 Copyright© The Open CAE Society of Japan

# OpenFOAM の Arrhenius-Newtonian 温度依存粘度モデルにおける動粘度更新不具合の原因特定と修正提案

中川　慎二 [1†]

[1] 富山県立大学

# Problem Identification and Correction in OpenFOAM's Arrhenius-Newtonian Temperature-Dependent Viscosity Model

Shinji NAKAGAWA[*†]

[*]Toyama Prefectural University

Abstract

Combining the Arrhenius class, which adds temperature dependence to the Newtonian viscosity model in the OpenCFD version of OpenFOAM, fails to update kinematic viscosity correctly due to excessive temperature effect. The cause is identified as the function “correct” in the Newtonian class, and a method for its correction is proposed. Basic test cases demonstrate that the proposed correction yields correct behavior. The suggested fix has been reported upstream as a bug.

*Keywords*: OpenFOAM, OpenCFD, viscosity model, temperature dependence, Arrhenius, Newtonian

## 1. 緒言

OpenFOAM のニュートン粘度モデルおよび非ニュートン粘度モデルに温度依存性を付与する `Arrhenius` クラスについて検証する．OpenCFD 版 OpenFOAM v2512 までにおいて，`Arrhenius` クラスと `Newtonian` 粘度モデルを組み合わせた場合，温度変化が粘度へ正しく反映されない問題があることを確認した．本研究では，この不具合を明確に再現できる計算例を作成し，ソースコード上の原因を示す．

さらに，この問題を修正するコードを作成し，その有効性を検証した．この修正は OpenFOAM（OpenCFD 社版）のバグ報告に登録[1]した．

まず，問題の生じない非ニュートン粘度モデルとの組合せにより，`Arrhenius` クラスが期待通りに動作することを示す．次に，`Newtonian` 粘度モデルとの組合せにおいて不具合が発生することを示す．最後に，`Newtonian` 粘度モデルを修正することで，正しい温度依存性が得られることを確認する．

温度場の計算を伴う定常および非定常ソルバとして `buoyantBoussinesqSimpleFoam` および `buoyantBoussinesqPimpleFoam` を用いる．単純な定常非等温流れ場として非等温平行平板間クエット流れ，非定常 3 次元問題として室内自然対流のテストケースを作成した．

## 2. 計算方法

OpenCFD 社版 OpenFOAM v2506 を使用した．本報で報告する問題と解決策は，OpenFOAM v1712 から v2512 に共通するものである．

### 2.1. `Arrhenius` 粘度モデル

`Arrhenius` 型粘度モデルは，2017 年 10 月に OpenFOAM に追加された[2]モデルであり，エネルギー輸送方程式を解くソルバと組み合わせることで，粘度の温度依存性を表現できる．ただし，本論文で扱うモデルは，一般的な反応速度論で用いられる Arrhenius 式そのものではなく，基準温度まわりで温度依存性を簡便に表現する近似的なモデルである．本モデルでは，温度の上昇に伴って減少する補正係数を定義し，それを既存の粘度モデルに乗じることで温度依存性を与える．本モデルは標準的には温度依存性を扱うが，必ずしも温度に限定されるものではない．スカラー場が与えられれば，その場に対する依存性として利用すること

† *E-mail address of corresponding author: nakagawa-lab-pub@ml.pu-toyama.ac.jp*

もできる．

温度への依存性を表す補正係数を$C_{Arr}$，元の粘度モデルで定まる動粘度を$\nu_{base}$，Arrhenius 型を組み合わせた後に計算へ用いる動粘度を$\nu$とすると，次の関係が成り立つ．

$$C_{Arr} = e^{-\alpha(T - T_\alpha)} \tag{1}$$

$$\nu = \nu_{base} C_{Arr} \tag{2}$$

ここで，$T$は任意の位置の温度，$\alpha$はモデル係数，$T_\alpha$は基準となる温度である．

なお，本論文で扱う `Arrhenius` クラスは，`surfaceFilmModels` 内の `ArrheniusViscosity` クラスとは別のものである．`ArrheniusViscosity` クラスでは $C_{Arr} = \exp(k1(1/(T + k2) - 1/(T_{ref} + k2)))$ とされており，一般的な Arrhenius 型表現により近い．ここで $k1$，$k2$ はモデル定数，$T$は任意の位置の温度，$T_{ref}$は基準となる温度である．

