ゼミ レポート 01

山田朔也

2022 年 4 月 25 日

1　本レポートについて

本レポートでは、熱伝導方程式を数値的に解く問題の結果を示すものである。また問題はいくつかに分かれているが、基本事項は同じのため、それらについては問題の結果の前にまとめて記述しておく。

2　各種基本事項の説明

2.1　問題について

本問題は以下の図 1 のような一次元の棒の両端を 0 ℃としたときの熱伝導方程式を、数値的に解くことを目的としている。つまり、時間と空間を離散化して、各代表点の温度変化を計算して行けば良い。

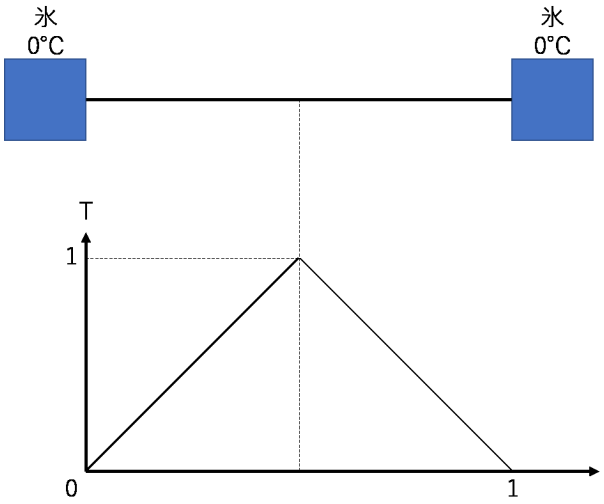


図 1: 本問題における初期条件

与えられる初期条件は、棒の長さが 1 で、棒の両端の温度が 0 ℃。それ以外の点は図 1 のように中央に近づくにつれて高くなっていき、中央は 1 ℃となる。そして、棒の両端は温度 0 ℃の氷と接触しており、温度は 0 ℃を維持する。以上が初期条件となっている。

また、長さ方向の座標軸を x 軸、時刻を t、温度 T を x と t からなる関数とする。このとき熱伝導方程式は

$$\frac{\partial T(x,t)}{\partial t} = \frac{\partial^2 T(x,t)}{\partial x^2} \tag{1}$$

となり、境界条件は

$$T(0,t) = T(1,t) = 0 \tag{2}$$

と表せる。

2.2 計算モデルについて

計算モデルを用意するために、式 1 を離散化する必要があるため、それを行っていく。まずは関数 T を各代表点 i と時間 t からなる関数とみなす。ここから、まずは中心差分近似を用いて、熱伝導方程式の空間微分を離散化する。この条件で関数 T をテイラー展開する。すると

$$T(i+1,t) = T(i,t) + dxT'(i,t) + \frac{dx^2}{2}T''(i,t) + \frac{dx^3}{6}T'''(i,t) + O(dx^4) + \dots \tag{3}$$

$$T(i-1,t) = T(i,t) - dxT'(i,t) + \frac{dx^2}{2}T''(i,t) - \frac{dx^3}{6}T'''(i,t) + O(dx^4) + \dots \tag{4}$$

の 2 式が得られる。この 2 式の左辺と右辺をそれぞれ足し合わせると、

$$T(i+1,t) + T(i-1,t) = 2T(i,t) + dx^2T''(i,t) + O(dx^4) \tag{5}$$

$$\rightarrow T''(i,t) = \frac{\partial^2 T}{\partial x^2} \sim \frac{T(i-1,t) - 2T(i,t) + T(i+1,t)}{dx^2} \tag{6}$$

という式が得られる。式 6 にあるように、この式は熱伝導方程式の右辺に近似できる式となっている。これにより、空間が離散化されたときの近似式が得られた。

次はオイラー法を用いて、式 1 の時間を離散化していく。オイラー法は、仮に求めたい式を y として、$\frac{dy}{dt} = f(t, y(x))$ の形で既知の関数 $f(t, y(x))$ で表せるとき。

$$y(t_0 + \Delta t) = y(t_0) + \Delta t y'(t) + \dots \tag{7}$$

という形で近似する手法となっている。これは以下の図 2 にあるように、$y(t_0 + \Delta t)$ の値を、t_0 における傾き $\frac{dy}{dt}$ と Δt を掛け合せた値に、$y(t_0)$ の値を足したもので近似している。

