアルゴリズムである。陰解法であり、差分方程式を連立方程式として解く必要がある。

3.2.1 差分方程式の導出

オイラー陽解法では式 (7) の右辺を時刻 $n\Delta t$ の値で近似したが、クランク・ニコルソン法ではこれを台形公式で近似する。すなわち、式 (7) は

$$u(x,(n+1)\Delta t) - u(x,n\Delta t) \simeq -\frac{\Delta t}{2}[\frac{\partial}{\partial x}J(x,(n+1)\Delta t) + \frac{\partial}{\partial x}J(x,n\Delta t)] \tag{15}$$

となり、オイラー陽解法と同様の議論により

$$u_j^{n+1} - u_j^n \simeq \frac{1}{2}c(u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}) + \frac{1}{2}c(u_{j-1}^n - 2u_j^n + u_{j+1}^n) \tag{16}$$

を得る。式 (14) と異なり、式 (16) では右辺にも時刻 $n+1$ の、すなわち未知の値が含まれている。このような数値解法を陰解法と呼ぶ。

3.2.2 連立方程式の解法 (LU 分解)

式 (16) を変形すると

$$\begin{aligned} &-\frac{c}{2}u_{j-1}^{n+1} + (1+c)u_j^{n+1} - \frac{c}{2}u_{j+1}^{n+1} \\ =&(1-c)u_j^n + \frac{c}{2}u_{j-1}^n + \frac{c}{2}u_{j+1}^n \end{aligned} \tag{17}$$

となる。ただし、境界条件は

Dirichret **境界条件** ： $u_0^n = u_{\mathrm{L}}, u_{N+1}^n = u_{\mathrm{R}}$ より

$$\begin{aligned} &(1+c)u_1^{n+1} - \frac{c}{2}u_2^{n+1} \\ =&(1-c)u_1^n + cu_{\mathrm{L}} + \frac{c}{2}u_2^n \\ &-\frac{c}{2}u_{N-1}^{n+1} + (1+c)u_N^{n+1} \\ =&(1-c)u_N^n + cu_{\mathrm{R}} + \frac{c}{2}u_{N-1}^n \end{aligned}$$

Neumann **境界条件** ：$u_0^n = u_1^n - J_{\mathrm{L}}\Delta x, u_{N+1}^n = u_N^n + J_{\mathrm{R}}\Delta x$ より

$$\begin{aligned} &\left(1+\frac{c}{2}\right)u_1^{n+1} - \frac{c}{2}u_2^{n+1} \\ =&\left(1-\frac{c}{2}\right)u_1^n - cJ_{\mathrm{L}}\Delta x + \frac{c}{2}u_2^n \\ &-\frac{c}{2}u_{N-1}^{n+1} + \left(1+\frac{c}{2}\right)u_N^{n+1} \\ =&\left(1-\frac{c}{2}\right)u_N^n + cJ_{\mathrm{R}}\Delta x + \frac{c}{2}u_{N-1}^n \end{aligned}$$

となる。
連立方程式 (17) を解く際には、まず LU 分解を用いる。
式 (17) と境界条件を合わせた連立方程式は $\boldsymbol{x} \equiv (u_1^{n+1}, \ldots, u_N^{n+1})^\top$ と既知な量を成分とするベクトル $\boldsymbol{z}$ およびある行列 A を用いて

$$A\boldsymbol{x} = \boldsymbol{z} \tag{18}$$

と表される。A は Dirichlet 境界条件では

$$A = \begin{pmatrix} 1+c & -\frac{c}{2} & 0 & \cdots & 0 \\ -\frac{c}{2} & 1+c & -\frac{c}{2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\frac{c}{2} & 1+c & -\frac{c}{2} \\ 0 & \cdots & 0 & -\frac{c}{2} & 1+c \end{pmatrix}$$

Neumann 境界条件では

$$A = \begin{pmatrix} 1+\frac{c}{2} & -\frac{c}{2} & 0 & \cdots & 0 \\ -\frac{c}{2} & 1+c & -\frac{c}{2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\frac{c}{2} & 1+c & -\frac{c}{2} \\ 0 & \cdots & 0 & -\frac{c}{2} & 1+\frac{c}{2} \end{pmatrix}$$

となる。また、$\boldsymbol{z}$ は Dirichlet 境界条件では

$$\begin{aligned} z_1 =& (1-c)u_1^n + cu_{\mathrm{L}} + \frac{c}{2}u_2^n \\ z_j =& (1-c)u_j^n + \frac{c}{2}u_{j-1}^n + \frac{c}{2}u_{j+1}^n \quad (2 \le j \le N-1) \\ z_N =& (1-c)u_N^n + cu_{\mathrm{R}} + \frac{c}{2}u_{N-1}^n \end{aligned}$$

Neumann 境界条件では

$$\begin{aligned} z_1 =& \left(1-\frac{c}{2}\right)u_1^n - cJ_{\mathrm{L}}\Delta x + \frac{c}{2}u_2^n \\ z_j =& (1-c)u_j^n + \frac{c}{2}u_{j-1}^n + \frac{c}{2}u_{j+1}^n \quad (2 \le j \le N-1) \\ z_N =& \left(1-\frac{c}{2}\right)u_N^n + cJ_{\mathrm{R}}\Delta x + \frac{c}{2}u_{N-1}^n \end{aligned}$$

である。

行列 A について LU 分解を行う。A を

$$A = \begin{pmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ c_1 & a_2 & b_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & c_{N-2} & a_{N-1} & b_{N-1} \\ 0 & \cdots & 0 & c_{N-1} & a_N \end{pmatrix}$$

と書き、L, U を

$$L = \begin{pmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ c_1 & \alpha_2 & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & c_{N-2} & \alpha_{N-1} & 0 \\ 0 & \cdots & 0 & c_{N-1} & \alpha_N \end{pmatrix}, \quad U = \begin{pmatrix} 1 & \beta_1 & 0 & \cdots & 0 \\ 0 & 1 & \beta_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & 1 & \beta_{N-1} \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

とおくと、

$$LU = \begin{pmatrix} \alpha_1 & \alpha_1\beta_1 & 0 & \cdots & 0 \\ c_1 & \alpha_2 + c_1\beta_1 & \alpha_2\beta_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & c_{N-2} & \alpha_{N-1} + c_{N-2}\beta_{N-2} & \alpha_{N-1}\beta_{N-1} \\ 0 & \cdots & 0 & c_{N-1} & \alpha_N + c_{N-1}\beta_{N-1} \end{pmatrix}$$

となり、これが A と一致するので

$$\begin{aligned} \alpha_1 &= a_1 \\ \beta_1 &= \frac{b_1}{\alpha_1} \\ \alpha_2 &= a_2 - c_1\beta_1 \\ \beta_2 &= \frac{b_2}{\alpha_2} \\ &\vdots \\ \alpha_{N-1} &= a_{N-1} - c_{N-2}\beta_{N-2} \\ \beta_{N-1} &= \frac{b_{N-1}}{\alpha_{N-1}} \\ \alpha_N &= a_N - c_{N-1}\beta_{N-1} \end{aligned}$$