`Arrhenius` クラスは既存の粘度クラスを継承してインスタンス化される．Code 1 に `Arrhenius` クラスの宣言を示す．ここで `ViscousModel` はテンプレートクラスであり，実際には既存粘度クラスのいずれかが使われることになる．

Code 1　Declaration of Arrhenius class.

```
1    template<class ViscousModel>
2    class Arrhenius
3    :
4        public ViscousModel
```

Code 2 に複数の Arrhenius 型粘度モデルを作るためのソースファイル `Arrheniuss.C` の一部を示す．`makeArrheniusTypes` は `makeArrheniusTypes.H` ファイル内で定義されているマクロである．これにより，`ArrheniusCrossPowerLaw` や `ArrheniusNewtonian` といった名前の粘度モデルが使用できるようになる．`ArrheniusCrossPowerLaw` および `ArrheniusNewtonian` は，Code 1 のテンプレートクラス `ViscousModel` に `CrossPowerLaw` および `Newtonian` を使用した粘度となる．このように複数のモデルが作られるが，問題となるのは `Newtonian` モデルだけである．

Code 2　Macro calls in Arrheniuss.C to create several temperature-dependent models.

```
1    makeArrheniusTypes(Arrhenius, BirdCarreau);
2    makeArrheniusTypes(Arrhenius, Casson);
3    makeArrheniusTypes(Arrhenius, CrossPowerLaw);
4    makeArrheniusTypes(Arrhenius, HerschelBulkley);
5    makeArrheniusTypes(Arrhenius, Newtonian);
```

`Arrhenius` クラスの構造は極めてシンプルである．Code 3 に `Arrhenius` クラスの `correct` 関数，Code 4 に `calcNu` 関数の定義を示す．両者を組み合わせると，元の粘度クラスで定めた動粘度$\nu_{base}$に係数$C_{Arr}$を乗じた動粘度$\nu$が得られる．その手順についてコードを説明する．

ソルバ内で粘性を使った計算を実行する前，その時点での動粘度の値へ更新するために `correct` 関数が呼ばれる．`Arrhenius` クラスの `correct` 関数では，まず親クラスである `ViscousModel` の `correct` 関数を実行し，その後に `Arrhenius` クラスの `calcNu(volScalarField&)`関数で求めた補正係数$C_{Arr}$を乗じる．

ここで，動粘度が演算 `*=` で求められていることに注意が必要である．`this->nu_`は，`Arrhenius` と組み合わせて使われている元の粘度クラスのメンバ変数である．非ニュートン粘度タイプでは，`correct` 関数内でその時点の速度場から `nu_`が再計算されるため問題は生じない．一方，`Newtonian` クラスでは `correct` 関数が空であり，`nu_`が初期値に戻されないまま補正係数だけが繰り返し乗じられるため，問題が生じる．

Code 3　Definition of correct function of Arrhenius class.

```
1    //- Correct the laminar viscosity
2    virtual void correct()
3    {
4        ViscousModel::correct();
```

```
5
6       const auto* fldPtr = mesh_.findObject<volScalarField>(fieldName_);
7
8       if (fldPtr)
9       {
10          this->nu_ *= calcNu(*fldPtr);
11      }
12  }
```

Code 4　Definition of calcNu function of Arrhenius class.

```
1   template<class ViscousModel>
2   Foam::tmp<Foam::volScalarField>
3   Foam::viscosityModels::Arrhenius<ViscousModel>::calcNu
4   (
5       const volScalarField& field
6   ) const
7   {
8       return exp(-alpha_*(field - Talpha_));
9   }
```

## 2.2.　正しく動作する Arrhenius 粘度モデル例（CrossPowerLaw）

CrossPowerLaw モデルの correct 関数のコードを Code 5 に示す．動粘度を表す変数 nu_に calcNu 関数の戻り値が格納される．Code 6 に calcNu 関数の定義を示す．

Code 5　Definition of correct function of CrossPowerLaw class.

```
1   //- Correct the laminar viscosity
2   virtual void correct()
3   {
4       nu_ = calcNu();
5   }
```

Code 6　Definition of calcNu function of CrossPowerLaw class.

```
1   Foam::tmp<Foam::volScalarField>
2   Foam::viscosityModels::CrossPowerLaw::calcNu() const
3   {
4       return (nu0_ - nuInf_)/(scalar(1) + pow(m_*strainRate(), n_)) + nuInf_;
5   }
```

