応用数値解析特論 演習課題 A
〜 非圧縮ポテンシャル流 〜

桂田 祐史

2024 年 7 月 9 日,2024 年 7 月 9 日

目 次

1 課題文

2 次元非圧縮ポテンシャル流の定常流で、流体の占める領域 Ω と、その境界 $\Gamma = \partial\Omega$ での流速の法線成分 $V_n := \boldsymbol{v} \cdot \boldsymbol{n}$ が分かっている場合に、速度ポテンシャル ϕ を計算して、等ポテンシャル線と速度場を可視化せよ。

領域 Ω としては、円盤領域以外のものを選ぶこと (多角形領域は比較的簡単でお勧め。楕円領域を考える人がいると思われるが、楕円に沿っての積分は難しいことが多いのであまり勧めない。)。境界値 (流速の法線成分) V_n も自分で興味のあるもの、自分の都合の良いものを選んで良い (後の注意を読んでおくこと)。

以下でサンプル・プログラムを紹介する。それを理解した上で、領域 Ω と境界データ V_n を自分が決めたものに書き換えれば良い。自由度は高いので、工夫・遊び心発揮を期待する。

出来る限り、流線も可視化せよ。流線の書き方には色々なやり方があるが、流れ関数を求めるやり方を勧めておく (それを求めるにもポテンシャル問題が使えたりするので)。

2 参考情報など

- FreeFem++ の使い方については、
  - `FreeFEM-documentation.pdf` (公式ドキュメント)
    https://doc.freefem.org/pdf/FreeFEM-documentation.pdf
    (本当は Mac の中にあるだけれど、FreeFem++ のバージョンによって場所が違うので…[1])
  - 「FreeFem++ の紹介」桂田が書いた紹介文書
    https://m-katsurada.sakura.ne.jp/labo/text/welcome-to-freefem/
  - 「FreeFem++ノート」桂田が書いた文法メモ
    https://m-katsurada.sakura.ne.jp/labo/text/freefem-note/
- サンプル・プログラムを用意してある。入手法をまとめて示しておく。

ターミナルで実行する

```
curl -O https://m-katsurada.sakura.ne.jp/program/fem/poisson-kikuchi.edp
curl -O https://m-katsurada.sakura.ne.jp/ana/potential2d-v-2024.edp
curl -O https://m-katsurada.sakura.ne.jp/ana/sample0.edp
curl -O https://m-katsurada.sakura.ne.jp/ana/sample1.edp
curl -O https://m-katsurada.sakura.ne.jp/ana/sample2.edp
```

`poisson-kikuchi.edp` は、正方形領域における Poisson 方程式の境界値問題を解くプログラムで、すでに紹介済み。これについては、「FreeFem++ で菊地 [1] の Poisson 方程式の例題を解く」を見るとよい。多角形領域の扱い方が分かるはず。

それ以外の `potential2d-v-2024.edp`, `sample0.edp`, `sample1.edp`, `sample2.edp` は、円盤領域で

$$\triangle\phi = 0 \quad (\text{in } \Omega) \tag{1a}$$

$$\frac{\partial\phi}{\partial\boldsymbol{n}} = V_n \quad (\text{on } \partial\Omega) \tag{1b}$$

を解くプログラムである。中身は境界条件の設定の仕方が違うだけで他はほとんど同じ。

弱形式は、すでに説明したように

$$\iint_\Omega (\phi_x v_x + \phi_y v_y)\,dx\,dy = \int_{\partial\Omega} V_n v\,d\sigma \quad (\text{任意の試験関数 } v \text{ に対して}).$$

この左辺は FreeFem++ のプログラム中では、`int2d(Th)(dx(phi)*dx(v)+dy(phi)*dy(v))` のようにすれば良い。右辺は少し工夫が必要である (FreeFem++ の機能が今ひとつなため)。

3 注意 (必ず読むこと)

次が非常に重要である。

V_n は $\displaystyle\int_{\partial\Omega} V_n\,d\sigma = 0$ を満たしている必要がある。実際、$\frac{\partial\phi}{\partial n}$ の定義と Gauss の発散定理から

$$\int_{\partial\Omega} V_n\,d\sigma = \int_{\partial\Omega} \frac{\partial\phi}{\partial n}\,d\sigma = \int_{\partial\Omega} \operatorname{grad}\phi\cdot\boldsymbol{n}\,d\sigma = \int_\Omega \operatorname{div}\operatorname{grad}\phi\,d\boldsymbol{x} = \int_\Omega \triangle\phi\,d\boldsymbol{x} = \int_\Omega 0\,d\boldsymbol{x} = 0.$$

