

数理工学実験  
テーマ:熱方程式の差分法

田中風帆 (1029321151)  
実施場所:自宅

実施:2021 年 10 月 25 日  
提出:2021 年 11 月 7 日

## 1 概要

このレポートでは、全体を通して熱方程式を差分法によって解き、議論する。問題 1 ではオイラー陽解法とクランクニコルソン法、および Dirichlet 境界条件と Neumann 境界条件を組み合わせて拡散方程式を解き、グラフとして出力する。問題 2 では、拡散方程式に非線形項を加えた偏微分方程式である Fisher 方程式を、オイラー陽解法を用いて数値的に解く。問題 3 では、一次元調和振動子のシュレディンガー方程式を数値的に解き、いくつかの時刻で得られた確率密度の値をプロットする。

## 2 問題 1

この問題では、拡散方程式  $\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} \dots (*)$  を、初期条件  $u_0 = \frac{\exp(-\frac{(x-5)^2}{2})}{\sqrt{2\pi}}$  のもとで解く。この際、境界条件として Dirichlet 境界条件および Neumann 境界条件を、手法としてオイラー陽解法およびクランクニコルソン法を用いた。なお、以下  $u_j^n$  は座標  $x$ , 時刻  $t(n = t/\Delta t, j = x/\Delta x$ , ただし  $\Delta t$  は時間の刻み幅、 $\Delta x$  は座標の刻み幅) における方程式の解とする。

1. ここでは、Dirichlet 境界条件において  $u_L = u_R = 0$  を使い、オイラー陽解法  $u_j^{n+1} - u_j^n = \frac{\Delta t(u_{j-1}^n - 2u_j^n + u_{j+1}^n)}{\Delta x^2}$  によって  $(*)$  を解いた。ただし、Dirichlet 境界条件は座標を  $N$  区間に離散化したとき  $u_0^n = u_L, u_{N+1}^n = u_R$  と表せる。時刻の刻み幅は  $\Delta t = 0.01$ , 座標の刻み幅は  $\Delta x = 0.5$  として、 $x = 0$  から  $x = 10$  までを対象に計算を行った。計算の結果得られた出力  $u$  の値は  $x = 5$  に関して対称であり、 $x = 4.25, 4.75$  の時が最も大きく、 $x$  の値がそれらから離れれば離れるほど小さくなって行った。結果のグラフは以下ようになった。

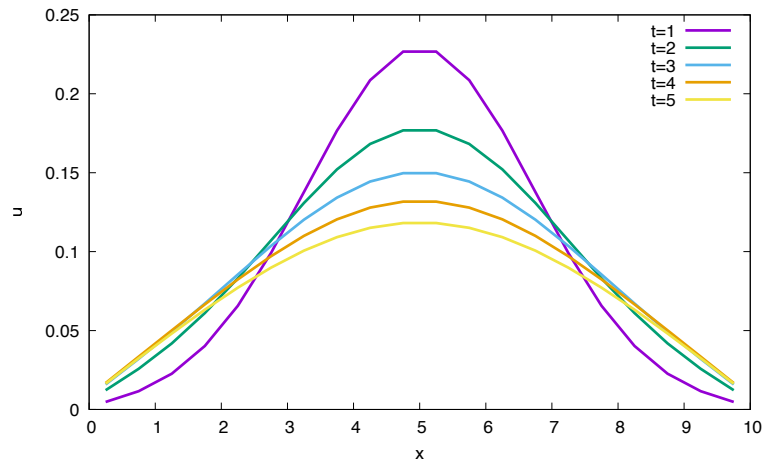


図 1: オイラー陽解法、Dirichlet 境界条件を用いて解いた拡散方程式の解

コードは以下である。

コード 1: オイラー陽解法、Dirichlet 境界条件を用いて拡散方程式を解くコード

---

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <stdbool.h>
5
6 double u_0(double x){
7     return exp(-(x-5)*(x-5)/2) / sqrt(2*M_PI);
8 }
9
10 int main(){
11     //ハイパーパラメータの定義
12     int N = 20;
13     double dt = 0.01; //時間の刻み幅
14     double dx = 0.5; //座標の刻み幅
15     double u[N+2], u_tmp[N+2]; //u の値
16     double time_limit = 6.0;
17     double step_num = (int)time_limit/dt; //繰り返し回数
18     double t = 0; //時刻の初期値
19
20     FILE *fp1_1_1; //ファイルに値を出力
21     FILE *fp1_1_2;
22     FILE *fp1_1_3;
23     FILE *fp1_1_4;
24     FILE *fp1_1_5;
25     fp1_1_1 = fopen("kadai1_1_1.txt", "w");
26     fp1_1_2 = fopen("kadai1_1_2.txt", "w");
27     fp1_1_3 = fopen("kadai1_1_3.txt", "w");
28     fp1_1_4 = fopen("kadai1_1_4.txt", "w");
29     fp1_1_5 = fopen("kadai1_1_5.txt", "w");
30
31
32     //境界条件
33     double u_L = 0, u_R = 0;
34     u[0] = u_L; u[N+1] = u_R; u_tmp[0] = u_L; u_tmp[N+1] = u_R;
35
36     //の初期条件 u
37     for(int j=1; j<=N; j++){
38         u[j] = (u_0((j-1)*dx) + u_0(j*dx)) / 2;
39         u_tmp[j] = 0;
40     }
41
42     //更新
43     for(int i=1; i<=step_num; i++){
44         t += dt;
45         for(int j=1; j<=N; j++){
46             u_tmp[j] = u[j] + dt*(u[j-1]-2*u[j]+u[j+1])/(dx*dx);
47         }
48
49         for(int j=1; j<=N; j++){
50             u[j] = u_tmp[j];
51         }
52
53         //出力
54         if(i==100){
55             for(int j=1; j<=N; j++){
56                 double x = dx*((double)j-0.5);
57                 fprintf(fp1_1_1, "%.8f_%.8f\n", x, u[j]);
58             }
59         }
60     }
```

