

“私にとっての”
反応拡散系数値シミュレーション入門
基礎編¹

2002 年度横浜市立大学集中講義ノート + 2005 年度金沢大学特別セミナー
+ 2008 年度京都大学 GCOE セミナー講義ノート + 2009 年度さきがけ数学塾 + CD-ROM

長山 雅晴

北海道大学電子科学研究所 教授

E-mail : nagayama@es.hokudai.ac.jp

¹2012 年 08 月 10 日：多少のミスを見つけたので修正した。2013 年 03 月 06 日：動画作成部分を更新した。

目次

第 1 章	はじめに	7
1.1	数値シミュレーションとは	7
1.2	数値シミュレーションプログラムのあらすじ	8
1.3	入門編の目標	8
第 2 章	常微分方程式の数値計算法	11
2.1	Euler 法 [9, 10, 11]	11
2.2	Runge-Kutta 法 [10]	11
2.3	問題	13
2.3.1	問題 1	13
2.3.2	問題 2	14
2.3.3	問題 3	14
第 3 章	拡散方程式の数値計算法 (差分法)	15
3.1	熱方程式 (拡散方程式) の導出 [3]	15
3.2	1 次元初期境界値問題 [3, 4]	16
3.2.1	陽解法	17
3.2.2	陰解法	18
3.2.3	反復改良法 [6]	21
3.2.4	問題	22
3.3	2 次元長方形領域における拡散方程式の数値計算	23
3.3.1	陽解法 [4]	23
3.3.2	陰解法 [4]	23
3.3.3	ADI 法 [5]	24
3.3.4	2 次元 SSI 法 [5]	24
3.4	3 次元直方体領域での数値計算法	25
3.4.1	陽解法	25
3.4.2	ADI 法 [5]	25
3.4.3	陰解法	26
第 4 章	反応拡散方程式の数値計算法 (差分法)	27
4.1	1 次元反応拡散系の数値計算法	27
4.1.1	1 次元反応拡散系の初期・境界値問題	27
4.1.2	2 変数反応拡散系の数値計算法 (半陰解法)	28
4.1.3	差分方程式の連立 1 次方程式表示	31
4.2	2 次元反応拡散系の数値計算法	35
4.2.1	方程式の離散化	35
4.2.2	非線形項の離散化	36
4.2.3	初期条件の離散化	36
4.2.4	境界条件の離散化	36

4.2.5	ADI 法	37
第 5 章	可視化	43
5.1	数値データの可視化	43
5.2	必要なソフトウェアのインストール	43
5.2.1	Mac OS X の計算機環境を作る	44
5.2.2	ImageMagick のインストール	45
5.2.3	ffmpeg のインストール	46
5.2.4	mpeg_encode のインストール	46
5.2.5	mpeg_play のインストール	47
5.3	必要になるかもしれないソフトウェアのインストール	47
5.3.1	OpenGL のインストール	47
5.3.2	GLSC のインストール	48
5.3.3	vlc のインストール	48
5.4	GNUPLOT による数値計算結果の可視化 [15]	49
5.4.1	プリンターへの出力	49
5.4.2	GNUPLOT を使ってパラパラ動画を作ろう	50
5.4.3	GNUPLOT を使って JPEG 形式の画像ファイルを作る	51
5.4.4	GNUPLOT を使って GIF 形式の画像ファイルを作る	52
5.4.5	GNUPLOT を使って PNG 形式の画像ファイルを作る	53
5.5	GNUPLOT と ImageMagick による動画の作成	54
5.5.1	アニメーション GIF の作成	54
5.5.2	MPEG 動画の作成	55
5.5.3	動画作成 その 1 (秋山さん作成)	57
5.5.4	動画作成 (MPEG1) その 2	59
5.5.5	動画作成 (MPEG1) その 3	60
5.6	GNUPLOT を C 言語から呼び出す [16, 17]	61
5.7	GNUPLOT を用いた 2 次元データの可視化	63
5.8	OpenGL によるオフスクリーンレンダリング	64
5.9	可視化の参考プログラム	66
5.9.1	GLSC による鳥瞰図の参考プログラム	66
5.9.2	OSMesa(OpenGL) のプログラム	68
5.9.3	おまけ (VNC を使った画面共有の方法)	80
5.9.4	メモ書き	82
付録 A	数値スキームの安定性	83
A.1	拡散方程式の数値スキームの安定性解析 その 1	83
A.1.1	陽解法 (Explicit Scheme) の安定性解析	83
A.1.2	完全陰解法 (Implicit Scheme) の安定性解析	83
A.2	拡散方程式の数値スキームの安定性解析 その 2	84
A.2.1	差分方程式の行列表示	84
A.2.2	差分方程式の安定性	85
A.3	差分スキームの収束性	89

付録 B 発展編の完成部分	91
B.1 軸対称問題の数値計算法 (有限体積近似)	91
B.2 2次元円板領域での数値計算法	93
B.3 円柱領域での数値計算法	94
B.4 移流拡散方程式の数値計算法	95
B.4.1 1次の風上差分	95
B.4.2 3次の風上差分	95
B.5 非線形拡散方程式の離散化 (差分法)	96

第1章 はじめに

1.1 数値シミュレーションとは

与えられたモデル方程式(ここでは常微分方程式系や偏微分方程式系)の解挙動を調べることは、モデル方程式の妥当性や新しい現象を発見するために重要です。数理モデルの妥当性等を調べるためには解の遷移過程やダイナミックな変化を捉えることが必要となり、**数値シミュレーション**という手段を必要とします。数値シミュレーションはただ数値計算をすることではなく次の三段階から成り立っています [8]。

- (1) モデル方程式に対する理解：方程式の持つ1つ1つの項の意味をしっかりと理解する
- (2) モデル方程式を数値的に解く：離散化と数値解法
- (3) 得られた結果の評価：モデルの妥当性や解法の妥当性を含む^a。

^a数値計算結果の成否は何で判断すべきかなのか?この問題に対する明確な回答はまだないと思われる。一つの回答は、解析解が求まる方程式を対象として、数値解と理論解を定性的かつ定量的に比較することではないか。

(2)と(3)を繰り返し行うことで、「直感的に微分方程式の解挙動が理解できるようになる(モデル方程式に対する理解が深まる)¹」

実際の数値計算ではパラメータを動かして解のダイナミクスを調べるのだが、このときモデルの妥当性や数値計算結果に対する理解が重要となる。このような数値シミュレーションを行う上で何が重要なのか?その答えは難しいがあえて答えるとするならば、次のようなことではないだろうか²：

- (1) 非線形現象を多く知ること、数値解に対して物理的理解(解釈)が出来ること。
- (2) 数値計算法の特徴を知ること(理論+実践)
- (3) 数学的な理解(例えば、分岐理論的理解)

このような数値シミュレーションの基本的な訓練方法はまだ確立されていないと思うが、あえて言うなら次のように考えたらいいのではないか³：

¹この解説書では(2)の方法を解説する。

²私自身はモデル方程式の数値計算を繰り返し行うことで、その方程式の特徴を理解しようとする、そして説明は出来ないけども、パラメータをどう動かしたら目的の解を得ることができるのかがわかってくる。しかしこれでは数値解の現象が説明できないけど…

³長山提案ですが、数値計算をしているときに師匠にいつも言われていたことと同じです。

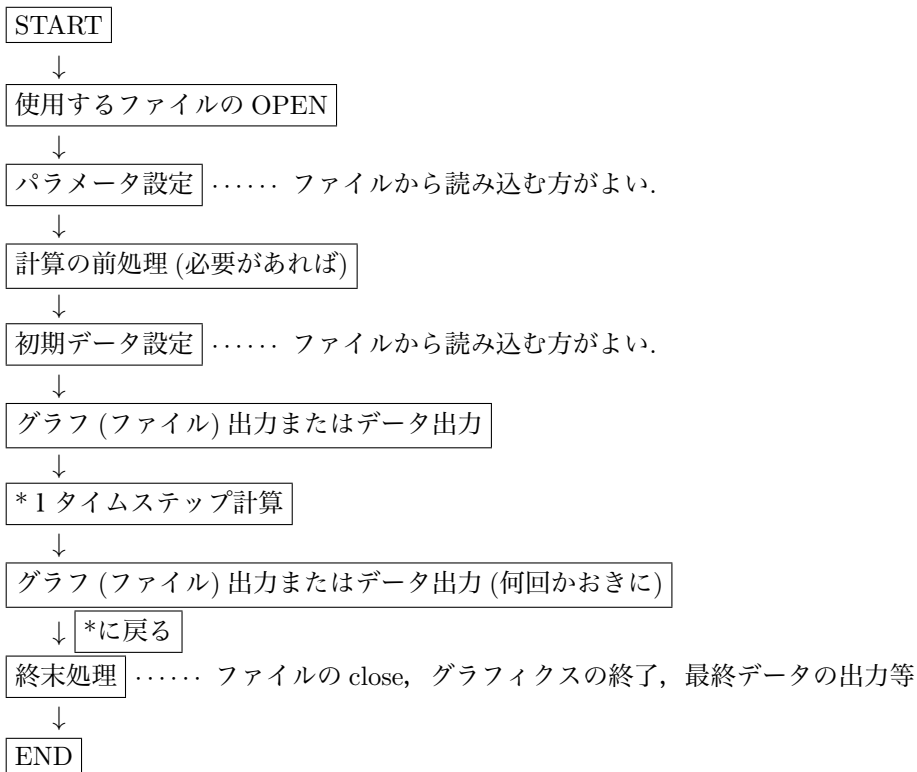
- (1) 与えられたモデル方程式に対して出現するであろう数値解の直観的な説明を考える。(当然ではあるが、理論的にわかることは最初にチェックすべきである。)
- (2) 数値計算によって得られた結果と比較する。
- (3) 直観と異なっていた場合、最初に数値解法を確認する。数値解法が正しい場合は、数値計算結果から直感的説明をつける^a。
- (4) その説明を他人に聞いてもらう (これが結構重要)^b。

^a力学系の理解や分岐理論的理解を用いることも重要

^bこのことによって自分の考察の問題点が明らかになる場合が多い。

1.2 数値シミュレーションプログラムのあらすじ

数値計算プログラムは大体次のようなあらすじで書くことができます。



注意: 後で自分の作ったプログラムが何をしているかどうか分かる程度には、面倒でもプログラムにドキュメント書いておくこと。後から何をしているのかわからなくなります。

1.3 入門編の目標

入門編では次のような反応拡散系を数値計算するための基本的な数値計算法とその可視化方法について解説する:

$$\frac{\partial \mathbf{u}}{\partial t} = D\Delta \mathbf{u} + \mathbf{f}(\mathbf{u}). \quad (1.1)$$

ただし、D は非負対角行列とする。(1.1) に含まれるモデル方程式としては次のような方程式が挙げられる:

神経繊維上の電位の伝播モデル (FitzHugh-Nagumo 方程式)[27]

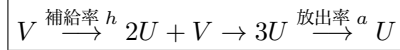
$$\begin{cases} \frac{\partial u}{\partial t} = du_{xx} + \frac{1}{\varepsilon} (u(1-u)(u-a) - v), \\ \frac{\partial v}{\partial t} = u - \gamma v. \end{cases} \quad (1.2)$$

ただし, $0 < a < 1/2$ であり, u は電位を表し, v はイオンチャネルの開き具合を表している.

BZ 反応モデル [25, 26, 27]

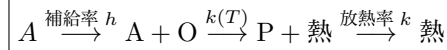
$$\begin{cases} \frac{\partial u}{\partial t} = d_u \Delta u + \frac{1}{\varepsilon} \left(u(1-u) - fw \frac{u-q}{u+q} \right), \\ \frac{\partial w}{\partial t} = d_w \Delta w + u - \gamma w. \end{cases} \quad (1.3)$$

Gray-Scott モデル [26]



$$\begin{cases} \frac{\partial u}{\partial t} = d_u \Delta u - ku + vu^2, \\ \frac{\partial v}{\partial t} = d_v \Delta v + h(1-v) - vu^2. \end{cases} \quad (1.4)$$

発熱反応モデル



$$\begin{cases} \frac{\partial T}{\partial t} = d_T \Delta T + \frac{1}{\varepsilon} \left(-kT + a \exp\left(\frac{T}{1+T/c}\right) \right), \\ \frac{\partial a}{\partial t} = d_a \Delta a + h(a_0 - a) - a \exp\left(\frac{T}{1+T/c}\right). \end{cases} \quad (1.5)$$

ここで, T は温度, a は反応物質を表している.

これらの方程式に対して, 反応項に対する数値計算

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}) \quad (1.6)$$

と拡散項に対する数値計算

$$\frac{\partial u}{\partial t} = d \Delta u \quad (1.7)$$

の二つの部分に分けて解説する.

反応拡散方程式の数値計算は, 反応項の性質を調べることから始まるので, 常微分方程式系の数値計算法を知っておくことは必要である. また, 拡散方程式の数値計算ができれば, 反応拡散系の数値計算は一応可能となる.

第2章 常微分方程式の数値計算法

反応項に対する数値計算法を解説する。ここでは次のような非自励系を含む一般の常微分方程式系に対する初期値問題の数値解法を解説する。

$$\begin{cases} \frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}), \\ \mathbf{u}(t_0) = \mathbf{u}_0. \end{cases} \quad (2.1)$$

ただし, $\mathbf{u} : \mathbf{R} \rightarrow \mathbf{R}^N, \mathbf{f} : \mathbf{R}^N \times \mathbf{R}_+ \rightarrow \mathbf{R}^N$ である。時間差分を Δt とし, $t_n = t_0 + n \times \Delta t$,

$$\mathbf{u}(t_n) = \mathbf{u}_n, \quad (2.2)$$

と書く。数値計算法を説明するために関数 $\mathbf{f}(x)$ を $x = x_0$ の周りでのテイラー展開を書いておく。

$$\mathbf{f}(x) = \mathbf{f}(x_0) + \frac{d}{dx} \mathbf{f}(x_0)(x - x_0) + \frac{1}{2} \frac{d^2}{dx^2} \mathbf{f}(x_0)(x - x_0)^2 + \frac{1}{3!} \frac{d^3}{dx^3} \mathbf{f}(x_0)(x - x_0)^3 + O((x - x_0)^4). \quad (2.3)$$

2.1 Euler 法 [9, 10, 11]

Euler 法は最も単純な計算方法であり, テイラー展開を用いて公式を導く。

Euler 法

$$\mathbf{u}_0 = \mathbf{u}(t_0), \quad (2.4)$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t \mathbf{f}(t_n, \mathbf{u}_n). \quad (2.5)$$

このスキームは $O(\Delta t)$ の誤差を持つ。

簡単な証明

$\mathbf{u}(t + \Delta t)$ を t のまわりでテイラー展開すると¹

$$\mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \frac{d}{dt} \mathbf{u}(t) \Delta t + \frac{1}{2} \frac{d^2}{dt^2} \mathbf{u}(t) (\Delta t)^2 + O(\Delta t^3).$$

故に

$$\frac{d}{dt} \mathbf{u}(t) = \frac{\mathbf{u}(t + \Delta t) - \mathbf{u}(t)}{\Delta t} - \frac{1}{2} \frac{d^2}{dt^2} \mathbf{u}(t) \Delta t + O(\Delta t^2). \quad (2.6)$$

2.2 Runge-Kutta 法 [10]

テイラー展開法とは異なる, \mathbf{u} の導関数を使わない方法。次のように導出する:

$$\mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \Delta t \Phi(t, \mathbf{u}(t)) \quad (2.7)$$

という形式を仮定する。このとき

$$\Phi(t, \mathbf{u}(t)) = \alpha k_1 + \beta k_2, \quad (2.8)$$

$$k_1 = \mathbf{f}(t, \mathbf{u}(t)), \quad (2.9)$$

$$k_2 = \mathbf{f}(t + p\Delta t, \mathbf{u}(t) + q\Delta t k_1) \quad (2.10)$$

¹(2.3) において $x = t + \Delta t, x_0 = t$ して代入すると求めることができる。

である。ここで (2.10) の右辺を t の周りでテーラー展開すると

$$k_2 = \mathbf{f}(t, \mathbf{u}(t)) + \frac{\partial}{\partial t} \mathbf{f}(t, \mathbf{u}(t)) p \Delta t + D_u \mathbf{f}(t, \mathbf{u}(t)) k_1 q \Delta t + O(\Delta t^2).$$

ただし、 $D_u \mathbf{f}(t, \mathbf{u}(t)) = \left(\frac{\partial f_i}{\partial u_j} \right)_{1 \leq i, j \leq N}$ である²。(2.9) より

$$k_2 = \mathbf{f}(t, \mathbf{u}(t)) + \frac{\partial}{\partial t} \mathbf{f}(t, \mathbf{u}(t)) p \Delta t + D_u \mathbf{f}(t, \mathbf{u}(t)) \mathbf{f}(t, \mathbf{u}(t)) q \Delta t + O(\Delta t^2). \quad (2.11)$$

(2.11) を (2.8) に代入して (2.7) を計算すると

$$\begin{aligned} \mathbf{u}(t + \Delta t) &= \mathbf{u}(t) + \Delta t (\alpha \mathbf{f}(t, \mathbf{u}(t)) + \beta (\mathbf{f}(t, \mathbf{u}(t)) \\ &\quad + p \Delta t \frac{\partial}{\partial t} \mathbf{f}(t, \mathbf{u}(t)) + D_u \mathbf{f}(t, \mathbf{u}(t)) \mathbf{f}(t, \mathbf{u}(t)) q \Delta t)) + O(\Delta t^3). \\ &= \mathbf{u}(t) + \Delta t (\alpha + \beta) \mathbf{f}(t, \mathbf{u}(t)) \\ &\quad + \Delta t^2 \beta (p \frac{\partial}{\partial t} \mathbf{f}(t, \mathbf{u}(t)) + q D_u \mathbf{f}(t, \mathbf{u}(t)) \mathbf{f}(t, \mathbf{u}(t))) + O(\Delta t^3). \end{aligned} \quad (2.12)$$

今、 $\mathbf{u}(t + \Delta t)$ を t の周りでテーラー展開すると

$$\mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \Delta t (\mathbf{f}(t, \mathbf{u}(t))) + \frac{\Delta t^2}{2} \frac{d^2}{dt^2} \mathbf{u} \quad (2.13)$$

となる。ところで

$$\begin{aligned} \frac{d}{dt} \mathbf{u} &= \mathbf{f}(t, \mathbf{u}), \\ \frac{d^2}{dt^2} \mathbf{u} &= \frac{d}{dt} \mathbf{f} + D_u \mathbf{f} \frac{d}{dt} \mathbf{u}, \\ &= \frac{d}{dt} \mathbf{f} + D_u \mathbf{f} \mathbf{f} \end{aligned} \quad (2.14)$$

となるので、(2.14) を (2.13) に代入すると

$$\mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \Delta t (\mathbf{f}(t, \mathbf{u}(t))) + \frac{\Delta t^2}{2} \left(\frac{d}{dt} \mathbf{f} + D_u \mathbf{f} \mathbf{f} \right) \quad (2.15)$$

と書ける。(2.12), (2.15) の係数比較から

$$\alpha + \beta = 1, \quad \beta p = \frac{1}{2}, \quad \beta q = \frac{1}{2}$$

が得られる。ここで

$$\alpha = 0, \quad \beta = 1, \quad p = \frac{1}{2}, \quad q = \frac{1}{2}$$

とおくと、次の原型版修正 Euler 法が得られる。

修正 Euler 法 (原型版)

$$\mathbf{u}_0 = \mathbf{u}(t_0), \quad (2.16)$$

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{u}_n), \quad (2.17)$$

$$\mathbf{k}_2 = \mathbf{f}\left(t_n + \frac{\Delta t}{2}, \mathbf{u}_n + \frac{\Delta t}{2} \mathbf{k}_1\right), \quad (2.18)$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t \mathbf{k}_2. \quad (2.19)$$

通常は上記の方法から変形した次の通常版修正 Euler が使われる。

²要するにヤコビ行列ということです。

修正 Euler 法 (通常版)

$$\mathbf{u}_0 = \mathbf{u}(t_0), \quad (2.20)$$

$$\mathbf{k} = \mathbf{u}_n + \frac{\Delta t}{2} \mathbf{f}(t_n, \mathbf{u}_n), \quad (2.21)$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t \mathbf{f}(t_n + \frac{\Delta t}{2}, \mathbf{k}). \quad (2.22)$$

このスキームは $O(\Delta t^2)$ の誤差を持つ。また、

$$\alpha = \beta = \frac{1}{2}, \quad p = q = 1 \quad (2.23)$$

とおくと、Heun 法と呼ばれる。

Heun 法

$$\mathbf{u}_0 = \mathbf{u}(t_0), \quad (2.24)$$

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{u}_n, t_n), \quad (2.25)$$

$$\mathbf{k}_2 = \mathbf{f}(t_n + \Delta t, \mathbf{u}_n + \Delta t \mathbf{k}_1), \quad (2.26)$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{\Delta t}{2} (\mathbf{k}_1 + \mathbf{k}_2). \quad (2.27)$$

次に示す Runge-Kutta 法は $O(\Delta t^4)$ の誤差を持つ：

4 次の Runge-Kutta 法

$$\mathbf{u}_0 = \mathbf{u}(t_0), \quad (2.28)$$

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{u}_n), \quad (2.29)$$

$$\mathbf{k}_2 = \mathbf{f}(t_n + \frac{1}{2}\Delta t, \mathbf{u}_n + \frac{1}{2}\Delta t \mathbf{k}_1), \quad (2.30)$$

$$\mathbf{k}_3 = \mathbf{f}(t_n + \frac{1}{2}\Delta t, \mathbf{u}_n + \frac{1}{2}\Delta t \mathbf{k}_2), \quad (2.31)$$

$$\mathbf{k}_4 = \mathbf{f}(t_n + \Delta t, \mathbf{u}_n + \Delta t \mathbf{k}_3), \quad (2.32)$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{\Delta t}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \quad (2.33)$$

この数値計算方法の導出は [11] に詳しく解説されている。

常微分方程式の数値計算には通常 4 次の Runge-Kutta 法を用いる。

2.3 問題

2.3.1 問題 1

例題 1 (ロジスティック方程式)[27]

$$\frac{du}{dt} = au(1-u), \quad u(0) = u_0. \quad (2.34)$$

ただし、 a は正定数。

(1.1.1) $u_0 > 0$ の初期値に対して Euler 法や Runge-Kutta 法で数値計算しなさい。得られた解を (t, u) 平面でグラフ表示しなさい³。

(1.1.2) Euler 法で数値計算したとき、 Δt の大きさを変化させて (2.34) の解を求めなさい。

2.3.2 問題2

問題2 (興奮現象モデル)

$$\begin{cases} \frac{du}{dt} = \frac{1}{\varepsilon}(u(1-u)(u-a) - v) \equiv \frac{1}{\varepsilon}f(u, v), \\ \frac{dv}{dt} = u - \gamma v \equiv g(u, v), \\ (u, v)(0) = (u_0, v_0). \end{cases} \quad t > 0, \quad (2.35)$$

ただし、 $a = 0.2, \varepsilon = 0.001$ とする。

(1.2.1) γ を適当に固定して $f(u, v) = 0, g(u, v) = 0$ を (u, v) 平面に表示しなさい。

(1.2.2) 平衡解 $f(\bar{u}, \bar{v}) = 0, g(\bar{u}, \bar{v}) = 0$ の線形化安定性を調べよ。

(1.2.3) 初期値を $(u_0, v_0) = (0.18, 0.0)$ と $(u_0, v_0) = (0.22, 0.0)$ と置いたときの解軌道を (u, v) 平面で表示しなさい。

2.3.3 問題3

問題3 (2体問題の微分方程式, 惑星や人工衛星の軌道を求める微分方程式)[7]

$$\begin{cases} \frac{d^2x}{dt^2} = -\frac{x}{(x^2 + y^2)^{3/2}}, \\ \frac{d^2y}{dt^2} = -\frac{y}{(x^2 + y^2)^{3/2}}, \\ x(0) = 1 - e, \quad \dot{x}(0) = 0, \quad y(0) = 0, \quad \dot{y}(0) = \sqrt{(1+e)/(1-e)}, \quad e = 0.9. \end{cases} \quad t > 0, \quad (2.36)$$

e が 1 に近いと数値的に解き難いことが知られている。

(1.3.1) Euler 法, 修正 Euler 法, Runge-Kutta 法で (2.36) を解き, (x, y) 平面に表示しなさい。

³表示方法およびプリンターへの出力方法は第 5.4 節参照のこと。

第3章 拡散方程式の数値計算法（差分法）

「拡散方程式の数値計算は行列計算」と言っても過言ではありません.. いかにも速く精度よく連立一次方程式を計算することが重要になってきます.

3.1 熱方程式（拡散方程式）の導出 [3]

長さ L の細長い針金を考える. 断面積 Ω が十分小さいときは1次元として近似できる. 区間 $[0, L]$ 上の任意の場所 x , 時刻 t における温度を $u(t, x)$ とする. 針金の密度 $\rho(x)$, 比熱 $c(x)$, 熱伝導係数 $k(x)$ とする. 任意の場所 x , 時刻 t における熱量を $q(t, x)$ とする. 熱方程式は次のフーリエ則に基づいて導出される:

フーリエ則 (Fourier law) (実験)

単位時間当りの熱量の流出はその場所 (x) での温度勾配に比例

$$\frac{dq}{dt}(t, x) \propto \frac{\partial u}{\partial x}(t, x).$$

すなわち

$$\frac{\partial q}{\partial t}(t, x) = -k(x) \frac{\partial u}{\partial x}(t, x).^a$$

^a説明重要

$(0, L)$ の内の任意の区間 (a, b) での総熱量は

$$Q(t) = \int_a^b c(x)\rho(x)u(t, x)dx$$

となる. このとき単位時間当りの総熱量の変化は

$$\frac{dQ}{dt}(t) = \int_a^b c(x)\rho(x) \frac{\partial u}{\partial t}(t, x)dx \quad (3.1)$$

と書ける. ところで熱量の変化は, 「両端 a, b を通してどれだけ熱量が流入流出したか」で決まるので

$$\begin{aligned} \frac{dQ}{dt}(t) &= K(b) \frac{\partial u}{\partial x}(t, b) - K(a) \frac{\partial u}{\partial x}(t, a)^a \\ &= \int_a^b \frac{\partial}{\partial x} \left(K(x) \frac{\partial u}{\partial x}(t, x) \right) dx. \end{aligned} \quad (3.2)$$

^aこの符号の意味をしっかりと説明する

となる. (3.1) と (3.2) から

$$\int_a^b c(x)\rho(x) \frac{\partial u}{\partial t}(t, x)dx = \int_a^b \frac{\partial}{\partial x} \left(K(x) \frac{\partial u}{\partial x}(t, x) \right) dx$$

故に

$$c(x)\rho(x) \frac{\partial u}{\partial t}(t, x) = \frac{\partial}{\partial x} \left(K(x) \frac{\partial u}{\partial x}(t, x) \right).$$

c, ρ, K が定数ならば

$$\frac{\partial u}{\partial t} = \frac{K}{c\rho} \frac{\partial^2 u}{\partial x^2}$$

となる.

