反応拡散系に対する分岐計算ソフトウェア RD-AUTO の使用マニュアル Ver.0.98

林　悠帆（金沢大学大学院自然科学研究科）
田中　直毅（金沢大学理工学域数物科学類）
樋口　亮（金沢大学大学院自然科学研究科）
若井　健（金沢大学大学院自然科学研究科）
長山　雅晴（北海道大学電子科学研究所）

平成 26 年 12 月 9 日

マニュアル更新履歴

年月日	更新者	更新内容
2010.03.20	林悠帆	マニュアルの大枠作成．デモ 1 作成.
2011.03.22	田中直毅	第 2 章の詳しい説明の追加．デモ 2 作成.
		鳥瞰図の節を作成．他.
2011.10.17	樋口亮	デモ 1，デモ 2 の修正．デモ 3 作成.
2012.12.20	若井健	デモの不具合を修正．Cross-Diffusion の定常解計算プログラム作成
2013.08.10	長山雅晴	本ソフトウエアの正式名称を「RD-AUTO」と決定
2013.08.10	長山雅晴	公開準備のためデモ 2 を削除
今後の予定		Cross-diffusion のデモ作成
		双安定反応拡散系のデモ作成
		3 変数反応拡散系のデモ作成
		空間 2 次元定常解とその安定性のデモ作成

目 次

第 1 章　はじめに **3**

第 2 章　RD-AUTO のインストール **4**
2.1 この章で扱う記号について 4
2.2 必要な環境 4
2.2.1 gnuplot のインストール 4
2.2.2 mpich のインストール 5
2.2.3 RSA 鍵を使った mpi のプロセス通信設定 7
2.3 RD-AUTO の解凍 8
2.3.1 Mac OSX を使って RD-AUTO を使う方法 9

第 3 章　発展方程式 **10**
3.1 使い方 10
3.1.1 時間発展方程式のプログラムに対するフローチャート 10
3.1.2 実行前の設定 11
3.1.3 実行コマンドの説明 17
3.1.4 出力されるデータファイル 19
3.1.5 gnuplot の設定 22
3.1.6 周期解の周期の求め方 23
3.1.7 データカット 26
3.2 付録 28

第 4 章　分岐計算プログラム **29**
4.1 使い方 29
4.1.1 実行前の設定 29
4.1.2 実行コマンドの説明 36
4.1.3 解の枝の安定性を求める方法 37
4.1.4 出力されるデータファイル 38

第 5 章　デモ 1 **41**
5.1 1 次元 Gray-Scott モデルの時間発展方程式 43
5.2 1 次元 Gray-Scott モデルの分岐計算 52

第 6 章　デモ 2 **95**
6.1 時間発展方程式 97
6.2 1 次元発熱反応モデルの分岐計算 106

第 **7** 章　付録 **143**
7.1　鳥瞰図 143
7.2　実行時に起きるエラーの対処法 146

第1章　はじめに

本マニュアルは，次の空間 1 次元 n 変数反応拡散系に対する，定常解と周期解の分岐構造を数値計算するソフトウェア (RD-AUTO) の解説書です:

$$\frac{\partial \boldsymbol{u}}{\partial t} = D\frac{\partial^2 \boldsymbol{u}}{\partial x^2} + \boldsymbol{f}(\boldsymbol{u}), \quad t > 0, \quad x \in (0, L),$$

ただし，

$$D = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix}, \quad d_i \geq 0 \quad (i = 1, 2, \ldots, n), \quad \boldsymbol{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \in \mathbb{R}^n, \quad \boldsymbol{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \in \mathbb{R}^n.$$

※ d_i が全て 0 のとき方程式は常微分方程式になるが，ソフトウェアは動きます．ただし，常微分方程式を解く場合は AUTO の使用を推奨します．

RD-AUTO は Newton 法により定常解と周期解を求めるため，Newton 法を用いるための適切な初期値が必要です．その初期値は時間発展方程式を解いて求めるため，本マニュアルでは時間発展方程式を解くプログラムの使い方も説明します．自ら適切な初期値を与えることができる場合は，本マニュアルで解説する時間発展方程式を使って初期値を与える必要はありませんが，RD-AUTO に受け渡すデータ形式に合わせる必要がありますので，注意してください．なお，データ形式についての解説は現時点[1]で解説していません．RD-AUTO や時間発展方程式を解くプログラムの使い方を説明するために，例として 2 変数反応拡散系に現れるパルス波の分岐構造を求めいています．そのためデモはすべて周期境界条件下で計算していますが，Dirichlet 境界条件や Neumann 境界条件でも計算可能です．本マニュアルは次のような構成となっています．

第 2 章では，時間発展方程式と RD-AUTO のプログラムを動かすために必要な環境設定を解説します．

第 3 章では，時間発展方程式を解くプログラムの説明を行います．

第 4 章では，RD-AUTO の説明を行います．数値計算法の具体的な解説は後日公開します．

第 5 章，第 6 章では，具体的な方程式を用いて使い方の説明を行います．

[1] 2015 年 8 月 11 日

第2章　RD-AUTOのインストール

Intel製の有料コンパイラであるicc，ifortが使えることを前提としていますが，gcc, gfortanを用いてRD-AUTOを動かすことも可能です．周期解の数値計算では計算量が多くなるのである程度の計算機性能が必要です．ノートPCを使う場合は定常解まで計算できればよいと思ってください．

2.1　この章で扱う記号について

最初に，この章で扱うコマンドプロンプトの記号の説明を行う．
$ のあとに書かれているコマンドは一般ユーザで入力する．
# のあとに書かれているコマンドはスーパーユーザで入力する．
斜体で *version* と書かれているところには，そのソフトウェアのバージョンの数字が入る．
枠で囲んであるコマンドは必ず入力するものである．
枠で囲んでいないコマンドは必ずしも入力する必要のないものである．

2.2　必要な環境

2.2.1　gnuplotのインストール

RD-AUTOではバージョン4.0以降を必要とするので，Versionが古い場合Updateしてください．

1. gnuplotのダウンロード先

   以下のサイトから最新のgnuplotのバージョン (`gnuplot-` *version*`.tar.gz`) を入手する．

   `http://www.gnuplot.info/`

2. gnuplotのインストール

   `/usr/local/gnuplot` にインストールする場合を例とする．

   `gnuplot-` *version*`.tar.gz` を展開する．

   ```
   $ tar -zxvf gnuplot- version.tar.gz ↵
   ```

   展開後，`gnuplot-` *version* に移動し

   ```
   $ ./configure -prefix=/usr/local/gnuplot ↵
   $ make ↵
   $ su ↵ (スーパーユーザーのホームに移動した場合は，gnuplot- version まで移動する．)
   # make install ↵
   # exit ↵
   ```

を行う.

3. パス[1] を通す

使用しているシェル[2] の任意の行に

```
setenv PATH ${PATH}:/usr/local/gnuplot/
```

と記述.

ただし，上記は tcsh を使用している場合の例であり，シェルによって書き方は異なる.

自分が使用しているシェルが分からない場合は，以下のコマンドで使用しているシェルが分かる.

$ echo $SHELL ↵

シェルを emacs で開きたいときは，自分のホームで以下のコマンドを使う．(シェルに tcsh を使っている場合)

$ emacs .tcshrc[注 1] ↵

または

$ emacs .tcshrc.mine[注 1] ↵

(注 1) 自分のホームに.tcshrc.mine が存在する場合はそれを編集する.

シェルに記述したら，以下のコマンドを入力する.

```
$ source .tcshrc[注 2] ↵
```

(注 2).tcshrc.mine を編集した場合は$ source .tcshrc.mine ↵

シェルが正しく書かれているか確認するために，コマンドプロンプトを立ち上げなおす.

もし，コマンドプロンプトに図 2.2.1 のようなコメントが出たら，シェルの書き方が間違っている.

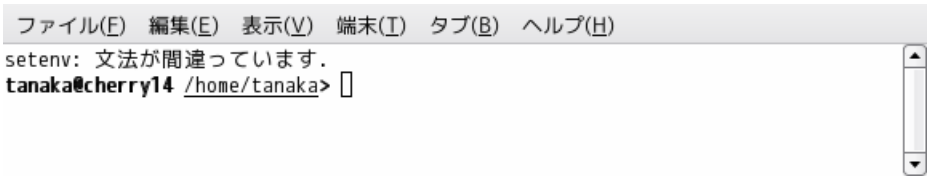


図 2.2.1:

パスが通っているか確認するには以下のコマンドを入力すればよい.

$ which gnuplot ↵

このコマンドで gnuplot の場所が出れば，パスは通っている.

[1]ディレクトリツリーの中のある場所からある場所への経路のこと.
[2]ユーザと OS 間のコマンドのやりとりを仲介する．インタフェイスとしての役割を持つプログラムの総称.

2.2.2 mpich のインストール

1. MPICH のダウンロード

   以下のコマンドを使い，MPICH(mpich.tar.gz) を入手する．

```
$ wget ftp://ftp.mcs.anl.gov/pub/mpi/mpich.tar.gz ↵
```

   このコマンドを使うと，コマンドプロンプトに図 2.2.2 のようなメッセージが出て，ダウンロードできる．

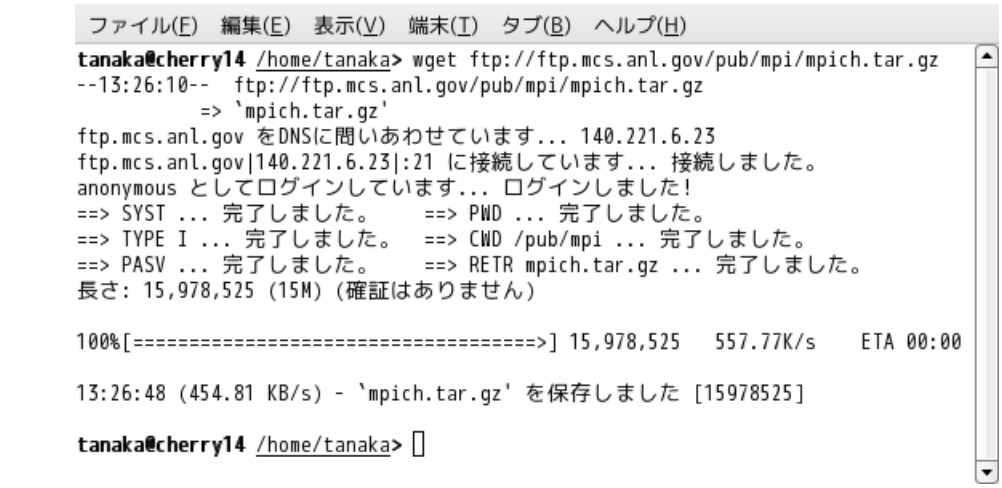


図 2.2.2:

   wget コマンドが使えない場合，インターネットブラウザの URL 欄に

   ftp://ftp.mcs.anl.gov/pub/mpi/mpich.tar.gz

   と入力すればダウンロードできる．

2. MPICH のインストール

   /usr/local/mpich にインストールする場合を例とする．

   mpich.tar.gz を展開する．

```
$ tar -zxvf mpich.tar.gz ↵
```

   展開後，mpich-*version* に移動 (`$ cd mpich-`*version* ↵) し，次のコマンドでインストールを行う．

```
$ ./configure -fc=/opt/***[注3]/ifort -f90=/opt/***[注3]/ifort
  -cc=/opt/***[注3]/icc -rsh=ssh -prefix=/usr/local/mpich ↵
$ make ↵
$ su ↵ (スーパーユーザーのホームに移動した場合は，mpich-version まで移動する．)
# make install ↵
# exit ↵
```

(注 3)./configure の***の所には，ifort と icc のフルパス[3] を入力する．フルパスが分からない場合は，以下のコマンドで調べる．

$ which ifort ↵

$ which icc ↵

Mac OSX の場合

```
$ ./configure --prefix=/opt/local CPP='icc -E' CC=icc CXX=icpc
   FC=ifort F77=ifort ↵
```

とする．

3. パスを通す

使用しているシェルの任意の場所に

```
setenv PATH ${PATH}:/usr/local/mpich/bin
```

と記述．

ただし，上記は tcsh を使用している場合の例であり，シェルによって書き方は異なる．

自分が使用しているシェルが分からない場合は，以下のコマンドで使用しているシェルが分かる．

$ echo $SHELL ↵

シェルを emacs で開きたいときは，自分のホームで以下のコマンドを使う．(シェルに tcsh を使っている場合)

$ emacs .tcshrc[注 4] ↵

または

$ emacs .tcshrc.mine[注 4] ↵

(注 4) 自分のホームに.tcshrc.mine が存在する場合はそれを編集する．

シェルに記述したら，以下のコマンドを入力する．

```
$ source .tcshrc[注 5] ↵
```

(注 5).tcshrc.mine を編集した場合は$ source .tcshrc.mine ↵

シェルが正しく書かれているか確認するために，コマンドプロンプトを立ち上げなおす．

もし，コマンドプロンプトに図 2.2.1 のようなコメントが出たら，シェルの書き方が間違っている．

[3]ルートディレクトリ (/) を基点として，ファイルの場所を表わす文字列．

パスが通っているか確認するには，以下のコマンドを入力する．

```
$ which mpicc ↵
```

2.2.3 RSA 鍵を使った mpi のプロセス通信設定

MPICH の通信に ssh を使用すると，プログラム実行のたびにパスワードを入力しなければならなくなる．そのため，以下の操作によりそれを省略する．マルチコア[4] のマシンを使用する場合，サーバーのマシン自体の RSA 鍵[5] も登録する必要がある．

1. 使用するマシンの RSA 鍵を生成する

   自分のホームで次のコマンドを入力する．

```
$ ssh-keygen -t rsa ↵
Generating public/private rsa key pair.
Enter file in which to save the key (/home/***/.ssh/id_rsa): ↵
Enter passphrase (empty for no passphrase): ↵
Enter same passphrase again: ↵
Your identification has been saved in /home/***/.ssh/id_rsa.
Your public key has been saved in /home/***/.ssh/id_rsa.pub.
The key fingerprint is:
**:**:**:**:**:**:**:**:**:**:**:**:**:** ***@***
```

   ~/.ssh に id_rsa.pub が生成される．

2. サーバーとなるマシンに鍵を登録する

   1 で生成した id_rsa.pub をサーバーとなるマシンに登録する．

   自分のホームに移動 (cd $HOME ↵) する．

   .ssh の中に authorized_keys が存在するか確認する．($ ls .ssh/ ↵)

   ~/.ssh/authorized_keys が存在しない場合

```
$ cat .ssh/id_rsa.pub > .ssh/authorized_keys ↵
```

   ~/.ssh/authorized_keys が存在する場合

```
$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys ↵
```

   同様にして，使用する全てのマシンの RSA 鍵をサーバーとなるマシンに追加登録する．

3. 所有者に対してのみ読み込みと書き込みの権限を与える

```
$ chmod 600 .ssh/authorized_keys ↵
```

[4]複数のプロセッサコアを 1 つのパッケージに集約したマイクロプロセッサ．
[5]公開鍵暗号の 1 つ

2.3 RD-AUTO の解凍

1. RD-AUTO(分岐計算を行うソフトウエア) の入手先
   長山研究室の〜から RD.tgz をダウンロードする．
2. 展開
   適当なディレクトリで RD.tgz を展開する．展開するコマンドは

```
$ tar -zxvf RD.tgz ↵
```

   と実行する．このとき，RD.tgz は図 2.3.1 のように展開する．

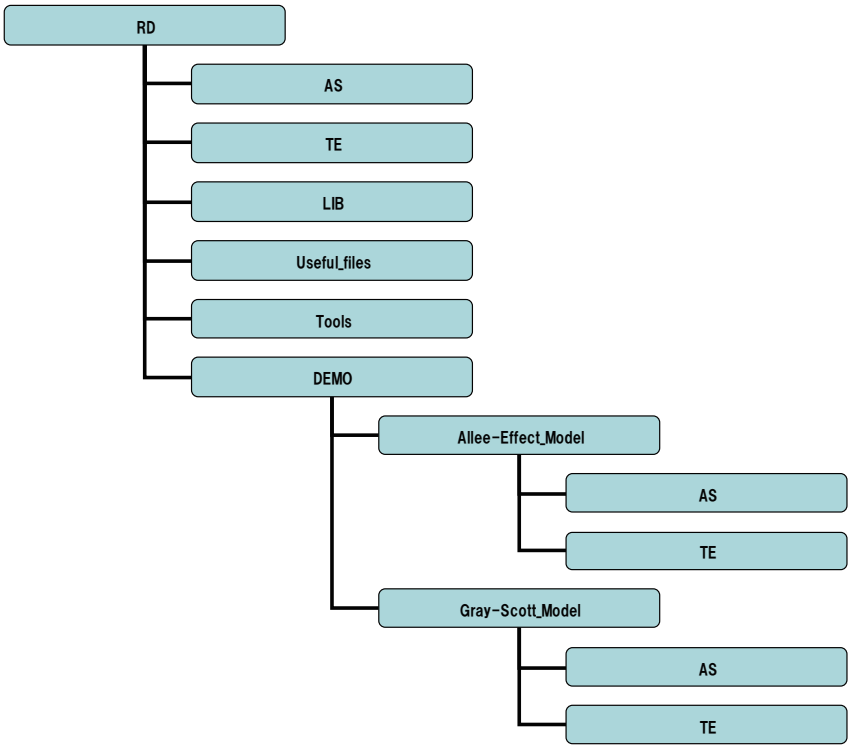


図 2.3.1: 展開図

RD/AS は分岐計算を行う RD-AUTO のディレクトリである．RD/TE は時間発展方程式を計算するディレクトリである．RD/LIB と RD/Useful_Files は後述するスクリプトファイル (datacut , extremum など) のソースファイルが入っている．RD/Tools には mpich などが入っている．RD/DEMO は本マニュアルで説明する分岐計算などを行うデモプログラムが入っている．

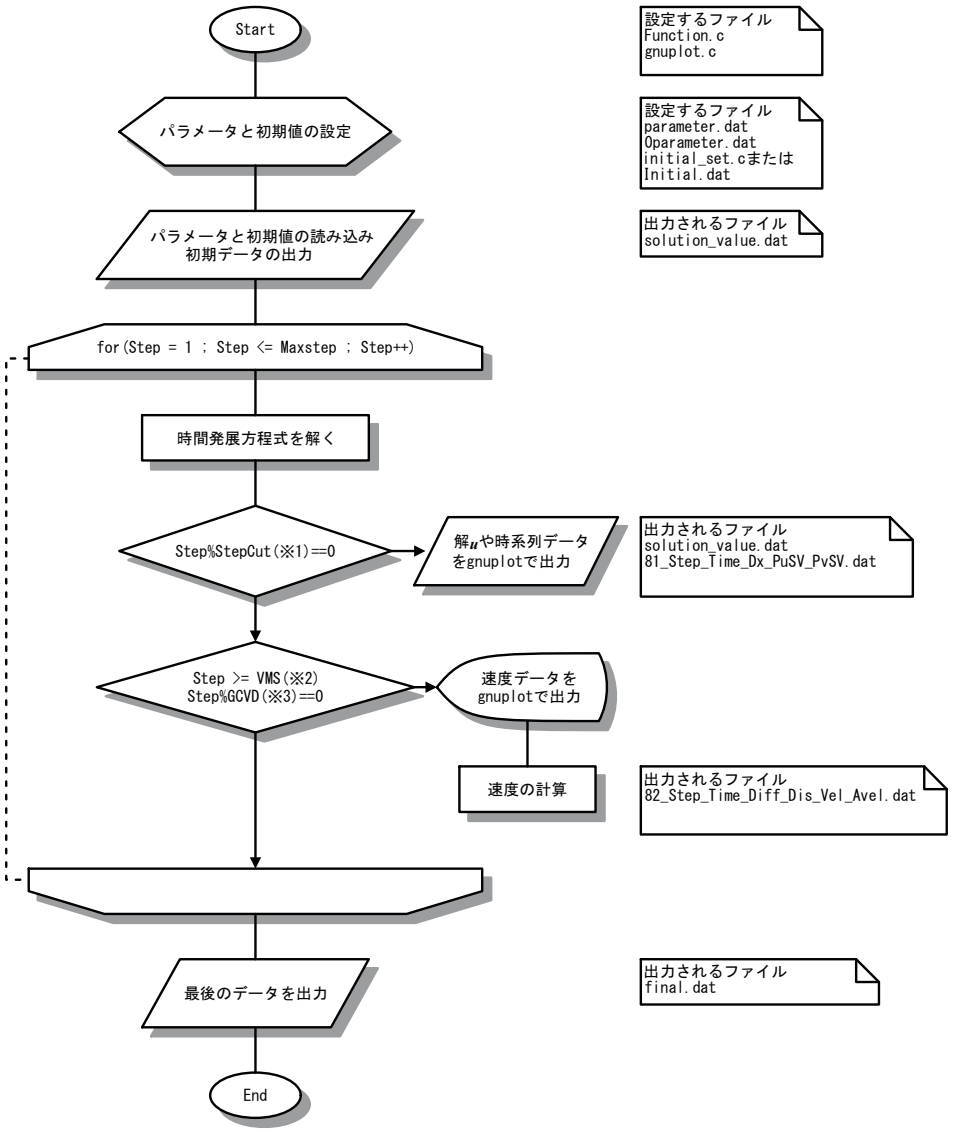
2.3.1 Mac OSX を使って RD-AUTO を使う方法

Mac 上で RD-AUTO を使うためには概ね上記の方法で動かせることができる．しかし，MPI を使って計算することから NotePC 等で固定 IP アドレスを使わないで Local.host として MPI を使う場合は注意が必要である．mpichi のインストールでは Mac Ports を使ってインストールした場合，gfortran を認識するれば問題ないが，gfortran を認識しない場合は，上記の方法で mpich をインストールしてください．

第3章　発展方程式

3.1　使い方

3.1.1　時間発展方程式のプログラムに対するフローチャート



※ 1 StepCut については parameter.dat の StepCut の項目を参照 (→ 14 ページ).
※ 2 VMS については parameter.dat の Velocity_Measure_Start の項目を参照 (→ 13 ページ).
※ 3 GCVD については parameter.dat の Gnuplot_CutN_Vel_Dis の項目を参照 (→ 13 ページ).

3.1.2 実行前の設定

ここでは，次の Gray-Scott モデルを例に時間発展方程式を解くための設定を行う．

$$\begin{cases} u_t = \Delta u + \dfrac{1}{\varepsilon}(-au + u^2 v), \quad (*1) & t > 0, \quad 0 < x < 2\pi, \\ v_t = d\Delta v + h(v_c - v) - u^2 v, \end{cases}$$

ただし，次のようにスケーリングを行って計算するものとする．

$$\begin{cases} u_t = \dfrac{4\pi^2}{L^2}\Delta u + \dfrac{1}{\varepsilon}(-au + u^2 v), \quad (*1) & t > 0, \quad 0 < x < 2\pi, \\ v_t = d\dfrac{4\pi^2}{L^2}\Delta v + h(v_c - v) - u^2 v, \end{cases}$$

設定に必要なファイルは次の 5 つである:

1. `0parameter.dat`
2. `Function.c`
3. `parameter.dat`
4. `boundary.dat`
5. `initial_set.c`

1. `0parameter.dat`

   反応項及び拡散項で使われるパラメーターの設定．

```
 1.0   ;  du   ; par[1]  方程式 u の拡散係数を第 1 行に記入
 3.0   ;  dv   ; par[2]  方程式 v の拡散係数を第 2 行に記入
 1.0   ;  eps  ; par[3]
 0.07  ;  a    ; par[4]
 1.0   ;  vc   ; par[5]
0.018  ;  h    ; par[6]
 500   ;  Lx   ; par[7]
```

   3 以降は Gray-Scott モデル (∗1) に現れるパラメータ (パラメータの順番は自由)．拡散係数は最初に入れなければならないので，例えば u, v, w の 3 変数の場合には第 3 行にパラメータ dw を入れる．

2. `Function.c`

   反応項 (関数 `Reaction`) 及び拡散項 (関数 `Diffusion`) の設定．RD-AUTO は反応拡散系の空間離散化に対して差分法と Spectral 法のいづれかを用いており，Spectral 法を選択して数値計算する場合は，$y = \frac{2\pi}{L}x$ と変数変換を行った次の方程式を解くように設定する必要がある．

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{4\pi^2}{L^2} D \frac{\partial^2 \boldsymbol{u}}{\partial y^2} + \boldsymbol{f}(\boldsymbol{u}), \quad t > 0, \quad y \in [0, 2\pi]$$

```
#include <stdio.h>
  ...........

void  Reaction( TEMP TP, double *par, double *x, double *F )
{
  int i;
  int N = TP.N;
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  for( i = 1; i <= N; i++ ){
              /* ↓ u の反応項 f(u) を記述 ↓ */
    F[i  ]  = (-par[4]*x[i]+x[i]*x[i]*x[i+N]) / par[3];
              /* ↓ v の反応項 g(u) を記述 ↓ */
    F[i+N]  = par[6]*(par[5]-x[i+N]) - x[i]*x[i]*x[i+N];
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  }
}

void  Diffusion( TEMP TP, double *par )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
              /* ↓ 変数変換を行う記述 ↓ */
  par[TP.N_EP+1]  =  (4.0*M_PI*M_PI*par[1]) / (par[7]*par[7]);
  par[TP.N_EP+2]  =  (4.0*M_PI*M_PI*par[2]) / (par[7]*par[7]);
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

3. parameter.dat

