卒業研究要旨

Java を用いた 1 次元熱方程式の数値解析

大葉敏文 佐藤晴郎 (桂田研究室)

2001 年 2 月 9 日

1 差分法

1.1 Dirichlet Boundary Condition

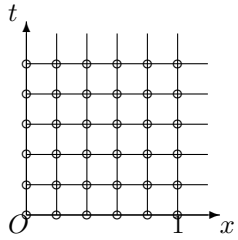
まずは Dirichlet Boundary Condition (以下 Boundary Condition を B.C. と略) の熱方程式の初期値境界値問題を差分法で考える。

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad (t > 0, 0 < x < 1) \tag{1}$$

$$u(0,t) = A, \quad u(1,t) = B \quad (t > 0) \tag{2}$$

$$u(x,0) = f(x) \quad (0 \le x \le 1) \tag{3}$$

解 $u = u(x,t)$ の定義域 $[0,1] \times [0,+\infty)$ を右図のような格子に切り、



各格子点 (x_i, t_j) における u の値

$$u_{i,j} \equiv u(x_i, t_j) \tag{4}$$

の近似値 $U_{i,j}$ を求める。x の区間 $[0,1]$ を N 等分して、その点を順番に x_0, x_1, x_2, …, x_N とする。

刻み幅 h を $h = 1/N$ とすると

$$x_i = ih \quad (i = 0, 1, 2, \cdots, N) \tag{5}$$

同様に時間変数 t に関する刻み幅を τ (> 0) として

$$t_j = j\tau \quad (j = 0, 1, 2, \cdots) \tag{6}$$

と置く。格子点において $\dfrac{\partial u}{\partial t}$ を前進差分近似、$\dfrac{\partial^2 u}{\partial x^2}$ を 2 階中心差分近似すると

$$\frac{\partial u}{\partial t}(x_i, t_j) = \frac{u_{i,j+1} - u_{i,j}}{\tau} + O(\tau) \quad (\tau \to 0) \tag{7}$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_j) = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2) \quad (h \to 0) \tag{8}$$

式 (1) を満たすことから τ, h を充分に小さくし、$\lambda = \tau/h^2$ と置き換え、u を U で近似すると

$$U_{i,j+1} - U_{i,j} = \lambda(U_{i+1,j} - 2U_{i,j} + U_{i-1,j}) \tag{9}$$

移項して

$$U_{i,j+1} = (1-2\lambda)U_{i,j} + \lambda(U_{i-1,j} + U_{i+1,j}) \quad (1 \leq i \leq N-1;\ j = 0, 1, \cdots) \tag{10}$$

また、式 (2),(3) から以下の二式が得られる

$$\begin{aligned} U_{0,j} &= A, \quad U_{N,j} = B \quad (j = 1, 2, \cdots) & (11) \\ U_{i,0} &= f(x_i) \quad (0 \leq i \leq N) & (12) \end{aligned}$$

ここまでの方法を陽解法と呼ぶのだが、陽解法では計算結果が満足行かない状況におちいってしまう場合がある (不安定現象)。陽解法においてこのような不安定現象を起こさない必要十分条件として λ が $0 < \lambda \leq 1/2$ を満たす必要があることがわかっている。

また λ の条件を緩和するための方法として計算法を陰解法に変えるという方法がある。陰解法では前進差分近似で得た式の変わりに後退差分近似を行うことで得られる式を用いる。陰解法を用いると λ が $0 < \lambda \leq 1/2$ の条件を満たさなくても安定した結果を得られる。

ここではもっと一般的に前進差分近似で得られた式と後退差分近似で得られた式を $1-\theta,\quad \theta \quad (0 \leq \theta \leq 1)$ の重みで重ねた公式を使おう。

$$\frac{U_{i,j+1} - U_{i,j}}{\tau} = (1-\theta)\frac{U_{i+1,j+1} - 2U_{i,j} + U_{i-1,j+1}}{h^2} + \theta\frac{U_{i+1,j+1} - 2U_{i,j+1} + U_{i-1,j+1}}{h^2} \tag{13}$$

これを移項して整理すると

$$(1+2\theta\lambda)U_{i,j+1} - \theta\lambda(U_{i+1,j+1} + U_{i-1,j+1}) = \{1-2(1-\theta)\lambda\}U_{i,j} + (1-\theta)\lambda(U_{i+1,j} + U_{i-1,j}) \tag{14}$$

この公式を用いる方法を θ 法と呼び、特に $\theta = 1/2$ の時をクランク - ニコルソン法とよぶ。（ちなみに $\theta = 0$ で陽解法、 $\theta = 1$ で陰解法になる。）