[1] 一応 `find /Applications/FreeFem++.app -name FreeFEM-documentation.pdf -print` で表示されますが…

非圧縮流 ($\operatorname{div}\boldsymbol{v}=0$) の速度場 $\boldsymbol{v}$ が分かっている場合に、$V_n=\boldsymbol{v}\cdot\boldsymbol{n}$ と置いた場合は、この条件は必ず成り立っている ($\displaystyle\int_{\partial\Omega} V_n\,d\sigma=\int_{\partial\Omega}\boldsymbol{v}\cdot\boldsymbol{n}\,d\sigma=\int_{\Omega}\operatorname{div}\boldsymbol{v}\,d\boldsymbol{x}=\int_{\Omega}0\,d\boldsymbol{x}=0$)。

そうでない場合には、$\displaystyle\int_{\partial\Omega} V_n\,d\sigma=0$ が成り立つように、注意深く V_n を選ぶ必要がある。

4 FAQ (よくされる質問)

「境界値 V_n を場合分けを含む関数としたいが、FreeFem++ がプログラムを受け付けてくれません。どうすればいいですか？」

(FreeFem++ ってしょうがないですね…) 例えば以下の 2 つの解決策があります。

(a) 弱形式には複数の `int1d()` が指定できるので、境界を分割して、その部分ごとに (場合分けを含まない) 境界値を与えるようにプログラムを書く。

(b) 例えば `(x>1 && y>0)` のような式は、条件が成り立つならば 1, 成り立たないならば 0 という値を持つので、`if` を使わずに、式だけで場合分けを含む関数が記述できる。

次のサンプル・プログラムでは、(b) を用いているが、ある程度以上複雑になった場合は、私のおすすめは (a) である。付録の `sample1.edp` を見て下さい。

5 サンプルプログラム potential2d-v-2024.edp 解説

potential2d-v-2024.edp

```
// potential2d-v-2024.edp
//   curl -O https://m-katsurada.sakura.ne.jp/complex2/potential2d-v-2024.edp
//   2 次元非圧縮ポテンシャル流
//      速度ポテンシャル，速度を求め、等ポテンシャル線，速度場を描く

border Gamma(t=0,2*pi) { x = cos(t); y = sin(t); } // 円盤領域
int m=40;
mesh Th=buildmesh(Gamma(m));
plot(Th, wait=1, ps="Th.eps");
// 次の 2 行は区分 1 次多項式を使うという意味
fespace Vh(Th,P1);
Vh phi, v;
// 境界条件の設定 右上と左下のみ Vn=x+2y (これは確かに ∫ Vn d σ=0 を満たす)
func Vn=((x>0&&y>0) || (x<0&&y<0))*(x+2*y);

// 速度ポテンシャルφを求める
solve Laplace(phi,v) =
  int2d(Th)(dx(phi)*dx(v)+dy(phi)*dy(v)) -int1d(Th,Gamma)(Vn*v);

// 平均を 0 にする (解の一意性がないため、時々生じる大きなズレを消す
real meanphi = int2d(Th)(phi)/Th.area; // Th.area は int2d(Th)(1); でも計算可能
phi = phi - meanphi;

// φの等高線 (等ポテンシャル線) を描く
plot(phi,ps="contourpotential.eps",wait=1);

// ベクトル場 (v1,v2)=∇φ を描く (ちょっと雑なやり方)
Vh v1, v2;
v1=dx(phi); v2=dy(phi);
plot([v1,v2],ps="vectorfield.eps",wait=1);

// 等ポテンシャル線とベクトル場を同時に描く
plot([v1,v2],phi,ps="both.eps", wait=1);
```

- 6 行目で領域 Ω の境界 $\Gamma := \partial\Omega$ を指定している (それで Ω が決まる)。

- 8 行目で、Γ を m 分割して、Γ の囲む範囲を三角形分割して、それを `mesh` 型の変数 `Th` に代入している。

- 11 行目、有限要素空間 V_h を区分的 1 次関数の空間と定義している。この辺についてはこの講義では説明を省略する (知りたい人は有限要素法のテキストを読んで下さい)。