```
         256   ;  N                          ;  空間分割数
         128   ;  N_Wave                     ;  切断波数
          -1   ;  T_SELECT                   ;  ③
          -1   ;  VEL_SELECT                 ;  ④
     1.0E-05   ;  dt                         ;  時間分割幅
6.283185307179586;  Lx                       ;  ⑥
         100   ;  Gnuplot_CutN_Vel_Dis       ;  ⑦
        0.20   ;  Velocity_Site              ;  ⑧
      200000   ;  Velocity_Measure_Start    ;  ⑨
           3   ;  BC_Select                  ;  ⑩
           1   ;  EE_Select                  ;  ⑪
        5000   ;  StepCut                    ;  ⑫
      300000   ;  MaxStep                    ;  最大ステップ数
          12   ;  Gnuplot_Select             ;  ⑭
           2   ;  Initial_Select             ;  ⑮
```

説明のないものは下記参照.

③ T_SELECT

T_SELECT の値に応じて，プログラムでは時間発展方程式の与え方が異なる．T_SELECT = 1 のとき，次の方程式を解く:

$$\frac{\partial \boldsymbol{u}}{\partial t} = T\left(D\frac{\partial^2 \boldsymbol{u}}{\partial x^2} + \boldsymbol{f}(\boldsymbol{u})\right),$$

ただし，T は周期 (Initial.dat から読み込む)．T_SELECT $\neq 1$ のとき，次の方程式を解く:

$$\frac{\partial \boldsymbol{u}}{\partial t} = D\frac{\partial^2 \boldsymbol{u}}{\partial x^2} + \boldsymbol{f}(\boldsymbol{u}).$$

④ VEL_SELECT

VEL_SELECT の値に応じて，プログラムでは時間発展方程式の与え方が異なる．VEL_SELECT = 1 のとき，次の方程式を解く;

$$\frac{\partial \boldsymbol{u}}{\partial t} = D\frac{\partial^2 \boldsymbol{u}}{\partial x^2} + c\frac{\partial \boldsymbol{u}}{\partial x} + \boldsymbol{f}(\boldsymbol{u}),$$

ただし，c は速度 (Initial.dat から読み込む)．VEL_SELECT $\neq 1$ のとき，次の方程式を解く;

$$\frac{\partial \boldsymbol{u}}{\partial t} = D\frac{\partial^2 \boldsymbol{u}}{\partial x^2} + \boldsymbol{f}(\boldsymbol{u}).$$

T_SELECT と VEL_SELECT の両方が 1 のとき，次の方程式を解く:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{1}{T}\left(D\frac{\partial^2 \boldsymbol{u}}{\partial x^2} + c\frac{\partial \boldsymbol{u}}{\partial x} + \boldsymbol{f}(\boldsymbol{u})\right).$$

⑥ Lx

区間 x．ただし，Spectral 法を用いて計算する場合は 2π を指定する．

⑦ Gnuplot_CutN_Vel_Dis

Gnuplot_CutN_Vel_Dis = n のとき，n ステップ[1]おきに速度データをファイル 82_Step_Time_Diff_Dis_Vel_Avel.dat に出力する．

⑧ Velocity_Site

Velocity_Site は解の速度計算に必要とする．解 $\boldsymbol{u}$ の高さの位置を指定する (図 3.1.1).

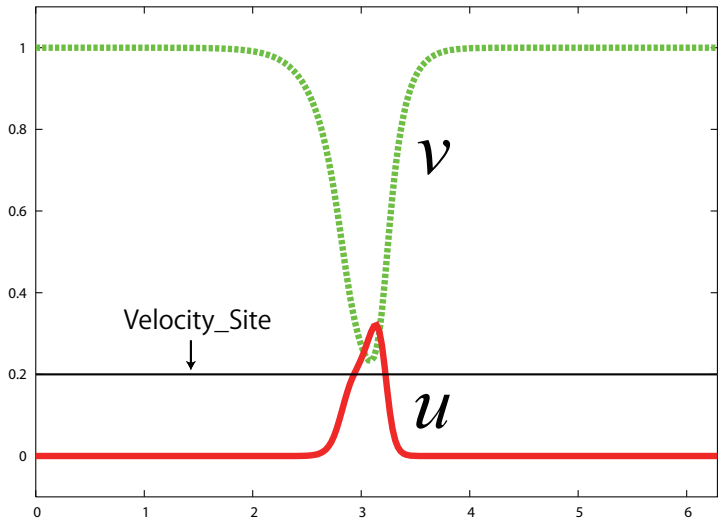


図 3.1.1: Velocity_Site

⑨ Velocity_Measure_Start

解の速度を測り始める Step 数を指定する．

⑩ BC_Select

境界条件の選択

1:Dirichlet 境界条件

2:Neumann 境界条件

3:周期境界条件

⑪ EE_Select

離散スキームの選択

1:差分法 (陰解法)

2:Spectral 法 (Euler 法)

3:Spectral 法 (Runge-Kutta 法)

ただし，Spectral 法の場合は境界条件を周期境界条件として与えること．

⑫ StepCut

StepCut = n のとき，n ステップおきにデータ (solution_value.dat (21 ページ), 81_Step_Time_Sup_L1_L2.dat (21 ページ)) を出力する．

[1]ステップの定義はフローチャートの step のこと

⓮ Gnuplot_Select

時間発展方程式を解きながら計算途中の解情報を gnuplot を用いて表示する．デフォルトでは 1 ~ 6 の gnuplot 画面の設定がされている．Gnuplot_Select = 0 と指定したとき，6 つの gnuplot 画面が表示され，Gnuplot_Select = $n_1n_2n_3\ldots$ と指定すると $n_1, n_2, n_3, \ldots$ で与えた数字の gnuplot 画面が表示されない．ただし，$n_1, n_2, n_3, \ldots$ は 1 ~ 6 の値である．例えば Gnuplot_Select = 12 のとき 3,4,5,6 の gnuplot 画面が表示される．gnuplot 画面 1 ~ 6 の意味は次の通りである．

1:初期値の出力

2:u(赤)，v(青) の出力

3:u だけ出力

4:Velocity_Site で指定した値を用いてパルス $\boldsymbol{u}$ のフロントとバックの距離 d を出力 (Velocity_Site = 0.2 で与えたときの図 1 を参照).

5:速度 (赤)，平均速度 (青) の出力

6:$\|\boldsymbol{u}\|_\infty$ を時系列データとして出力 ($\|\boldsymbol{u}\|_{L^1}$，$\|\boldsymbol{u}\|_{L^2}$ を選択することも可能．)

ただし，$\boldsymbol{u}(x_i) = u(i)$ として

$$\|\boldsymbol{u}\|_\infty := \max(|u(1)|, \ldots, |u(N)|)$$
$$\|\boldsymbol{u}\|_{L^1} := |u(1)| + \cdots + |u(N)|$$
$$\|\boldsymbol{u}\|_{L^2} := \sqrt{u(1)^2 + \cdots + u(N)^2}$$

gnuplot.c の項目 (→ 22 ページ) も参照のこと．

⓯ Initial_Select

初期値と使用するパラメータの選択

1:初期値 $\boldsymbol{u}_0(x)$ は initial_set.c (16 ページ) で設定したものを用いる．パラメータは 0parameter.dat のものを用いる．

2:初期値 $\boldsymbol{u}_0(x)$ は Initial.dat の 5 列目に入っているものを用いる．パラメータは Initial.dat の 3 列目に入っているものを用いる．

3:初期値 $\boldsymbol{u}_0(x)$ は Initial.dat の 5 列目に入っているものを用いる．パラメータは 0parameter.dat のものを用いる．

4. boundary.dat

```
0.000000000;  BC_000_U  ; BC[1]
0.000000000;  BC_N+1_U  ; BC[2]
1.000000000;  BC_000_V  ; BC[3]
1.000000000;  BC_N+1_V  ; BC[4]
```

境界の値．周期境界条件の場合は値を使うことはないが，変数の個数はここで判定しているために変数の個数 ×2 個の境界条件は設定しておかなければならない．

- Dirichlet 境界条件のとき

  $u(t,0)$ =BC[1]，$u(t,L)$ =BC[2]，$v(t,0)$ =BC[3]，$v(t,L)$ =BC[4].

- Neumann 境界条件のとき

  $u_x(t,0)$ =BC[1]，$u_x(t,L)$ =BC[2]，$v_x(t,0)$ =BC[3]，$v_x(t,L)$ =BC[4].

- 周期境界条件のとき
  $u(t,0) = u(t,L), \quad v(t,0) = v(t,L).$

5. initial_set.c

```
#include <stdio.h>
  ..........

void  initial_set2( TEMP *TP, double *par, double *BC, double *x )
{
  ..........
  if( IS  ==  1 ){
    ..........              /*ここに初期値を記述する*/
  }
  else if( IS == 2  ||  IS == 3 ){
    ..........
  }else{
    ..........
  }
  ..........
}

  ..........
```

初期値 $\boldsymbol{u}_0(x), \boldsymbol{v}_0(x)$ の設定は関数 initial_set2 内で行う．例えば図 3.1.2 のステップ関数は次のようになる．

```
  if( IS  ==  1 ){
    for( i = 1; i <= N; i++ ){
      x[i  ]  =  0.000000000;
      x[i+N]  =  1.000000000;
      if( N/10 <= i  &&  N/5>= i )
        x[i]  =  0.4;
      if( 1 <= i  &&  N/10>= i )
        x[i+N]  =  0.4;
    }
  }
```

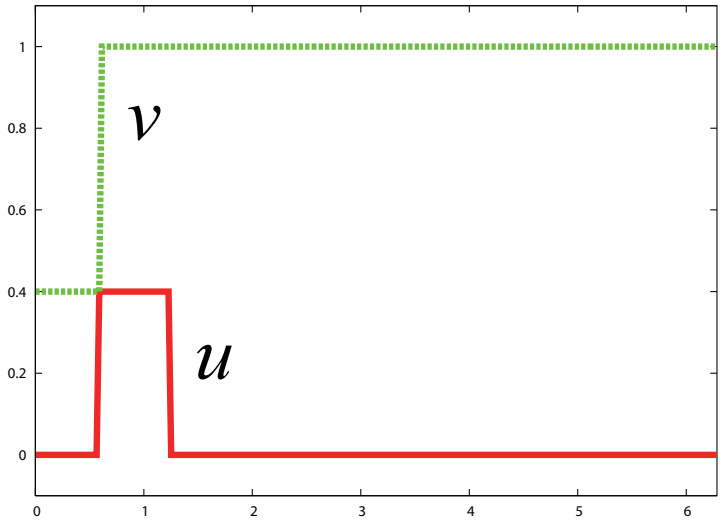


図 3.1.2: ステップ関数

3.1.3 実行コマンドの説明

$ ./aout ↵ で時間発展方程式のプログラムを実行する．

aout はコンパイルを行い，コンパイルが成功したらプログラムを実行するまでのスクリプトである．aout は次のコマンドを行う．

```
$ make ↵
$ time␣./a.out ↵
```

aout のオプションについて

- $ ./aout␣f ↵ のコマンドを実行すると，

  ```
  $ cp␣final.dat␣Initial.dat ↵
  $ ./aout ↵
  ```

  と同等のことを行う．

- $ ./aout␣1 ↵ のコマンドを実行すると，オブジェクトファイルを再構成して実行する．すなわち，

  ```
  $ rm␣-rf␣ZZZZ_OBJ ↵
  $ make ↵
  $time␣./a.out ↵
  ```

  と同等のことを行う．ただし，ZZZZ_OBJ はオブジェクトディレクトリのことで，全てのオブジェクトファイルはこのディレクトリの中にある．

- $ ./aout␣*directryname* ↵

  ディレクトリ (*directryname*) を作り，その中に

  0parameter.dat

81_Step_Time_Sup_L1_L2.dat
82_Step_Time_Diff_Dis_Vel_Avel.dat
Initial.dat
boundary.dat
final.dat
parameter.dat
solution_value.dat

をコピーする.

- $ ./extremum ↵

  $\|\boldsymbol{u}\|_{\infty}, \|\boldsymbol{u}\|_{L^1}, \|\boldsymbol{u}\|_{L^2}$ の時系列データ (81_Step_Time_Sup_L1_L2.dat) の極大値 (もしくは極小値) を求め，周期解の周期を推測するときに用いるスクリプトファイル (→ 23 ページ).

- $ ./datacut ↵

  solution_value.dat を用いて時間発展方程式の解を描写する (→ 26 ページ).

3.1.4 出力されるデータファイル

1. final.dat(Initial.dat)

行数	Step	パラメータ 1	パラメータ 2	方向ベクトル	解の値
1	MaxStep	N	par[1]	0.0	x[1] = u[1]
2	⋮	N_EP	par[2]	⋮	⋮
3		Lx	par[3]		
4		dt	⋮		
5		NM			
6		PF	⋮		
7		EES	par[n]		
8		NW	0.0		
9		ME	⋮		
10		REV_i_E			
11		0.0			
⋮		⋮			
19		0.0			
20		N_VAR			
21		BC[1]			
22		BC[2]			
23		BC[3]			
24		BC[4]			
25		c_MP			
26		0.0			
⋮		⋮			⋮
Ns					x[Ns] = u[Ns]
$Ns+1$					x[Ns+1] = v[1]
⋮					⋮
$2Ns$					x[2Ns] = v[Ns]
$2Ns+1$					T
$2Ns+2$	⋮	⋮	⋮	⋮	c
$2Ns+3$	MaxStep	0.0	0.0	0.0	λ

- 1 列目
  最大ステップ数
- 2 列目
  - N
    空間分割数
  - N_EP
    パラメータの個数
  - Lx
    区間

- dt
  時間分割幅
- NM
  時間発展方程式では使用しない (分岐図での方で用いる)
- PF
  時間発展方程式では使用しない (分岐図での方で用いる)
- EES
  離散スキームの選択
  1:差分法
  2:スペクトル法 (Euler)
  3:スペクトル法 (Runge-Kutta)
- NW
  スペクトル法を用いて計算する場合の切断波数 (差分法では意味のない値)
- ME
  時間発展方程式では使用しない (分岐図での方で用いる)
- REV_i_E
  時間発展方程式では使用しない (分岐図での方で用いる)
- N_VAR
  変数の個数
- BC[1] ~ BC[4]
  境界の値 (→ 15 ページ)
- c_MP
  時間発展方程式では使用しない (分岐図での方で用いる)

- 3 列目
  パラメータ par[1]~par[N_EP] の値．ただし，N_EP は設定したパラメータの個数．
- 4 列目
  分岐計算の場合は方向ベクトルが入る．(発展方程式の場合は 0 の値が入る)
- 5 列目
  上から順に，
  - 解の値 (u[1]...u[Ns],v[1]...v[Ns])
  - T …周期
  - c …解の平均速度
  - λ …コントロールパラメータ

  の値が入る．

2. `81_Step_Time_Sup_L1_L2.dat`

1 列目	2 列目	3 列目	4 列目	5 列目
ステップ数	時間	$\|\|\boldsymbol{u}\|\|_{\infty}$	$\|\|\boldsymbol{u}\|\|_{L^1}$	$\|\|\boldsymbol{u}\|\|_{L^2}$
⋮	⋮	⋮	⋮	⋮

3. `82_Step_Time_Diff_Dis_Vel_Avel.dat`

1 列目	2 列目	3 列目	4 列目	5 列目	6 列目
ステップ数	時間	※	あとから	解の速度	解の平均速度
⋮	⋮	⋮	⋮	⋮	⋮

3 列目の ※ は `Velocity_Site` (→ 13 ページ) の高さでのパルスのフロントとバックの距離 d(図 3.1.3)

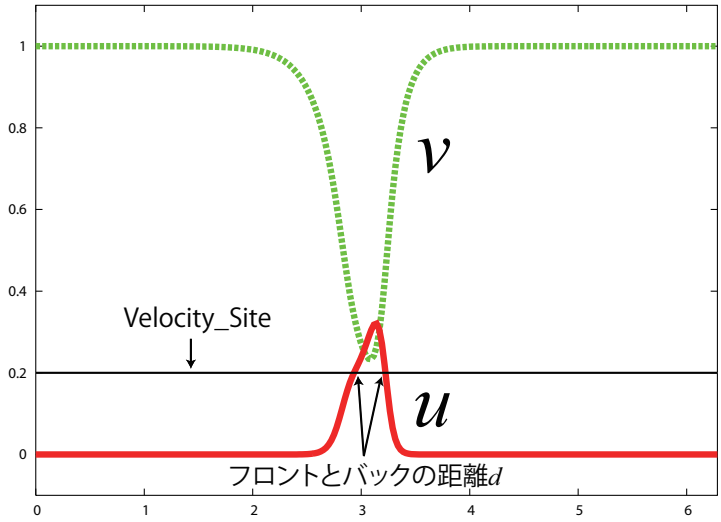


図 3.1.3: *distance*

4. `solution_value.dat`

時間 t	位置 x	解 $\boldsymbol{u}$ の値	解 $\boldsymbol{v}$ の値
t_1	x_1	`u[1]` の値	`v[1]` の値
⋮	x_2	`u[2]` の値	`v[2]` の値
⋮	⋮	⋮	⋮
t_1	x_{N_s}	`u[Ns]` の値	`v[Ns]` の値
t_2	x_1	`u[1]` の値	`v[1]` の値
⋮	x_2	`u[2]` の値	`v[2]` の値
⋮	⋮	⋮	⋮
t_2	x_{N_s}	`u[Ns]` の値	`v[Ns]` の値
⋮	⋮	⋮	⋮

3.1.5 gnuplot の設定

1. gnuplot.c

```
#include <stdio.h>
  ..........

void  Gnuplot_InitialCommand( TEMP *TP, int *c_G )
{
  ..........
/*- - - - ↓解の描写範囲を指定する↓- - - - - - - - - - - -*/
  fprintf( GP2, "set yr[-0.1:1.1]\n" );
  fprintf( GP3, "set yr[-0.1:1.1]\n" );
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
  ..........
}

void  Gnuplot_Draw( TEMP TP, int *c_G )
{
  ..........
}

void  Gnuplot_Close( TEMP *TP )
{
  ..........
}

void  Gnuplot_DrawCommand( TEMP TP, FILE *GP, int SELECT )
{
  ..........
  switch( SELECT ){
    ..........
    case 6:
      /*↓u 2:3 だと sup, u 2:4 だと L1, u 2:5 だと L2 の norm を出力する*/
      fprintf( GP, "plot '%s' u 2:3 pt 7 ps 2.0 lt 1\n", G3 );
    ..........
  }
  ..........
}
```

関数 Gnuplot_InitialCommand では gnuplot の初期設定を行う．解の描写範囲もここで指定する．

関数 Gnuplot_Draw では関数 Gnuplot_DrawCommand を呼び出し gnuplot による画面出力を行う．

関数 Gnuplot_Close では gnuplot を閉じる．

関数 Gnuplot_DrawCommand では各 gnuplot 画面にどのような出力表示を与えるか指定する.

3.1.6 周期解の周期の求め方

ここでは 81_Step_Time_Sup_L1_L2.dat の 5 列目のデータ $\|\boldsymbol{u}(\hat{T})\|_{L^2}$ を用いて周期を近似的に求める方法を説明する．具体的な使い方は，第 4 章 2 節の Appendix1.1(→ 88 ページ) で説明する．$\|\boldsymbol{u}\|_{L^2}$ の時系列データが次の図 3.1.4 のように得られたとき，extremum を実行すると図 3.1.5 の時系列の極大値 $\|\boldsymbol{u}(T_1)\|_{L^2}, \|\boldsymbol{u}(T_2)\|_{L^2}, \ldots, \|\boldsymbol{u}(T_n)\|_{L^2}$ が求められ，$T_{i+1} - T_i$ とその平均の値 $T_{ave} = \frac{T_n - T_1}{n-1}$ も求められる．ただし n は極大値の個数．$\|\boldsymbol{u}(T,x)\|_{L^2} := \sqrt{\int_{-L}^{L} u^2 dx}$．同様に極小値も求められる．

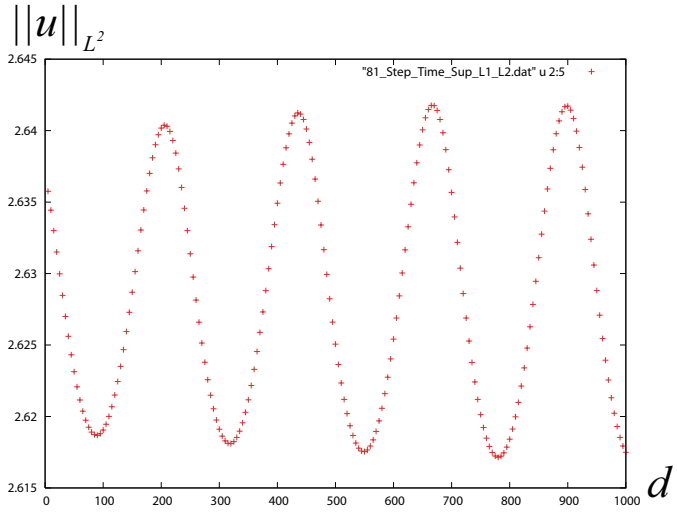


図 3.1.4:

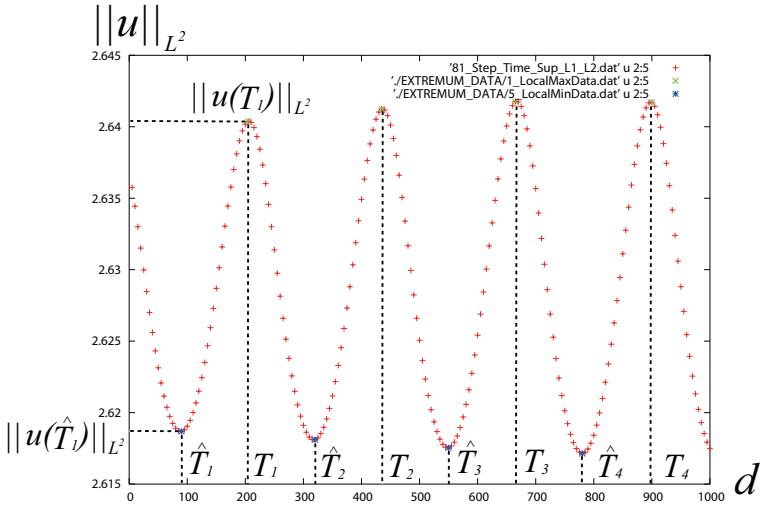


図 3.1.5:

$ ./extremum ↵ と実行する．

解の極大値もしくは極小値の間隔と周期，及びその平均値が求められ，ディレクトリ EXTREMUM_DATA にファイルに出力される．出力されるファイルについては後述．ここで求めた周期を使って周期解の分岐計算を行う．

- extremum の設定について

```
#!/bin/sh

#-------------------------------------------------------------------
  Index_Comparison_Data=5
  Index_Period_Data=2
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  N_Comparison_Max=10
  N_Comparison_Min=10
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Gnuplot_Command0="set term dumb"
  Gnuplot_Command="u 2:5"
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  ............
```

  - Index_Comparison_Data

    81_Step_Time_Sup_L1_L2.dat の norm データを用いて周期を求める．

    例 Index_Comparison_Data=5 の場合，81_Step_Time_Sup_L1_L2.dat の左から 5 列目のデータ (つまり $\|\boldsymbol{u}\|_{L^2}$) を使って周期を求める．

  - N_Comparison_Max，N_Comparison_Min

    時系列データの点を順番に $y_1, y_2, \ldots$ と表記すると，y_i が極大値であるか調べるには $y_{i-l}, \ldots, y_{i-1} < y_i$ かつ $y_i > y_{i+1}, \ldots, y_{i+l}$ となればよい．ただし，l は正の定数で l の値を N_Comparison_Max に指定する．同様に，y_i が極小値であるか調べるには $y_{i-r}, \ldots, y_{i-1} < y_i$ かつ $y_i > y_{i+1}, \ldots, y_{i+r}$ となればよいので，r の値を N_Comparison_Min に指定する．

    例 N_Comparison_Min が 10 だったとすると，ある点の 10 個前と 10 個後全てがその点より大きかった場合，その点を極小値とする (図 3.1.6).

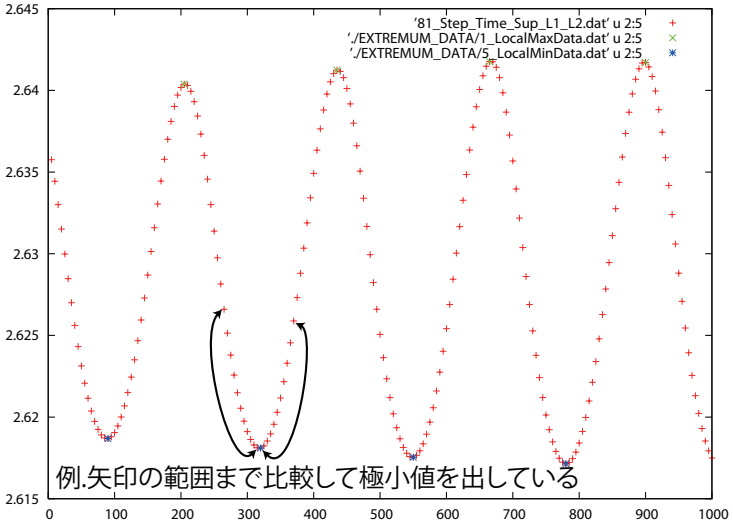


図 3.1.6:

- Gnuplot_Command

  Gnuplot_Command は"u n:m"と与える．ただし，n と m は n = Index_Period_Data，m = Index_Comparison_Data である．

- 出力されるファイルについて

  実行後、ディレクトリ EXTREMUM_DATA が作成される．

  - 1_LocalMaxData.dat

    すべての極大値の値 $(\|\boldsymbol{u}(T_1)\|_{L^2}, \|\boldsymbol{u}(T_2)\|_{L^2}, \ldots, \|\boldsymbol{u}(T_n)\|_{L^2})$
  - 2_LocalMaxPeriod.dat

    すべての極大値の時間の間隔 $(T_2 - T_1, T_3 - T_2, \ldots, T_n - T_{n-1})$
  - 3_LocalMaxAveData.dat

    極大値の平均値 $(\dfrac{\|\boldsymbol{u}(T_1)\|_{L^2} + \|\boldsymbol{u}(T_2)\|_{L^2} + \cdots + \|\boldsymbol{u}(T_n)\|_{L^2}}{n})$
  - 4_LocalMaxAvePeriod.dat

    極大値の時間の間隔の平均値 $(T_{ave} = \dfrac{T_n - T_1}{n-1})$
  - 5_LocalMinData.dat

    すべての極小値の値 $(\|\boldsymbol{u}(\hat{T}_1)\|_{L^2}, \|\boldsymbol{u}(\hat{T}_2)\|_{L^2}, \ldots, \|\boldsymbol{u}(\hat{T}_n)\|_{L^2})$
  - 6_LocalMinPeriod.dat

    すべての極小値の時間の間隔 $(\hat{T}_2 - \hat{T}_1, \hat{T}_3 - \hat{T}_2, \ldots, \hat{T}_n - \hat{T}_{n-1})$
  - 7_LocalMinAveData.dat

    極小値の平均値 $(\dfrac{\|\boldsymbol{u}(\hat{T}_1)\|_{L^2} + \|\boldsymbol{u}(\hat{T}_2)\|_{L^2} + \cdots + \|\boldsymbol{u}(\hat{T}_n)\|_{L^2}}{n})$
  - 8_LocalMinAvePeriod.dat

    極小値の時間の間隔の平均値 $(\hat{T}_{ave} = \dfrac{\hat{T}_n - \hat{T}_1}{n-1})$

3.1.7 データカット

solution_value.dat のデータを用いて解 $\boldsymbol{u}(x)$ の描写を行うとき，
$ ./datacut ↵
と実行する．

- 設定