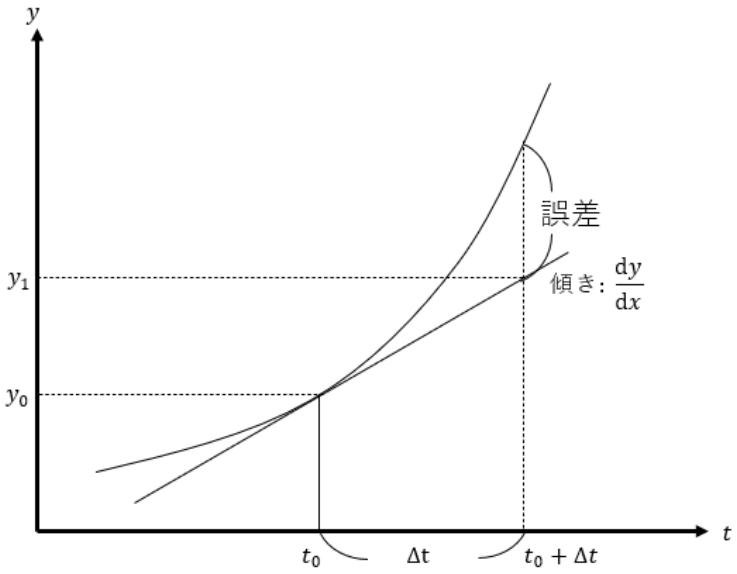


図 2: オイラー法の視覚的説明

ただし、計算しているのはあくまで傾きであるため、図 2 のように誤差が出る。そのため、Δt を十分に小さくする必要がある。また、式 7 では所謂傾きである一次微分までしか行っていないが、より精度を求めるなら二次以降も計算すると良い。

これを熱伝導方程式に適用すると

$$T(i,t+\Delta t) = T(i,t) + \Delta \frac{T(i-1,t) - 2T(i,t) + T(i+1,t)}{dx^2} \tag{8}$$

という式が得られる。

ここから、以下のようなアルゴリズムを用いることで数値的に解が得られる。

1. 各種初期条件を設定する。
   - 棒の分割数 n、またこれによって代表点間の距離 $dx = \frac{1}{n}$ が決定する。
   - 温度の初期条件を T に設定する。
   - 時間の刻み幅 dt の決定
2. 初期条件と式 8 から各代表点の dt 後の状態を計算する。
3. 上の項目 2 を任意の回数行う。

2.3 使用したプログラムについて

まず、前の章で策定したアルゴリズムを元に作成した、C 言語のプログラムのソースコードを以下に記載する。

Listing 1: 熱伝導方程式の数値的に解を得るプログラム

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

```c
#include <stdbool.h>

int printout(int n, double* T, int t) {
	// 最初に呼ばれたときのみ実行
	static bool IsFirst = true;
	int i;
	if(IsFirst) {
		// 表最初の行を出力
		printf("time␣");
		for(i = 0; i <= n; i++){
			printf("T%d␣", i);
		}
		printf("\n");
		IsFirst = false;
	}

	// 中身を出力
	printf("%d*dt␣", t);
	for(i = 0; i <= n; i++) {
		printf("%.4lf␣", T[i]);
	}
	printf("\n");
}

int calcDT(double* T, double* DT, double dx, double dt, int n) {
	int i = 0;
	DT[0] = 0; DT[n] = 0;
	for (i = 1; i < n; i++) {
		DT[i] = dt*(T[i-1] - 2*T[i] + T[i+1])/(dx*dx);
	}
}

int calcT(double* T, double* DT, int n) {
	int i = 0;
	for (i = 1; i < n; i++) {
		T[i] = T[i] + DT[i];
	}
}

int main() {
	// 各種初期値の取得
	int n = 0;
	scanf("%d", &n);

	double dt = 0.0;
	scanf("%lf", &dt);
```

```
49
50          double dx;
51          dx = (double)1.0 / (double)n;
52
53          double T[n+1];
54          double DT[n+1];
55
56          // 初期条件設定
57          int i = 0;
58          for (i = 0; i <= n/2; i++) {
59                  T[i] = 2*dx*i;
60                  T[n-i] = T[i];
61          }
62          printout(n, T, 0);
63
64          // 計算のメインループ
65          int t = 0;
66          for (t = 1; t <= 10; t++) {
67                  calcDT(T, DT, dx, dt, n);
68                  calcT(T, DT, n);
69                  printout(n, T, t);
70          }
71
72          return 0;
73  }
```

