流体シミュレーション入門

中心差分法とガウス・ザイデル法による解法

目次

第 1 章

序論

ここでは、流体の運動の数値解析について取り扱っていく。流体の運動は古くから研究されてきたが、一般に現象の厳密解が求まることは稀有だといえる。しかし、特に理工においては現象を解明するのに必ずしも厳密解は必要ではない。対象の現象に対する定量的な理解が得られれば、十分解明に寄与することができる。あるいは、厳密解が求まっている現象だったとしても、それに定量的な理解を付加することで更なる理解や発見が期待できる、ということができるかもしれない。そういった意味で、対象の現象に対して数値解析によるアプローチの手段をもっていることは大変有用なだけでなく、強力な武器になるのである。流体の数値計算の本は数多く出版されているが、入門書として書かれている本は少ないように感じた。ここでは、新しいスキームや理論体系を構築するのではなく、流体の数値計算に必要な基本的な概念やスキームを理解することに重点を置き話を進めていくことにする。

1.1 スキームの種類

流体の数値解析は、非圧縮性流体か圧縮性流体かによって、そのスキームが大きく異なる。これは基礎となる支配方程式が異なることに起因してる。一般に非圧縮性流体の方が方程式系がきれいであり、理解が容易であることから、ここでは非圧縮性流体について考えていくことにする。次に流れ場の数値解析法であるが、大きく分けて差分法と有限要素法に分けることができる。有限要素法は複雑な条件や現象の解明に適しているが、そのスキームもまた難解になってしまうため、ここでは基本的で応用範囲の広い差分法について取り扱っていくことにする。

以下に各スキームの種類をあげておく。この分け方は必ずしも適切ではないが、大まかな内訳を知ることができるだろう。また、この他にも多くのスキームが存在する。中には流体に限定したものではなく、一般的な偏微分方程式を差分化するためのスキーム等も含まれている。ここでは、最も基本的な MAC(Marker and Cell) 法 [1, 2] について取り扱っていくつもりである。MAC 法は 1965 年に Harlow と Welch によって開発されたスキームである。その後、多くの改良が重ねられ、現在ではほぼ完成されたスキームとなっ

ている。MAC 法を学ぶことで、基本的な流体スキームの理解と他の高度なスキームへの拡張が期待できるわけである。

非圧縮性流体

- MAC (Marker and Cell) 法
- SMAC 法
- SIMPLE 法

圧縮性流体

- TVD (Total variation diminishing) 法
- non-TVD 法
- CIP 法

粒子法

- PIC (Particle in Cell) 法
- MPS (Moving particle semi-implicit method) 法
- SPH 法

Fig.1：各スキームの種類

1.2 目的

ここでは理論的な基本方程式から差分方程式を導出し、簡単な 2 次元での流体シミュレーションを設計、開発していく。その際、できるだけ難しい議論は避け、基本的な概念やスキームのみを導入していくことにする。数値解析の大まかな流れを掴んでいくことが本稿での大きな目標であり、目的である。また、設計したプログラムを使って、実際に流体シミュレーションを行っていく。本稿が単なるレポートに留まらず、これから流体の数値解析を学ぶ人の入門書となることを願う。

第 2 章

基礎方程式

まず、最初に非圧縮性流体における基礎方程式を挙げる。

$$\frac{\partial \rho}{\partial t} + (\boldsymbol{\nabla} \cdot \rho \boldsymbol{v}) = 0 \tag{2.1}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v} = -\frac{1}{\rho}\boldsymbol{\nabla} P + \nu \boldsymbol{\nabla}^2 \boldsymbol{v} \tag{2.2}$$

(2.1) 式を連続の方程式、(2.2) 式を Navier-Stokes 方程式という。ここで ρ は密度、P は圧力、ν は動粘性係数である。

2.1　圧力方程式の導出

次に圧力に関する方程式を導出する。ここでは、2 次元における流体の運動について考えていくことにする。(2.2) 式を各成分ごとに表記すると

$$\frac{\partial v_x}{\partial t} + v_x \frac{\partial v_x}{\partial x} + v_y \frac{\partial v_x}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\left(\frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2}\right) \tag{2.3}$$

$$\frac{\partial v_y}{\partial t} + v_x \frac{\partial v_y}{\partial x} + v_y \frac{\partial v_y}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial y} + \nu\left(\frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2}\right) \tag{2.4}$$

となる。ここで、両式の発散をとる。

$$\frac{\partial}{\partial t}\left(\frac{\partial v_x}{\partial x}\right) + \frac{\partial^2 v_x^2}{\partial x^2} + \frac{\partial^2 v_x v_y}{\partial x \partial y} = -\frac{1}{\rho}\frac{\partial^2 P}{\partial x^2} + \nu \frac{\partial}{\partial x}\boldsymbol{\nabla}^2 v_x \tag{2.5}$$

$$\frac{\partial}{\partial t}\left(\frac{\partial v_y}{\partial y}\right) + \frac{\partial^2 v_x v_y}{\partial x \partial y} + \frac{\partial^2 v_y^2}{\partial y^2} = -\frac{1}{\rho}\frac{\partial^2 P}{\partial y^2} + \nu \frac{\partial}{\partial y}\boldsymbol{\nabla}^2 v_y \tag{2.6}$$

次に (2.5) 式と (2.6) 式の和をとると

$$\frac{\partial}{\partial t}\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right) + \frac{\partial^2 v_x^2}{\partial x^2} + 2\frac{\partial^2 v_x v_y}{\partial x \partial y} + \frac{\partial^2 v_y^2}{\partial y^2} = -\frac{1}{\rho}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)P + \nu \boldsymbol{\nabla}^2 \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right) \tag{2.7}$$