のように、L, U が順次求まる。

次に、$U\boldsymbol{x} \equiv \boldsymbol{y}$ とおくと, 方程式 (18) は

$$\begin{pmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ c_1 & \alpha_2 & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & c_{N-2} & \alpha_{N-1} & 0 \\ 0 & \cdots & 0 & c_{N-1} & \alpha_N \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{N-1} \\ z_N \end{pmatrix}$$

となるから、$\boldsymbol{y}$ は順次求まり、

$$\begin{aligned} y_1 &= \frac{z_1}{\alpha_1} \\ y_2 &= \frac{z_2 - c_1 y_1}{\alpha_2} \\ &\vdots \end{aligned}$$

となる。その後、$U\boldsymbol{x} = \boldsymbol{y}$ から、$\boldsymbol{x}$ は降順に

$$
\begin{aligned}
&x_N = y_N \\
&x_{N-1} = y_{N-1} - \beta_{N-1} x_N \\
&\vdots \\
&x_2 = y_2 - \beta_2 x_3 \\
&x_1 = y_1 - \beta_1 x_2
\end{aligned}
$$

となる。このようにして連立方程式 (17) を解くことができる。

3.3 計算結果

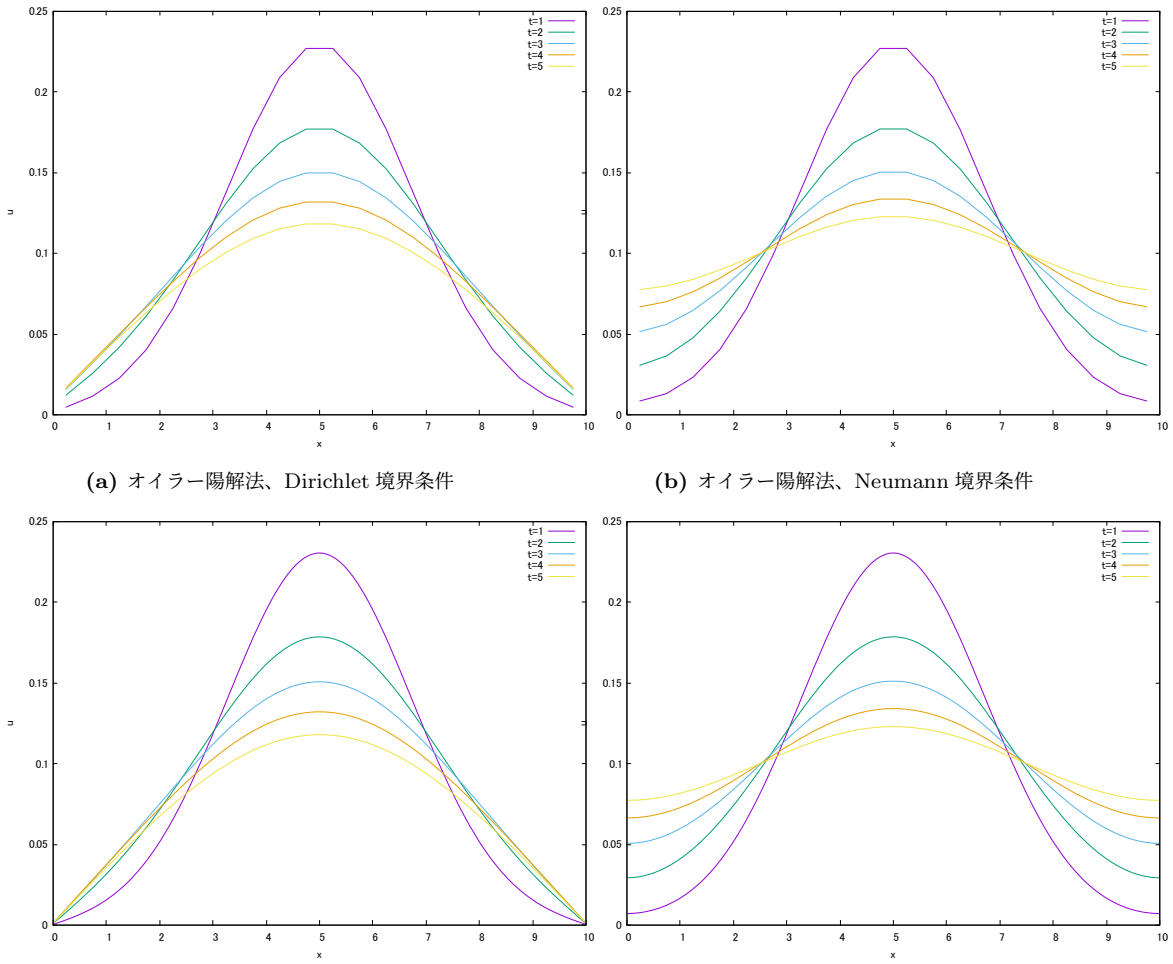
計算結果を表 1,2,3,4 および図 1 にまとめる。なお、空間の離散化にあたって、初期条件は

$$u_j^0 = \frac{1}{2}[u_0((j-1)\Delta x) + u_0(j\Delta x)] \tag{19}$$

各 j に対応する座標は

$$x = (j - \frac{1}{2})\Delta x \tag{20}$$

としている。



(a) オイラー陽解法、Dirichlet 境界条件

(b) オイラー陽解法、Neumann 境界条件

(c) クランク・ニコルソン法、Dirichlet 境界条件

(d) クランク・ニコルソン法、Neumann 境界条件

図 1: アルゴリズム、境界条件ごとの計算結果

4 問題 2

拡散方程式に非線形項 $f(u) = u(1-u)$ を加えた偏微分方程式

$$\frac{\partial u}{\partial t} = u(1-u) + \frac{\partial^2 u}{\partial x^2} \tag{21}$$

を考える。この方程式は Fisher 方程式と呼ばれ、遺伝学における優性遺伝子の空間伝播を表現するために提案された。初期条件

$$u_0(x) = \frac{1}{(1+e^{bx-5})^2} \tag{22}$$

境界条件 ($L = 200$)

