**流れ関数-渦度法による2次元キャビティ流れのシミュレーション**

1865095 篠塚 温志

**１. はじめに**

**1. 1　プログラムの背景**

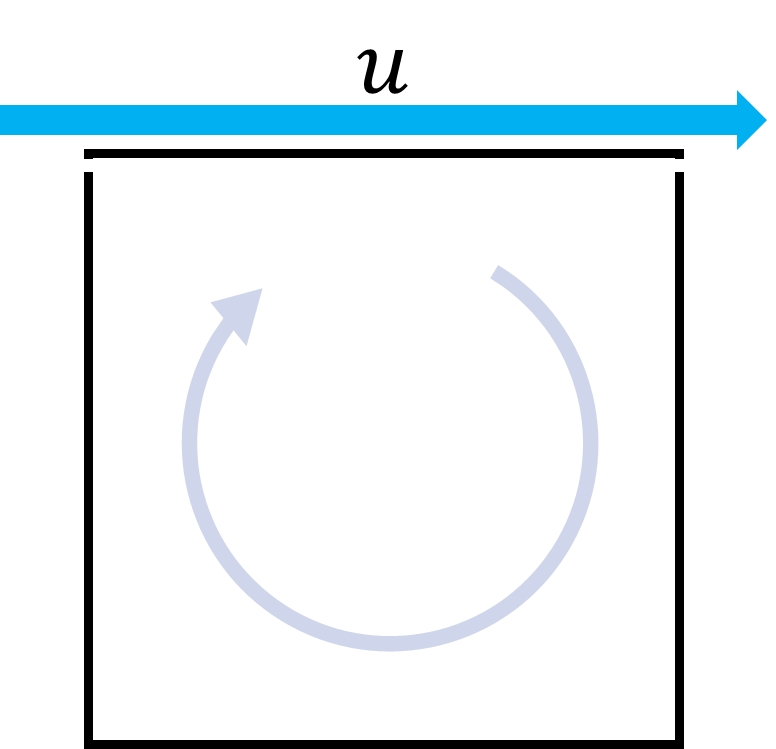
　流体の流れは座標と時間の非線形偏微分方程式であるNavier-Stokes 方程式(以下NS方程式)で記述され，一般的にこの解を求めるのは困難である．このため複雑な流れの解析は計算機を用いて行われるが，これは空間を離散化し各地点で離散化した偏微分方程式を解くことで流れをシミュレートするものである．今回作成したプログラムはこの実現である．

**1. 2　プログラムの目的**

　実際に作成したプログラムは，『2次元キャビティ流れについて流れをシミュレートするプログラム』である．ここでキャビティ流れとは，図1に示したように正方形のうち1辺のみが一定速度で動いている時に正方形である．

**1．3　用いた手法**

　図1に示した条件の流れについて，支配方程式であるNS方程式を流れ関数と渦度で記述し，それらについての偏微分方程式を中心差分近似したものをガウス-ザイデル法を用いて解くことで解を求めた．