と書ける。ここで

$$D = \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} \right) \tag{2.8}$$

$$p = \frac{P}{\rho} \tag{2.9}$$

と置き、(2.7) 式を p について解けば

$$\boldsymbol{\nabla}^2 p = -\left(\frac{\partial^2 v_x^2}{\partial x^2} + 2\frac{\partial^2 v_x v_y}{\partial x \partial y} + \frac{\partial^2 v_y^2}{\partial y^2} \right) + \nu \boldsymbol{\nabla}^2 D - \frac{\partial}{\partial t} D \tag{2.10}$$

圧力方程式が得られる。これは圧力 p に関する Poisson 方程式になっている。理論的には右辺の D は (2.1) 式より $D = 0$ を満たしていなければならない。しかし、数値解を求める場合試算回数の増大に伴って誤差が増え、$D = 0$ を満たさなくなる。それを最初から $D = 0$ とおいてしまうと精度を悪くするだけだという指摘がなされている [3]。そこで、ここでは補正項として D の項を残している。

2.2 支配方程式

以上より、2 次元での非圧縮性流体シミュレーションにおいて用いられる方程式は

$$\frac{\partial v_x}{\partial t} + v_x \frac{\partial v_x}{\partial x} + v_y \frac{\partial v_x}{\partial y} = -\frac{\partial p}{\partial x} + \nu \boldsymbol{\nabla}^2 v_x \tag{2.11}$$

$$\frac{\partial v_y}{\partial t} + v_x \frac{\partial v_y}{\partial x} + v_y \frac{\partial v_y}{\partial y} = -\frac{\partial p}{\partial y} + \nu \boldsymbol{\nabla}^2 v_y \tag{2.12}$$

$$\boldsymbol{\nabla}^2 p = -\left(\frac{\partial^2 v_x^2}{\partial x^2} + 2\frac{\partial^2 v_x v_y}{\partial x \partial y} + \frac{\partial^2 v_y^2}{\partial y^2} \right) + \nu \boldsymbol{\nabla}^2 D - \frac{\partial}{\partial t} D \tag{2.13}$$

となる。

第 3 章

シミュレーション

この章では、前節に導出した方程式を差分化し、実際にプログラム上で用いる差分方程式を導出する。差分法は、比較的導出が簡単で、直感的に理解しやすい中心差分法を用いることにする。また、プログラミングに必要な概念とスキームも同時に導入していく。

3.1 スタッガード格子

コンピュータ上で偏微分方程式を取り扱うには、方程式を差分化しなければならない。これはコンピュータが有限な値しか扱えないことに起因している。差分化が行われた時点で、もはや空間や時間は連続的でなく、離散的なものとなる。ここでは、スタッガード格子を導入し、空間の離散値を Fig.2 のように定義することにする。スタッガード格子とは、1 つのセルに対して物理量をズラして定義するようなメッシュ体系のことをいう。この場合、圧力はセルの中心で定義され、速度 (v_x, v_y) は各セル境界上で定義されることとなる。スタッガード格子は単純に空間を離散化するだけでなく、圧力や速度の数値振動を防ぐ効果もあることが分かっている。

Fig.2 における白色の部分が実質上のシミュレーション領域となる。この領域内において、流体の数値解析が行われることとなる。外側の灰色の部分がシミュレーション領域の境界を表している。境界上の格子点には設定した境界条件を適応する必要がある。詳しくは後述する。プログラム上で、格子点は配列を用いた変数によって記述することになる。例えば、C 言語で書くと

```
static  double  P[imx + 2][jmx + 2]
```

となる。ここで imx、jmx は x 方向、y 方向の最大格子点数を表している。2 が足されているのは、境界条件のために各方向に 2 個余分に格子点が必要だからである。これは、ここで用いる差分法が中心差分法であることに起因しており、ある格子点上の物理量を求めるために、その周囲 2 個分の格子点が必要だからである。3 次精度の差分法等の場合はそれに応じて、境界における格子点数を増やす必要がある。