$$u(0,t) = 1 \tag{23}$$

$$u(L,t) = 0 \tag{24}$$

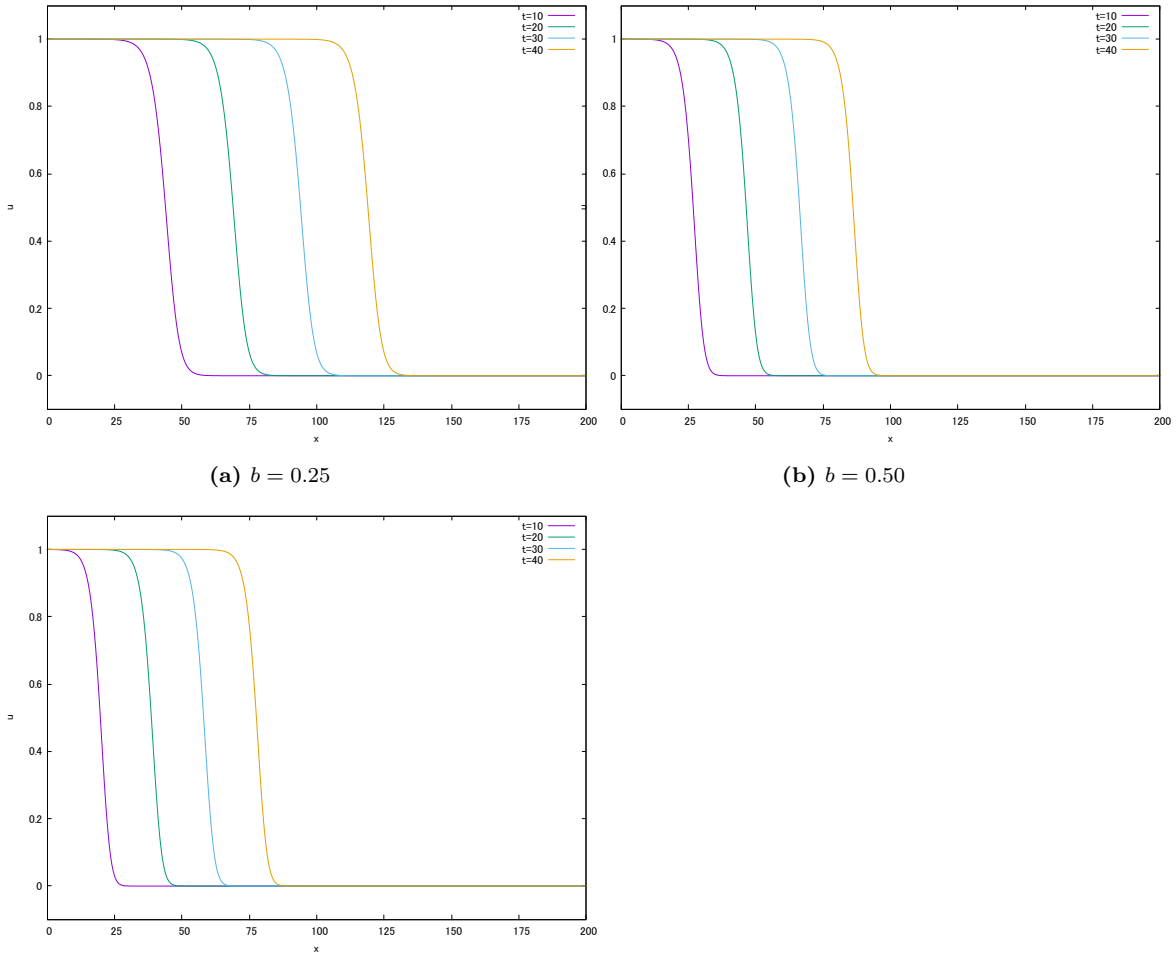
のもとで、$b = 0.25, 0.5, 1.0$ の場合に Fisher 方程式をオイラー陽解法を用いて解く。差分方程式は

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = f(u_j^n) + \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} \tag{25}$$

となる。

4.1 計算結果

計算結果を図 2 および表 5,6,7 にまとめる。



(a) $b = 0.25$ **(b)** $b = 0.50$

(c) $b = 1.0$

図 2: Fisher 方程式の解

4.2 結果の考察

図 2 を見ると、ある距離までは $u = 1$ 付近でほぼ一定で、そこから急激に u の値が減少して 0 に漸近している。同じ b の値の中では、時刻が進むほど $u \simeq 1$ となる範囲が広がっており、広がる幅はほぼ等間隔であ

る。これは、まさに空間伝播を表していると考えられる。時間が経つごとに一定の速さで優性遺伝子が伝播することがわかる。また、同じ t の値を比べると、b が小さいほど $u \simeq 1$ である範囲が広い。加えて、伝播は b が小さいほど速くなっている。このことについて、初期条件 (22) を見れば、b が大きいほど u の 0 への漸近は早くなる。つまり、優性遺伝子を持つ者が多いほど伝播も速くなるということである。

5 問題 3

量子力学において、1 次元調和振動子のダイナミクスは、以下の Schrödinger 方程式

$$i\hbar\frac{\partial\psi}{\partial t}(x,t) = \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{k}{2}x^2\right)\psi(x,t) \tag{26}$$

に従う。波動関数 ψ を $\psi(x,t) = \psi_R(x,t) + i\psi_I(x,t)$ $(\psi_R, \psi_I \in \mathbb{R})$ のように実部と虚部に分けると、ψ は 2 つの拡散方程式

$$\begin{aligned}\hbar\frac{\partial\psi_R}{\partial t}(x,t) &= \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{k}{2}x^2\right)\psi_I(x,t)\\ -\hbar\frac{\partial\psi_I}{\partial t}(x,t) &= \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{k}{2}x^2\right)\psi_R(x,t)\end{aligned} \tag{27}$$

に従う。また、この際、粒子の位置の確率密度は $P(x,t) = |\psi(x,t|^2 = \psi_R^2 + \psi_I^2$ で与えられる。いま、$\hbar = 1, m = 1, k = 1$ として、初期条件

$$\psi(x,0) = \frac{\sqrt{2}}{\pi^{\frac{1}{4}}}e^{-2(x-5)^2} \tag{28}$$

空間領域 $[-L/2, L/2]$、境界条件は周期境界条件

$$\psi(\frac{L}{2},t) = \psi(-\frac{L}{2},t) \tag{29}$$

のもとで数値的に解く。

5.1 数値解法 (Visscher のスキーム)

計算には Visscher のスキームを用いる。Visscher のスキームは、式 (**??**) のような実部と虚部が連動する方程式において、実部と虚部の時間発展を半ステップだけずらして解く陽解法である。離散化された波動関数の実部、虚部をそれぞれ $\left\{R_j^n\right\}, \left\{I_j^n\right\}$ と書くと、$1 \le j \le N$ $(L = N\Delta x)$ に対して時間発展は

$$\begin{aligned}R_j^{n+1} &= R_j^n + \Delta t\left(-\frac{1}{2}\frac{I_{j-1}^n - 2I_j^n + I_{j+1}^n}{\Delta x^2} + \frac{1}{2}x_j^2 I_j^n\right)\\ I_j^{n+1} &= I_j^n - \Delta t\left(-\frac{1}{2}\frac{R_{j-1}^{n+1} - 2R_j^{n+1} + R_{j+1}^{n+1}}{\Delta x^2} + \frac{1}{2}x_j^2 R_j^{n+1}\right)\end{aligned}$$

と記述される。ただし、

$$x_j \equiv \left(j - \frac{1}{2}\right)\Delta x - \frac{L}{2}$$

であり、境界条件は全ての n で

$$\begin{aligned} R_0^n &= R_N^n \\ R_{N+1}^n &= R_1^n \\ I_0^n &= I_N^n \\ I_{N+1}^n &= I_1^n \end{aligned}$$

と表される。また、初期条件は $1 \leq j \leq N$ で

$$\begin{aligned} R_j^0 =& \frac{1}{2}\left[\psi\left((j-1)\Delta x - \frac{L}{2}, 0\right) + \psi\left(j\Delta x - \frac{L}{2}, 0\right)\right] \\ I_j^0 =& -\Delta t\left(-\frac{1}{2}\frac{R_{j-1}^0 - 2R_j^0 + R_{j+1}^0}{\Delta x^2} + \frac{1}{2}x_j^2 R_j^0\right) \end{aligned}$$