3.2 1次元初期境界値問題 [3, 4]

$$\frac{\partial u}{\partial t} = d \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < L, \quad t > 0 \quad (3.3)$$

ただし, d は拡散係数であり, 正定数である.

初期条件¹ :

$$u(x, 0) = u_0(x). \quad (3.4)$$

Dirichlet 境界条件

$$u(0, t) = \alpha, \quad u(L, t) = \beta. \quad (3.5)$$

Neumann 境界条件

$$\frac{\partial}{\partial x} u(t, 0) = \alpha, \quad \frac{\partial}{\partial x} u(t, L) = \beta. \quad (3.6)$$

周期境界条件

$$u(t, 0) = u(t, L), \quad \frac{\partial u}{\partial x}(t, 0) = \frac{\partial u}{\partial x}(t, L). \quad (3.7)$$

方程式の離散化

$$\frac{\partial u}{\partial t} = \frac{u^{n+1} - u^n}{\Delta t} + O(\Delta t). \quad (3.8)$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + O(\Delta x^2). \quad (3.9)$$

(2.5) から時間離散化誤差は $O(\Delta t)$ である. 空間離散化誤差は $O(\Delta x^2)$ である.

証明

$u(x + \Delta x)$ および $u(x - \Delta x)$ を x のまわりでテーラー展開する² :

$$u(x + \Delta x) = u(x) + u_x(x)\Delta x + \frac{1}{2}u_{xx}(x)\Delta x^2 + \frac{1}{6}u_{xxx}(x)\Delta x^3 + \frac{1}{24}u_{xxxx}(x)\Delta x^4 + O(\Delta x^5). \quad (3.10)$$

$$u(x - \Delta x) = u(x) - u_x(x)\Delta x + \frac{1}{2}u_{xx}(x)\Delta x^2 - \frac{1}{6}u_{xxx}(x)\Delta x^3 + \frac{1}{24}u_{xxxx}(x)\Delta x^4 + O(\Delta x^5). \quad (3.11)$$

(3.10)+(3.11) より

$$u(x + \Delta x) + u(x - \Delta x) = 2u(x) + u_{xx}(x)\Delta x^2 + \frac{1}{12}u_{xxxx}(x)\Delta x^4 + O(\Delta x^5). \quad (3.12)$$

$$u_{xx}(x) = \frac{u(x + \Delta x) - 2u(x) + u(x - \Delta x)}{\Delta x^2} - \frac{1}{12}u_{xxxx}(x)\Delta x^2 + O(\Delta x^3). \quad (3.13)$$

従って, 離散化誤差は $O(\Delta x^2)$ となる.

¹ $u_0(x)$ は連続関数であれば特に問題ない

²(2.3) において $x = x + \Delta x, x_0 = x$ して代入すると求めることができる.

境界条件の離散化

$$\text{Dirichlet 境界条件} \quad u_0^n = \alpha, \quad u_N^n = \beta. \quad (3.14)$$

$$\text{Neumann 境界条件}^a \quad u_1^n - u_{-1}^n = 2\Delta x\alpha, \quad u_{N+1}^n - u_{N-1}^n = 2\Delta x\beta. \quad (3.15)$$

$$\text{周期境界条件} \quad u_0^n = u_N^n, \quad u_{-1}^n = u_{N-1}^n. \quad (3.16)$$

^a中心差分することによって、境界での差分誤差を $O(\Delta x^2)$ とすることができる。

3.2.1 陽解法

陽解法公式

$$u_i^{n+1} = dr u_{i-1}^n + (1-2dr)u_i^n + dr u_{i+1}^n, \quad 0 \leq i \leq N. \quad (3.17)$$

ただし、 $r = \Delta t / \Delta x^2$ である。

数値スキームの安定性 (A.1) より

$$\lambda_k = 1 - 4dr \sin^2\left(\frac{k\Delta x\pi}{2}\right) \quad (3.18)$$

から

$$|\lambda_k| \leq 1 \Leftrightarrow d \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}. \quad (3.19)$$

従って、(3.19) を満たさない場合は数値計算が不安定化する³。

安定性の直感的理解

図 3.1 のように u_i^{n+1} の値は $u_{i-1}^n, u_i^n, u_{i+1}^n$ の3点が重み $dr, 1-2dr, dr$ で平均化されていることを意味しており、重みが正でなければ本当の平均ではなくなる。従って、 $1-2dr > 0$ が必要となる。

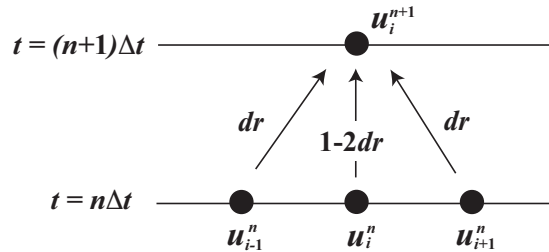


図 3.1: 陽解法の安定性の直感的理解

陽解法の誤差解析

$$\frac{\partial u}{\partial t} = \frac{u(x, t + \Delta t) - u(x, t)}{\Delta t} - \frac{\Delta t}{2} \frac{\partial^2 u}{\partial t^2} - \frac{\Delta t^2}{6} \frac{\partial^3 u}{\partial t^3} + O(\Delta t^3), \quad (3.20)$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x + \Delta x, t) - 2u(x, t) + u(x - \Delta x, t)}{\Delta x^2} - \frac{\Delta x^2}{12} \frac{\partial^4 u}{\partial x^4} - O(\Delta x^4) \quad (3.21)$$

を (3.3) に代入すると

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = d \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} - d \frac{\Delta x^2}{12} \frac{\partial^4 u}{\partial x^4} + \frac{\Delta t}{2} \frac{\partial^2 u}{\partial t^2} + O(\Delta x^4, \Delta t^2), \quad (3.22)$$

³注意：理論的には $d \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$ で数値計算法は安定であるが、数値計算による誤差によって (3.19) を満たしていても不安定化する場合がありますので、 $d \frac{\Delta t}{\Delta x^2} \leq \frac{1}{6}$ 程度にしていた方が安全である。計算が途中で発散した場合は必ず $d \frac{\Delta t}{\Delta x^2}$ を小さくしてみよう。

ここで,

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial t} d \frac{\partial^2 u}{\partial x^2} = d^2 \frac{\partial^4 u}{\partial x^4} \quad (3.23)$$

より

$$\begin{aligned} \frac{u_i^{n+1} - u_i^n}{\Delta t} &= d \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \left(\frac{\Delta t}{2} d^2 - d \frac{\Delta x^2}{12} \right) \frac{\partial^4 u}{\partial x^4} + O(\Delta x^4, \Delta t^2), \\ &= d \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \frac{1}{2} d \Delta x^2 \left(d \frac{\Delta t}{\Delta x^2} - \frac{1}{6} \right) \frac{\partial^4 u}{\partial x^4} + O(\Delta x^4, \Delta t^2). \end{aligned} \quad (3.24)$$

故に

$$d \frac{\Delta t}{\Delta x^2} = \frac{1}{6} \quad (3.25)$$

のとき誤差は $O(\Delta x^4, \Delta t^2)$ となり, それ以外のときの誤差は $O(\Delta x^2, \Delta t)$ となる. しかしながら, 境界条件の誤差から空間離散化の誤差は $O(\Delta x^2)$ となる.

3.2.2 陰解法

陰解法公式

$$u_i^{n+1} - u_i^n = d\theta(ru_{i-1}^{n+1} - 2ru_i^{n+1} + ru_{i+1}^{n+1}) + d(1-\theta)(ru_{i-1}^n - 2ru_i^n + ru_{i+1}^n), \quad 0 \leq i \leq N. \quad (3.26)$$

ここで, $\theta = 1/2$ のとき Crank-Nicolson 公式とよばれる. $\theta = 1$ のとき完全陰解法とよばれる. 陰解法数値スキームの安定性を調べる.

$$\lambda_n = \frac{1 - 4dr(1-\theta) \sin^2\left(\frac{n\Delta x\pi}{2}\right)}{1 + 4dr\theta \sin^2\left(\frac{n\Delta x\pi}{2}\right)}, \quad 0 \leq n \leq N. \quad (3.27)$$

から, $0 < \theta < 1/2$ ならば条件安定, $1/2 \leq \theta \leq 1$ ならば無条件安定であることが分かる. 陰解法は無条件安定なスキームであるが次のような連立一次方程式を解かなければならない.

$$A\mathbf{x} = \mathbf{p}. \quad (3.28)$$

ただし, A は三重対角行列で, その成分は $a_{i,i} = 1 + 2d\theta r$, $a_{i,i-1} = a_{i,i+1} = -d\theta r$ である.

3.2.2.1 LU 分解法

三重対角行列の連立一次方程式の計算は LU 分解法を用いて行う.

$$A = LU$$

ただし, 境界条件が (3.14) あるいは (3.15) の場合

$$A = \begin{pmatrix} a_1 & c_1 & & & & \\ b_2 & a_2 & c_2 & & & \mathbf{0} \\ \cdots & \cdots & \cdots & & & \\ \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \\ & & b_{N-1} & a_{N-1} & c_{N-1} & \\ & & & b_N & a_N & \end{pmatrix} \quad (3.29)$$

となる.

$$L = \begin{pmatrix} L_{11} & & & & & & & & & \mathbf{0} \\ L_{22} & L_{12} & & & & & & & & \\ & \cdots & \cdots & & & & & & & \\ & & \cdots & \cdots & \cdots & & & & & \\ & & & & L_{2N-1} & L_{1N-1} & & & & \\ \mathbf{0} & & & & & L_{2N} & L_{1N} & & & \end{pmatrix}, \quad U = \begin{pmatrix} 1 & U_1 & & & & & & & & \mathbf{0} \\ & 1 & U_2 & & & & & & & \\ & & \cdots & \cdots & & & & & & \\ & & & \cdots & \cdots & \cdots & & & & \\ & & & & & \cdots & \cdots & & & \\ \mathbf{0} & & & & & & 1 & U_{N-1} & & \\ & & & & & & & 1 & & \end{pmatrix}. \quad (3.30)$$

と A を LU 分解し

$$L\mathbf{y} = \mathbf{p},$$

$$U\mathbf{x} = \mathbf{y}$$

を順に解くことで計算できる.

LU 分解のアルゴリズム

```

L[1][1] = a[1];
for (i = 2; i <= N; i++) {
    L[2][i] = b[i];
}
for (i = 1; i <= N-1; i++)
{
    U[i] = c[i] / L[1][i];
    L[1][i+1] = a[i+1] - U[i]*L[2][i+1];
}

```

$L\mathbf{y} = \mathbf{p}$, $U\mathbf{x} = \mathbf{y}$ の解法アルゴリズム

```

y[1] = p[1] / L[1][1];
for (i = 2; i <= N; i++)
{
    y[i] = (p[i] - L[2][i]*y[i-1]) / L[1][i];
}
x[N] = y[N];
for (i = N-1; i >= 1; i--)
{
    x[i] = y[i] - U[i]*x[i+1];
}

```

境界条件が (3.16) の場合, 行列 A は次のようになる

$$A = \begin{pmatrix} a_1 & c_1 & & & & & & & & & b_1 \\ b_2 & a_2 & c_2 & & & & & & & & \mathbf{0} \\ & \cdots & \cdots & \cdots & & & & & & & \\ & & \cdots & \cdots & \cdots & & & & & & \\ \mathbf{0} & & & & b_{N-1} & a_{N-1} & c_{N-1} & & & & \\ c_N & & & & & b_N & a_N & & & & \end{pmatrix}. \quad (3.31)$$

このとき A を LU 分解すると行列 L, U は次のようになる.

$$L = \begin{pmatrix} L_{1,1} & & & & & \\ L_{2,2} & L_{1,2} & & & & \\ & \cdots & \cdots & & & \\ & & \cdots & \cdots & & \\ & & & \cdots & L_{2,N-1} & L_{1,N-1} \\ L_{3,1} & L_{3,2} & \cdots & \cdots & L_{3,N-1} & L_{3,N} \end{pmatrix}, U = \begin{pmatrix} 1 & U_{1,1} & & & & U_{2,1} \\ & 1 & U_{1,2} & & & U_{2,2} \\ & & \cdots & \cdots & & \cdots \\ & & & \cdots & U_{1,N-2} & \\ & & & & 1 & U_{2,N-1} \\ \mathbf{0} & & & & & 1 \end{pmatrix}. \quad (3.32)$$

周期境界条件下での LU 分解のアルゴリズム

```

L[1][1] = a[1];
U[1][1] = c[1]/L[1][1];
U[2][1] = b[1]/L[1][1];
for (i = 2; i <= N-2; i++)
{
    L[2][i] = b[i];
    L[1][i] = a[i] - U[1][i-1]*L[2][i];
    U[1][i] = c[i]/L[1][i];
    U[2][i] = -L[2][i]*U[2][i-1]/L[1][i];
}
i = N-1;
L[2][i] = b[i];
L[1][i] = a[i] - U[1][i-1]*L[2][i];
U[2][i] = (c[i] - L[2][i]*U[2][i-1])/L[1][i];
L[3][1] = c[N];
for (i = 2; i <= N-2; i++)
{
    L[3][i] = -U[1][i-1] * L[3][i-1];
}
i = N-1;
L[3][i] = b[N] - U[1][i-1] * L[3][i-1];
sum = 0.0;
for (i = 1; i <= N-1; i++)
{
    sum = sum + L[3][i] * U[2][i];
}
L[3][N] = a[N] - sum;

```

周期境界条件下での $Ly = p$, $Ux = y$ の解法アルゴリズム

```

y[1] = p[1]/L[1][1];
for (i = 2; i <= N-1; i++)
{
  y[i] = ( p[i] - L[2][i] * y[i-1])/L[1][i];
}
sum = 0.0;
for (i = 1; i <= N-1; i++)
{
  sum = sum + L[3][i]*y[i];
}
y[N] = (p[N] - sum)/L[3][N];
x[N] = y[N];
x[N-1] = y[N-1] - U[2][N-1] * x[N];
for (i = N-2; i >= 1; i-)
{
  x[i] = y[i] - U[1][i] * x[i+1] - U[2][i] * x[N];
}

```

3.2.3 反復改良法 [6]

直接解法では数値計算中の誤差があるために、精度のよい数値解が得られない場合がある。このとき次のような改良をほどこすことによって精度のよい数値解を得ることができる場合がある。

```

A を LU 分解する :  $A = LU$ ;
 $Ly = b, Ux = y$  を解いて近似解  $x = x^{(0)}$  を求める;
 $r^{(0)} := b - Ax^{(0)}$ ;
(反復改良)
For  $m := 1, 2, \dots$  until (終了条件) do
begin
   $Ly = r^{(m-1)}, Uz = y$  を解いて修正量  $z = z^{(m)}$  を求める;
   $x^{(m)} := x^{(m-1)} + z^{(m)}$ ;  $r^{(m)} := b - Ax^{(m)}$ 
end

```

終了条件は、 r_i の丸め誤差限界の評価値を採用する。

$$\delta r_i = (|J_i| + 1.1)\varepsilon_M \left(|b_i| + \sum_{j \in J_i} |a_{i,j}| |x_j| \right) \quad (i = 1, \dots, N) \quad (3.33)$$

として、 $|r_i| \leq \delta r_i$ ($i = 1, \dots, N$) を採用する。ただし、 J_i は A の第 i 行に非零要素のある行番号の集合 (はその要素数) を表し、 ε_M はマシンイプシロンを表す⁴。といっても $\|r^{(m)}\| < 10^{-8}$ で十分だと思う。

3.2.3.1 SSI(Symmetrical Semi-Implicit) 法 [5]

無条件安定な差分にもかかわらず、行列計算が不必要な差分法である。

⁴行列計算の精度に問題があるかどうか確認するためにも使うことができる。

$$u_i^{n+1} - u_i^n = -d \frac{\Delta t}{\Delta x^2} \left(2u_i^{n+1} - \left(\frac{3}{2}u_{i+1}^n + u_i^n + \frac{3}{2}u_{i-1}^n \right) + \left(\frac{1}{2}u_{i+1}^{n-1} + u_i^{n-1} + \frac{1}{2}u_{i-1}^{n-1} \right) \right). \quad (3.34)$$

この方法には1つだけ明らかな問題点がある。それは u_i^1 を求めるとき u_i^{-1} が必要になり、そのようなデータを与えることはできないことである。陽解法あるいは陰解法を使って u_i^1 を計算し、 (u_i^0, u_i^1) を用いて $n=2$ 以降に SSI 法を使うことでこの問題を回避する必要がある。この方法がどの程度有効なのか確認していないので、誰か確認してほしい⁵。

3.2.4 問題

問題1. 熱方程式

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & 0 < x < 1, t > 0, \\ u(0, x) = 4 \sin(\pi x), & 0 \leq x \leq 1, \quad (\text{初期条件}), \\ u(t, 0) = 0, \quad u(t, 1) = 0, & t > 0, \quad (\text{Dirichlet 境界条件}). \end{cases} \quad (3.35)$$

(3.35) の理論解は

$$u(x, t) = 4 \sin(\pi x) \exp(-\pi^2 t) \quad (3.36)$$

なっている。理論解と陽解法、陰解法、SSI 法の数値解を比較してみましょう。

⁵一度試してみたことがあるが、あまり好ましい数値解を得られなかった (時間差分を大きく取ると数値解が波打つような解になった。)

3.3 2次元長方形領域における拡散方程式の数値計算

$$\frac{\partial u}{\partial t} = d_x \frac{\partial^2 u}{\partial x^2} + d_y \frac{\partial^2 u}{\partial y^2}, \quad t > 0, (x, y) \in (0, L_x) \times (0, L_y). \quad (3.37)$$

3.3.1 陽解法 [4]

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \frac{d_x}{\Delta x^2} (u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \frac{d_y}{\Delta y^2} (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n).$$

陽解法の安定性条件は

$$d_x \frac{\Delta t}{\Delta x^2} + d_y \frac{\Delta t}{\Delta y^2} \leq \frac{1}{2}. \quad (3.38)$$

従って、1次元問題より Δt を小さく取る必要が生じる。実用上は使えないと思うだろうが実際には使うことがある。空間2次元になると数値計算の実行時間の問題により $\Delta x, \Delta y$ を大きくした計算をやらざるを得なくなり、(3.38) の条件を満足するためである。

3.3.2 陰解法 [4]

$$\begin{aligned} \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} &= \frac{d_x}{\Delta x^2} (u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}) \\ &+ \frac{d_y}{\Delta y^2} (u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}). \end{aligned} \quad (3.39)$$

1次元問題と同様にやはり安定な数値計算法であるが、次のような連立一次方程式を計算しなければならない：

$$A\mathbf{u}^{n+1} = \mathbf{u}^n. \quad (3.40)$$

2次元領域の場合、行列 A は疎な5重対角行列となっており、直接解法 (LU 分解法) を用いると、行列が密行列となってしまう、計算に要する時間が膨大になってしまう。この問題を解決する方法として、CG 法等の反復解法による連立一次方程式の数値計算法がある⁶。CG 法等を用いた数値計算法は発展編において解説する。

⁶ 今回の講義では解説しない。

3.3.3 ADI法 [5]

$$\frac{u_{i,j}^{n+1/2} - u_{i,j}^n}{\Delta t/2} = \frac{d_x}{\Delta x^2}(u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}) + \frac{d_y}{\Delta y^2}(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n),$$

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n+1/2}}{\Delta t/2} = \frac{d_x}{\Delta x^2}(u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}) + \frac{d_y}{\Delta y^2}(u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}).$$

ADI法は無条件安定であることが知られている。証明が気になる方には桂田先生(明治大学)の資料が参考になる [13]。ADI法は1次元の陰解法のアルゴリズムがそのまま使えるので、プログラムを作るのが大変楽である。

ADI法のアルゴリズム

```

For  $k := 1, 2, \dots$  until  $times$  do
  begin
    For  $j := 0, 1, 2, \dots$  until  $N_y$  do
      begin
         $A_x u_j^{n+1} = B_j u_j^n$ 
      end
    For  $i := 0, 1, 2, \dots$  until  $N_x$  do
      begin
         $A_y u_i^{n+1} = B_i u_i^n$ 
      end
    end
  end

```

3.3.4 2次元 SSI法 [5]

$$\begin{aligned} \left(1 + 2d \frac{\Delta t}{\Delta x^2} + 2d \frac{\Delta t}{\Delta y^2}\right) u_{i,j}^{n+1} &= u_{i,j}^n + d \frac{\Delta t}{\Delta x^2} \left(\frac{3}{2} u_{i+1,j}^n + u_{i,j}^n + \frac{3}{2} u_{i-1,j}^n\right) \\ &- d \frac{\Delta t}{\Delta x^2} \left(\frac{1}{2} u_{i+1,j}^{n-1} + u_{i,j}^{n-1} + \frac{1}{2} u_{i-1,j}^{n-1}\right) \\ &+ d \frac{\Delta t}{\Delta y^2} \left(\frac{3}{2} u_{i,j+1}^n + u_{i,j}^n + \frac{3}{2} u_{i,j-1}^n\right) \\ &- d \frac{\Delta t}{\Delta y^2} \left(\frac{1}{2} u_{i,j+1}^{n-1} + u_{i,j}^{n-1} + \frac{1}{2} u_{i,j-1}^{n-1}\right). \end{aligned} \quad (3.41)$$

この解法がどこまで有効なのかわからない。一度試してみたい気がする。

3.4 3次元直方体領域での数値計算法

$$\frac{\partial u}{\partial t} = d_x \frac{\partial^2 u}{\partial x^2} + d_y \frac{\partial^2 u}{\partial y^2} + d_z \frac{\partial^2 u}{\partial z^2}, \quad t > 0, (x, y, z) \in (0, L_x) \times (0, L_y) \times (0, L_z). \quad (3.42)$$

3.4.1 陽解法

3次元問題だと使う場合が結構ある。私も3次元問題は陽解法を使うことがある。

$$\begin{aligned} \frac{u_{i,j,k}^{n+1} - u_{i,j,k}^n}{\Delta t} &= \frac{d_x}{\Delta x^2} (u_{i+1,j,k}^n - 2u_{i,j,k}^n + u_{i-1,j,k}^n) \\ &+ \frac{d_y}{\Delta y^2} (u_{i,j+1,k}^n - 2u_{i,j,k}^n + u_{i,j-1,k}^n) \\ &+ \frac{d_z}{\Delta z^2} (u_{i,j,k+1}^n - 2u_{i,j,k}^n + u_{i,j,k-1}^n). \end{aligned}$$

陽解法の安定性条件は

$$d_x \frac{\Delta t}{\Delta x^2} + d_y \frac{\Delta t}{\Delta y^2} + d_z \frac{\Delta t}{\Delta z^2} \leq \frac{1}{2}. \quad (3.43)$$

従って、2次元問題より Δt を小さく取る必要が生じるが、2次元問題と同様の理由で使うことが多い。

3.4.2 ADI法 [5]

無条件安定なので安心して使うことができる !!

$$\begin{aligned} \frac{u_{i,j,k}^{n+1/3} - u_{i,j,k}^n}{\Delta t} &= \frac{d_x}{2\Delta x^2} (u_{i+1,j,k}^{n+1/3} - 2u_{i,j,k}^{n+1/3} + u_{i-1,j,k}^{n+1/3} + u_{i+1,j,k}^n - 2u_{i,j,k}^n + u_{i-1,j,k}^n) \\ &+ \frac{d_y}{\Delta y^2} (u_{i,j+1,k}^n - 2u_{i,j,k}^n + u_{i,j-1,k}^n) \\ &+ \frac{d_z}{\Delta z^2} (u_{i,j,k+1}^n - 2u_{i,j,k}^n + u_{i,j,k-1}^n), \end{aligned}$$

$$\begin{aligned} \frac{u_{i,j,k}^{n+2/3} - u_{i,j,k}^{n+1/3}}{\Delta t} &= \frac{d_x}{2\Delta x^2} (u_{i+1,j,k}^{n+1/3} - 2u_{i,j,k}^{n+1/3} + u_{i-1,j,k}^{n+1/3} + u_{i+1,j,k}^n - 2u_{i,j,k}^n + u_{i-1,j,k}^n) \\ &+ \frac{d_y}{2\Delta y^2} (u_{i,j+1,k}^{n+2/3} - 2u_{i,j,k}^{n+2/3} + u_{i,j-1,k}^{n+2/3} + u_{i,j+1,k}^n - 2u_{i,j,k}^n + u_{i,j-1,k}^n) \\ &+ \frac{d_z}{\Delta z^2} (u_{i,j,k+1}^n - 2u_{i,j,k}^n + u_{i,j,k-1}^n), \end{aligned}$$

$$\begin{aligned} \frac{u_{i,j,k}^{n+1} - u_{i,j,k}^{n+2/3}}{\Delta t} &= \frac{d_x}{2\Delta x^2} (u_{i+1,j,k}^{n+1/3} - 2u_{i,j,k}^{n+1/3} + u_{i-1,j,k}^{n+1/3} + u_{i+1,j,k}^n - 2u_{i,j,k}^n + u_{i-1,j,k}^n) \\ &+ \frac{d_y}{2\Delta y^2} (u_{i,j+1,k}^{n+2/3} - 2u_{i,j,k}^{n+2/3} + u_{i,j-1,k}^{n+2/3} + u_{i,j+1,k}^n - 2u_{i,j,k}^n + u_{i,j-1,k}^n) \\ &+ \frac{d_z}{2\Delta z^2} (u_{i,j,k+1}^{n+1} - 2u_{i,j,k}^{n+1} + u_{i,j,k-1}^{n+1} + u_{i,j,k+1}^n - 2u_{i,j,k}^n + u_{i,j,k-1}^n). \end{aligned}$$

3.4.3 陰解法

$$\begin{aligned}\frac{u_{i,j,k}^{n+1} - u_{i,j,k}^n}{\Delta t} &= \frac{d_x}{\Delta x^2} (u_{i+1,j,k}^{n+1} - 2u_{i,j,k}^{n+1} + u_{i-1,j,k}^{n+1}) \\ &+ \frac{d_y}{\Delta y^2} (u_{i,j+1,k}^{n+1} - 2u_{i,j,k}^{n+1} + u_{i,j-1,k}^{n+1}) \\ &+ \frac{d_z}{\Delta z^2} (u_{i,j,k+1}^{n+1} - 2u_{i,j,k}^{n+1} + u_{i,j,k-1}^{n+1}).\end{aligned}$$

この解法も無条件安定であることがわかるが、七重対角行列の数値計算を行なわなければならない。この場合はやはりCG法等の反復法ををを使わなければ計算速度が遅くなる。陰解法を用いた数値計算法は発展編において解説する。

第4章 反応拡散方程式の数値計算法（差分法）

本章では、次のような反応拡散系の数値計算法を解説する：

$$\mathbf{u}_t = D\Delta\mathbf{u} + \mathbf{f}(\mathbf{u}), \quad t > 0, \quad \mathbf{x} \in \mathbf{R}^n. \quad (4.1)$$

ただし、

$$\begin{aligned} \mathbf{x} &= (x_1, x_2, \dots, x_n), \quad \mathbf{u} = (u_1, u_2, \dots, u_m), \quad \mathbf{f} = (f_1, f_2, \dots, f_m), \\ \mathbf{u} &: \mathbf{R}_+ \times \mathbf{R}^n \longrightarrow \mathbf{R}^m, \\ D &: \text{非負対角行列 } (d_i \geq 0), \quad \Delta = \sum_{k=1}^n \frac{\partial^2}{\partial x_k^2}. \end{aligned}$$

(4.1) で表されるものを**反応拡散系**という。

例：FitzHugh-Nagumo 方程式 [27]

$$\begin{cases} u_t = u_{xx} + \frac{1}{\varepsilon} (u(1-u)(u-a) - v), \\ v_t = u - \gamma v. \end{cases} \quad (4.2)$$

ただし、 $x \in \mathbf{R}$, $u, v : \mathbf{R}_+ \times \mathbf{R} \longrightarrow \mathbf{R}$.