```
#!/usr/bin/perl -w

#-----------------------------------------------------------------
  $InitialSelect   =  3;
  $OutPut_NumData  =  1;
  $sleep           =  0.0;
  $command0        =  "set yr[-0.1:1.1]";
  $command11       =  "";
  $command12       =  "u 1:2 w l, u 1:3 w l";
  $command13       =  "";
  $command10       =  "plot";
  $DivsionPoint    =  1;
  $DeletingData    =  "1";
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  ............
```

  - デフォルトのオプション指定

    $InitialSelect の値によって描画の表示が変化する．

    $InitialSelect = 1 ならば，Return Key を押すことで 1 枚ずつ描写する．

    $InitialSelect = 2 ならば，Return Key を押すことで最初から最後まで自動で描写する．描写が終わったら，もう一度 Return Key を押すことで終了する．

    $InitialSelect = 3 ならば，実行すると最初から最後まで自動で描写する．描写が終わったら自動的に終了する．

  - 縦軸の描写範囲を変えたい場合

    $command0 = "set yr[-0.1:1.1]"; の数字を変える．

- 出力されるファイルについて

  実行後、ディレクトリ cut が作成される．

  データは全てその中に入る．

- オプション

  - $ ./datacut␣i1 ↵

    Return Key を押すことで 1 枚ずつ描写する ($InitialSelect = 1 で実行するのと同じ)．

  - $ ./datacut␣i2 ↵

    Return Key を押すことで最初から最後まで自動で描写する．描写が終わったら，もう一度 Return Key を押すことで終了する ($InitialSelect = 2 で実行するのと同じ)．

- `$ ./datacut␣i3` ↵
  実行すると最初から最後まで自動で描写する．描写が終わったら自動的に終了する (`$InitialSelect` = 3 で実行するのと同じ).
- `$ ./datacut␣o`*n* ↵
  画面を *n* 回置きに表示する.
- `$ ./datacut␣s`*x* ↵
  画面を *x* 秒置きに表示する (`$InitialSelect` = 2, 3 のときのみ).

3.2 付録

1. プログラムで扱う変数について

   (a) N と Ns の定義

   N の値は境界条件によって異なる．空間分割数を $\hat{N}$ と与えたすると，

   Dirichlet 境界条件のとき，$N = \hat{N} - 1$

   Neumann 境界条件のとき，$N = \hat{N} + 1$

   周期境界条件のとき，$N = \hat{N}$

   となる．また，$Ns = \hat{N} + 1$

   (b) `x` と `t_x` の定義

   区間 L を間隔 $\Delta x = \frac{L}{N}$ で離散化して，区間の格子点を $x_j = (j-1)\Delta x, \quad (1, \ldots, N+1)$ と表す．このとき，定常解 $u(x)$ の格子点上の $u(x_i)$ の値を配列 `u[i]` に保存し，周期解の場合は $u(0, x_i)$ の格子上の値を `u[i]` に保存する．

   v についても同様.

   配列 `x` の $1 \sim N$ には `u[i]` の値が，$N+1 \sim 2N$ には `v[i]` の値が入っている，配列 `t_x` の $1 \sim Ns$ には `u[i]` の値が，$Ns+1 \sim 2Ns$ には `v[i]` の値が入っている:

$$\texttt{x} = \begin{pmatrix} \texttt{u[1]} \\ \vdots \\ \texttt{u[N]} \\ \texttt{v[1]} \\ \vdots \\ \texttt{v[N]} \end{pmatrix}, \quad \texttt{t\_x} = \begin{pmatrix} \texttt{u[1]} \\ \vdots \\ \texttt{u[Ns]} \\ \texttt{v[1]} \\ \vdots \\ \texttt{v[Ns]} \end{pmatrix}$$

第4章　分岐計算プログラム

4.1　使い方

4.1.1　実行前の設定

1. aout

   RD-AUTO は並列計算 (MPICH) を用いるため，最初に並列計算を行うための設定を行う．MPICH で使用するプロセス数を変更するには，AS/aout を開き，$NumProcs の数を変えることで変更することができる．次の例は，4CPU を持つ cherry09 のマシン環境での設定である．使用するマシンの設定は$HeadName_Machine と@i_Machine を組み合わせて行い，$MachineFileName で指定されたファイルに使用するマシン名が出力される．$NumProcs を 1 にすると，1 プロセスでの計算になる．(ホスト名に数字がない場合等の設定は未完成)

```
#!/usr/bin/perl -w

#-------------------------------------------------------------------------
  $NumProcs    =  4;
  @i_Machine   =  (9,9,9,9);
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  ............
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  $MachineFileName    =  "9_MachineFileName";
  $HeadName_Machine   =  'cherry0';
#-------------------------------------------------------------------------
  ............
```

この場合 9_MachineFileName に次のようにマシン名が書き込まれ (ただし，() 内は注釈である．),

```
cherry09  (の 1 個目の CPU)
cherry09  (の 2 個目の CPU)
cherry09  (の 3 個目の CPU)
cherry09  (の 4 個目の CPU)
```

上から順に$NumProcs の数だけ使用する．

また次の例は，4CPU を持つ peach201〜peach204 を繋げたマシン環境での設定である．

```
#!/usr/bin/perl -w

#-------------------------------------------------------------------------
  $NumProcs    =  16;
  @i_Machine   =  (1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4);
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  ............
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  $MachineFileName   =  "9_MachineFileName";
  $HeadName_Machine  =  'peach20';
#-------------------------------------------------------------------------
  ............
```

この場合も 9_MachineFileName に次のようにマシン名が書き込まれ (ただし，() 内は注釈である．),

```
peach201  (の 1 個目の CPU)
peach202  (の 1 個目の CPU)
peach203  (の 1 個目の CPU)
peach204  (の 1 個目の CPU)
peach201  (の 2 個目の CPU)
peach202  (の 2 個目の CPU)
⋮
peach203  (の 4 個目の CPU)
peach204  (の 4 個目の CPU)
```

この場合も上から順に$NumProcs の数だけ使用する．

2. Function.c

   反応項 (関数 Reaction) 及び拡散項 (関数 Diffusion) の設定を行う．この設定は時間発展方程式の Function.c で設定したものと同じである．

```
#include <stdio.h>
  ..........

void  Reaction( TEMP TP, double *par, double *x, double *F )
{
  ..........
  for( i = 1; i <= N; i++ ){
              /* ↓ u の反応項 f(u) を記述 ↓ */
    F[i  ]  = (-par[4]*x[i]+x[i]*x[i]*x[i+N]) / par[3];
              /* ↓ v の反応項 g(u) を記述 ↓ */
    F[i+N]  = par[6]*(par[5]-x[i+N]) - x[i]*x[i]*x[i+N];
  }
}

void  PartialDiff_Function( TEMP TP, double *par, ..... )
{
  ..........
  for( i = 1; i <= N; i++ ){
  ..........
  }
}

void  i_Diffusion( int *i_par )
{
  ..........
}

void  Diffusion( TEMP TP, double *par )
{
  ..........
}

void  PartialDiff_Function_Di( TEMP TP, double *par, double *parD )
{
  ..........
}
```

3. Initial.dat

   時間発展方程式によって求めた final.dat や, 分岐計算によって求めた final.dat , middle_final.dat , 9middle_final.dat , 固有関数のデータなどを Initial.dat にコピーして使用する. どのような場合にどの初期値を使うのかはデモを参照すること.

4. parameter.dat

```
0.000000   ;  ds              ; パラメータの変化量
  0.0001   ;  PseudoNewton   ; ②
      -1   ;  KellerEqn      ; ③
      30   ;  MaxIter        ; ④
 1.0E-03   ;  dt              ; ⑤
       1   ;  N_MSM          ; ⑥
      -1   ;  PeriodFix      ; ⑦
       2   ;  ConPar         ; ⑧
       1   ;  SolSelect      ; ⑨
       3   ;  BC_Select      ; ⑩
       6   ;  TE_Select      ; ⑪
      -1   ;  EigenValue     ; ⑫
      -1   ;  EigenVector    ; ⑬
 1.0E-08   ;  MachineEpsion  ; ⑭
       1   ;  StepCut        ; ⑮
       1   ;  MaxStep        ; 最大 step 数
       0   ;  Gnuplot        ; ⑰
       1   ;  Initial        ; ⑱
```

② PseudoNewton

Newton 法の反復 n 回目に対する近似解の norm 値 (CC_Norm) が PseudoNewton で指定した値 (0.0001) より小さいとき，反復 n 回目は Newton 法から擬似 Newton 法へ変更する．

③ KellerEqn

KellerEqn = 1 なら疑似弧長法を用いる．

KellerEqn $\neq$ 1 なら疑似弧長法を用いない．

④ MaxIter

MaxIter は Newton 法による最大反復回数

最大反復回数を越えた場合は，解が収束しなかったとしてプログラムを終了する．

⑤ dt

時間分割幅 (周期解の分岐計算に必要とする)

⑥ N_MSM

N_MSM = 1 のとき，single shooting 法を用いる．

N_MSM $\geq$ 2 のとき，multiple 法を用いる．multiple 法は周期軌道上の n 個の初期値を用意する．

⑦ PeriodFix

周期をコントロールパラメータとして扱うかどうかの設定．

PeriodFix = 1 のとき，周期をコントロールパラメータとする．

PeriodFix $\neq$ 1 のとき，ConPar で指定したパラメータをコントロールパラメータとする．

⑧ ConPar

コントロールパラメータとして扱うパラメータ番号の設定.

ConPar = n のとき，Initial.dat の左から 3 列目，上から n 行目のパラメータをコントロールパラメータとして与える.

⑨ SolSelect

求めるべき解を指定する.

SolSelect = 1 :定常解.

SolSelect = 2 :進行解.

SolSelect = 3 :脈動定常解.

SolSelect = 4 :脈動進行解.

⑩ BC_Select

境界条件の選択.

1:Dirichlet 境界条件.

2:Neumann 境界条件.

3:周期境界条件.

⑪ TE_Select

離散スキームの選択.

1:差分法を用いる.

2:Spectral 法による波数計算を Euler 法で解く.

3:Spectral 法による波数計算を 4 次の Runge-Kutta 法で解く.

4:差分法を用いる．(一周期分の各時刻の解をメモリに保存する．メモリを大量に使用するが，1 より高速に解ける．).

5:Spectral 法による波数計算を Euler 法で解く (一周期分の各時刻の解をメモリに保存する．メモリを大量に使用するが，2 より高速に解ける．).

6:Spectral 法による波数計算を 4 次の Runge-Kutta 法で解く (一周期分の各時刻の解をメモリに保存する．メモリを大量に使用するが，3 より高速に解ける．).

しかし 4，5，6 を選ぶと非常にメモリーを使うので、エラーが出たら 1 か 2 か 3 にする.

⑫ EigenValue

EigenValue = 1 なら解 ($\boldsymbol{x}^{(*)}$) を求めた後，線形化行列 $A(\boldsymbol{x}^{(*)})$ の固有値計算を行う.

⑬ EigenVector

EigenVector = 1 なら解 ($\boldsymbol{x}^{(*)}$) を求めた後，線形化行列 $A(\boldsymbol{x}^{(*)})$ の固有値に対する固有関数の計算をする.

⑭ MachineEpsion

Newton 法の反復 n 回目に対する近似解の norm 値 (CC_Norm) が MachineEpsilon で指定した値 (10^{-8}) より小さいとき，反復 n 回目の近似解を Newton 法により解が収束したと判定する.

⑮ StepCut

StepCut = n のとき，n ステップおきにデータを出力する.

⑰ Gnuplot (未完成)

Initial (後述)≠ 9 のとき，Newton 法を解きながら計算途中の解情報を gnuplot を用いて表示する．デフォルトでは 1 ~ 7 または 1 ~ 9 の gnuplot 画面の設定がされている.

Gnuplot_Select = 0 と指定したとき，n つの gnuplot 画面が表示され，Gnuplot_Select = $n_1n_2n_3\ldots$ と指定すると $n_1, n_2, n_3, \ldots$ で与えた数字の gnuplot 画面が表示されない．ただし，$n_1, n_2, n_3, \ldots$ は 1 ~ n の値である．例えば Gnuplot_Select = 12 のとき 3,4,5,6 の gnuplot 画面が表示される．gnuplot 画面 1 ~ n の意味は次の通りである．

定常解，進行解のとき

1:初期値の出力 (赤が u，青が v)
2:u の解と v の解の出力
3:u の解
4:横軸にパラメータ，縦軸に $\|\boldsymbol{u}\|_\infty$ を与えた分岐図
5:横軸にパラメータ，縦軸に固有値の実部の値
6:5 の拡大図
7:横軸にパラメータ，縦軸に $\|\boldsymbol{u}\|_{L^2}$ を与えた分岐図 (Newton 法の反復途中の解も分岐図中に表示する)

脈動定常解，脈動進行解のとき

1:初期値の出力 (赤が u，青が v)
2:u の解と v の解の出力
3:u の解
4:横軸にパラメータ，縦軸に $\|\boldsymbol{u}\|_\infty$ を与えた分岐図
5:横軸にパラメータ，縦軸に固有値の実部の値
6:5 の拡大図
7:横軸にパラメータ，縦軸に $\|\boldsymbol{u}\|_{L^2}$ を与えた分岐図 (Newton 法の反復途中の解も分岐図中に表示する)
8:横軸に時間，縦軸に $\|\boldsymbol{u}\|_\infty$ の値
9:横軸に時間，縦軸に $\|\boldsymbol{u}\|_{L^2}$ の値

18 Initial

1:解構造を求める．
2:分岐点における固有ベクトルのデータを用いて分岐点から伸びる解の枝を求める．
7:既に求めたデータの固有値を求める
8:既に求めたデータの固有ベクトルを求める
9:既に求めたデータをディスプレイに表示する．また，midle_final.dat のステップ数を修正する (→ 64 ページ).

5. 0_BranchChangeParameter.dat

parameter.dat の Initial = 2 のとき使用する．

```
    0.01000  ;  eps             ; u_ini の構成に必要
       0.00  ;  period_delta    ; 周期の微小変量
        0.0  ;  velocity_delta  ; 速度の微小変量
 0.00000000  ;  par_delta       ; パラメータの微小変量
      0.000  ;  time            ; u_ini の構成に必要
```

0_BranchChangeParameter.dat で指定したパラメータを用いて，分岐点から伸びる解の枝を求めるための初期値を構成する．

- 進行解の場合

  $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon\boldsymbol{\Phi}_R, \quad c_{ini} = \delta_c, \quad p_{ini} = \delta_p,$

  ただし，$\boldsymbol{u}_0$ は定常解の pitch-fork 分岐点における解 $\boldsymbol{u}$，$\boldsymbol{\Phi}_R$ は分岐点における零固有値に対する固有関数の実部，c は解の速度，p はコントロールパラメータ，$\varepsilon, \delta_c, \delta_p$ はそれぞれ `0_BranchChangeParameter.dat` における `eps` , `velocity_delta` , `par_delta` である．

- 脈動定常解の場合

  $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon(\sin(2\pi t)\boldsymbol{\Phi}_R + \cos(2\pi t)\boldsymbol{\Phi}_I) \quad , T_{ini} = \dfrac{2\pi}{\beta} + \delta_T \quad , p_{ini} = \delta_p,$

  ただし，$\boldsymbol{u}_0$ は定常解の Hopf 分岐点における解 $\boldsymbol{u}$，$\boldsymbol{\Phi}_R, \boldsymbol{\Phi}_I$ はそれぞれ分岐点における零固有値に対する固有関数の実部と虚部，T は解の周期 p はコントロールパラメータ，β は定常解の Hopf 分岐点における固有値虚部の値，$\varepsilon, \delta_T, \delta_p, t$ はそれぞれ `0_BranchChangeParameter.dat` における `eps` , `period_delta` , `par_delta` , `time` である．β:定常解の Hopf 分岐点における固有値虚部の値

- 脈動進行解の場合

  $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon(\sin(2\pi t)\boldsymbol{\Phi}_R + \cos(2\pi t)\boldsymbol{\Phi}_I), \quad T_{ini} = \dfrac{2\pi}{\beta} + \delta_T, \quad c_{ini} = c_0 + \delta_c \quad , p_{ini} = \delta_p,$

  ただし，$\boldsymbol{u}_0$ は進行解の Hopf 分岐点における解 $\boldsymbol{u}$，$\boldsymbol{\Phi}_R, \boldsymbol{\Phi}_I$ はそれぞれ分岐点における零固有値に対する固有関数の実部と虚部，c は解の速度，T は解の周期 p はコントロールパラメータ，c_0 は進行解の Hopf 分岐点における速度，$\varepsilon, \delta_T, \delta_c, \delta_p, t$ はそれぞれ `0_BranchChangeParameter.dat` における `eps` , `period_delta` , `velocity_delta` , `par_delta` , `time` である．

4.1.2 実行コマンドの説明

$ ./aout ↵ で分岐計算のプログラムを実行する．

aout はコンパイルを行い，コンパイルが成功したらプログラムを実行するまでのスクリプトである．aout は次のコマンドを行う．
$ make ↵
$ time␣mpirun␣-np␣$NumProcs␣-machinefile␣$MachineFileName␣./a.out ↵
ただし，$NumProcs は使用するプロセス数，$MachineFileName は使用するマシン名の記述されたファイル (→ 29 ページ参照)．

aout のオプションについて

- $ ./aout␣f ↵

  $ cp␣final.dat␣Initial.dat ↵
  $ ./aout ↵

  と同等のことを行う．

- $ ./aout␣1 ↵ のコマンドを実行すると，

  オブジェクトファイルを再構成して実行する．

  すなわち，

  $ rm␣-rf␣オブジェクトファイル ↵
  $ make ↵
  $ ./aout ↵

  と同等のことを行う．

- $ ./aout␣*directryname* ↵

  ディレクトリ (*directryname*) を作り，その中に

  0_BranchChangeParameter.dat
  81_CP_TV_NR.dat
  82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat
  83_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat
  9midle_final.dat
  Initial.dat
  midle_final.dat
  multiple_points.dat
  parameter.dat

  をコピーする．(Initial.dat は final.dat と中身は同じ)

4.1.3 解の枝の安定性を求める方法

ある解の枝を求めたとして，その安定性を調べたいとする．このとき以下の作業を行う．
ただし，middle_final.dat には解の枝の情報が入っている．

1. middle_final.dat を Initial.dat に変更する．

   cp␣middle_final.dat␣Initial.dat ↵

2. parameter.dat の Initial の値を 8 にする．

3. 実行

   $ ./aout ↵

4.1.4 出力されるデータファイル

1. middle_final.dat

   6 列目，7 列目は場合に応じて出力される．

行数	Step 数		パラメータ	方向ベクトル	解の値	固有値実部	固有値虚部
1	1	N	par[1]	$\dot{u}$[1]	x[1]= u[1]	ReΦ[1]	ImΦ[1]
2	⋮	N_EP	par[2]	⋮	⋮	⋮	⋮
3		Lx	par[3]				
4		dt	⋮				
5		NM					
6		PF	⋮				
7		EES	par[n]				
8		NW	0.0				
9		ME	⋮				
10		REV_i_E					
11		0.0					
⋮		⋮					
19		0.0					
20		N_VAR					
21		BC[1]					
22		BC[2]					
23		BC[3]					
24		BC[4]					
25		c_MP					
26		0.0					
⋮		⋮		⋮	⋮		
Ns				$\dot{u}$[Ns]	x[Ns]= u[Ns]		
$Ns+1$				$\dot{v}$[1]	x[Ns+1]= v[1]		
⋮				⋮	⋮	⋮	⋮
$2Ns$				$\dot{v}$[Ns]	x[2Ns]= v[Ns]	ReΦ[2Ns]	ImΦ[2Ns]
$2Ns+1$				$\dot{T}$	T	0.0	0.0
$2Ns+2$	⋮	⋮	⋮	$\dot{c}$	c	0.0	0.0
$2Ns+3$	1	0.0	0.0	$\dot{\lambda}$	λ	0.0	0.0
$2Ns+4$	2	N	par[1]	$\dot{u}$(1)	x[1]	ReΦ[1]	ImΦ[1]
$2Ns+5$	⋮	N_EP	par[2]	⋮	⋮	⋮	⋮

- 1 列目

  Step 数

- 2 列目
  - NM

- PF
  parameter.dat (→ 31 ページ) の PeriodFix の値.
- ME
  parameter.dat (→ 31 ページ) の MachineEpsion の値.
- REV_i_E
  34 ページの β の値.
- c_MP
  特に気にする必要はない (削除される予定).

時間発展方程式の final.dat (→ 19 ページ) の 2 列目と同様.

- 3 列目
  時間発展方程式の final.dat の 3 列目と同様.
- 4 列目
  方向ベクトル
- 5 列目
  時間発展方程式の final.dat の 5 列目と同様.
- 6 列目
  固有値実部
- 7 列目
  固有値虚部

2. 9middle_final.dat

   middle_final.dat の第 1 列から第 5 列目までの情報.

   9middle_final.dat のデータが存在する場合，9middle_final.dat の必要なステップ部分以外のデータを全て消して Initial.dat に変えて，初期値として使うことができる.

   vi を用いて 9middle_final.dat を編集する方法.
   d G のコマンドでカーソル位置からファイルの末尾までを削除する.
   d 1 G のコマンドでカーソル位置からファイルの先頭までを削除する.

3. final.dat

   9middle_final.dat の最後のステップのデータ.

4. 81_CP_TV_NR.dat

1 列目	2 列目	3 列目	4 列目	5 列目	6 列目
コントロールパラメータ	解の周期	解の速度	$\|\|\boldsymbol{u}\|\|_{\infty}$	$\|\|\boldsymbol{u}\|\|_{L^1}$	$\|\|\boldsymbol{u}\|\|_{L^2}$
⋮	⋮	⋮	⋮	⋮	⋮

5. 82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat

   固有値計算をしているときのみ出力される.

1 列目	2 列目	3 列目	4 列目	5 列目	6 列目	7 列目
ステップ番号	コントロールパラメータ	解の周期	解の速度	$\|\|\boldsymbol{u}\|\|_{\infty}$	$\|\|\boldsymbol{u}\|\|_{L^1}$	$\|\|\boldsymbol{u}\|\|_{L^2}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮

8 列目	9 列目	10 列目	11 列目
固有値の実部	固有値の虚部	固有値の絶対値	不安定な固有値の数 (未完成)
⋮	⋮	⋮	⋮

6. `solution_value.dat`

1 列目	2 列目	3 列目	4 列目
時間	位置 x	u の値	v の値
⋮	⋮	⋮	⋮

第5章　デモ1

この章では，Arrow Stay を用いて 1 次元 Gray-Scott モデルの分岐図を描く工程を説明する．分岐点の分岐の種類についても調べる．
このデモに必要なデータは DEMO ディレクトリの中に図 5.0.1 のように入っている．

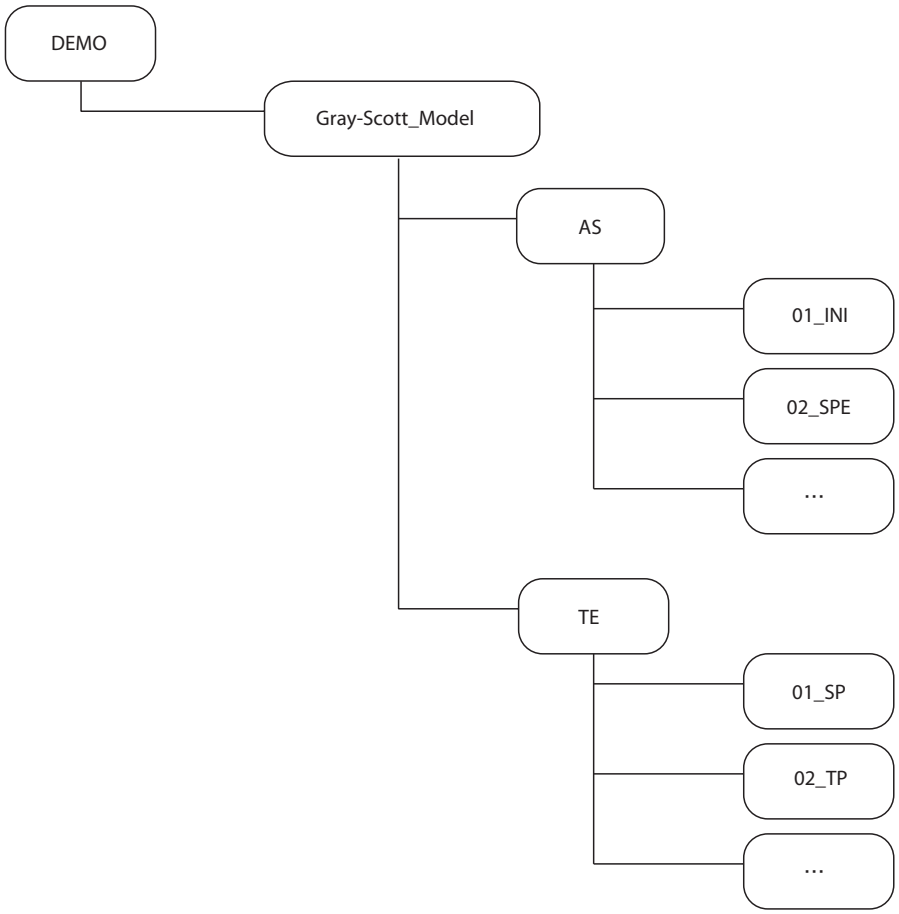


図 5.0.1: DEMO の中身

DEMO 以下には計算に必要な設定ファイルだけがある．実行する場合は，TE（発展方程式計算）あるいは AS（分岐構造計算）にコピーして使用すること[1]．

[1]問題ごとに TE 内あるいは AS 内の全てのファイル

デモプログラム	内容	実行後の保存先
01_SP	定常解 (TE)	SP
02_TP	進行解 (TE)	TP
03_TB1	脈動進行解 (TE)	
04_TB2	脈動進行解 (TE)	TB
01_INI	定常解 (AS)	INI
02_SPE	定常解 (AS)	
03_SP1	定常解 (AS)	SP1
04_SP2	定常解 (AS)	SP2
05_SP3	定常解 (AS)	SP
06_TP1	進行解 (AS)	TP_INI
07_TP2	定常解 (AS)	TP
08_TP3	定常解 (AS)	TP
09_TB1	脈動進行解 (AS)	
10_AP1	脈動進行解 (AS)	
11_AP2	脈動進行解 (AS)	
12_SB1	脈動定常解 (AS)	
13_SB2	脈動定常解 (AS)	SB_INI
14_SB3	脈動定常解 (AS)	SB

5.1 1次元 Gray-Scott モデルの時間発展方程式

ここでは，1次元 Gray-Scott モデルに現れるパルス波を求める．カレントディレクトリは時間発展方程式を解くところに移動 (`$ cd_RD/TE` ↵) して，最初に次の方程式の設定を行う．

$$
\begin{cases}
\left.\begin{aligned} u_t &= \frac{4\pi^2}{L^2}\Delta u + \frac{1}{\varepsilon}(-au + u^2 v), \qquad t > 0, \quad 0 < x < 2\pi, \\ v_t &= d\frac{4\pi^2}{L^2}\Delta v + h(v_c - v) - u^2 v, \end{aligned}\right\} (*1) \\
\left.\begin{aligned} u(t,0) &= u(t,2\pi), \quad u_x(t,0) = u_x(t,2\pi), \quad t > 0, \\ v(t,0) &= v(t,2\pi), \quad v_x(t,0) = v_x(t,2\pi), \quad t > 0, \end{aligned}\right\} (*2) \\
\left.\begin{aligned} u(0,x) &= u_0(x), \quad 0 \le x \le 2\pi, \\ v(0,x) &= v_0(x), \quad 0 \le x \le 2\pi \end{aligned}\right\} (*3)
\end{cases}
$$

①方程式 (∗1) の設定

方程式の拡散項と反応項の設定は `Function.c` で行う．拡散項は関数 `Diffusion` を

