2 次元流体計算コード fluid2d

峰田竜二
contact@solarphys.com

2022 年 11 月 24 日

このコードは、2 次元の一般座標・構造格子において、圧縮性粘性流体 (ナビエ-ストークス方程式) の計算を行う。格子生成や初期条件の入力、出力結果の解析・可視化には Python、メインの数値計算には Fortran を用いる。

目次

1 動かし方

1.1 Python による格子生成

モジュール fluid.py 内にクラス Fluid2d が定義されている。例えば

```
import fluid
a = fluid.Fluid2d(Ni, Nj, Nb=2)
```

によって、オブジェクトを生成する。Ni, Nj はそれぞれ i,j 方向のグリッド総数。デフォルトでは端から Nb=2 個のグリッドは境界条件用の仮想グリッドであり、内部の (Ni-2*Nb)*(Nj-2*Nb) 個のグリッドについて方程式が評価される。

更に、

```
a.initiate_idealgas_uniform(alpha, Re, Pr, gamma=1.4)
```

によって、理想気体の一様流に各基本変数が初期化される。各無次元数の定義は次の通り (詳しくは後述)。

$$\alpha = \frac{p_\infty}{\rho_\infty U_\infty^2}, \quad Re = \frac{\rho_\infty L U_\infty}{\mu}, \quad Pr = \frac{1}{\gamma - 1}\frac{k_B}{m}\frac{\mu}{\kappa} \tag{1}$$

```
a.Cylinder_analytic(dx, coef)
```

によって、円柱周りの流れの計算のための 2 次元構造格子が生成される。dx は円柱周りの最小グリッド幅、coef は境界層外のグリッド幅をどれだけ荒くするかの係数 (詳細は後述)。

データを出力したいディレクトリ (dir_o) に data という名のディレクトリを生成した後、次のコマンドを打ち込む。

```
a.output_coordinate(dir_o + "coordinate.dat")
a.output_basic(dir_o + "data/b0000000.dat")
```

“coordinate.dat” には格子の座標、“data/b0000000.dat” には初期条件が、後の Fortran プログラムに適する書式で出力される。

1.2 Fortran メインプログラム

メインプログラムは “main.f90” に記述されている。他必要なモジュールが記述された複数の .f90 ファイルが存在する。“parameters_io.f90” を開き、各パラメータを望む値に書き換える。

- `Ni, Nj`: それぞれ i, j 方向のグリッド数。
- `Nb`: 端の境界条件用仮想グリッドの数。内部の `(Ni-2*Nb)*(Nj-2*Nb)` 個のグリッドについて、方程式が評価される。
- `Re, Pr`: レイノルズ数、プラントル数。
- `dir`: データインプット/アウトプット用のディレクトリ。内部にはデータ出力用のディレクトリ data が作られていなければならない。また、格子の座標 “coordinate.dat” と初期条件 “data/b0000000.dat” ファイルも必要。
- `Tmax`: 時間積分する時間 (無次元)。
- `Nout`: データを出力する回数。

ターミナルに

```
> gfortran main.f90
```

と入力し、コンパイルする。`-Ofast` オプションで最適化も可能。

```
> ./a.out
```

で実行。ファイルが出力されるたびに、次の値がターミナルに返される。

- elapsed time: 実行時間。
- tstep: ファイル出力回数。
- t: 出力時の計算時刻 (無次元)。
- iteration: 前回のファイル出力からの時間積分スキーム反復回数。
- rest time: 残り時間の推定値。