```

59     }
60     else if(i==200){
61         for(int j=1; j<=N; j++){
62             double x = dx*((double)j-0.5);
63             fprintf(fp1_1_2,"%f\n", x, u[j]);
64         }
65     }
66     else if(i==300){
67         for(int j=1; j<=N; j++){
68             double x = dx*((double)j-0.5);
69             fprintf(fp1_1_3,"%f\n", x, u[j]);
70         }
71     }
72     else if(i==400){
73         for(int j=1; j<=N; j++){
74             double x = dx*((double)j-0.5);
75             fprintf(fp1_1_4,"%f\n", x, u[j]);
76         }
77     }
78     else if(i==500){
79         for(int j=1; j<=N; j++){
80             double x = dx*((double)j-0.5);
81             fprintf(fp1_1_5,"%f\n", x, u[j]);
82         }
83     }
84 }
85 //ファイルを閉じる
86 fclose(fp1_1_1);
87 fclose(fp1_1_2);
88 fclose(fp1_1_3);
89 fclose(fp1_1_4);
90 fclose(fp1_1_5);
91 }
92 }

```

コードの説明を行う。まず、 $u$  の初期条件  $u_0 = \frac{\exp(-\frac{(x-5)^2}{2})}{\sqrt{2\pi}}$  を定義する。コード内において、 $u[i]$  に  $x_i$  の値を代入する。 $u[0]$  および  $u[N+1]$  は境界である。実行部分では、境界条件を定義したのち、 $u$  を表す配列に初期条件を格納する。この時、 $u_j$  に区間  $[(j-1)\Delta t, j\Delta t]$  を割り当てていることから、初期条件として  $u_j^0 = [u_0((j-1)\Delta x) + u_0(j\Delta x)]$  を採用した。その後、時刻  $t$  が 5 を超えるまで更新を繰り返し、 $t = 1, 2, 3, 4, 5$  の時の  $x$  および  $u$  の値を時刻ごとに別々のファイルに出力する。なお、初期条件の採用と同じ理由により、出力する  $x$  座標として  $(j-1/2)\Delta x$  を採用した。

2. ここでは、オイラー陽解法  $u_j^{n+1} - u_j^n = \frac{\Delta t(u_{j-1}^n - 2u_j^n + u_{j+1}^n)}{\Delta x^2}$  および Neumann 境界条件 ( $J_L = J_R = 0$ ) を用いて数値的に解を求める。ただし、Neumann 境界条件は座標を  $N$  区間に離散化したとき  $u_0^n = u_1^n - J_L \Delta x$ ,  $u_{N+1}^n = u_N^n + J_R \Delta x$  と表せる。ここでは、 $\Delta t = 0.01$ ,  $\Delta x = 0.5$  とした。計算の結果得られた出力  $u$  の値は  $x = 5$  に関して対称であり、 $x = 4.25, 4.75$  の時が最も大きく、 $x$  の値がそれらから離れれば離れるほど小さくなって行った。結果のグラフは以下である。

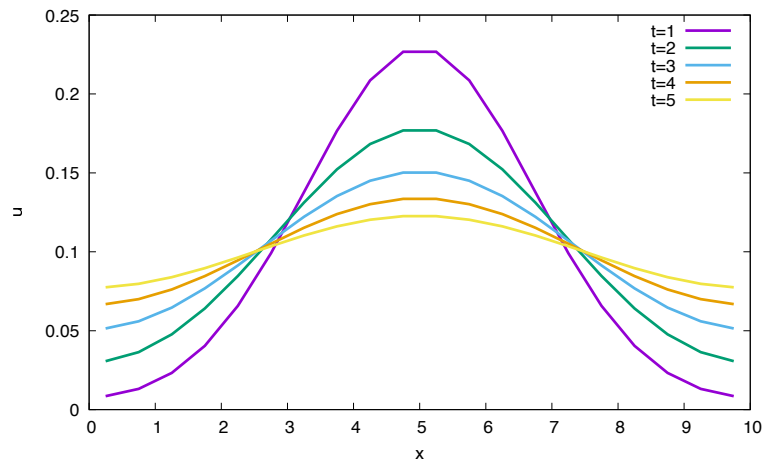


図 2: オイラー陽解法、Neumann 境界条件を用いて解いた拡散方程式の解

使用したコードは以下である。

コード 2: オイラー陽解法、Neumann 境界条件を用いて拡散方程式を解く  
コード

---

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <stdbool.h>
5
6  double u_0(double x){
7      return exp(-(x-5)*(x-5)/2) / sqrt(2*M_PI);
8  }
9
10 int main(){
11     //ハイパーパラメータの定義
12     int N = 20;
13     double dt = 0.01;
14     double dx = 0.5;
15     double u[N+2], u_tmp[N+2];
16     double time_limit = 6.0;
17     double step_num = (int)time_limit/dt;
18     double t = 0;
19     double J_L = 0, J_R = 0;
20     bool recorded_1=false, recorded_2=false, recorded_3=false, recorded_4
        =false, recorded_5=false;
21
22     FILE *fp1_1_1;//ファイルに値を出力
23     FILE *fp1_1_2;
24     FILE *fp1_1_3;
25     FILE *fp1_1_4;
26     FILE *fp1_1_5;
27     fp1_1_1= fopen("kadai1_2_1.txt", "w");
28     fp1_1_2 = fopen("kadai1_2_2.txt", "w");
29     fp1_1_3 = fopen("kadai1_2_3.txt", "w");
30     fp1_1_4 = fopen("kadai1_2_4.txt", "w");
31     fp1_1_5 = fopen("kadai1_2_5.txt", "w");
32

```

```

33
34 //境界条件
35 double u_L = 0, u_R = 0;
36 u[0] = u_L; u[N+1] = u_R; u_tmp[0] = u_L; u_tmp[N+1] = u_R;
37
38 //の初期条件 u
39 for(int j=1; j<=N; j++){
40     u[j] = (u_0((j-1)*dx) + u_0(j*dx)) / 2;
41     u_tmp[j] = 0;
42 }
43
44 for(int i=1; i<=step_num; i++){
45     for(int j=1; j<=N; j++){
46         u_tmp[j] = u[j] + dt*(u[j-1]-2*u[j]+u[j+1])/(dx*dx);
47     }
48
49     for(int j=1; j<=N; j++){
50         u[j] = u_tmp[j];
51     }
52     //境界条件の更新
53     u_tmp[0] = u[1] - J_L*dx;
54     u_tmp[N+1] = u[N] + J_R*dx;
55     u[0] = u_tmp[0];
56     u[N+1] = u_tmp[N+1];
57
58     t += dt;
59
60     //ファイルに出力
61     if(i==100){
62         recorded_1 = true;
63         printf("%.8f", t);
64         for(int j=1; j<=N; j++){
65             double x = dx*((double)j-0.5);
66             fprintf(fp1_1_1, "%.8f_%.8f\n", x, u[j]);
67         }
68     }
69     else if(i==200){
70         recorded_2 = true;
71         printf("%.8f", t);
72         for(int j=1; j<=N; j++){
73             double x = dx*((double)j-0.5);
74             fprintf(fp1_1_2, "%.8f_%.8f\n", x, u[j]);
75         }
76     }
77     else if(i==300){
78         recorded_3 = true;
79         printf("%.8f", t);
80         for(int j=1; j<=N; j++){
81             double x = dx*((double)j-0.5);
82             fprintf(fp1_1_3, "%.8f_%.8f\n", x, u[j]);
83         }
84     }
85     else if(i==400){
86         recorded_4 = true;
87         printf("%.8f", t);
88         for(int j=1; j<=N; j++){
89             double x = dx*((double)j-0.5);
90             fprintf(fp1_1_4, "%.8f_%.8f\n", x, u[j]);
91         }
92     }
93     else if(i==500){
94         recorded_5 = true;
95         printf("%.8f", t);