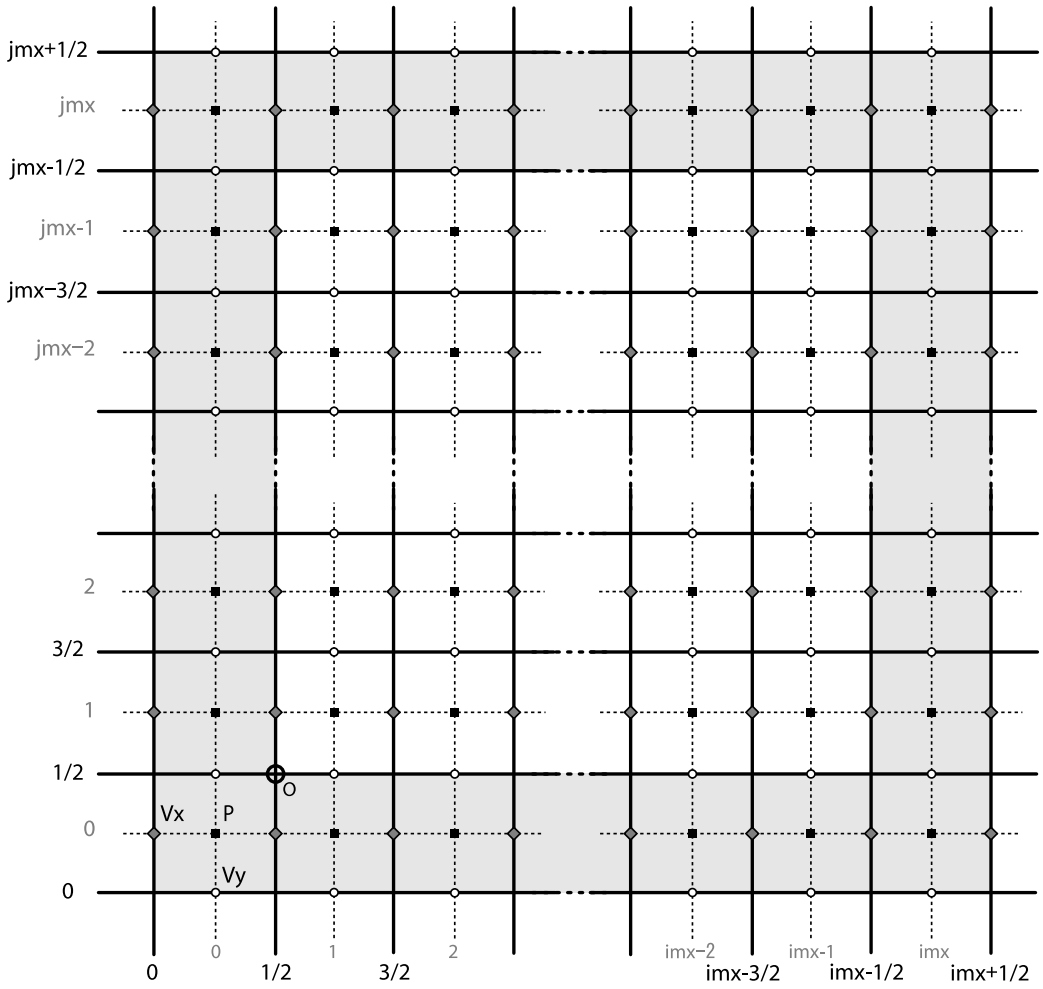

Fig.2：スタッガード格子の定義

3.2 差分方程式

3.2.1 Navier-Stokes 方程式の差分化

(2.3), (2.4) 式より

$$\frac{\partial v_x}{\partial t} + v_x\frac{\partial v_x}{\partial x} + v_y\frac{\partial v_x}{\partial y} = -\frac{\partial p}{\partial x} + \nu\boldsymbol{\nabla}^2 v_x \tag{3.1}$$

$$\frac{\partial v_y}{\partial t} + v_x\frac{\partial v_y}{\partial x} + v_y\frac{\partial v_y}{\partial y} = -\frac{\partial p}{\partial y} + \nu\boldsymbol{\nabla}^2 v_y \tag{3.2}$$

中心差分法を用いて Navier-Stokes 方程式を差分化する。まず、最初に各項を差分化しておく。

$$\begin{aligned}
\mathrm{IIx}_{i+\frac{1}{2},j} &= \left(v_x\frac{\partial v_x}{\partial x}\right)_{i+\frac{1}{2},j} \\
&= \frac{1}{\Delta}\left\{v^2_{x(i+\frac{1}{2},j)} - v^2_{x(i,j)}\right\} && (3.3)\\
\mathrm{JIy}_{i+\frac{1}{2},j} &= \left(v_y\frac{\partial v_x}{\partial y}\right)_{i+\frac{1}{2},j} \\
&= \frac{1}{\Delta}\left\{v_{x(i+\frac{1}{2},j+\frac{1}{2})}v_{y(i+\frac{1}{2},j+\frac{1}{2})} - v_{x(i+\frac{1}{2},j-\frac{1}{2})}v_{y(i+\frac{1}{2},j-\frac{1}{2})}\right\} && (3.4)\\
\mathrm{IJx}_{i,j+\frac{1}{2}} &= \left(v_x\frac{\partial v_y}{\partial x}\right)_{i,j+\frac{1}{2}} \\
&= \frac{1}{\Delta}\left\{v_{x(i+\frac{1}{2},j+\frac{1}{2})}v_{y(i+\frac{1}{2},j+\frac{1}{2})} - v_{x(i-\frac{1}{2},j+\frac{1}{2})}v_{y(i-\frac{1}{2},j+\frac{1}{2})}\right\} && (3.5)\\
\mathrm{JJy}_{\mathrm{i,j}+\frac{1}{2}} &= \left(v_y\frac{\partial v_y}{\partial y}\right)_{i,j+\frac{1}{2}} \\
&= \frac{1}{\Delta}\left\{v^2_{y(i,j+\frac{1}{2})} - v^2_{y(i,j)}\right\} && (3.6)\\
\mathrm{Px} &= \frac{\partial p}{\partial x} \\
&= \frac{1}{\Delta}\left\{p_{i+1,j} - p_{i,j}\right\} && (3.7)\\
\mathrm{Py} &= \frac{\partial p}{\partial y} \\
&= \frac{1}{\Delta}\left\{p_{i,j+1} - p_{i,j}\right\} && (3.8)\\
\mathrm{Ixx}_{\mathrm{i}+\frac{1}{2},\mathrm{j}} &= \left(\frac{\partial^2 v_x}{\partial x^2}\right) \\
&= \frac{1}{\Delta^2}\left\{v_{x(i+\frac{3}{2},j)} - 2v_{x(i+\frac{1}{2},j)} + v_{x(i-\frac{1}{2},j)}\right\} && (3.9)\\
\mathrm{Iyy}_{\mathrm{i}+\frac{1}{2},\mathrm{j}} &= \left(\frac{\partial^2 v_x}{\partial y^2}\right) \\
&= \frac{1}{\Delta^2}\left\{v_{x(i+\frac{1}{2},j+1)} - 2v_{x(i+\frac{1}{2},j)} + v_{x(i+\frac{1}{2},j-1)}\right\} && (3.10)\\
\mathrm{Jxx}_{\mathrm{i,j}+\frac{1}{2}} &= \left(\frac{\partial^2 v_y}{\partial x^2}\right) \\
&= \frac{1}{\Delta^2}\left\{v_{y(i+1,j+\frac{1}{2})} - 2v_{y(i,j+\frac{1}{2})} + v_{y(i-1,j+\frac{1}{2})}\right\} && (3.11)\\
\mathrm{Jyy}_{\mathrm{i,j}+\frac{1}{2}} &= \left(\frac{\partial^2 v_y}{\partial y^2}\right) \\
&= \frac{1}{\Delta^2}\left\{v_{y(i,j+\frac{3}{2})} - 2v_{y(i,j+\frac{1}{2})} + v_{y(i,j-\frac{1}{2})}\right\} && (3.12)
\end{aligned}$$