このプログラムを実行した後、まずは分割数 n を標準入力で入力。続いて dt も入力する。するとそれを元に、初期条件及び、熱伝導方程式の数値的に解を得ることができる。

結果の出力方法は、標準出力へとなっている。出力形式は以下の表**??**のようになっている。計算を行うにあたって、配列 T と DT を用意し、それぞれ別に計算を行っている。これは、配列 T は $T(i,t+\Delta t)$ の値を保持し、DT は $\Delta\frac{T(i-1,t)-2T(i,t)+T(i+1,t)}{dx^2}$ を保持することで、上書きなどを避けるためである。各関数の説明は以下のようになる。

- 関数 printout。配列 T とその他の変数から、必要な情報を出力する。
- 関数 calcDT。配列 DT に入る数値を、前のループで計算してあった T から計算する。
- 関数 calcT。前のループでの計算結果の残っている配列 T と、新たに計算した DT から、今ループでの T を計算する。

3 問題への解答

3.1 問題 1

問題の内容は、熱伝導方程式の解を数値的に得るプログラムを作成し、それが与えられたテキストの表 1 と一致しているかを確認するものである。

以下にプログラムを実行して計算した結果を表 1 として記載する。

表 1: 作成したプログラムを $n = 10, dt = 0.001$ で実行した出力結果

time	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0*dt	0.0000	0.2000	0.4000	0.6000	0.8000	1.0000	0.8000	0.6000	0.4000	0.2000	0.0000
1*dt	0.0000	0.2000	0.4000	0.6000	0.8000	0.9600	0.8000	0.6000	0.4000	0.2000	0.0000
2*dt	0.0000	0.2000	0.4000	0.6000	0.7960	0.9280	0.7960	0.6000	0.4000	0.2000	0.0000
3*dt	0.0000	0.2000	0.4000	0.5996	0.7896	0.9016	0.7896	0.5996	0.4000	0.2000	0.0000
4*dt	0.0000	0.2000	0.4000	0.5986	0.7818	0.8792	0.7818	0.5986	0.4000	0.2000	0.0000
5*dt	0.0000	0.2000	0.3998	0.5971	0.7732	0.8597	0.7732	0.5971	0.3998	0.2000	0.0000
6*dt	0.0000	0.2000	0.3996	0.5950	0.7643	0.8424	0.7643	0.5950	0.3996	0.2000	0.0000
7*dt	0.0000	0.1999	0.3992	0.5924	0.7551	0.8268	0.7551	0.5924	0.3992	0.1999	0.0000
8*dt	0.0000	0.1999	0.3986	0.5893	0.7460	0.8125	0.7460	0.5893	0.3986	0.1999	0.0000
9*dt	0.0000	0.1998	0.3978	0.5859	0.7370	0.7992	0.7370	0.5859	0.3978	0.1998	0.0000
10*dt	0.0000	0.1996	0.3968	0.5822	0.7281	0.7867	0.7281	0.5822	0.3968	0.1996	0.0000

今回の与えられたテキストの表 1 と、本レポートの表 1 は一致しているため、これを満たしている。

3.2 問題 2

問題の内容は、問題一の結果を gnuplot を用いてグラフとして描画することである。
実際に計算結果を gnuplot でグラフにすると以下の図 3 ようになった。