と表される。本問では、$N = 400,\ \Delta x = 0.05\ (L = 20),\ \Delta t = 0.001$ とする。

5.2 計算結果

計算により得られた確率密度 $\left(R_j^n\right)^2 + I_j^n I_j^{n-1}$ の値を図 3 と表 8 に示す。

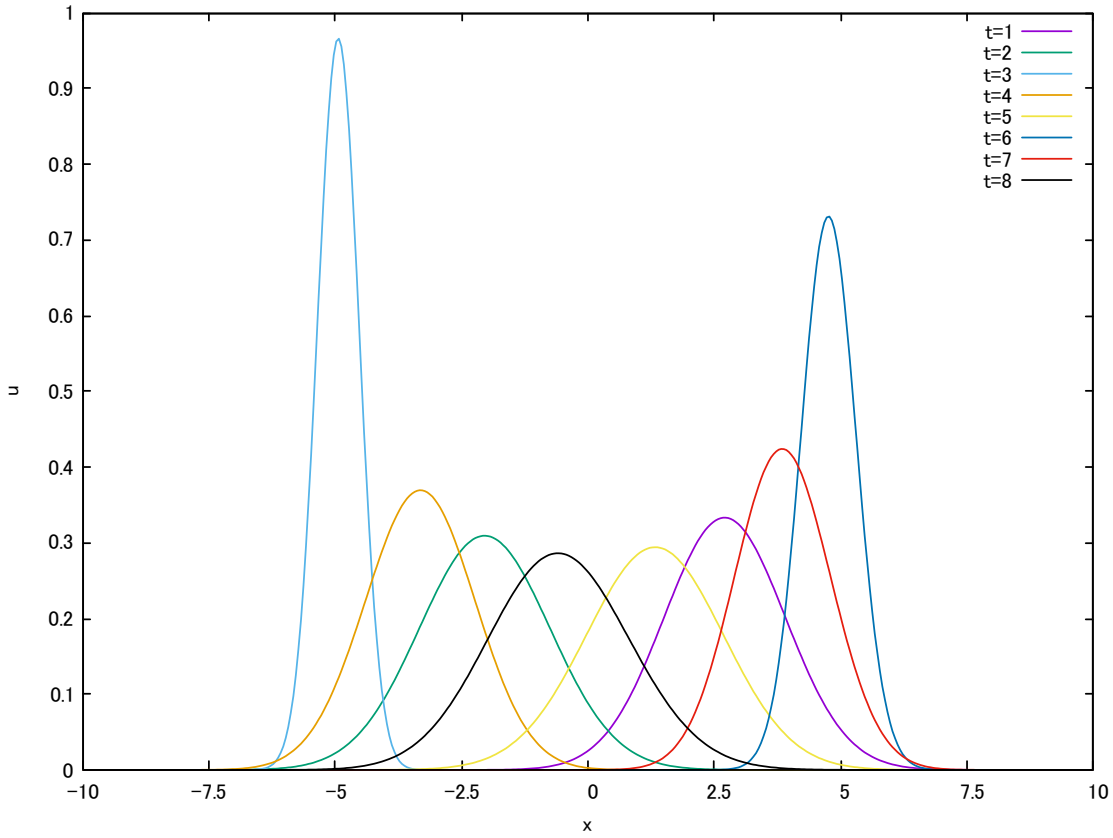


図 3: 確率密度

6 付録

表をまとめて示す。

表 1: オイラー陽解法、Dirichret 境界条件

x	$u(x,1)$	$u(x,2)$	$u(x,3)$	$u(x,4)$	$u(x,5)$
0.25	0.00473066	0.0121855	0.0158097	0.0167033	0.0163048
0.75	0.0115542	0.0257629	0.0321057	0.0334227	0.0324223
1.25	0.0226647	0.04185	0.0492076	0.0500984	0.0481345
1.75	0.040201	0.0610265	0.0671306	0.0665337	0.0631695
2.25	0.0656733	0.0831094	0.085503	0.0823648	0.0771919
2.75	0.0990364	0.107029	0.103559	0.097067	0.0898099
3.25	0.137785	0.13087	0.120218	0.109999	0.1006
3.75	0.17668	0.152135	0.13423	0.120478	0.109143
4.25	0.208638	0.168194	0.144387	0.127875	0.115071
4.75	0.226757	0.17685	0.14973	0.131705	0.118108
5.25	0.226757	0.17685	0.14973	0.131705	0.118108
5.75	0.208638	0.168194	0.144387	0.127875	0.115071
6.25	0.17668	0.152135	0.13423	0.120478	0.109143
6.75	0.137785	0.13087	0.120218	0.109999	0.1006
7.25	0.0990364	0.107029	0.103559	0.097067	0.0898099
7.75	0.0656733	0.0831094	0.085503	0.0823648	0.0771919
8.25	0.040201	0.0610265	0.0671306	0.0665337	0.0631695
8.75	0.0226647	0.0418	,0.0492076	0.0500984	0.0481345
9.25	0.0115542	0.0257629	0.0321057	0.033422	,0.0324223
9.75	0.00473066	0.0121855	0.0158097	0.0167033	0.0163048

表 2: オイラー陽解法、Neumann 境界条件

x	$u(x,1)$	$u(x,2)$	$u(x,3)$	$u(x,4)$	$u(x,5)$
0.25	0.0085096	0.0306674	0.0514459	0.0668294	0.0775135
0.75	0.0131201	0.0364048	0.0559694	0.070027	0.079704
1.25	0.0232712	0.0477084	0.0646608	0.0761281	0.0838747
1.75	0.0404209	0.0641117	0.0768108	0.0845659	0.0896239
2.25	0.0657481	0.0846649	0.0913721	0.094545	0.0963953
2.75	0.0990603	0.10778	0.107008	0.105107	0.10353
3.25	0.137792	0.131219	0.122188	0.115219	0.11033
3.75	0.176682	0.152291	0.135339	0.123872	0.116125
4.25	0.208638	0.168264	0.145032	0.130188	0.120342
4.75	0.226757	0.176888	0.150174	0.133518	0.122561
5.25	0.226757	0.176888	0.150174	0.133518	0.122561
5.75	0.208638	0.168264	0.145032	0.130188	0.120342
6.25	0.176682	0.152291	0.135339	0.123872	0.116125
6.75	0.137792	0.131219	0.122188	0.115219	0.11033
7.25	0.0990603	0.10778	0.107008	0.105107	0.10353
7.75	0.0657481	0.0846649	0.0913721	0.094545	0.0963953
8.25	0.0404209	0.0641117	0.0768108	0.0845659	0.0896239
8.75	0.0232712	0.0477084	0.0646608	0.0761281	0.0838747
9.25	0.0131201	0.0364048	0.0559694	0.070027	0.079704
9.75	0.0085096	0.0306674	0.0514459	0.0668294	0.0775135