ここでは、簡単のため空間の刻み幅を $\Delta = \Delta x = \Delta y$ とした。時間の刻み幅を Δt とする。定義されていない点での速度は

$$v_{x(i+\frac{1}{2},j+\frac{1}{2})} = \frac{1}{2}\Big\{v_{x(i+\frac{1}{2},j+1)} + v_{x(i+\frac{1}{2},j)}\Big\} \tag{3.13}$$

$$v_{y(i+\frac{1}{2},j+\frac{1}{2})} = \frac{1}{2}\Big\{v_{y(i+1,j+\frac{1}{2})} + v_{y(i,j+\frac{1}{2})}\Big\} \tag{3.14}$$

$$v_{x(i,j)} = \frac{1}{2}\Big\{v_{x(i+\frac{1}{2},j)} + v_{x(i-\frac{1}{2},j)}\Big\} \tag{3.15}$$

$$v_{y(i,j)} = \frac{1}{2}\Big\{v_{y(i,j+\frac{1}{2})} + v_{y(i,j-\frac{1}{2})}\Big\} \tag{3.16}$$

のように、各定義点との平均から求めることができる。以上の式を用いて、次の時間の速度は、

$$v^{n+1}_{x(i+\frac{1}{2},j)} = v^{n}_{x(i+\frac{1}{2},j)} - \Delta t(\mathrm{IIx} + \mathrm{JIy}) - \mathrm{Px}\Delta t + \nu\Delta t(\mathrm{Ixx} + \mathrm{Iyy}) \tag{3.17}$$

$$v^{n+1}_{y(i,j+\frac{1}{2})} = v^{n}_{y(i,j+\frac{1}{2})} - \Delta t(\mathrm{IJx} + \mathrm{JJy}) - \mathrm{Py}\Delta t + \nu\Delta t(\mathrm{Jxx} + \mathrm{Jyy}) \tag{3.18}$$

より算出することができる。プログラム上では、(3.13)　(3.16) 式を予め求め、別の配列に格納しておくと便利である。また、次の時間の速度を求める際も、(3.3)　(3.12) 式をそれぞれ変数に格納し、最後に (3.17), (3.18) 式を用いれば、プログラムを見やすく書くことができる。

3.2.2 圧力方程式の差分化

次に圧力方程式の差分化を行う。ここでも中心差分法を用いて差分化を行っていく。

$$\boldsymbol{\nabla}^2 p = -\left(\frac{\partial^2 v_x^2}{\partial x^2} + 2\frac{\partial^2 v_x v_y}{\partial x \partial y} + \frac{\partial^2 v_y^2}{\partial y^2}\right) + \nu\boldsymbol{\nabla}^2 D - \frac{\partial}{\partial t}D \tag{3.19}$$

前小節と同様に、最初に各項を差分化する

$$\boldsymbol{\nabla}^2 p = \frac{1}{\Delta^2}\Big\{p_{i+1,j} + p_{i-1,j} - 4p_{i,j} + p_{i,j+1} + p_{i,j-1}\Big\} \tag{3.20}$$

$$\begin{aligned}\mathrm{Vxx_{i,j}} &= \left(\frac{\partial^2 v_x^2}{\partial x^2}\right)_{i,j} \\ &= \frac{1}{\Delta^2}\Big\{v^2_{x(i+1,j)} - 2v^2_{x(i,j)} + v^2_{x(i-1,j)}\Big\}\end{aligned} \tag{3.21}$$

$$\begin{aligned}\mathrm{Vyy_{i,j}} &= \left(\frac{\partial^2 v_y^2}{\partial y^2}\right)_{i,j} \\ &= \frac{1}{\Delta^2}\Big\{v^2_{y(i,j+1)} - 2v^2_{y(i,j)} + v^2_{y(i,j-1)}\Big\}\end{aligned} \tag{3.22}$$

$$\begin{aligned}
\mathrm{Vxy_{i,j}} &= \left(2\frac{\partial^2 v_x v_y}{\partial x \partial y}\right)_{i,j} \\
&= \frac{2}{\Delta^2}\Big\{v_{x(i+\frac{1}{2},j+\frac{1}{2})}v_{y(i+\frac{1}{2},j+\frac{1}{2})} - v_{x(i+\frac{1}{2},j-\frac{1}{2})}v_{y(i+\frac{1}{2},j-\frac{1}{2})} \\
&\quad - v_{x(i-\frac{1}{2},j+\frac{1}{2})}v_{y(i-\frac{1}{2},j+\frac{1}{2})} + v_{x(i-\frac{1}{2},j-\frac{1}{2})}v_{y(i-\frac{1}{2},j-\frac{1}{2})}\Big\} \qquad (3.23) \\
\mathrm{D_{i,j}} &= \left(\frac{\partial}{\partial t}D\right)_{i,j} \\
&= \frac{1}{\Delta}\Big\{v_{x(i+\frac{1}{2},j)} - v_{x(i-\frac{1}{2},j)} + v_{y(i,j+\frac{1}{2})} - v_{y(i,j-\frac{1}{2})}\Big\} \qquad (3.24) \\
\mathrm{Dxx_{i,j}} &= \left(\frac{\partial^2 D}{\partial x^2}\right)_{i,j} \\
&= \frac{1}{\Delta^2}\Big\{D_{i+1,j} - 2D_{i,j} + D_{i-1,j}\Big\} \qquad (3.25) \\
\mathrm{Dyy_{i,j}} &= \left(\frac{\partial^2 D}{\partial y^2}\right)_{i,j} \\
&= \frac{1}{\Delta^2}\Big\{D_{i,j+1} - 2D_{i,j} + D_{i,j-1}\Big\} \qquad (3.26)
\end{aligned}$$