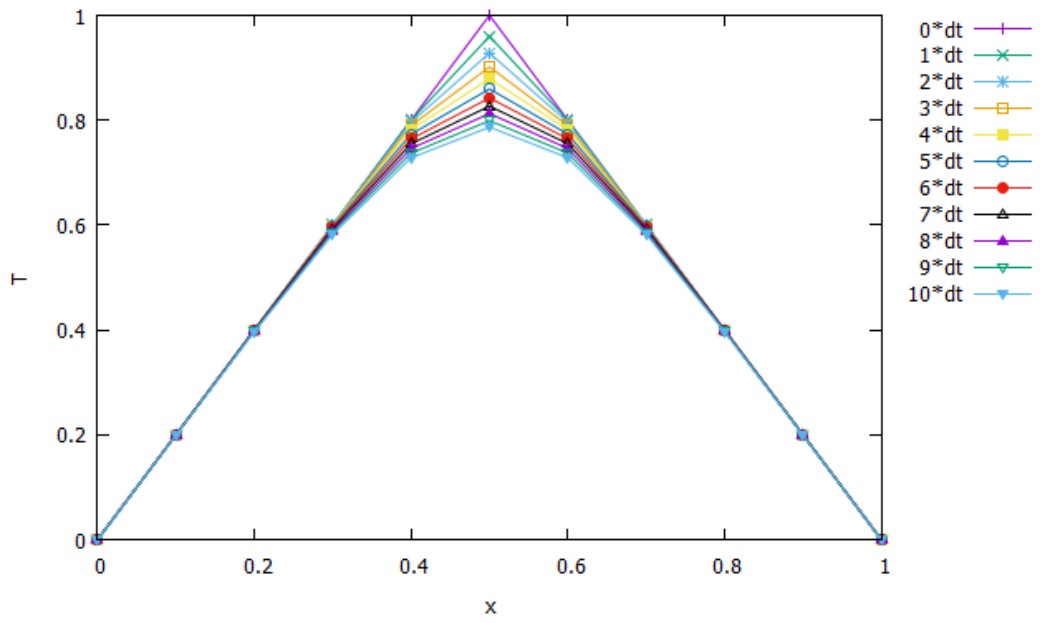
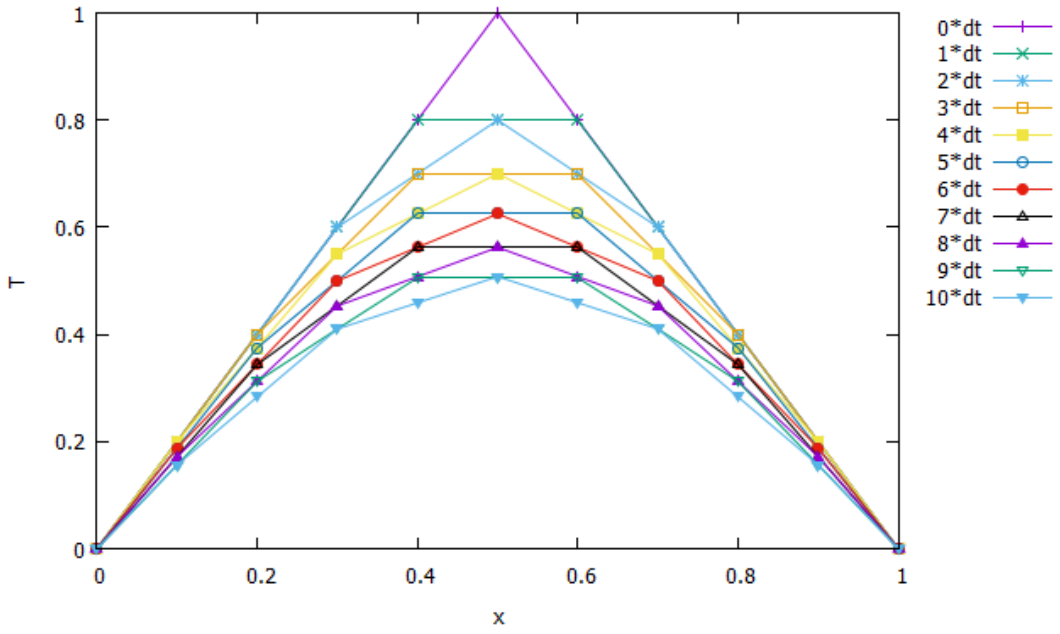