ArrheniusCrossPowerLaw モデルでの動粘度の計算方法をまとめると，次の手順となる．ここで，右肩の添字 n および n+1 は時間（あるいは反復回数）を表す．

1. CrossPowerLaw クラスの correct 関数で，現在の速度場に応じた動粘度を算出（温度場の影響はクリアされる），

$$\nu^{n+1} = f(U^{n+1}) \tag{3}$$

2. Arrhenius クラスの calcNu 関数で，現在の温度場に応じた補正係数を算出，

$$C_{Arr}{}^{n+1} = e^{-\alpha(T^{n+1} - T_\alpha)} \tag{4}$$

3. 動粘度と補正係数を乗じる．

$$\nu^{n+1} = \nu^{n+1} C_{Arr}{}^{n+1} = f(U^{n+1})\, C_{Arr}{}^{n+1} \tag{5}$$

このため，CrossPowerLaw と Arrhenius の組合せでは，期待通りの結果が得られる．

### 2.3. 問題のある Arrhenius 粘度モデル（Newtonian）

Newtonian クラスの correct 関数は Code 7 のように定義されており，空となっている．これは，単純な（非 Arrhenius 型）ニュートン流体では粘度が不変であり，計算のたびに再計算する必要がないからである．

この Newtonian クラスを Arrhenius クラスと組み合わせると問題が発生する．温度依存性を考慮した場合，粘度が不変でなくなるためである．ある時刻（あるいは反復回数）で適用した補正係数に重ねて，次の時刻でもさらに補正係数を多重に乗じることになる．つまり，各反復で動粘度が初期値に戻されないため，温度場の変化が累積的に反映され，正しい更新にならない．計算方法をまとめると，次の手順となる．

1. Newtonian クラスの correct 関数では動粘度に対する操作が行われないため，現在の動粘度が維持される（前回の温度場の影響がクリアされない），

$$\nu^{n+1} = \nu^{n} \tag{6}$$

2. Arrhenius クラスの calcNu 関数で，現在の温度場に応じた補正係数を算出，

$$C_{Arr}{}^{n+1} = e^{-\alpha(T^{n+1} - T_{\alpha})} \tag{7}$$

3. 動粘度と補正係数を乗じる．（温度場に応じた係数を多重に乗じる）．

$$\nu^{n+1} = \nu^{n+1} C_{Arr}{}^{n+1} = \nu^{n} C_{Arr}{}^{n+1} = \nu^{n-1} C_{Arr}{}^{n} C_{Arr}{}^{n+1} = \nu_0 C_{Arr}{}^{1} C_{Arr}{}^{2} \cdots C_{Arr}{}^{n+1} \tag{8}$$

Code 7 Definition of correct function of Newtonian class.

```
1    //- Correct the laminar viscosity (not appropriate, viscosity constant)
2    virtual void correct()
3    {}
```

### 2.4. Newtonian クラスの修正

前節の問題を解決するため，Newtonian クラスの correct 関数を修正し，呼び出しのたびに動粘度を初期の一定値 nu0_に戻すようにした．これにより，前回までに適用された温度依存の補正係数がリセットされ，補正係数の多重乗算を防ぐことができる．この手順を式(9)から(11)にまとめる．修正例を Code 8 に示す．別の実装方法としては，CrossPowerLaw クラスと同様に calcNu 関数を作成し，その戻り値を nu0_にすることも考えられる．

$$\nu^{n+1} = \nu_0 \tag{9}$$

$$C_{Arr}{}^{n+1} = e^{-\alpha(T^{n+1} - T_{\alpha})} \tag{10}$$

$$\nu^{n+1} = \nu^{n+1} C_{Arr}{}^{n+1} = \nu_0 C_{Arr}{}^{n+1} \tag{11}$$

Code 8 Definition of correct function of modified Newtonian class.

```
1    //- Correct the laminar viscosity
2    virtual void correct()
3    {
4        nu_ = nu0_; // or create calcNu() with the same line
5    }
```

# 3. 検証用モデル

## 3.1. 非等温平行平板間クエット流れ

Arrhenius クラスと Newtonian クラスとの組み合わせ（ArrheniusNewtonian 粘性モデル）において生じる問題を明らかにするため，単純な熱流動例題として，温度差を伴う平行平板間クエット流れを計算対象とする．図 1 にモデル概要を示す．平板間距離 0.01 m の 2 次元流れであり，流れ方向計算領域も 0.01 m とした．計算領域を 20×20 に等分割した．