2 Fortran プログラムの構成

各ファイルに記述されたモジュールの機能は次の通り。

- main.f90: メインプログラム。`include` 文で他のファイルからモジュールをインポートしている。

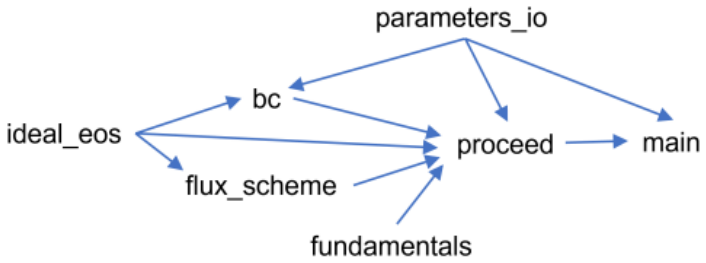

図 1 モジュールの依存関係

- parameters_io.f90/`parameters_io`: 計算のためのパラメータの宣言。ファイル入出力、格子の設定、初期条件の入力のためのサブルーチン。各パラメータはモジュール外部から参照できるが、変更は内部のサブルーチンにしか行えない (protected 属性)。
- proceed.f90/`proceed`: 方程式の右辺を計算し、時間積分するためのサブルーチン。時間積分スキームはここ。
- flux_scheme.f90/`flux_scheme`: 方程式右辺の数値流束を評価するためのサブルーチン。差分化スキームはここ。
- fundamentals.f90/`fundamentals`: 数値計算スキームのための汎用的なサブルーチン。内挿スキームはここ。
- eos.f90/`ideal_eos`: 状態方程式の計算用サブルーチン。
- bc.f90/`bc`: 境界条件の設定用サブルーチン。

モジュールの依存関係は図 1 の通り。

3 Python/Fluid2d クラス内のメソッド

3.1 初期化

3.1.1 `initialte_idealgas_uniform(alpha, Re, Pr, gamma=1.4)`

- `alpha, Re, Pr`: 各無次元数 (定義は後述)。
- `gamma`: 比熱比。デフォは地球大気を想定して 1.4。

理想気体の一様流を想定した初期条件を設定する。

3.2 格子生成

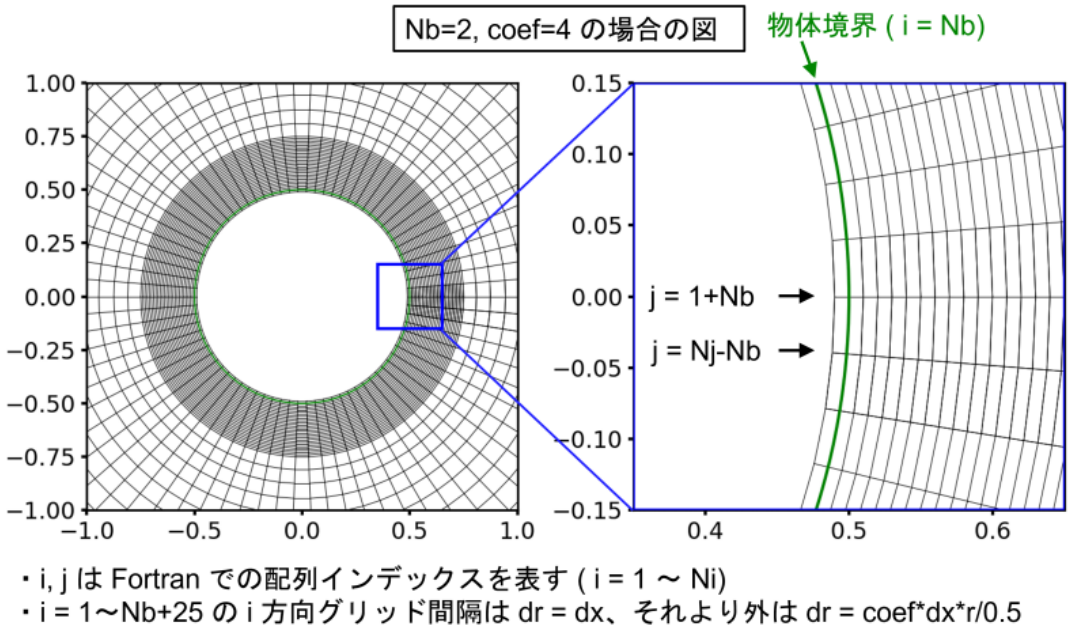

図 2 生成される格子の形状: `Nb=2, coef=4` の場合。

3.2.1 `Cylinder_analytic(dx, coef)`

円柱周りの流れの計算のための O 型構造格子を解析的に生成する (図 2)。物体付近の格子は、境界層を解像するために、r 方向 (i 方向) に細かい。その最小間隔は `dx`。