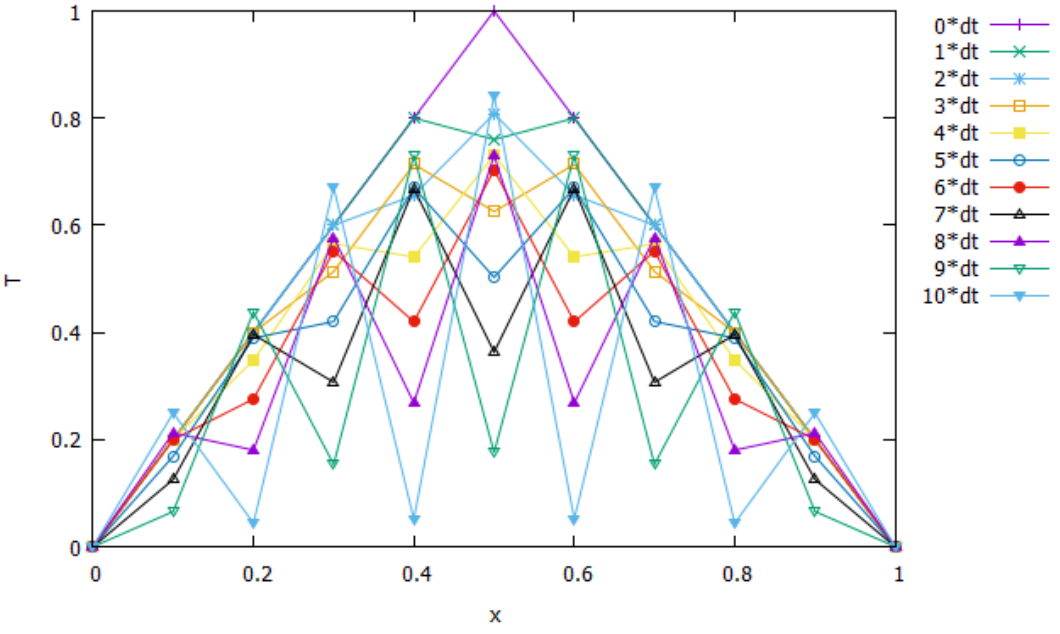
図 3: 表 1 をグラフに出力したもの

3.3 問題 3

問題の内容は、dt の値を変化させて計算させたとき、計算が破綻する dt の値を調べることである。この計算モデルの特徴から、dt の値を減少させればさせるほど精度が増していくため、今回は dt の値を増加させて検証していく。まずは、$dt = 0.001$ から dt を 0.001 ずつ増加させて計算していった。このとき、$dt = 0.006$ のとき初めて明らかに計算が破綻していた。(図 4b)



(a) $dt = 0.005$ の時の実行結果



(b) $dt = 0.006$ の時の実行結果

図 4: 0.001 ずつ増加させたときの実行結果

更に、$dt = 0.0051$ の場合を計算すると、以下の図 5 のようになった。

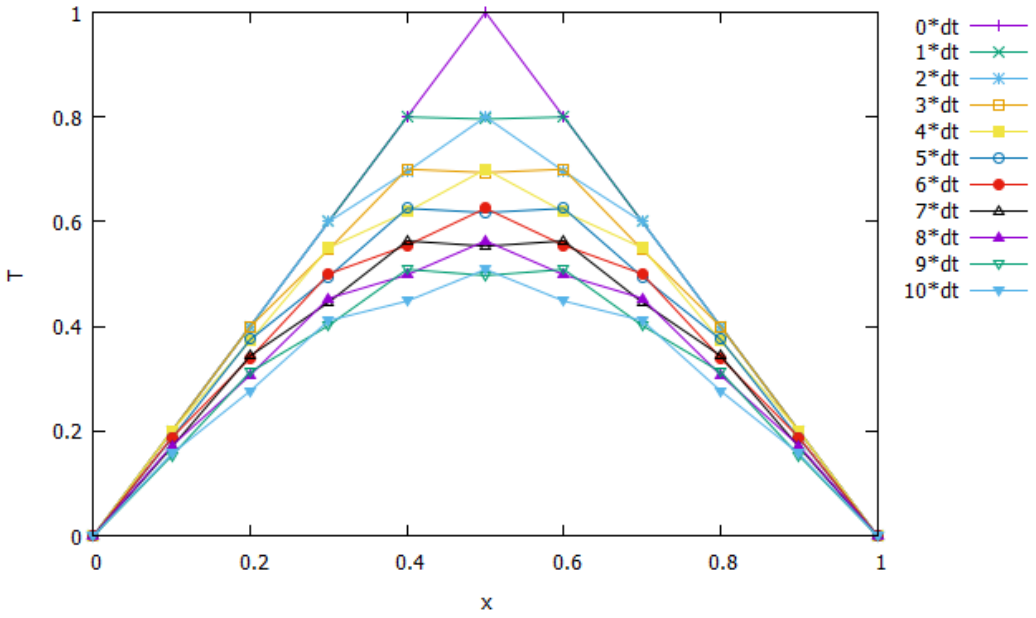
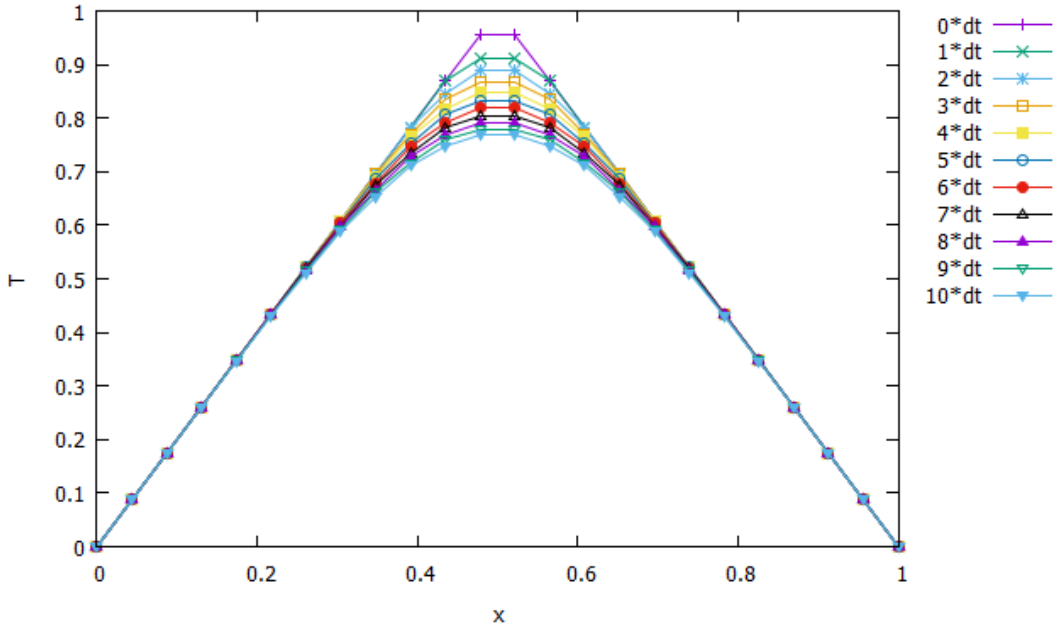