4.1 1次元反応拡散系の数値計算法

ここでは1次元反応拡散系の初期値境界値問題の数値計算法について解説する。

4.1.1 1次元反応拡散系の初期・境界値問題

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = D \frac{\partial^2}{\partial x^2} \mathbf{u} + \mathbf{f}(\mathbf{u}), & t > 0, \quad x \in I = (0, L), \\ \mathbf{u}(0, x) = \mathbf{u}_0(x), & x \in \bar{I}, \\ \text{境界条件.} \end{cases} \quad (4.3)$$

4.1.1.1 境界条件

1. Dirichlet 条件

$$\mathbf{u}(t, 0) = \mathbf{a}, \quad \mathbf{u}(t, L) = \mathbf{b}. \quad (4.4)$$

- 多くの場合、 \mathbf{a}, \mathbf{b} は $\mathbf{f}(\mathbf{u}) = 0$ の安定平衡点にとる。

2. Neumann 条件

$$\frac{\partial \mathbf{u}}{\partial x}(t, 0) = \mathbf{a}, \quad \frac{\partial \mathbf{u}}{\partial x}(t, L) = \mathbf{b}. \quad (4.5)$$

- 多くの場合、 $\mathbf{a} = \mathbf{b} = \mathbf{0}$.
- その場合、流量なし条件とか断熱条件という。

3. 周期境界条件

$$\begin{cases} \mathbf{u}(t, 0) = \mathbf{u}(t, L), \\ \frac{\partial \mathbf{u}}{\partial x}(t, 0) = \frac{\partial \mathbf{u}}{\partial x}(t, L). \end{cases} \quad (4.6)$$

- 物理でよく使う境界条件 (境界に影響しないパターンダイナミクスを考えるとときに有効).
- 進行波解 (一定速度, 一定波形) の性質を調べるときに用いる.

4. 第3種境界条件 (滅多に使わない)

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial x}(t, 0) = -\alpha_1 (\mathbf{a} - \beta_1 \mathbf{u}(t, 0)), \\ \frac{\partial \mathbf{u}}{\partial x}(t, L) = \alpha_2 (\mathbf{b} - \beta_2 \mathbf{u}(t, L)). \end{cases} \quad (4.7)$$

- $\alpha_i \rightarrow 0$ ($i = 1, 2$) ならば, Neumann 条件と同じになる.
- $\alpha_i \rightarrow \infty$ ($i = 1, 2$) ならば, Dirichlet 条件と同じになる.

4.1.2 2変数反応拡散系の数値計算法 (半陰解法)

$$\begin{cases} u_t = d_u u_{xx} + f(u, v), \\ v_t = d_v v_{xx} + g(u, v). \end{cases} \quad (4.8)$$

を

初期条件: $u(0, x) = u_0(x)$, $v(0, x) = v_0(x)$ および境界条件: (4.4)-(4.7) のいずれか, のもとで数値計算をする方法を説明する.

4.1.2.1 方程式の離散化 (差分化)

- 熱方程式の部分を陰解法
- 非線形項 f, g の部分は陽解法

設定: $\Delta x = \frac{L}{N}$, $x_i = i \times \Delta x (i = 1, 2, \dots, N)$, $t_n = n \times \Delta t$ (適切な値)

$$\begin{aligned} & \frac{u(t_{n+1}, x_i) - u(t_n, x_i)}{\Delta t} \\ &= d_u \frac{u(t_{n+1}, x_{i+1}) - 2u(t_{n+1}, x_i) + u(t_{n+1}, x_{i-1}))}{\Delta x^2} + f(u(t_n, x_i), v(t_n, x_i)). \end{aligned} \quad (4.9)$$

v についても同様である. ここで, 次のように書くことにする:

$$u(t_n, x_i) = u_i^n, \quad v(t_n, x_i) = v_i^n.$$

このとき (4.8) の差分方程式は

$$\begin{cases} u_i^{n+1} - u_i^n = \frac{\Delta t}{\Delta x^2} d_u (u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) + \Delta t f(u_i^n, v_i^n), \\ v_i^{n+1} - v_i^n = \frac{\Delta t}{\Delta x^2} d_v (v_{i+1}^{n+1} - 2v_i^{n+1} + v_{i-1}^{n+1}) + \Delta t g(u_i^n, v_i^n) \end{cases} \quad (4.10)$$

と書くことができる。ここで、 $r = \frac{\Delta t}{\Delta x^2}$ とおくと

$$\Rightarrow \begin{cases} -rd_u u_{i-1}^{n+1} + (1 + 2rd_u)u_i^{n+1} - rd_u u_{i+1}^{n+1} = u_i^n + \Delta t f(u_i^n, v_i^n), \\ -rd_v v_{i-1}^{n+1} + (1 + 2rd_v)v_i^{n+1} - rd_v v_{i+1}^{n+1} = v_i^n + \Delta t g(u_i^n, v_i^n) \end{cases} \quad (4.11)$$

となる。

$n = 0$ のとき

$$\begin{cases} u_i^0 = u(0, x_i) = u_0(x_i), \\ v_i^0 = v(0, x_i) = v_0(x_i) \end{cases} \quad (4.12)$$

であるから、初期条件より右辺の値は確定する。よって、(4.11)の数値計算をすることで、順次求まる。

4.1.2.2 境界条件の離散化

1. Dirichlet 条件

$$\begin{cases} u(t, 0) = a_u, & u(t, L) = b_u, \\ v(t, 0) = a_v, & v(t, L) = b_v. \end{cases} \\ \Rightarrow \begin{cases} u_0^n = a_u, & u_N^n = b_u, \\ v_0^n = a_v, & v_N^n = b_v. \end{cases} \quad (4.13)$$

2. Neumann 条件 (4.5) から

$$\begin{cases} \frac{\partial u}{\partial x}(t, 0) = a_u, \\ \frac{\partial u}{\partial x}(t, N) = b_u. \end{cases} \\ \Rightarrow \begin{cases} \frac{u_1^n - u_{-1}^n}{2\Delta x} + O(\Delta x^2) = a_u, \\ \frac{u_{N+1}^n - u_{N-1}^n}{2\Delta x} + O(\Delta x^2) = b_u. \end{cases}$$

従って

$$\begin{cases} u_{-1}^n = u_1^n - 2\Delta x a_u, \\ u_{N+1}^n = u_{N-1}^n - 2\Delta x b_u. \end{cases} \quad (4.14)$$

と離散化することができる。 v についても同様に

$$\begin{cases} v_{-1}^n = v_1^n - 2\Delta x a_v, \\ v_{N+1}^n = v_{N-1}^n - 2\Delta x b_v \end{cases} \quad (4.15)$$

とできる。

3. 周期境界条件 (4.6) より

$$u(t, 0) = u(t, L) \Rightarrow u_0^n = u_N^n. \quad (4.16)$$

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

⇒

$$\frac{u_1^n - u_{-1}^n}{2\Delta x} + O(\Delta x^2) = \frac{u_{N+1}^n - u_{N-1}^n}{2\Delta x} + O(\Delta x^2).$$

従って

$$u_{-1}^n = u_{N-1}^n$$

となる. 以上をまとめると

$$u_0^n = u_N^n, \quad u_{-1}^n = u_{N-1}^n \quad (4.17)$$

となる. v についても同様に

$$v_0^n = v_N^n, \quad v_{-1}^n = v_{N-1}^n \quad (4.18)$$

となる.

4. 第3種境界条件

$$\begin{cases} \frac{\partial u}{\partial x}(t, 0) = \alpha_{u1}(a_u - \beta_{u1}u(t, 0)), \\ \frac{\partial u}{\partial x}(t, L) = \alpha_{u2}(b_u - \beta_{u2}u(t, L)). \end{cases}$$

$$\Rightarrow \begin{cases} \frac{u_1^n - u_{-1}^n}{2\Delta x} + O(\Delta x^2) = \alpha_{u1}(a_u - \beta_{u1}u_0^n), \\ \frac{u_{N+1}^n - u_{N-1}^n}{2\Delta x} + O(\Delta x^2) = \alpha_{u1}(b_u - \beta_{u2}u_N^n). \end{cases}$$

従って

$$\begin{cases} u_{-1}^n = u_1^n - 2\Delta x\alpha_{u1}(a_u - \beta_{u1}u_0^n), \\ u_{N+1}^n = u_{N-1}^n + 2\Delta x\alpha_{u2}(b_u - \beta_{u2}u_N^n). \end{cases} \quad (4.19)$$

4.1.3 差分方程式の連立1次方程式表示

4.1.3.1 Dirichlet 条件のとき

$i = 0, N$ での値はすでに求まっているので, $u_0^{n+1}, u_N^{n+1}, v_0^{n+1}, v_N^{n+1}$ は解かなくてもよい. $i = 1, \dots, N-1$ で(4.11)を解けばよい.

(4.13)より, $i = 1$ のとき

$$-rd_u a_u + (1 + 2rd_u)u_1^{n+1} - rd_u u_2^{n+1} = u_1^n + \Delta t f(u_1^n, v_1^n)$$

となる.

(4.13)より, $i = N-1$ のとき

$$-rd_u u_{N-2}^{n+1} + (1 + 2rd_u)u_{N-1}^{n+1} - rd_u b_u = u_{N-1}^n + \Delta t f(u_{N-1}^n, v_{N-1}^n)$$

となる. $2 < i < N-2$ のときは, (4.11)と同じ差分方程式となる. 従って, 計算すべき連立1次方程式は次のようになる:

$$\begin{pmatrix} 1 + 2rd_u & -rd_u & 0 & 0 & 0 & \dots & 0 \\ -rd_u & 1 + 2rd_u & -rd_u & 0 & 0 & \dots & 0 \\ 0 & -rd_u & 1 + 2rd_u & -rd_u & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & -rd_u & 1 + 2rd_u \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} u_1^n + \Delta t f(u_1^n, v_1^n) + rd_u a_u \\ \vdots \\ u_i^n + \Delta t f(u_i^n, v_i^n) \\ \vdots \\ u_{N-1}^n + \Delta t f(u_{N-1}^n, v_{N-1}^n) + rd_u b_u \end{pmatrix}. \quad (4.20)$$

v についても同様である.

4.1.3.2 Neumann 条件のとき

$0 \leq i \leq N$ において, (4.11) を解く.

(4.14) より, $i = 0$ のとき

$$-rd_u(u_1^{n+1} - 2\Delta xa_u) + (1 + 2rd_u)u_0^{n+1} - rd_u u_1^{n+1} = u_0^n + \Delta t f(u_0^n, v_0^n)$$

\Leftrightarrow

$$(1 + 2rd_u)u_0^{n+1} - 2rd_u u_1^{n+1} = u_0^n + \Delta t f(u_0^n, v_0^n) - 2rd_u \Delta xa_u \quad (4.21)$$

となる.

(4.14) より, $i = N$ のとき

$$-rd_u u_{N-1}^n + (1 + 2rd_u)u_N^{n+1} - rd_u(u_{N-1}^n + 2\Delta xb_u) = u_N^n + \Delta t f(u_N^n, v_N^n)$$

\Leftrightarrow

$$-2rd_u u_{N-1}^n + (1 + 2rd_u)u_N^{n+1} = u_N^n + \Delta t f(u_N^n, v_N^n) + 2rd_u \Delta xb_u \quad (4.22)$$

となる. $i = 1, \dots, N-1$ のときは, (4.11) と同じ差分方程式となる. 従って, 計算すべき連立1次方程式は次のようになる:

$$\begin{pmatrix} \frac{1}{2}(1 + 2rd_u) & -rd_u & 0 & 0 & 0 & \dots & 0 \\ -rd_u & 1 + 2rd_u & -rd_u & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -rd_u & 1 + 2rd_u & -rd_u \\ 0 & 0 & 0 & 0 & 0 & -rd_u & \frac{1}{2}(1 + 2rd_u) \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} \\
 = \begin{pmatrix} \frac{1}{2}(u_0^n + \Delta t f(u_0^n, v_0^n) - 2rd_u \Delta xa_u) \\ \vdots \\ u_i^n + \Delta t f(u_i^n, v_i^n) \\ \vdots \\ \frac{1}{2}(u_N^n + \Delta t f(u_N^n, v_N^n) + 2rd_u \Delta xb_u) \end{pmatrix}. \quad (4.23)$$

v についても同様である.

4.1.3.3 周期境界条件のとき

(4.17) から $u_0^{n+1} = u_N^{n+1}$ となるので, u_i^{n+1} ($0 \leq i \leq N-1$) において, (4.11) を解けばよい.
 (4.17) より, $i=0$ のとき

$$\begin{aligned} & -rd_u u_{N-1}^{n+1} + (1+2rd_u)u_0^{n+1} - rd_u u_1^{n+1} = u_0^n + \Delta t f(u_0^n, v_0^n) \\ \Leftrightarrow & (1+2rd_u)u_0^{n+1} - rd_u u_1^{n+1} - rd_u u_{N-1}^{n+1} = u_0^n + \Delta t f(u_0^n, v_0^n) \end{aligned} \quad (4.24)$$

となる.

(4.17) より, $i=N-1$ のとき

$$\begin{aligned} & -rd_u u_{N-2}^n + (1+2rd_u)u_{N-1}^{n+1} - rd_u u_1^n = u_N^n + \Delta t f(u_{N-1}^n, v_{N-1}^n) \\ \Leftrightarrow & -rd_u u_1^n - rd_u u_{N-2}^n + (1+2rd_u)u_{N-1}^{n+1} = u_N^n + \Delta t f(u_{N-1}^n, v_{N-1}^n) \end{aligned} \quad (4.25)$$

となる. $i=1, \dots, N-2$ のときは, (4.11) と同じ差分方程式となる. 従って, 計算すべき連立1次方程式は次のようになる:

$$\begin{aligned} & \begin{pmatrix} 1+2rd_u & -rd_u & 0 & 0 & \cdots & 0 & -rd_u \\ -rd_u & 1+2rd_u & -rd_u & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & -rd_u & 1+2rd_u & -rd_u \\ -rd_u & 0 & \cdots & 0 & 0 & -rd_u & 1+2rd_u \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ \vdots \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \end{pmatrix} \\ & = \begin{pmatrix} u_0^n + \Delta t f(u_0^n, v_0^n) \\ \vdots \\ u_i^n + \Delta t f(u_i^n, v_i^n) \\ \vdots \\ \vdots \\ u_{N-1}^n + \Delta t f(u_{N-1}^n, v_{N-1}^n) \end{pmatrix}. \end{aligned} \quad (4.26)$$

v についても同様である.

4.1.3.4 問題

問題 1. 神経繊維上の電位の伝播モデル (FitzHugh-Nagumo 方程式)[27]

$$\begin{cases} \frac{\partial u}{\partial t} = d_u u_{xx} + \frac{1}{\varepsilon} (u(1-u)(u-a) - v), & t > 0, 0 < x < L \\ \frac{\partial v}{\partial t} = u - \gamma v. \end{cases} \quad (4.27)$$

$\gamma = 1.0, \varepsilon = 0.001, a = 0.125, d_u = 1.0, d_v = 0.1, L = 20.0$ において数値計算を行ってみなさい. ただし, 境界条件は Neumann 境界条件

$$\frac{\partial u}{\partial x}(t, 0) = \frac{\partial u}{\partial x}(t, L) = 0, \quad t > 0$$

と周期境界条件

$$u(t, 0) = u(t, L), \quad \frac{\partial u}{\partial x}(t, 0) = \frac{\partial u}{\partial x}(t, L), \quad t > 0,$$

の両方の場合に行なってみなさい。そして d_v を大きくとると (例えば, $d_v = 10.0$) どうなるか数値実験してみなさい。

4.1.3.5 注意

陰解法は数値スキームが安定なので、 Δt を大きく取ることができる。しかし、「スキームが安定なこと」と「モデル方程式の解挙動を近似すること」は同じ意味ではないことを忘れてはいけない。 Δt を大きくしたのではゴースト解¹を見ってしまう恐れがある。また、陰解法が最も有効な場合は解の漸近挙動 (時間が十分経過した後) を見ることである。

¹差分方程式の解であって連続方程式の解でないもの、 $\Delta t \rightarrow 0, \Delta x \rightarrow 0 (\Delta t / \Delta x^2 \text{は一定})$ とすると見えなくなってしまう

4.2 2次元反応拡散系の数値計算法

ここでは無条件安定な数値計算法の一つである ADI 法 [5] を用いた数値計算法を解説する。

4.2.1 方程式の離散化

次の2変数反応拡散方程式系を考える：

$$\begin{cases} u_t = d_u \Delta u + f(u, v), \\ v_t = d_v \Delta v + g(u, v), \end{cases} \quad t > 0, \quad \mathbf{x} \in \Omega. \quad (4.28)$$

ただし、 Ω は長方形領域 $\Omega = (0, L_x) \times (0, L_y)$ とし

$$\begin{cases} u = u(t, \mathbf{x}) = u(t, x, y), \\ v = v(t, \mathbf{x}) = v(t, x, y), \end{cases} \quad \mathbf{x} \in \bar{\Omega}. \quad (4.29)$$

と書く。

初期条件は

$$\begin{cases} u(0, x, y) = u_0(x, y), \\ v(0, x, y) = v_0(x, y), \end{cases} \quad (x, y) \in \bar{\Omega}. \quad (4.30)$$

とする。ここで

$$\Delta x = \frac{L_x}{Nx}, \quad \Delta y = \frac{L_y}{Ny} (\Delta x = \Delta y), \quad x_i = i \times \Delta x, \quad y_j = j \times \Delta y$$

とおき、 Δt を与えて

$$t_n = n \times \Delta t$$

とおく。以降、 $u(t_n, x_i, y_j) = u_{i,j}^n$ と書くことにする。(4.28) の離散化を行なう。2階微分を次の中心差分で離散化を行なう：

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} = \frac{u(t, x_i + \Delta x, y_j) - 2u(t, x_i, y_j) + u(t, x_i - \Delta x, y_j)}{\Delta x^2} + O(\Delta x^2), \\ \frac{\partial^2 u}{\partial y^2} = \frac{u(t, x_i, y_j + \Delta y) - 2u(t, x_i, y_j) + u(t, x_i, y_j - \Delta y)}{\Delta y^2} + O(\Delta y^2). \end{cases} \quad (4.31)$$

さらに、時間に関しては次のように前進差分で離散化する：

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{u(t_n + \Delta t, x_i, y_j) - u(t_n, x_i, y_j)}{\Delta t} + O(\Delta t) \\ &\approx \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t}. \end{aligned}$$

- 陽解法：(4.31) において、 $t = t_n$ とする。このとき、数値解法の安定性より

$$\Delta t \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right) \leq \frac{1}{2}$$

の条件が必要である。

- 陰解法：(4.31) において、 $t = t_{n+1}$ とする。このとき、無条件安定であるが、 $(Nx + 1)(Ny + 1) \times (Nx + 1)(Ny + 1)$ 行列の5重対角連立1次方程式の数値解法が必要になってくる。

4.2.2 非線形項の離散化

$$\begin{cases} f(u, v) = f(u_{i,j}^n, v_{i,j}^n), \\ g(u, v) = g(u_{i,j}^n, v_{i,j}^n). \end{cases} \quad (4.32)$$

4.2.3 初期条件の離散化

$$\begin{cases} u_{i,j}^0 = u_0(x_i, y_j), \\ v_{i,j}^0 = v_0(x_i, y_j). \end{cases} \quad (4.33)$$

4.2.4 境界条件の離散化

4.2.4.1 斉次 Neumann 境界条件

ここで、境界条件は次のような斉次 Neumann 境界条件を与える：

$$\frac{\partial u}{\partial n} = \frac{\partial v}{\partial n} = 0, \quad t > 0, \quad \mathbf{x} \in \partial\Omega \quad (4.34)$$

とするただし、 n は $\partial\Omega$ の外向き法線ベクトルとする。 $\Omega = (0, L_x) \times (0, L_y)$ のとき、式 (4.34) は

$$\begin{cases} \frac{\partial u}{\partial y}(x, 0) = \frac{\partial u}{\partial y}(x, L_y) = 0, & x \in (0, L_x), \\ \frac{\partial u}{\partial x}(0, y) = \frac{\partial u}{\partial x}(L_x, y) = 0, & y \in (0, L_y) \end{cases} \quad (4.35)$$

と書ける。

このとき、境界条件の離散化は

$$\begin{cases} \frac{\partial u}{\partial x}(t, 0, y) \approx \frac{u(t, x_1, y_j) - u(t, x_{-1}, y_j)}{2\Delta x}, & (0 \leq j \leq Ny), \\ \frac{\partial u}{\partial x}(t, L_x, y) \approx \frac{u(t, x_{Nx+1}, y_j) - u(t, x_{Nx-1}, y_j)}{2\Delta x}, & (0 \leq j \leq Ny), \\ \frac{\partial u}{\partial y}(t, x, 0) \approx \frac{u(t, x_i, y_1) - u(t, x_i, y_{-1})}{2\Delta y}, & (0 \leq i \leq Nx), \\ \frac{\partial u}{\partial y}(t, x, L_y) \approx \frac{u(t, x_i, y_{Ny+1}) - u(t, x_i, y_{Ny-1})}{2\Delta y}, & (0 \leq i \leq Nx) \end{cases} \quad (4.36)$$

となる。(4.36) より

$$\begin{cases} u_{-1,j}^n = u_{1,j}^n, & u_{Nx+1,j}^n = u_{Nx-1,j}^n, & (0 \leq j \leq Ny), \\ u_{i,-1}^n = u_{i,1}^n, & u_{i,Ny+1}^n = u_{i,Ny-1}^n, & (0 \leq i \leq Nx) \end{cases} \quad (4.37)$$

となる。

4.2.4.2 周期境界条件

境界条件が周期境界条件ならば

$$\begin{cases} u(0, y) = u(L_x, y), & u(x, 0) = u(x, L_y), \\ \frac{\partial u}{\partial x}(0, y) = \frac{\partial u}{\partial x}(L_x, y), & \frac{\partial u}{\partial y}(x, 0) = \frac{\partial u}{\partial y}(x, L_y) \end{cases} \quad (4.38)$$

となり, その離散化は

$$\begin{cases} u(t, x_0, y_j) = u(t, x_{Nx}, y_j) \implies u_{0,j}^n = u_{Nx,j}^n, & (0 \leq j \leq Ny), \\ u(t, x_i, y_0) = u(t, x_i, y_{Ny}) \implies u_{i,0}^n = u_{i,Ny}^n, & (0 \leq i \leq Nx), \\ u(t, x_{-1}, y_j) = u(t, x_{Nx-1}, y_j) \implies u_{-1,j}^n = u_{Nx-1,j}^n, & (0 \leq j \leq Ny), \\ u(t, x_i, y_{-1}) = u(t, x_i, y_{Ny-1}) \implies u_{i,-1}^n = u_{i,Ny-1}^n, & (0 \leq i \leq Nx) \end{cases} \quad (4.39)$$

となる.

4.2.5 ADI法

ここでは, 無条件安定な計算法かつ空間1次元の陰解法の応用で数値計算可能なADI法を用いて数値計算する方法を説明する.