- 12 行目、ϕ と試験関数 v を `Vh` の要素とする。

- 14 行目、境界値 V_n の設定をしている。ここでは

$$V_n(x,y) = \begin{cases} x + 2y & ((x > 0 \text{ かつ } y > 0) \text{ または } (x < 0 \text{ かつ } y < 0) \text{ のとき}) \\ 0 & (\text{それ以外}) \end{cases}$$

としている。(後の例 `sample0.edp` では $V_n(x,y) = x + 2y = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \boldsymbol{n}$ としていて、これは $\boldsymbol{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ という一様流となる。) これは確かに $\displaystyle\int_{\partial\Omega} V_n \, d\sigma = 0$ を満たしている。

- 17〜18 行目、弱形式の定義。

- 21〜22 行目、Neumann 境界値問題は定数だけの不定性を持つため (ϕ が解ならば $\phi+$定数も解)、時々大きなズレが生じ、解の描画などで問題が生じる。平均値 $\dfrac{\displaystyle\iint_\Omega \phi(x,y) dx\, dy}{\displaystyle\iint_\Omega dx\, dy}$ を引くことで、平均 $= 0$ にしておく。

6 流線を描くために: 流れ関数 ψ を求める

流線があると流れの様子がよく分かるので、流線を描く価値は高い。

2 次元の場合は、流れ関数の等高線として流線がかける。流れ関数 ψ は、線積分で表すことができる:

$$\psi(\boldsymbol{x}) = \int_{C_{\boldsymbol{x}}} \psi_x \, dx + \psi_y \, dy = \int_{C_{\boldsymbol{x}}} \begin{pmatrix} -v \\ u \end{pmatrix} \cdot d\boldsymbol{r} = \int_{C_{\boldsymbol{x}}} \begin{pmatrix} -v \\ u \end{pmatrix} \cdot \boldsymbol{t} \, ds \quad (\boldsymbol{x} \in \overline{\Omega})$$

ここで $C_{\boldsymbol{x}}$ は定点 $\boldsymbol{a}$ と $\boldsymbol{x}$ を結ぶ曲線である。

領域が Jordan 領域 (1 つの単純閉曲線で囲まれた領域) であれば、$\boldsymbol{x}$ が境界上の点である場合、定点 $\boldsymbol{a}$ と $C_{\boldsymbol{x}}$ を境界上に選ぶことで、($\begin{pmatrix} -v \\ u \end{pmatrix} \cdot \boldsymbol{t} = \boldsymbol{v} \cdot \boldsymbol{n}$. また境界上での $\boldsymbol{v} \cdot \boldsymbol{n} = V_n$ は知っていると仮定しているので)

$$\psi(\boldsymbol{x}) = \int_{C_{\boldsymbol{x}}} \boldsymbol{v} \cdot \boldsymbol{n} \, ds = \int_{C_{\boldsymbol{x}}} V_n \, ds \quad (\boldsymbol{x} \in \partial\Omega).$$

これから、境界上の点 $\boldsymbol{x}$ における ψ の値 $g(\boldsymbol{x}) := \psi(\boldsymbol{x})$ $(x \in \partial\Omega)$ が得られる。それを用いて

$$\triangle\psi = 0 \quad (\text{in } \Omega), \quad \psi(\boldsymbol{x}) = g(\boldsymbol{x}) \quad (\text{on } \partial\Omega)$$

という Laplace 方程式の Dirichlet 境界値問題を解くことで流れ関数 ψ が得られる。

Dirichlet 境界条件のある場合のプログラムは、上記の `poisson-kikuchi.edp` が参考になる。

A　サンプル・プログラム解説

以下すべて単位円版領域において

$$\triangle\phi = 0 \quad \text{in } \Omega, \qquad \frac{\partial\phi}{\partial n} = V_n \quad \text{on } \partial\Omega$$

を解くプログラムである。境界条件の指定の部分が違っている。

A.1　一様流のプログラム sample0.edp

サンプルプログラム sample0.edp では、単位円盤領域 $\Omega = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 < 1\}$ における一様流 $\boldsymbol{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ の場合のプログラムである。単位円盤なので $\boldsymbol{n} = \begin{pmatrix} x \\ y \end{pmatrix}$ (ここは良く考えること。例えば楕円の場合は $\boldsymbol{n}$ はかなり複雑な式になる。). これから

$$V_n = \boldsymbol{v} \cdot \boldsymbol{n} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = x + 2y.$$