となる。2 階微分の項は分母が Δ^2 になっている。上式を用いてまとめると

$$\begin{aligned}
&p_{i+1,j} + p_{i-1,j} - 4p_{i,j} + p_{i,j+1} + p_{i,j-1} \\
&\qquad = -\Delta^2(\mathrm{Vxx} + 2\mathrm{Vxy} + \mathrm{Vyy}) + \frac{\Delta^2}{\Delta t}D_{i,j} + \nu(\mathrm{Dxx} + \mathrm{Dyy}) \qquad (3.27)
\end{aligned}$$

となる。この式は前小節の用に単純な代数学的な式にすることができず、$(i \times j)$ 元の連立方程式の形になっている。こうなると単純な代入操作だけでは、この方程式を解くことはできない。詳細は後述するが、解析的な手法ではなく、数値的なスキームによってこの方程式を解くことにする。

3.3 境界条件

3.3.1 速度の境界条件

次に境界上における速度の振る舞いを設定する。境界条件には大きく分けて固着 (no-slip) 条件と自由すべり (free-slip) 条件がある。今考えている流体には粘性が存在することから、壁面での振る舞いは固着条件を採用すべきであるが、簡単のためにここでは自由すべり条件を採用することにする。

流体は壁面を通過しないことから、壁面上の速度はつねに 0 に設定する。境界上の格子点 (Fig.2 における灰色部分の格子点) の設定は、自由すべりか固着かによって異なってくる。自由すべりの場合は、境界上の格子点の速度を以下のように設定すればよい。

- 水平方向

$$\begin{aligned} v_{x(1,j)} &= 0 & v_{x(\mathrm{imx},j)} &= 0 \\ v_{x(0,j)} &= -v_{x(2,j)} & v_{x(\mathrm{imx}+1,j)} &= -v_{x(\mathrm{imx}-1,j)} \\ v_{y(0,j)} &= v_{y(1,j)} & v_{y(\mathrm{imx},j)} &= v_{y(\mathrm{imx}-1,j)} \end{aligned}$$

- 垂直方向

$$\begin{aligned} v_{y(i,1)} &= 0 & v_{y(i,\mathrm{jmx})} &= 0 \\ v_{x(i,0)} &= v_{x(i,1)} & v_{x(i,\mathrm{jmx})} &= v_{x(i,\mathrm{jmx}-1)} \\ v_{y(i,0)} &= -v_{y(i,2)} & v_{y(i,\mathrm{jmx}+1)} &= -v_{y(i,\mathrm{jmx}-1)} \end{aligned}$$

3.3.2 圧力の境界条件

圧力と速度の境界条件は終始一貫してなければならない。なぜなら、圧力方程式の数値決定に、境界上の速度が必要となってくるからである。自由すべり条件の場合、圧力の設定は速度対して変わらない。

- 水平方向

$$p_{0,j} = p_{1,j} \qquad p_{\mathrm{imx},j} = p_{\mathrm{imx}-1,j}$$

- 垂直方向

$$p_{i,0} = p_{i,1} \qquad p_{i,\mathrm{jmx}} = p_{i,\mathrm{jmx}-1}$$

圧力の格子点は壁面上には存在しないので、上記の設定で足りる。

3.4 ガウス・ザイデル法

次に (3.27) 式を解くことを考える。左辺に注目し、右辺はまとめて $b_{i,j}$ とおくと

$$p_{i+1,j} + p_{i-1,j} - 4p_{i,j} + p_{i,j+1} + p_{i,j-1} = b_{i,j} \tag{3.28}$$

と書ける。このとき、左辺を上手く $p_{i,j} = p_{\mathrm{imx}\cdot i+j}$ のような規則を設けることで、$p_{i,j}$ を 2 次元配列から 1 次元配列へと書き換えることができる。すると以下のように行列の形で (3.27) 式を書くことができる。

$$AQ = B \tag{3.29}$$

ここで、Q は $p_{i,j}$ を任意の規則を用いて変換し、括り出した 1 次元行列である。B は同様の規則を用いて 1 次元行列に変換した $b_{i,j}$ である。A は $p_{i,j}$ を括り出したときに残る定数の 2 次元行列である。ただし、A の形は自明ではない。ここでは、厳密解を導出するのではなく、なんらかの適当な近似解を見つけることを考える。今 k 番目の近似解を Q^k と書くことにする。行列 A を形式的に以下のように分解する。

$$A = D + N \tag{3.30}$$

ここで、D は行列 A の対角要素であり、N はその残りである。これを用いると以下のように書ける。

$$DQ^{k+1} = -NQ^k + B \tag{3.31}$$

D は対角行列であるので、この方程式は解くことができる [5]。$Q^{k+1} = Q^k$ のとき、元の方程式を満たしていることが分かる。プログラム上では、収束判定として $|Q^{k+1} - Q^k|$ を計算し、その値が設定した値よりも小さかったら、収束したものとする。