4.2.5.1 斉次 Neumann 境界条件の場合

1 段目

$$\begin{aligned} & \frac{u_{i,j}^{n+\frac{1}{2}} - u_{i,j}^n}{\left(\frac{\Delta t}{2}\right)} \\ &= d_u \left(\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{\Delta x^2} \right) + d_u \left(\frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right) + f(u_{i,j}^n, v_{i,j}^n). \end{aligned}$$

2 段目

$$\begin{aligned} & \frac{u_{i,j}^{n+1} - u_{i,j}^{n+\frac{1}{2}}}{\left(\frac{\Delta t}{2}\right)} \\ &= d_u \left(\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{\Delta x^2} \right) + d_u \left(\frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2} \right) + f(u_{i,j}^{n+\frac{1}{2}}, v_{i,j}^{n+\frac{1}{2}}). \end{aligned}$$

1 段目に関して

$$r_x = \frac{1}{\Delta x^2} \times \left(\frac{\Delta t}{2}\right), \quad r_y = \frac{1}{\Delta y^2} \times \left(\frac{\Delta t}{2}\right)$$

とおくと

$$\begin{aligned} & -d_u r_x u_{i+1,j}^{n+\frac{1}{2}} + (1 + 2d_u r_x) u_{i,j}^{n+\frac{1}{2}} - d_u r_x u_{i-1,j}^{n+\frac{1}{2}} \\ &= u_{i,j}^n + d_u r_y (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) + \frac{\Delta t}{2} f(u_{i,j}^n, v_{i,j}^n) \end{aligned}$$

と書ける. ここで

$$A = \begin{pmatrix} \frac{1}{2}(1+2d_u r_x) & -d_u r_x & 0 & 0 & 0 & 0 \\ -d_u r_x & 1+2d_u r_x & -d_u r_x & 0 & 0 & 0 \\ 0 & -d_u r_x & 1+2d_u r_x & -d_u r_x & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -d_u r_x & 1+2d_u r_x & -d_u r_x \\ 0 & 0 & 0 & 0 & -d_u r_x & \frac{1}{2}(1+2d_u r_x) \end{pmatrix}$$

とすると, この式は各 j に対して

$$A u_j^{n+\frac{1}{2}} = b_j, \quad 0 \leq j \leq N_y \quad (4.40)$$

と書くことができる連立一次方程式を解く問題に帰着される. ただし, 斉次 Neumann 境界条件の離散化 (4.37) より, b_j の各要素は

$$b_0 = \begin{cases} b_0 = \frac{1}{2} \left(u_{0,0}^n + d_u r_y (2u_{0,1}^n - 2u_{0,0}^n) + \frac{\Delta t}{2} f(u_{0,0}^n, v_{0,0}^n) \right), \\ b_i = \left(u_{i,0}^n + d_u r_y (2u_{i,1}^n - 2u_{i,0}^n) + \frac{\Delta t}{2} f(u_{i,0}^n, v_{i,0}^n) \right), \quad (1 \leq i \leq N_x - 1), \\ b_{N_x} = \frac{1}{2} \left(u_{N_x,0}^n + d_u r_y (2u_{N_x,1}^n - 2u_{N_x,0}^n) + \frac{\Delta t}{2} f(u_{N_x,0}^n, v_{N_x,0}^n) \right), \end{cases} \quad (4.41)$$

$$b_j = \begin{cases} b_0 = \frac{1}{2} \left(u_{0,j}^n + d_u r_y (u_{0,j+1}^n - 2u_{0,j}^n + u_{0,j-1}^n) + \frac{\Delta t}{2} f(u_{0,j}^n, v_{0,j}^n) \right), \\ b_i = \left(u_{i,j}^n + d_u r_y (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) + \frac{\Delta t}{2} f(u_{i,j}^n, v_{i,j}^n) \right), \quad (1 \leq i \leq N_x - 1), \\ b_{N_x} = \frac{1}{2} \left(u_{N_x,j}^n + d_u r_y (u_{N_x,j+1}^n - 2u_{N_x,j}^n + u_{N_x,j-1}^n) + \frac{\Delta t}{2} f(u_{N_x,j}^n, v_{N_x,j}^n) \right), \end{cases} \quad (4.42)$$

$$b_{N_y} = \begin{cases} b_0 = \frac{1}{2} \left(u_{0,N_y}^n + d_u r_y (2u_{0,N_y-1}^n - 2u_{0,N_y}^n) + \frac{\Delta t}{2} f(u_{0,N_y}^n, v_{0,N_y}^n) \right), \\ b_i = \left(u_{i,N_y}^n + d_u r_y (2u_{i,N_y-1}^n - 2u_{i,N_y}^n) + \frac{\Delta t}{2} f(u_{i,N_y}^n, v_{i,N_y}^n) \right), \quad (1 \leq i \leq N_x - 1), \\ b_{N_x} = \frac{1}{2} \left(u_{N_x,N_y}^n + d_u r_y (2u_{N_x,N_y-1}^n - 2u_{N_x,N_y}^n) + \frac{\Delta t}{2} f(u_{N_x,N_y}^n, v_{N_x,N_y}^n) \right) \end{cases} \quad (4.43)$$

となる. 各 j に対して, (4.40) を解くと

$$u_{i,j}^{n+\frac{1}{2}}, \quad (0 \leq i \leq N_x, \quad 0 \leq j \leq N_y)$$

が求まる.

2 段目に関しては

$$\begin{aligned} & -d_u r_y u_{i,j+1}^{n+1} + (1+2d_u r_y) u_{i,j}^{n+1} - d_u r_y u_{i,j-1}^{n+1} \\ & = d_u r_x \left(u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}} \right) + \frac{\Delta t}{2} f \left(u_{i,j}^{n+\frac{1}{2}}, v_{i,j}^{n+\frac{1}{2}} \right) \end{aligned}$$

より

$$B = \begin{pmatrix} \frac{1}{2}(1+2d_u r_y) & -d_u r_y & 0 & 0 & 0 & 0 \\ -d_u r_y & 1+2d_u r_y & -d_u r_y & 0 & 0 & 0 \\ 0 & -d_u r_y & 1+2d_u r_y & -d_u r_y & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -d_u r_y & 1+2d_u r_y & -d_u r_y \\ 0 & 0 & 0 & 0 & -d_u r_y & \frac{1}{2}(1+2d_u r_y) \end{pmatrix}$$

とすると、この式は各 i に対して

$$B\mathbf{u}_i^{n+1} = \mathbf{b}_i, \quad (0 \leq i \leq Nx) \quad (4.44)$$

と書くことができる連立一次方程式を解く問題に帰着される。ただし、斉次 Neumann 境界条件の離散化 (4.37) より、 \mathbf{b}_i の各要素は

$$\mathbf{b}_0 = \begin{cases} b_0 = \frac{1}{2} \left(u_{0,0}^{n+\frac{1}{2}} + d_u r_x (2u_{1,0}^{n+\frac{1}{2}} - 2u_{0,0}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{0,0}^{n+\frac{1}{2}}, v_{0,0}^{n+\frac{1}{2}}) \right), \\ b_j = \left(u_{0,j}^{n+\frac{1}{2}} + d_u r_x (2u_{1,j}^{n+\frac{1}{2}} - 2u_{0,j}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{0,j}^{n+\frac{1}{2}}, v_{0,j}^{n+\frac{1}{2}}) \right), \quad (1 \leq j \leq N_y - 1), \\ b_{N_y} = \frac{1}{2} \left(u_{0,N_y}^{n+\frac{1}{2}} + d_u r_x (2u_{1,N_y}^{n+\frac{1}{2}} - 2u_{0,N_y}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{0,N_y}^{n+\frac{1}{2}}, v_{0,N_y}^{n+\frac{1}{2}}) \right) \end{cases} \quad (4.45)$$

$$\mathbf{b}_i = \begin{cases} b_0 = \frac{1}{2} \left(u_{i,0}^{n+\frac{1}{2}} + d_u r_x (u_{i+1,0}^{n+\frac{1}{2}} - 2u_{i,0}^{n+\frac{1}{2}} + u_{i-1,0}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{i,0}^{n+\frac{1}{2}}, v_{i,0}^{n+\frac{1}{2}}) \right), \\ b_j = \left(u_{i,j}^{n+\frac{1}{2}} + d_u r_x (u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{i,j}^{n+\frac{1}{2}}, v_{i,j}^{n+\frac{1}{2}}) \right), \quad (1 \leq j \leq N_y - 1), \\ b_{N_y} = \frac{1}{2} \left(u_{i,N_y}^{n+\frac{1}{2}} + d_u r_x (u_{i+1,N_y}^{n+\frac{1}{2}} - 2u_{i,N_y}^{n+\frac{1}{2}} + u_{i-1,N_y}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{i,N_y}^{n+\frac{1}{2}}, v_{i,N_y}^{n+\frac{1}{2}}) \right) \end{cases} \quad (4.46)$$

$$\mathbf{b}_{N_x} = \begin{cases} b_0 = \frac{1}{2} \left(u_{N_x,0}^{n+\frac{1}{2}} + d_u r_x (2u_{N_x-1,0}^{n+\frac{1}{2}} - 2u_{N_x,0}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{N_x,0}^{n+\frac{1}{2}}, v_{N_x,0}^{n+\frac{1}{2}}) \right), \\ b_j = \left(u_{N_x,j}^{n+\frac{1}{2}} + d_u r_x (2u_{N_x-1,j}^{n+\frac{1}{2}} - 2u_{N_x,j}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{N_x,j}^{n+\frac{1}{2}}, v_{N_x,j}^{n+\frac{1}{2}}) \right), \quad (1 \leq j \leq N_y - 1), \\ b_{N_y} = \frac{1}{2} \left(u_{N_x,N_y}^{n+\frac{1}{2}} + d_u r_x (2u_{N_x-1,N_y}^{n+\frac{1}{2}} - 2u_{N_x,N_y}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{N_x,N_y}^{n+\frac{1}{2}}, v_{N_x,N_y}^{n+\frac{1}{2}}) \right) \end{cases} \quad (4.47)$$

となる。各 i に対して、(4.44) を解くと

$$u_{i,j}^{n+1}, \quad (0 \leq i \leq Nx, \quad 0 \leq j \leq Ny)$$

が求まる。

4.2.5.2 周期境界条件の場合

1 段目に関しては

$$A = \begin{pmatrix} 1+2d_u r_x & -d_u r_x & 0 & 0 & 0 & -d_u r_x \\ -d_u r_x & 1+2d_u r_x & -d_u r_x & 0 & 0 & 0 \\ 0 & -d_u r_x & 1+2d_u r_x & -d_u r_x & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -d_u r_x & 1+2d_u r_x & -d_u r_x \\ -d_u r_x & 0 & 0 & 0 & -d_u r_x & 1+2d_u r_x \end{pmatrix}$$

となる。ただし A は $N_x \times N_x$ の正方行列である。この式は各 j に対して

$$A \mathbf{u}_j^{n+\frac{1}{2}} = \mathbf{b}_j, \quad 0 \leq j \leq N_y - 1 \quad (4.48)$$

という連立一次方程式を解く問題に帰着される。ただし、 \mathbf{b}_j は (4.39) より

$$\begin{cases} \mathbf{b}_0 = \left(u_{i,0}^n + d_u r_y (u_{i,1}^n - 2u_{i,0}^n + u_{i,N_y-1}^n) + \frac{\Delta t}{2} f(u_{i,0}^n, v_{i,0}^n) \right), \\ \mathbf{b}_j = \left(u_{i,j}^n + d_u r_y (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) + \frac{\Delta t}{2} f(u_{i,j}^n, v_{i,j}^n) \right), \\ \mathbf{b}_{N_y-1} = \left(u_{i,N_y-1}^n + d_u r_y (u_{i,N_y-2}^n - 2u_{i,N_y-1}^n + u_{i,0}^n) + \frac{\Delta t}{2} f(u_{i,N_y-1}^n, v_{i,N_y-1}^n) \right) \end{cases} \quad (4.49)$$

となる。ただし、 $0 \leq j \leq N_x - 1$ である。(4.49) を解くと

$$u_{i,j}^{n+\frac{1}{2}}, \quad (0 \leq i \leq N_x - 1, \quad 0 \leq j \leq N_y - 1)$$

が求まる。

2 段目に関しては

$$B = \begin{pmatrix} 1+2d_u r_y & -d_u r_y & 0 & 0 & 0 & -d_u r_y \\ -d_u r_y & 1+2d_u r_y & -d_u r_y & 0 & 0 & 0 \\ 0 & -d_u r_y & 1+2d_u r_y & -d_u r_y & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -d_u r_y & 1+2d_u r_y & -d_u r_y \\ -d_u r_y & 0 & 0 & 0 & -d_u r_y & 1+2d_u r_y \end{pmatrix}$$

とする。ただし B は $N_y \times N_y$ の正方行列である。この式は各 i に対して

$$B \mathbf{u}_i^{n+1} = \mathbf{b}_i, \quad (0 \leq i \leq N_x - 1) \quad (4.50)$$

という連立一次方程式を解く問題に帰着される。ただし、 \mathbf{b}_i は (4.39) より

$$\begin{cases} \mathbf{b}_0 = \left(u_{0,j}^{n+\frac{1}{2}} + d_u r_x (u_{1,j}^{n+\frac{1}{2}} - 2u_{0,j}^{n+\frac{1}{2}} + u_{N_x-1,j}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{0,j}^{n+\frac{1}{2}}, v_{0,j}^{n+\frac{1}{2}}) \right), \\ \mathbf{b}_i = \left(u_{i,j}^{n+\frac{1}{2}} + d_u r_x (u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{i,j}^{n+\frac{1}{2}}, v_{i,j}^{n+\frac{1}{2}}) \right), \\ \mathbf{b}_{N_x-1} = \left(u_{N_x-1,j}^{n+\frac{1}{2}} + d_u r_x (u_{N_x-2,j}^{n+\frac{1}{2}} - 2u_{N_x-1,j}^{n+\frac{1}{2}} + u_{0,j}^{n+\frac{1}{2}}) + \frac{\Delta t}{2} f(u_{N_x-1,j}^{n+\frac{1}{2}}, v_{N_x-1,j}^{n+\frac{1}{2}}) \right) \end{cases} \quad (4.51)$$

となる。ただし、 $0 \leq j \leq N_y - 1$ である。(4.51) を解くと

$$u_{i,j}^{n+1}, \quad (0 \leq i \leq N_x - 1, \quad 0 \leq j \leq N_y - 1)$$

が求まる。

反応拡散系の数値計算上の注意

拡散項の数値計算法に陰解法を使っているからといって、 Δt を大きく取れるわけではない。反応項の数値計算は Euler 法で計算しているので Δt を大きく取ると Euler 法によって発散してしまう !!

4.2.5.3 問題

問題 1. 拡張した FitzHugh-Nagumo 方程式 [27]

$$\begin{cases} \frac{\partial u}{\partial t} = d_u \Delta u + \frac{1}{\varepsilon} (u(1-u)(u-a) - v), \\ \frac{\partial v}{\partial t} = d_v \Delta v + u - \gamma v. \end{cases} \quad t > 0, \quad x \in (0, L) \times (0, L) \equiv \Omega \quad (4.52)$$

ただし、 $0 < a < 1/2$.

$\gamma = 1.0, \varepsilon = 0.001, a = 0.125, d_u = 1.0, d_v = 0.1, L = 20.0$ において数値計算を行い、スパイラル波を見つけなさい。そして d_v を大きくとると (例えば、 $d_v = 10.0$) どうなるか数値実験してみなさい。ただし、境界条件は斉次 Neumann 境界条件

$$\frac{\partial u}{\partial x}(t, x, y) = \frac{\partial u}{\partial y}(t, x, y) = 0, \quad \frac{\partial v}{\partial x}(t, x, y) = \frac{\partial v}{\partial y}(t, x, y) = 0, \quad t > 0, x \in \partial\Omega$$

と周期境界条件の両方の場合とする。

第5章 可視化

数値計算結果を可視化することは数値シミュレーションを行う上でも、方程式に対するイメージを掴むためにも非常に重要です。結果をすばやく正確に可視化できれば数値シミュレーションの効率もグッと上昇するでしょう！

5.1 数値データの可視化

(1) できあいのプログラムを用いる

- GNUPLOT → フリーソフト、最近ではグラデーションも表示できる。
- Mathematica → 有料でしかも高価。
- AVS → 有料（欲しいけど買えない！）。

長所：定型的なグラフならば美しくかつ楽に出せる。

短所：シミュレーションを走らせながらモニターするには向かない¹。楽な反面、かゆいところに手がとどかない。

(2) Graphic Library を用いる

- OpenGL, MesaGL 系 ([18],[19]) + glut [20].
- GLSC 系 (手作り, 広島大学の小林亮先生らが作ったもの)([21, 22]).
- GLSC3D(手作り, 北大電子研の秋山先生らが作った GLSC の 3D 対応, OpenGL ベースで書かれている ([23]). 詳しい解説 PDF マニュアルがある ([24]).

長所：シミュレーションプログラムを走らせながらモニターするのに便利²。細かいところまで自分の思うようにグラフが出せる。

短所：プログラムを書く手間がかかる。

以下の解説には **GNUPLOT ver.4.0** 以上と **ImageMagick** がインストールされていなければなりません。入っていない場合は前もってインストールしてください。Cygwin はすべて入っています。Linux だと GNUPLOT はパッケージインストールする必要があります。

5.2 必要なソフトウェアのインストール

フルパッケージの Cygwin をインストールしてもインストールされないソフトウェアをインストールする手順を解説します。以下のソフトウェアは Linux にもインストールする必要があります。Mac の場合は、macports をインストールしておいてください。

¹GNUPLOT を使えばシミュレーション結果をモニターすることがお手軽にできるが、2次元の描画は時間コストが高い。1次元グラフなら GNUPLOT を用いるのが一番かもしれない

²表示する時間コストが少ないという点で有利であるという意味

5.2.1 Mac OS X の計算機環境を作る

Mac OS X 上で数値計算をするためには開発環境のインストールが必要です。

5.2.1.1 Xcode のインストール

Xcode をインストールについては、以下のサイトを見るとよいです。

```
http://www.cse.kyoto-su.ac.jp/~oomoto/lecture/program/tips/Xcode_install/index.html#Xcode_install
```

Xcode のパッケージは Mac App Store にあります。Apple ID を作ればダウンロードできます。
注意すべき点: Unix のコマンドラインで使う make 等のコマンドが見つからない時は、Xcode の Preference → Download → Command Line Toos をインストールしてください。

5.2.1.2 Mac Ports のインストール

次に、Mac Ports をインストールします。Mac Ports の公式サイト

```
http://www.macports.org/
```

から OS のバージョンにあった Mac Ports をダウンロードし、pkg を実行します。インストーラーの指示どおりにインストールします。

```
> more ~/.profile
```

として、環境設定が次のようになっているか確認してください。

```
export PATH=/opt/local/bin:/opt/local/sbin:$PATH
export MANPATH=/opt/local/share/man:/opt/local/man:$MANPATH
```

なってなければ、修正してください。

```
> source ~/.profile
```

として、次のようにパスが通っているか確認してください。

```
> port version
Version: 2.1.3
```

使い始める前に、MacPorts 自身のアップデートと ports リストの更新をしてください。

```
> sudo port selfupdate
> sudo port sync
```

これで Mac Ports のインストール完了です。

5.2.1.3 gfortran のインストール

```
http://r.research.att.com/tools/  
あるいは  
http://cran.r-project.org/bin/macosx/tools/
```

から gfortran-4.2.3.dmg をダウンロードしてインストールします。

5.2.1.4 gnuplot のインストール

```
> sudo port selfupdate  
> sudo port install gnuplot
```

5.2.1.5 OpenGL 関連のインストール

OpenGL 関連で必要なものをインストールします。

```
/opt/X11/include/GL
```

の中に、freeglut.h osmesa.h がインストールされていない場合は以下の方法でインストールしてください。

```
> sudo port selfupdate  
> sudo port install mesa  
> sudo port install glw  
> sudo port install freeglut
```

5.2.2 ImageMagick のインストール

Mac OS X の場合

```
> sudo port -d selfupdate  
> sudo port -d sync  
> sudo port install ImageMagick
```

Linux(Cent OS 6.4) の場合は root になって

```
# yum search ImageMagick
とすると
ImageMagick-c++.i686 : ImageMagick Magick++ library (C++ bindings)
ImageMagick-c++.x86_64 : ImageMagick Magick++ library (C++ bindings)
ImageMagick-c++-devel.i686 : C++ bindings for the ImageMagick library
ImageMagick-c++-devel.x86_64 : C++ bindings for the ImageMagick library
ImageMagick-devel.i686 : Library links and header files for ImageMagick app
                        : development
ImageMagick-devel.x86_64 : Library links and header files for ImageMagick app
                        : development
ImageMagick-doc.x86_64 : ImageMagick html documentation
ImageMagick-perl.x86_64 : ImageMagick perl bindings
ImageMagick.i686 : An X application for displaying and manipulating images
ImageMagick.x86_64 : An X application for displaying and manipulating images
とでてくるので
# yum install ImageMagick-devel.x86_64
とすればインストールできる.
```

Cygwin の場合 :

Windows では ImageMagick のバイナリーが用意されているので, Cygwin 上では用意する必要がない.

5.2.3 ffmpeg のインストール

Mac OS X の場合

```
> sudo port -d selfupdate
> sudo port -d sync
> sudo port install ffmpeg
```

Linux(CentOS 7.x) の場合 :

以下の方法が便利です.

```
> sudo yum -y install epel-release
> sudo rpm -Uvh http://li.nux.ro/download/nux/dextop/el7/x86_64/
nux-dextop-release-0-5.el7.nux.noarch.rpm
> sudo yum install ffmpeg
```

Cygwin を使う場合 :

Windows では ffmpeg のバイナリーが用意されているので, Cygwin 上では用意する必要がない.

5.2.4 mpeg_encode のインストール

このソフトウェアはもう古いので使わない.

Cygwin 上でのインストールは次に行う.

```
$ tar -xvzf mpeg_encode_cygwin_src.tar.gz
$ cd mpeg_encode/jpeg
$ make
$ cd ../
$ make
warning が一杯出るが, mpeg_encode.exe はできている.
$ cp mpeg_encode.exe /usr/local/bin/.
としてインストール終わり.
```

5.2.5 mpeg_play のインストール

このソフトウェアはもう古いので使わない。
Cygwin 上でのインストールは次のように行う。

```
$ tar -xvzf mpeg_play_cygwin_src.tar.gz
$ cd mpeg_play
$ make
warning が一杯出るが, mpeg_play.exe はできている.
$ cp mpeg_play.exe /usr/local/bin/.
としてインストール終わり.
```

5.3 必要になるかもしれないソフトウェアのインストール

5.3.1 OpenGL のインストール

Cygwin に OpenGL がインストールされていない場合や glut がインストールされていない場合は自分でインストールしてみよう！

```
$ tar -xvzf MesaLib-7.2.tar.gz
$ tar -xvzf MesaGLUT-7.2.tar.gz
$ tar -xvzf MesaDemos-7.2.tar.gz
$ cd Mesa-7.2
$ ./configure
$ make
最後に Err が出て終わるが, OpenGL のコンパイルは出来ている.
少なくとも私が使った範囲では動いてくれた.
$ make install
としてインストール終わり.
最後に Err が出たら確認のため
$ cd lib
$ cp * /usr/local/lib/.
とすればインストールが出来ているはず.
```

Linux(Cent OS 6.3) の場合: OSMesa がインストールされていないので yum を使ってインストールする。

```
# yum search OSMesa
とすると
mesa-libOSMesa.i686 : Mesa offscreen rendering libraries
mesa-libOSMesa.x86_64 : Mesa offscreen rendering libraries
mesa-libOSMesa-devel.i686 : Mesa offscreen rendering development package
mesa-libOSMesa-devel.x86_64 : Mesa offscreen rendering development package
と出力されるので
# yum install mesa-libOSMesa-devel.x86_64
とすればよい.
```

5.3.2 GLSC のインストール

私は主に GLSC は鳥瞰図を論文に載せたいときに使っている³.
Cygwin 上でのインストールは次のように行う.

```
$ tar -xvzf glsc-3.5-patch.tar.gz
このソースファイルは正規の GLSC から鳥瞰図を白塗りに変更したものである.
$ cd glsc-3.5
$ make
最後に Err が出て終わるが, GLSC のコンパイルは出来ている.
$ make install
でインストール完了.
確認したい場合は
$ cd test
$ ./ctestd.exe
としてみよう. GLSC が実行されるはずですよ.
```

5.3.3 vlc のインストール

動画再生で困った場合は vlc をインストールすれば, ほぼ解決するので一応解説しておく.
Cygwin の場合: 以下のサイトに Windows 用のバイナリーファイルが用意されているので, Cygwin にインストールする必要はありません.
Mac OS X の場合: Mac 用にバイナリーファイルが用意されているので, 以下のサイトからインストールしてください.

```
http://www.videolan.org/vlc/
```

Linux(Cent OS) の場合: 次の手順に従ってインストールしてください.
RPMforge リポジトリの導入 (参照: <http://ufuso.jp/wp/?p=9979>)

³計算しながら結果を表示するためなら GNU PLOT で十分である.


```
# wget http://dag.wieers.com/rpm/packages/RPM-GPG-KEY.dag.txt
← RPMforge の GPG キーをダウンロード
# rpm --import RPM-GPG-KEY.dag.txt
← RPMforge の GPG キーをインストール
# rm -f RPM-GPG-KEY.dag.txt
← RPMforge の GPG キーを削除
# vi /etc/yum.repos.d/rpmforge.repo
rpmforge.repo ファイルの作成する。以下の内容を書き込む。
[rpmforge]
name=RPMforge RPM repository for Red Hat Enterprise Linux
baseurl=http://ftp.riken.jp/Linux/dag/redhat/el6/en/$basearch/rpmforge/
gpgcheck=1
enabled=0
```

yum を使って vlc のインストール (参照: <http://ufuso.jp/wp/?p=10301>)

```
# yum --enablerepo=rpmforge install vlc
```

5.4 GNUPLOT による数値計算結果の可視化 [15]

数値シミュレーションでは数値計算結果を分かりやすく表現するも重要である。また、解の遷移過程や漸近挙動のパラメータ依存性をすばやく調べるためには数値計算と同時に可視化することが望ましい。そのためには数値計算プログラム以外のプログラムを組み合わせることが必要となり、初心者には困難な作業と必要なる。そこで GNUPLOT を使うことによって可視化プログラム作成コストを少なくする。数値計算中の途中結果データを GNUPLOT で表示できるように出力することで、ある程度計算過程での数値計算結果を調べることができる。GNUPLOT の詳しい使い方は [15] を見て下さい。例えば、数値計算データが $(t, u(t))$ の順番にならんでいたら、次のようにすればよい:

```
% gnuplot
gnuplot> plot 'sol_ODE.dat' using 1:2 with lines
```

数値データの形式が $(t, x, u(t, x))$ のとき

```
% gnuplot
gnuplot> splot 'sol_1DRD.dat' using 1:2:3 with lines
```

5.4.1 プリンターへの出力

プリンターへの出力は例えば次のようにする。

```
% gnuplot
# gnuplot の起動
gnuplot> plot 'sol_ODE.dat' using 1:2 with lines
# この時点まではデータの可視化と同じ。
gnuplot> set term postscript
# この時点でウィンドウが消える
gnuplot> set output 'fig01.ps'
# fig01.ps というファイルを開く
gnuplot> replot
# plot 'sol_ODE.dat' using 1:2 with lines と同じ
# sol_ODE.dat に対するグラフの postscript データが fig01.ps に出力される
gnuplot> quit
# gnuplot の終了
% lpr fig01.ps
# fig01.ps をプリンターに出力する
```

上記の作業を繰り返し行う場合には、次のようなスクリプトファイル (g_plot) を作りそれを繰り返し使う方が効率がよい。

```
plot 'sol_ODE.dat' using 1:2 with lines
pause -1 "Hit Return key"
# 正しくデータがグラフ表示されているか確認する
# リターンキーを押せば次に進む
set nokey
# グラフの中のタイトルを消去する
# タイトルが不要の場合は書いおけばよい
set term postscript
set output 'fig01.ps'
replot
```

そして

```
% gnuplot g_plot
```

とすれば、fig01.ps というファイルができて gnuplot が自動的に終了するので、満足いくグラフならプリンターに出力すればよい。また、fig01.ps の内容を確認したいなら以下のようにすればよい。

```
% gs fig01.ps
```

5.4.2 GNUPLOT を使ってパラパラ動画を作ろう

数値データの構造が

時間 空間 u の値 v の値 ...