下部壁面（$y = 0$ m）は固定し，上部壁面（$y = 0.01$ m）は $x$ 正方向に 1 m/s（$U$x = 1 m/s）で移動させた．上下壁面は滑りなし，流れ方向には周期境界条件を設定した．上部壁面を高温（310 K），下部壁面を低温（300 K）に固定した．流れは定常な層流とし，buoyantBoussinesqSimpleFoam ソルバを用いた．

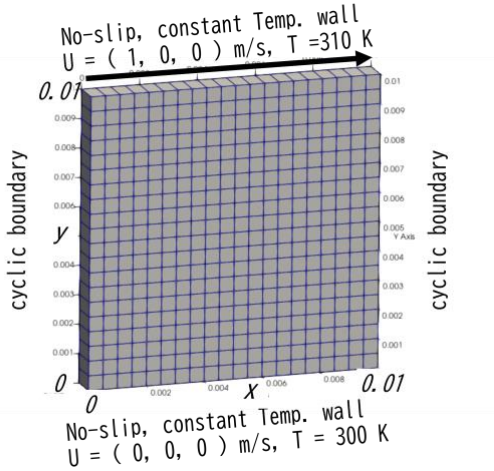


Fig.1 Mesh for two-dimensional planar Couette flow (dimensions in m).

OpenFOAM 標準の 4 つの粘度モデルおよび修正した 1 つの粘度モデルを使用した．1 つはニュートン流体である Newtonian モデル，もう 1 つは非ニュートン流体として Cross-Power Law モデル（CrossPowerLaw）とする．この 2 つに Arrhenius タイプを組み合わせて温度依存性を反映したモデルに加えて，Newtonian モデルの問題点を修正して温度依存性も考慮した ArrheniusNewtonianMod モデルである．

1. 温度依存性なし
    1. Newtonian
    2. CrossPowerLaw
2. 温度依存性（Arrhenius タイプ）あり
    1. ArrheniusNewtonian
    2. ArrheniusCrossPowerLaw
    3. ArrheniusNewtonianMod

CrossPowerLaw モデルでは，動粘度は次の式で求められる．

$$\nu = \nu_\infty + \frac{\nu_0 - \nu_\infty}{1 + (m\dot{\gamma})^n} \tag{12}$$

ArrheniusCrossPowerLaw モデルでは上記の動粘度を$\nu_{base}$とし，次の式によって温度依存性を考慮した動粘度が求められる．

$$\nu = \nu_{base} e^{-\alpha(T - T_\alpha)} = \left(\nu_\infty + \frac{\nu_0 - \nu_\infty}{1 + (m\dot{\gamma})^n}\right) e^{-\alpha(T - T_\alpha)} \tag{13}$$

ArrheniusCrossPowerLaw モデルに対する設定ファイルの一部を Code 9 に示す．今回の検証では，温度の影響だけを明確にするため，せん断速度の指数 $n$ を 0 とし，せん断速度依存性をなくした．この設定では動粘度は$\nu = \nu_0\, e^{-\alpha(T - T_\alpha)}$ となる．

ArrheniusNewtonian モデルに対する設定ファイルの一部を Code 10 に示す．温度に関する係数は Code 9 と同一であり，動粘度基準値も Code 9 の nu0 と同じである．従って，この設定では ArrheniusCrossPowerLaw

モデルと ArrheniusNewtonian モデルは，同一の温度場に対して実質的に同じ動粘度を与えるはずである．

Code 9　A part of transportProperties dictionary file for ArrheniusCrossPowerLaw.

```
transportModel ArrheniusCrossPowerLaw;
alpha 0.01;
Talpha 300;
CrossPowerLawCoeffs
{
    nu0           [0 2 -1 0 0 0 0] 1e-05;
    nuInf         [0 2 -1 0 0 0 0] 1e-05;
    m             [0 0  1 0 0 0 0] 1;
    n             [0 0  0 0 0 0 0] 0;
}
```

Code 10　A part of transportProperties dictionary file for ArrheniusNewtonian.

```
transportModel ArrheniusNewtonian;
// Arrhenius
alpha 0.01;
Talpha 300;
// Newtonian
nu            1e-05;
```