|  |  |
| --- | --- |
| 支配方程式 | NS方程式 |
| 渦 | あり |
| 粘性 | あり |
| 圧縮性 | なし |

図1　2次元キャビティ流れと計算条件

**2.　関連する理論**

**2. 1　NS方程式**

支配方程式であるNS方程式はベクトル表示で以下の通りである．

いま，流体の圧縮性が無視できるものとすると，非圧縮性流体の連続の方程式

が成り立つ．さらに質量力も働かない()とすると，式は以下のように書き直すことができる．

式について，次のような変数変換により方程式を無次元化する．

これらの変数をに代入し，2次元で書き直すと

となる．(はレイノルズ数). 今回作成したプログラムではこの式を元に流れの解析を行った．

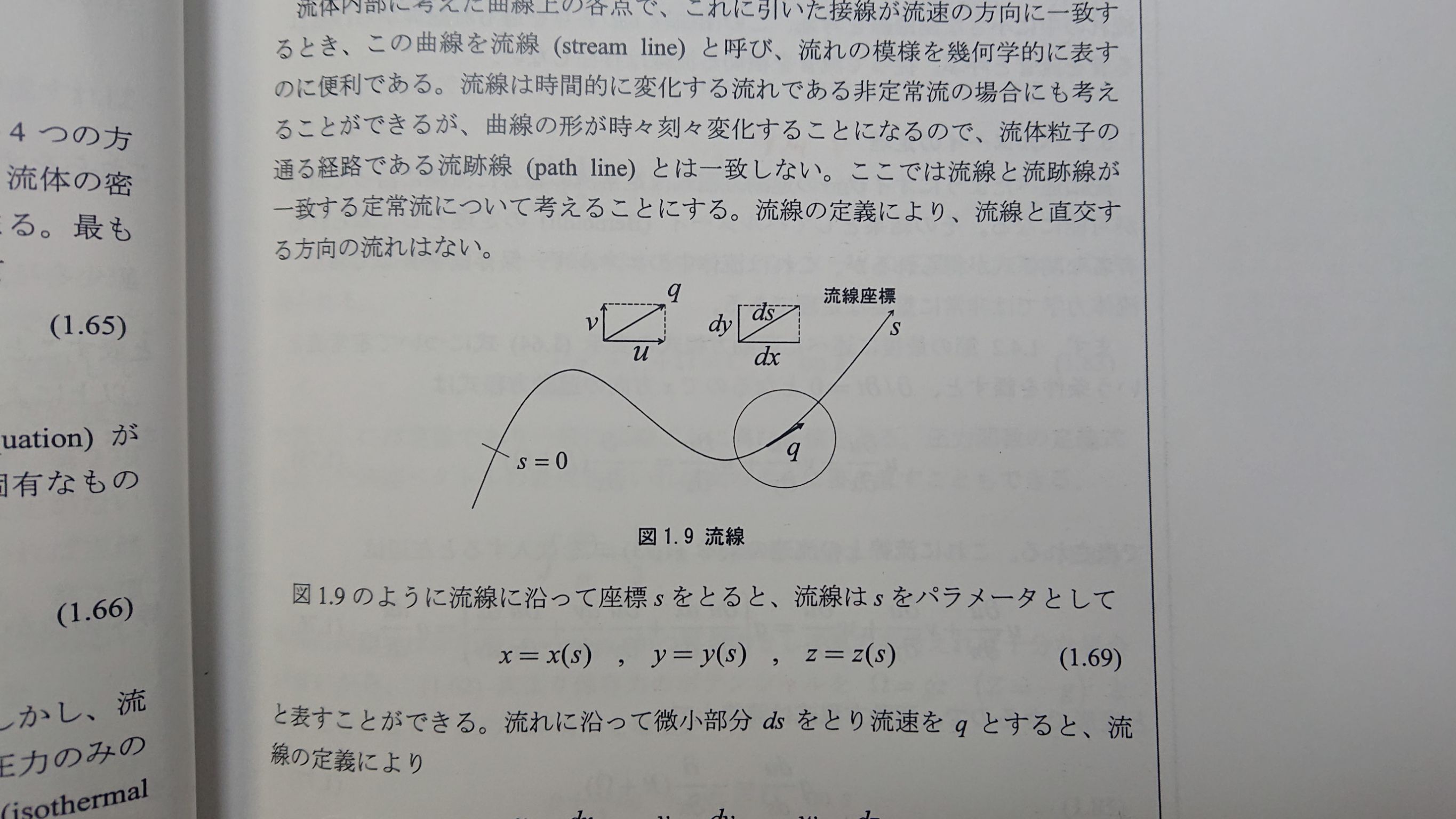
**2. 2　流れ関数・渦度と流線**

　流れ関数,渦度は次のように定義される．

このとき,

が恒等的に成り立つ．すなわち流れ関数を用いれば必然的に連続の式が成り立つ．

　流線とは，曲線の各点でこれに引いた線が流速の方向に一致する曲線のことである．

  
図2　流線　(出典：参考文献[1] p.19)

2次元の流れにおいて流線は以下の微分方程式の解で与えられる．

ここで流れ関数の全微分を求めてみると，

となり，とすれば流線の式に一致する．であるから，一定である曲線が流線となることが分かる．つまり2次元の座標に対して高さ方向に各地点のをとった3次元のグラフを考えたとき，流線は等高線となる．