図 5: $dt = 0.0051$ の時の実行結果

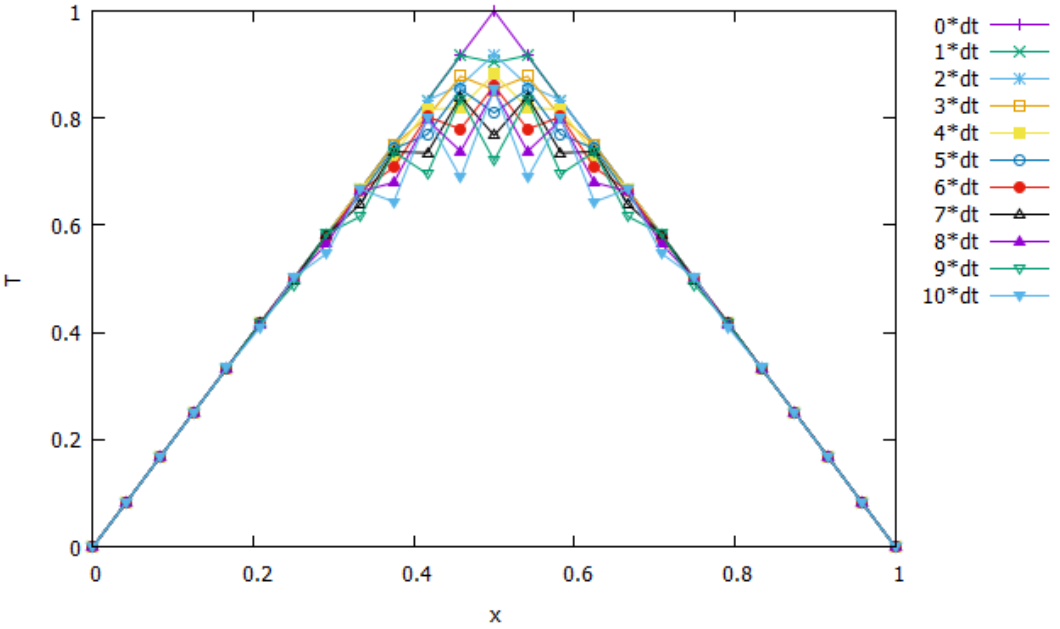
左端の代表点を第 0 番目としたときの、第 4 番目の代表点を見ると顕著であるが。$dt = 0.005$ の時の計算結果では、時間経過によって代表点の温度が同じであっても、それ以前より上昇することはなかった。しかし、それを少しでも超えるた場合、ある代表点では時間経過によって、それ以前の温度より上昇するという計算結果となった。このことから、計算が破綻する dt の値は $dt = 0.005$ より大きい値の時だとわかる。