プログラムでは、`func Vn=x+2*y;` で `Vn` を定義して、弱形式中で `int1d(Th,Gamma)(Vn*v)` と使っている。

($\boldsymbol{v}$ が Ω で定数関数なので) $\operatorname{div}\boldsymbol{v} = 0$ であるから、当然 $\displaystyle\int_\Gamma V_n\, d\sigma = 0$ も成り立つ。

sample0.edp

```
// sample0.edp
//   https://m-katsurada.sakura.ne.jp/complex2/sample0.edp
//   ２次元非圧縮ポテンシャル流
//     速度ポテンシャル，速度を求め、等ポテンシャル線，速度場を描く

border Gamma(t=0,2*pi) { x = cos(t); y = sin(t); } // 円盤領域
int m=60;
mesh Th=buildmesh(Gamma(m));
plot(Th, wait=1, ps="Th.eps");
// 次の２行は区分１次多項式を使うという意味
fespace Vh(Th,P1);
Vh phi, v, v1, v2;
// 境界条件の設定
func Vn=x+2*y;// Ωが単位円で，V=(1,2) のとき V・n=x+2y

// 速度ポテンシャルφを求め、その等高線（等ポテンシャル線）を描く
solve Laplace(phi,v) =
  int2d(Th)(dx(phi)*dx(v)+dy(phi)*dy(v)) -int1d(Th,Gamma)(Vn*v);

real meanphi=int2d(Th)(phi)/Th.area;
phi=phi-meanphi;
plot(phi,ps="contourpotential.eps",wait=1);

// ベクトル場 (v1,v2)=∇φ を描く（ちょっと雑なやり方）
v1=dx(phi); v2=dy(phi);
plot([v1,v2],ps="vectorfield.eps",wait=1);

// 等ポテンシャル線とベクトル場を同時に描く
plot([v1,v2],phi,ps="both.eps", wait=1);
```

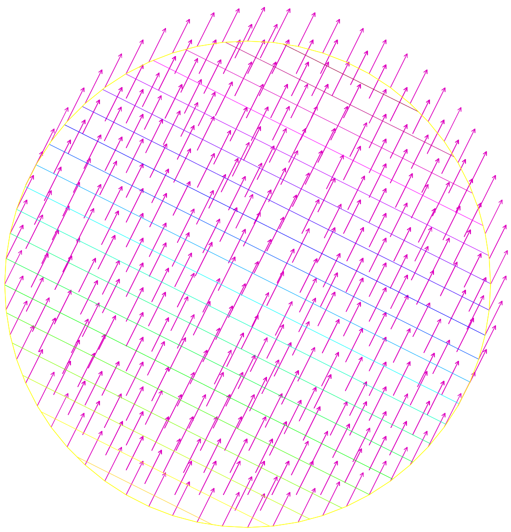

図 1: 一様流 (sample0.edp)

A.2 sample1.edp, sample2.edp

単位円盤領域の境界である単位円周上で、$0 \le \theta \le \pi/6$, $\pi/6 \le \theta \le \pi$, $\pi \le \theta \le 4\pi/3$ $4\pi/3 \le \theta \le 2\pi$ の範囲をそれぞれ $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ とする。すなわち

$$\begin{aligned}
\Gamma_1 &= \{(\cos\theta, \sin\theta) \mid 0 \le \theta \le \pi/6\}, \\
\Gamma_2 &= \{(\cos\theta, \sin\theta) \mid \pi/6 \le \theta \le \pi\}, \\
\Gamma_3 &= \{(\cos\theta, \sin\theta) \mid \pi \le \theta \le 4\pi/3\}, \\
\Gamma_4 &= \{(\cos\theta, \sin\theta) \mid 4\pi/3 \le \theta \le 2\pi\}.
\end{aligned}$$

境界条件 $\frac{\partial\phi}{\partial n} = V_n$ における V_n は、次のようになっているとする。

$$V_n = \begin{cases} -2 & (\text{on } \Gamma_1) \\ 0 & (\text{on } \Gamma_2) \\ 1 & (\text{on } \Gamma_3) \\ 0 & (\text{on } \Gamma_4). \end{cases}$$