3.3 クラス変数

生成されたオブジェクトに初期化と格子生成を施すと、次の変数が格納される。

- `Ni, Nj, Nb, alpha, Re, Pr, gamma`: 基本パラメータ。詳細は他の節で。
- `q[Ni,Nj,4]`: 流体の基本変数。`q[:,:,0]` が密度 ρ、`q[:,:,1]` が速度の x 成分 u、`q[:,:,2]` が速度の y 成分 v、`q[:,:,3]` が単位体積あたりの全エネルギー e。
- `x[Ni,Nj,2]`: グリッドの座標。`x[:,:,0]` が x 座標、`x[:,:,2]` が y 座標。

3.4 ファイル入出力

3.4.1 `input_basic(name)`

ファイル `name` に記述された基本変数がクラス変数に読み込まれる。スペースまたは改行区切りで `q[0,0,0], q[1,0,0], q[2,0,0],..., q[Ni-1,0,0], q[0,1,0], q[1,1,0],..., q[Ni-1,Nj-1,3]` の順に読み込まれる。

3.4.2 output_basic(name)

ファイル name に基本変数を出力する。入力と同じ順序でスペース区切りで出力。

3.4.3 output_coordinate(name)

ファイル name にグリッド座標を出力する。x[0,0,0], x[1,0,0],...,x[Ni-1,0,0], x[0,1,0],..., x[Ni-1,Nj-1,1] の順序。

3.4.4 adjust_basic(fname_of_old_coordinate, Ni_old,Nj_old, fname_of_basic)

ファイル fname_of_old_coordinate のグリッド座標において評価された基本変数をファイル fname_of_basic から読み込み、現在クラス変数に格納されている座標で再評価し、その補完された基本変数をクラス変数に格納する。Ni_old, Nj_old は古い座標のグリッド数。

3.5 可視化

3.5.1 plot_whole(file, time, s_vor, s_p, s_B, plot_step)

$x = -2 : 8, y = -2 : 2$ の範囲で、渦度、流速と流線、圧力の無限遠での値からの変化分、ベルヌーイ関数の無限遠での値からの変化分をプロットする。

- file: 出力ファイル名
- time: タイトルに表示される計算時刻
- s_vor, s_p, s_B: それぞれ渦度、圧力、ベルヌーイ関数の描画スケール。渦度以外の各量は無限遠での値で規格化される。
- plot_step: 描画の際に、データを間引く間隔。

3.5.2 plot_detail(file, time, s_vor, s_p)

$x = -0.55 : 0.95, y = -0.75 : 0.75$ の範囲で、渦度、流速と流線、圧力の無限遠での値からの変化分をプロットする。各引数の意味は plot_whole と共通。

3.6 数値計算

3.6.1 proceed(T, pbar, dt_bar)

時間 T だけ時間積分する。時間 dt_tbar ごとに tqdm の進捗バー (pbar) を更新する。Fortran との計算速度比較のために Roe 型 FDS 法で数値積分するメソッドを実装したが、拡散項の与え方がちょっとおかしいので改善の余地あり。

3.7　その他便利コマンド

ffmpeg で連番 png から gif を作る。

```
> ffmpeg -i %07d.png -vf palettegen palette.png
> ffmpeg -f image2 -r 10 -i %07d.png -i palette.png -filter_complex
    paletteuse anim.gif
```