```

```

96         for(int j=1; j<=N; j++){
97             double x = dx*((double)j-0.5);
98             fprintf(fp1_1_5,"%%.8f_%.8f\n", x, u[j]);
99         }
100     }
101 }
102
103 fclose(fp1_1_1);
104 fclose(fp1_1_2);
105 fclose(fp1_1_3);
106 fclose(fp1_1_4);
107 fclose(fp1_1_5);
108 }

```

---

コードの構造は先ほどのものと同じである。 $J_L, J_R$  を定義し、境界条件として Neumann 境界条件を採用しているのが相違点である。

3. ここでは、クランクニコルソン法および Delichlet 境界条件を用いて解を求める。 $\Delta t = 0.01, \Delta x = 0.05$  とした。計算の結果得られた出力  $u$  の値は  $x = 5$  に関して対称であり、 $x = 4.25, 4.75$  の時が最も大きく、 $x$  の値がそれらから離れれば離れるほど小さくなって行った。以下が結果のグラフである。

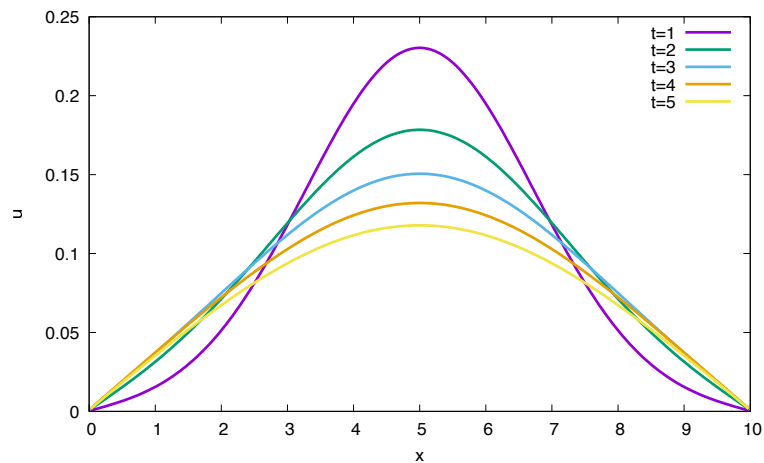


図 3: クランクニコルソン法、Delichlet 境界条件を用いて解いた拡散方程式の解

コードは以下である。

コード 3: クランクニコルソン法、Delichlet 境界条件を用いて拡散方程式を解くコード

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <stdbool.h>
5
6 double u_0(double x){

```

```

7     return exp(-(x-5)*(x-5)/2) / sqrt(2*M_PI);
8 }
9
10 int main(){
11     //ハイパーパラメータの定義
12     int N = 200;
13     double dx = 0.05;
14     double dt = 0.01;
15     double t = 0;
16     double time_limit = 6.0;
17     double c_const = dt/(dx*dx);
18     double u_L = 0, u_R = 0;
19     double a[N+2], b[N+1], c[N+1], alpha[N+2], beta[N+2], z[N+2], u[N
        +2], y[N+2]; //クランクニルソン法に必要な
        変数
20     //初期化
21     a[0] = 0; a[N+1] = 0; b[0] = 0; c[0] = 0;
22     alpha[0] = 0; alpha[N+1] = 0; beta[0] = 0; beta[N+1] = 0; u[N+1] =
        0;
23
24     int step_num = (int)time_limit/dt;
25
26     FILE *fp1_1_1; //ファイルに値を出力
27     FILE *fp1_1_2;
28     FILE *fp1_1_3;
29     FILE *fp1_1_4;
30     FILE *fp1_1_5;
31     fp1_1_1 = fopen("kadai1_3_1.txt", "w");
32     fp1_1_2 = fopen("kadai1_3_2.txt", "w");
33     fp1_1_3 = fopen("kadai1_3_3.txt", "w");
34     fp1_1_4 = fopen("kadai1_3_4.txt", "w");
35     fp1_1_5 = fopen("kadai1_3_5.txt", "w");
36
37     //u の初期条件
38     for(int j=1; j<=N; j++){
39         u[j] = (u_0((j-1)*dx) + u_0(j*dx)) / 2;
40     }
41
42     //Ax=z における A から a,b,c を求める
43     for(int j=1; j<=N; j++){
44         a[j] = 1+c_const;
45         b[j] = -c_const/2;
46         c[j] = -c_const/2;
47     }
48
49     //LU 分解
50     for(int j=1; j<=N; j++){
51         alpha[j] = a[j] - c[j-1]*beta[j-1];
52         beta[j] = b[j]/alpha[j];
53     }
54     beta[N] = 0.0;
55
56     //更新
57     for(int i=1; i<=step_num; i++){
58         t += dt;
59         //z を計算
60         z[1] = (1-c_const)*u[1] + c_const*u_L + c_const*u[2]/2;
61         for(int j=2; j<=N-1; j++){
62             z[j] = (1-c_const)*u[j] + c_const*u[j-1]/2 + c_const*u[j
                +1]/2;
63         }
64         z[N] = (1-c_const)*u[N] + c_const*u_R + c_const*u[N-1]/2;
65

```