3.4.1 連立方程式の解法

以上の議論を踏まえて、圧力方程式を解く前に、簡単な連立方程式を解くことを考える。

$$\begin{cases} 5x_0 - 4x_1 - 2x_2 + x_3 & = 5 \\ 2x_0 + 3x_1 + x_2 + 2x_3 & = 21 \\ -2x_0 + 2x_1 + 8x_2 + x_3 & = 16 \\ 4x_0 + x_1 - 3x_2 + 5x_3 & = 18 \end{cases} \tag{3.32}$$

(3.31) 式をこの場合に当てはめると

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n-1} a_{ij} x_j^k \right) \tag{3.33}$$

による反復計算を行えばよい。このままだと少しわかりずらいので、プログラムで表記すると

```
for(;;){
  err = 0.0;
  for(i=0;i<N;i++){
    sum = b[i];
    for(j=0;j<N;j++)
      sum -= a[i][j]*x[j];

    sum /= a[i][i];
    x[i] += sum;

    err  += fabs(sum);
  }
  printf("%e %e %e %e\n",x[0],x[1],x[2],x[3]);
  if(err < 1.0E-6) break;
}
```

このプログラムでは $|x^{k+1} - x^k| < 10^{-6}$ のとき、収束したと判定している。上記のプロ

グラムを実行すれば、

$$x_0 = 4.000\text{e+}00 \tag{3.34}$$

$$x_1 = 3.000\text{e+}00 \tag{3.35}$$

$$x_2 = 2.000\text{e+}00 \tag{3.36}$$

$$x_3 = 1.000\text{e+}00 \tag{3.37}$$

へと値が収束していく様子が分かる。

3.4.2 圧力方程式の解法

圧力方程式を解く場合は、(3.28) 式を

$$p_{i,j} = \frac{1}{4}\Big\{p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - b_{i,j}\Big\} \tag{3.38}$$

のように変形し、ガウス・ザイデル法にかければよい。プログラム上では

```
for(;;)
  {
    err = 0.0;
    for(i=1;i<=imx-1;i++)
      {
        for(j=1;j<=jmx-1;j++)
          {
            sum = p[i][j];
            p[i][j] = (p[i][j-1]+p[i][j+1]+p[i+1][j]+p[i-1][j]
                      -b[i][j])/4.0;
          }
        err  += fabs(sum);
       }
     if(err < 1.0E-6) break;
   }
```

のようにすればいい。ループ回数がそれぞれ 1　imx-1,1　jmx-1 なのは、圧力の格子点がシミュレーション領域内にある範囲を反復計算する必要があるからである。

3.5 シミュレーションの流れ

最後に全体のシミュレーションの流れを示す。境界条件の設定は、圧力・速度の各計算過程において行っている。初期化では全変数の初期化と初期値の設定を行う。

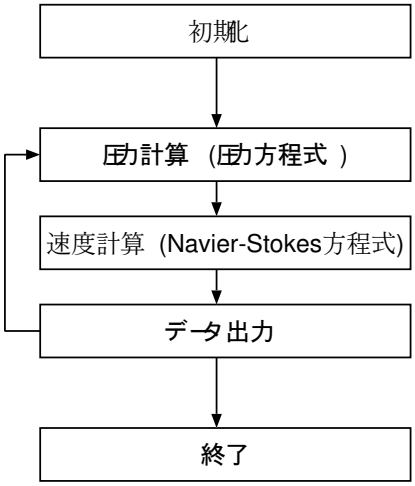

Fig.3：計算の流れ

第 4 章

数値実験

設計したシミュレータを用いて、興味ある現象をシミュレーションしてみる。また、設計したシミュレータの有効性と妥当性も同時に検証する。

4.1 キャビティ問題

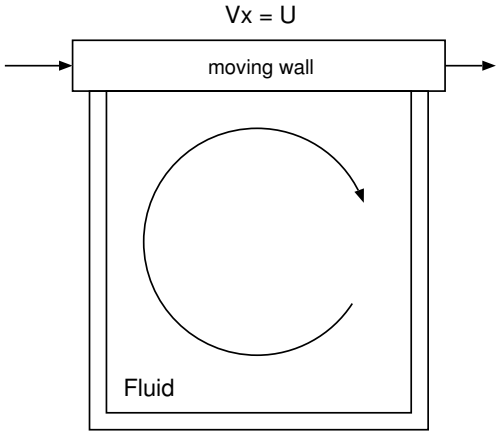

Fig.4：モデル図

Fig.4 のように四方を壁で囲まれているような系を考える。その上辺の壁の 1 つが一定速度で動くような場合、内部の流体がどのような振る舞いをするのかをシミュレーションする。これはキャビティ問題 [4] として知られており、実験とシミュレーションを含め数多くの事例が報告されている。また、キャビティ内部に生じる流れは、層流であると考えられている。そのため乱流モデルを含まないシミュレータで、シミュレーションを行えることが期待できるわけである。

4.2 シミュレーションパラメータ

次にシミュレーションパラメータを示す。境界条件は、自由すべり (free-slip) 条件を全方位に採用する。ただし、上辺の壁は x 方向に一定速度で動いているものとする。

Tab.1：シミュレーションパラメータ

キャビティ辺長	0.25 (m)
上辺壁の速度	0.404×10^{-3} (m/s)
動粘性率	1.0×10^{-3}
時間刻み幅	1.0×10^{-4} (s)
セル数 $(x \times y)$	25×25
レイノルズ数	100