```
void  Diffusion( TEMP TP, double *par )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  par[TP.N_EP+1]  =  (4.0*M_PI*M_PI*par[1]) / (par[7]*par[7]);
  par[TP.N_EP+2]  =  (4.0*M_PI*M_PI*par[2]) / (par[7]*par[7]);
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述して，反応項は関数 `Reaction` を

```
void  Reaction( TEMP TP, double *par, double *x, double *F )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  int i;
  int N = TP.N;
  for( i = 1; i <= N; i++ ){
    F[i  ]  =  (-par[4]*x[i]+x[i]*x[i]*x[i+N]) / par[3];
    F[i+N]  =  par[6]*(par[5]-x[i+N]) - x[i]*x[i]*x[i+N];
  }
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述する．

②境界条件 (∗2) の設定

境界条件の設定は `parameter.dat` を開き，`BC_Select` を

```
            ............
      3   ;  BC_Select      ;  OP[9]
            ............
```

と与える．BC_Select = 3 は周期境界条件を表す．

③初期値 (∗3) の設定

時間発展方程式で用いる初期値は求めるパルス波に依存して異なるので，各パルス波を求めるところで説明する．

④①〜③以外の設定

パルス波を求めるには①〜③以外にも設定する箇所 (parameter.dat や 0parameter.dat など) があるが，これらの設定は求めるパルス波に対して異なるので，各パルス波を求めるところで説明する．

Step1 定常パルス波を時間発展方程式を用いて求める

ステップ関数を初期値として進行パルス波を求める．

1. 実行前の設定

   (a) 0parameter.dat

   Function.c 内のパラメータの値を設定する．今回は以下のように設定する．

```
    1.0   ;  du  ; par[1]
    2.0   ;  dv  ; par[2]
    1.1   ;  eps ; par[3]
    0.07  ;  a   ; par[4]
    1.0   ;  vc  ; par[5]
   0.018  ;  h   ; par[6]
    500   ;  Lx  ; par[7]
```

   (b) Function.c

   反応項は

$$f(u,v) = -au + u^2 v$$
$$g(u,v) = h(v_c - v) - u^2 v$$

   であるので、関数 Reaction 内で

```
for( i = 1 ; i <= N ; i++ ){
  F[i  ] = ( -par[4]*x[i] + x[i]*x[i]*x[i+N] ) / par[3];
  F[i+N] = par[6]*( par[5] - x[i+N] ) - x[i]*x[i]*x[i+N];
}
```

   と記述．

   (c) parameter.dat

   今回は以下のように設定する．

```
       256   ;  N                          ; 空間分割数は 256
       128   ;  N_Wave                     ; 切断波数は 128
        -1   ;  T_SELECT                   ; (*1)
        -1   ;  VEL_SELECT                 ; (*1)
   1.0E-01   ;  dt                         ; 時間の刻み幅は 10^-1
6.283185307179586;  Lx                     ; 区間は 2π
       100   ;  Gnuplot_CutN_Vel_Dis      ; (*2)
       0.2   ;  Velocity_Site             ; (*3)
     10000   ;  Velocity_Measure_Start    ; step10000 から速度を測る
         3   ;  BC_Select                 ; 周期境界条件
         3   ;  redEE_Select              ; (*4)
       500   ;  StepCut                   ; 500 回おきにデータを出力
    100000   ;  MaxStep                   ; 最大 step 数
         0   ;  Gnuplot_Select            ; 全ての gnuplot 画面を表示
         1   ;  Initial_Select            ; 初期値はステップ関数
```

(*1) 時間発展方程式は $\boldsymbol{u}_t = \boldsymbol{u}_{xx} + f(\boldsymbol{u}), \boldsymbol{v}_t = \boldsymbol{v}_{xx} + g(\boldsymbol{v})$ を解く
(*2) 速度データを 100 回おきに出力する
(*3) $\boldsymbol{u}$ のパルス波の高さ 0.2 のところで速度を測る
(*4)Spectral 法 (Runge-Kutta 法)

(d) `boundary.dat`
今回は周期境界条件のため，`BC[1]` ~ `BC[4]` は特に設定する必要はない．

(e) `initial_set.c`
初期値の設定を行う．43 ページの (*3) の初期値は図 5.1.1 のステップ関数を与える．

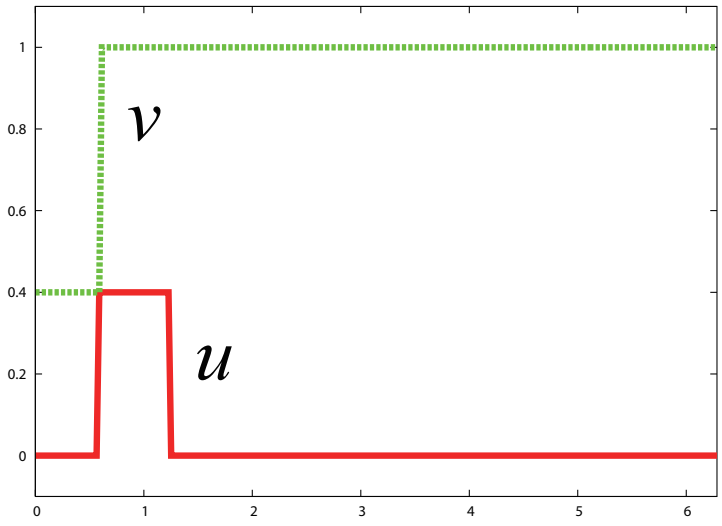


図 5.1.1:

プログラムでは次のように設定する[2]．

[2]プログラムで x[i] = 0 , x[i+N] = 1 と与えたのは，Gray-Scott モデルの拡散項を除いた常微分方程式の平衡点が $u = 0, v = 1$ だからである．

```
void  initial_set2( TEMP *TP, double *par, ····· )
{
  ·····
  if( IS  ==  1 ){
    for( i = 1; i <= N; i++ ){
      x[i  ]  =  0.000000000;
      x[i+N]  =  1.000000000;
      if( N/10 <= i  &&  N/5>= i )
        x[i]  =  0.4;
      if( 1 <= i  &&  N/10>= i )
        x[i+N]  =  0.4;
    }
  }
  else if( IS == 2  ||  IS == 3 ){
    ·····
  }
  ·····
}
```

(f) gnuplot.c

gnuplot で描写する領域を指定する．図 5.1.1 の縦軸の範囲を −0.1 ~ 1.1 で設定する．

```
void  Gnuplot_InitialCommand( TEMP *TP, int *c_G )
{
  ·····
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
  fprintf( GP2, "set yr[-0.1:1.1]\n" );
  fprintf( GP3, "set yr[-0.1:1.1]\n" );
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
  ·····
}
```

2. 実行

$ ./aout␣1 ↵

で実行する．

デモプログラム **1**

Step1 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/TE/01_SP/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Gray-Scott_Model/TE/01_SP には Step1 の設定で必要なファイル．

- 0parameter.dat
- Function.c
- parameter.dat
- initial_set.c
- gnuplot.c

が存在する．

3. 実行結果

   定常パルス波が出れば成功．

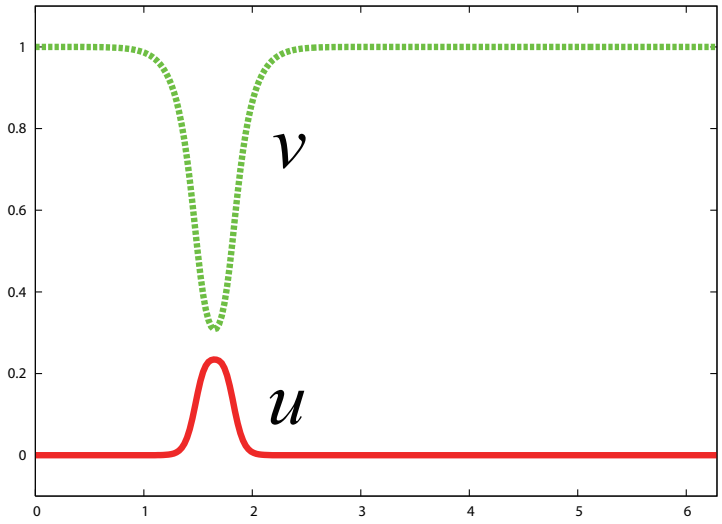


図 5.1.2: 定常パルス波

4. データの保存

   ここで求めた定常パルス波のデータを SP というディレクトリに保存する．次のコマンドを実行することでデータをディレクトリ SP に保存できる．

```
$ ./aout␣SP ↵
```

一般に，データをディレクトリ *directryname* に保存する場合は次のコマンドを実行する．

```
$ ./aout␣directryname ↵
```

ただし，*directryname* には *f* 及び数字から始まる文字列を指定することはできない．

Step2 進行パルス波

ステップ関数を初期値として，進行パルス波を求める．

1. 実行前の設定

   (a) 0parameter.dat

   ```
                  ...........
          3.0     ;   dv   ; par[2]
                  ...........
   ```

   と変更する．

   (b) parameter.dat
   変更点なし．

   (c) initial_set.c
   変更点なし．

2. 実行

   $ ./aout␣1 ↵ で実行する．

   デモプログラム **2**

   Step2 の設定から実行までの操作は

   $ cp␣../DEMO/Gray-Scott_Model/TE/02_TP/*␣./ ↵

   $ ./aout␣1 ↵

   で行える．

   RD/DEMO/Gray-Scott_Model/TE/02_TP には Step2 の設定で必要なファイル．

   - 0parameter.dat
   - Function.c
   - parameter.dat
   - initial_set.c
   - gnuplot.c

   が存在する．

3. 結果

   進行パルス波が出れば成功．

4. データの保存

   $ ./aout␣TP ↵

   でディレクトリ TP にデータを保存する．

Step3.1 脈動進行パルス波

Step2 の結果を初期値として，脈動進行パルス波を求める．

1. 実行前の設定

   (a) 0parameter.dat

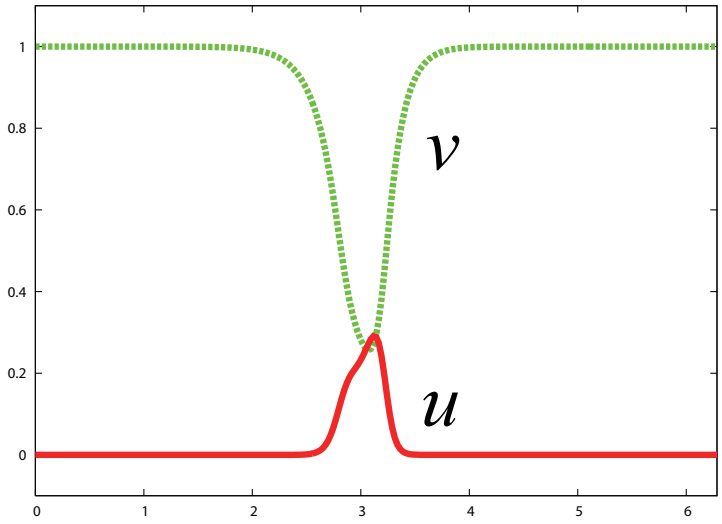


図 5.1.3: 進行パルス波

```
            ...........
    3.38    ;   dv   ; par[2]
            ...........
```

と変更する．

(b) parameter.dat

```
                ...........
          3     ;  Initial_Select          ; (*1)
                ...........
     300000     ;  MaxStep                 ; 最大 step 数
                ...........
```

(*1) 初期値 $\boldsymbol{u}_0(x)$ は Initial.dat に入っているものを用いる．パラメータは 0parameter.dat のものを用いる．

と変更する．脈動進行解を安定させるために最大 step 数を大きくとる．

(c) Initial.dat

Step2 で出力された final.dat を Initial.dat として使用する．

2. 実行

$ cp␣TP/final.dat␣Initial.dat ↵

$ ./aout␣1 ↵

で実行する．

デモプログラム **3.1**

Step3.1 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/TE/03_TB1/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Gray-Scott_Model/TE/03_TB1 には Step3.1 の設定で必要なファイル.

- 0parameter.dat
- Function.c
- parameter.dat
- Initial.dat
- gnuplot.c

が存在する.

3. 結果

脈動進行パルス波が出れば成功.

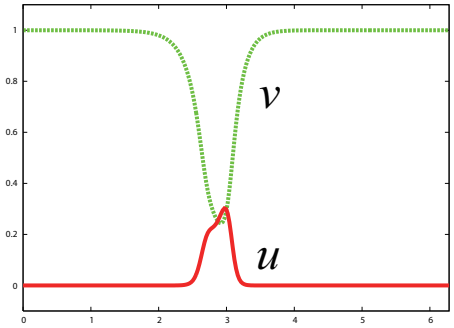


図 5.1.4: 脈動進行パルス波 1

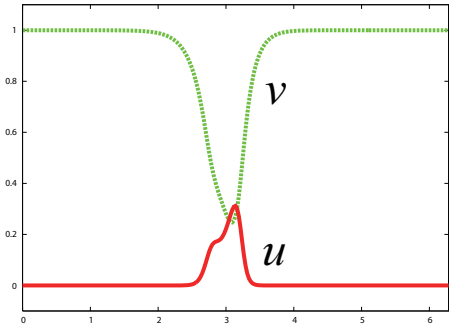


図 5.1.5: 脈動進行パルス波 2

Step3.2 脈動進行パルス波

安定した脈動解を 30000step ほど計算する.

1. 実行前の設定

(a) 0parameter.dat
と変更点なし.

(b) parameter.dat

```
                .....
    30000   ;   MaxStep                  ;
                .....
```

と変更する.

(c) Initial.dat

Step3.1 で出力された final.dat を Initial.dat として使用する.

2. 実行

   $ cp␣final.dat␣Initial.dat ↵

   $ ./aout␣1 ↵

   で実行する.

> デモプログラム **3.2**
>
> Step3.2 の設定から実行までの操作は
>
> $ cp␣../DEMO/Gray-Scott_Model/TE/04_TB2/*␣./ ↵
>
> $ ./aout␣1 ↵
>
> で行える.
>
> RD/DEMO/Gray-Scott_Model/TE/04_TB2 には Step3.2 の設定で必要なファイル.
>
> - 0parameter.dat
> - Function.c
> - parameter.dat
> - Initial.dat
> - gnuplot.c
>
> が存在する.

3. データの保存

   $ ./aout␣TB ↵

   でディレクトリ TB にデータを保存する.

5.2 1次元 Gray-Scott モデルの分岐計算

ここでは，5.1 節で求めたパルス波の分岐図を求める．ただし，d を分岐パラメータとして与える．カレントディレクトリは分岐図を求めるところに移動 (`$ cd␣RD/AS ↵`) して，最初に Gray-Scott モデル (43 ページの (∗1)) の拡散項と反応項の設定を行う．これらの設定は時間発展方程式と同じであり，拡散項は `Function.c` の関数 `Diffusion` を

```
void  Diffusion( TEMP TP, double *par )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  par[TP.N_EP+1]  =  (4.0*M_PI*M_PI*par[1]) / (par[7]*par[7]);
  par[TP.N_EP+2]  =  (4.0*M_PI*M_PI*par[2]) / (par[7]*par[7]);
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述して，反応項は `Function.c` の関数 `Reaction` を

```
void  Reaction( TEMP TP, double *par, double *x, double *F )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  int i;
  int N = TP.N;
  for( i = 1; i <= N; i++ ){
    F[i  ]  =  (-par[4]*x[i]+x[i]*x[i]*x[i+N]) / par[3];
    F[i+N]  =  par[6]*(par[5]-x[i+N]) - x[i]*x[i]*x[i+N];
  }
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述する．次に周期境界条件 (43 ページの (∗2)) の設定を行う．境界条件の設定は `parameter.dat` を開き，`BC_Select` を

```
                ............
          3   ;  BC_Select      ;  OP[9]
                ............
```

と与える．ただし，`BC_Select` = 3 は周期境界条件を表す．

Arrow Stay の分岐計算ソフトウェアは Newton 法を用いて，解構造を数値的に求める．解構造の計算は図 5.2.1(a) のように，各 step ごとにパラメータを固定しながら $\tilde{u}_1 \to \tilde{u}_2 \to \ldots$ と順次に求める方法や図 (b) のように，パラメータを動かしながら $\hat{u}_1 \to \hat{u}_2 \to \ldots$ と順次に求める方法がある．

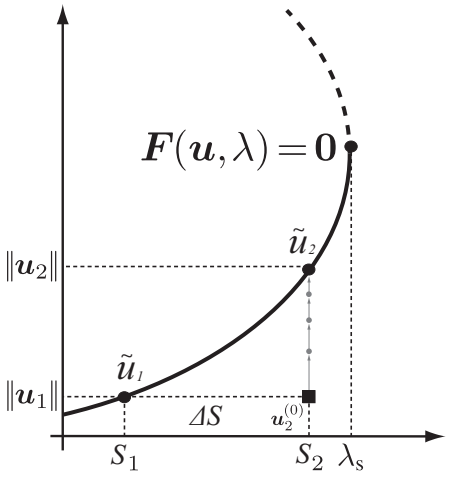


図 5.2.1: (a)

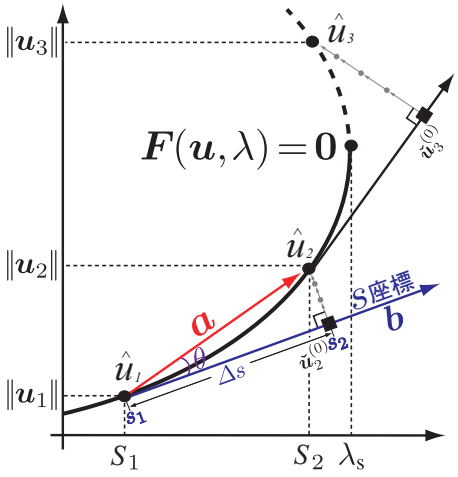


図 5.2.2: (b)

(a), (b) の選択およびパラメータの変化量 Δs は parameter.dat 内で設定する．このとき，Newton 法で用いる初期値は求めるパルス波に依存して変わるため，各パルス波の分岐図を求めるところで説明する．分岐計算を行うには，他にも parameter.dat 内のパラメータ調整や必要に応じて 0_BranchChangeParameter の設定を必要とするが，これらの設定はデモプログラム内で説明する．

Step1.1 定常解の分岐計算

5.1 節で求めた Step1 の解を用いて，定常解の分岐図を求める．ただし，Step1.1 では定常解の安定性を計算しないとする．定常解の選択およびその安定性計算による設定は parameter.dat を開き，次のように設定する．

```
            ............
  1     ;   SolSelect       ;   OP[8]
            ............
 -1     ;   EigenValue      ;   OP[11]
 -1     ;   EigenVector     ;   OP[12]
            ............
```

ただし，SolSelect = 1 は定常解を選択し，EigenValue と EigenVector を 1 以外の値に選択すると解の安定性を計算しない．

1. 実行前の設定

   (a) Initial.dat

   5.1 節の Step1 で求めた解を Newton 法の初期値に与える．

   $ cp␣../TE/SP/final.dat␣Initial.dat ↵

   (b) parameter.dat

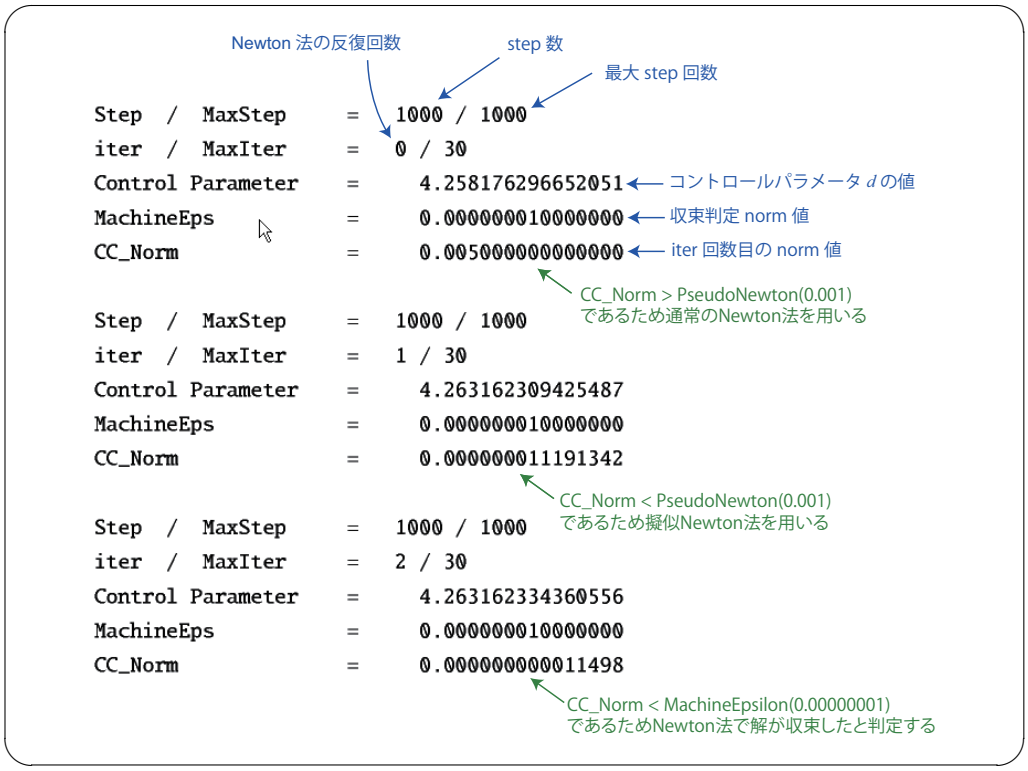
   今回は以下のように設定する．

```
0.000000   ;  ds              ; パラメータの変化量なし
  0.0001   ;  PseudoNewton   ; (*1)
      -1   ;  KellerEqn      ; 擬似弧長法の設定なし (図 4.2.1(a))
      30   ;  MaxIter        ; Newton 法の最大反復回数
 1.0E-03   ;  dt              ; 定常解の分岐計算では不必要な設定
       1   ;  N_MSM           ; 定常解の分岐計算では不必要な設定
      -1   ;  PeriodFix      ; 定常解の分岐計算では不必要な設定
       2   ;  ConPar          ; d を分岐パラメータに選択
       1   ;  SolSelect       ; 定常解を選択
       3   ;  BC_Select       ; 周期境界条件を選択
       6   ;  TE_Select       ; 定常解の分岐計算では不必要な設定
      -1   ;  EigenValue      ; 固有値の計算をしない
      -1   ;  EigenVector     ; 固有ベクトルの計算をしない
 1.0E-08   ;  MachineEpsion   ; (*2)
       1   ;  StepCut         ; 毎 step ごとにデータを出力する
       1   ;  MaxStep         ; 最大 step 回数
       0   ;  Gnuplot         ; 全ての gnuplot 画面を出力する
       1   ;  Initial         ; 分岐図を求めることを選択
```

(∗1)Newton 法の反復 n 回目に対する近似解の norm 値 (CC_Norm) が PseudoNewton で指定した値 (0.0001) より小さいとき，反復 n 回目は Newton 法から擬似 Newton 法へ変更する．

(∗2)Newton 法の反復 n 回目に対する近似解の norm 値 (CC_Norm) が MachineEps で指定した値 (10^{-8}) より小さいとき，反復 n 回目の近似解を Newton 法により解が収束したと判定する．

分岐計算のプログラムを実行すると，標準出力に CC_Norm の値などが出力される．例えば，実行後次の画面が出力されたとすると，



反復 1 回目で CC_Norm ＜ PseudoNewton (0.0001) であるため，Newton 法から擬似 Newton 法になる．
反復 2 回目で CC_Norm ＜ MachineEps (10^{-8}) であるため，解が収束したと判定する．

2. 実行

上記の設定が終わったら，

```
$ ./aout␣1 ↵
```

で実行する．

デモプログラム **1.1**

Step1.1 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/AS/01_INI/*␣./ ↵
```

```
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Gray-Scott_Model/AS/01_INI には Step1.1 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

以下のように「Convergence !!」と表示されれば，Newton 法を用いて解が収束したと判

定する．

```
.............        Newton 法の反復回数   step 数
                                          最大 step 回数
Step  /  MaxStep     =   1 / 1
iter  /  MaxIter     =   0 / 30
Control Parameter    =     2.000000000000000 ← コントロールパラメータ d の値
MachineEps           =     0.000000010000000 ← 収束判定 norm 値
CC_Norm              =     0.000007185424841 ← iter 回数目の norm 値