```

66 //y を計算
67 for(int j=1; j<=N; j++){
68     y[j] = (z[j]-c[j-1]*y[j-1])/alpha[j];
69 }
70
71 //u を計算
72 for(int j=N; j>=1; j--){
73     u[j] = y[j] - beta[j]*u[j+1];
74 }
75
76 //ファイル出力
77 if(i==100){
78     printf("%.8f", t);
79
80     for(int j=1; j<=N; j++){
81         double x = dx*((double)j-0.5);
82         fprintf(fp1_1_1, "%.8f_%.8f\n", x, u[j]);
83     }
84 }
85
86 else if(i==200){
87     printf("%.8f", t);
88     for(int j=1; j<=N; j++){
89         double x = dx*((double)j-0.5);
90         fprintf(fp1_1_2, "%.8f_%.8f\n", x, u[j]);
91     }
92 }
93
94 else if(i==300){
95     printf("%.8f", t);
96     for(int j=1; j<=N; j++){
97         double x = dx*((double)j-0.5);
98         fprintf(fp1_1_3, "%.8f_%.8f\n", x, u[j]);
99     }
100 }
101
102 else if(i==400){
103     printf("%.8f", t);
104     for(int j=1; j<=N; j++){
105         double x = dx*((double)j-0.5);
106         fprintf(fp1_1_4, "%.8f_%.8f\n", x, u[j]);
107     }
108 }
109
110 else if(i==500){
111     printf("%.8f", t);
112     for(int j=1; j<=N; j++){
113         double x = dx*((double)j-0.5);
114         fprintf(fp1_1_5, "%.8f_%.8f\n", x, u[j]);
115     }
116 }
117
118 fclose(fp1_1_1);
119 fclose(fp1_1_2);
120 fclose(fp1_1_3);
121 fclose(fp1_1_4);
122 fclose(fp1_1_5);
123 }

```

コードの説明を行う。コードの構造は先ほどまでのものと同じである。実行部では境界条件として Dirichlet 境界条件を用い、クランクニ科尔ソン法を実行している。クランクニ科尔ソン法について説明する。まず、初期条

件として  $u_j^n (j = 1 \dots N)$  および境界条件  $u_L = u_R = 0$  を設定する。以下、 $c = \Delta t / \Delta x^2$  とする。次に、 $\alpha_1 = 1 + c, \beta_1 = \frac{-c/2}{\alpha_1}, \alpha_2 = a_2 - c_1 \beta_1, \dots, \alpha_{N-1} = (1 + c) - (-c/2) \beta_{N-2}, \beta_{N-1} = \frac{-c/2}{\alpha_{N-1}}, \alpha_N = a_N - (-c/2) \beta_{N-1}$  に従い  $\alpha_j, \beta_j$  を計算する。次に時刻ごとに更新を行う。時刻  $n$  において、 $u_j^n (j = 1 \dots N)$  から  $z$  ベクトルを  $z_1 = (1 - c)u_1^n + cu_L + \frac{cu_2^n}{2}, z_j = (1 - c)u_j^n + \frac{cu_{j-1}^n}{2} + \frac{cu_{j+1}^n}{2} (2 \leq j \leq N - 1), z_N = (1 - c)u_N^n + cu_R + \frac{cu_{N-1}^n}{2}$  に従って計算する。次に、 $y_1 = \frac{z_1}{\alpha_1}, y_2 = \frac{z_2 - c_1 y_1}{\alpha_2}, \dots, y_N = \frac{z_N - c_{N-1} y_{N-1}}{\alpha_N}$  に従って  $y$  ベクトルを計算し、 $x_N = y_N, x_{N-1} = y_{N-1} - \beta_{N-1} x_N, \dots, x_1 = y_1 - \beta_1 x_2$  に従って  $x$  ベクトルを計算する。これをこの時刻における解とし、出力する。

4. ここでは、クランクニ科尔ソン法および Neumann 境界条件を用いて解を求める。 $\Delta t = 0.01, \Delta x = 0.05$  とした。計算の結果得られた出力  $u$  の値は  $x = 5$  に関して対称であり、 $x = 4.25, 4.75$  の時が最も大きく、 $x$  の値がそれらから離れれば離れるほど小さくなって行った。以下が結果のグラフである。

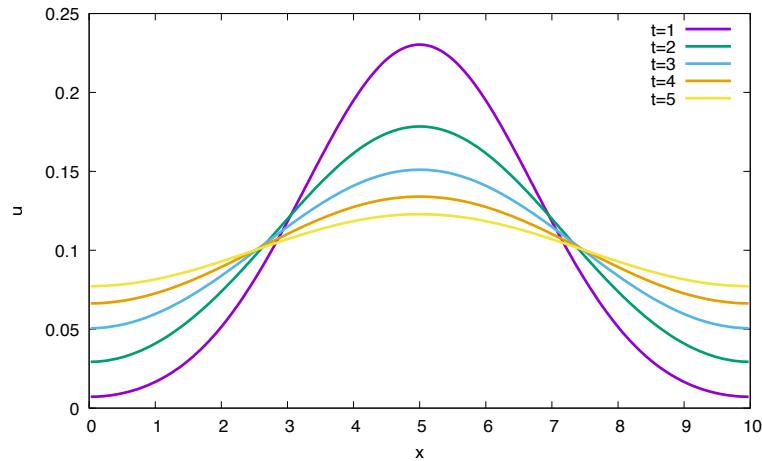


図 4: クランクニ科尔ソン法、Neumann 境界条件を用いて解いた拡散方程式の解

コードは以下である。

コード 4: クランクニ科尔ソン法、Neumann 境界条件を用いて拡散方程式を解くコード

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <stdbool.h>
5
6 //u の初期条件
7 double u_0(double x){
8     return exp(-(x-5)*(x-5)/2) / sqrt(2*M_PI);
```

```

9  }
10
11 int main(){
12     //ハイパーパラメータの定義
13     int N = 200;
14     double dx = 0.05;
15     double dt = 0.01;
16     double t = 0;
17     double time_limit = 6.0;
18     double c_const = dt/(dx*dx);
19     double J_L = 0, J_R = 0;
20     double a[N+2], b[N+1], c[N+1], alpha[N+2], beta[N+2], z[N+2], u[N
        +2], y[N+2];
21     a[0] = 0; a[N+1] = 0; b[0] = 0; c[0] = 0;
22     alpha[0] = 0; alpha[N+1] = 0; beta[0] = 0; beta[N+1] = 0; u[N+1] =
        0;
23     int step_num = (int)time_limit/dt;
24
25     FILE *fp1_1_1; //ファイルに値を出力
26     FILE *fp1_1_2;
27     FILE *fp1_1_3;
28     FILE *fp1_1_4;
29     FILE *fp1_1_5;
30     fp1_1_1 = fopen("kadai1_4_1.txt", "w");
31     fp1_1_2 = fopen("kadai1_4_2.txt", "w");
32     fp1_1_3 = fopen("kadai1_4_3.txt", "w");
33     fp1_1_4 = fopen("kadai1_4_4.txt", "w");
34     fp1_1_5 = fopen("kadai1_4_5.txt", "w");
35
36     //u の初期条件
37     for(int j=1; j<=N; j++){
38         u[j] = (u_0((j-1)*dx) + u_0(j*dx)) / 2;
39     }
40
41     //Ax=z における A から a,b,c を求める
42     for(int j=1; j<=N; j++){
43         a[j] = 1+c_const;
44         b[j] = -c_const/2;
45         c[j] = -c_const/2;
46     }
47     a[1] = 1+c_const/2;
48     a[N] = 1+c_const/2;
49
50     //LU 分解
51     for(int j=1; j<=N; j++){
52         alpha[j] = a[j] - c[j-1]*beta[j-1];
53         beta[j] = b[j]/alpha[j];
54     }
55     beta[N] = 0.0;
56
57     for(int i=1; i<=step_num; i++){
58         t += dt;
59         //z を計算
60         z[1] = (1-c_const/2)*u[1] + c_const*J_L*dx + c_const*u[2]/2;
61         for(int j=2; j<=N-1; j++){
62             z[j] = (1-c_const)*u[j] + c_const*u[j-1]/2 + c_const*u[j
                +1]/2;
63         }
64         z[N] = (1-c_const/2)*u[N] + c_const*J_R*dx + c_const*u[N
            -1]/2;
65
66         //y を計算
67         for(int j=1; j<=N; j++){