3.4 問題 4

問題の内容は、分割数 n の値を変化させて計算させたとき、計算が破綻する n の値を調べることである。まずは、$n = 10$ から n を 1 ずつ減少させて計算していった。このときは計算は破綻しなかった。次に $n = 10$ から n を 1 ずつ増加させて計算していった。結果は、$n = 24$ のとき、計算の破綻が確認された。以下の図 6b がその時のグラフとなっている。



(a) $n = 23$ の時の実行結果



(b) $n = 24$ の時の実行結果

図 6: n を 1 ずつ増加させたときの実行結果

グラフの中央付近からもわかるように、時間が経過しているにもかかわらず温度が上昇するといった明らかな計算の破綻が見られている。

3.5　問題 5

問題の内容は、dt, n を共に変化させて計算を行い、計算が破綻しない条件を探すことである。

まず、n の値を 6 ~ 48 の範囲で変化させ、その中で更に dt を $dt = 0.0001$ から 0.0001 ずつ増やしつつ、計算が破綻する直前の dt の値を調べた。また、計算の破綻は、どこかの代表点が時間経過前の温度を超過した場合とする。それらの値を、n の値を横軸に、dt の値を縦軸に取ると以下の図**??**のようなグラフになった。

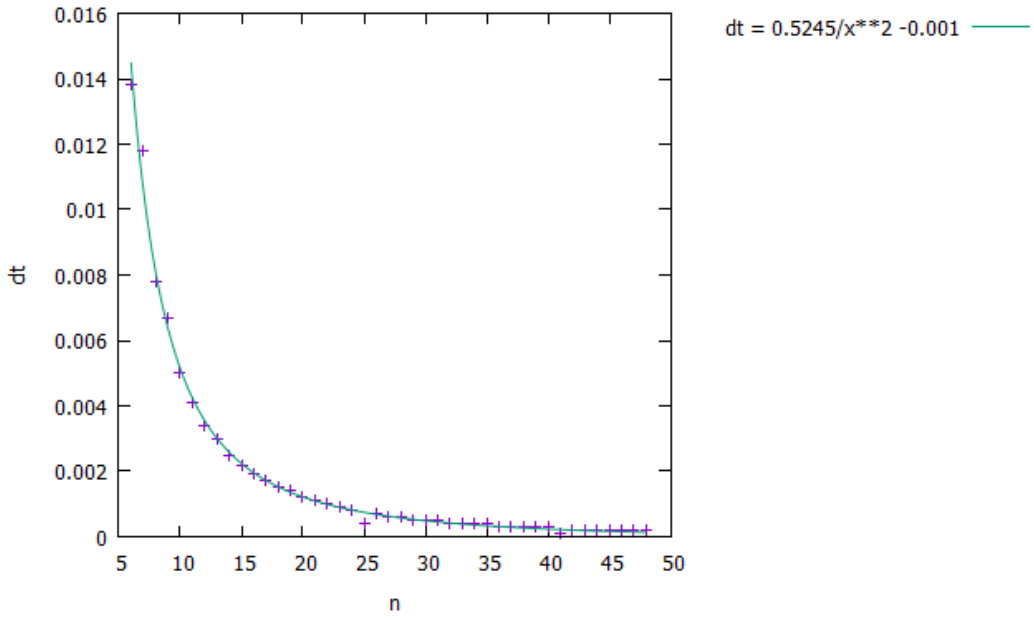


図 7: 各 n の値に対して、ちょうど計算が破綻しない dt の値とその近似曲線

この図にもあるように、ちょうど計算が破綻しない dt の値から近似曲線を取ると以下のような式になった。

$$dt = \frac{0.5245}{x^2} - 0.001 \tag{9}$$

このことから、解が安定して求められる条件は、dt の値が式**??**の右辺の値を超えなければ良いと分かる。

4　参考文献

- 配布されたテキスト