となっている数値データ (ファイル名: sample.data.dat) に対して、各時間ステップごとの数値データファイルに分割する perl スクリプト (ファイル名: datacut) を作る:

```

#! /usr/bin/perl

# データを分割するディレクトリ
$data_dir = 'cut';

$i = 0;
open(GA, ">$data_dir/$i");
open(HA, ">./plot1d");

print HA "set yrange[0:20]\n";
print HA "plot './$data_dir/$i' using 1:2 with lines,\\"
'./$data_dir/$i' using 1:3 with lines \n";

$j = 0;
while(<>) {
    $a = $&.$' if /\d/;
    @b = split(/\s+/, $a);
    if($b[0] != $i){
        $i = $b[0];
        $j += 1;
        print HA "plot './$data_dir/$j' using 1:2 with lines,\\"
'./$data_dir/$j' using 1:3 with lines \n";
        close(GA);
        open(GA, ">$data_dir/$j");
        print GA " $b[1] $b[2] $b[3]\n";
    }
    else {
        print GA " $b[1] $b[2] $b[3]\n";
    }
    $i = $b[0];
}
print HA "pause -1 'Hit return key !!' \n";
close(GA); close(HA);

```

このスクリプトを用いて次の操作を行うことでパラパラ動画を作ることができます⁴。

```

% ./datacut < sample_data.dat
# ./cut の中に分割された数値データファイルが作られる。
# ./ に plot1d という gnuplot のスクリプトファイルが作られる。
% gnuplot plot1d
# 動画のように見えますよね! ?

```

5.4.3 GNUPLOT を使って JPEG 形式の画像ファイルを作る

GNUPLOT のバージョンが 4.0 以上なら GNUPLOT で表示された画像を JPEG フォーマットのファイルとして保存することができる。

⁴事前に cut というディレクトリを作っておかなければなりません。

```
% gnuplot
# gnuplot の起動
gnuplot> plot 'sol_ODE.dat' using 1:2 with lines lw 3
# この時点まではデータの可視化と同じ.
# lw は線の太さを変更するオプション. デフォルトは1.
gnuplot> set term jpeg
# この時点でウィンドウが消える
gnuplot> set output 'fig01.jpg'
# fig01.jpg というファイルを開く
gnuplot> replot
# plot 'sol_ODE.dat' using 1:2 with lines と同じ
# sol_ODE.dat に対する jpeg フォーマットの画像が fig01.jpg に出力される
gnuplot> quit
# gnuplot の終了
% display fig01.jpg
# fig01.jpg が表示される.
```

JPEG ファイルを表示して見るとわかるが、JPEG 形式の画像ファイルは劣化する。線画ではこの劣化が致命傷となる場合があるので、JPEG 形式の画像ファイルは2次元以上の数値データに対するグラデーション表示に使うべきである。どうしても線画を JPEG 形式で保存したい場合は線幅を太くするとよい。上記の例では lw オプションで線を太くしている。表示する線を太くするには次のように使えばよい：

```
% gnuplot
# gnuplot の起動
gnuplot> plot 'sol_ODE.dat' using 1:2 with lines lw 3
# lw オプションは線の太さを変更するデフォルトは1である.
```

5.4.4 GNUPLOT を使って GIF 形式の画像ファイルを作る

前節と同様にバージョン 4.0 以上の GNUPLOT で表示された画像を gif 形式のファイルとして保存することができる。ただし、Fedora Core5 の GNUPLOT ver4.0 では GNUPLOT をインストールし直す必要があります。

```
% gnuplot
# gnuplot の起動
gnuplot> plot 'sol_ODE.dat' using 1:2 with lines
# この時点まではデータの可視化と同じ.
# GIF 形式にする場合は
gnuplot> set term gif
# この時点でウインドウが消える
gnuplot> set output 'fig01.gif'
# fig01.gif というファイルを開く
gnuplot> replot
# plot 'sol_ODE.dat' using 1:2 with lines と同じ
# sol_ODE.dat に対する gif フォーマットの画像が fig01.gif に出力される
gnuplot> quit
# gnuplot の終了
% display fig01.gif
# fig01.gif が表示される.
```

線画を JPEG 形式で圧縮するとどうしても解像度が落ちるので GIF 形式で保存する方がよい。後から PPT に貼り付けたり動画化するのにも便利です。

5.4.5 GNUPLOT を使って PNG 形式の画像ファイルを作る

ライセンスの関係で GIF 形式で保存できない場合は、PNG 形式で保存するとよい。この場合も線画が綺麗に保存できる。

```
% gnuplot
# gnuplot の起動
gnuplot> plot 'sol_ODE.dat' using 1:2 with lines
# この時点まではデータの可視化と同じ.
# PNG 形式にする場合は
gnuplot> set term png
# この時点でウインドウが消える
gnuplot> set output 'fig01.png'
# fig01.png というファイルを開く
gnuplot> replot
# plot 'sol_ODE.dat' using 1:2 with lines と同じ
# sol_ODE.dat に対する PNG フォーマットの画像が fig01.png に出力される
gnuplot> quit
# gnuplot の終了
% display fig01.png
# fig01.png が表示される.
```

5.5 GNUPLOT と ImageMagick による動画の作成

5.5.1 アニメーション GIF の作成

この節では、5.4.4,5.4.5 で作った GIF 画像や PNG 画像の時系列画像ファイルを用いてアニメーション GIF ファイルの作成を行う。第 5.4.2 節で用いた perl スクリプトを改良して次のようなスクリプト (data2image) を作る：

```
#!/usr/bin/perl

# データを分割するディレクトリ
$data_dir = 'cut';

# 画像を保存するディレクトリ
$image_dir = 'image';

# -> png とすれば PNG 形式の画像となる。
$format = 'gif';

$i = 0;

open(GA, ">$data_dir/$i");
open(HA, ">./plot_$format");
print HA "set yrange[0:20]\n";
print HA "set term $format\n";
print HA "set output './$image_dir/fig0000.$format' \n";
print HA "plot './$data_dir/$i' using 1:2 with lines,\n\
 './$data_dir/$i' using 1:3 with lines \n";
$j = 0;
while(<>) {
    $a = $&.$' if /\d/;
    @b = split(/\s+/, $a);
    if($b[0] != $i){
        $i = $b[0];
        $j += 1;
        if($j <= 9){
            print HA "set output './$image_dir/fig000$j.$format'\n";
        }
        elsif($j >= 10 && $j <= 99){
            print HA "set output './$image_dir/fig00$j.$format'\n";
        }
        else{
            print HA "set output './$image_dir/fig0$j.$format'\n";
        }
        print HA "plot './$data_dir/$j' using 1:2 with lines,\n\
 './$data_dir/$j' using 1:3 with lines \n";
        close(GA);
        open(GA, ">$data_dir/$j");
        print GA " $b[1] $b[2] $b[3]\n";
    }
    else {
        print GA " $b[1] $b[2] $b[3]\n";
    }
    $i = $b[0];
}
close(GA);
```

```
close(HA);
```

このスクリプトを用いて⁵

```
% ./data2image < sample_data.dat
# ./cut の中に分割された数値データファイルが作られる。
# ./ に plot_gif という gnuplot のスクリプトファイルが作られる。
% gnuplot plot_gif
# ./image の中に fig0000.gif~fig0xxx.gif の GIF ファイルができる
```

もしも PNG 形式の画像を作るのだったら data2image ファイルにおいて

```
$format = 'gif';
を
$format = 'png';
と書き換えればよい。
```

以下のように実行すると PNG ファイルができる：

```
% gnuplot plot_png
# ./image の中に fig0000.png~fig0xxx.png の PNG ファイルができる
```

ImageMagick がインストールされているならば、次のようにすればアニメーション GIF を作成することができます：

```
% convert -delay 20 ./image/fig*.gif animation.gif
# -delay は再生速度を調整するオプションです。
```

PNG ファイルならば

```
% convert -delay 20 ./image/fig*.png animation.gif
# -delay は再生速度を調整するオプションです。
```

とすることでアニメーション GIF を作成することができます。

できた animation.gif は animate を使って再生することができます⁶：

```
% animate animation.gif
```

もし、animate が見つからない場合は ImageMagick をインストールしてください。

5.5.2 MPEG 動画の作成

前節のスクリプトファイル (data2image) を次のように書き換える：

```
#! /usr/bin/perl
```

⁵事前に cut と image というディレクトリを作っておかなければなりません。

⁶Windows で再生する場合は再生ソフトをインストールする必要がある。例えば QuickTime があれば再生できます。

```
# データを分割するディレクトリ
$data_dir = 'cut';

# 画像を保存するディレクトリ
$image_dir = 'image';

# 修正したのはこの部分
$format = 'jpeg';

$i = 0;

open(GA, ">$data_dir/$i");
open(HA, ">./plot_format");
print HA "set yrange[0:20]\n";
print HA "set term $format\n";
print HA "set output './$image_dir/fig0000.$format' \n";
print HA "plot './$data_dir/$i' using 1:2 with lines,\n\
 './$data_dir/$i' using 1:3 with lines \n";
$j = 0;
while(<>) {
    $a = $_ if /\d/;
    @b = split(/\s+/, $a);
    if($b[0] != $i){
        $i = $b[0];
        $j += 1;
        if($j <= 9){
            print HA "set output './$image_dir/fig000$j.$format'\n";
        }
        elsif($j >= 10 && $j <= 99){
            print HA "set output './$image_dir/fig00$j.$format'\n";
        }
        else{
            print HA "set output './$image_dir/fig0$j.$format'\n";
        }
        print HA "plot './$data_dir/$j' using 1:2 with lines,\n\
 './$data_dir/$j' using 1:3 with lines \n";
        close(GA);
        open(GA, ">$data_dir/$j");
        print GA " $b[1] $b[2] $b[3]\n";
    }
    else {
        print GA " $b[1] $b[2] $b[3]\n";
    }
    $i = $b[0];
}
close(GA);
close(HA);
```

この perl スクリプトを用いて


```
% ./data2image < sample_data.dat
# ./cut の中に分割された数値データファイルが作られる。
# ./ に plot_jpeg という gnuplot のスクリプトファイルが作られる。
% gnuplot plot_jpeg
# ./image の中に fig0000.jpeg~fig0xxx.jpeg の JPEG ファイルができる
```

5.5.3 動画作成 その1 (秋山さん作成)

Linux 上で前節のようにして作られた時系列 JPEG 画像ファイルをまとめて MPEG 等の動画ファイルにする方法を解説する。ここでは ffmpeg を使って動画を作成します。Test というディレクトリにファイル名 000001.jpg から 000099.jpg が入っているとします。output.mpeg という mpeg1 形式の動画をつくと仮定します。このとき、もっとも簡単な方法は

```
> ffmpeg -i ./Test/0000%2d.jpg output.mpeg
```

であり、動画の再生は次のようにすればよい。

```
> animate output.mpeg
```

しかしながら、上記のアニメーションでは動画の劣化が激しいため、秋山さんが作ってくれた次のパラメータファイル (makemov_big, makemov_light) を使って動画作成をするとよい。使い方は次の通りである。

```
> ./makemov_big Test
```

Test は連番画像が入っているディレクトリであり、Test.mov という画像ができる。動画は MOV ファイル (QuickTime) 形式となっており、Window と Mac OS 上では問題なく再生できる。また、Linux (Cent OS 6.4) 上では標準的な再生ソフト Dragon Player を使うか vlc を使えば再生できる。

5.5.3.1 パラメータファイル (makemov_big)

```
#!/bin/bash
# Copyright (c) Masakazu Akiyama, 2010/2/12 ALL RIGHTS RESERVED
# 再配布 OK、改良 OK、変更点をメールください。

if [ $# -ne 1 ]
then
echo "Please type 'makemov folder'"
exit
fi

#if [ ! -e /opt/local/bin/ffmpeg ]
#then
# echo "Please install ffmpeg from macports!!(http://www.macports.org/)"
# exit
#fi

#if [ ! -e identify ]
#then
#echo "Please install ImageMagick from macports!!(http://www.macports.org/)"
#exit
```

```

#fi

folder=$1
folder='basename $folder'

#CPU_CORES=2
CPU_CORES=$(/usr/bin/getconf _NPROCESSORS_ONLN)

#judge extension & numfilenum
cd ${folder}
firstfile='ls -1 | head -1'
cd ..

hoge=$firstfile
foo=${#hoge}
bar='echo ${hoge} | sed -e 's/\.[0-9a-zA-Z]*$//' | wc -c'
ext='echo ${hoge} | cut -b ${bar}-${foo}'
numfilenum='expr $bar - 1'

#GetfileSize
lastfile='ls -1 "${folder}"/ | tail -1'
width='identify -format "%w" "${folder}/${lastfile}"'
hight='identify -format "%h" "${folder}/${lastfile}"'
if [ 'expr $width % 2' -eq 1 ]
then
    width='expr $width + 1'
fi
if [ 'expr $hight % 2' -eq 1 ]
then
    hight='expr $hight + 1'
fi
ffmpeg -y -threads ${CPU_CORES} -i "${folder}"/"%0${numfilenum}d${ext}"
-vcodec mjpeg -qscale 0 -s "${width}"x"${hight}" "${folder}".mov

```

5.5.3.2 パラメータファイル (makemov.light)

```

#!/bin/bash
# Copyright (c) Masakazu Akiyama, 2010/2/12 ALL RIGHTS RESERVED
# 再配布 OK、改良 OK、変更点をメールください。

if [ $# -ne 1 ]
then
echo "Please type 'makemov folder'"
exit
fi

#if [ ! -e /opt/local/bin/ffmpeg ]
#then
#    echo "Please install ffmpeg from macports!!(http://www.macports.org/)"
#    exit
#fi
#if [ ! -e identify ]
#then
#echo "Please install ImageMagick from macports!!(http://www.macports.org/)"

```

```

#exit
#fi

folder=$1
folder='basename $folder'

#CPU_CORES=2
CPU_CORES=$(/usr/bin/getconf _NPROCESSORS_ONLN)

#judge extension & numfilenum
cd ${folder}
firstfile='ls -1 | head -1'
cd ..

hoge=$firstfile
foo=${#hoge}
bar='echo ${hoge} | sed -e 's/\.[0-9a-zA-Z]*$//' | wc -c'
ext='echo ${hoge} | cut -b ${bar}-${foo}'
numfilenum='expr $bar - 1'

#GetfileSize
lastfile='ls -1 "${folder}"/ | tail -1'
width='identify -format "%w" "${folder}"/"${lastfile}"'
hight='identify -format "%h" "${folder}"/"${lastfile}"'
if [ 'expr $width % 2' -eq 1 ]
then
    width='expr $width + 1'
fi
if [ 'expr $hight % 2' -eq 1 ]
then
    hight='expr $hight + 1'
fi

ffmpeg -y -threads ${CPU_CORES} -i "${folder}"/"%0${numfilenum}d${ext}"
-vcodec libx264 -f mp4 -s "${width}"x"${hight}" -level 30 -crf 30
-coder 0 -r 30000/1001 -flags +loop -partitions +parti4x4 -me_method dia
-subq 1 -me_range 1 -g 150 -qcomp 0.7 -keyint_min 25 -sc_threshold 0
-i_qfactor 0.71 -b_strategy 0 -qmin 9 -qmax 41 -rc_eq 'blurCplx^(1-qComp)'
-qdiff 4 -i_qfactor 0.714286 -bf 0 -bidir_refine 1 -refs 1 "${folder}".mov

```

5.5.4 動画作成 (MPEG1) その2

この方法は多少古くなったので、そろそろ止めてもいいかなあ…って思いますが、書き残しておきます。MPEG 化するためにパラメーターファイル (mpeg_encode_para) を適切に書くことによって、次のように実行すれば MPEG の動画ファイルが作成される：

```
% mpeg_encode mpeg_encode_para
```

ここでは、sample.mpg が作成されたとすると、その動画を再生するには

```
% mpeg_play sample.mpg
```

とするだけでよい⁷。パラメーターファイルに関しては、

⁷ここで作成した sample.mpg は Windows 上でも再生できるので、PowerPoint 等に貼り付けて講演するときに有効である。

“http://www.not-enough.org/abe/manual/command/mpeg_encode.html”
を参照のこと!!

5.5.4.1 パラメータファイル (mpeg_encode_para) のサンプル

```
PATTERN          IBBPBBPBBPBBPBBP
# this must be one of {YUV, PPM, PNM}
BASE_FILE_FORMAT      PPM

#INPUT_CONVERT      convert -size 320x240 * ppm:-
INPUT_CONVERT        convert -size 640x480 * ppm:-
# 動画のサイズを決定する.

GOP_SIZE           30
SLICES_PER_FRAME   1

# 出力するファイル名を指定する
OUTPUT             sample.mpeg

# 画像ファイルの存在するディレクトリーを指定する
INPUT_DIR          image

# 画像ファイルの記述始まり
INPUT
# 拡張子が jpg なら Image*.jpg と書き換えること
Image*.jpeg        [0000-0100]
# Image0001.jpeg...Image0100.jpeg という画像ファイルがエンコードされる
# 画像ファイルの記述始まり
END_INPUT

# this must be one of {FULL, HALF}
PIXEL              HALF
RANGE              10
# this must be one of {EXHAUSTIVE, SUBSAMPLE, LOGARITHMIC}
PSEARCH_ALG        LOGARITHMIC
# this must be one of {SIMPLE, CROSS2, EXHAUSTIVE}
BSEARCH_ALG        CROSS2

IQSCALE            8
PQSCALE            10
BQSCALE            25

# this must be ORIGINAL or DECODED
REFERENCE_FRAME     ORIGINAL
FRAME_RATE          30
```

5.5.5 動画作成 (MPEG1) その3

ImageMagick を使って MPEG 動画を作ることも可能です。このときは事前に `mpeg2vidcodec` をインストールする必要があります⁸

Cygwin 上では

⁸ mpeg2vidcodec は rpm 形式で配布されているのでインストールは簡単です。mpeg_encode や mpeg_play をインストールするよりは簡単なので、mpeg_encod をインストールすることができない人にはこちらがお勧めです。

```
% rpm -ivh mpeg2vidcodec-1.2-5.src.rpm
```

Fedora8 上では

```
% rpm -ivh mpeg2vidcodec-1.2-1.i386.rpm
```

でインストールできます。

インストールができたなら次のように入力してください。MPEG ファイルを作ることができます。

```
% convert *.jpeg test.mpg
```

この方法で MPEG ファイルを作るときは、静止画が BMP でも PPM でも何でも大丈夫です。

この場合も簡単に動画を作ることができますが、Cygwin 上で実行したところ、MPEG ファイルの画質の劣化がはっきりと見てとれました。

5.6 GNUPLOT を C 言語から呼び出す [16, 17]

Fortran から呼び出して使う場合、いちいち GNUPLOT を終了させなければならないので画面がチラつく。ちらつきが嫌な場合は、メインルーチン部分も C 言語で書いた方がよい。具体的にはパイプ機能を用いて行います。次の例は [16] に記載されていたサンプルを改良したものです (sample_gnuplot01.c)。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char **argv)
{
    FILE *gpid;
    int i;
    double x;

    /* gnuplot を開く */
    gpid = popen("gnuplot","w");

    /* gnuplot が見つからない場合のエラー処理 */
    if(gpid == NULL){
        printf("I can't find gnuplot.\n");
        exit(1);
    }

    /* gnuplot でグラフを書く */
    for(i=1; i < 100; i++){
        fprintf(gpid, "set xrange [0:4]\n");
        x = 0.03*i;
        fprintf(gpid, "plot sin(x - %f*pi)\n", x);
    }
}
```

```

/* 最後の画面を10秒静止する */
/* C言語から呼び出すときは pause -1 が効かないので注意すること */
    fprintf(gpid, "pause 10\n");

    pclose(gpid);
}

```

サンプルプログラムのコンパイル方法と実行方法 (run_sample.sh を参照してください)

```

$ gcc -O3 -Wall sample_gnuplot01.c -o a.out -lm
$ ./a.out # 実行

```

上記のプログラムを改造すれば、数値計算を実行しながら GNUPLOT で計算結果を可視化することができる。詳しくは [16] を参考にしてください。以下のサンプルプログラムは熱方程式の近似解を陽解法により求めながら、GNUPLOT を用いて数値解を表示するプログラムである (sample_gnuplot02.c) :

```

include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define dt 0.0005
#define N 10
#define T 1000

int main()
{
    double u[N+1], new_u[N+1];
    int i, j;
    double dx = 1. / (float)N;
    double r = dt / (dx * dx);
    FILE *gpid;

    /* gnuplot を開く */
    if((gpid = popen("gnuplot -persist", "w")) == NULL) {
        printf("I can't find gnuplot.\n");
        exit(1);
    }

    /* y 方向の描画領域を固定 */
    fprintf(gpid, "set yrange[-4:4]\n");

    for(i = 0; i <= N; i++){
        u[i] = 4 * cos(M_PI * i * dx);
    }
}

```

```

for(j = 0; j < T; j++){
  /* 熱方程式を陽解法で解く */
  for(i = 1; i < N; i++){
    new_u[i] = r * u[i - 1] + (1. - 2. * r) * u[i] + r * u[i + 1];
  }
  new_u[0] = 2. * r * u[1] + (1. - 2. * r) * u[0];
  new_u[N] = 2. * r * u[N - 1] + (1. - 2. * r) * u[N];

  for(i = 0; i <= N; i++){
    u[i] = new_u[i];
  }

  /* gnuplot でグラフを描く */
  fprintf(gpid, "plot '-' w l t '%d'\n", j);
  for(i = 0; i <= N; i++){
    fprintf(gpid, "%lf %lf\n", i * dx, u[i]);
  }
  fprintf(gpid, "e\n");
  fflush(gpid);
}

/* 最後の画面を 10 秒静止する */
/* C 言語から呼び出すときは pause -1 が効かないので注意 */
fprintf(gpid, "pause 10\n");

/* gnuplot を閉じる */
pclose(gpid);

return 0;
}

```

サンプルプログラムのコンパイル方法と実行方法 (run_sample.sh を参照してください)

```

$ gcc -O3 -Wall sample_gnuplot01.c -o a.out -lm
$ ./a.out # 実行

```

5.7 GNUPLOT を用いた 2 次元データの可視化

2 次元熱方程式等の数値解を GNU PLOT で可視化する方法を解説します。数値計算データが横に x 座標, y 座標, その座標でのデータ となっているデータがあったとすると, 以下のように実行すると可視化できます。

```
gnuplot> set view map
gnuplot> set size ratio -1
gnuplot> set palette defined(1"#0000ff",2"#0080ff",3"#00ffff",4"#00ff80",
5"#00ff00",6"#80ff00",7"#ffff00",8"#ff8000",9"#ff0000")
gnuplot> splot "data" using 1:2:3 with pm3d
```

上記の意味は次の通りです。

- 1 行目：視点を x-y 平面を垂直に見下ろす位置に変更
- 2 行目：x 軸と y 軸の比が 1:1 になるように表示
- 3 行目：カラーパレットの設定
- 4 行目：カラーレットで表示

この方法が一番簡単に結果を表示できます。

5.8 OpenGL によるオフスクリーンレンダリング

大事なおまじないは次の行である。

```
...
#include "GL/osmesa.h"
...
int main()
{
    OSMesaContext ctx;
    void *buffer;
    ...
    /* Create an RGBA-mode context */
    ctx = OSMesaCreateContext( GL_RGBA, NULL );
    /* Allocate the image buffer */
    buffer = malloc( WIDTH * HEIGHT * 4 );
    /* Bind the buffer to the context and make it current */
    OSMesaMakeCurrent( ctx, buffer, GL_UNSIGNED_BYTE, WIDTH, HEIGHT );
    ...(#1)
    OSMesaDestroyContext(ctx);
}
```

オフスクリーンレンダリングはバックグラウンドで数値計算を行いながら静止画の時系列画像を作成するのに便利です⁹。従って、画像をファイルにする必要があります。その関数は次のように記述します。ここでは PPM ファイル形式で出力します。

```
save_file(FILE *f, void *buffer){
    if (f) {
```

⁹大規模計算では数値データだけを出力して、あとで可視化することをお勧めします。


```

int i, x, y;
GLubyte *ptr;
ptr = buffer;
fprintf(f,"P6\n");
fprintf(f,"# ppm-file created\n");
fprintf(f,"%i %i\n", Width,Height);
fprintf(f,"255\n");
for (y=Height-1; y>=0; y--) {
    for (x=0; x<Width; x++) {
        i = (y*Width + x) * 4;
        fputc(ptr[i], f); /* write red */
        fputc(ptr[i+1], f); /* write green */
        fputc(ptr[i+2], f); /* write blue */
    }
}
fclose(f);
}
printf("all done\n");
}

```

PPM ファイルはファイルサイズが大きくなるので ImageMagick を用いて JPEG ファイルに変換します。次のような関数を用意します。

```

void change_image_format(char *in, char *out)
{
    char string[300];
    sprintf(string, "convert %s %s", in, out);
    system(string);
}

```

そして main 関数の中で (#1) の部分のところに次のように書き込みます。

```

...
/* PPM ファイル名の決定 */
sprintf(ppmfile, "%s/%s%04d.ppm", image_dir, image_file, fn);
/* JPEG ファイル名の決定 */
sprintf(jpegfile, "%s/%s%04d.jpg", image_dir, image_file, fn);
if ((fp = fopen(ppmfile, "w")) == NULL){
    perror(ppmfile);
    exit(1);
}
/* PPM ファイルに保存 */
save_file(fp, buffer);

```

```

/* PPM file -> JPEG file */
change_image_format(ppmfile, jpegfile);
/* PPM ファイルを消去 */
unlink(ppmfile);
/* image buffer の開放*/
free(buffer);

```

詳しくはサンプルプログラム (5.9.2.1) を参照してください。ここで作られた時系列 JPEG 画像データは 5.5.2 で解説した方法で MPEG 動画に変換することができます。

5.9 可視化の参考プログラム

この節に掲載しているサンプルプログラムの実行方法は CD-ROM の Sample_Program 中の run_script, run_glse を参考にしてください。

5.9.1 GLSC による鳥瞰図の参考プログラム

このプログラム (GLSC_sample01.c) は上山大信氏 (明治大学) によるプログラムに手を加えたもの。論文に掲載する鳥瞰図を作るために使っている。

```

#include "/usr/local/include/glsc.h"
#include <stdio.h>
#define Imax    (251) /* x 軸の分点数 */
#define Jmax    (100) /* y 軸の分点数 */
#define Xleft   (0.0)
#define Xright  (1.0)
#define Ybottom (0.0) /* グラフ化時の最小値の調節 */
#define Ytop    (30.0) /* グラフ化時の最大値の調節 */
#define Jstep   (1)
#define Istep   (Imax)
#define X       (100.0)
#define Y       (200.0)
#define Xmar    (10.0)
#define Ymar    (10.0)
#define Xwid    (80.0)
#define Ywid    (180.0)
void main(int argc, char **argv)
{
    int i, j;
    int imax = Imax;
    int jmax = Jmax;
    int istep = Istep;
    int jstep = Jstep;
    FILE *fp = stdin;
    double *u, *v;

    fp = fopen(argv[1], "r");
    /* Memory allocation */
    u = (double *)malloc((imax)*(jmax)*sizeof(double));
    if(u == NULL)

```