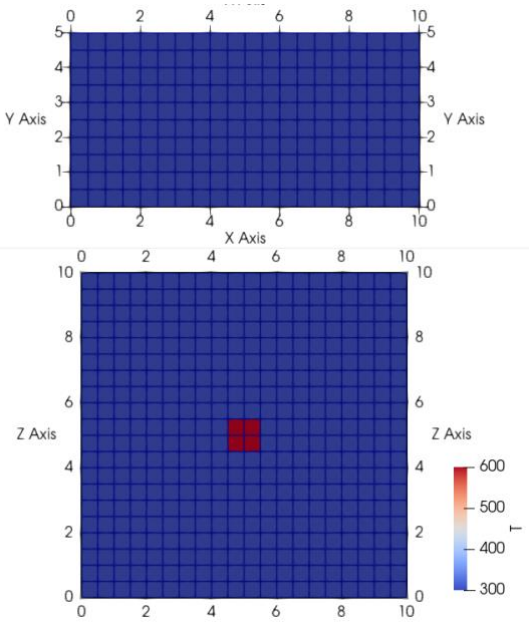


Fig.2 Mesh for hotRoom tutorial (dimensions in m). Front and bottom views.

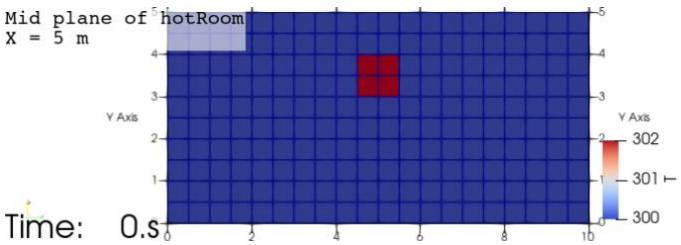


Fig.3 Initial temperature distribution at the center section for modified hotRoom tutorial.

### 3.2.　底面の一部が加熱される室内対流 hotRoom

修正した粘度モデルの有効性をより複雑な流れ場でも確認するため，buoyantBoussinesqPimpleFoam ソルバ付属の hotRoom 例題を元に，粘度モデルおよび温度初期条件を変更した．前節の例とは異なり，非定常

な乱流を扱う．乱流モデルは RANS 型の $k-\varepsilon$ モデルである．

図 2 にモデルの概要を示す．床面積 10 m × 10 m，高さ 5 m の直方体領域を 1 辺の長さ 0.5 m の立方体セルに分割した．底面中央部には高温（600 K）領域が存在する．その他の全ての面の温度は 300 K に固定する．初期条件として，床から 3〜4 m の高さに 310 K の高温領域を設定した．図 3 に計算領域の中央断面における初期温度分布を示す．

## 4. 結果と考察

### 4.1. 非等温平行平板間クエット流れ（CrossPowerLaw での検証）

非ニュートン粘度モデルの `CrossPowerLaw` クラスを使って，`Arrhenius` クラスとの組み合わせが正しく働く場合の動作を確認する．図 4 と図 5 に計算領域中央での $x$ 方向速度，温度，動粘度の分布を示す．図 4 は `CrossPowerLaw` の結果，図 5 は `ArrheniusCrossPowerLaw` の結果である．

`CrossPowerLaw` では，温度が変化しても粘度が一定となっていることが確認された．これに温度依存性を加えた `ArrheniusCrossPowerLaw` では，温度が上昇する上部壁面に近づくほど動粘度が低下していることがわかる．

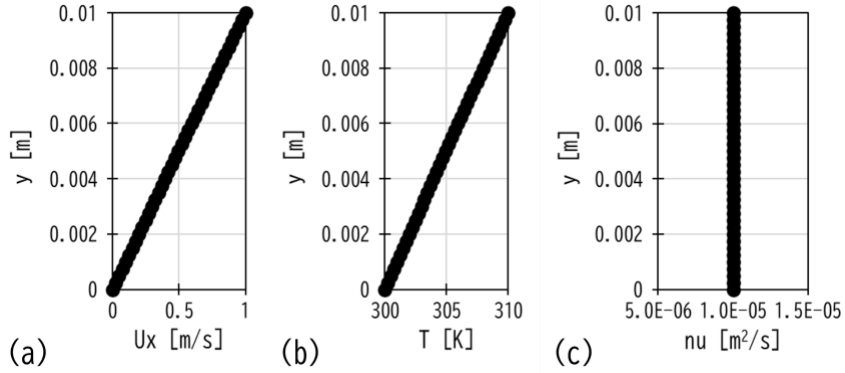


Fig.4 Profiles of (a) stream-wise velocity *Ux*, (b) temperature *T*, and (c) kinematic viscosity $\nu$ for CrossPowerLaw viscosity model at the center of calculation domain (*x* = 0.005m).