**2. 3　流れ関数-渦度法**

支配方程式として，無次元化した2次元のNS方程式をとをで記述し直したものを用いることを考える．これを流れ関数-渦度法という．

)式をで微分したものから式をで微分したものを引くと，に注意して

となる．これを渦度輸送方程式という．また渦度の定義式である式は式を用いて

と表すことができる．式と式はとについて閉じた方程式となっており，連立させて解くことでとを求めることができる．

**2. 4　2次元キャビティ流れの境界条件**

　粘性のある2次元キャビティ流れについて，壁面における流速は壁面と同じである．つまり左右及び下面では流速が0，上面では流速が無次元化されたである．壁面上に点Qをとりそこから一つ内側の接点をPとすると，Qにおける流れ関数はPのまわりのテイラー展開で

**3　計算手法**

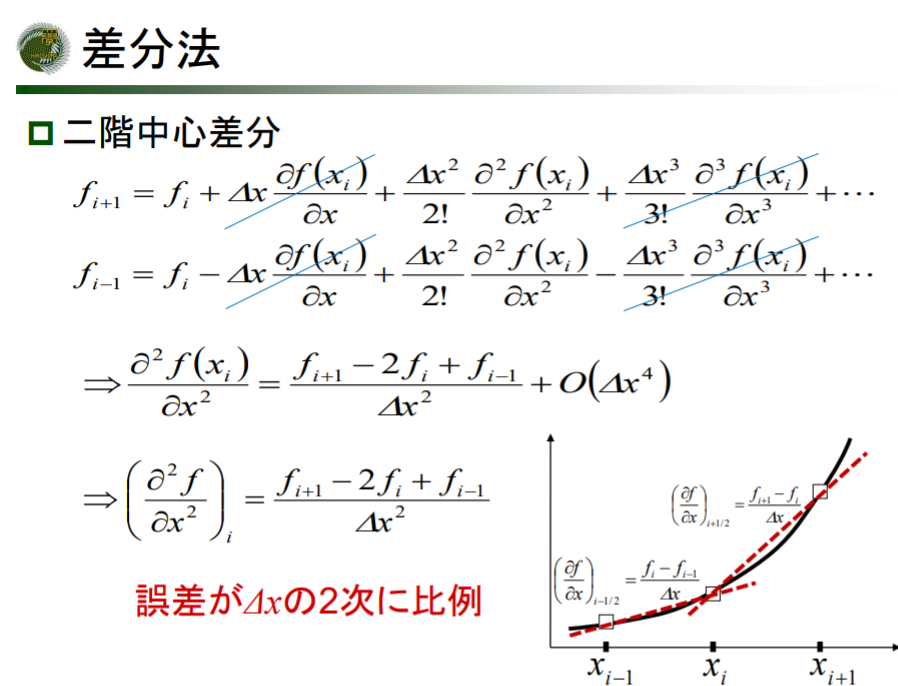
**3. 1　セル接点法**

今回のプログラムでは有限体積法のセル接点法を用いた．つまり平面を格子に切り，その格子の各接点でとを計算することで流れを解析している．

**3. 2　中心差分近似**

2階の中心差分近似は,ある点の前後の点をにおける関数の値を用いて偏微分を求める方法である．具体的に例えばを求めることを考えると，以下のような手続きを行う．

式と式は2階の偏微分方程式であり，今回のプログラムではその離散化に中心差分近似を用いた．

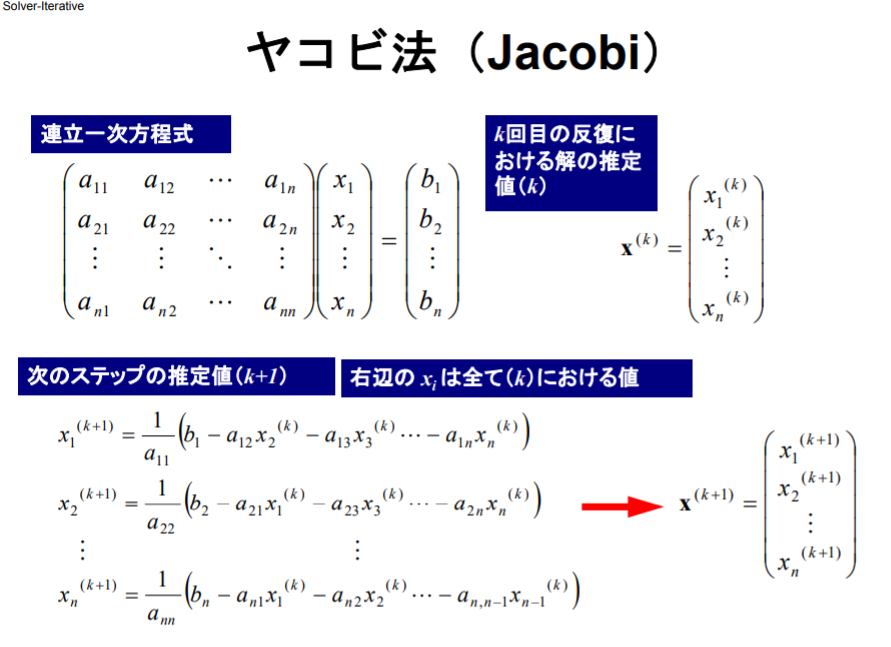
****

**3. 3　ガウス-ザイデル法**

以下の連立1次方程式を解くことを考える．

　ガウスザイデル法では以下のような反復計算によって解を求める．