念のため $\displaystyle\int_{\partial\Omega} V_n\, d\sigma = 0$ の確認

$\displaystyle\int_\gamma d\sigma = \gamma$ の長さ に注意すると

$$\begin{aligned}
\int_{\partial\Omega} V_n\, d\sigma &= \int_{\Gamma_1} V_n\, d\sigma + \int_{\Gamma_2} V_n\, d\sigma + \int_{\Gamma_3} V_n\, d\sigma + \int_{\Gamma_4} V_n\, d\sigma \\
&= -2\int_{\Gamma_1} d\sigma + 0 + 1\cdot\int_{\Gamma_3} d\sigma + 0 = -2\cdot\frac{\pi}{6} + 1\cdot\frac{\pi}{3} = 0.
\end{aligned}$$

sample1.edp, sample2.edp はともにこの問題を解くプログラムである。

もしも、プログラム中で V_n がうまく定義できれば、sample0.edp と同様に

```
solve Laplace(phi,v) =
   int2d(Th)(dx(phi)*dx(v)+dy(phi)*dy(v)) -int1d(Th,Gamma)(Vn*v);
```

というコードが使える。

しかし V_n を1行の関数として定義するのはあまり簡単でない。一応出来なくはなくて、それは後で紹介するが (sample2.edp)、ここでは次のようにすることを勧める。

要点は

$$\int_{\partial\Omega} V_n v\, d\sigma = \sum_{j=1}^{4} \int_{\Gamma_j} V_n v\, d\sigma = \int_{\Gamma_1} V_n v\, d\sigma + \int_{\Gamma_3} V_n v\, d\sigma$$

と分解することである (Γ_2, Γ_4 では $V_n = 0$ であるから、$\displaystyle\int_{\Gamma_j} V_n v\, d\sigma$ $(j = 2, 4)$ は必要ない)。

Γ_1, Γ_3 における V_n を、それぞれ Vn1, Vn3 と定義すれば

```
solve Laplace(phi,v) =
   int2d(Th)(dx(phi)*dx(v)+dy(phi)*dy(v))
  -int1d(Th,Gamma1)(Vn1*v)-int1d(Th,Gamma3)(Vn3*v);
```

とすれば良い 。このためには、もちろんプログラム中で $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ を用意する必要があるが、それは難しくない (境界を4つに分けた正方形領域のプログラム poisson-kikuchi.edp と同様のことをすれば良い)。以上の考察から、次のプログラムが得られる。

sample1.edp

```
// sample1.edp
//   https://m-katsurada.sakura.ne.jp/complex2/sample1.edp
//   2次元非圧縮ポテンシャル流
//      速度ポテンシャル, 速度を求め、等ポテンシャル線, 速度場を描く

border Gamma1(t=0,pi/6)      { x = cos(t); y = sin(t); }
border Gamma2(t=pi/6,pi)     { x = cos(t); y = sin(t); }
border Gamma3(t=pi,4*pi/3)   { x = cos(t); y = sin(t); }
border Gamma4(t=4*pi/3,2*pi) { x = cos(t); y = sin(t); }
int m=5;
mesh Th=buildmesh(Gamma1(m)+Gamma2(5*m)+Gamma3(2*m)+Gamma4(4*m));
plot(Th, wait=1, ps="Th.eps");
// 次の2行は区分1次多項式を使うという意味
fespace Vh(Th,P1);
Vh phi, v;

// Gamma1, Gamma3 上での Vn
func Vn1 = -2.0;
func Vn3 = 1.0;

// 速度ポテンシャルφを求め、その等高線（等ポテンシャル線）を描く
solve Laplace(phi,v) =
  int2d(Th)(dx(phi)*dx(v)+dy(phi)*dy(v))
  -int1d(Th,Gamma1)(Vn1*v)-int1d(Th,Gamma3)(Vn3*v);

real mean=int2d(Th)(phi)/Th.area; // Th.area は int2d(Th)(1.0)
phi=phi-mean;
plot(phi,ps="contourpotential.eps",wait=1);

// ベクトル場 (v1,v2)=∇φ を描く（ちょっと雑なやり方）
Vh v1, v2;
v1=dx(phi); v2=dy(phi);
plot([v1,v2],ps="vectorfield.eps",wait=1);

// 等ポテンシャル線とベクトル場を同時に描く
plot([v1,v2],phi,ps="both.eps", wait=1);
```

以下、参考までに、V_n を1命令(長いので2行になっている)で書いたプログラム sample2.edp を紹介しておく。いわゆる偏角の主値 (値が $(-\pi, \pi]$ に属する) を返す関数 atan2(y,x) を使