4　メインプログラムの計算内容

メインプログラムは 2 次元圧縮性粘性流体の支配方程式系を一般的な構造格子上で解く。保存形式の方程式が差分法で離散化される。

4.1　無次元化

各量を次のように無次元化する。L は物体の大きさ (円柱の場合は直径)、τ は適当な量 (後から決定する)、他の量のスケールは無限遠での値である。

$$\text{密度：}\quad \rho = \rho_\infty \rho' \tag{2}$$
$$\text{流速：}\quad \boldsymbol{v} = U_\infty \boldsymbol{v}' \tag{3}$$
$$\text{圧力：}\quad p = p_\infty p' \tag{4}$$
$$\text{温度：}\quad T = T_\infty T' \tag{5}$$
$$\text{単位体積あたりの全エネルギー：}\quad e = e_\infty e' \tag{6}$$
$$\text{長さ：}\quad x = L x' \tag{7}$$
$$\text{時間：}\quad t = \tau t' \tag{8}$$

スケールの間に次の関係を課す。

$$\tau = \frac{L}{U_\infty}, \quad e_\infty = p_\infty, \quad \rho_\infty = (\gamma - 1)\frac{m}{k_B}\frac{p_\infty}{T_\infty} \tag{9}$$

γ は比熱比、k_B はボルツマン定数、m は平均分子質量である。すると、ナビエ-ストークス方程式系と理想気体の状態方程式は次のように無次元化される。

$$\text{質量保存則：}\quad \frac{\partial \rho'}{\partial t'} + \nabla' \cdot (\rho' \boldsymbol{v}') = 0 \tag{10}$$

$$\text{運動量保存則：}\quad \frac{\partial}{\partial t'}(\rho' \boldsymbol{v}') + \nabla' \cdot \left[\rho' \boldsymbol{v}' \boldsymbol{v}' + \alpha p' \mathbf{I} - \frac{1}{Re}\boldsymbol{\tau}^{\nu}\right] = 0 \tag{11}$$

$$\text{全エネルギー保存則：}\quad \frac{\partial}{\partial t'}(\alpha e') + \nabla' \cdot \left[(\alpha e' + \alpha p')\boldsymbol{v}' - \frac{1}{Re}\boldsymbol{\tau}^{\nu} \cdot \boldsymbol{v}' - \frac{1}{RePr}\nabla'(\alpha T')\right] = 0 \tag{12}$$

$$\text{状態方程式 1：}\quad \alpha p' = (\gamma - 1)\left(\alpha e' - \frac{1}{2}\rho' {v'}^2\right) \tag{13}$$

$$\text{状態方程式 2：}\quad \alpha T' = \frac{\alpha e'}{\rho'} - \frac{{v'}^2}{2} \tag{14}$$

$$\text{粘性応力：}\quad \tau_{ij}^{\nu} = \frac{\partial v_i'}{\partial x_j'} + \frac{\partial v_j'}{\partial x_i'} - \frac{2}{3}\delta_{ij}\sum_k \frac{\partial v_k'}{\partial x_k'} \tag{15}$$

ここで、各無次元数は次のように定義される。

$$\text{力学 vs 熱力学エネルギー比：}\quad \alpha = \frac{p_\infty}{\rho_\infty U_\infty^2} = \frac{1}{\alpha M a^2} \tag{16}$$

$$\text{レイノルズ数：}\quad Re = \frac{\rho_\infty L U_\infty}{\mu} \tag{17}$$

$$\text{プラントル数：}\quad Pr = \frac{1}{\gamma - 1}\frac{k_B}{m}\frac{\mu}{\kappa} \tag{18}$$

Ma はマッハ数である。μ は粘性率、κ は熱伝導率であり、一様定数としている。プログラム内では、$\alpha p', \alpha e', \alpha T'$ がそれぞれ p, e, T と書かれ、e を基本変数にして解かれる。

プログラム内では次の熱力学的量も用いられる。

$$\text{単位質量あたりの全エンタルピー：}\quad H = \frac{e + p}{\rho} \tag{19}$$

$$\text{音速：}\quad c_s = \sqrt{\frac{\gamma p}{\rho}} \tag{20}$$

4.2 一般座標変換

2 次元の物理空間デカルト座標 x, y を計算空間座標 ξ, η に変換した上で方程式が解かれる。両者の間の座標変換は、離散化したグリッドが計算空間上で等間隔に並ぶように規定される。離散化幅は $d\xi = d\eta = 1$ である。 グリッドの座標一覧が構造的に与えられれば、数値的に各変換係数が計算される。