ここで式中のは反復回数である．適当に与えた初期値をに代入し，そこから以下の条件を満たすまで反復を繰り返して解を求める．

****

**4　作成したプログラム**

**4. 1　計算の流れ**

　計算の流れの概略を図　に示す．

なお本レポートで使用した記号とPARAM.datの対応は表　の通りである．

**4. 2　計算上の注意**

今回の計算において注意しなければならないのは以下の2点である．

①　無限ループに入らないこと

②　計算を発散させないこと

　①についてはStop\_itr

　②についてはCalc\_max

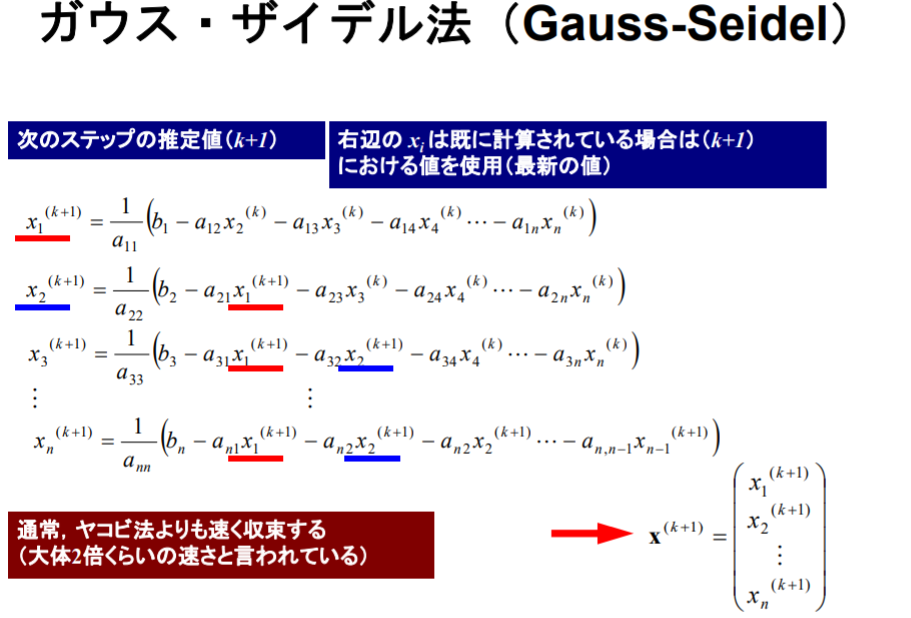
**5．結果**

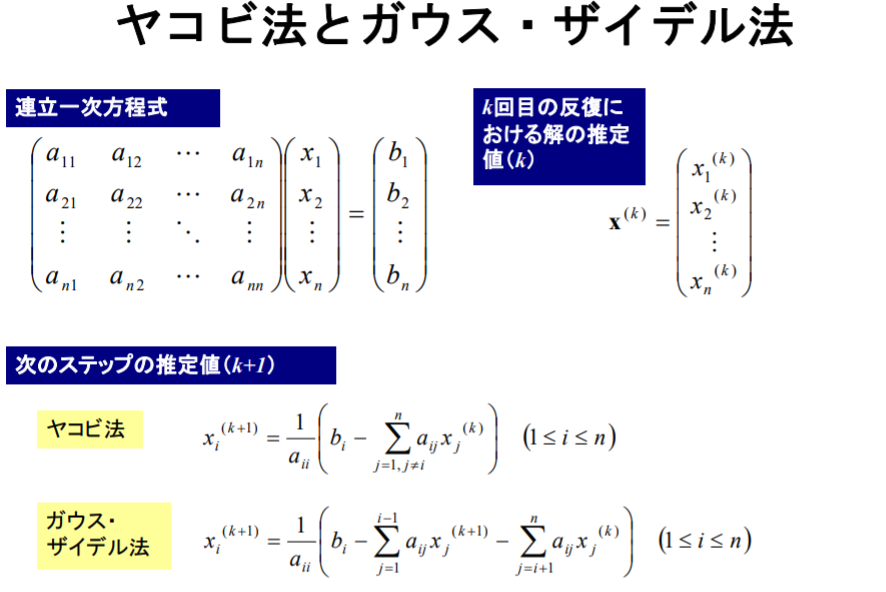
**6．考察**

**6. 1計算過程について**

　実行中の表示を観察すると，図　のようにステップ数が増えるにつれガウスザイデル法の反復回数が減っていることが分かる．これは時間を進めるにつれ流れが収束してくることが原因であると考えられる．今回収束判定による自動計算終了の機能は実装していないが，ガウスザイデル法の反復回数を元にした収束判定が可能であると考えられる．

**7．結論**





<http://nkl.cc.u-tokyo.ac.jp/13n/SolverIterative.pdf>

**2. 4　Runge-Kutta Fehlberg法**

Runge-Kutta Fehlberg法では，以下の計算方法で順次を求める．

①以下のを求める．

②を用いて及びを以下の式で求める．

③を計算する．であれば を解とし，刻み幅ものまま変更せず①へ戻る．