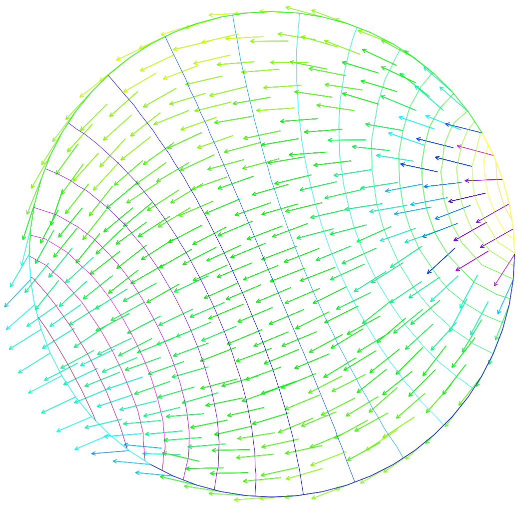

図 2: sample1.edp の結果 ($\theta \in [0, \pi/6]$ で勢い良く入り、$\theta \in [\pi, 4\pi/3]$ でゆっくり出て行く)

うが、やや不自然な感じがするのは否めない (角度の範囲が $[0, 2\pi)$ でないことにも注意が必要である)。

強引に 1 行関数を作る

```
func Vn=((atan2(y,x)>=0.0&&atan2(y,x)<=pi/6) * (-2.0)
        + (atan2(y,x)>=-pi&&atan2(y,x)<=-2.0*pi/3.0) * (1.0));
```

sample2.edp

```
// sample2.edp
//   https://m-katsurada.sakura.ne.jp/complex2/sample2.edp
//   2 次元非圧縮ポテンシャル流
//     速度ポテンシャル，速度を求め、等ポテンシャル線，速度場を描く

border Gamma(t=0,2*pi) { x = cos(t); y = sin(t); } // 円盤領域
int m=60;
mesh Th=buildmesh(Gamma(m));
plot(Th, wait=1, ps="Th.eps");
// 次の 2 行は区分 1 次多項式を使うという意味
fespace Vh(Th,P1);
Vh phi, v, v1, v2;
// 境界条件の設定 (atan2(y,x) は [-pi,pi) の範囲の値を返す)
func Vn=((atan2(y,x)>=0.0&&atan2(y,x)<=pi/6) * (-2.0)
       + (atan2(y,x)>=-pi&&atan2(y,x)<=-2.0*pi/3.0) * (1.0));

// 速度ポテンシャルφを求め、その等高線 (等ポテンシャル線) を描く
solve Laplace(phi,v) =
  int2d(Th)(dx(phi)*dx(v)+dy(phi)*dy(v)) -int1d(Th,Gamma)(Vn*v);
real meanphi=int2d(Th)(phi)/Th.area;
phi=phi-meanphi;
plot(phi,ps="contourpotential.eps",wait=1);

// ベクトル場 (v1,v2)=∇φ を描く (ちょっと雑なやり方)
v1=dx(phi); v2=dy(phi);
plot([v1,v2],ps="vectorfield.eps",wait=1);

// 等ポテンシャル線とベクトル場を同時に描く
plot([v1,v2],phi,ps="both.eps", wait=1);
```

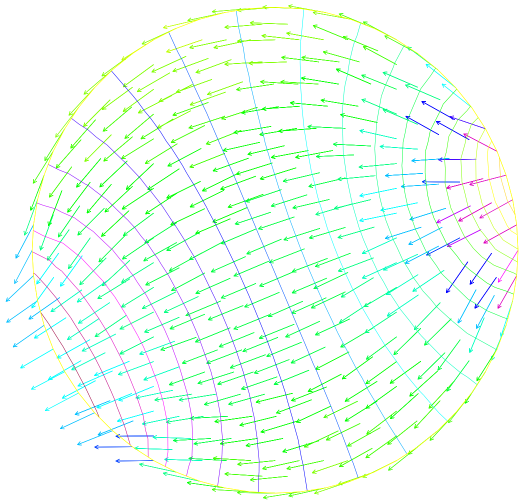

図 3: `sample2.edp` の結果 ($\theta \in [0, \pi/6]$ で勢い良く入り、$\theta \in [\pi, 4\pi/3]$ でゆっくり出て行く)