表 3: クランク・ニコルソン法、Dirichlet 境界条件

x	$u(x,1)$	$u(x,2)$	$u(x,3)$	$u(x,4)$	$u(x,5)$
0.025	0.000574429	0.00143559	0.00178272	0.00182558	0.00174313
0.075	0.00115145	0.00287264	0.00356582	0.00365107	0.00348597
0.125	0.00173364	0.00431263	0.00534969	0.00547636	0.00522825
0.175	0.00232361	0.00575702	0.00713472	0.00730134	0.00696969
0.225	0.00292393	0.00720726	0.00892127	0.00912593	0.00871
0.275	0.00353721	0.00866477	0.0107097	0.01095	0.0104489
0.325	0.00416605	0.010131	0.0125004	0.0127735	0.0121861
0.375	0.00481304	0.0116073	0.0142937	0.0145962	0.0139213
0.425	0.00548078	0.013095	0.01609	0.016418	0.0156543
0.475	0.00617188	0.0145956	0.0178895	0.0182389	0.0173846
⋮	⋮	⋮	⋮	⋮	⋮
4.775	0.228385	0.177493	0.150007	0.131605	0.117522
4.825	0.229148	0.177849	0.150226	0.131764	0.117649
4.875	0.229721	0.178116	0.150391	0.131883	0.117744
4.925	0.230105	0.178295	0.150501	0.131962	0.117808
4.975	0.230296	0.178384	0.150556	0.132002	0.11784
5.025	0.230296	0.178384	0.150556	0.132002	0.11784
5.075	0.230105	0.178295	0.150501	0.131962	0.117808
5.125	0.229721	0.178116	0.150391	0.131883	0.117744
5.175	0.229148	0.177849	0.150226	0.131764	0.117649
5.225	0.228385	0.177493	0.150007	0.131605	0.117522
⋮	⋮	⋮	⋮	⋮	⋮
9.525	0.00617188	0.0145956	0.0178895	0.0182389	0.0173846
9.575	0.00548078	0.013095	0.01609	0.016418	0.0156543
9.625	0.00481304	0.0116073	0.0142937	0.0145962	0.0139213
9.675	0.00416605	0.010131	0.0125004	0.0127735	0.0121861
9.725	0.00353721	0.00866477	0.0107097	0.01095	0.0104489
9.775	0.00292393	0.00720726	0.00892127	0.00912593	0.00871
9.825	0.00232361	0.00575702	0.00713472	0.00730134	0.00696969
9.875	0.00173364	0.00431263	0.00534969	0.00547636	0.00522825
9.925	0.00115145	0.00287264	0.00356582	0.00365107	0.00348597
9.975	0.000574429	0.00143559	0.00178272	0.00182558	0.00174313

表 4: クランク・ニコルソン法、Neumann 境界条件

x	$u(x,1)$	$u(x,2)$	$u(x,3)$	$u(x,4)$	$u(x,5)$
0.025	0.00715622	0.0293034	0.0505747	0.0663232	0.0772302
0.075	0.00719989	0.0293619	0.0506211	0.0663559	0.0772526
0.125	0.00728734	0.029479	0.0507139	0.0664214	0.0772973
0.175	0.00741879	0.0296547	0.050853	0.0665195	0.0773643
0.225	0.00759457	0.0298888	0.0510384	0.0666502	0.0774536
0.275	0.00781512	0.0301813	0.0512698	0.0668134	0.077565
0.325	0.008081	0.0305322	0.0515471	0.0670088	0.0776985
0.375	0.00839284	0.0309413	0.05187	0.0672363	0.0778539
0.425	0.00875142	0.0314086	0.0522384	0.0674958	0.078031
0.475	0.00915757	0.0319339	0.0526518	0.0677869	0.0782298
⋮	⋮	⋮	⋮	⋮	⋮
4.775	0.228385	0.177527	0.150491	0.133665	0.122611
4.825	0.229148	0.177881	0.150701	0.133801	0.122702
4.875	0.229721	0.178148	0.150859	0.133903	0.12277
4.925	0.230105	0.178326	0.150964	0.133971	0.122815
4.975	0.230296	0.178415	0.151017	0.134005	0.122837
5.025	0.230296	0.178415	0.151017	0.134005	0.122837
5.075	0.230105	0.178326	0.150964	0.133971	0.122815
5.125	0.229721	0.178148	0.150859	0.133903	0.12277
5.175	0.229148	0.177881	0.150701	0.133801	0.122702
5.225	0.228385	0.177527	0.150491	0.133665	0.122611
⋮	⋮	⋮	⋮	⋮	⋮
9.525	0.00915757	0.0319339	0.0526518	0.0677869	0.0782298
9.575	0.00875142	0.0314086	0.0522384	0.0674958	0.078031
9.625	0.00839284	0.0309413	0.05187	0.0672363	0.0778539
9.675	0.008081	0.0305322	0.0515471	0.0670088	0.0776985
9.725	0.00781512	0.0301813	0.0512698	0.0668134	0.077565
9.775	0.00759457	0.0298888	0.0510384	0.0666502	0.0774536
9.825	0.00741879	0.0296547	0.050853	0.0665195	0.0773643
9.875	0.00728734	0.029479	0.0507139	0.0664214	0.0772973
9.925	0.00719989	0.0293619	0.0506211	0.0663559	0.0772526
9.975	0.00715622	0.0293034	0.0505747	0.0663232	0.0772302

表 5: Fisher 方程式、$b = 0.25$

x	$u(x, 10)$	$u(x, 20)$	$u(x, 30)$	$u(x, 40)$
0.025	1.00000	1.00000	1.00000	1.00000
0.075	1.00000	1.00000	1.00000	1.00000
0.125	1.00000	1.00000	1.00000	1.00000
0.175	1.00000	1.00000	1.00000	1.00000
0.225	1.00000	1.00000	1.00000	1.00000
0.275	1.00000	1.00000	1.00000	1.00000
0.325	1.00000	1.00000	1.00000	1.00000
0.375	1.00000	1.00000	1.00000	1.00000
0.425	1.00000	1.00000	1.00000	1.00000
0.475	1.00000	1.00000	1.00000	1.00000
⋮	⋮	⋮	⋮	⋮
99.775	1.23482E-12	3.28813E-07	7.55618E-02	9.98987E-01
99.825	1.20433E-12	3.20695E-07	7.39303E-02	9.98969E-01
99.875	1.17459E-12	3.12777E-07	7.23296E-02	9.98951E-01
99.925	1.14559E-12	3.05054E-07	7.07593E-02	9.98932E-01
99.975	1.11731E-12	2.97522E-07	6.9219E-02	9.98914E-01
100.025	1.08972E-12	2.90177E-07	6.77083E-02	9.98894E-01
100.075	1.06282E-12	2.83012E-07	6.62268E-02	9.98875E-01
100.125	1.03658E-12	2.76024E-07	6.4774E-02	9.98855E-01
100.175	1.01098E-12	2.69209E-07	6.33496E-02	9.98835E-01
100.225	9.86022E-13	2.62563E-07	6.19531E-02	9.98814E-01
⋮	⋮	⋮	⋮	⋮
199.525	1.06395E-34	2.82783E-29	7.52961E-24	2.00502E-18
199.575	9.55675E-35	2.54003E-29	6.7633E-24	1.80096E-18
199.625	8.47994E-35	2.25382E-29	6.00122E-24	1.59803E-18
199.675	7.40843E-35	1.96902E-29	5.24289E-24	1.3961E-18
199.725	6.34153E-35	1.68546E-29	4.48783E-24	1.19504E-18
199.775	5.27858E-35	1.40294E-29	3.73558E-24	9.94729E-19
199.825	4.21892E-35	1.1213E-29	2.98567E-24	7.95038E-19
199.875	3.16189E-35	8.40363E-30	2.23762E-24	5.95844E-19
199.925	2.10683E-35	5.59951E-30	1.49097E-24	3.97023E-19
199.975	1.05309E-35	2.79888E-30	7.45252E-25	1.98449E-19