```

{
    fprintf(stderr, "Can't allocate memory.\n", argv[0]);
    exit(1);
}
/* Read the data */
for(j = 0; j < jmax; j++)
{
    for(i = 0; i < imax; i++)
    {
        fscanf(fp, "%lf", &u[i*jmax + j]);
    }
}
g_init("Graph", X, Y);
g_device(G_BOTH);
g_hidden(10.0, 15.0, 10.0, Ybottom, Ytop,
         500.0, 0.0, 80.0,
         Xmar, Ymar, Xwid, Ywid,
         u, imax, jmax, 1,
         1, istep, jstep);
g_sleep(G_STOP);
}

```

このときの Makefile(Makefile_GLSC_sample01) は次のようになる。

```

#make
LIB = -lglscd -lX11 -lm
DIR = -I/usr/X11R6/include -I/usr/local/include \
-L/usr/X11R6/lib -L/usr/local/lib
OBJECT = GLSC_sample01.c
CC = gcc -g
all : main01
main01 : $(OBJECT)
$(CC) -o a.out $(OBJECT) $(DIR) $(LIB)

```

コンパイル方法と実行方法

```

$ make -f Makefile_GLSC_sample01
$ ./a.out sample_data_glsc01.dat
data_file_name 中は1列に並んだデータ
x 軸データが Imax と一致し, y 軸データが Jmax と一致している必要がある。
例えば, Sample_Program 内の sol01.dat を使うときは
Imax = 501, Jmax = 100 として make をやり直す。
$ make -f Makefile_GLSC_sample01
$ ./a.out sol01.dat

```

とすると, 鳥瞰図が出力される。鳥瞰図上でマウスの左ボタンを押せば終了。
最後に次のコマンドを実行。

```

$ g_out -v Graph

```

これで Graph.ps という PS ファイルができる。

これらの作業を自動で行うためのスクリプトファイル (run_glsc) を用意しておく と便利。

```
#!/bin/sh
make -f Makefile_GLSC_sample01
./a.out sample_data_glsc01.dat
g_out -v Graph
```

5.9.2 OSMesa(OpenGL) のプログラム

OpenGL の `osmesa` と ImageMagick の `convert` を使って計算しながらグラフをファイルに落すためのプログラムの例。数値計算プログラムの中からは次のように呼び出す。

C から呼び出す場合

```
mk_color_full(u, rr, gg, bb, mod, umax, umin, nx0, nx, ny0, ny);
draw_(ii, Mgx, Mgy, dgx, dgy, rr, gg, bb, "u", "image");
```

次のプログラムは上記の2つのプログラムを用いて熱方程式の数値結果を画像にして保存するサンプルプログラム (heat2d_ADI.c) です：

```
#include <stdio.h>
#include <math.h>
void mk_color_full(double (*)[], double[], double[], double[],
int, double, double, int, int, int, int);
void draw_(int, int, int, double, double, double *, double *,
double *, char *, char *);

#define TIME 500 /* 計算回数 */
#define CUT 10 /* 何回毎に出力するか */
#define DT 0.00001 /* Δt */
#define NX 100 /* 要素の個数 */
#define NY 50 /* 要素の個数 */

#define LENGTH_X 1 /* 長さ */
#define LENGTH_Y 0.5 /* 長さ */
#define DX 1 /* 拡散係数 */
#define DY 1 /* 拡散係数 */

/*OpenGL*/
#define nx0 0
#define ny0 0
#define mod 1

void initial_u(double u[NX + 1][NY + 1],
double dx, double dy, int nx, int ny)
{
int i, j;
double pi = acos(-1.);
double x, y;

for(i = 0; i <= nx; i++) {
x = dx * i;
for(j = 0; j <= ny; j++) {
y = dy * j;
u[i][j] = 4 * cos(3 * pi * x) * cos(2 * pi * y);
```

```

    }
  }
}

/* LU分解のプログラム */
void lu_divide(double a[], double b[], double c[],
double l[][2], double u[], int n)
{
  int i;

  l[0][0] = a[0];
  for(i = 1; i <= n; i++) {
    l[i][1] = b[i];
  }
  for(i = 0; i <= n - 1; i++) {
    u[i] = c[i] / l[i][0];
    l[i + 1][0] = a[i + 1] - u[i] * l[i + 1][1];
  }
}

/* LU x = b の解法*/
void lu_solve(double y[], double b[], double l[][2], double u[],
double x[], int n)
{
  int i;

  y[0] = b[0] / l[0][0];
  for(i = 1; i <= n; i++) {
    y[i] = (b[i] - l[i][1] * y[i - 1]) / l[i][0];
  }
  x[n] = y[n];
  for(i = n - 1; i >= 0; i--) {
    x[i] = y[i] - u[i] * x[i + 1];
  }
}

int main()
{
  int i, j, k;
  int time = TIME;
  int nx = NX;
  int ny = NY;
  int cut = CUT;
  double Dx = DX;
  double Dy = DY;
  double length_x = LENGTH_X;
  double length_y = LENGTH_Y;
  double dt = DT;
  double dx = length_x / (double)nx;
  double dy = length_y / (double)ny;
  double rx = dt / (dx * dx);
  double ry = dt / (dy * dy);
  double old_u[NX + 1][NY + 1], u[NX + 1][NY + 1];
  double um[NX + 1][NY + 1];
  double umx[NY + 1][NX + 1];
  double ax[NX + 1], bx[NX + 1], cx[NX + 1];

```

```

double ay[NY + 1], by[NY + 1], cy[NY + 1];
double Lx[NX + 1][2], Ux[NX], Bx[NX + 1];
double Ly[NY + 1][2], Uy[NY], By[NY + 1];
double Yx[NX + 1], Yy[NY + 1];
double drx = Dx * rx;
double dry = Dy * ry;

/*OpenGL*/
int Mgx, Mgy;
double dgx, dgy;
double rr[(NX + 1) * (NY + 1)];
double gg[(NX + 1) * (NY + 1)];
double bb[(NX + 1) * (NY + 1)];
double umax = 1.;
double umin = -1.;

/*OpenGL*/
Mgx = NX / mod;
Mgy = NY / mod;
dgx = dx * (double)mod;
dgy = dy * (double)mod;

for(i = 0; i <= nx; i++) {
    ax[i] = 2 * (1 + drx);
    bx[i] = - drx;
    cx[i] = - drx;
}
cx[0] = - 2 * drx;
bx[nx] = - 2 * drx;

for(i = 0; i <= ny; i++) {
    ay[i] = 2 * (1 + dry);
    by[i] = - dry;
    cy[i] = - dry;
}
cy[0] = - 2 * dry;
by[ny] = - 2 * dry;

initial_u(old_u, dx, dy, nx, ny);
initial_u(u, dx, dy, nx, ny);

/*OpenGL*/
/* 最大値と最小値を求める */
umax = u[0][0];
umin = u[0][0];
for(i=0; i<=NX; i+=mod){
    for(j=0; j<=NY; j+=mod){
        if(umax < u[i][j])    umax = u[i][j];
        if(umin > u[i][j])    umin = u[i][j];
    }
}
mk_color_full(u, rr, gg, bb, mod, umax, umin, nx0, nx, ny0, ny);
draw_(0, Mgx, Mgy, dgx, dgy, rr, gg, bb, "u", "image");

lu_divide(ax, bx, cx, Lx, Ux, nx);
lu_divide(ay, by, cy, Ly, Uy, ny);

```

```

for(i = 1; i < time; i++) {
  for(k = 0; k <= nx; k++) {
    Bx[k] = 2 * (1 - dry) * u[k][0] + 2 * dry * u[k][1];
  }
  lu_solve(Yx, Bx, Lx, Ux, umx[0], nx);

  for(k = 0; k <= nx; k++) {
    Bx[k] = 2 * (1 - dry) * u[k][ny] + 2 * dry * u[k][ny - 1];
  }
  lu_solve(Yx, Bx, Lx, Ux, umx[ny], nx);

  for(j = 1; j <= ny - 1; j++) {
    for(k = 0; k <= nx; k++) {
      Bx[k] = dry * u[k][j - 1] + 2 * (1 - dry) * u[k][j] + dry * u[k][j + 1];
    }
    lu_solve(Yx, Bx, Lx, Ux, umx[j], nx);
  }
  for(j = 0; j <= ny; j++) {
    for(k = 0; k <= nx; k++) {
      um[k][j] = umx[j][k];
    }
  }

  for(k = 0; k <= ny; k++) {
    By[k] = 2 * (1 - drx) * um[0][k] + 2 * drx * um[1][k];
  }
  lu_solve(Yy, By, Ly, Uy, u[0], ny);

  for(k = 0; k <= ny; k++) {
    By[k] = 2 * (1 - drx) * um[nx][k] + 2 * drx * um[nx - 1][k];
  }
  lu_solve(Yy, By, Ly, Uy, u[nx], ny);

  for(j = 1; j <= nx - 1; j++) {
    for(k = 0; k <= ny; k++) {
      By[k] = drx * um[j - 1][k] + 2 * (1 - drx) * um[j][k] + drx * um[j + 1][k];
    }
    lu_solve(Yy, By, Ly, Uy, u[j], ny);
  }

  if(i % cut == 0) {
    /*OpenGL*/
    mk_color_full(u, rr, gg, bb, mod, umax, umin, nx0, nx, ny0, ny);
    draw_(i / cut, Mgx, Mgy, dgx, dgy, rr, gg, bb, "u", "image");
  }
}
return 0;
}

```

5.9.2.1 オフスクリーンレンダリングのプログラム (Draw_OSMesa2.c)

このサンプルは空間2次元の数値データを可視化するためのもの関数である。

```

/* Draw_OSMesa2.c */
/* Reference osdemo.c */
/* Demo of off-screen Mesa rendering */
/*
 * See Mesa/include/GL/osmesa.h for documentation of the OSMesa functions.
 *
 * If you want to render BIG images you'll probably have to increase
 * MAX_WIDTH and MAX_HEIGHT in src/config.h.
 *
 * This program is in the public domain.
 *
 * Brian Paul
 *
 * PPM output provided by Joerg Schmalzl.
 */

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include "GL/osmesa.h"
#include "GL/glu.h"

#define PIXEL 400
#define MS 260

int Width = PIXEL;
int Height = PIXEL;

/* argument */
int Mgx,Mgy;
int fn;
double dgx,dgy;
double *u,*v,*w;
double x_range,y_range,xy_range;

static void render_image(void)
{
    int i,j;
    double *p,*q,*r,*pp,*qq,*rr;

    p = u; q = v; r = w;
    pp = u + (Mgx + 1); qq = v + (Mgx + 1); rr = w + (Mgx + 1);
    for (j=0;j<Mgy;j++){
        for (i=0;i<Mgx;i++){
            glBegin(GL_POLYGON);
            glColor3f(*p,*q,*r);
            glVertex2f(dgx*i,dgy*j);
            glColor3f(*(p+1), *(q+1), *(r+1));
            glVertex2f(dgx*(i+1),dgy*j);
            glColor3f(*(pp+1), *(qq+1), *(rr+1));
            glVertex2f(dgx*(i+1),dgy*(j+1));
            glColor3f(*pp,*qq,*rr);
            glVertex2f(dgx*i,dgy*(j+1));
            glEnd();
            p++;q++;r++;pp++;qq++;rr++;
        }
    }
}

```



```

    p++;q++;r++;pp++;qq++;rr++;
}
glBegin(GL_LINE_LOOP);
glColor3f(0.0,0.0,0.0);
glVertex2f(0.0,0.0);
glVertex2f(x_range,0.0);
glVertex2f(x_range,y_range);
glVertex2f(0.0,y_range);
glEnd();
}
void set_view(){
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(-0.5*x_range,-0.5*y_range,0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-0.5*x_range-0.1*xy_range,0.5*x_range+0.1*xy_range,
        -0.5*y_range-0.1*xy_range,0.5*y_range+0.1*xy_range);
    glRectf(-0.1*x_range,-0.1*y_range,1.1*x_range,1.1*y_range);
}
void save_file(FILE *f, void *buffer){
    /* write PPM file */
    if (f) {
        int i, x, y;
        GLubyte *ptr;
        ptr = buffer;
        fprintf(f,"P6\n");
        fprintf(f,"# ppm-file created\n");
        fprintf(f,"%i %i\n", Width,Height);
        fprintf(f,"255\n");
        for (y=Height-1; y>=0; y--) {
            for (x=0; x<Width; x++) {
                i = (y*Width + x) * 4;
                fputc(ptr[i], f); /* write red */
                fputc(ptr[i+1], f); /* write green */
                fputc(ptr[i+2], f); /* write blue */
            }
        }
        fclose(f);
    }
    printf("all done\n");
}
void change_image_format(char *in, char *out)
{
    char string[300];
    sprintf(string, "convert %s %s", in, out);
    system(string);
}

/* For Fortran */
/* void draw_( int *p_fn, int *p_Mgx, int *p_Mgy, double *p_dgx, double *p_dgy,
    double *p, double *q, double *r, char *image_file, char *image_dir)
*/
/* For C */
void draw_( int p_fn, int p_Mgx, int p_Mgy, double p_dgx, double p_dgy,

```

```

        double *p, double *q, double *r, char *image_file, char *image_dir)
{
    char ppmfile[BUFSIZ], jpegfile[BUFSIZ];
    FILE *fp;
    OSMesaContext ctx;
    void *buffer;

    fn = p_fn; Mgx = p_Mgx; Mgy = p_Mgy; dgx = p_dgx; dgy = p_dgy;
    u = p; v = q; w = r;
    printf("%d %d %d %lf %lf\n", fn, Mgx, Mgy, dgx, dgy);

    x_range = dgx*Mgx; y_range=dgy*Mgy;
    xy_range = ((x_range > y_range) ? x_range : y_range);
    Width = (int)((x_range / xy_range + 0.2) / 1.2 * PIXEL);
    Height = (int)((y_range / xy_range + 0.2) / 1.2 * PIXEL);

    /* Create an RGBA-mode context */
    ctx = OSMesaCreateContext( GL_RGBA, NULL );

    /* Allocate the image buffer */
    buffer = malloc( Width * Height * 4 );

    /* Bind the buffer to the context and make it current */
    OSMesaMakeCurrent( ctx, buffer, GL_UNSIGNED_BYTE, Width, Height );

    set_view();
    render_image();
    glFlush();

    sprintf(ppmfile, "%s/%s%04d.ppm", image_dir, image_file, fn);
    sprintf(jpegfile, "%s/%s%04d.jpg", image_dir, image_file, fn);

    if ((fp = fopen(ppmfile, "w")) == NULL){
        perror(ppmfile);
        exit(1);
    }
    save_file(fp, buffer);
    /*fclose(fp); */
    change_image_format(ppmfile, jpegfile);
    unlink(ppmfile);
    /* free the image buffer */
    free(buffer);
    /* destroy the context */
    OSMesaDestroyContext(ctx);
    /* return 0; */
}

```

5.9.2.2 mk_color_full.c

mk_color_full は数値データから RGB 値を決める関数である。

```

#include<stdio.h>
#define Nx      100
#define Ny      50
void mk_color_full(double x[][Ny+1], double R[], double G[], double B[], \
    int mod, double xmax, double xmin, \
    int nx0, int nx, int ny0, int ny)
{
    int i, j, n;
    double color;
    n = 0;
    for(j=ny0; j<=ny; j+=mod){
        for(i=nx0; i<=nx; i+=mod){
            color = (x[i][j]-xmin)/(xmax - xmin);
            if(color < 0.0){
                R[n] = 0.0;
                G[n] = 0.0;
                B[n] = 1.0;
            }else if(color >= 1.0){
                R[n] = 1.0;
                G[n] = 0.0;
                B[n] = 0.0;
            }else if( (color >= 0.0) && (color < 0.250)){
                R[n] = 0.0;
                G[n] = 4.0*color;
                B[n] = 1.0;
            }else if( (color >= 0.250) && (color < 0.50)){
                R[n] = 0.0;
                G[n] = 1.0;
                B[n] = 2.0 - 4.0*color;
            }else if( (color >= 0.50) && (color < 0.750)){
                R[n] = 4.0*color - 2.0;
                G[n] = 1.0;
                B[n] = 0.0;
            }else if( (color >= 0.750) && (color < 1.0)){
                R[n] = 1.0;
                G[n] = 4.0 - 4.0*color;
                B[n] = 0.0;
            }
            n += 1;
        }
    }
    printf("n = %d\n", n-1);
}

```

5.9.2.3 Makefile (C 言語用)

上記の Draw_OSMesa2.c と mk_color_full.c を C 言語の数値計算プログラム heat1d.c に挿入して使用するための Makefile のサンプル (Makefile_C.OSMesa) :

```

# <- #で始まる行はコメント

# CC <- C コンパイラを指定

```

```

# default では CC=cc
CC      = gcc

# CFLAGS <- C コンパイラに渡されるオプションを指定
CFLAGS = -O3 -Wall -Wextra

# LDLIBS <- リンクするライブラリを指定
LDLIBS = -lOSMesa -lGLU -lGL -lm

# ALL <- 非標準, ここでは最終生成物を指定
ALL     = heat1d

# all <- よく使われるデフォルトターゲット
# ${ALL} <- 変数を展開
all     : ${ALL}

# clean <- 生成物を消去するためのターゲット
clean  :
rm -f *.o ${ALL}

# heat1d を生成するための依存関係を記述
# heat1d.c は何も書かなくても探しに行く
heat1d : Draw_OSMesa2.o mk_color_full.o

```

サンプルプログラムのコンパイル方法と実行方法

```

$ make -f Makefile_C_OSMesa
実行する前に
$ mkdir image
として, 画像ファイルが保存されるディレクトリを作っておく.
$ ./a.out # 実行

```

Makefile の書き方は以下を参照のこと：情報は古いですが文法は変わっていないので充分役に立ちます。

https://www.ecoop.net/coop/translated/GNUMake3.77/make_toc.jp.html

5.9.2.4 数値計算をおこないながら結果をモニターする (OpenGL 版)

以下のサンプル (OpenGL_Disp01.c) は数値計算を行いながら可視化し、アニメーションに必要な画像 (JPEG ファイル) をつくるためのものである。

```

#include <stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<GL/glut.h>
#include<GL/glu.h>
#include<GL/gl.h>

/* 画像のピクセル数 */
#define WIDTH 500
#define HEIGHT 500

```

```
/* 数値計算に必要なパラメータ */
/* 当然数値計算の内容によって書き換える必要がある */
#define MAXTIME 10
#define DT 0.0005
#define E 0.8
#define G 9.8

/* 必要な宣言 */
GLuint startList;
void display(double y);

/* 画像を PPM ファイルとして保存する */
void save_file(FILE *f){
    if (f) {
        int i, x, y;
        GLubyte *ptr;
        ptr = malloc( WIDTH * HEIGHT * 4 );

        glReadPixels(0, 0, WIDTH, HEIGHT, GL_RGBA,
                     GL_UNSIGNED_BYTE, ptr);
        fprintf(f,"P6\n");
        fprintf(f,"# ppm-file created\n");
        fprintf(f,"%i %i\n", WIDTH,HEIGHT);
        fprintf(f,"255\n");
        for (y = HEIGHT-1; y >= 0; y--) {
            for (x = 0; x < WIDTH; x++) {
                i = (y*WIDTH + x) * 4;
                fputc(ptr[i], f); /* write red */
                fputc(ptr[i+1], f); /* write green */
                fputc(ptr[i+2], f); /* write blue */
            }
        }
        free(ptr);
    }
}
/* PPM ファイル → JPEG ファイル : ImageMagick を使っている */
void change_image_format(char *in, char *out)
{
    char string[300];
    sprintf(string, "convert %s %s", in, out);
    system(string);
}

/* 数値計算する部分 */
double f1(double t, double x, double v)
{
    return - G;
}

double f2(double t, double x, double v)
{
    return v;
}
void rakka()
{
    int i;
```

```

double x, v;
double new_x, new_v;
double t;
double dt = DT;
double e = E;
double maxtime = MAXTIME;
int n = (int)(maxtime / dt);

char ppmfile[BUFSIZ], jpegfile[BUFSIZ];
FILE *fp;

x = 0.;
v = 4.9;

for(i = 1; i <= n; i++) {
    t = dt * i;
    new_x = x + dt * f2(t, x, v);
    new_v = v + dt * f1(t, x, v);
    x = new_x;
    v = new_v;
    //printf("%lf %lf\n", t, x);
    display(x);
    if(x <= 0) {
        v = - e * v;
        x = 0.;
    }

    if((i-1)%50 == 0){
        sprintf(ppmfile, "./image/Image%04d.ppm", (i-1)/50);
        sprintf(jpegfile, "./image/Image%04d.jpg", (i-1)/50);
        if ((fp = fopen(ppmfile, "w")) == NULL){
            perror(ppmfile);
            exit(1);
        }
        save_file(fp);
        fclose(fp);
        change_image_format(ppmfile, jpegfile);
        unlink(ppmfile);
    }
}
return;
}

/* エラー処理 */
void errorCallback(GLenum errorCode)
{
    const GLubyte *estring;

    estring = gluErrorString(errorCode);
    fprintf(stderr, "Quadric Error: %s\n", estring);
    exit(0);
}

/* 画面の初期化 */
void init(void)
{

```

```

GLUQuadricObj *qobj;
GLfloat mat_ambient[] = { 1.0, 0.0, 0.0, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 50.0 };
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
GLfloat model_ambient[] = { 0.5, 0.5, 0.5, 1.0 };

glClearColor(1.0, 1.0, 1.0, 1.0);

glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, model_ambient);

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);

startList = glGenLists(1);
qobj = gluNewQuadric();
/* gluQuadricCallback(qobj, GLU_ERROR, errorCallback); */
gluQuadricDrawStyle(qobj, GLU_FILL);
gluQuadricNormals(qobj, GLU_SMOOTH);
glNewList(startList, GL_COMPILE);
gluSphere(qobj, 0.1, 10, 10);
glEndList();
}

/* 可視化部分：一番大事 */
void display(double y)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glEnable(GL_LIGHTING);
    glShadeModel(GL_SMOOTH);
    glTranslatef(0.0, y, 0.0);
    glCallList(startList);
    glPopMatrix();
    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if( w <= h ){
        glOrtho(-2.5, 2.5, -2.5*(GLfloat)h/(GLfloat)w, 2.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
    }else{
        glOrtho(-2.5*(GLfloat)h/(GLfloat)w, 2.5*(GLfloat)h/(GLfloat)w, -2.5, 2.5, -10.0, 10.0);
    }

    glMatrixMode(GL_MODELVIEW);
}

```

```
    glLoadIdentity();
}

/* キーボードからの入力 */
void keyboard(unsigned char key, int x, int y)
{
    switch( key ){
    case 27:
        exit(0);
        break;
    }
}

/* メインループ */
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE| GLUT_RGB| GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(200,200);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(rakka);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

サンプルプログラムのコンパイル方法と実行方法

```
$ gcc -O3 -Wall OpenGL_Disp01.c -o a.out -I/usr/X11R6/include -L/usr/X11R6/lib
-lglut -lGLU -lGL -lXmu -lXi -lXext -lX11 -lm
実際は一行で書くこと！
実行する前に
$ mkdir image
として、画像ファイルが保存されるディレクトリを作っておく。
$ ./a.out # 実行
```

5.9.3 おまけ (VNC を使った画面共有の方法)

いつも失念するので VNC を使って Linux の画面を Mac 上に共有して使う方法を解説しておきます。
Linux 上での設定
VNC サーバをインストールします。


```
# yum search vnc
とするとリストが出てきますが、その中で
tigervnc-server.x86_64
を選択して
# yum install tigervnc-server.x86_64
としてインストール出来ます。
```

vnc サーバーの設定 (参照: <http://d.hatena.ne.jp/rdera/20090303/1236071478>)

```
# vi /etc/sysconfig/vncservers
```

```
# VNCSERVERS="2:myusername"
# VNCSERVERARGS[2]="-geometry 800x600 -nolisten tcp -nohttpd -localhost"
これを以下のように編集する。
VNCSERVERS="1:nagayama"
VNCSERVERARGS[1]="-geometry 1280x1024 -nolisten tcp -nohttpd"
VNC パスワードを設定する

さきほど登録したユーザ nagayama のパスワードを設定する。
# su nagayama
[nagayama]> vncpasswd
Password:
Verify:
~nagayama/.vnc/xstartup を編集する
ユーザ nagayama で VNC 関連のファイルを編集する。
[nagayama]> vi ~/.vnc/xstartup
以下の部分
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
twm &
を以下のように変更する。
# xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
gnome-session &
```

VNC サーバを起動:

```
[root]# /etc/rc.d/init.d/vncserver start
```

起動の自動化:

以下のコマンドを実行することにより、OS を再起動時に VNC サーバが自動的に起動する。

```
[root]# chkconfig vncserver on
```

最後に、メニューバーからシステム→設定→リモート・デスクトップでリモート・デスクトップの設定から共有にチェックを入れ、セキュリティでは「パスワード設定」と「自動的にネットワークを設定する」にチェックを入れて、パスワードを設定してください (図 5.1).