4.3　シミュレーション結果-1

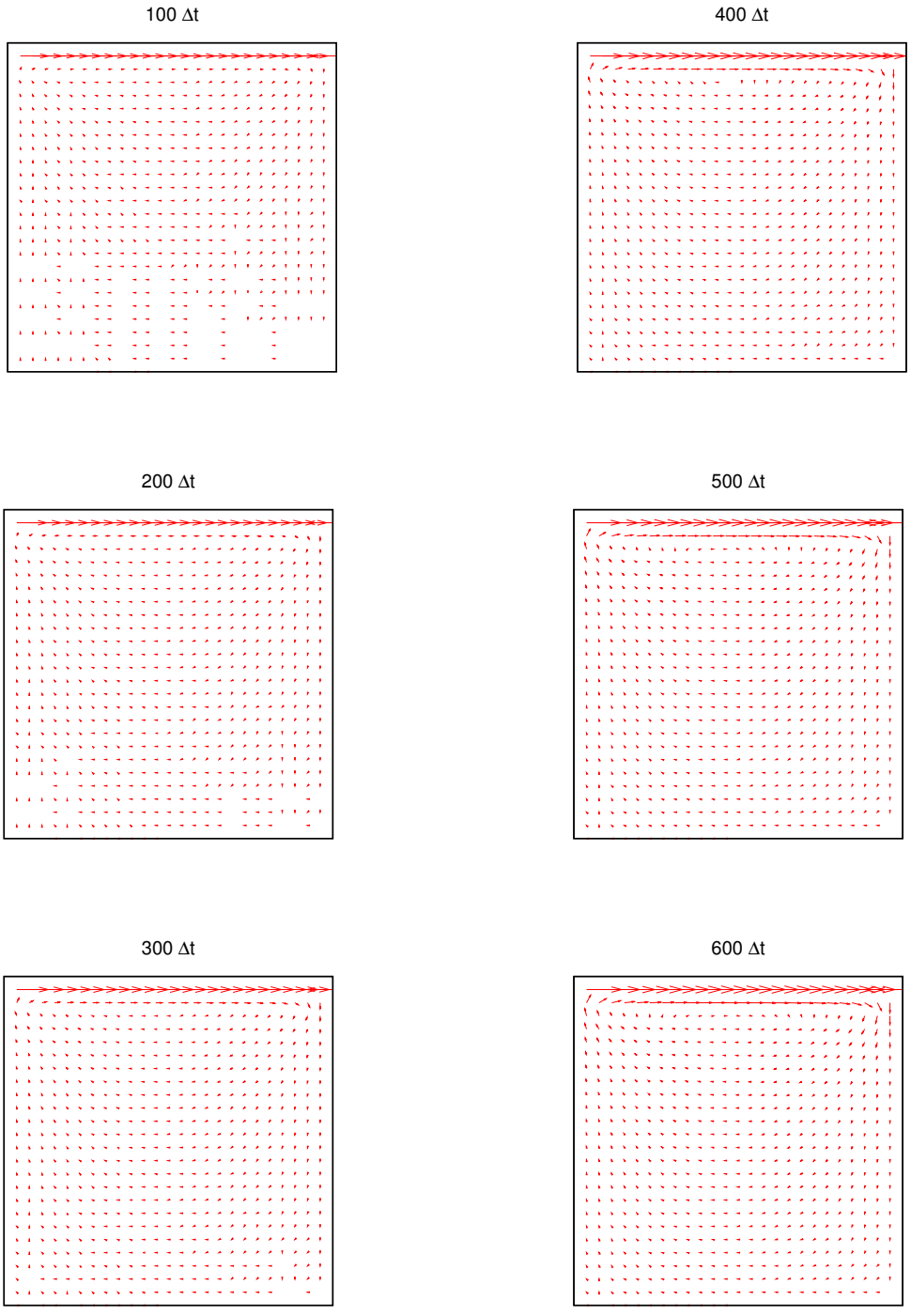

Fig.5：各時間における速度ベクトル場

4.4　シミュレーション結果-2

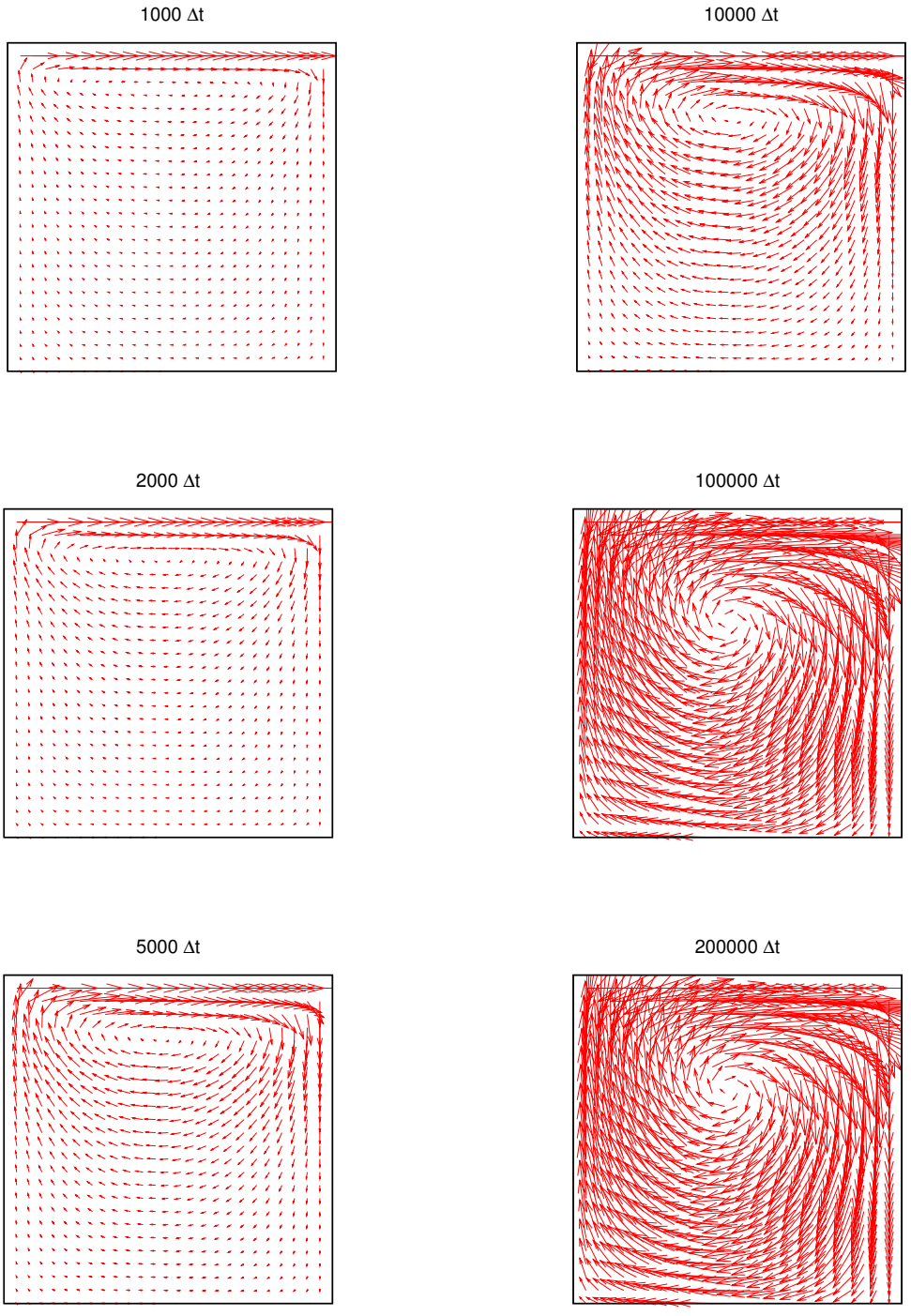

Fig.6：各時間における速度ベクトル場

4.5 考察・検証

各時間ごとに見ていくと、最初は穏やかだったキャビティ内が上辺壁の運動により、徐々に撹拌されていく様子が分かる。$1000\Delta t$ 付近では渦 (循還流) が発生しているのが見て取れる。さらに時間が経過すると、発生した循還流がキャビティ内中央へと移動しているのが分かる ($200000\Delta t$)。これに対応する実験が行われており、似たような挙動を示すことが報告されている [6]。

4.6 拡張

最後に今回設計したシミュレータの拡張を述べて終わる。本稿では流体シミュレーションへの入門ということで、理論的・スキーム的に簡単なものだけを導入し、数値解析全体の流れを把握できるよう配慮してきた。今までの議論を踏まえれば、これまでは複雑に思えたスキームや概念も比較的容易に学習することが期待できる。以下に改善点を挙げる。

- 中心差分法の代わりに風上差分法を導入する。
- ガウス・ザイデル法の代わりに SOR 法を導入する。
- 自由すべり条件の代わりに固着条件を採用する。
- MAC 法から SMAC 法への拡張
- その他のスキームへの拡張・導入
- 2 次元から 3 次元への次元拡張

等が挙げられる。今回、差分法は中心差分法を用いたが、中心差分法では物理的に理解しがたい数値振動が起こることが知られているため、流体シミュレーションでは風上差分法を用いることが一般的となっている。ガウス・ザイデル法は圧力方程式を解くときに使用したスキームであるが、一般に収束速度が遅い。ザイデル法を改良し、収束速度を早めたものが SOR 法である。簡単のために、境界条件では自由すべり条件を採用したが、固着条件の方がより現実に近い。固着条件を採用することで、さらに事実に促したシミュレーションを期待することができる。

参考文献

[1] F.H.Harlow and J.E.Welch,Phys.Fluids,**8** (1965),p.2182.