表 6: Fisher 方程式、$b = 0.50$

x	$u(x, 10)$	$u(x, 20)$	$u(x, 30)$	$u(x, 40)$
0.025	1	1	1	1
0.075	9.99999E-01	1.00000	1.00000	1.00000
0.125	9.99999E-01	1.00000	1.00000	1.00000
0.175	9.99999E-01	1.00000	1.00000	1.00000
0.225	9.99998E-01	1.00000	1.00000	1.00000
0.275	9.99998E-01	1.00000	1.00000	1.00000
0.325	9.99998E-01	1.00000	1.00000	1.00000
0.375	9.99998E-01	1.00000	1.00000	1.00000
0.425	9.99997E-01	1.00000	1.00000	1.00000
0.475	9.99997E-01	1.00000	1.00000	1.00000
⋮	⋮	⋮	⋮	⋮
99.775	4.65321E-31	2.21754E-22	1.0562E-13	3.56927E-05
99.825	4.42627E-31	2.10939E-22	1.0047E-13	3.40628E-05
99.875	4.21039E-31	2.00651E-22	9.55708E-14	3.25064E-05
99.925	4.00505E-31	1.90865E-22	9.09108E-14	3.10203E-05
99.975	3.80972E-31	1.81556E-22	8.64779E-14	2.96013E-05
100.025	3.62392E-31	1.72702E-22	8.22612E-14	2.82465E-05
100.075	3.44718E-31	1.64279E-22	7.82501E-14	2.69529E-05
100.125	3.27906E-31	1.56267E-22	7.44345E-14	2.57179E-05
100.175	3.11914E-31	1.48646E-22	7.0805E-14	2.45389E-05
100.225	2.96701E-31	1.41396E-22	6.73525E-14	2.34133E-05
⋮	⋮	⋮	⋮	⋮
199.525	1.4050E-74	6.6957E-66	3.19091E-57	1.52066E-48
199.575	1.25468E-74	5.97931E-66	2.8495E-57	1.35796E-48
199.625	1.10749E-74	5.27787E-66	2.51523E-57	1.19866E-48
199.675	9.63074E-75	4.58963E-66	2.18724E-57	1.04235E-48
199.725	8.21063E-75	3.91287E-66	1.86472E-57	8.88652E-49
199.775	6.81106E-75	3.24589E-66	1.54686E-57	7.37174E-49
199.825	5.42853E-75	2.58702E-66	1.23287E-57	5.87539E-49
199.875	4.05956E-75	1.93463E-66	9.21968E-58	4.39374E-49
199.925	2.70075E-75	1.28707E-66	6.13367E-58	2.92307E-49
199.975	1.34869E-75	6.42732E-67	3.06301E-58	1.45971E-49

表 7: Fisher 方程式、$b = 1.0$

x	$u(x, 10)$	$u(x, 20)$	$u(x, 30)$	$u(x, 40)$
0.025	9.99996E-01	1.00000	1.00000	1.00000
0.075	9.99992E-01	1.00000	1.00000	1.00000
0.125	9.99988E-01	1.00000	1.00000	1.00000
0.175	9.99984E-01	1.00000	1.00000	1.00000
0.225	9.9998E-01	1.00000	1.00000	1.00000
0.275	9.99976E-01	1.00000	1.00000	1.00000
0.325	9.99972E-01	1.00000	1.00000	1.00000
0.375	9.99968E-01	1.00000	1.00000	1.00000
0.425	9.99964E-01	1.00000	1.00000	1.00000
0.475	9.9996E-01	1.00000	1.00000	1.00000
⋮	⋮	⋮	⋮	⋮
99.775	2.04937E-61	9.54931E-40	2.13122E-21	2.30302E-09
99.825	1.85435E-61	8.64314E-40	1.97314E-21	2.17425E-09
99.875	1.67788E-61	7.82292E-40	1.82672E-21	2.05261E-09
99.925	1.51821E-61	7.08051E-40	1.6911E-21	1.9377E-09
99.975	1.37374E-61	6.40852E-40	1.56549E-21	1.82916E-09
100.025	1.24301E-61	5.80028E-40	1.44915E-21	1.72664E-09
100.075	1.12472E-61	5.24974E-40	1.3414E-21	1.6298E-09
100.125	1.01769E-61	4.75144E-40	1.24162E-21	1.53835E-09
100.175	9.20843E-62	4.30041E-40	1.14921E-21	1.45197E-09
100.225	8.33213E-62	3.89218E-40	1.06364E-21	1.37039E-09
⋮	⋮	⋮	⋮	⋮
199.525	4.04315E-148	1.91314E-126	9.0526E-105	4.28313E-83
199.575	3.53162E-148	1.67109E-126	7.90728E-105	3.74124E-83
199.625	3.05543E-148	1.44577E-126	6.8411E-105	3.23679E-83
199.675	2.60982E-148	1.23492E-126	5.84338E-105	2.76474E-83
199.725	2.19033E-148	1.03642E-126	4.90415E-105	2.32035E-83
199.775	1.79277E-148	8.48304E-127	4.01401E-105	1.89919E-83
199.825	1.41315E-148	6.68674E-127	3.16403E-105	1.49703E-83
199.875	1.04767E-148	4.95736E-127	2.34573E-105	1.10986E-83
199.925	6.92674E-149	3.2776E-127	1.55089E-105	7.33793E-84
199.975	3.44612E-149	1.63064E-127	7.71586E-106	3.6507E-84