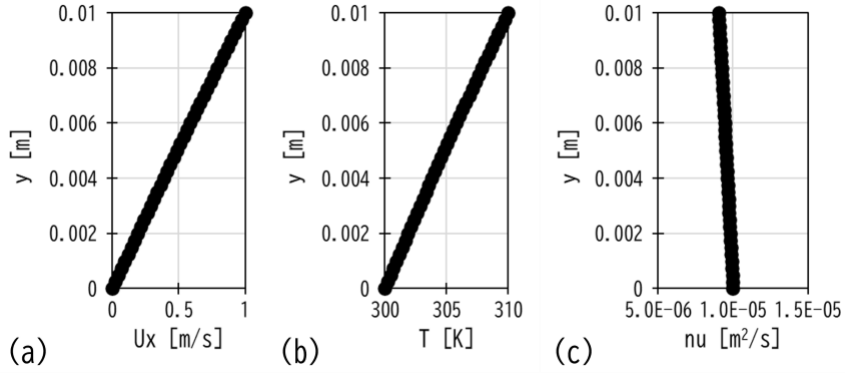


Fig.5 Profiles of (a) stream-wise velocity *Ux*, (b) temperature *T*, and (c) kinematic viscosity $\nu$ for ArrheniusCrossPowerLaw viscosity model at the center of calculation domain (*x* = 0.005m).

### 4.2. 非等温平行平板間クエット流れ（Newtonian での問題）

温度依存性を考慮していない `Newtonian` モデルでの結果を図 6 に示す．このモデルでは動粘度は温度に依存せず一定値であることが確認できる．これに対し，図 7 に示す `ArrheniusNewtonian` モデルでは，結果が大きく異なる．本来は図 5 と同じ動粘度の分布となるべきであるが，温度に応じた補正が累積した結果，動粘度が過小評価されている．`buoyantBoussinesqSimpleFoam` では，温度拡散率を $\alpha = \nu / Pr$ として求めているため，動粘度の低下は温度場にも影響を及ぼす．粘度が非一様となることで速度分布の直線性が失われる．

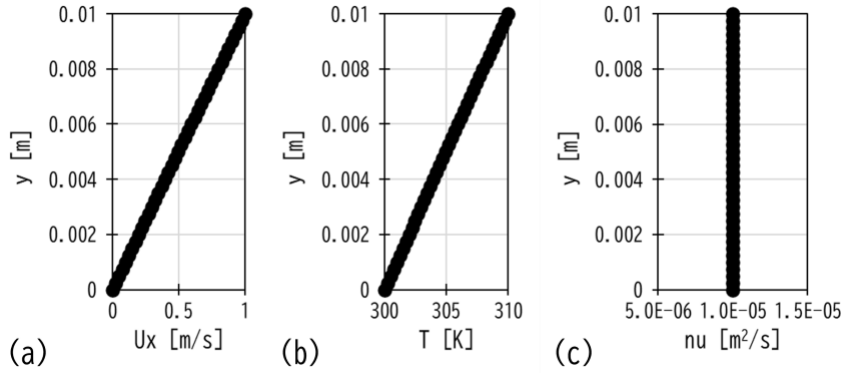


Fig.6 Profiles of (a) stream-wise velocity *Ux*, (b) temperature *T*, and (c) kinematic viscosity $\nu$ for Newtonian viscosity model at the center of calculation domain (*x* = 0.005 m).

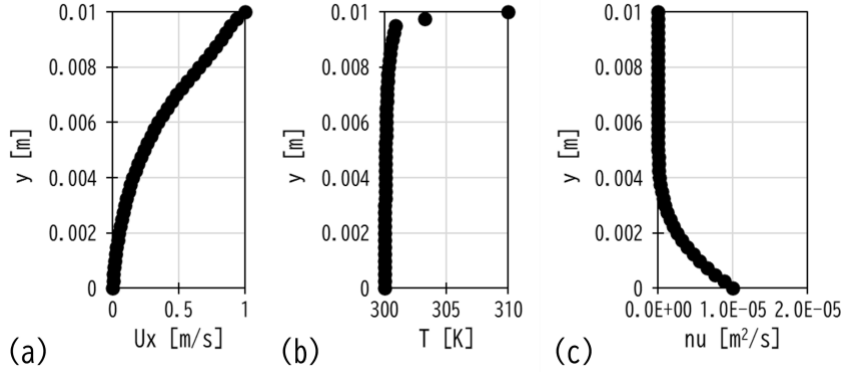


Fig.7 Profiles of (a) stream-wise velocity *Ux*, (b) temperature *T*, and (c) kinematic viscosity $\nu$ for ArrheniusNewtonian viscosity model at the center of calculation domain (*x* = 0.005 m).