Step  /  MaxStep     =   1 / 1
iter  /  MaxIter     =   1 / 30
Control Parameter    =     2.000000000000000
MachineEps           =     0.000000010000000
CC_Norm              =     0.000000000039912
-------------------------------------------------------------
                      Convergence  !!
-------------------------------------------------------------
.............
```

4. データの保存

   Step1.1 で求めた定常解のデータを INI というディレクトリに保存する．
   次のコマンドを実行することでデータをディレクトリ INI に保存できる．

   $ ./aout␣INI ↵

> 時間発展方程式のデモプログラムで説明したように，一般には次のコマンドを実行することでデータをディレクトリ *directryname* に保存できる．
> $ ./aout␣*directryname* ↵
> ただし，*directryname* には *f* 及び数字から始まる文字列を指定することはできない．

Step1.2 定常解の分岐計算

次に，Step1.1 で求めた解を用いてパラメータを変化させながら定常解の枝を求める．

1. 実行前の設定

   (a) Initial.dat
   初期値は Step1.1 で求めた解を用いる．

   (b) parameter.dat

```
  0.005   ;  ds              ; パラメータの変化量 0.005
             .............
    10    ;  StepCut         ; 10step ごとにデータを出力する
  1000    ;  MaxStep         ; 最大 step 回数
             .............
```

   と変更する．

2. 実行

   $ cp␣INI/9middle_final.dat␣Initial.dat ↵

$ ./aout␣1 ↵

デモプログラム **1.2**

Step1.2 の設定から実行までの操作は

$ cp␣../DEMO/Gray-Scott_Model/AS/02_SPE/*␣./ ↵

$ ./aout␣1 ↵

で行える．

RD/DEMO/Gray-Scott_Model/AS/02_SPE には Step1.2 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

```
  ...........
  Step  /  MaxStep     =   73 / 1000
  iter  /  MaxIter     =   29 / 30
  Control Parameter    =     2.364999999999992
  MachineEps           =     0.0000000100000000
  CC_Norm              =     0.000006462960947

  Step  /  MaxStep     =   73 / 1000
  iter  /  MaxIter     =   30 / 30
  Control Parameter    =     2.364999999999992
  MachineEps           =     0.0000000100000000
  CC_Norm              =     0.000006461318227
-----------------------------------------------------------------
                     Not Convergence  !!
-----------------------------------------------------------------


-----------------------------------------------------------------------
Iteration Over !! (30)
Iteration Over !! (30)
Iteration Over !! (30)
Iteration Over !! (30)
Iteration Over !! (30)
-----------------------------------------------------------------------
  ...........
```

73step において，Newton 法による反復回数が最大反復回数 (この場合は 30 回) を越えてしまい，73step の定常解が収束しなかったと判定される．

Newton 法により解が収束しない原因として，

- saddle-node 分岐により解が存在しないところで計算した
- パラメータの変化量を大きくしすぎた

などが考えられる．

ここでは，73step 近傍で saddle-node 分岐が起きるため，解が収束しない (図 5.2.3 参照)．図 5.2.3 の横軸はコントロールパラメータ d，縦軸は $\|\boldsymbol{u}\|_\infty$ である．

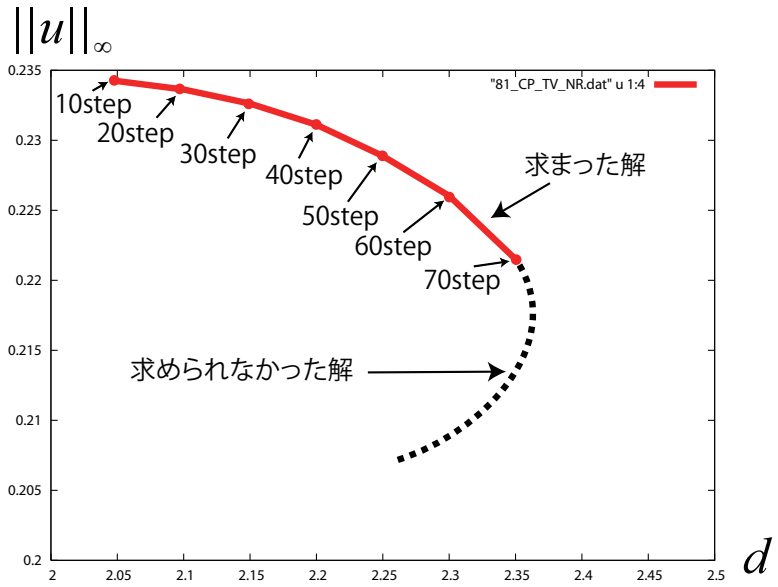


図 5.2.3: 定常パルス波の分岐図

Step1.3 定常解の分岐計算

Step1.2 の 73step 以降を求めるには擬似弧長法を用いる．ここでは，擬似弧長法を用いて Step1.1 の結果から再び定常解の分岐計算を行う．

1. 実行前の設定
   (a) `Initial.dat`
   初期値は step1.1 で求めた `9middle_final.dat` を用いる．
   (b) `parameter.dat`

```
-0.005    ;  ds              ; パラメータの変化量 −0.005
                ..........
     1    ;  KellerEqn      ; 擬似弧長法の設定あり
                ..........
```

   と変更する[3]．
2. 実行

```
$ cp␣INI/9middle_final.dat␣Initial.dat ↵
```

[3] ds は負の値に変更したが，擬似弧長法を用いると自分の進みたい方向と逆方向に分岐の解の枝が進むことがおきる．この場合，例えば ds の正負の値を変更して自分の進みたい方向に解の枝が進むよう調整する．

```
$ ./aout␣1 ↵
```

デモプログラム **1.3**

Step1.3 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/AS/03_SP1/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Gray-Scott_Model/AS/03_SP1 には Step1.3 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

```
  ............
  Step  /  MaxStep     =   1000 / 1000
  iter  /  MaxIter     =   0 / 30
  Control Parameter    =     4.087204836309247
  MachineEps           =     0.0000000010000000
  CC_Norm              =     0.0050000000000000

  Step  /  MaxStep     =   1000 / 1000
  iter  /  MaxIter     =   1 / 30
  Control Parameter    =     4.092168559232388
  MachineEps           =     0.0000000010000000
  CC_Norm              =     0.0000000011138078

  Step  /  MaxStep     =   1000 / 1000
  iter  /  MaxIter     =   2 / 30
  Control Parameter    =     4.092168623444494
  MachineEps           =     0.0000000010000000
  CC_Norm              =     0.0000000000011126
------------------------------------------------------------
                        Convergence  !!
------------------------------------------------------------
  ............
```

実行結果の分岐図を見るには，gnuplot を開き

```
$ plot "81_CP_TV_NR.dat" u 1:4
```

と実行する[4]．分岐図は図 5.2.4 となり，saddle-node 分岐点が 2 つ存在する．点 S は

[4]81_CP_TV_NR.dat は 1 列目にコントロールパラメータ，2 列目に解の周期，3 列目に解の速度，4 列目に $\|\boldsymbol{u}\|_\infty$，5

step1.3 で求めた最初の定常解であり，点 S から点 E1 まで矢印の方向に定常解の枝が求まる．

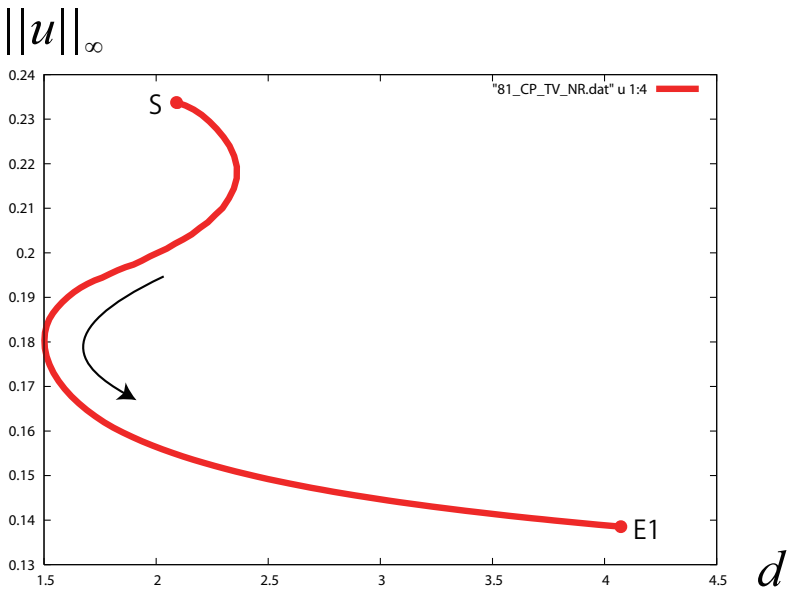


図 5.2.4: 定常パルス波の分岐図

4. データの保存

   step1.3 で求めた分岐図のデータを SP1 のディレクトリに保存する．

   `$ ./aout␣SP1` ↵

Step1.4 定常解の分岐計算

次に，図 5.2.4 の点 S から左側の定常解の分岐計算をする．

1. 実行前の設定

   (a) `Initial.dat`

   点 S を初期値として設定するために，SP1 の `9middle_final.dat` を `Initial.dat` にコピーし (`$ cp␣SP1/9middle_final.dat␣Initial.dat` ↵)， `Initial.dat` の最初のステップ以外のデータを全て削除する．この削除方法に vi を用いる．以下に vi を用いてファイルを編集する方法を説明する．

列目に $\|\boldsymbol{u}\|_{L^1}$，6 列目に $\|\boldsymbol{u}\|_{L^2}$ が入っている．gnuplot で u 1:4 と指定すると，横軸にコントロールパラメータ，縦軸に $\|\boldsymbol{u}\|_\infty$ のグラフが表示する．

i. vi で Initial.dat を開く.

$ vi␣Initial.dat ↵

ii. 削除したい部分の最初の行にカーソルを移動する.

$ /␣␣␣␣2 ↵ と入力すると 2step 目のデータの最初の行にカーソルが移動する.

iii. カーソル位置からファイルの末尾までを削除する.

d G のコマンドでカーソル位置からファイルの末尾までを削除することができる.

iv. ファイルを保存して終了する.

$:wq ↵ と入力することでデータを保存して終了する.

(b) parameter.dat

```
0.005   ;  ds              ; パラメータの変化量 0.005
           ............
```

と変更する.

2. 実行

$ ./aout␣1 ↵

で実行する.

デモプログラム **1.4**

Step1.4 の設定から実行までの操作は

$ cp␣../DEMO/Gray-Scott_Model/AS/04_SP2/*␣./ ↵

$ ./aout␣1 ↵

で行える.

RD/DEMO/Gray-Scott_Model/AS/04_SP2 には Step1.4 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat

が存在する.

3. 実行結果

```
  ...........
  Step   /   MaxStep      =    1000 / 1000
  iter   /   MaxIter      =    0 / 30
  Control Parameter       =      4.258176296652051
  MachineEps              =      0.0000000010000000
  CC_Norm                 =      0.005000000000000

  Step   /   MaxStep      =    1000 / 1000
  iter   /   MaxIter      =    1 / 30
  Control Parameter       =      4.263162309425487
  MachineEps              =      0.0000000010000000
  CC_Norm                 =      0.000000011191342

  Step   /   MaxStep      =    1000 / 1000
  iter   /   MaxIter      =    2 / 30
  Control Parameter       =      4.263162334360556
  MachineEps              =      0.0000000010000000
  CC_Norm                 =      0.000000000011498
-----------------------------------------------------------------
                         Convergence   !!
-----------------------------------------------------------------
  ...........
```

Step1.4 で求めた定常解の分岐図を Step1.3 と同様な方法で gnuplot で表示すると図 5.2.5 の結果になる.

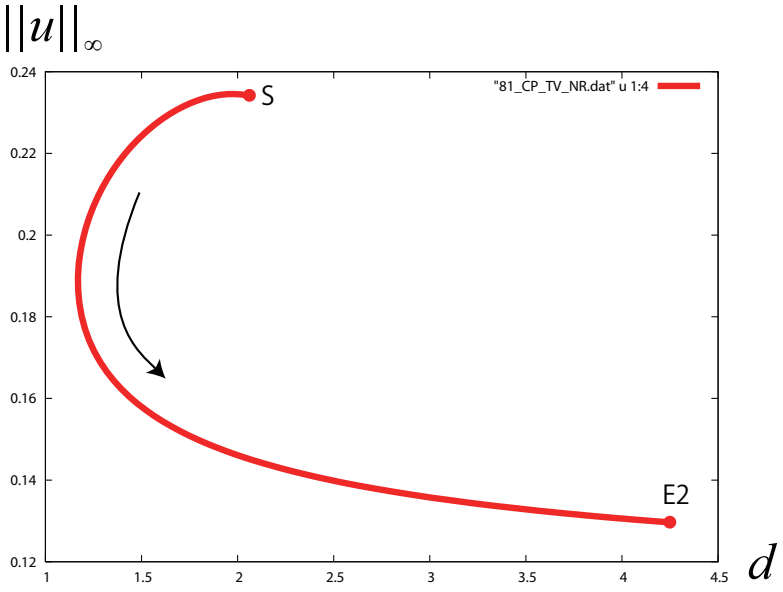


図 5.2.5: 定常パルス波の分岐図

4. データの保存

step1.4 で求めた分岐図のデータを SP2 というディレクトリに保存する．

```
$ ./aout␣SP2 ↵
```

Step1.5 定常解の分岐計算

Step1.3 と Step1.4 で求めたデータを 1 つのデータに結合する．データを結合するだけなら

```
$ cat␣SP1/middle_final.dat␣SP2/middle_final.dat > newfilename ↵
```

と実行することでデータをまとめることができる．しかし，今回の場合は図 5.2.4 と図 5.2.5 で分岐計算を進めた方向が違うため，Step1.3 か Step1.4 で求めた middle_final.dat のどちらかの step の進み方を逆にする必要がある．ここでは Step1.4 で求めた middle_final.dat の step の進み方を逆にする．

1. Step1.4 で求めた middle_final.dat の step の進み方を逆にする

   Sort は midle_final.dat の step の進み方を逆に並び替えるスクリプトである．
   $ ./Sort ↵
   と実行すると，middle_final.dat の step の進み方が逆になる．

   midle_final.dat の中身が以下のようであったとする．

step 数	…	…	…
1	⋮	⋮	⋮
⋮	⋮	⋮	⋮
1	⋮	⋮	⋮
2	⋮	⋮	⋮
⋮	⋮	⋮	⋮
2	⋮	⋮	⋮
⋮	⋮	⋮	⋮
n	⋮	⋮	⋮
⋮	⋮	⋮	⋮
n	⋮	⋮	⋮

   $ ./Sort ↵ と実行することで，以下のように並び替わる．

step 数	...	...	...
n	⋮	⋮	⋮
⋮	⋮	⋮	⋮
n	⋮	⋮	⋮
⋮	⋮	⋮	⋮
2	⋮	⋮	⋮
⋮	⋮	⋮	⋮
2	⋮	⋮	⋮
1	⋮	⋮	⋮
⋮	⋮	⋮	⋮
1	⋮	⋮	⋮

ここではStep1.4のデータを逆の方向に並べ替えたいため，まずSP2に保存したmiddle_final.datを持ってくる ($ cp␣SP2/middle_final.dat␣middle_final.dat ↵).
そして$ ./Sort ↵ を実行する．

2. データを結合する方法

   一般に，*filename*1 と *filename*2 を結合して *newfilename* というファイルに保存する場合には以下のコマンドを実行する．

   $ cat␣*filename*1␣*filename*2 > *newfilename* ↵

   3つのデータを結合させる場合

   $ cat␣*filename*1␣*filename*2␣*filename*3 > *newfilename* ↵

   このコマンドを使い，順番を並べ替えたSP2のmiddle_final.datと，SP1のmiddle_final.datのデータを結合し適当なファイル (ここではtmp.dat) に保存するには，

   $ cat␣middle_final.dat␣SP1/middle_final.dat > tmp.dat ↵

   と実行する．

3. 結合したデータのステップ数を修正する方法

   (a) 結合したデータのファイル名をmiddle_final.datに変更する．
   $ mv␣tmp.dat␣middle_final.dat ↵

   (b) parameter.datのInitialの値を9に変更する．

   (c) $ ./aout␣1 ↵ と実行する．

   実行後，結合したmiddle_final.datに対応する

   - 81_CP_TV_NR.dat
   - 82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat
   - solution_value.dat
   - 1
   - 11

   が出力される．ただし，1はステップ数が修正されたmiddle_final.dat，11はステップ数が修正された9middle_final.datである．

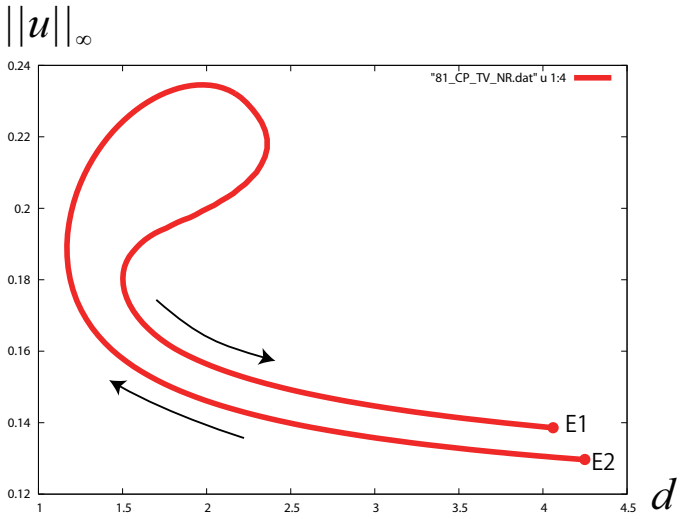


図 5.2.6: 定常パルス波の分岐図

結合した分岐図を Step1.3 と同様な方法で gnuplot で表示すると図 5.2.6 のようになる．

縦軸を $\|u\|_{L^2}$ にすると図 5.2.7 のようになる．

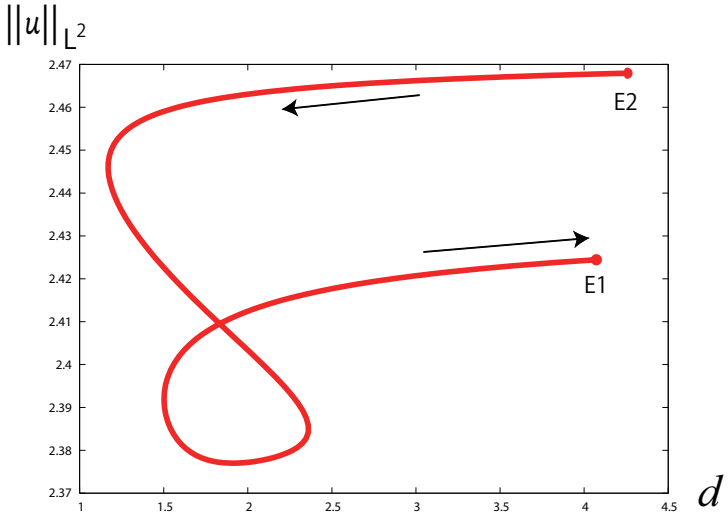


図 5.2.7: 定常パルス波の分岐図

1，11 をそれぞれ middle_final.dat，9middle_final.dat に上書きし，データを SP というディレクトリに保存する．

$ ./aout␣SP ↵

1 つのデータにまとめたので SP1 と SP2 のデータは$ rm␣-rf␣SP1␣SP2 で削除する．

Step1.6 定常解の分岐計算

ここでは Step1.5 で求めた定常解の安定性を求める方法について説明する．解の安定性は，

①解の枝を求めながら安定性を求める方法と

②解の枝を求め終わった後、それらの解の安定性を調べる方法

の 2 通りある．ここでは，定常解の枝を Step1.5 で既に求めたので 2 の方法を用いる

1. 実行前の設定
   (a) Initial.dat
   Step1.5 で保存したディレクトリ SP の中から安定性計算に必要とする 9middle_final.dat を Initial.dat に変更して計算するディレクトリに持ってくる．
   $ cp␣SP/9middle_final.dat␣Initial.dat
   (b) parameter.dat

   ```
                  .............
          8    ;  Initial          ; (*1)
   (*1) 各 step における解の固有値と固有関数の計算を行う
   ```

   と変更する．
2. 実行
   $ ./aout␣1 ↵ で実行する．
   (37 ページにも解説がある．)

   デモプログラム **1.5**

   Step1.6 の設定から実行までの操作は

   $ cp␣../DEMO/Gray-Scott_Model/AS/05_SP3/*␣./ ↵

   $ ./aout␣1 ↵

   で行える．

   RD/DEMO/Gray-Scott_Model/AS/05_SP3 には Step1.6 の設定で必要なファイル．

   - Function.c
   - Initial.dat
   - parameter.dat

   が存在する．
3. 実行結果
   安定性の結果は 82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat のデータを参照する．gnuplot を開き$ gnuplot > plot␣"82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat"␣u␣1:8 と実行することで横軸にステップ数，縦軸に固有値実部を与えたグラフが表示される (図 5.2.8)．

step の変化と共に固有値の実部が 0 を横切るとき，分岐が起きる．この場合，6 つの分岐が存在する．

図 5.2.9 の 1〜6 の分岐点の種類について述べる．

1 の分岐点について

1 の分岐点は step= 70 の近くで起こり，82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat の step= 70 の近傍では次の結果となる．

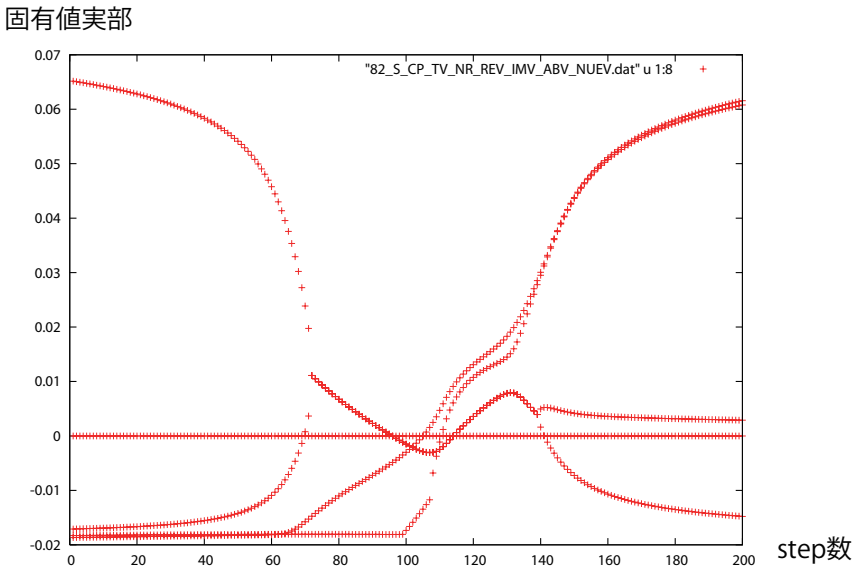


図 5.2.8:

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値．

```
                          ..........
69  ......................  2.72392992E-02  0.00000000E+00  ..........
69  ......................  6.77088438E-09  0.00000000E+00  ..........
69  ......................  -1.38528671E-03  0.00000000E+00  ..........
69  ......................  -1.62637187E-02  0.00000000E+00  ..........
69  ......................  -1.80527589E-02  0.00000000E+00  ..........
70  ......................  2.38602696E-02  0.00000000E+00  ..........
70  ......................  7.67131139E-04  0.00000000E+00  ..........
70  ......................  6.04405127E-09  0.00000000E+00  ..........
70  ......................  -1.57752778E-02  0.00000000E+00  ..........
70  ......................  -1.80527990E-02  0.00000000E+00  ..........
                          ..........
```

step= 70 近くで固有値実部が 1 つ 0 を横切ることがわかる．ただし，定常解の周期境界条件の影響でシフトの自由度があるため，固有値のひとつは零固有値を持つ．この零固有値は安定性の条件に影響を与えない．固有値実部の 1 つが 0 を横切るとき，次の 3 つのタイプの分岐が現れる:

saddle-node 分岐

pitch-fork 分岐

transcritical 分岐

このうち step の変化と共に分岐点近傍でコントロールパラメータの進む向きが変わるとき，saddle-node 分岐が起きる．gnuplot を開き，

```
$ gnuplot > plot␣"82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat"␣␣u␣1:2
```

と実行して横軸に step 数，縦軸にコントロールパラメータ d を表示させる (図 5.2.10)．このとき step= 70, 110, 141 でコントロールパラメータの向きが変化することがわか

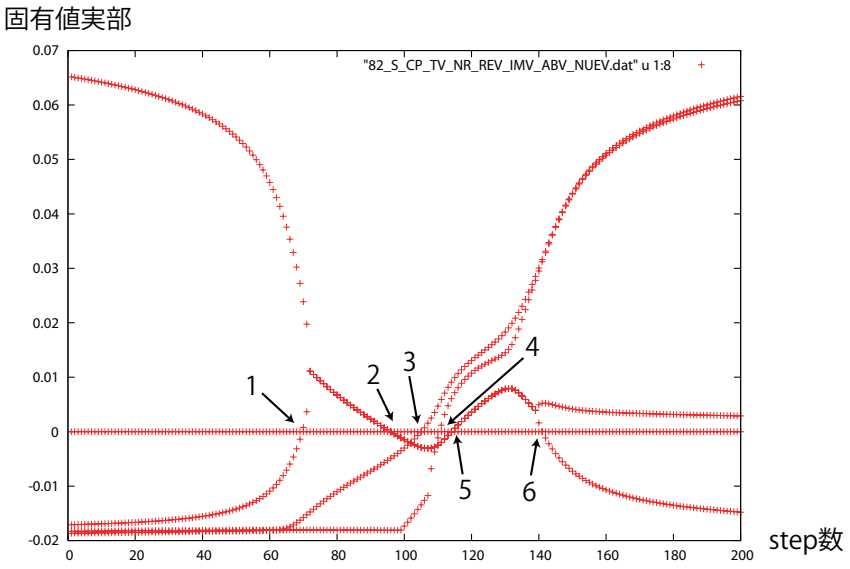


図 5.2.9:

る．すなわち，step= 70, 110, 141 で saddle-node 分岐が起きる．従って，図 5.2.9 の 1 は saddle-node 分岐点である．同様に，図 5.2.9 の 4 と 6 も saddle-node 分岐点であることがわかる (図 5.2.11).

2 の分岐点について

2 の分岐点は step= 96 の近くで起こり，`82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat` の step= 96 をみると

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値.