```

```

68     y[j] = (z[j]-c[j-1]*y[j-1])/alpha[j];
69 }
70
71 //u を計算
72 for(int j=N; j>=1; j--){
73     u[j] = y[j] - beta[j]*u[j+1];
74 }
75
76 //ファイル出力
77 if(i==100){
78     printf("%.8f", t);
79     for(int j=1; j<=N; j++){
80         double x = dx*((double)j-0.5);
81         fprintf(fp1_1_1, "%.8f_%.8f_%.8f\n", x, u[j]);
82     }
83 }
84 else if(i==200){
85     printf("%.8f", t);
86     for(int j=1; j<=N; j++){
87         double x = dx*((double)j-0.5);
88         fprintf(fp1_1_2, "%.8f_%.8f_%.8f\n", x, u[j]);
89     }
90 }
91 else if(i==300){
92     printf("%.8f", t);
93     for(int j=1; j<=N; j++){
94         double x = dx*((double)j-0.5);
95         fprintf(fp1_1_3, "%.8f_%.8f_%.8f\n", x, u[j]);
96     }
97 }
98 else if(i==400){
99     printf("%.8f", t);
100    for(int j=1; j<=N; j++){
101        double x = dx*((double)j-0.5);
102        fprintf(fp1_1_4, "%.8f_%.8f_%.8f\n", x, u[j]);
103    }
104 }
105 else if(i==500){
106     printf("%.8f", t);
107     for(int j=1; j<=N; j++){
108         double x = dx*((double)j-0.5);
109         fprintf(fp1_1_5, "%.8f_%.8f_%.8f\n", x, u[j]);
110     }
111 }
112 }
113
114 fclose(fp1_1_1);
115 fclose(fp1_1_2);
116 fclose(fp1_1_3);
117 fclose(fp1_1_4);
118 fclose(fp1_1_5);
119 }

```

コードの説明を行う。

構造は Dirichlet 境界条件を用いたコードと同じである。相違点は、境界条件として Neumann 境界条件を用いているため、境界の初期条件を  $J_L = J_R = 0$  としている点と  $\alpha, \beta$  を  $\alpha_1 = 1 + c/2, \beta_1 = \frac{-c/2}{\alpha_1}, \dots, \alpha_{N-1} = (1 + c/2) - (-c/2)\beta_{N-2}, \beta_{N-1} = \frac{-c/2}{\alpha_{N-1}}, \alpha_N = a_N - (-c/2)\beta_{N-1}$  と更新している点、そして  $z$  ベクトルを  $z_1 = (1-c)u_1^n + cu_L + \frac{cu_2^n}{2}, z_j = (1-c/2)u_j^n + \frac{cu_{j-1}^n}{2} + \frac{cu_{j+1}^n}{2} (2 \leq$

$j \leq N-1$ ),  $z_N = (1 - c/2)u_N^n + cu_R + \frac{cu_{N-1}^n}{2}$  に従って計算する点である。

以上 1 から 4 の結果より、この拡散方程式の解は  $x = 5$  に関して対称な分布となることが視覚的にもわかる。また、Dirichlet 境界条件を用いると境界における  $u$  の値が 0 に近くなり、Neumann 境界条件を用いると  $t$  の値により異なる境界の値が得られることがわかる。さらに、 $t$  の値が大きいほど  $x=5$  における  $u$  の値が小さくなることがわかる。また、 $x = 5$  において、 $t$  が大きいほど解の値は小さくなることがわかる。

### 3 問題 2

ここでは、初期条件  $u_0(x) = \frac{1}{(1+\exp(bx-5))^2}$ 、境界条件  $u(0, t) = 1, u(L, t) = 0 (L = 200)$  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  $\frac{u_j^{n+1} - u_j^n}{\Delta t} = f(u_j^n) + \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2}$  を用いて解く。刻み幅は  $\Delta x = 0.05, \Delta t = 0.001$  とし、 $t = 10, 20, 30, 40$  の  $u(x, t)$  の値を出力する。また、 $u(x, t)$  の値を  $b$  の値ごとに横軸  $x$ 、縦軸  $u$  の図に出力し、得られた結果について考察する。なお、以下  $u_j^n$  は座標  $x$ 、時刻  $t (n = t/\Delta t, j = x/\Delta x)$ 、ただし  $\Delta t$  は時間の刻み幅、 $\Delta x$  は座標の刻み幅) における方程式の解とする。得られた出力は初期条件  $u = 1$  に始まり、 $x$  の値が大きくなるにつれて単調減少するものであった。得られた図は以下である。