### 4.3. 非等温平行平板間クエット流れ（改良 Newtonian での検証）

図 8 に問題点を修正した `ArrheniusNewtonianMod` での結果を示す．上部壁面に向かって直線的に上昇する温度分布に対応して，動粘度が減少している．

図 8(d)には `ArrheniusCrossPowerLaw` と `ArrheniusNewtonianMod` とで得られた動粘度の差を示した．両者の差は全領域で `0` であり，設定上同一の動粘度を与えるべき 2 つのモデルが一致していることが確認できる．従って，温度に依存する前の粘度を初期値に戻すことで，`Newtonian` モデルでも他の粘度モデルと同様に，温度の影響が正しく反映されることを確認した．

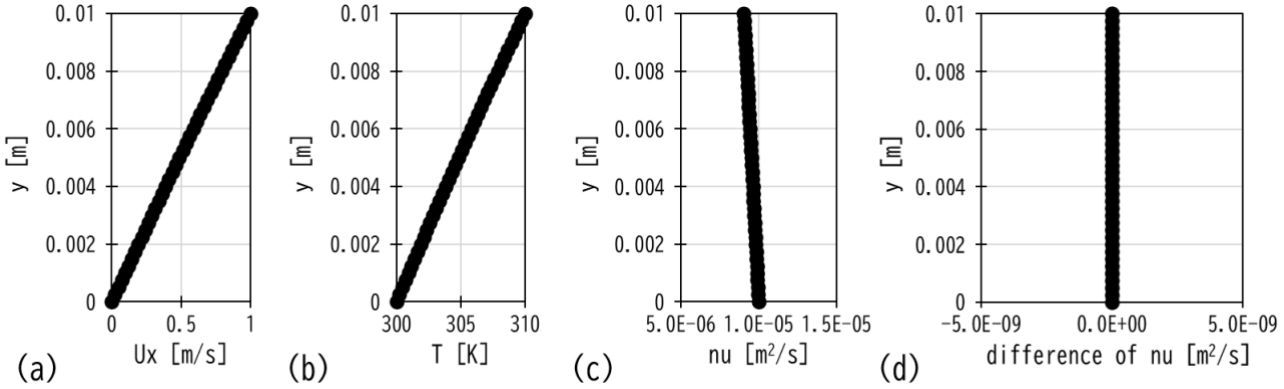


Fig.8 Profiles of (a) stream-wise velocity *Ux*, (b) temperature *T*, (c) kinematic viscosity $\nu$ for ArrheniusNewtonianMod viscosity model, and (d) difference of $\nu$ between ArrheniusCrossPowerLaw and ArrheniusNewtonianMod at the center of calculation domain (*x* = 0.005 m).

### 4.4. 室内対流 hotRoom（Newtonian モデルの改良効果の検証）

非定常乱流自然対流を伴う hotRoom 例題の計算結果を図 9 に示す．図 9 の左側は修正前の `Arrhenius` と `Newtonian` の組み合わせ，右側は修正した `Newtonian` クラスを使った結果である．計算領域中心断面での温度分布と動粘度分布を示す．計算初期（2 s）の(a)と(c)では，初期値として配置した高温空気部分で動粘度が小さいことが確認できる．

図 9 (b)および(d)は 12 秒経過後である．修正前の `ArrheniusNewtonian` モデルでは，高温領域が自然対流によって上方へ移動・拡散しているにもかかわらず，初期の低動粘度領域が元の位置に残り続けている．さらに，その位置での動粘度は初期よりも低下している．修正後の `ArrheniusNewtonianMod` モデルでは，温度場の変化に応じて動粘度が周囲と近い値へ戻っており，不自然な温度依存性の累積効果が解消されている．