Mac OS X 上での設定は、open vnc://server の IP アドレス:590x (userID) で終わりです。具体的には次

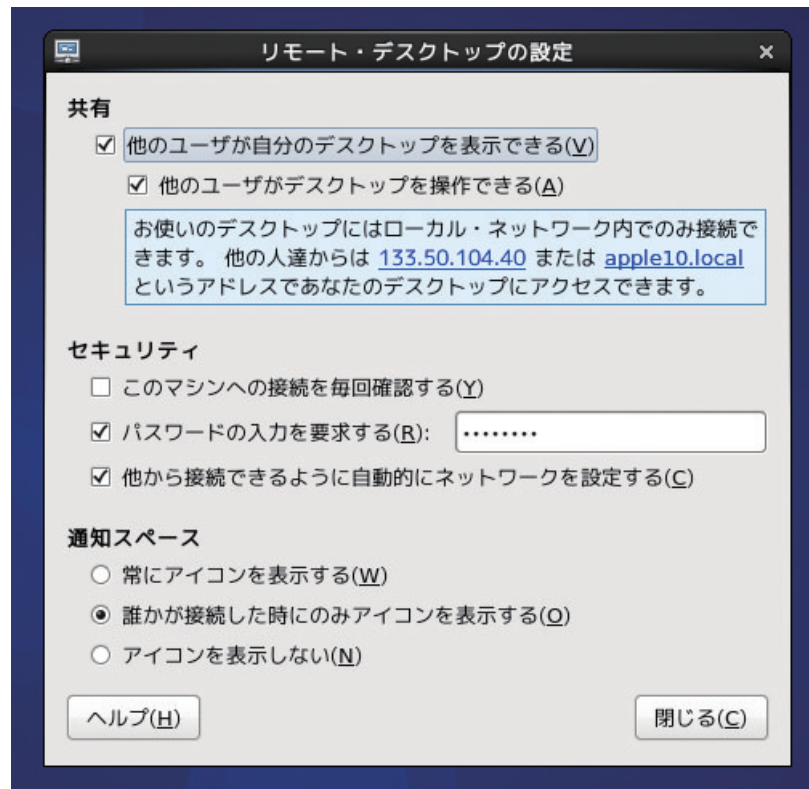


図 5.1: リモート・デスクトップの設定の画面

のようになります。

```
$ open vnc://133.50.104.40:5901
```

5.9.4 メモ書き

Mac のスクリーンショット

1. 「コマンド」キー + 「Shift」キー + 「3」キー
画面全体
2. 「コマンド」キー + 「Shift」キー + 「4」キー
ドラッグで選択した部分
3. 「コマンド」キー + 「Shift」キー + 「4」キー + 「Space」キー
カメラアイコンが表示される。カメラアイコンで選択した一つのウインドウ

付録 A 数値スキームの安定性

A.1 拡散方程式の数値スキームの安定性解析 その1

拡散方程式の数値スキームの安定性に関しては [11] を参照するとよい.

A.1.1 陽解法 (Explicit Scheme) の安定性解析

$$u_j^{n+1} = \mu u_{j+1}^n + (1 - 2\mu)u_j^n + \mu u_{j-1}^n \quad (\text{A.1})$$

ただし, $\mu = D\Delta x^2/\Delta t$, $x_j = j\Delta x$ である. 波数 k の正弦波成分の成長を見るために

$$u_j^n = \lambda^n \exp(ikx_j)$$

とおく. もしある波数 k に対して $|\lambda| > 1$ であるならば, ノイズの中の波数 k の成分が指数的に成長してしまう → 数値的不安定性

よって全ての波数 k に対し $|\lambda| \leq 1$ でなければならない.

$$\begin{aligned} \lambda^{n+1} \exp(ikx_j) &= \mu \lambda^n \exp(ikx_{j+1}) + (1 - 2\mu)\lambda^n \exp(ikx_j) + \mu \lambda^n \exp(ikx_{j-1}) \\ \lambda &= \mu \exp(ik\Delta x) + (1 - 2\mu) + \mu \exp(-ik\Delta x) \\ &= 1 - 2\mu(1 - \cos k\Delta x) = 1 - 4\mu \sin^2 \frac{k\Delta x}{2} \end{aligned}$$

よって $\lambda \leq 1$ は常に成立するので $\lambda \geq -1$ をチェック

$$1 - 4\mu \sin^2 \frac{k\Delta x}{2} \geq -1 \quad \text{for all } k,$$

$$1 - 4\mu \geq -1.$$

故に

$$D \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}.$$

A.1.2 完全陰解法 (Implicit Scheme) の安定性解析

$$-\mu u_{j+1}^{n+1} + (1 + 2\mu)u_j^{n+1} - \mu u_{j-1}^{n+1} = u_j^n. \quad (\text{A.2})$$

ただし, $\mu = D\Delta x^2/\Delta t$, $x_j = j\Delta x$ である.

$$\begin{aligned} -\mu \lambda^{n+1} \exp(ikx_{j+1}) + (1 + 2\mu)\lambda^{n+1} \exp(ikx_j) - \mu \lambda^{n+1} \exp(ikx_{j-1}) &= \lambda^n \exp(ikx_j), \\ \lambda(-\mu \exp(ik\Delta x) + (1 + 2\mu) - \mu \exp(-ik\Delta x)) &= 1, \\ \lambda\{1 + 2\mu(1 - \cos k\Delta x)\} &= 1, \\ \lambda &= \frac{1}{1 + 4\mu \sin^2 \frac{k\Delta x}{2}}. \end{aligned}$$

故に, 任意の k に対して $|\lambda| \leq 1$ となるので陰解法は無条件安定である.

A.2 拡散方程式の数値スキームの安定性解析 その2

A.2.1 差分方程式の行列表示

$$\frac{\partial^2 u}{\partial x^2} \cong \frac{u(t, (j+1)\Delta x) - 2u(t, j\Delta x) + u(t, (j-1)\Delta x)}{\Delta x^2} + O(\Delta x^2) \quad (\text{A.3})$$

に対し, $t = (k+1)\Delta t$, Δt を混合して考える.

$$\begin{aligned} (\text{A.3}) \text{ の右辺} &\cong \theta \frac{u((k+1)\Delta t, (j+1)\Delta x) - 2u((k+1)\Delta t, j\Delta x) + u((k+1)\Delta t, (j-1)\Delta x)}{\Delta x^2} \\ &\quad + (1-\theta) \frac{u(k\Delta t, (j+1)\Delta x) - 2u(k\Delta t, j\Delta x) + u(k\Delta t, (j-1)\Delta x)}{\Delta x^2} \end{aligned} \quad (\text{A.4})$$

と近似する. このとき

1. $\theta = 0$ のとき **完全陰解法.**
2. $\theta = 1/2$ のとき **クランク・ニコルソン法.**
3. $\theta = 1$ のとき **陽解法.**

となる.

(A.4) を用いると熱方程式の差分方程式は

$$\frac{u_j^{k+1} - u_j^k}{\Delta t} = \theta \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{\Delta x^2} + (1-\theta) \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{\Delta x^2}$$

となる. ここで, $r = \Delta t / \Delta x^2$ とおくと

$$-r\theta u_{j-1}^{k+1} + (1+2\theta r)u_j^{k+1} - \theta r u_{j+1}^{k+1} = (1-\theta)r u_{j-1}^k + (1-2(1-\theta)r)u_j^k + (1-\theta)r u_{j+1}^k. \quad (\text{A.5})$$

と書ける. 境界条件 $u_0^k = a$ $u_N^k = b$ より

1. $j = 1$ のとき

$$(1+2\theta r)u_1^{k+1} - \theta r u_2^{k+1} = ar + (1-2(1-\theta)r)u_1^k + (1-\theta)r u_2^k. \quad (\text{A.6})$$

2. $j = N-1$ のとき

$$-r\theta u_{N-2}^{k+1} + (1+2\theta r)u_{N-1}^{k+1} = (1-\theta)r u_{N-2}^k + (1-2(1-\theta)r)u_{N-1}^k + rb. \quad (\text{A.7})$$

(A.5)-(A.7) より

$$\begin{aligned} &\begin{pmatrix} 1+2\theta r & -\theta r & 0 & 0 & \cdots & 0 \\ -\theta r & 1+2\theta r & -\theta r & 0 & \cdots & 0 \\ 0 & -\theta r & 1+2\theta r & -\theta r & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -\theta r & 1+2\theta r & -\theta r \\ 0 & \cdots & 0 & 0 & -\theta r & 1+2\theta r \end{pmatrix} \begin{pmatrix} u_1^{k+1} \\ u_2^{k+1} \\ \vdots \\ \vdots \\ u_{N-2}^{k+1} \\ u_{N-1}^{k+1} \end{pmatrix} \\ &= \begin{pmatrix} ra + (1-2(1-\theta)r)u_1^k + (1-\theta)r u_2^k \\ \vdots \\ (1-\theta)r u_{j-1}^k + (1-2(1-\theta)r)u_j^k + (1-\theta)r u_{j+1}^k \\ \vdots \\ (1-\theta)r u_{N-2}^k + (1-2(1-\theta)r)u_{N-1}^k + rb \end{pmatrix} \end{aligned} \quad (\text{A.8})$$

となり, これは

$$A = \begin{pmatrix} 1+2\theta r & -\theta r & 0 & 0 & \cdots & 0 \\ -\theta r & 1+2\theta r & -\theta r & 0 & \cdots & 0 \\ 0 & -\theta r & 1+2\theta r & -\theta r & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -\theta r & 1+2\theta r & -\theta r \\ 0 & \cdots & 0 & 0 & -\theta r & 1+2\theta r \end{pmatrix}, \mathbf{u}^{k+1} = \begin{pmatrix} u_1^{k+1} \\ u_2^{k+1} \\ \vdots \\ \vdots \\ u_{N-2}^{k+1} \\ u_{N-1}^{k+1} \end{pmatrix},$$

$$\mathbf{b}^k = \begin{pmatrix} ra + (1-2(1-\theta)r)u_1^k + (1-\theta)ru_2^k \\ \vdots \\ (1-\theta)ru_{j-1}^k + (1-2(1-\theta)r)u_j^k + (1-\theta)ru_{j+1}^k \\ \vdots \\ (1-\theta)ru_{N-2}^k + (1-2(1-\theta)r)u_{N-1}^k + rb \end{pmatrix}$$

とすると, 差分方程式の解は連立一次方程式

$$A\mathbf{u}^{k+1} = \mathbf{b}^k$$

を解くことによって求められる. ところで

$$(A.8) \text{ の右辺} = \begin{pmatrix} 1-2(1-\theta)r & (1-\theta)r & & & & 0 \\ (1-\theta)r & 1-2(1-\theta)r & (1-\theta)r & & & \\ & & \ddots & \ddots & \ddots & \\ & & & (1-\theta)r & 1-2(1-\theta)r & (1-\theta)r \\ 0 & & & & (1-\theta)r & 1-2(1-\theta)r \end{pmatrix} \begin{pmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_{N-2}^k \\ u_{N-1}^k \end{pmatrix} + \begin{pmatrix} ra \\ 0 \\ \vdots \\ 0 \\ rb \end{pmatrix}.$$

$$\stackrel{\text{def}}{=} B\mathbf{u}^k + \mathbf{b}.$$

と書けるので, 連立一次方程式は次のように書き下すことができる.

$$A\mathbf{u}^{k+1} = B\mathbf{u}^k + \mathbf{b}.$$

ここで

$$M = \begin{pmatrix} -2 & 1 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & 1 & \\ & & & 1 & -2 \end{pmatrix}$$

とすると

$$A = I - \theta r M,$$

$$B = I + (1-\theta)r M$$

と書ける. ただし, I は単位行列である.

A.2.2 差分方程式の安定性

A.2.2.1 陽解法の安定性

陽解法 ($\theta = 0$ のとき) の安定性は

$$\begin{cases} A = I, \\ B = I + rM \end{cases}$$

となるので

$$\mathbf{u}^{k+1} = B\mathbf{u}^k + \mathbf{b}$$

と書ける。従って

$$\begin{aligned} \mathbf{u}^{k+1} &= B(B\mathbf{u}^{k-1} + \mathbf{b}) + \mathbf{b} \\ &\vdots \\ &= B^{k+1}\mathbf{u}^0 + \sum_{l=0}^k B^l \mathbf{b}. \end{aligned}$$

今、 $\rho(B) = \max_{1 \leq i \leq N-1} |\lambda_i|$ (λ_i は B の固有値とする) とおくと

$$\rho(B) < 1 \iff \lim_{k \rightarrow \infty} B^k = 0,$$

$$\rho(B) > 1 \iff \lim_{k \rightarrow \infty} B^k = \infty.$$

差分方程式が安定であるためには (発散しないためには), $\lim_{k \rightarrow \infty} B^k = 0$ でなければならない。つまり $\rho(B) < 1$ を満たさなければならない。従って、 B の固有値を調べればよい。今、 M の固有値を λ , 固有空間を \mathbf{x} とすると

$$B\mathbf{x} = (I + rM)\mathbf{x} = (1 + r\lambda)\mathbf{x}$$

より、 B の固有値は $(1 + r\lambda)$ となる。従って M の固有値を調べる。

$$M\mathbf{x} = \lambda\mathbf{x}$$

とする。

$$\begin{pmatrix} \lambda+2 & -1 & & & 0 \\ -1 & \lambda+2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & \lambda+2 & -1 \\ 0 & & & -1 & \lambda+2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

より

$$\begin{aligned} (\lambda+2)x_1 - x_2 &= 0, \\ -x_{j-1} + (\lambda+2)x_j - x_{j+1} &= 0, \quad (j = 2, 3, \dots, N-2), \\ -x_{N-2} + (\lambda+2)x_{N-1} &= 0. \end{aligned}$$

を得る。ここで、

第二種チェビシェフ多項式について

$$-U_{j-1}(x) + 2xU_j(x) - U_{j+1}(x) = 0 \quad (\#)$$

の解は

$$U_j(x) = \sin(j\theta), \quad x = \cos \theta$$

と書ける。

故に、 θ を任意として $x_j(\lambda) = \sin(j\theta)$ 。ただし、 $\lambda/2 + 1 = \cos \theta$ となる。 $-x_{N-2} + (\lambda+2)x_{N-1} = 0$ より

$$-\sin(N-2)\theta + 2\cos \theta \sin(N-1)\theta = 0. \quad (\text{A.9})$$

$\sin j\theta$ は $j = N - 1$ において漸化式 (#) をみたすので

$$-\sin(N-2)\theta + 2\cos\theta\sin(N-1)\theta - \sin N\theta = 0. \quad (\text{A.10})$$

(A.9) かつ (A.10) をみたすような θ を決める.

$$\begin{aligned} \sin N\theta &= 0, \\ N\theta &= m\pi, \quad (1 \leq m \leq N-1), \\ \therefore \theta_m &= \frac{m\pi}{N}. \end{aligned}$$

故に

$$\begin{aligned} \lambda_m &= -2(1 - \cos\theta), \\ &= -2\left(1 - \cos\frac{m\pi}{N}\right), \quad (1 \leq m \leq N-1), \\ &= -4\sin^2\frac{m\pi}{2N}. \end{aligned} \quad (\text{A.11})$$

固有空間 \mathbf{x}^m は

$$\mathbf{x}^m = \begin{pmatrix} \vdots \\ \sin\frac{jm\pi}{N} \\ \vdots \end{pmatrix}_{1 \leq j \leq N-1}.$$

(A.11) より, B の固有値は $1 - 4r\sin^2\frac{m\pi}{2N}$, $1 \leq m \leq N-1$. 従って, $-1 < 1 - 4r\sin^2\frac{m\pi}{2N} < 1$ のとき, $\lim_{k \rightarrow \infty} B^k = 0$ となり, 差分方程式は安定となる. 従って

$$\begin{aligned} -1 &< 1 - 4r, \\ r &< \frac{1}{2}. \end{aligned}$$

A.2.2.2 陰解法の安定性

完全陰解法 ($\theta = 1$ のとき) の安定性

$$\begin{cases} A = I - rM, \\ B = I \end{cases}$$

となるので, 差分方程式は $A\mathbf{u}^{k+1} = \mathbf{u}^k$ となる.

ここで M の固有値 λ_m と固有空間 \mathbf{v}_m に対して

$$A\mathbf{v}_m = (I - rM)\mathbf{v}_m,$$

$$1 - r\lambda_m = 1 + 4r\sin^2\frac{m\pi}{2N} \geq 1$$

より, A は逆行列 A^{-1} を持つ. 従って

$$\mathbf{u}^{k+1} = A^{-1}\mathbf{u} = \dots = (A^{-1})^{k+1}\mathbf{u}_0.$$

安定性は $\rho(A^{-1}) < 1$ で決まる.

A^{-1} の固有値は A の固有値 $\lambda_A = 1 - r\lambda_m$ の逆数より, A^{-1} の固有値は

$$\begin{aligned} \lambda_{A^{-1}} &= \frac{1}{1 - r\lambda_m}, \\ &= \frac{1}{1 + 4r\sin^2\frac{m\pi}{2N}}. \end{aligned}$$

ここで

$$\left| \frac{1}{1 + 4r \sin^2 \frac{m\pi}{2N}} \right| \leq 1$$

より、任意の r に対して $\rho(A^{-1}) < 1$ となる。" $r = \Delta t / \Delta x^2$ に制約無し " という意味で、**無条件安定** であるという。

A.2.2.3 クランク–ニコルソン法の安定性

クランク–ニコルソン法 ($\theta = 1/2$ のとき) の安定性.

$$\begin{cases} A = I - \frac{r}{2}M, \\ B = I + \frac{r}{2}M. \end{cases}$$

なので、差分方程式は $A\mathbf{u}^{k+1} = B\mathbf{u}^k$ となる。これまで同様 A の固有値を調べる。 A の固有値は

$$1 + 2r \sin^2 \frac{m\pi}{2N} \geq 1$$

より、 A は逆行列を持つ。従って

$$\mathbf{u}^{k+1} = A^{-1}B\mathbf{u}^k = \dots = (A^{-1}B)^{k+1}\mathbf{u}_0.$$

安定性は $\rho(A^{-1}B) \leq 1$ で決まる。ただし、数値計算上では $\rho(A^{-1}B) < 1$ としなければならない。 M の固有値 λ_m と固有空間 \mathbf{v}_m に対して、

$$\begin{aligned} B\mathbf{v}_m &= \left(I + \frac{r}{2}M\right)\mathbf{v}_m = \left(1 + \frac{r}{2}\lambda_m\right)\mathbf{v}_m, \\ A\mathbf{v}_m &= \left(I - \frac{r}{2}M\right)\mathbf{v}_m = \left(1 - \frac{r}{2}\lambda_m\right)\mathbf{v}_m. \end{aligned}$$

従って

$$A^{-1}\mathbf{v}_m = \frac{1}{\left(1 - \frac{r}{2}\lambda_m\right)}\mathbf{v}_m$$

となるので

$$\begin{aligned} A^{-1}B\mathbf{v}_m &= A^{-1}\left(1 + \frac{r}{2}\lambda_m\right)\mathbf{v}_m, \\ &= \left(1 + \frac{r}{2}\lambda_m\right)A^{-1}\mathbf{v}_m, \\ &= \frac{\left(1 + \frac{r}{2}\lambda_m\right)}{\left(1 - \frac{r}{2}\lambda_m\right)}\mathbf{v}_m. \end{aligned}$$

$A^{-1}B$ の固有値 $\lambda_{A^{-1}B}$ は

$$\begin{aligned} \lambda_{A^{-1}B} &= \frac{\left(1 + \frac{r}{2}\lambda_m\right)}{\left(1 - \frac{r}{2}\lambda_m\right)}, \\ &= \frac{1 - 2r \sin^2 \frac{m\pi}{2N}}{1 + 2r \sin^2 \frac{m\pi}{2N}}. \end{aligned}$$

従って、安定であるためには

$$-1 \leq \frac{1 - 2r \sin^2 \frac{m\pi}{2N}}{1 + 2r \sin^2 \frac{m\pi}{2N}} \leq 1$$

となればよい。 $\theta = 1/2$ のとき、この不等式は自動的に成立するので無条件に安定である。

問題；混合解法

$$u_i^{n+1} - u_i^n = D\theta(ru_{i-1}^{n+1} - 2ru_i^{n+1} + ru_{i+1}^{n+1}) + D(1-\theta)(ru_{i-1}^n - 2ru_i^n + ru_{i+1}^n), \quad 0 \leq i \leq N. \quad (\text{A.12})$$

について安定性条件を求めよ.

A.3 差分スキームの収束性

差分スキームの安定性 (Stability) についてはこれまで議論してきた. ここでは, 差分スキームの適合性 (Consistency), 収束性 (Convergence) について解説する.

Definition 1 微分方程式 (熱方程式の初期値・境界値問題が) が *Well-posed* である.

$\stackrel{\text{def}}{\iff}$

解が一意的に存在し, パラメータを連続に変化させたとき, 解も連続に変化する.

Definition 2 差分スキームが適合である.

$\stackrel{\text{def}}{\iff}$

$\Delta t, \Delta x$ を $\downarrow 0$ としたとき, 差分方程式が元の微分方程式に収束する.

Definition 3 差分スキームが収束する.

$\stackrel{\text{def}}{\iff}$

$\Delta t, \Delta x$ を $\downarrow 0$ としたとき, 差分方程式の解が元の微分方程式の解に収束する.

熱方程式の差分スキームの適切性は, 差分化の手続きを見れば明らかにわかるが, 収束性は全くもって自明ではない. そこで, 次の Lax の同等定理が重要となってくる.

Theorem 1 線形微分方程式の初期値・境界値問題が *Well-posed* であり, 差分スキームが適合かつ安定である.

$\stackrel{\text{必要十分}}{\iff}$

差分スキームは収束する.

注意すべき点は, この定理は線形方程式の場合に成立しているということである. 非線形方程式の場合は成り立つとは限らないので注意が必要である. 非線形問題の場合は, 差分スキームが安定であるからといって, 正しい数値計算結果 (元の微分方程式の解の近似になっている) を得ているという保証がないことを忘れてはならない. 数値計算結果がもっともらしいということはどういうことなのだろうかという疑問を持って数値計算をして欲しいものです.

付録B 発展編の完成部分

発展編で特に重要であると思われる部分で完成した文章を付録として掲載する。

B.1 軸対称問題の数値計算法（有限体積近似）

次の軸対称問題において一様な差分化を行うと $r = 0$ の近くで著しく精度が悪くなる。そこで原点近くで離散化誤差が小さくなるように不均一な差分化を行う手法を解説する。

$$\frac{\partial u}{\partial t} = d \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + f(u), \quad t > 0, 0 < r < R, \quad (\text{B.1})$$

$$\frac{\partial u}{\partial r}(0, r) = u_0(r), \quad 0 \leq x \leq R, \quad (\text{B.2})$$

$$\frac{\partial u}{\partial r}(t, 0) = \frac{\partial u}{\partial r}(t, R) = 0, \quad t > 0. \quad (\text{B.3})$$

$$\int_0^R u_t r dr = \int_0^R d \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr + \int_0^R f(u) r dr, \quad (\text{B.4})$$

$$\begin{aligned} \int_0^R (u_t - f(u)) r dr &= \int_0^{\frac{1}{2}r_1} (u_t - f(u)) r dr + \sum_{j=1}^{N-1} \int_{\frac{1}{2}(r_{j-1}+r_j)}^{\frac{1}{2}(r_j+r_{j+1})} (u_t - f(u)) r dr \\ &\quad + \int_{\frac{1}{2}(r_{N-1}+r_N)}^{r_N} (u_t - f(u)) r dr. \end{aligned} \quad (\text{B.5})$$

$u(t, r)$ に対して $u_j = u(t, r_j)$ とおく。

$$\int_0^{\frac{1}{2}r_1} (u_t - f(u)) r dr \simeq \left(\frac{d}{dt} u_0 - f(u_0) \right) \int_0^{\frac{1}{2}r_1} r dr = \frac{1}{8} r_1^2 \left(\frac{d}{dt} u_0 - f(u_0) \right) \quad (\text{B.6})$$

$$\begin{aligned} \int_{\frac{1}{2}(r_{N-1}+r_N)}^{r_N} (u_t - f(u)) r dr &\simeq \left(\frac{d}{dt} u_N - f(u_N) \right) \int_{\frac{1}{2}(r_{N-1}+r_N)}^{r_N} r dr \\ &= \left(\frac{d}{dt} u_N - f(u_N) \right) \left(\frac{1}{2} r_N^2 - \frac{1}{8} (r_{N-1} + r_N)^2 \right). \end{aligned} \quad (\text{B.7})$$

$$\begin{aligned} \int_{\frac{1}{2}(r_{j-1}+r_j)}^{\frac{1}{2}(r_j+r_{j+1})} (u_t - f(u)) r dr &\simeq \left(\frac{d}{dt} u_j - f(u_j) \right) \int_{\frac{1}{2}(r_{j-1}+r_j)}^{\frac{1}{2}(r_j+r_{j+1})} r dr \\ &= \left(\frac{d}{dt} u_j - f(u_j) \right) \left(\frac{1}{8} (r_{j+1} + r_j)^2 - \frac{1}{8} (r_j + r_{j-1})^2 \right). \end{aligned} \quad (\text{B.8})$$

拡散項の離散化

$$\begin{aligned} \int_0^R d \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr &= \int_0^{\frac{1}{2}r_1} d \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr + \sum_{j=1}^{N-1} \int_{\frac{1}{2}(r_{j-1}+r_j)}^{\frac{1}{2}(r_j+r_{j+1})} d \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr \\ &\quad + \int_{\frac{1}{2}(r_{N-1}+r_N)}^{r_N} d \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr. \end{aligned} \quad (\text{B.9})$$

$$\begin{aligned}
\int_0^{\frac{1}{2}r_1} d\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr &= [dru_r]_0^{\frac{1}{2}r_1} \\
&= d\frac{1}{2}r_1 \frac{u_1 - u_0}{r_1 - r_0} = d\frac{1}{2}r_1 \frac{u_1 - u_0}{r_1} = \frac{d}{2}(u_1 - u_0). \tag{B.10}
\end{aligned}$$

$$\begin{aligned}
\int_{\frac{1}{2}(r_{N-1}+r_N)}^{r_N} d\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr &= [dru_r]_{\frac{1}{2}(r_{N-1}+r_N)}^{r_N} \\
&= -d\frac{1}{2}(r_{N-1} + r_N) \frac{u_N - u_{N-1}}{r_N - r_{N-1}}. \tag{B.11}
\end{aligned}$$

$$\begin{aligned}
\int_{\frac{1}{2}(r_{j-1}+r_j)}^{\frac{1}{2}(r_j+r_{j+1})} d\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) r dr &= [dru_r]_{\frac{1}{2}(r_{j-1}+r_j)}^{\frac{1}{2}(r_j+r_{j+1})} \\
&= d\frac{1}{2}(r_{j+1} + r_j) \frac{u_{j+1} - u_j}{r_{j+1} - r_j} - d\frac{1}{2}(r_j + r_{j-1}) \frac{u_j - u_{j-1}}{r_j - r_{j-1}}. \tag{B.12}
\end{aligned}$$

従って, (B.6), (B.10) より

$$\frac{1}{8}r_1^2 \left(\frac{d}{dt}u_0 - f(u_0) \right) = \frac{d}{2}(u_1 - u_0),$$

$j = 0$ の離散方程式

$$\frac{d}{dt}u_0 = \frac{4}{r_1^2}d(u_1 - u_0) + f(u_0). \tag{B.13}$$

(B.7), (B.11) より

$$\left(\frac{d}{dt}u_N - f(u_N) \right) \left(\frac{1}{2}r_N^2 - \frac{1}{8}(r_{N-1} + r_N)^2 \right) = -d\frac{1}{2}(r_{N-1} + r_N) \frac{u_N - u_{N-1}}{r_N - r_{N-1}},$$

$j = N$ の離散方程式

$$\frac{d}{dt}u_N = -\frac{d}{2} \frac{r_{N-1} + r_N}{\frac{1}{2}r_N^2 - \frac{1}{8}(r_{N-1} + r_N)^2} \frac{u_N - u_{N-1}}{r_N - r_{N-1}} + f(u_N). \tag{B.14}$$

(B.8), (B.12) より

$$\begin{aligned}
&\left(\frac{d}{dt}u_j - f(u_j) \right) \left(\frac{1}{8}(r_{j+1} + r_j)^2 - \frac{1}{8}(r_j + r_{j-1})^2 \right) \\
&= \frac{d}{2} \left((r_{j+1} + r_j) \frac{u_{j+1} - u_j}{r_{j+1} - r_j} - (r_j + r_{j-1}) \frac{u_j - u_{j-1}}{r_j - r_{j-1}} \right),
\end{aligned}$$

$j = 1, 2, \dots, N-2, N-1$ の離散方程式

$$\begin{aligned}
\frac{d}{dt}u_j &= \frac{d}{2} \frac{1}{\frac{1}{8}(r_{j+1} + r_j)^2 - \frac{1}{8}(r_j + r_{j-1})^2} \\
&\quad \left((r_{j+1} + r_j) \frac{u_{j+1} - u_j}{r_{j+1} - r_j} - (r_j + r_{j-1}) \frac{u_j - u_{j-1}}{r_j - r_{j-1}} \right) + f(u_j). \tag{B.15}
\end{aligned}$$

B.2 2次元円板領域での数値計算法

$$\frac{\partial u}{\partial t} = D\Delta u, \quad t > 0, \quad x^2 + y^2 < R^2, \quad (\text{B.16})$$

$$u(0, x, y) = u_0(x, y), \quad x^2 + y^2 < R^2, \quad (\text{B.17})$$

$$\frac{\partial u}{\partial n} = 0, \quad x^2 + y^2 = R^2, \quad t > 0. \quad (\text{B.18})$$

ただし, n は外向き法線ベクトル,

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad (\text{B.19})$$

(B.16) に対して変数変換

$$x = r \cos \theta, \quad y = r \sin \theta \quad (\text{B.20})$$

を行うと (B.19) は次のように書き換えられる:

$$\Delta_r = \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2}. \quad (\text{B.21})$$

このとき, (B.16)–(B.18) は次のように書き換えられる:

$$\frac{\partial u}{\partial t} = D \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} \right) u, \quad t > 0, 0 < r < R, 0 < \theta < 2\pi, \quad (\text{B.22})$$

$$u(0, r, \theta) = u_0(r, \theta), \quad 0 < r < R, 0 < \theta < 2\pi, \quad (\text{B.23})$$

$$\frac{\partial u}{\partial r} = 0, \quad t > 0, r = 0, R, 0 < \theta < 2\pi, \quad (\text{B.24})$$

$$u(t, r, 0) = u(t, r, 2\pi), \quad \frac{\partial}{\partial \theta} u(t, r, 0) = \frac{\partial}{\partial \theta} u(t, r, 2\pi), \quad t > 0, 0 < r < R. \quad (\text{B.25})$$

今, N_r, N_θ を自然数とし

$$\Delta r = \frac{R}{N_r}, \quad \Delta \theta = \frac{2\pi}{N_\theta}$$

と置き

$$r_i = i \cdot \Delta r \quad (0 \leq i \leq N_r), \quad \theta_j = j \cdot \Delta \theta \quad (0 \leq j \leq N_\theta)$$

を定義する. 分点は O (原点) と (r_i, θ_j) ($1 \leq i \leq N_r; 0 \leq j \leq N_\theta$) になり, $u(t, r, \theta)$ を

$$u_{i,j}^m = u(m\Delta t, r_i, \theta_j) \quad (\text{B.26})$$

と離散化する. ただし, $\Delta t > 0$ は時間離散間隔である. ここで,

$$r_{i+1/2} = (r_i + r_{i+1})/2$$

とおき, $0 \leq a \leq 1$ に対して

$$u_{i,j+1,k}^{m+a} = a u_{i,j+1}^m + (1-a) u_{i,j+1}^{m+1}. \quad (\text{B.27})$$

を定義する¹.