[2] F.H.Harlow and J.E.Welch,Phys.Fluids,**9** (1966),p.842.

[3] P.J.Roache,Computational Fluid Dynamics,Hermosa Publishers Inc.

[4] http://dns.uchiyama.be/dns6.htm

[5] http://www.geocities.jp/team_zero_three/LaplaceEq/

[6] Analytical and numerical studies of the structure of steady separated flows, J. Fluid Mech., Vol.24, pp.113-151, 1966.

プログラムコード

コンパイル・実行方法

付録として今回設計・使用したプログラムコードを載せる。言語は C(ANSI 規格遵守) で書かれている。解説は UNIX/Linux 環境を前提に書かれているが、Windows や MacOS でも ANSI 規格の C コンパイラがあれば動作する。

コンパイルには以下のコマンドを入力する。

```
gcc -o fluid fluid.c -lm -O2 -Wall
```

実行は

```
time ./fluid
```

#define で定義されているパラメータを変更することで任意のシミュレーションを行うことができるが、設定したパラメータによっては、圧力解法の計算で値が収束せず、計算が止まってしまうことがある。その場合は収束条件を緩めるか、他のパラメータを変更すると改善する場合がある。

出力ファイル

圧力と速度のデータファイルが設定した各タイムステップごとに出力される。速度の出力ファイル名が「vxy*****.dat」、圧力が「pwr*****.dat」となっている。*****には出力時のタイムステップ数が出力される。出力されたデータファイルを可視化ソフトで処理すれば、本稿のような図を得ることができる。

ファイルの出力間隔を変えるためには、プログラムコードの

```
#define STEP 100
```

の値を変えればよい。設定した値の間隔でデータファイルが出力されるようになる。