座標変換のヤコビアン J とその逆 S を次のように定義する。

$$J = \begin{vmatrix} \partial\xi/\partial x & \partial\xi/\partial y \\ \partial\eta/\partial x & \partial\eta/\partial y \end{vmatrix}, \qquad S = \frac{1}{J} \tag{21}$$

グリッドの座標 (x_{ij}, y_{ij}) が与えられると、$\partial x/\partial\xi$ などは数値的に計算できるので、各メトリックスとヤコビアンの逆は次のように計算できる。ただし、$\partial\xi/\partial x = \xi_x$ のように簡略化する。

$$\xi_x = \frac{y_\eta}{S}, \quad \eta_x = -\frac{y_\xi}{S}, \quad \xi_y = -\frac{x_\eta}{S}, \quad \eta_y = \frac{x_\xi}{S} \tag{22}$$

$$S = x_\xi y_\eta - x_\eta y_\xi \tag{23}$$

プログラムでは、各メトリックスは 2 次精度中央差分を用いてグリッド点上 (i, j) で評価される。グリッド境界 $(i + 1/2, j), (i, j + 1/2)$ での値を評価する場合は、隣接する 2 グリッド点上の値の平均をとる。S については、グリッド点によって張られる四角形の面積を計算することにより、グリッド頂点 $(i + 1/2, j + 1/2)$ で評価される。グリッド境界または点上での値を評価する場合は、隣接する 2 個または 4 個の値の平均をとる。具体的には次のように計算される。

$$(\xi_x S)_{ij} = \frac{y_{i,j+1} - y_{i,j-1}}{2} \tag{24}$$

$$(\xi_y S)_{ij} = -\frac{x_{i,j+1} - x_{i,j-1}}{2} \tag{25}$$

$$(\eta_x S)_{ij} = -\frac{y_{i+1,j} - y_{i-1,j}}{2} \tag{26}$$

$$(\eta_y S)_{ij} = \frac{x_{i+1,j} - x_{i-1,j}}{2} \tag{27}$$

例えば、 $$(\xi_x S)_{i+1/2,j} = \frac{(\xi_x S)_{i+1,j} + (\xi_x S)_{i-1,j}}{2} \tag{28}$$

$$S_{i+1/2,j+1/2} = \frac{1}{2}\left[(x_{i+1,j+1} - x_{i,j})(y_{i,j+1} - y_{i+1,j})\right. \tag{29}$$

$$\left. -(y_{i+1,j+1} - y_{i,j})(x_{i,j+1} - x_{i+1,j})\right] \tag{30}$$

$$S_{i+1/2,j} = \frac{S_{i+1/2,j+1/2} + S_{i+1/2,j-1/2}}{2} \tag{31}$$

$$S_{i,j} = \frac{S_{i+1/2,j+1/2} + S_{i+1/2,j-1/2} + S_{i-1/2,j+1/2} + S_{i-1/2,j-1/2}}{4} \tag{32}$$

4.3 解かれるシステム方程式

一般座標変換を施した保存形式のナビエ-ストークス方程式系が解かれる。具体的には次の通り。

$$\frac{\partial \vec{Q}}{\partial t} = -\frac{\partial}{\partial \xi}(\vec{E}_c - \vec{E}_v) - \frac{\partial}{\partial \eta}(\vec{F}_c - \vec{F}_v) \tag{33}$$

$$\vec{Q} = S\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ e\end{pmatrix}, \quad \vec{E}_c = S\begin{pmatrix}\rho U\\ \rho u U + \xi_x p\\ \rho v U + \xi_y p\\ (e+p)U\end{pmatrix}, \quad \vec{F}_c = S\begin{pmatrix}\rho V\\ \rho u V + \eta_x p\\ \rho v V + \eta_y p\\ (e+p)V\end{pmatrix} \tag{34}$$