境界条件 (B.24) の $r = R$ に対する離散化は $0 \leq j \leq N_\theta - 1, m \geq 0$ に対して

$$u_{N_r+1,j}^m = u_{N_r-1,j}^m \quad (\text{B.28})$$

¹ $a = 1$ なら陽解法, $a = 0$ なら完全陰解法にそれぞれ対応する.

となり、境界条件 (B.25) に対する離散化は $1 \leq i \leq N_r$, $m \geq 0$ に対して

$$u_{i,0}^m = u_{i,N_\theta}^m, \quad u_{i,-1}^m = u_{i,N_\theta-1}^m \quad (\text{B.29})$$

となる。原点での離散化は境界条件 (B.24) を考慮して、次のように書くことができる：

$$\frac{\pi}{4} \Delta r^2 \cdot \frac{1}{\Delta t} (u_0^{m+1} - u_0^m) = \sum_{j=1}^{N_\theta} \frac{1}{2} \Delta r \Delta \theta \frac{1}{\Delta r} (u_{1,j}^{m+a} - u_0^{m+a}). \quad (\text{B.30})$$

また

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} = \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right). \quad (\text{B.31})$$

であるから、前節の有限体積法による軸対称問題の離散化において等間隔で空間を分割すると

$$\begin{aligned} \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) &\sim \frac{1}{r_i} \left(r_{i+1/2} \frac{u_{i+1,j} - u_{i,j}}{\Delta r^2} - r_{i-1/2} \frac{u_{i,j} - u_{i-1,j}}{\Delta r^2} \right) \\ &= \left(\frac{2i+1}{i} \frac{u_{i+1,j} - u_{i,j}}{\Delta r^2} - \frac{2i-1}{i} \frac{u_{i,j} - u_{i-1,j}}{\Delta r} \right) \end{aligned} \quad (\text{B.32})$$

となることから、 $1 \leq i \leq N_r$, $0 \leq j \leq N_\theta - 1$ に対しては次のように離散化できる：

$$\begin{aligned} \frac{1}{\Delta t} (u_{i,j}^{m+1} - u_{i,j}^m) &= D \left(\frac{1}{\Delta r^2} \left(\frac{2i+1}{i} (u_{i+1,j}^{m+a} - u_{i,j}^{m+a}) - \frac{2i-1}{i} (u_{i,j}^{m+a} - u_{i-1,j}^{m+a}) \right) \right. \\ &\quad \left. + \left(\frac{1}{r_i \Delta \theta} \right)^2 (u_{i,j+1}^{m+a} - 2u_{i,j}^{m+a} + u_{i,j-1}^{m+a}) \right). \end{aligned} \quad (\text{B.33})$$

B.3 円柱領域での数値計算法

$$\frac{\partial u}{\partial t} = D \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} + \frac{\partial^2}{\partial z^2} \right) u, \quad t > 0, (r, \theta, z) \in \partial \Omega, \quad (\text{B.34})$$

初期条件：

$$u(0, r, \theta, z) = u_0(r, \theta, z), \quad (r, \theta, z) \in \bar{\Omega}, \quad (\text{B.35})$$

境界条件：

$$\frac{\partial}{\partial r} u(t, r, \theta, z) = 0, \quad t > 0, \quad r = 0, R, \quad 0 \leq \theta < 2\pi, \quad 0 < z < L_z, \quad (\text{B.36})$$

$$\frac{\partial}{\partial z} u(t, r, \theta, z) = 0, \quad t > 0, \quad 0 < r < R, \quad 0 \leq \theta < 2\pi, \quad z = 0, L_z, \quad (\text{B.37})$$

$$u(t, r, 0, z) = u(t, r, 2\pi, z), \quad t > 0, \quad 0 < r < R, \quad 0 < z < L_z, \quad (\text{B.38})$$

$$\frac{\partial}{\partial \theta} u(t, r, 0, z) = \frac{\partial}{\partial \theta} u(t, r, 2\pi, z), \quad t > 0, \quad 0 < r < R, \quad 0 < z < L_z. \quad (\text{B.39})$$

この数値計算は円板領域での数値計算法を z 方向に拡張すればよいだけである。

B.4 移流拡散方程式の数値計算法

$$u_t = du_{xx} - h(x)u_x, \quad x \in I. \quad (\text{B.40})$$

d が非常に小さい場合, $h(x)u_x$ の差分化には注意が必要である. このような場合は風上差分といわれる差分化を行う.

B.4.1 1 次の風上差分

移流項の差分化を $h(u)$ の符号によって場合分けする:

$$h(x)u_x \simeq \begin{cases} h(x_j) \frac{u_j - u_{j-1}}{\Delta x} + O(\Delta x), & h(x_j) \geq 0, \\ h(x_j) \frac{u_{j+1} - u_j}{\Delta x} + O(\Delta x), & h(x_j) < 0. \end{cases} \quad (\text{B.41})$$

(B.41) を書き換えるとおぎのようになる:

$$h(x)u_x \simeq \frac{h(x_j) + |h(x_j)|}{2} \left(\frac{u_j - u_{j-1}}{\Delta x} \right) + \frac{h(x_j) - |h(x_j)|}{2} \left(\frac{u_{j+1} - u_j}{\Delta x} \right). \quad (\text{B.42})$$

従って, 次の 1 次風上差分の公式が得られる.

1 次の風上差分

$$h(x) \frac{\partial u}{\partial x} \simeq h(x_j) \frac{u_{j+1} - u_{j-1}}{2\Delta x} - |h(x_j)| \frac{u_{j+1} - 2u_j + u_{j-1}}{2\Delta x}. \quad (\text{B.43})$$

第 2 項は人工粘性項であり, 1 次の風上差分では, 拡散項の離散化が人工粘性として与えられる.

B.4.2 3 次の風上差分

移流項の打ち切り誤差を $O(\Delta x^3)$ とすると, 風上差分は次のように書ける:

$$h(x)u_x \simeq \begin{cases} h(x_j) \frac{u_{j-2} - 6u_{j-1} + 3u_j + 2u_{j+1}}{6\Delta x} + O(\Delta x^3), & h(x_j) > 0, \\ h(x_j) \frac{-u_{j+2} + 6u_{j+1} - 3u_j - 2u_{j-1}}{6\Delta x} + O(\Delta x^3), & h(x_j) < 0. \end{cases} \quad (\text{B.44})$$

(B.44) は次のように書き換えられる:

$$h(x)u_x \simeq \frac{h(x_j) + |h(x_j)|}{2} \left(\frac{u_{j-2} - 6u_{j-1} + 3u_j + 2u_{j+1}}{6\Delta x} \right) + \frac{h(x_j) - |h(x_j)|}{2} \left(\frac{-u_{j+2} + 6u_{j+1} - 3u_j - 2u_{j-1}}{6\Delta x} \right). \quad (\text{B.45})$$

従って, 3 次風上差分の公式を得る:

3 次の風上差分

$$h(x) \frac{\partial u}{\partial x} \simeq h(x_j) \frac{-u_{j+2} + 8u_{j+1} - 8u_{j-1} + u_{j-2}}{12\Delta x} + |h(x_j)| \frac{u_{j+2} - 4u_{j+1} + 6u_j - 4u_{j-1} + u_{j-2}}{12\Delta x}. \quad (\text{B.46})$$

第 2 項は人工粘性項と呼ばれる.

B.5 非線形拡散方程式の離散化（差分法）

次のような非線形拡散方程式の初期値境界値問題に対する差分法を説明する：

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right), \quad t > 0, 0 < x < L, \quad (\text{B.47})$$

$$u(0, x) = u_0(x), \quad 0 \leq x \leq L, \quad (\text{B.48})$$

$$\frac{\partial u}{\partial x}(t, 0) = \frac{\partial u}{\partial x}(t, L) = 0. \quad (\text{B.49})$$

区間 $(0, L)$ を $(x_0, x_1, \dots, x_i, \dots, x_n)$ と等分割する．このとき $\Delta x = L/N$ となる．(B.47) の右辺に対して以下のように中心差分を行う．

$$\begin{aligned} \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) &\sim \frac{1}{\Delta x} \left(d(u(t, x + \Delta x/2)) \frac{\partial u(t, x + \Delta x/2)}{\partial x} - d(u(t, x - \Delta x/2)) \frac{\partial u(t, x - \Delta x/2)}{\partial x} \right) \\ &\sim \frac{1}{\Delta x} \left(d(u(t, x + \Delta x/2)) \frac{u(t, x + \Delta x) - u(t, x)}{\Delta x} - d(u(t, x - \Delta x/2)) \frac{u(t, x) - u(t, x - \Delta x)}{\Delta x} \right) \\ &= \frac{1}{\Delta x^2} \left(d \left(u(t, x + \frac{\Delta x}{2}) \right) u(t, x + \Delta x) - \left(d \left(u(t, x + \frac{\Delta x}{2}) \right) + d \left(u(t, x - \frac{\Delta x}{2}) \right) \right) u(t, x) \right. \\ &\quad \left. + d \left(u(t, x - \frac{\Delta x}{2}) \right) u(t, x - \Delta x) \right). \end{aligned} \quad (\text{B.50})$$

ここで

$$u \left(t, x + \frac{\Delta x}{2} \right) = \frac{u(t, x + \Delta x) + u(t, x)}{2}, \quad u \left(t, x - \frac{\Delta x}{2} \right) = \frac{u(t, x) + u(t, x - \Delta x)}{2}$$

とする．さらに， $u(k \times \Delta t, i \times \Delta x) = u_i^k(t)$ と記述すると，右辺は次のように与えられる．

$$\begin{aligned} &\frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) \\ &\sim \frac{1}{\Delta x^2} \left(d \left(\frac{u_{i+1}^k + u_i^k}{2} \right) u_{i+1}^k - \left(d \left(\frac{u_{i+1}^k + u_i^k}{2} \right) + d \left(\frac{u_i^k + u_{i-1}^k}{2} \right) \right) u_i^k + d \left(\frac{u_i^k + u_{i-1}^k}{2} \right) u_{i-1}^k \right). \end{aligned}$$

境界条件から

$$u_{-1}^k = u_1^k, \quad u_{n-1}^k = u_{n+1}^k, \quad (k = 1, 2, 3, \dots)$$

より， $i = 0$ のとき

$$\frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) \sim \frac{1}{\Delta x^2} \left(2d \left(\frac{u_1^k + u_0^k}{2} \right) u_1^k - \left(2d \left(\frac{u_1^k + u_0^k}{2} \right) \right) u_0^k \right)$$

となり， $i = n$ のとき

$$\frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) \sim \frac{1}{\Delta x^2} \left(2d \left(\frac{u_n^k + u_{n-1}^k}{2} \right) u_{n-1}^k - \left(2d \left(\frac{u_n^k + u_{n-1}^k}{2} \right) \right) u_n^k \right)$$

となる．

上記で示した非線形拡散方程式に対する差分方程式は有限体積法を用いると自然に導出できる．いま， $u(t, x)$ の空間離散化を考える．区間 $(0, L)$ を $(x_0, x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$ と分割し，図 B.1 のよう

に $u(t, x)$ を区分定数関数 $u_i(t)$ で近似する²。このとき、(B.47) の両辺を区間 $(0, L)$ で積分すると

$$\int_0^L \frac{\partial u}{\partial t} dx = \int_0^L \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) dx$$

となる。

$$\sum_{i=0}^N \int_{\frac{x_{i-1}+x_i}{2}}^{\frac{x_i+x_{i+1}}{2}} \frac{d}{dt} u dx = \sum_{i=0}^N \int_{\frac{x_{i-1}+x_i}{2}}^{\frac{x_i+x_{i+1}}{2}} \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) dx$$

と書き換えることができる。ただし、 $i=0$ のときは

$$\int_{x_0}^{\frac{x_0+x_1}{2}} \frac{\partial}{\partial t} u dx = \int_{x_0}^{\frac{x_0+x_1}{2}} \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) dx,$$

$i=n$ のときは

$$\int_{\frac{x_{n-1}+x_n}{2}}^{x_n} \frac{\partial}{\partial t} u dx = \int_{\frac{x_{n-1}+x_n}{2}}^{x_n} \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) dx$$

とする。ここで、区分定数関数 $u_i(t)$ を用いると $1 \leq i \leq n-1$ において、左辺は

$$\frac{d}{dt} u_i(t) \int_{\frac{x_{i-1}+x_i}{2}}^{\frac{x_i+x_{i+1}}{2}} dx = \frac{d}{dt} u_i(t) \times \frac{x_{i+1} - x_{i-1}}{2}$$

となり、右辺は

$$\int_{\frac{x_{i-1}+x_i}{2}}^{\frac{x_i+x_{i+1}}{2}} \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) dx = \left[d(u) \frac{\partial u}{\partial x} \right]_{\frac{x_{i-1}+x_i}{2}}^{\frac{x_i+x_{i+1}}{2}} \quad (\text{B.51})$$

$$= d \left(u \left(t, \frac{x_i + x_{i+1}}{2} \right) \right) \frac{\partial}{\partial x} u \left(t, \frac{x_i + x_{i+1}}{2} \right) - d \left(u \left(t, \frac{x_{i-1} + x_i}{2} \right) \right) \frac{\partial}{\partial x} u \left(t, \frac{x_{i-1} + x_i}{2} \right). \quad (\text{B.52})$$

ここで

$$\begin{aligned} \frac{\partial}{\partial x} u \left(t, \frac{x_i + x_{i+1}}{2} \right) &\sim \frac{u_{i+1} - u_i}{x_{i+1} - x_i}, \\ \frac{\partial}{\partial x} u \left(t, \frac{x_{i-1} + x_i}{2} \right) &\sim \frac{u_i - u_{i-1}}{x_i - x_{i-1}} \end{aligned}$$

と近似する。また、図 B.1 より

$$u \left(t, \frac{x_i + x_{i+1}}{2} \right) = \frac{u_i(t) + u_{i+1}(t)}{2}, \quad u \left(t, \frac{x_{i-1} + x_i}{2} \right) = \frac{u_{i-1}(t) + u_i(t)}{2}$$

となるので、(B.52) は

$$= d \left(\frac{u_i(t) + u_{i+1}(t)}{2} \right) \frac{u_{i+1} - u_i}{x_{i+1} - x_i} - d \left(\frac{u_{i-1}(t) + u_i(t)}{2} \right) \frac{u_i - u_{i-1}}{x_i - x_{i-1}} \quad (\text{B.53})$$

となる。(B.51)、(B.53) から

$$\frac{d}{dt} u_i(t) = d \left(\frac{u_i(t) + u_{i+1}(t)}{2} \right) \frac{2(u_{i+1} - u_i)}{(x_{i+1} - x_i)(x_{i+1} - x_{i-1})} - d \left(\frac{u_{i-1}(t) + u_i(t)}{2} \right) \frac{2(u_i - u_{i-1})}{(x_i - x_{i-1})(x_{i+1} - x_{i-1})}$$

を得る。ここで、等間隔での差分化 $\Delta x = x_i - x_{i-1} (0 \leq i \leq N)$ を考えると

$$\frac{d}{dt} u_i(t) = d \left(\frac{u_i(t) + u_{i+1}(t)}{2} \right) \frac{u_{i+1} - u_i}{\Delta x^2} - d \left(\frac{u_{i-1}(t) + u_i(t)}{2} \right) \frac{u_i - u_{i-1}}{\Delta x^2} \quad (\text{B.54})$$

となり、非線形拡散方程式に対する差分方程式を得る。 $i=0$ と $i=n$ の場合は、境界条件を考慮することによって

$$\int_{x_0}^{\frac{x_0+x_1}{2}} \frac{\partial}{\partial t} u dx = \frac{d}{dt} u_0(t) \times \frac{x_1 - x_0}{2},$$

²ここでの分割は等間隔の分割である必要性はない。

$$\begin{aligned} \int_{x_0}^{\frac{x_0+x_1}{2}} \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) dx &= d \left(u(t, \frac{x_0+x_1}{2}) \right) \frac{\partial}{\partial x} u(t, \frac{x_0+x_1}{2}) \\ &= d \left(\frac{u_0(t) + u_1(t)}{2} \right) \frac{u_1 - u_0}{x_1 - x_0}. \end{aligned} \quad (\text{B.55})$$

$$\int_{\frac{x_{n-1}+x_n}{2}}^{x_n} \frac{\partial}{\partial t} u dx = \frac{d}{dt} u_n(t) \times \frac{x_n - x_{n-1}}{2},$$

$$\begin{aligned} \int_{\frac{x_{n-1}+x_n}{2}}^{x_n} \frac{\partial}{\partial x} \left(d(u) \frac{\partial u}{\partial x} \right) dx &= -d \left(u(t, \frac{x_{n-1}+x_n}{2}) \right) \frac{\partial}{\partial x} u(t, \frac{x_{n-1}+x_n}{2}) \\ &= -d \left(\frac{u_{n-1}(t) + u_n(t)}{2} \right) \frac{u_n - u_{n-1}}{x_n - x_{n-1}}. \end{aligned} \quad (\text{B.56})$$

ここで、等間隔の差分化を考えると、 $i=0$ のとき

$$\frac{d}{dt} u_0(t) = d \left(\frac{u_0(t) + u_1(t)}{2} \right) \frac{2(u_1 - u_0)}{\Delta x^2}. \quad (\text{B.57})$$

$i=n$ のとき

$$\frac{d}{dt} u_n(t) = -d \left(\frac{u_{n-1}(t) + u_n(t)}{2} \right) \frac{2(u_{n-1} - u_n)}{\Delta x^2} \quad (\text{B.58})$$

となり、差分法による近似と同じ差分方程式を得る。

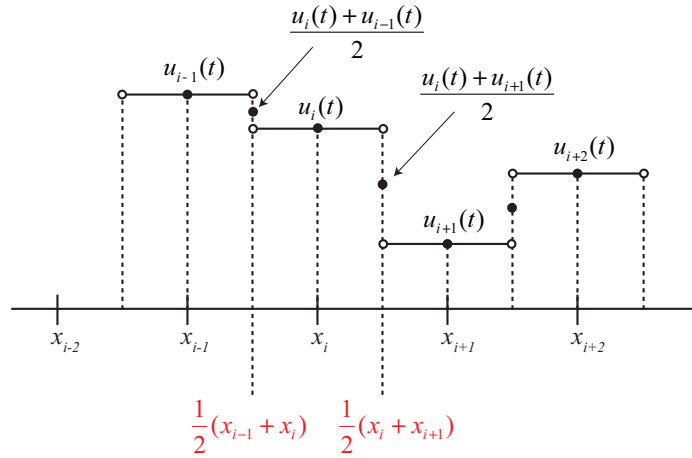


図 B.1: 有限体積法を用いるための区分定数近似

空間 2 次元問題や空間 3 次元問題でも同様の方法で導出することができる。

関連図書

- [1] “やさしく学べる C 言語入門 - 基礎から数値計算入門まで - ”, 皆本晃弥, サイエンス社
- [2] “C 言語による数値計算入門 - 解法・アルゴリズム・プログラム - ”, 皆本晃弥, サイエンス社
- [3] “偏微分方程式の数値シミュレーション”, 登坂 宣好, 大西和榮, 東京大学出版会.
- [4] “コンピュータによる偏微分方程式の解法 [新訂版] ”, G.D. スミス, サイエンス社.
- [5] “Fortran & C 言語によるシミュレーション技法入門”, 矢部 孝, 井門 俊治, 日刊工業新聞社.
- [6] “岩波講座 応用数学「線形計算」”, 森 正武, 杉原 正顯, 室田 一雄, 岩波書店.
- [7] “Fortran77 による数値計算ソフトウェア”, 渡部 力, 名取 亮, 小国 力, 丸善.
- [8] “行列計算ソフトウェア”, 小国 力編著, 丸善.
- [9] “NUMERICAL RECIPES in C”, 技術評論社.
- [10] “数値解析入門”, 山本 哲朗, サイエンス社.
- [11] “数値解析の基礎”, 篠原 能材, 日新出版.
- [12] “数値計算の常識”, 伊理正夫, 藤野和建, 共立出版.
- [13] “ <http://www.math.meiji.ac.jp/mk/labo/text/heat-fdm-1.pdf> ”.
- [14] “コンピュータ流体力学”, フレッチャー, シュプリンガーフェアラー東京.
- [15] “使いこなす GNUPLOT Ver4.0 対応”, 大竹 敢, テクノプレス.
- [16] “<http://ayapin.film.s.dendai.ac.jp/matuda/TeX/PDF/40th.pdf>”
- [17] “<http://www-sens.sys.es.osaka-u.ac.jp/wakate/tutorial/group3/gnuplot/>”
- [18] “OpenGL プログラミングガイド”, OpenGL ARB, アジソンウェスレイ.
- [19] “OpenGL リファレンスマニュアル”, OpenGL ARB, アジソンウェスレイ.
- [20] “<http://www.wakayama-u.ac.jp/tokoi/opengl/libglut.html>”.
- [21] “<ftp://ftp.ryukoku.ac.jp/Ryukoku/software/math/glsc-3.5.tar.Z>”.
- [22] “<ftp://ftp.ryukoku.ac.jp/Ryukoku/software/math/glsc-3.3.man.tar.Z>”.
- [23] “<http://www-mmc.es.hokudai.ac.jp/masakazu/>”
- [24] “<http://nalab.mind.meiji.ac.jp/mk/labo/text/glsc3d.pdf> ”
- [25] “非平衡系の科学 I”, 北原 和夫, 吉川 研一, 講談社.
- [26] “非平衡系の科学 III”, 三池 秀敏, 森 義仁, 山口 智彦, 講談社.
- [27] “非線形科学”, 吉川 研一, 学会出版センター.

おわりに

2018年5月, ffmpeg のインストール部分を書き換えました。今後は静止画から動画を作る部分を再更新する予定ですが, いつ完成するのか…

2017年6月, この解説書の可視化部分についてはかなり古くなりました。今後改定したいと思います。

2012年4月から北海道大学電子科学研究所に異動しました。本解説書は, 北海道電子科学研究所数理科学研究部門動的数理モデリング研究分野で配布している数値計算の入門書です。本書は入門編, 発展編, 並列計算編の3編から成っていますが, 現在のところ完成しているのは入門編だけであり, 発展編, 並列計算編の完成には到っていません³。基本的に非公開の入門書でしたが, 金沢大学に所属していたときに, いろいろな事情⁴があって一般に公開することになりました。非公開文書を公開したからといって真面目に修正をしているわけではないので, 多くの誤植や間違いがあると思います。一部修正しましたが, まだまだ間違いがあると思いますので, ご指摘して頂ければ修正したいと思います。

数値計算結果の可視化部分がかかなり古くなってきたので, 少し更新しました。全面改定には至っていませんが, Linux, Mac を使って数値計算するためには十分であると思います。Cygwin 上での数値計算の可視化はこれから整備していきたいと思います。

2020年6月

長山 雅晴

³完成しない可能性大です

⁴さきがけ数学塾で使用することもその一つです。