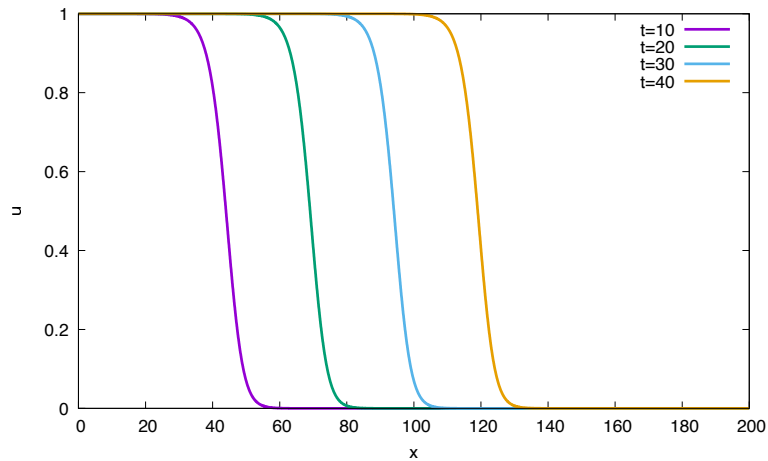


図 5:  $b=0.25$  の時

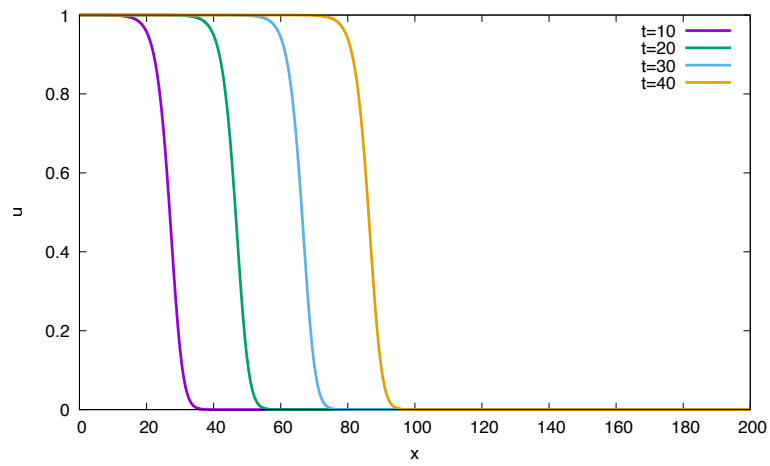


図 6:  $b=0.5$  の時

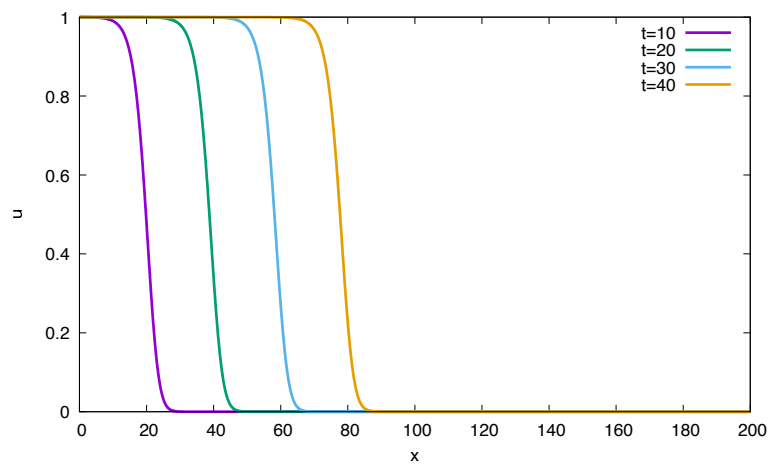


図 7:  $b=1.0$  の時

コードは以下である。

コード 5: Fisher 方程式を解くコード

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <stdbool.h>
5
6 double u_0(double x, double b){
7     return 1/((1+exp(b*x-5))*(1+exp(b*x-5)));
8 }
```

```

9
10 double f(double u_){
11     return u_*(1-u_);
12 }
13
14 int main(){
15     //ハイパーパラメータの定義
16     int N = 4000;
17     double dt = 0.001;//時刻の刻み幅
18     double dx = 0.05;//座標の刻み幅
19     double b;
20     double time_limit = 50.0;
21     double step_num = (int)time_limit/dt;//繰り返し回数
22
23     for(int b_=0; b_<=2; b_++){
24         FILE *fp1.1.1;//ファイルに値を出力
25         FILE *fp1.1.2;
26         FILE *fp1.1.3;
27         FILE *fp1.1.4;
28         double u[N+2], u_tmp[N+2];
29         double t = 0;
30
31         if(b_==0){
32             b = 0.25;
33             fp1.1.1= fopen("kadai2_b025_10.txt", "w");
34             fp1.1.2 = fopen("kadai2_b025_20.txt", "w");
35             fp1.1.3 = fopen("kadai2_b025_30.txt", "w");
36             fp1.1.4 = fopen("kadai2_b025_40.txt", "w");
37             printf("%d_",b_);
38         }
39         if(b_==1){
40             b = 0.5;
41             fp1.1.1= fopen("kadai2_b05_10.txt", "w");
42             fp1.1.2 = fopen("kadai2_b05_20.txt", "w");
43             fp1.1.3 = fopen("kadai2_b05_30.txt", "w");
44             fp1.1.4 = fopen("kadai2_b05_40.txt", "w");
45             printf("%d_",b_);
46         }
47         if(b_==2){
48             b = 1.0;
49             fp1.1.1= fopen("kadai2_b10_10.txt", "w");
50             fp1.1.2 = fopen("kadai2_b10_20.txt", "w");
51             fp1.1.3 = fopen("kadai2_b10_30.txt", "w");
52             fp1.1.4 = fopen("kadai2_b10_40.txt", "w");
53             printf("%d_",b_);
54         }
55         //境界条件
56         double u_L = 1, u_R = 0;
57         u[0] = u_L; u[N+1] = u_R; u_tmp[0] = u_L; u_tmp[N+1] = u_R;
58
59         //初期条件
60         for(int j=1; j<=N; j++){
61             u[j] = (u_0((j-1)*dx,b) + u_0(j*dx,b)) / 2;
62             u_tmp[j] = 0;
63         }
64
65         //オイラー法
66         for(int i=1; i<=step_num; i++){
67             t += dt;
68             for(int j=1; j<=N; j++){
69                 u_tmp[j] = u[j] + (f(u[j])+(u[j-1]-2*u[j]+u[j+1]))/(dx*dx
70                     ))*dt;
71             }
72         }

```