```
                          ..........
95  ....................  3.39914803E-04  2.62426635E-02  ..........
95  ....................  3.39914803E-04 -2.62426635E-02  ..........
95  ....................  1.89058850E-08  0.00000000E+00  ..........
95  ....................  -5.28527110E-03  0.00000000E+00  ..........
95  ....................  -1.80952017E-02  0.00000000E+00  ..........
96  ....................  2.04794297E-08  0.00000000E+00  ..........
96  ....................  -4.20137537E-05 -2.64192278E-02  ..........
96  ....................  -4.20137537E-05  2.64192278E-02  ..........
96  ....................  -4.86485952E-03  0.00000000E+00  ..........
96  ....................  -1.80979881E-02  0.00000000E+00  ..........
                          ..........
```

となる．step の変化と共に 2 つの固有値が固有値虚部を持ちながら複素平面の虚軸を横切ることがわかる．この場合，Hopf 分岐が起きる．同様に，図 5.2.9 の 5 の場合も `82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat` の step= 115 をみると

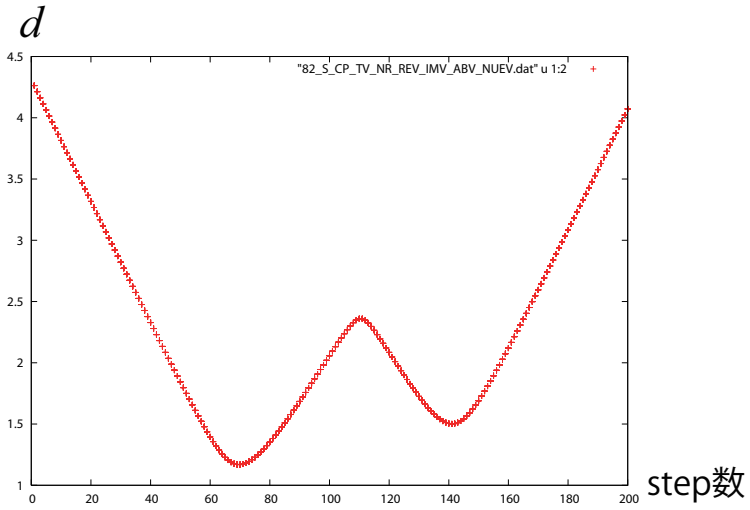


図 5.2.10:

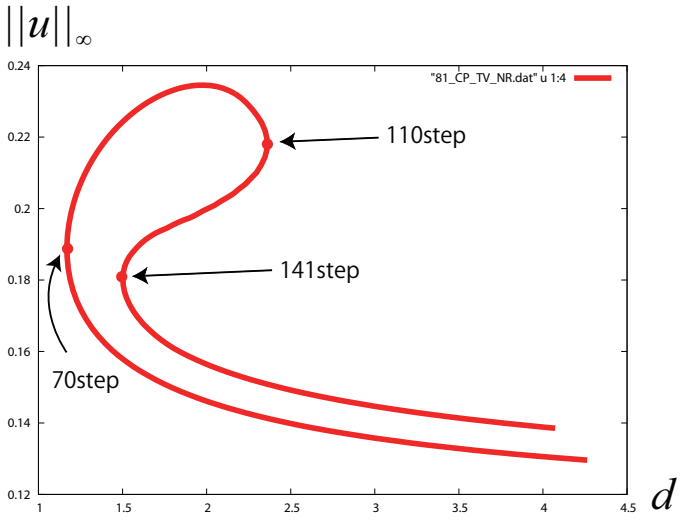


図 5.2.11:

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値.

```
                                  .....
114  ......................  9.09383226E-03  0.00000000E+00  .....
114  ......................  6.06194605E-03  0.00000000E+00  .....
114  ......................  -7.51816727E-09  0.00000000E+00  .....
114  ......................  -3.51838009E-05 -2.54489667E-02  .....
114  ......................  -3.51838009E-05  2.54489667E-02  .....
115  ......................  9.93006414E-03  0.00000000E+00  .....
115  ......................  7.15965142E-03  0.00000000E+00  .....
115  ......................  6.13616555E-04  2.47780506E-02  .....
115  ......................  6.13616555E-04 -2.47780506E-02  .....
115  ......................  -5.09436851E-09  0.00000000E+00  .....
                                  .....
```

同じことが起きるため，Hopf 分岐が起きている．

3 の分岐点について

3 の分岐点は step= 107 の近くで起こり，82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat をみると，

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値．

```
                              ·····
106  ·····························  7.19556478E-04  0.00000000E+00  ·····
106  ·····························  -1.35210633E-08  0.00000000E+00  ·····
106  ·····························  -3.01568890E-03  2.77459724E-02  ·····
106  ·····························  -3.01568890E-03 -2.77459724E-02  ·····
106  ·····························  -1.26064364E-02  4.66843397E-03  ·····
107  ·····························  1.55106188E-03  0.00000000E+00  ·····
107  ·····························  7.65809216E-09  0.00000000E+00  ·····
107  ·····························  -3.06796344E-03  2.78066326E-02  ·····
107  ·····························  -3.06796344E-03 -2.78066326E-02  ·····
107  ·····························  -1.17306086E-02  1.92013563E-03  ·····
                              ·····
```

となる．step の変化と共に 1 つの固有値が 0 を横切ることがわかる．この場合は，先程と同様に 3 つの分岐が現れるが，図 4.2.9 の情報から saddle-node 分岐ではないことがわかるため，pitch-fork 分岐か transcritical 分岐である．これ以上の判別は固有値の情報から理解できない．これを判別するには，3 の分岐点から伸びる解の枝を求める必要がある．詳しい説明は省略するが，3 は pitch-fork 分岐点であることが説明できる．以上の結果から，図 5.2.9 の 1〜6 では次の 3 つの種類の分岐が現れることがわかる．

- saddle-node 分岐 (1,4,6)
- Hopf 分岐 (2,5)
- pitch-fork 分岐 (3)

データの保存

Step1.6 で求めた分岐図のデータを SP というディレクトリに上書きする．

```
$ ./aout␣SP ↵
```

このとき，ディレクトリを上書きしますかと尋ねられるので，`$ yes ↵` と入力する．

Step2.1 進行解の数値計算

ここでは Step1.6 で求めた定常解の pitch-fork 分岐点における固有関数の情報を用いて進行解を求める．用いる固有関数の情報は，固有値の実部が正である step のものを用いる．Step1.6 でも示したように，pitch-fork 分岐点での固有値の情報は以下のようになっている．

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値．

```
                          ..........
106  ......................  7.19556478E-04  0.00000000E+00  ..........
106  ......................  -1.35210633E-08  0.00000000E+00  ..........
106  ......................  -3.01568890E-03  2.77459724E-02  ..........
106  ......................  -3.01568890E-03 -2.77459724E-02  ..........
106  ......................  -1.26064364E-02  4.66843397E-03  ..........
107  ......................  1.55106188E-03  0.00000000E+00  ..........
107  ......................  7.65809216E-09  0.00000000E+00  ..........
107  ......................  -3.06796344E-03  2.78066326E-02  ..........
107  ......................  -3.06796344E-03 -2.78066326E-02  ..........
107  ......................  -1.17306086E-02  1.92013563E-03  ..........
                          ..........
```

step106 では負の値だった 2 行目の固有値実部が，step107 では正の値になっている．したがって，ここでは step107 の固有関数の情報を用いる．

1. 実行前の設定
   (a) `Initial.dat`
   pitch-folk 分岐点の固有関数を `Initial.dat` として使用する．固有ベクトルのデータは `6_EigenVector_Data` に入っている．
   `6_EigenVector_Data/ 107_CP` ...[5]`/o1` ...[6]のデータを `Initial.dat` に変更して持ってくる．
   (b) `parameter.dat`

```
0.000000   ;  ds              ; パラメータの変化量なし
      -1   ;  KellerEqn       ; 擬似弧長法の設定なし
              ..........
       2   ;  SolSelect       ; 進行解を選択
              ..........
       1   ;  MaxStep         ; 最大 step 回数
              ..........
       2   ;  Initial         ; (*1)
```

(*1) 分岐点から伸びる解の枝を求める

   と変更する．
   (c) `0_BranchChangeParameter.dat`
   分岐点における零固有値に対する固有関数の情報と `0_BranchChangeParameter.dat` のパラメータを用いて Newton 法に用いる初期値 $(\boldsymbol{u}_{ini}, c_{ini}, p_{ini})$ を構成する．
   - 初期値の構成 $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon\boldsymbol{\Phi}_R$,
   - 速度の構成 $c_{ini} = \delta_c$,
   - パラメータの構成 $p_{ini} = \delta_p$,

[5] ここにはコントロールパラメータの値が入っている
[6] ここには固有値実部の値が入っている

ただし，$\boldsymbol{u}_0$ は定常解の pitch-fork 分岐点における解 $\boldsymbol{u}$，$\boldsymbol{\Phi}_R$ は分岐点における零固有値に対する固有関数の実部，$\varepsilon, \delta_c, \delta_p$ はそれぞれ `0_BranchChangeParameter.dat` における `eps` , `velocity_delta` , `par_delta` である．
ここでは，

```
      0.08  ;  eps             ; u_ini の構成に必要
      0.00  ;  period_delta    ; ここでは不必要な設定
       0.0  ;  velocity_delta  ; 速度の微小変量
0.00000000  ;  par_delta       ; パラメータの微小変量
     0.000  ;  time            ; ここでは不必要な設定
```

と変更する．

2. 実行

   `$ ./aout␣1` ↵ で実行する

デモプログラム **2.1**

Step2.1 の設定から実行までの操作は

`$ cp␣../DEMO/Gray-Scott_Model/AS/06_TP1/*␣./` ↵

`$ ./aout␣1` ↵

で行える．

`RD/DEMO/Gray-Scott_Model/AS/06_TP1` には Step2.1 の設定で必要なファイル．

- `Function.c`
- `Initial.dat`
- `parameter.dat`
- `0_BranchChangeParameter`

が存在する．

3. 実行結果

```
  ...........

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   6 / 30
  Control Parameter    =     2.300901915266015
  Velocity             =     0.000337775542350
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000043803793

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   7 / 30
  Control Parameter    =     2.300901915266015
  Velocity             =     0.000337643953297
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000015729003


  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   8 / 30
  Control Parameter    =     2.300901915266015
  Velocity             =     0.000337596714033
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000005653990
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
```

収束した解は速度を持っているため，進行解であるとわかる．

Newton法による計算が発散したり，進行解と異なる別の解が求まったときは，0_BranchChangeParameterの各パラメータを調整 (eps を大きくするなど) して，進行解が求まるまで再実行する．

データの保存

Step2.1 で求めた分岐図のデータを TP_INI というディレクトリに保存する．

```
$ ./aout␣TP_INI ↵
```

Step2.2 進行解の分岐計算

Step2.1 で収束した解を用いてパラメータを変化させながら進行解の枝を求める．

1. 実行前の設定
   (a) Initial.dat
   初期値は Step2.1 で収束させた解を用いる．
   (b) parameter.dat

```
 0.01   ;  ds              ; パラメータの変化量 0.01
            ...........
  200   ;  MaxStep         ; 最大 step 回数
    1   ;  StepCut         ; 1step ごとにデータを出力する
            ...........
    1   ;  Initial         ; 分岐図を求めることを選択
```

と変更する.

2. 実行

   $ cp␣TP_INI/middle_final.dat␣Initial.dat ↵

   $ ./aout␣1 ↵

デモプログラム **2.2**

Step2.2 の設定から実行までの操作は

$ cp␣../DEMO/Gray-Scott_Model/AS/07_TP2/*␣./ ↵

$ ./aout␣1 ↵

で行える.

RD/DEMO/Gray-Scott_Model/AS/07_TP2 には Step2.2 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する.

3. 実行結果

```
.....
Step  /  MaxStep     =   200 / 200
iter  /  MaxIter     =   0 / 30
Control Parameter    =     4.300901915265973
Velocity             =     0.001219755984295
MachineEps           =     0.000000010000000
CC_Norm              =     0.000036275013261

Step  /  MaxStep     =   200 / 200
iter  /  MaxIter     =   1 / 30
Control Parameter    =     4.300901915265973
Velocity             =     0.001220913746472
MachineEps           =     0.000000010000000
CC_Norm              =     0.000000033525231

Step  /  MaxStep     =   200 / 200
iter  /  MaxIter     =   2 / 30
Control Parameter    =     4.300901915265973
Velocity             =     0.001220904601074
MachineEps           =     0.000000010000000
CC_Norm              =     0.000000000157812
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
```

進行解の分岐図を表示すると図 5.2.12 のようになる.

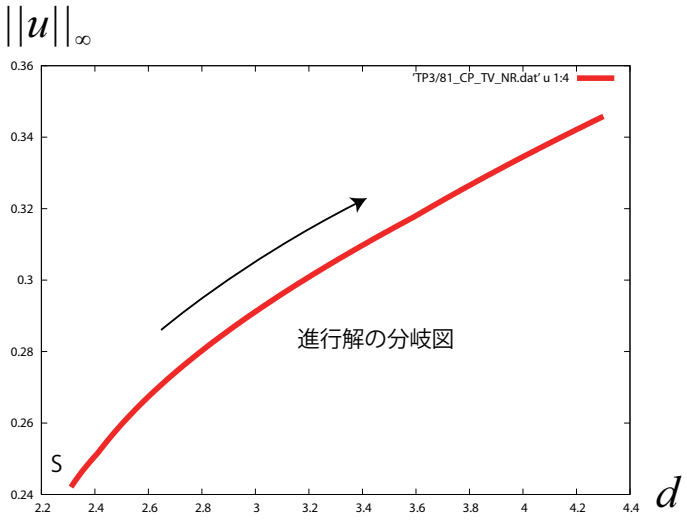


図 5.2.12: 進行解の分岐図

図 5.2.12 を定常解とあわせて表示すると次のようになる (※はデモでは求めていない部分の進行解の枝).

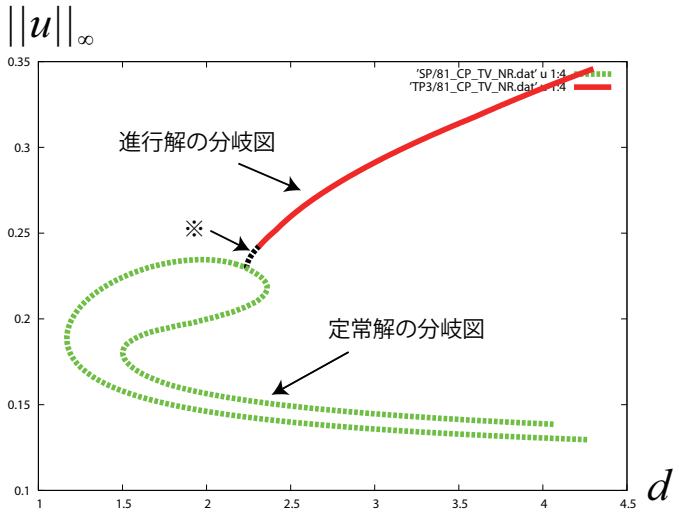


図 5.2.13: 進行解の分岐図

Step1.6 と Step2.3 で求める安定性を考慮し，縦軸を $\|u\|_{L^2}$ にすると図 5.2.14 のようになる．

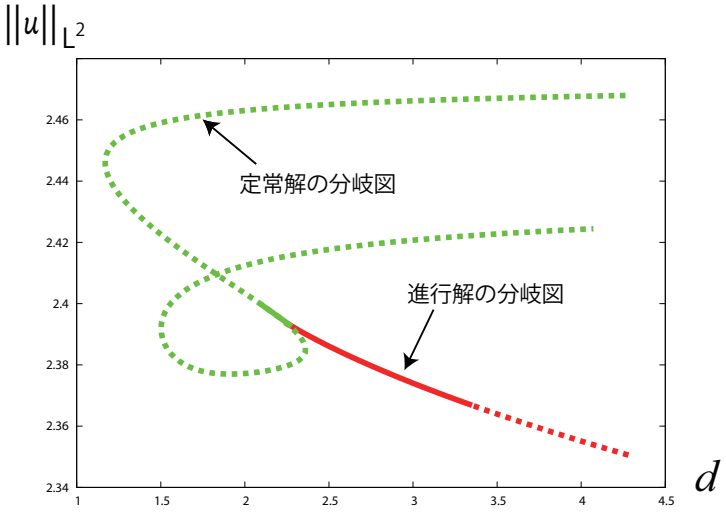


図 5.2.14: 進行解の分岐図

4. データの保存

   Step2.2 で求めた分岐図のデータを TP というディレクトリに保存する．

   `$ ./aout␣TP` ↵

Step2.3 進行解の分岐計算

ここでは Step2.2 で求めた進行解の安定性を求める方法について説明する．

1. 実行前の設定

   (a) `Initial.dat`

   まず，Step2.2 で保存したディレクトリ TP の中から固有値計算に必要なデータ `9middle_final.dat` を `Initial.dat` に変更して計算するディレクトリに持ってくる．

$ cp␣TP/9middle_final.dat␣Initial.dat

(b) parameter.dat

```
                .....
        8   ;   Initial          ; 安定性の計算を選択
```

と変更する．

2. 実行

$ ./aout␣1 ↵ で実行する．

デモプログラム **2.3**

Step2.3 の設定から実行までの操作は

$ cp␣../DEMO/Gray-Scott_Model/AS/08_TP3/*␣./ ↵

$ ./aout␣1 ↵

で行える．

RD/DEMO/Gray-Scott_Model/AS/08_TP3 には Step2.3 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat のデータから横軸にステップ数，縦軸に固有値実部をグラフで示すと図 5.2.15 となる．

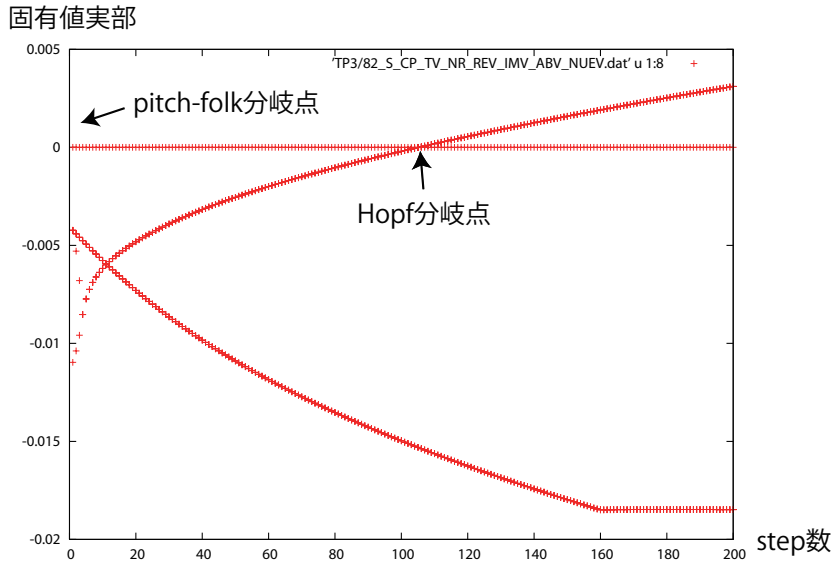


図 5.2.15:

進行解の安定性は 120 ページの定常解の安定性と同様に調べる．デモプログラムを使

用した場合は，106step で 82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat の固有値の実部が 2 つ 0 を横切るため Hopf 分岐が存在する．Hopf 分岐点からは脈動進行解の枝が存在する．

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値．

```
                          ·····
105  ·····················  1.02512634E-08  0.00000000E+00  ·····
105  ·····················  -7.69879710E-06 -1.61517239E-02  ·····
105  ·····················  -7.69879710E-06  1.61517239E-02  ·····
105  ·····················  -1.53054324E-02 -3.86482927E-02  ·····
105  ·····················  -1.53054324E-02  3.86482927E-02  ·····
106  ·····················  3.08429993E-05  1.61766085E-02  ·····
106  ·····················  3.08429993E-05 -1.61766085E-02  ·····
106  ·····················  1.02497315E-08  0.00000000E+00  ·····
106  ·····················  -1.53711572E-02  3.87068421E-02  ·····
106  ·····················  -1.53711572E-02 -3.87068421E-02  ·····
                          ·····
```

4. データの保存

   Step2.3 で求めた分岐図のデータを TP というディレクトリに上書きする．

   $ ./aout␣TP ↵

   このとき，ディレクトリを上書きしますかと尋ねられるので，$ yes ↵ と入力する．

Step3.1 脈動進行解の分岐計算

ここでは Step2.3 で求めた進行解の Hopf 分岐点の固有関数の情報を用いて脈動進行解を求める．

1. 実行前の設定

   (a) Initial.dat

   Hopf 分岐点の固有関数を Initial.dat として使用する．固有ベクトルのデータは 6_EigenVector_Data に入っている．6_EigenVector_Data/ 106_CP ...[7]/o1 ...[8]のデータを Initial.dat に変更して持ってくる．

   (b) parameter.dat

[7]ここにはコントロールパラメータの値が入っている

[8]ここには固有値実部の値が入っている

```
0.000000   ;  ds              ; パラメータの変化量なし
              ...........
       4   ;  SolSelect      ; 脈動進行解を選択
              ...........
       1   ;  StepCut        ; 毎 step ごとにデータを出力する
       1   ;  MaxStep        ; 最大 step 回数
              ...........
       2   ;  Initial        ; (*1)
(*1) 分岐点から伸びる解の枝を求める.
```

と変更する.

(c) `0_BranchChangeParameter.dat`

分岐点における零固有値に対する固有関数の情報と `0_BranchChangeParameter.dat` のパラメータを用いて Newton 法に用いる初期値 $(\boldsymbol{u}_{ini}, T_{ini}, c_{ini}, p_{ini})$ を構成する.

- 初期値の構成 $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon(\sin(2\pi t)\boldsymbol{\Phi}_R + \cos(2\pi t)\boldsymbol{\Phi}_I)$,
- 周期の構成 $T_{ini} = \dfrac{2\pi}{\beta} + \delta_T$
- 速度の構成 $c_{ini} = \delta_c$,
- パラメータの構成 $p_{ini} = \delta_p$,

ただし, $\boldsymbol{u}_0$ は定常解の pitch-fork 分岐点における解 $\boldsymbol{u}$, $\boldsymbol{\Phi}_R, \boldsymbol{\Phi}_I$ はそれぞれ分岐点における零固有値に対する固有関数の実部と虚部, c は解の速度, T は解の周期 p はコントロールパラメータ, c_0 は進行解の Hopf 分岐点における速度, $\varepsilon, \delta_T, \delta_c, \delta_p, t$ はそれぞれ `0_BranchChangeParameter.dat` における `eps` , `period_delta` , `velocity_delta` , `par_delta` , `time` である.

ここでは,

```
      0.05  ;  eps            ;    SS_2  &  SS_3  &  SS_4
      0.00  ;  period_delta    ;          &  SS_3  &  SS_4
       0.0  ;  velocity_delta  ;    SS_2  &        &  SS_4
0.00000000  ;  par_delta       ;    SS_2  &  SS_3  &  SS_4
     0.000  ;  time            ;          &  SS_3  &  SS_4
```

と変更する.

2. 実行

`$ ./aout␣1` ↵ で実行する

デモプログラム **3.1**

Step3.1 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/AS/09_TB1/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Gray-Scott_Model/AS/09_TB1 には Step3.1 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat
- 0_BranchChangeParameter

が存在する.

3. 実行結果

```
  ...........

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   7 / 30
  Control Parameter    =     3.360901915265992
  Period               =   394.759258820564980
  Velocity             =     0.001064648438453
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000060381561

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   8 / 30
  Control Parameter    =     3.360901915265992
  Period               =   394.759140363937661
  Velocity             =     0.001064648372300
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000013427036

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   9 / 30
  Control Parameter    =     3.360901915265992
  Period               =   394.759166707443910
  Velocity             =     0.001064648387011
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000002985457
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
  ...........
```

ただし，プログラムのバグで「Convergence !!」と表示されていても，周期 (Period) が負になっていたり，極端に大きな値になっていたりする場合は，求めたい解に収束していないことに注意する．

Step3.2 脈動進行解の分岐計算

定常解，進行解のときと同様に脈動進行解の分岐図を求める．

脈動進行解の枝を追うのは非常に難しい．小まめにデータを保存しながら少しずつ進めていく必要がある．そのためデモプログラムは省略する．

分岐図は以下のようになる．

Step4.1 脈動定常解の分岐計算

ここでは，Step1.6 で求めた定常解の Hopf 分岐点における固有関数の情報を用いて脈動定常解を求める．方法は Step2.1 とほぼ同様であるが，固有関数は保存されていないため求めな

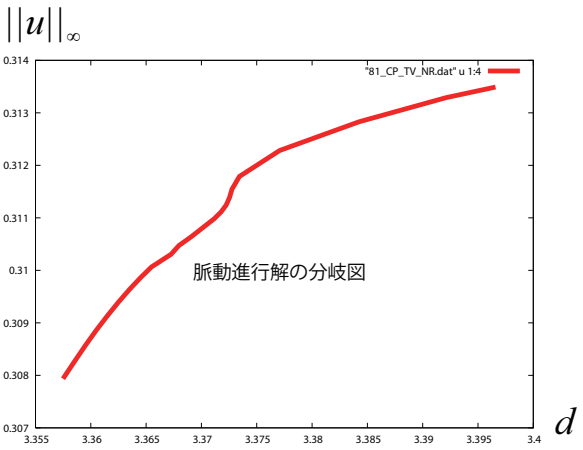


図 5.2.16:

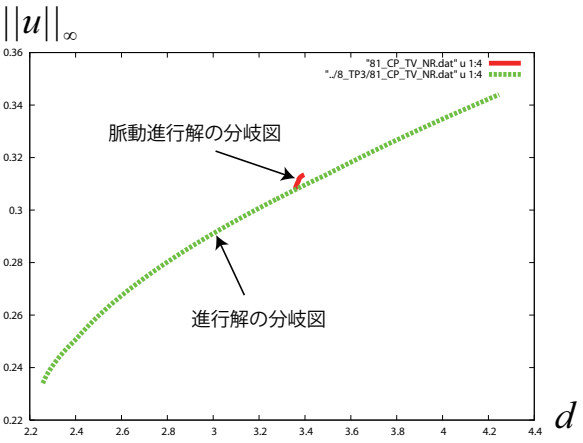


図 5.2.17:

おす必要がある．Step4.1 では，Hopf 分岐点における固有関数を求めなおす．まず，Step1.6 で保存したデータを AS にもってくる．

```
$ cp␣SP/*␣./ ↵
```

Step1.6 でも示したように，最初に現れる Hopf 分岐点での固有値の情報は以下のようになっている．

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値．

```
                          ·····
95  ·····················  3.39914803E-04  2.62426635E-02  ·····
95  ·····················  3.39914803E-04 -2.62426635E-02  ·····
95  ·····················  1.89058850E-08  0.00000000E+00  ·····
95  ·····················  -5.28527110E-03  0.00000000E+00  ·····
95  ·····················  -1.80952017E-02  0.00000000E+00  ·····
96  ·····················  2.04794297E-08  0.00000000E+00  ·····
96  ·····················  -4.20137537E-05 -2.64192278E-02  ·····
96  ·····················  -4.20137537E-05  2.64192278E-02  ·····
96  ·····················  -4.86485952E-03  0.00000000E+00  ·····
96  ·····················  -1.80979881E-02  0.00000000E+00  ·····
                          ·····
```

95step では正の値であった 1 列目と 2 列目の固有値実部の値が，96step では 2 列目と 3 列目に負の値として記録されていることがわかる．したがって，ここでは 95step の固有関数を求める．

1. 実行前の設定
   (a) `Initial.dat`
   Hopf 分岐点の `9middle_final.dat` を `Initial.dat` として使用する．`9middle_final.dat` を任意のテキストエディタで開き，95step 以外の全ての値を消去し，保存する．その後，`9middle_final.dat` を `Initial.dat` に変更する (`$ cp␣9middle_final.dat ␣Initial.dat` ↵).
   (b) `parameter.dat`

```
                    ·····
        8    ;  Initial          ; (*1)
(*1) 各 step における解の固有値と固有関数の計算を行う
```

   と変更する．
2. 実行
   `$ ./aout␣1` ↵ で実行する

デモプログラム **4.1**

Step4.1 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/AS/11_SB1/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Gray-Scott_Model/AS/11_SB1 には Step4.1 の設定で必要なファイル．

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

```
  ............