1.2 Neumann B.C.

Neumann B.C. の場合。境界条件である式 (2) が

$$u_x(0,t) = A, \quad u_x(1,t) = B \quad (t > 0) \tag{15}$$

と変更になることで、式 (11) が以下のように変更になる。

$$\frac{U_{1,j+1} - U_{0,j+1}}{h} = A \tag{16}$$

$$\frac{U_{N,j+1} - U_{N-1,j+1}}{h} = B \tag{17}$$

つまり

$$U_{0,j+1} = U_{1,j+1} - Ah, \quad U_{N,j+1} = U_{N-1,j+1} + Bh \tag{18}$$

これと式 (1) に θ 法を用いたものから $i = 1,\ i = N-1,\ 1 < i < N-1$ の 3 つの式が生成される。それぞれ順番に示すと

$$\begin{aligned} (1+\theta\lambda)U_{1,j+1} - \theta\lambda U_{2,j+1} &= [1-2(1-\theta)\lambda]U_{1,j} + (1-\theta)\lambda(U_{2,j} + U_{0,j}) - A\theta\lambda h & (19) \\ (1+\theta\lambda)U_{N-1,j+1} - \theta\lambda U_{N-2,j+1} &= [1-2(1-\theta)\lambda]U_{N-1,j} \\ &\quad +(1-\theta)\lambda(U_{N,j} + U_{N-2,j}) + B\theta\lambda h & (20) \\ (1+2\theta\lambda)U_{i,j+1} - \theta\lambda(U_{i+1,j+1} + U_{i-1,j+1}) &= [1-2(1-\theta)\lambda]U_{i,j} \quad + (1-\theta)\lambda(U_{i+1,j} + U_{i-1,j}) & (21) \\ (j = 0, 1, 2, \cdots \quad , \quad i = 1, 2, \cdots, N-1) & \end{aligned}$$

1.3 連立方程式を解くにあたって

1.1, 1.2 で導入した差分方程式は連立 1 次方程式であるが、その係数行列は三重対角行列 (対角線とその両隣にのみ非零成分がある) になり、 Gauss の消去法ではその性質をうまく利用して、非常に効率的に解を求めることができる。

2 Java によるシミュレーションの意義

2.1 Java 言語の特徴

Java は 1991 年に米国 Sun Microsystems 社によって作成されたオブジェクト指向プログラミング言語であり、近年のインターネットの急速な普及と共に注目されている。本来 Java は家庭電化製品用のソフトウェア環境を構築するために研究された言語であった。しかし、 Java を使用するとプラットホームに依存せず、様々なハードウェア / ソフトウェア環境で動作するプログラムを作成することができるため、現在では多種多様な業界において重要なアプリケーションを開発するのに使われている。

また、 Java 言語には以下のような優れた特徴がある。

- Java 言語はオブジェクト指向であるためコードを大幅に再利用することがきる。
- 例外処理機能が用意されており、実行時に発生する問題を整然とした方法で処理することができる。
- ガーベッジコレクション機能を装備し使用しなくなったメモリーソースを自動的に再利用できる。
- C++ の文法を簡略化したものであり、 Java 言語の構文は非常にシンプルに構成されている。
- マルチスレッドプログラミングがサポートされている。
- クラスライブラリに豊富な機能が用意されている。

これらの機能により、 Java ではプログラムをとても簡単に作成する事ができるようになっている。

2.2 Java におけるシミュレーション

シミュレーションとは現実世界の現象をコンピュータで表現し、数値計算を用いその現象に関連する結果を推定する分野である。オブジェクト指向の出発点も現実世界をコンピュータ上に直接表現することであり、シミュレーションとオブジェクト指向の考え方は非常に類似している。そのため、 Java などのオブジェクト指向言語により数値計算を行うのは自然なことである。また、 Java のクラスライブリーに備え付けられた機能を用いれば GUI(グラフィック・ユーザー・インターフェース) を容易に作成することができる。

Java においてシミュレーションを行うことには次のようなメリットがある。

1. Portability(移植可能性)
   従来グラフィックスを利用するプログラムは各々のシステムに依存した形になっていたため、あるシステムで作成したグラフィックスを別のシステムで動かすことができなかった。しかし、 Java は基本的にインタプリタ言語であるので、グラフィックスも含めて、 ”Write once, run anywhere” (一度作成してしまえばどこでも実行できる) という利点がある。

2. GUI の利便性
   Java ではアプレット (Web ブラウザ又はアプレットビューアで実行されるプログラム) で図形やイメージを表示する。アプレットの GUI にはボタン、チェックボックス、リスト、テキストエリアやテキストフィールドなど様々なコンポーネントを入れることができる。ユーザーはこれらを介しアプレットと対話することができ、また、マウスやキーボードを使用して入力することもできる。
   Java による GUI のプログラミングは従来のものに比べて簡易化された上、やはり一度書いてしまえば他のシステムでも実行できるようになっている。

さらに、 Java ではアニメーションを書くことや音声対応の標準化も進んでいるため、マルティメディア教材をも扱うこともできる。なお、 Java は開発途上の言語であり、今後は操作性や即応性などの点で改善されることが期待されている。

3 Java による数値解析実験

実際に Java のアプレットを作り、熱方程式を差分法にて数値解析してみる。アプレットは前記の通り、 Web ブラウザ上でアクセスし実行することができるので、以下に URL を記しておく。

http://www.math.meiji.ac.jp/~ee78038/9745/soturon/heat1d.html