$$U = \xi_x u + \xi_y v, \quad V = \eta_x u + \eta_y v \tag{35}$$

$$\vec{E}_v = S\begin{pmatrix}0\\ \xi_x\tau_{xx} + \xi_y\tau_{xy}\\ \xi_x\tau_{xy} + \xi_y\tau_{yy}\\ \xi_x\beta_x + \xi_y\beta_y\end{pmatrix}, \quad \vec{F}_v = S\begin{pmatrix}0\\ \eta_x\tau_{xx} + \eta_y\tau_{xy}\\ \eta_x\tau_{xy} + \eta_y\tau_{yy}\\ \eta_x\beta_x + \eta_y\beta_y\end{pmatrix} \tag{36}$$

$$\tau_{xx} = \frac{2}{3Re}(2u_x - v_y), \quad \tau_{yy} = \frac{2}{3Re}(2v_y - u_x), \quad \tau_{xy} = \frac{1}{Re}(u_y + v_x) \tag{37}$$

$$\beta_x = \tau_{xx}u + \tau_{xy}v + \frac{1}{RePr}T_x, \quad \beta_y = \tau_{xy}u + \tau_{yy}v + \frac{1}{RePr}T_y \tag{38}$$

質量密度 ρ、流速の x 成分 u、流速の y 成分 v、単位体積あたりの全エネルギー e を基本変数に取る。基本変数から右辺のフラックスを構成し、時間積分スキームを用いて次時間ステップでの $\vec{Q}$ を求め、それを基本変数に戻す操作をくり返すことで、時間発展問題を解く。このときのフラックスの評価方法が差分化スキームである。

5 時間積分スキーム

時間ステップ n での方程式 (33) の右辺が、差分化スキームによって基本変数の組 q^n から $\mathrm{RHS}(q^n)$ と評価されたとする。ここから次の時間ステップ $(n+1)$ での保存変数 Q^{n+1} を計算するのが時間積分スキームの役割である。時間ステップ幅を dt とする。

プログラムに実装されているスキームを列挙する。スキームはファイル “proceed.f90” にあるモジュール `proceed` 内に記述されている。スキームの変更は、同モジュール内のサブルーチン `proceed_time` を書き換えることで行う。

5.1 SSP ルンゲ-クッタ法 (Shu & Osher, 1988)

大きな CFL でも TVD 条件を保てる陽解法。

$$\text{2 次精度：} \quad Q^{(1)} = Q^n + dt\mathrm{RHS}(q^n) \tag{39}$$

$$Q^{n+1} = \frac{1}{2}Q^n + \frac{1}{2}\left[Q^{(1)} + dt\mathrm{RHS}(q^{(1)})\right] \tag{40}$$

$$\text{3 次精度：} \quad Q^{(1)} = Q^n + dt\mathrm{RHS}(q^n) \tag{41}$$

$$Q^{(2)} = \frac{3}{4}Q^n + \frac{1}{4}\left[Q^{(1)} + dt\mathrm{RHS}(q^{(1)})\right] \tag{42}$$

$$Q^{n+1} = \frac{1}{3}Q^n + \frac{2}{3}\left[Q^{(2)} + dt\mathrm{RHS}(q^{(2)})\right] \tag{43}$$

6 差分化スキーム

6.1 3 次精度 MUSCL-minmod (van Leer, 1977)

6.2 5 次精度 MP (Suresh & Huynh, 1997)

7 内挿スキーム

8 粘性項

あ

参考文献

Shu, C.-W., & Osher, S. (1988), Efficient implementation of essentially non-oscillatory shock-apturing schemes, Journal of Computational Physics, **77**, 439–471. doi: 10.1016/0021-9991(88)90177-5

Suresh, A., & Huynh, H. T. (1997), Accurate monotonicity-preserving schemes with Runge–Kutta time stepping, Journal of Computational Physics, **136**, 83–99. doi: 10.1006/jcph.1997.5745

van Leer, B. (1977), Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection, Journal of Computational Physics, **23**, 276–299. doi: 10.1016/0021-9991(77)90095-X