-----------------------------------------------------------------
  Step  /  MaxStep             =  1 / 1
  Control Parameter            =   1.869650891473
  Nearly Most Zero EigenValue  =   0.000000018906
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  REV[1] = 0.000339914803  IEV[1] = 0.026242663526   RI[1] = 0.026244864851
  REV[2] = 0.000339914803  IEV[2] = -0.026242663526  RI[2] = 0.026244864851
  REV[3] = 0.000000018906  IEV[3] = 0.000000000000   RI[3] = 0.000000018906
-----------------------------------------------------------------
```

Step4.2 脈動定常解の分岐計算

ここでは，Step4.1 で求めた定常解の Hopf 分岐点における固有関数の情報を用いて脈動定常解を求める．

1. 実行前の設定

   (a) Initial.dat
   Hopf 分岐点の固有関数を Initial.dat として使用する．固有ベクトルのデータは 6_EigenVector_Data に入っている．
   6_EigenVector_Data/ 1_CP...[9]/o1...[10]のデータを Initial.dat に変更して持ってくる．

   (b) parameter.dat

[9] ここにはコントロールパラメータの値が入っている
[10] ここには固有値実部の値が入っている

```
0.000000   ;  ds              ; パラメータの変化量なし
       1   ;  KellerEqn      ; 擬似弧長法の設定あり
              ...........
       3   ;  SolSelect      ; 脈動定常解を選択
              ...........
       1   ;  MaxStep        ; 最大 step 回数
              ...........
       2   ;  Initial        ; (*1)
(*1) 分岐点から伸びる解の枝を求める
```

と変更する.

(c) `0_BranchChangeParameter.dat`
分岐点における零固有値に対する固有関数の情報と `0_BranchChangeParameter.dat` のパラメータを用いて Newton 法に用いる初期値 $(\boldsymbol{u}_{ini}, p_{ini})$ を構成する.

- 初期値の構成 $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon(\sin(2\pi t)\boldsymbol{\Phi}_R + \cos(2\pi t)\boldsymbol{\Phi}_I)$,
- 周期の構成 $T_{ini} = \frac{2\pi}{\beta} + \delta_T$,
- パラメータの構成 $p_{ini} = \delta_p$,

ただし，$\boldsymbol{u}_0$ は定常解の Hopf 分岐点における解 $\boldsymbol{u}$，$\boldsymbol{\Phi}_R, \boldsymbol{\Phi}_I$ はそれぞれ分岐点における零固有値に対する固有関数の実部と虚部，T は解の周期 p はコントロールパラメータ，β は定常解の Hopf 分岐点における固有値虚部の値，$\varepsilon, \delta_T, \delta_p, t$ はそれぞれ `0_BranchChangeParameter.dat` における `eps` , `period_delta` , `par_delta` , `time` である. β:定常解の Hopf 分岐点における固有値虚部の値
ここでは,

```
      0.05  ;  eps             ; u_ini の構成に必要
      0.00  ;  period_delta    ; 周期の微小変量
       0.0  ;  velocity_delta  ; ここでは不必要な設定
0.00000000  ;  par_delta       ; パラメータの微小変量
     0.000  ;  time            ; u_ini の構成に必要
```

と変更する.

2. 実行

`$ ./aout␣1` ↵ で実行する

デモプログラム **4.2**

Step4.2 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/AS/12_SB2/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Gray-Scott_Model/AS/12_SB2 には Step4.2 の設定で必要なファイル．

- Function.c
- Initial.dat
- parameter.dat
- 0_BranchChangeParameter

が存在する．

3. 実行結果

```
  ...........

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   4 / 30
  Control Parameter    =     1.994435415511887
  Period               =   259.665896228138479
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000004054062691

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   5 / 30
  Control Parameter    =     1.994440834482065
  Period               =   259.670083565151799
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000050642485

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   6 / 30
  Control Parameter    =     1.994440898624708
  Period               =   259.670129810254650
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000454148
------------------------------------------------------------
                       Convergence  !!
------------------------------------------------------------
```

収束した解は周期を持っているため，脈動解であるとわかる．

今回のように，脈動解の場合は擬似弧長法の設定を入れると解を収束させやすい．

データの保存

Step4.2 で求めた分岐図のデータを SB_INI というディレクトリに保存する．

```
$ ./aout␣SB_INI ↵
```

Step4.3 脈動定常解の分岐計算

Step4.2 で収束した解を用いてパラメータを変化させながら脈動定常解の枝を求める．

1. 実行前の設定
   (a) Initial.dat
   初期値は Step4.2 で収束させた解を用いる．
   (b) parameter.dat

```
-0.00005   ;  ds            ; パラメータの変化量 −0.00005
              ...........
       1   ;  EigenValue    ; 固有値の計算あり
       1   ;  EigenVector   ; 固有ベクトル計算あり
              ...........
       1   ;  StepCut       ; 1step ごとにデータを出力する
     100   ;  MaxStep       ; 最大 step 回数
              ...........
       1   ;  Initial       ; 分岐図を求めることを選択
```

   と変更する．
2. 実行

```
$ cp␣SB_INI/middle_final.dat␣Initial.dat ↵
$ ./aout␣1 ↵
```

> デモプログラム **4.3**
>
> Step4.3 の設定から実行までの操作は
>
> ```
> $ cp␣../DEMO/Gray-Scott_Model/AS/13_SB3/*␣./ ↵
> $ ./aout␣1 ↵
> ```
>
> で行える．
>
> RD/DEMO/Gray-Scott_Model/AS/13_SB3 には Step4.3 の設定で必要なファイル
>
> - Function.c
> - Initial.dat
> - parameter.dat
>
> が存在する．

3. 実行結果
   途中で計算が止まってしまう．

```
  ............


------------------------------------------------------------
  Step  /  MaxStep             =  50 / 100
  Control Parameter            =   2.190080118610
  Nearly Most One EigenValue   =   0.961863982654
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  REV[1] = -1776.464390405427 IEV[1] = 0.000000000000   RI[1] = 1776.464390405427
  REV[2] = 11.796518082181 IEV[2] = 0.000000000000   RI[2] = 11.796518082181
  REV[3] = 2.095899744166  IEV[3] = 0.000000000000   RI[3] = 2.095899744166
------------------------------------------------------------

  Step  /  MaxStep     =   51 / 100
  iter  /  MaxIter     =   0 / 50
  Control Parameter    =     2.190080118609975
  Period               =   469.503536462149782
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000121647676807

  Step  /  MaxStep     =   51 / 100
  iter  /  MaxIter     =   1 / 50
  Control Parameter    =     2.184345104892049
  Period               =   481.258229677027430
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.060014529587429

  Step  /  MaxStep     =   51 / 100
  iter  /  MaxIter     =   2 / 50
  Control Parameter    =     2.171010326160396
  Period               =   489.541509841286711
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.235433907099954

  Step  /  MaxStep     =   51 / 100
  iter  /  MaxIter     =   3 / 50
  Control Parameter    =     1.698490166970823
  Period               =  -438.571793196803696
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000000000
------------------------------------------------------------
                       Convergence  !!
------------------------------------------------------------
```

「Convergence !!」と表示されてはいるが，周期の値が負になっている．横軸に step 数，縦軸に固有値の実部をとる (図 5.2.18) と，ひとつの固有値の実部が負の方向に大きく

なりすぎているために計算が止まったのだとわかる.

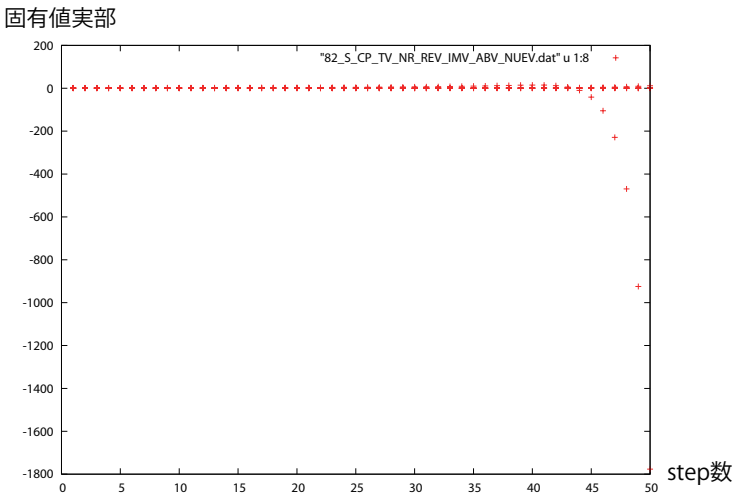


図 5.2.18: 固有値実部

脈動定常解の分岐図は以下のようになる.

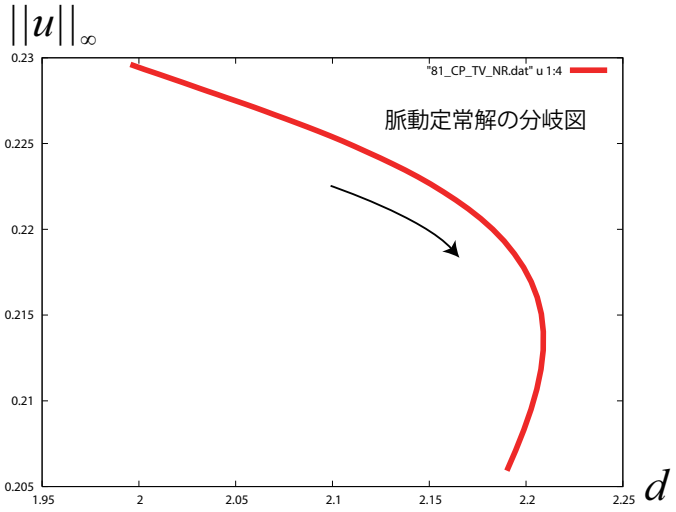


図 5.2.19: 定常解と脈動定常解の分岐図

図 5.2.19 を定常解とあわせて表示すると次のようになる (※はデモでは求めていない部分の脈動定常解の枝).

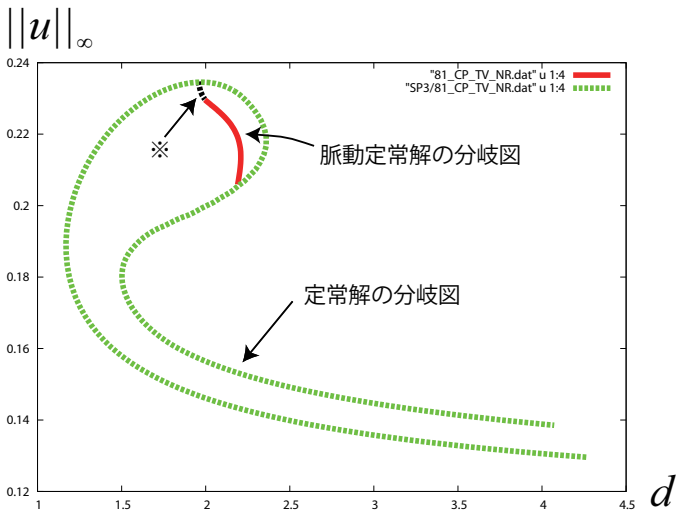


図 5.2.20: 定常解と脈動定常解の分岐図

4. データの保存

   Step4.3 で求めた分岐図のデータを SB というディレクトリに保存する.

   `$ ./aout␣SB` ↵

Appendix1.1 脈動進行解の分岐計算

Appendix1.1 では，5.1 節で求めた Step3.2 の解を用いて，脈動進行解を求める.

時間発展解で求めた脈動進行解を Newton 法の初期値として与える場合は脈動進行解の速度と周期を求める必要がある．速度は適切な `Velocity_Site` と `Velocity_Measure_Start` (13 ページ参照) を与えて求め，周期は 23 ページで説明した方法を用いる．Appendix1.1 では，この一連の作業について説明する.

まず，カレントディレクトリを時間発展方程式を求めるところに移動 (`$ cd␣RD/TE` ↵) し，3.2 節の Step3.2 で保存した全てのデータ `TB/*.dat` をカレントディレクトリに持ってくる (`$ cp␣TB/*␣./`). これによって，3.2 節の Step3.2 を実行したあとの状態が復元できる．その後 `final.dat` 開くと，下から 5 行の 1 列目と 5 列目は以下のようになっていることがわかる.

```
 ..........
30000 .......... .......... .......... 9.999979890995608E-01
30000 .......... .......... .......... 9.999978794605785E-01
30000 .......... .......... .......... 0.000000000000000E+00 ← 周期
30000 .......... .......... .......... 1.074709848399322E-03 ← 速度
30000 .......... .......... .......... 0.000000000000000E+00
```

1. 解の速度について

   適切な `Velocity_Site` と `Velocity_Measure_Start` を選択してあれば `final.dat` の左から 5 列目，下から 2 番目のところに速度の値が入っている.

(a) Velocity_Site
脈動進行解は解が上下に振動するため，パルスの高さに注意して Velocity_Site はパルス波のフロントとバックが常に存在する位置に設定する．デモでは 0.2 と与えている．

(b) Velocity_Measure_Start
Velocity_Measure_Start は MaxStep より小さな値を与える．

2. 解の周期について
final.dat の左から 5 列目，下から 3 番目には脈動進行解の周期を入れる必要がある．

(a) extremum を使う
$ ./extremum ↵
と実行すると，以下のような画面が表示される．

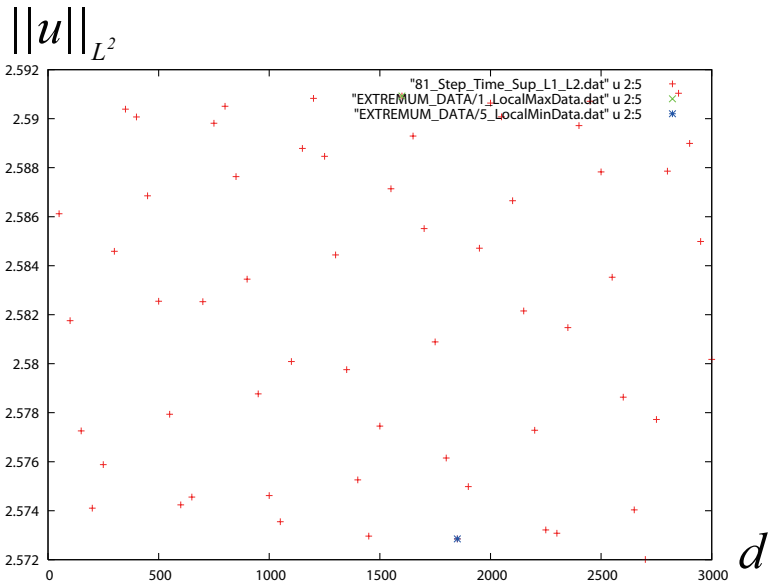


図 5.2.21:

しかし，この図では極大値と極小値が求められていない．これは StepCut が大きいゆえに，時系列データが粗くて極大値と極小値が求められていない (脈動解の周期が求められていない)．したがって，StepCut を小さくし，3.2 節の Step3.2 を再計算する．StepCut を小さくすると gnuplot 画面の表示が追いつかないので，gnuplot 画面の表示を変更する．

(b) StepCut を調整して再計算する
parameter.dat を

```
            ............
  200   ;   Gnuplot_CutN_Vel_Dis     ; (*1)
            ............
  200   ;   StepCut                  ; 200 回おきにデータを出力
            ............
 1245   ;   Gnuplot_Select           ; gnuplot 画面の 3,6 番を表示
            ............
```

(*1) 速度データを 200 回おきに出力する

と変更する．Initial.dat は 3.2 節の Step3.2 で使用した Initial.dat を使用する．
$ ./aout␣1 ↵
で実行する．

デモプログラム **A1.1**

Appendix1.1(b) の設定から実行までの操作は

$ cp␣../DEMO/Gray-Scott_Model/TE/05_TB3/*␣./ ↵

$ ./aout␣1 ↵

で行える．

RD/DEMO/Gray-Scott_Model/TE/05_TP3 には Appendix1.1(b) の設定で必要なファイル．

- 0parameter.dat
- Function.c
- parameter.dat
- Initial.dat
- gnuplot.c

が存在する．

(c) extremum を使う
再び extremum を使うと以下の図が表示される．

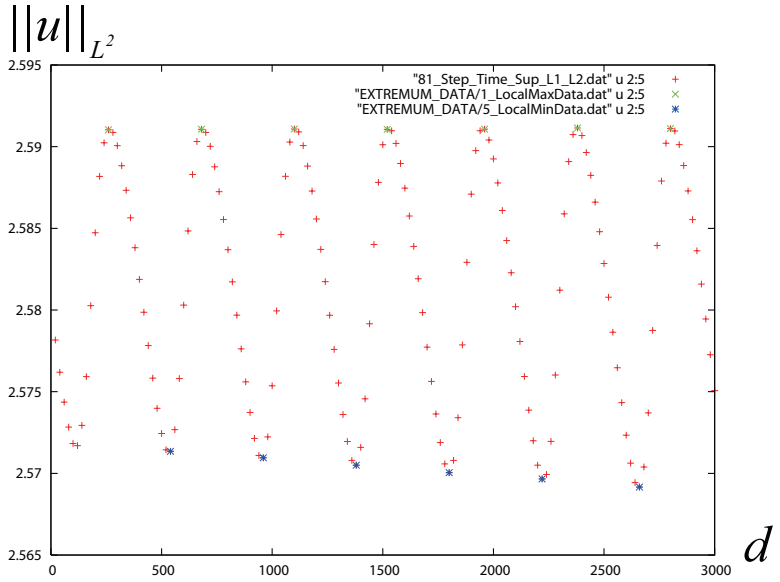


図 5.2.22:

4_LocalMaxAvePeriod.dat を開くと極大値の時間間隔値が,

```
423.3333333333
```

とわかるので，この値を脈動解の周期に用いる．極小値の時間間隔値 8_LocalMinAvePeriod.dat を用いることもできる．

(d) final.dat に入力

final.dat の左から 5 列目，下から 3 番目の 0.0 を周期の値に書き換える．

```
.....
30000 ..... ..... ..... 9.999979890995608E-01
30000 ..... ..... ..... 9.999978794605785E-01
30000 ..... ..... ..... 423.3333333333
30000 ..... ..... ..... 1.074709848399322E-03
30000 ..... ..... ..... 0.000000000000000E+00
```

Appendix1.2 脈動進行解の分岐計算

Appendix1.1 で求めた final.dat を Newton 法の初期値に与え脈動進行解を求める．カレントディレクトリを分岐図を求めるところに移動 ($ cd␣../AS ↵) する．

1. Initial.dat
   時間発展方程式で求めた final.dat を Initial.dat として使用する．
2. parameter.dat

```
0.000000   ;  ds              ; パラメータの変化量なし
              ...........
       4   ;  SolSelect      ; 脈動進行解を選択
              ...........
       1   ;  StepCut        ; 毎 step ごとにデータを出力する
       1   ;  MaxStep        ; 最大 step 回数
              ...........
       1   ;  Initial        ; 分岐図を求めることを選択
```

と変更する.

3. 実行

   $ cp␣../TE/final.dat␣Initial.dat ↵

   $ ./aout␣1 ↵

   で実行する.

デモプログラム **A1.2**

Appendix1.2 の設定から実行までの操作は

$ cp␣../DEMO/Gray-Scott_Model/AS/11_AP1/*␣./ ↵

$ ./aout␣1 ↵

で行える.

RD/DEMO/Gray-Scott_Model/AS/11_AP1 には Appendix1.2 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat

が存在する.

4. 実行結果

```
  ............
  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   0 / 30
  Control Parameter    =     3.380000000000000
  Period               =   423.333333333300004
  Velocity             =     0.001074709848399
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.010293873403380

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   1 / 30
  Control Parameter    =     3.380000000000000
  Period               =  3495.413967540044723
  Velocity             =     0.001649339719272
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000141766
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
```

「Convergence !!」と表示されているが，周期が明らかに大きすぎる値になっているので脈動進行解の枝に収束していないことがわかる．

Appendix1.3 脈動進行解の分岐計算

Appendix1.2 と同じ Initial.dat を初期値として再計算を行う．Appendix1.2 のようにうまく収束しなかった場合は，擬似弧長法を入れて試してみるとよい．

1. Initial.dat

   Appendix1.2 と同じものを用いる．

2. parameter.dat

```
              ............
        1   ;  KellerEqn       ; 擬似弧長法の設定あり
              ............
```

   と変更する．

3. 実行

   $ ./aout␣1 ↵

   で実行する．

デモプログラム **A1.3**

Appendix1.3 の設定から実行までの操作は

```
$ cp␣../DEMO/Gray-Scott_Model/AS/12_AP2/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Gray-Scott_Model/AS/12_AP2 には Appendix1.2 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat

が存在する.