④の時，を計算しこれを元にを修正する．

ただし急激なの変更を防ぐため，

(2.4.4)

の時，

の時　とした．

⑤修正したを用いて①から計算をやり直す．

⑥もしが計算範囲を超えたら終了とする．

Runge-Kutta Fehlberg法は『埋め込まれたRunge-Kutta法』のうちの一つである．４次以上のRunge-Kutta法は基本的に４個以上の計算が必要となるが，Fehlbergは５次のときに用いるを用いて４次の方法を表すことができる(＝埋め込まれている)ことを発見し，これを誤差評価に用いた．このため，４次の式について係数が特殊なものとなっている．

この計算法の特徴的な点は，誤差に応じて刻み幅を制御することである．誤差が大きすぎる場合は刻み幅を小さく，誤差がほとんど無い場合は刻み幅を大きくすることで，関数の傾きの変化に応じて適切な刻み幅をとることができる．これによって単に５次精度で計算するよりも精度を高く，また計算速度を速くすることが期待できる．

ここで，手順②の代わりに以下の式のようにを直接求めることを考える．これが以下であったときのみを計算することで計算手順が減るため，計算速度は速くなると考えられる．しかしながら今回はデバッグの容易さを鑑みておよびを個別に求めその差をとることとした．

(2.4.1)

**3. 作成**

**3.1　プログラムの仕様**

プログラムⅠは倍精度計算でEuler法，Heun法，Runge-Kutta法を用いて解を求めるプログラムであり，フローチャートは図2の通りである．

(2.4.2)

プログラムⅡは4倍精度計算でRunge-Kutta Fehlberg法を用いて解を求めるプログラムであり，フローチャートは図２の通りである．

(2.4.3)