表 8: 問題 3 における $t = 1, 2, 3, 4$ に関する表

x	$t = 1$	$t = 2$	$t = 3$	$t = 4$
-9.975	2.31579E-09	1.19114E-09	6.43519E-12	3.92017E-10
-9.925	1.87259E-09	1.57656E-09	7.37102E-12	5.41102E-10
-9.875	1.53733E-09	2.08947E-09	8.89227E-12	8.26309E-10
-9.825	1.27877E-09	2.76211E-09	1.08097E-11	1.30231E-09
-9.775	1.07568E-09	3.63317E-09	1.29555E-11	2.04420E-09
-9.725	9.13458E-10	4.74891E-09	1.51829E-11	3.15303E-09
-9.675	7.81872E-10	6.16440E-09	1.73737E-11	4.76260E-09
-9.625	6.73661E-10	7.94505E-09	1.94492E-11	7.04748E-09
-9.575	5.83586E-10	1.01686E-08	2.13810E-11	1.02326E-08
-9.525	5.07810E-10	1.29277E-08	2.31970E-11	1.46048E-08
⋮	⋮	⋮	⋮	⋮
-0.225	1.59587E-02	1.14669E-01	2.82033E-11	5.24742E-03
-0.175	1.77066E-02	1.08504E-01	4.25511E-13	4.54475E-03
-0.125	1.96097E-02	1.02516E-01	1.86050E-11	3.92651E-03
-0.075	2.16772E-02	9.67123E-02	6.46900E-11	3.38392E-03
-0.025	2.39184E-02	9.11000E-02	9.44196E-11	2.90869E-03
0.025	2.63426E-02	8.56857E-02	7.95185E-11	2.49334E-03
0.075	2.89591E-02	8.04742E-02	3.44247E-11	2.13125E-03
0.125	3.17770E-02	7.54681E-02	2.18020E-12	1.81667E-03
0.175	3.48050E-02	7.06684E-02	1.31595E-11	1.54439E-03
0.225	3.80517E-02	6.60747E-02	5.62023E-11	1.30963E-03
⋮	⋮	⋮	⋮	⋮
9.525	2.88564E-08	1.02648E-10	5.86884E-11	1.16324E-09
9.575	2.24643E-08	1.29692E-10	5.12651E-11	1.10660E-09
9.625	1.74371E-08	1.63734E-10	4.24634E-11	1.01669E-09
9.675	1.34990E-08	2.06764E-10	3.34827E-11	9.02701E-10
9.725	1.04275E-08	2.61474E-10	2.52452E-11	7.75195E-10
9.775	8.04348E-09	3.31561E-10	1.83403E-11	6.45337E-10
9.825	6.20358E-09	4.22167E-10	1.30429E-11	5.24529E-10
9.875	4.79302E-09	5.40536E-10	9.37809E-12	4.24495E-10
9.925	3.72046E-09	6.97011E-10	7.20574E-12	3.57986E-10
9.975	2.91362E-09	9.06562E-10	6.30320E-12	3.40310E-10

実験で用いたプログラムを記す。

コード 1: 問題 1 のオイラー陽解法

```
#include<iostream>
#include<cmath>
#include<vector>
#include<fstream>

#define dx 0.5
#define dt 0.01
using namespace std;

void euler_dir(vector<double> x, int x_size, string FILE_NAME){ //ディリクレ境界条件
    ofstream outputfile(FILE_NAME);

    int step = 500; //初期化
    vector<double> y(x_size);
    y[0] = 0;
    y[x_size-1] = 0;
    for (int i=1; i<step+1; i++){ //計算
        for (int j=1; j<x_size-1; j++){
            y[j] = x[j] + dt*(x[j-1] - 2*x[j] + x[j+1])/(dx*dx);
        }
        for (int k=1; k<x_size-1; k++){
            x[k] = y[k];
        }
        if (i % 100 == 0){
            for (int l=1; l<x_size-1; l++){
                outputfile << 0.25 + dx*(l-1) << "␣" << y[l] << endl;
            }
            outputfile << endl;
            }
    }
}

void euler_neu(vector<double> x, int x_size, string FILE_NAME){ //ノイマン境界条件
    ofstream outputfile(FILE_NAME);

    int step = 500;
    vector<double> y(x_size);
    y[0] = x[0];
    y[x_size-1] = x[x_size-1];
    for (int i=1; i<step+1; i++){
        for (int j=1; j<x_size-1; j++){
            y[j] = x[j] + dt*(x[j-1] - 2*x[j] + x[j+1])/(dx*dx);
        }
        y[0] = y[1];
        y[x_size-1] = y[x_size-2];
        for (int k=0; k<x_size; k++){
            x[k] = y[k];
        }
        if (i % 100 == 0){
            for (int l=1; l<x_size-1; l++){
                outputfile << 0.25 + dx*(l-1) << "␣" << y[l] << endl;
            }
            outputfile << endl;
            }
```

```
    }
}

int main(void){
    vector<double> x(21);
    vector<double> x_init(22);
    for (int i=0; i<21; i++){
        x[i] = exp(-0.5*(i*dx - 5.0)*(i*dx - 5.0))/sqrt(2*M_PI);
    }
    for (int i=1; i<21; i++){ //初期値の計算
        x_init[i] = 0.5*(x[i-1] + x[i]);
    }

    x_init[0] = x_init[1];
    x_init[21] = x_init[20];
    euler_neu(x_init, 22, "euler_neu.txt");

    x_init[0] = 0;
    x_init[21] = 0;
    euler_dir(x_init, 22, "euler_dir.txt");
    return 0;
}
```

コード 2: 問題 1 のクランク・ニコルソン法

```
#include<iostream>
#include<cmath>
#include<vector>
#include<fstream>

#define dx 0.05
#define dt 0.01
using namespace std;

void cn_dir(vector<double> x, double xL, double xR, int x_size, string FILE_NAME){ //ディリクレ境界条件
    ofstream outputfile(FILE_NAME);

    int step = 500; //初期化
    double c = dt/(dx*dx);
    double a = 1 + c;
    double b = -c/2.0;
    vector<double> alpha(x_size);
    vector<double> beta(x_size-1);
    vector<double> z(x_size);
    vector<double> y(x_size);
    vector<double> h(x_size);

    for (int i=0; i<x_size; i++){ //作業用配列
        h[i] = x[i];
    }

    alpha[0] = a;
    beta[0] = b/alpha[0];
    for (int i=1; i<x_size-1; i++){
        alpha[i] = a - beta[i-1]*(-c/2.0);
        beta[i] = b/alpha[i];
```

```
32      }
33      alpha[x_size-1] = a - beta[x_size-2]*(-c/2.0);
34
35
36      for (int i=1; i<step+1; i++){ //連立方程式を解く
37          z[0] = (1 - c)*h[0] + c*xL + c*h[1]/2.0;
38          z[x_size-1] = (1 - c)*h[x_size-1] +c*xR + c*h[x_size-2]/2.0;
39          for (int j=1;j<x_size-1;j++){
40              z[j] = (1 - c)*h[j] + c*h[j-1]/2.0 + c*h[j+1]/2.0;
41          }
42
43          y[0] = z[0]/alpha[0];
44          for (int j=1;j<x_size;j++){
45              y[j] = (z[j] - y[j-1]*(-c/2.0))/alpha[j];
46          }
47
48          h[x_size-1] = y[x_size-1];
49          for (int j = x_size-2; j>=0; j--){
50              h[j] = y[j] - beta[j]*h[j+1];
51          }
52          if (i % 100 == 0){
53              for (int l=0; l<x_size; l++){
54                  outputfile << 0.025 + dx*l << "␣" << h[l] << endl;
55              }
56              outputfile << endl;
57          }
58      }
59  }
60
61  void cn_neu(vector<double> x, double jL, double jR, int x_size, string FILE_NAME){ //ノイマ
        ン境界条件
62      ofstream outputfile(FILE_NAME);
63
64      int step = 500;
65      double c = dt/(dx*dx);
66      double a = 1 + c;
67      double b = -c/2.0;
68      vector<double> alpha(x_size);
69      vector<double> beta(x_size-1);
70      vector<double> z(x_size);
71      vector<double> y(x_size);
72      vector<double> h(x_size);
73
74      for (int i=0; i<x_size; i++){
75          h[i] = x[i];
76      }
77
78      alpha[0] = 1 + c/2.0;
79      beta[0] = b/alpha[0];
80      for (int i=1; i<x_size-1; i++){
81          alpha[i] = a - beta[i-1]*(-c/2.0);
82          beta[i] = b/alpha[i];
83      }
84      alpha[x_size-1] = 1 + c/2.0 - beta[x_size-2]*(-c/2.0);
85
86
87      for (int i=1; i<step+1; i++){
```

```
88          z[0] = (1 - c/2.0)*h[0] + c*jL*dx + c*h[1]/2.0;
89          z[x_size-1] = (1 - c/2.0)*h[x_size-1] +c*jR*dx + c*h[x_size-2]/2.0;
90          for (int j=1;j<x_size-1;j++){
91              z[j] = (1 - c)*h[j] + c*h[j-1]/2.0 + c*h[j+1]/2.0;
92          }
93
94          y[0] = z[0]/alpha[0];
95          for (int j=1;j<x_size;j++){
96              y[j] = (z[j] - y[j-1]*(-c/2.0))/alpha[j];
97          }
98
99          h[x_size-1] = y[x_size-1];
100         for (int j = x_size-2; j>=0; j--){
101             h[j] = y[j] - beta[j]*h[j+1];
102         }
103         if (i % 100 == 0){
104             for (int l=0; l<x_size; l++){
105                 outputfile << 0.025 + dx*l << "␣" << h[l] << endl;
106             }
107             outputfile << endl;
108         }
109     }
110 }
111
112 int main(void){
113     vector<double> x(201);
114     vector<double> x_init(200);
115     for (int i=0; i<201; i++){ //初期値の計算
116         x[i] = exp(-0.5*(i*dx - 5.0)*(i*dx - 5.0))/sqrt(2*M_PI);
117     }
118     for (int i=0; i<200; i++){
119         x_init[i] = 0.5*(x[i] + x[i+1]);
120     }
121
122
123     cn_dir(x_init, 0, 0, 200, "cn_dir.txt");
124     cn_neu(x_init, 0, 0, 200, "cn_neu.txt");
125     return 0;
126 }
```