```

71     for(int j=1; j<=N; j++){
72         u[j] = u_tmp[j];
73     }
74
75     if(i==10000){
76         printf("%.8f_\n",t);
77         for(int j=1; j<=N; j++){
78             double x = dx*((double)j-0.5);
79             fprintf(fp1_1.1, "%.8f_%.8f_\n", x, u[j]);
80         }
81     }
82     else if(i==20000){
83         printf("%.8f_\n",t);
84         for(int j=1; j<=N; j++){
85             double x = dx*((double)j-0.5);
86             fprintf(fp1_1.2, "%.8f_%.8f_\n", x, u[j]);
87         }
88     }
89     else if(i==30000){
90         printf("%.8f_\n",t);
91         for(int j=1; j<=N; j++){
92             double x = dx*((double)j-0.5);
93             fprintf(fp1_1.3, "%.8f_%.8f_\n", x, u[j]);
94         }
95     }
96     else if(i==40000){
97         printf("%.8f_\n",t);
98         for(int j=1; j<=N; j++){
99             double x = dx*((double)j-0.5);
100             fprintf(fp1_1.4, "%.8f_%.8f_\n", x, u[j]);
101         }
102     }
103     }
104     fclose(fp1_1.1);
105     fclose(fp1_1.2);
106     fclose(fp1_1.3);
107     fclose(fp1_1.4);
108 }
109 }

```

コードの説明を行う。まず、時刻の刻み幅や座標の刻み幅、繰り返し回数などの設定を行う。実行部分では、3種類の  $b$  の値それぞれに対し、同じ回数だけオイラー陽解法で解の更新を行う。 $t = 10, 20, 30, 40, 50$  の時には、設定されている  $b$  の値およびその時の  $t$  の値に応じたファイルに、各座標における  $x$  の値を出力する。

結果の考察を行う。まず、どの  $t, b$  においても、 $x$  の値が大きくなるにつれて  $u$  の値が単調減少していく。加えて、どの  $b$  においても  $t$  の値が大きいほど減衰のはじまる  $x$  の値が大きくなり、また  $u$  が 0 に収束する  $x$  の値も大きくなる。また、 $b$  の値が大きくなるほど  $u(x)$  が減衰し始める  $x$  の値が小さくなる。さらに、 $b$  の値が小さくなるほど、 $t=10, 20, 30, 40$  において同じ  $u$  の値を与える  $x$  の間隔が広がる。



## 4 問題3

ここでは、シュレディンガー方程式  $i\frac{\partial\psi}{\partial t} = (-\frac{1}{2}\frac{\partial^2}{\partial x^2} + \frac{x^2}{2})\psi(x,t)$  を、初期条件  $\psi(x,0) = \frac{\sqrt{2}\exp(-2(x-5)^2)}{\pi^{\frac{1}{4}}}$  のもとで数値的にとく。その際、空間領域は  $[-10,10]$  とした。座標  $x$ , 時刻  $t(n = t/\Delta t, j = x/\Delta x)$ , ただし  $\Delta t$  は時間の刻み幅、 $\Delta x$  は座標の刻み幅) における波動関数の実部、虚部をそれぞれ  $R_j^n, I_j^n$  とおき、 $1 \leq j \leq N(N\Delta x = 20)$  に対し  $R_j^{n+1} = R_j^n + \Delta t(-\frac{I_{j-1}^n - 2I_j^n + I_{j+1}^n}{2\Delta x^2} + \frac{x_j^2 I_j^n}{2})$ ,  $I_j^{n+1} = I_j^n - \Delta t(-\frac{R_{j-1}^{n+1} - 2R_j^{n+1} + R_{j+1}^{n+1}}{2\Delta x^2} + \frac{x_j^2 I_j^{n+1}}{2})$  の式に従って更新を行う。ただし  $x_j = (j - \frac{1}{2})\Delta x - 10$  とし、境界条件は全ての  $n$  において  $R_0^n = R_N^n, R_{N+1}^n = R_1^n, I_0^n = I_N^n, I_{N+1}^n = I_1^n$

と表す。さらに、初期条件は  $1 \leq j \leq N$  で  $R_j^0 = \frac{1}{2}[\psi((j-1)\Delta x - a)]$ ,  $- \Delta t(-\frac{R_{j-1}^0 - 2R_j^0 + R_{j+1}^0}{2\Delta x^2} + \frac{x_j^2 R_j^0}{2})$  とす

る。これらの条件下において、 $\Delta x = 0.05, \Delta t = 0.001$  として系を数値的に解き、 $t = 1, 2, 3, 4, 5, 6, 7, 8$  における確率密度  $(R_j^n)^2 + I_j^n I_j^{n-1} (1 \leq j \leq N)$  の値を出力した。また、得られた確率密度 (以下  $u$  とおく) を一つのグラフとして出力した。結果、それぞれの  $t$  において、途中まで単調増加し、途中からは単調減少する数値の出力が得られた。 $t$  により、どこで  $u$  がピークに達するかは異なっていた。

出力された数値を横軸  $x$ 、縦軸  $u$  としてプロットすると、それぞれの  $t$  に対し左右対称な確率密度関数が得られた。以下が得られたグラフである。

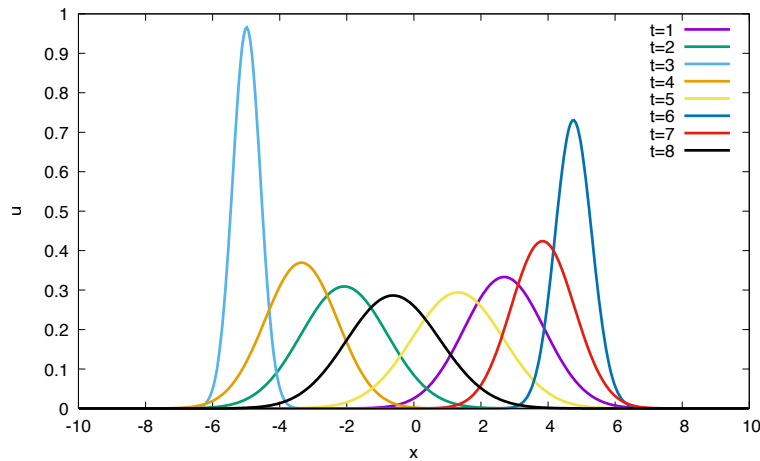


図 8: シュレディンガー方程式の解. 縦軸が確率密度  $u$ , 横軸が  $x$

以下がコードである。

コード 6: シュレディンガー方程式を解き確立密度を求めるコード