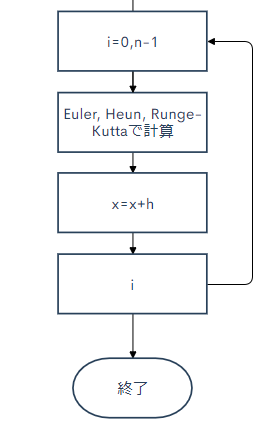
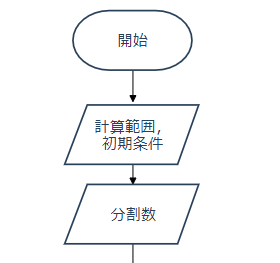


図１　プログラムⅠのフローチャート

テキスト が含まれている画像

自動的に生成された説明

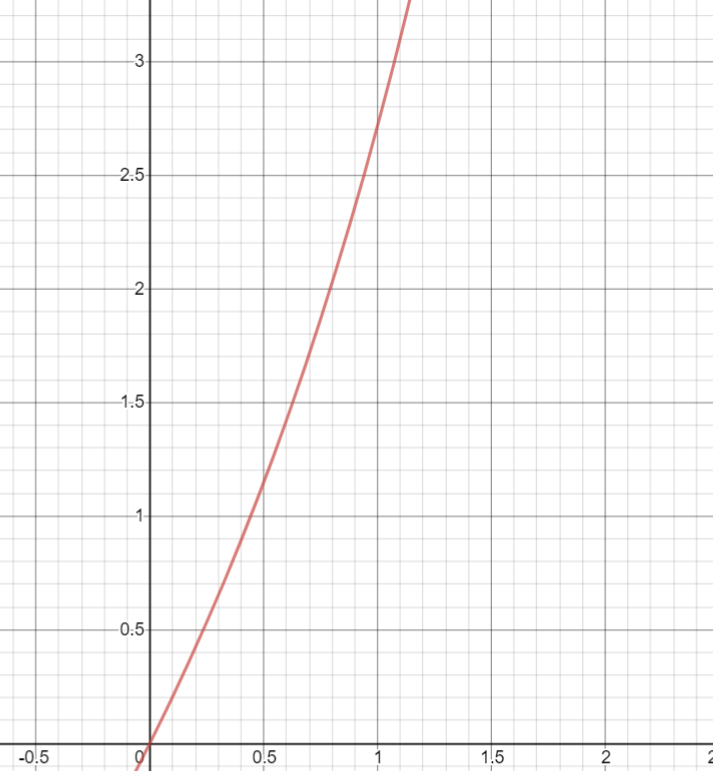
図２　プログラムⅡのフローチャート

**４．　実験**

**４．１ 実験１：適切な刻み幅の調査**

　プログラム１について,，計算区間で刻み幅を様々に変えながら解析解との誤差を測定した．

なお関数について初期条件での解析解はであり，これは以下のグラフのようになる．



グラフ１　のグラフ

計算区間内で関数も傾きも単調増加するため，次数が高くなるに従って計算精度が高くなると考えられる．

における計算結果(解析解)は以下のグラフのようになる．

計算区間の分割数

グラフ２　における計算結果

解が小数点4桁まで解析解に揃う刻み幅を調べたところ，以下のような結果となった．

Euler： 不明

Heun：

Runge-Kutta：

Euler法については刻み幅を小さくし続けても解が小数点4桁まで解析解に揃うことが無かったため『不明』とした．

**４．２　実験２：刻み幅制御**

４．１と同じ微分方程式について同じ条件の下で，Runge-Kutta Fehlberg法を用いて計算範囲をに狭めて計算した．なお初期刻み幅にはを用い，計算が１００0ステップ以上になった場合打ち切るものとした．以下はとにおける刻み　幅の関係を表す表である．

|  |  |  |
| --- | --- | --- |
|  |  | 備考 |
|  |  | 刻み幅変更なし |
| ・・・ | ・・・ | ・・・ |
|  |  | 刻み幅変更なし |
|  |  | 刻み幅変更あり |
| ・・・ | ・・・ | ・・・ |
|  |  | 刻み幅変更あり |
|  |  | 計算打ち切り |
| ・・・ | ・・・ | ・・・ |