コード 3: 問題 2

```
1  #include<iostream>
2  #include<cmath>
3  #include<vector>
4  #include<fstream>
5
6  #define dx 0.05
7  #define dt 0.001
8  using namespace std;
9
10 void fisher(vector<double> x, int x_size, string FILE_NAME){ //アルゴリズムを実行する関数
11     ofstream outputfile(FILE_NAME);
12
13     int step = 40000; //初期化
14     vector<double> y(x_size);
15     y[0] = 1;
```

```
16      y[x_size-1] = 0;
17      for (int i=1; i<step+1; i++){ //計算
18          for (int j=1; j<x_size-1; j++){
19              y[j] = x[j] + x[j]*(1 - x[j])*dt + dt*(x[j-1] - 2*x[j] + x[j+1])/(dx*dx);
20          }
21          for (int k=1; k<x_size-1; k++){
22              x[k] = y[k];
23          }
24          if (i % 10000 == 0){
25              for (int l=1; l<x_size-1; l++){
26                  outputfile << 0.025 + dx*(l-1) << "␣" << y[l] << endl;
27              }
28              outputfile << endl;
29          }
30      }
31  }
32
33  int main(void){
34      vector<double> x(4001);
35      vector<double> x_init(4002);
36      double b;
37      b = 0.25;
38      for (int i=0; i<4001; i++){
39          double e = 1 + exp(b*i*dx - 5);
40          x[i] = 1/(e*e);
41      }
42      for (int i=0; i<4001; i++){ //初期値の計算
43          x_init[i] = 0.5*(x[i] + x[i+1]);
44      }
45      x_init[0] = 1;
46      x_init[4001] = 0;
47      fisher(x_init, 4002, "fisher_0.25.txt");
48
49      b = 0.5;
50      for (int i=0; i<4001; i++){
51          double e = 1 + exp(b*i*dx - 5);
52          x[i] = 1/(e*e);
53      }
54      for (int i=0; i<4001; i++){
55          x_init[i] = 0.5*(x[i] + x[i+1]);
56      }
57      x_init[0] = 1;
58      x_init[4001] = 0;
59      fisher(x_init, 4002, "fisher_0.5.txt");
60
61      b = 1.0;
62      for (int i=0; i<4001; i++){
63          double e = 1 + exp(b*i*dx - 5);
64          x[i] = 1/(e*e);
65      }
66      for (int i=0; i<4001; i++){
67          x_init[i] = 0.5*(x[i] + x[i+1]);
68      }
69      x_init[0] = 1;
70      x_init[4001] = 0;
71      fisher(x_init, 4002, "fisher_1.0.txt");
72      return 0;
```

```
}
```

コード 4: 問題 3

```
#include<iostream>
#include<cmath>
#include<vector>
#include<fstream>

#define dx 0.05
#define dt 0.001
using namespace std;

double psi(double x){
    return sqrt(2)*exp(-2*(x-5)*(x-5))/sqrt(sqrt(M_PI));
}

void shdg(vector<double> R, vector<double> I, int size, string FILE_NAME){ //アルゴリズムを実行する関数
    ofstream outputfile(FILE_NAME);
    vector<double> I_old(size);
    vector<double> R_old(size);
    int step = 8000; //初期化
    for (int i=0; i<size; i++){
        I_old[i] = I[i];
        R_old[i] = R[i];
    }

    for (int i=1; i<step+1; i++){ //計算
        for (int j=1; j<size-1; j++){
            double xj = (j - 0.5)*dx - 10.0;
            R[j] = R_old[j] + dt*(-(I_old[j-1] - 2*I_old[j] + I_old[j+1])/(2.0*dx*dx) + xj*xj*I_old[j]/2.0);
        }
        R[0] = R[size-2];
        R[size-1] = R[1];
        for (int j=1; j<size-1; j++){
            double xj = (j - 0.5)*dx - 10.0;
            I[j] = I_old[j] - dt*(-(R[j-1] - 2*R[j] + R[j+1])/(2.0*dx*dx) + xj*xj*R[j]/2.0);
        }
        I[0] = I[size-2];
        I[size-1] = I[1];

        if (i % 1000 == 0){
            for (int l=1; l<size-1; l++){
                outputfile << (l - 0.5)*dx - 10.0 << "␣" << R[l]*R[l] + I[l]*I_old[l] << endl;
            }
            outputfile << endl;
        }

        for (int j=0; j<size; j++){
            R_old[j] = R[j];
            I_old[j] = I[j];
        }
    }
```

```
50 }
51
52 int main(void){
53     vector<double> R_init(402);
54     vector<double> I_init(402);
55
56     for (int i=1; i<401; i++){
57         R_init[i] = (psi((i-1)*dx-10.0) + psi(i*dx - 10.0))/2.0;
58     }
59     R_init[0] = R_init[400];
60     R_init[401] = R_init[1];
61
62     for (int i=1; i<401; i++){ //初期値の計算
63         double xi = (i - 0.5)*dx - 10.0;
64         I_init[i] = -dt*(-(R_init[i-1] - 2*R_init[i] + R_init[i+1])/(2.0*dx*dx) + xi*xi*
                R_init[i]/2.0);
65     }
66     I_init[0] = I_init[400];
67     I_init[401] = I_init[1];
68     shdg(R_init, I_init, 402, "sh.txt");
69
70     return 0;
71 }
```

参考文献

[1] 実験演習ワーキンググループ (2022),「数理工学実験テキスト_2022 年版」