```
1 #include <stdio.h>
```

```

2  #include <stdlib.h>
3  #include <math.h>
4  #include <stdbool.h>
5
6  //Ψ の定義
7  double psai(double x){
8      return sqrt(2)*exp(-2*(x-5)*(x-5)) / pow(M_PI, 1.0/4);
9  }
10
11 int main(){
12     //ハイパーパラメータ
13     int N = 400;
14     double dx = 0.05;
15     double dt = 0.001;
16     double R[N+2], I[N+2], I_kari[N+2], x[N+2];
17     double time_limit = 10;
18     int step_num = (int)time_limit/dt;
19     double L = 20;
20     double t=0;
21     FILE *fp;
22
23     //を定義 x
24     for(int i=1; i<=N; i++){
25         x[i] = (i-1/2)*dx - L/2;
26     }
27     //R の初期条件
28     for(int i=1; i<=N; i++){
29         R[i] = (psai((i-1)*dx-L/2) + psai(i*dx-L/2)) / 2;
30     }
31     //R の境界を初期化
32     R[0] = R[N];
33     R[N+1] = R[1];
34
35     //I の初期条件
36     for(int i=1; i<=N; i++){
37         I[i] = -dt * (-(R[i-1] - 2*R[i] + R[i+1])/(2*dx*dx) + x[i]*x[i]*
38             R[i]/2);
39     }
40     //I の境界を初期化
41     I[0] = I[N];
42     I[N+1] = I[1];
43
44     //出力対象の t によって出力ファイルを変更する
45     for(int i=1; i<=8; i++){
46         //ファイルを開く
47         if(i==1) fp = fopen("kadai3_1.txt","w");
48         if(i==2) fp = fopen("kadai3_2.txt","w");
49         if(i==3) fp = fopen("kadai3_3.txt","w");
50         if(i==4) fp = fopen("kadai3_4.txt","w");
51         if(i==5) fp = fopen("kadai3_5.txt","w");
52         if(i==6) fp = fopen("kadai3_6.txt","w");
53         if(i==7) fp = fopen("kadai3_7.txt","w");
54         if(i==8) fp = fopen("kadai3_8.txt","w");
55
56         //次の出力対象の t まで更新を行う
57         for(int k=1; k<=1000; k++){
58             t += dt;
59             //R の更新
60             for(int j=1; j<=N; j++){
61                 R[j] = R[j] + dt*(-(I[j-1]-2*I[j]+I[j+1]) / (2*dx*dx) +
62                     x[j]*x[j]*I[j]/2);
63             }

```

```

63         //R の境界を更新
64         R[0] = R[N];
65         R[N+1] = R[1];
66         //I の更新確率密度を求めるのに一つ前の時刻の I が必要なので
           に格納 I_kari.
67         for(int j=1; j<=N; j++){
68             I_kari[j] = I[j];
69             I[j] = I[j] - dt*(-(R[j-1]-2*R[j]+R[j+1]) / (2*dx*dx)
               + x[j]*x[j]*R[j]/2);
70         }
71         //I の境界を更新
72         I[0] = I[N];
73         I[N+1] = I[1];
74     }
75     //出力
76     for(int j=1; j<=N; j++){
77         double X = dx*((double)j-0.5)-L/2;
78         double mitudo = R[j]*R[j] + I[j]*I_kari[j]; //確率密度
79         fprintf(fp,"%0.8f_%.8f_\n", X, mitudo);
80     }
81     fclose(fp);
82 }
83 }

```

コードの説明を行う。まず、 $x$  に対して  $\psi(x, 0) = \frac{\sqrt{2} \exp(-2(x-5)^2)}{\pi^{\frac{1}{4}}}$  を返す関数 `psai` を定義する。実行部では、まず  $x$  を定義し、順に配列  $x$  に格納していく。次に  $R_1$   $R_N$  を求めて配列に格納し、 $R$  の境界条件を求める。その後  $I_1$  から  $I_N$  を求め、 $I$  の境界条件を求めてそれぞれ配列に格納する。その後は上述した方法によって  $R$  と  $I$  および  $R$  と  $I$  の境界条件の更新を行い、更新が 1000 回行われるごとに、その時の確率分布をファイルに出力する。なお、確率分布を求める際に一つ前の時刻の  $I$  の値が必要となるため、これを  $I_{kari}$  という配列に格納しておき、利用する。

## 5 まとめ

このレポートでは、問題 1 で拡散方程式  $\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} \dots (*)$  を、初期条件  $u_0 = \frac{\exp(\frac{-(x-5)^2}{2})}{\sqrt{2\pi}}$  のもとで解いた。この際、境界条件として Dirichlet 境界条件および Neumann 境界条件を、手法としてオイラー陽解法およびクラニニコルソン法を用いた。結果、 $u$  の値は  $x = 5$  に関して対称であり、 $x$  の値が山の中央から離れれば離れるほど小さくなって行った。さらに、Delichlet 境界条件を用いると境界における  $u$  の値が 0 に近くなり Neumann 境界条件を用いると  $t$  の値により異なる境界の値が得られること、そして  $t$  の値が大きいほど  $x=5$  における  $u$  の値が小さくなることがわかった。問題 2 では、初期条件  $u_0(x) = \frac{1}{(1+\exp(b(x-5)))^2}$ 、境界条件  $u(0,t) = 1, u(L,t) = 0 (L = 200)$  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  $\frac{u_j^{n+1}-u_j^n}{\Delta t} = f(u_j^n) + \frac{u_{j-1}^n-2u_j^n+u_{j+1}^n}{\Delta x^2}$  を用いて解き、 $b$  の値ごとに  $t = 10, 20, 30, 40$  の時の  $u$  の値を出力した。結果、どの  $t, b$  においても、 $x$  の値が大きくなるにつ

れて  $u$  の値が単調減少していくこと、どの  $b$  においても  $t$  の値が大きいほど減衰のはじまる  $x$  の値が大きくなり、また  $u$  が 0 に収束する  $x$  の値も大きくなること、 $b$  の値が大きくなるほど  $u(x)$  が減衰し始める  $x$  の値が小さくなること、 $b$  の値が小さくなるほど  $t=10,20,30,40$  において同じ  $u$  の値を与える  $x$  の間隔が広くなることがわかった。問題 3 では、シュレディンガー方程式  $i\frac{\partial\psi}{\partial t} = (-\frac{1}{2}\frac{\partial^2}{\partial x^2} + \frac{x^2}{2})\psi(x,t)$  を、初期条件  $\psi(x,0) = \frac{\sqrt{2}\exp(-2(x-5)^2)}{\pi^{\frac{1}{4}}}$  のもとで数値的に解き、 $t = 1, 2, 3, 4, 5, 6, 7, 8$  における解を出力、プロットした。結果、それぞれの  $t$  に対し左右対称な確率密度関数が得られた。また、対称軸となる  $x$  の値は  $t$  によって異なった。