微分方程式をRunge-Kutta Fehlberg法で解く場合，有効なの幅は極めて狭いことが分かった．

**４．３ 実験３：計算手法の比較**

これまでの実験と同じ微分方程式について，計算区間，初期刻み幅で各計算法を用いて計算した結果をグラフ３に表した．なおとした．

グラフ３に示したとおり，Euler法において若干の誤差はあるもののその他の計算法ではほとんど誤差がないことが分かった．

**５．　考察**

**５．１　実験１について**

グラフ３　各計算法の比較

仮説通り，次数の高い計算法ほど刻み幅が大きくても精度がよいことが分かった．

　Euler法については『不明』という結果になったが，これはEuler法が１次精度であることによる妥当な結果だといえる

**５．２　実験２について**

今回の実験では有効なの値の設定が極めて難しく，刻み幅の変更を行いながら計算し，かつステップが過剰にならないは非常に限られたものであった．Runge-Kutta Fehlberg法は刻み幅を自動で

制御するため適切な刻み幅を考える必要が無いことが特徴の一つである．しかしながら実験２の結果から分かるように，有効なの設定が非常に難しい．つまり誤差を指定した値以内に納めたい場合以外を除いて４次のRunge-Kutta法に対して優位性が無いと考えられる．

**５．３　実験３について**

Euler法以外のどの計算法においても精度よい結果が得られ，計算精度としては問題が無いといえる．実験1，2の結果と併せて考えると，が大きくても精度のよい4次のRunge-Kutta法が最も優れていると結論づけられる．

**６．　結論**

Euler法，Heun法，4次のRunge-Kutta法を用いて．微分方程式を解いた．刻み幅の大きさに対する精度の良さが最も優れているのは4次のRunge-Kutta法であり，次いで Heun法，Euler法であることが分かった．またRunge-Kutta Fehlberg法を用いて同じ微分方程式を解いたところ，刻み幅制御を用いて4次のRunge-Kutta法と同等の精度を出すことができた．しかしながら有効なの値の設定が非常に難しいことが分かった．

**参 考 文 献**

1) 加川友己「埋め込み型ルンゲ・クッタ公式の利用[www.aoni.waseda.jp/ykagawa/ode1html/node11.html](http://www.aoni.waseda.jp/ykagawa/ode1html/node11.html)（2019年1月9日閲覧）

2)sikino「ルンゲ=クッタ法の説明と刻み幅制御」<http://slpr.sakura.ne.jp/qp/runge-kutta-ex/> (2019年1月9日閲覧）

3) Dr\_ASA「オイラー法/ホイン法/ルンゲクッタ法をつかった常微分方程式の数値解析超入門」<https://qiita.com/Dr_ASA/items/d44560cf1367290aee6d>(2019年1月9日閲覧）

感想

　◎発表時先生に質問いただいた事項について

　発表後の質疑応答にて先生に以下の質問をいただきました．

　　Q1．Runge-Kutta Fehlberg法の4次の解の係数が一般的なものと異なるのはなぜか

　　Q2. Runge-Kutta Fehlberg法で4次精度と5次精度の解両方計算し精度を高めるならば，単に5次精度の解を計算するだけでよいのでは無いか

　　　Q1について，レポート内でも述べたとおり，Runge-Kutta Fehlberg法は『5次の解を求める「ついで」に4次の解を共通のを用いて求め，誤差の改善をする』というけいさんほうであるようなので，共通のを使うため係数が特殊なものになっていることが分かりました．

　　　Q2について，特に急峻に値が上昇する関数を考えると，たとえ5次精度でも誤差が出ると予想できます．これに対して刻み幅を制御して細かくしていくことで，固定幅の5次精度よりも精度が高くなることが予想されます．

　　◎Runge-Kutta Fehlberg法について

　　　　　今回有効なは非常に限られたものであり，あまり計算法に有効性はないと結論づけました．しかしながら，MATLABに実装され広く使われている常微分方程式ソルバ“ode45”はRunge-Kutta Fehlberg法を用いています．

有効なtolを素早く的確に決める方法があればRunge-Kutta Fehlberg法は有効な計算法になり得ると考えられるため，ode45はこの機能を持っているのではないかと予想しています．今後さらに調べを進めたいと思います．