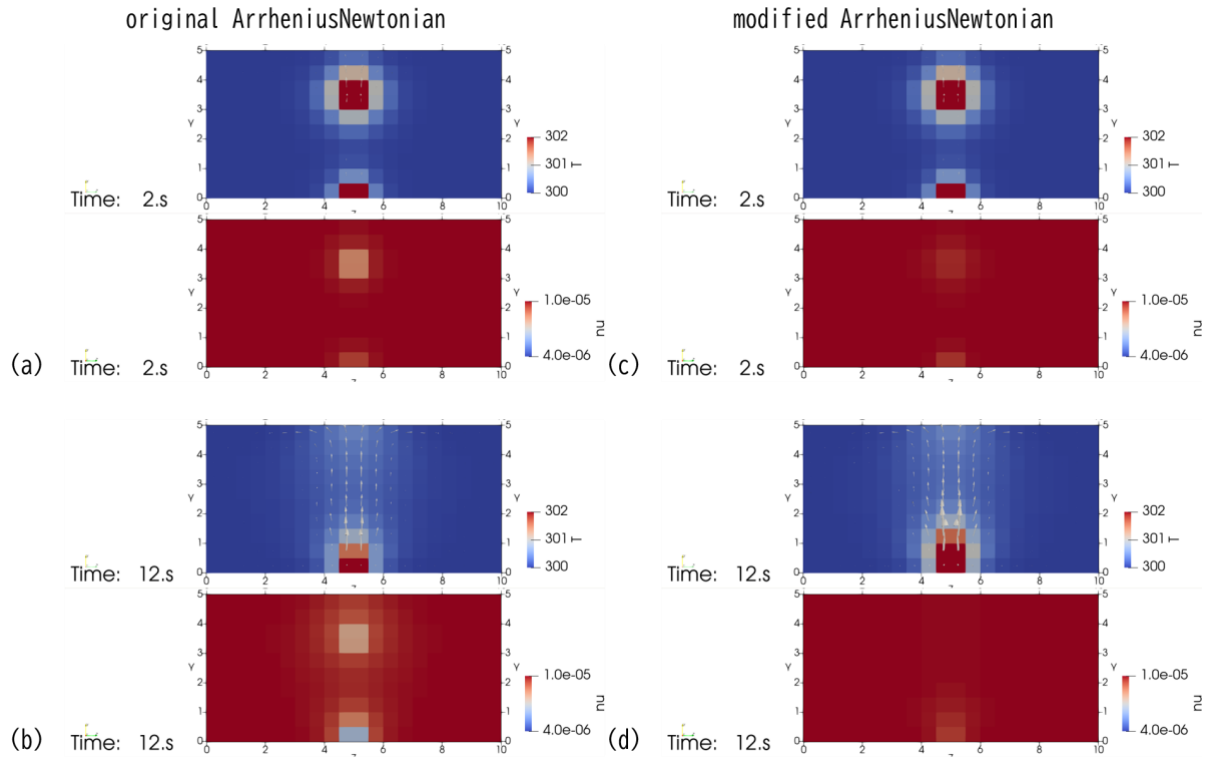


Fig.9 Temperature (upper panel) and kinematic viscosity (lower panel): (a, b) original ArrheniusNewtonian at 2 s and 12 s; (c, d) modified ArrheniusNewtonian (ArrheniusNewtonianMod) at 2 s and 12 s.

## 5. 結論

OpenFOAM のニュートン粘度モデルに温度依存性を付加する `Arrhenius` クラスを組み合わせると，動粘度が正しく計算されないことを発見した．本研究では，この原因が `Newtonian` クラスの `correct` 関数にあることを示し，動粘度を初期値へ戻す修正を提案した．

基本的な例題を使って，提案する修正により正しい挙動を得られることを示した．提案した修正は OpenFOAM の不具合報告として登録済み[1]であり，今後の OpenCFD 版 OpenFOAM での反映が期待される．

現時点での最新版である OpenFOAM v2512 においても，本論文で報告した問題は存在しているため，標準状態のままで `ArrheniusNewtonian` 粘度モデルを使用すると正しい結果が得られないことに注意が必要である．

## 参考文献

[1] “Issue #3432 ArrheniusNewtonian model fails to update viscosity according to temperature”. OpenFOAM Gitlab repository. https://gitlab.com/openfoam/core/openfoam/-/issues/3432, (accessed 2026-03-31).

[2] Mark Olesen. “ENH: Arrhenius viscosity model and energyTransport function-object”. OpenFOAM Gitlab repository. 2017-10-26. https://gitlab.com/openfoam/core/openfoam/-/commit/ffabc42dbf994cae732354e28dd2265ebd73a00e, (accessed 2026-03-31).