4. 実行結果

```
  ...........

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   2 / 30
  Control Parameter    =     3.372113877407729
  Period               =   423.332759369808628
  Velocity             =     0.001076007147629
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000115980494426

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   3 / 30
  Control Parameter    =     3.372366125499470
  Period               =   423.332638193281355
  Velocity             =     0.001076063757467
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000136682410

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   4 / 30
  Control Parameter    =     3.372366652005462
  Period               =   423.332638008723507
  Velocity             =     0.001076063893677
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000135279
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
```

時間発展方程式の解を初期値として，脈動進行解の枝を求めることができた．脈動解の枝を求めるときも同様である．

第6章　デモ2

この章では,RD-AUTO を用いて 1 次元発熱反応モデルの分岐図を描く工程を説明する．分岐の種類についても調べる．
このデモに必要なデータは DEMO ディレクトリの中に図 6.0.1 のように入っている．

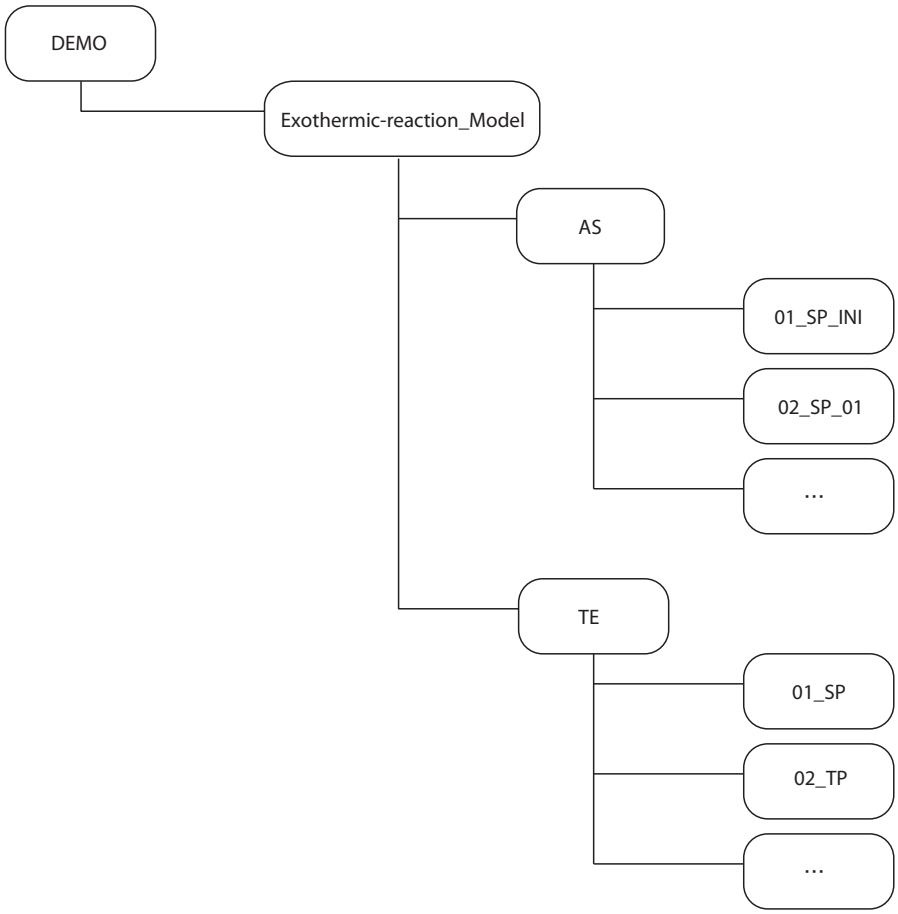


図 6.0.1: DEMO の中身

デモプログラム	内容	実行後の保存先
01_SP	定常解 (TE)	SP
02_TP	進行解 (TE)	TP
03_TB1	脈動進行解 (TE)	
04_TB2	脈動進行解 (TE)	TB
01_SPINI	定常解 (AS)	SP_INI
02_SP01	定常解 (AS)	SP_01
03_SP02	定常解 (AS)	SP_02
04_SP03	定常解 (AS)	SP_03
05_SP04	定常解 (AS)	SP_04
06_SP05	定常解 (AS)	SP_05
07_SP06	定常解 (AS)	SP_06
08_SP07	定常解 (AS)	SP_F
09_TP1_01	進行解 (AS)	TP1_F
10_TP2_INI	進行解 (AS)	TP2_INI
11_TP2_01	進行解 (AS)	TP2_F
12_TP2_02	進行解 (AS)	TP2_F
13_TB3_01	進行解 (AS)	
14_TB3_02	進行解 (AS)	
15_TB3_03	進行解 (AS)	
16_TB3_04	進行解 (AS)	
17_TB3_05	進行解 (AS)	TP3_F
18_TB_INI	脈動進行解 (AS)	TB_INI
19_TB_01	脈動進行解 (AS)	TB_01
20_TB_02	脈動進行解 (AS)	TB_02
21_TB_03	脈動進行解 (AS)	TB_03
22_TB_04	脈動進行解 (AS)	TB_04
23_TB_05	脈動進行解 (AS)	TB_05

6.1 時間発展方程式

$$
\left\{
\begin{array}{l}
\left.
\begin{array}{ll}
u_t = \dfrac{4\pi^2}{L^2}\Delta u + \dfrac{1}{\epsilon}(-au + k(u)v), & t > 0, \quad 0 < x < 2\pi, \\
v_t = \dfrac{4\pi^2}{L^2}d_v\Delta v + h(1-v) - k(u)v, &
\end{array}
\right\} (*1) \\
\left.
\begin{array}{l}
k(u) = \exp\left(\dfrac{u}{1+u/c}\right), \\
u(t,0) = u(t,2\pi), \quad u_x(t,0) = u_x(t,2\pi), \quad t > 0, \\
v(t,0) = v(t,2\pi), \quad v_x(t,0) = v_x(t,2\pi), \quad t > 0,
\end{array}
\right\} (*2) \\
\left.
\begin{array}{l}
u(0,x) = u_0(x), \quad 0 \le x \le 2\pi, \\
v(0,x) = v_0(x), \quad 0 \le x \le 2\pi
\end{array}
\right\} (*3)
\end{array}
\right.
$$

この節では，1 次元発熱反応モデルに現れるパルス波を求める．カレントディレクトリは時間発展方程式を解くところに移動 (`$ cd␣RD/TE` ↵) して，最初に方程式と境界条件の設定を行う．

①方程式 (∗1) の設定

方程式の拡散項と反応項の設定は `Function.c` で行う．拡散項は関数 `Diffusion` を

```
void  Diffusion( TEMP TP, double *par )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  par[TP.N_EP+1]  =  (4.0*M_PI*M_PI*par[1]) / (par[7]*par[7]);
  par[TP.N_EP+2]  =  (4.0*M_PI*M_PI*par[2]) / (par[7]*par[7]);
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述して，反応項は関数 `Reaction` を

```
void  Reaction( TEMP TP, double *par, double *x, double *F )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  int i;
  int N = TP.N;
  for( i = 1; i <= N; i++ ){
    F[i] = ((-par[4]*x[i])+exp(x[i]/(1.0+(x[i]/par[5])))*x[i+N])/par[3];
    F[i+N] = (par[6]*(1.0-x[i+N]))-exp(x[i]/(1.0+(x[i]+par[5]))*x[i+N]);
  }
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述する．

②境界条件の設定

境界条件の設定は `parameter.dat` を開き，`BC_Select` を

```
            ............
      3   ;  BC_Select      ;  OP[9]
            ............
```

と与える．BC_Select = 3 は周期境界条件を表す．

③①〜②以外の設定

①〜②以外の設定はパルス波の種類によって異なるため，それらの設定はパルス波を求めるところで逐次説明していく．

Step1 定常パルス波を時間発展方程式を用いて求める

ステップ関数を初期値として定常パルス波を求める．

1. 実行前の設定

   (a) 0parameter.dat

   Function.c 内のパラメータの値を設定する．今回は以下のように設定する．

```
    1.0   ;  du  ; par[1]
    5.0   ;  dv  ; par[2]
    1.1   ;  eps ; par[3]
    0.07  ;  a   ; par[4]
    1.0   ;  c   ; par[5]
   45.0   ;  h   ; par[6]
     10   ;  Lx  ; par[7]
```

   (b) Function.c

   反応項は

$$f(u,v) = \frac{1}{\epsilon}(-au + k(u)v) \qquad (k(u) = \exp\left(\frac{u}{1+u/c}\right))$$

$$g(u,v) = h(1-v) - k(u)v$$

   であるので，関数 Reaction 内で

```
for( i = 1 ; i <= N ; i++ ){
      F[i]=((-par[4]*x[i])+exp(x[i]/(1.0+(x[i]/par[5])))*x[i+N])/par[3];
    F[i+N]=(par[6]*(1.0-x[i+N]))-exp(x[i]/(1.0+(x[i]+par[5]))*x[i+N]);
}
```

   と記述．

   (c) parameter.dat

   今回は以下のように設定する．

```
       256   ;  N                          ; 空間分割数は 256
       128   ;  N_Wave                     ; 切断波数は 128
        -1   ;  T_SELECT                   ; (*1)
        -1   ;  VEL_SELECT                 ; (*1)
   1.0E-01   ;  dt                         ; 時間の刻み幅は 10^-1
6.283185307179586;  Lx                     ; 区間は 2π
       100   ;  Gnuplot_CutN_Vel_Dis      ; (*2)
       0.2   ;  Velocity_Site             ; (*3)
     10000   ;  Velocity_Measure_Start    ; step10000 から速度を測る.
         3   ;  BC_Select                 ; 周期境界条件
         3   ;  redEE_Select              ; (*4)
       500   ;  StepCut                   ; 500 回おきにデータを出力
    100000   ;  MaxStep                   ; 最大 step 数
         0   ;  Gnuplot_Select            ; gnuplot 画面をすべて表示
         1   ;  Initial_Select            ; 初期値はステップ関数
```

(*1) 発展方程式は $\boldsymbol{u}_t = \boldsymbol{u}_{xx} + f(\boldsymbol{u}), \boldsymbol{v}_t = \boldsymbol{v}_{xx} + g(\boldsymbol{v})$ を解く
(*2) 速度データを 100 回おきに出力する
(*3) $\boldsymbol{u}$ のパルス波の高さ 0.2 のところで高さを測る
(*4)Spectral 法 (Runge-Kutta 法)

(d) `boundary.dat`
今回は周期境界条件のため，`BC[1]` ~ `BC[4]` は特に変更する必要はない.

(e) `initial_set.c`
初期値の設定を行う．97 ページの (*3) の初期値は図 6.1.1 のステップ関数を与える.

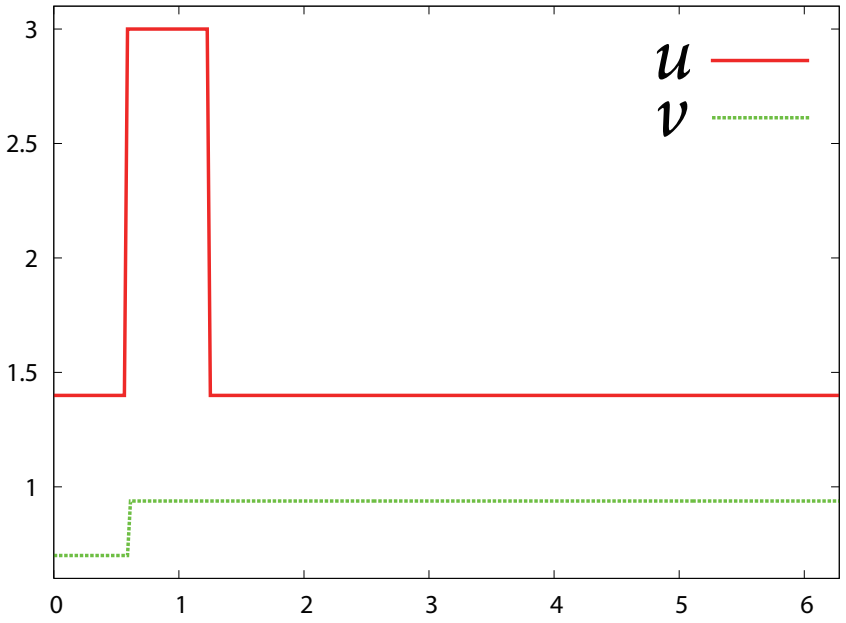


図 6.1.1:

よって，プログラムでは次のように設定する.

```
void  initial_set2( TEMP *TP, double *par, ····· )
{
  ···········
  if( IS  ==  1 ){
    for( i = 1; i <= N; i++ ){
      x[i  ]  =  1.3992660000;
      x[i+N]  =  0.9378105400;
      if( N/10 <= i  &&  N/5>= i )
        x[i]  =  5.0;
      if( 1 <= i  &&  N/10>= i )
        x[i+N]  =  0.4;
    }
  }
  else if( IS == 2  ||  IS == 3 ){
  ···········
  }
  ···········
}
```

(f) gnuplot.c
gnuplot で描写する領域を指定する．図 6.1.1 の縦軸の範囲を −1.0 ∼ 20.0 で設定する．

```
void  Gnuplot_InitialCommand( TEMP *TP, int *c_G )
{
 ···········
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
  fprintf( GP2, "set yr[-1.0:20.0]\n" );
  fprintf( GP3, "set yr[-1.0:20.0]\n" );
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
 ···········
}
```

2. 実行
3. $ ./aout␣1 ↵
   で実行する．

デモプログラム **1**

Step1 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/TE/01_SP/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Exothermic-reaction_Model/TE/01_SP には Step1 の設定で必要なファイル

- 0parameter.dat
- Function.c
- parameter.dat
- initial_set.c
- gnuplot.c

が存在する．

4. 実行結果

定常パルス波がでれば成功．

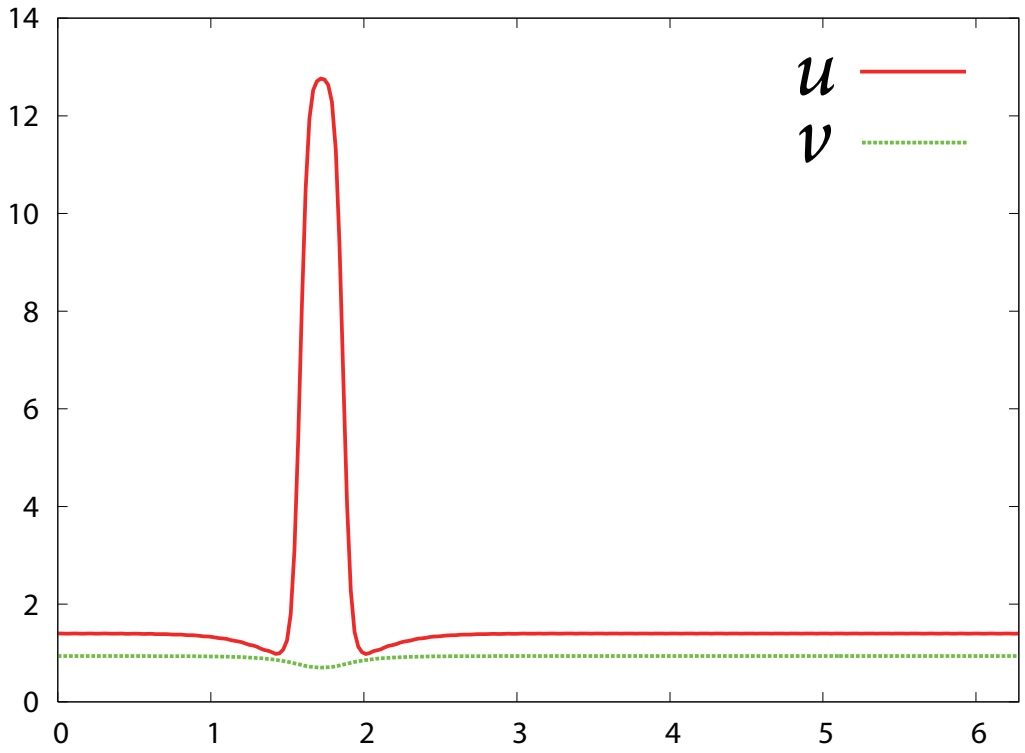


図 6.1.2: 定常パルス波

5. データの保存

ここで求めた定常パルス波のデータを SP というディレクトリの保存する．次のコマンドを実行することでデータをディレクトリ SP に保存できる．

$ ./aout␣SP ↵

Step2 進行パルス波

ステップ関数を初期値として，進行パルス波を求める．

1. 実行前の設定
   (a) 0parameter.dat

```
                .....
        3.0     ;  dv   ; par[2]
                .....
```

   と変更する．
   (b) parameter.dat
   変更点なし．
   (c) initial_set.c
   変更点なし．
2. 実行
3. $ ./aout␣1 ↵
   で実行する．

デモプログラム **2**

Step2 の設定から実行までの操作は

$ cp␣../DEMO/Exothermic-reaction_Model/TE/02_TP/*␣./ ↵

$ ./aout␣1 ↵

で行える．

RD/DEMO/Exothermic-reaction_Model/TE/02_TP には Step2 の設定で必要なファイル

- 0parameter.dat
- Function.c
- parameter.dat
- initial_set.c
- gnuplot.c

が存在する．

4. 結果
   進行パルス波がでれば成功．
5. データの保存
   $ ./aout␣TP ↵
   でディレクトリ TP にデータを保存する．

Step3.1 脈動進行パルス波

Step2 の結果を初期値として，脈動進行パルス波を求める．

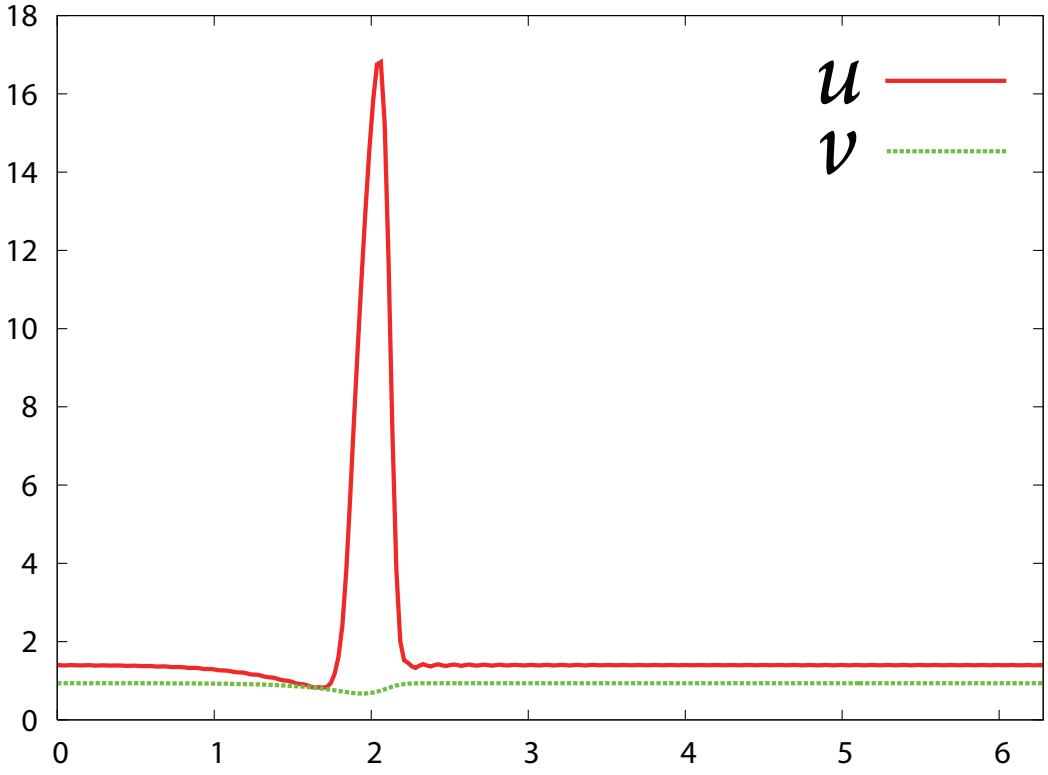


図 6.1.3: 進行パルス波

1. 実行前の設定

   (a) `0parameter.dat`

```
                .....
        3.38    ;   dv   ; par[2]
                .....
```

   と変更する.

   (b) `parameter.dat`

```
                .....
          3     ;   Initial_Select          ; (*1)
                .....
     100000     ;   MaxStep                 ; 最大 step 数
                .....
```

   (*1) 初期値 $\boldsymbol{u}_0(x)$ は `Initial.dat` に入っているものを用いる．パラメータは `0parameter.dat` のものを用いる.

   と変更する.

   (c) `Initial.dat`

   Step2 で出力された `final.dat` を `Initial.dat` として使用する.

2. 実行

```
$ cp␣TP/final.dat␣Initial.dat ↵
```

$ ./aout␣1 ↵
で実行する.

デモプログラム **3.1**

Step3.1 の設定から実行までの操作は

$ cp␣../DEMO/Exothermic-reaction_Model/TE/03_TB1/*␣./ ↵

$ ./aout␣1 ↵

で行える.

RD/DEMO/Exothermic-reaction_Model/TE/03_TB1 には Step3.1 の設定で必要なファイル.

- 0parameter.dat
- Function.c
- parameter.dat
- Initial.dat
- gnuplot.c

が存在する.

3. 結果
   脈動進行パルス波が出れば成功.

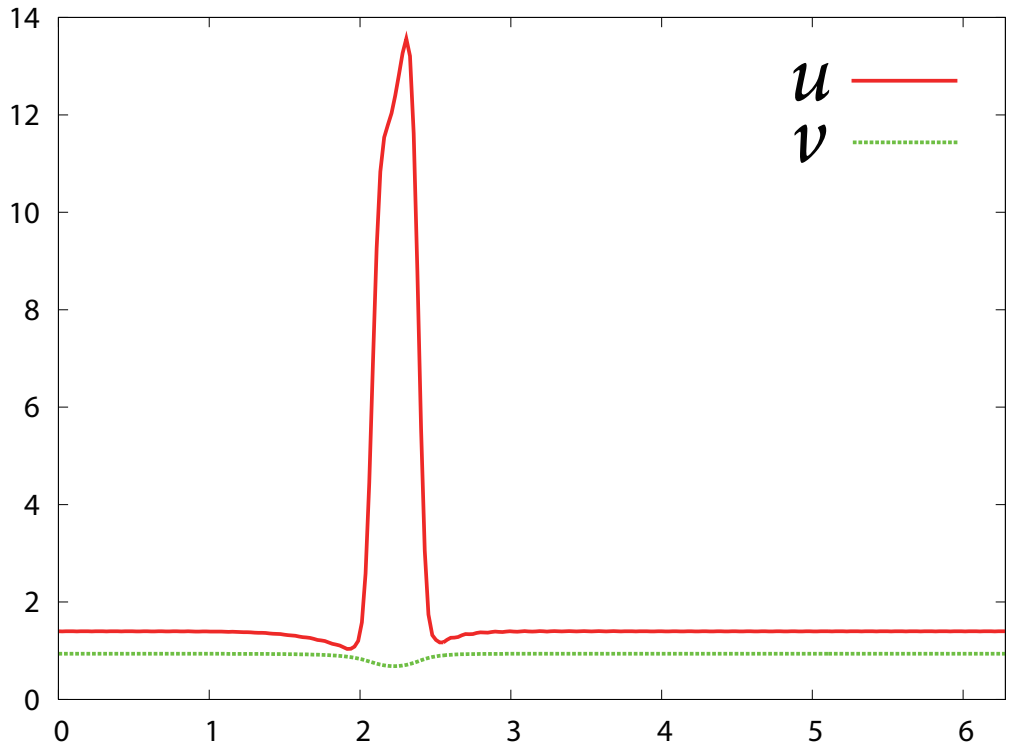


図 6.1.4: 脈動進行パルス波

Step3.2 脈動進行パルス波

安定した脈動解を 100000step ほど計算する.

1. 実行前の設定
   (a) 0parameter.dat
   変更点なし.
   (b) parameter.dat
   変更点なし.
   (c) Initial.dat
   Step3.1 で出力された final.dat を Initial.dat として使用する.
2. 実行
   $ cp␣final.dat␣Initial.dat ↵
   $ ./aout␣1 ↵
   で実行する.

   デモプログラム **3.2**

   Step3.2 の設定から実行までの操作は

   $ cp␣../DEMO/Exothermic-reaction_Model/TE/04_TB2/*␣./ ↵

   $ ./aout␣1 ↵

   で行える.

   RD/DEMO/Exothermic-reaction_Model/TE/04_TB2 には Step3.2 の設定で必要なファイル.

   - 0parameter.dat
   - Function.c
   - parameter.dat
   - Initial.dat
   - gnuplot.c

   が存在する.
3. データの保存
   $ ./aout␣TB ↵
   でディレクトリ TB にデータを保存する.

6.2 1次元発熱反応モデルの分岐計算

ここでは，7.1 節で求めたパルス波の分岐図を求める．ただし，d を分岐パラメータとして与える．カレントディレクトリは分岐図を求めるところに移動 (`$ cd_RD/AS` ↵) して，最初に発熱反応モデル (97 ページの (∗1)) の拡散項と反応項の設定を行う．これらの設定は時間発展方程式と同じであり，拡散項は `Function.c` の関数 `Diffusion` を

```
void  Diffusion( TEMP TP, double *par )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  par[TP.N_EP+1]  =  (4.0*M_PI*M_PI*par[1]) / (par[7]*par[7]);
  par[TP.N_EP+2]  =  (4.0*M_PI*M_PI*par[2]) / (par[7]*par[7]);
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述して，反応項は `Function.c` の関数 `Reaction` を

```
void  Reaction( TEMP TP, double *par, double *x, double *F )
{
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  int i;
  int N = TP.N;
  for( i = 1; i <= N; i++ ){
    F[i  ]  = ((-par[4]*x[i])+exp(x[i]/(1.0+(x[i]/par[5])))*x[i+N])/par[3];
    F[i+N]  = (par[6]*(1.0-x[i+N]))-exp(x[i]/(1.0+(x[i]+par[5]))*x[i+N]);
  }
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
}
```

と記述する．次に周期境界条件 (97 ページの (∗2)) の設定を行う．境界条件の設定は `parameter.dat` を開き，`BC_Select` を

```
                .............
          3     ;  BC_Select       ;  OP[9]
                .............
```

と与える．ただし，`BC_Select` = 3 は周期境界条件を表す．

Step1.1 定常解の分岐計算

7.1 節で求めた Step1 の解を用いて，定常解の分岐図を求める．ただし，Step1.1 では定常解の安定性を計算しないとする．定常解の選択およびその安定性計算による設定は `parameter.dat` を開き，次のように設定する．

```
          ...........
 1   ;  SolSelect      ;  OP[8]
          ...........
-1   ;  EigenValue     ;  OP[11]
-1   ;  EigenVector    ;  OP[12]
          ...........
```

ただし，SolSelect = 1 は定常解を選択し，EigenValue と EigenVector を 1 以外の値に選択すると解の安定性を計算しない．

1. 実行前の設定
   (a) Initial.dat
   7.1 節の Step1 で求めた解を Newton 法の初期値に与える．
   $ cp␣../TE/SP/final.dat␣Initial.dat ↵
   (b) parameter.dat

```
 0.000000   ;  ds             ;  OP[1]
 0.000101   ;  PseudoNewton   ;  OP[2]
       -1   ;  KellerEqn      ;  OP[3]
      200   ;  MaxIter        ;  OP[4]
 1.0E-03   ;  dt             ;  OP[5]
        1   ;  N_MSM          ;  OP[6]
       -1   ;  PeriodFix      ;  OP[7]
        2   ;  ConPar         ;  OP[7]
        1   ;  SolSelect      ;  OP[8]
        3   ;  BC_Select      ;  OP[9]
        6   ;  EE_Select      ;  OP[10]
       -1   ;  EigenValue     ;  OP[11]
       -1   ;  EigenVector    ;  OP[12]
 1.0E-08   ;  MachineEpsion  ;  OP[13]
        1   ;  StepCut        ;  OP[14]
        1   ;  MaxStep        ;  OP[15]
        0   ;  Gnuplot        ;  OP[16]
        1   ;  Initial        ;  OP[17]
```

2. 実行
   上記の設定が終わったら，
   $ ./aout␣1 ↵
   で実行する．

デモプログラム **1.1**

Step1.1 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/01_SPINI/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Exothermic-reaction_Model/AS/01_SPINI には Step1.1 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する.

3. 実行結果

```
  ...........

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   8 / 200
  Control Parameter    =     5.000000000000000
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000270406062100

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   9 / 200
  Control Parameter    =     5.000000000000000
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000012730538

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   10 / 200
  Control Parameter    =     5.000000000000000
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000264208
------------------------------------------------------------
                       Convergence  !!
------------------------------------------------------------
```

4. データの保存

Step1.1 で求めた定常解のデータを SP_INI というディレクトリに保存する.

次のコマンドを実行することでデータをディレクトリ SP_INI に保存できる.

```
$ ./aout␣SP_INI ↵
```

Step1.2 定常解の分岐計算

次に，Step1.1 で求めた解を用いてパラメータを変化させながら定常解の枝を求める．

1. 実行前の設定
   (a) `Initial.dat`
   初期値は Step1.1 で求めた解を用いる．
   (b) `parameter.dat`

```
0.005   ;  ds            ; パラメータの変化量 0.005
           ..........
  10    ;  StepCut       ; 10step ごとにデータを出力する
 400    ;  MaxStep       ; 最大 step 回数
           ..........
```

   と変更する．
2. 実行

```
$ cp␣SP_INI/middle_final.dat␣Initial.dat ↵
$ ./aout␣1 ↵
```

デモプログラム **1.2**

Step1.2 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/02_SP01/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

`RD/DEMO/Exothermic-reaction_Model/AS/02_SP01` には Step1.2 の設定で必要なファイル

- `Function.c`
- `Initial.dat`
- `parameter.dat`

が存在する．

3. 実行結果

```
  ...........
  Step  /  MaxStep     =   400 / 400
  iter  /  MaxIter     =   3 / 200
  Control Parameter    =     6.999999999999957
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000123511199

  Step  /  MaxStep     =   400 / 400
  iter  /  MaxIter     =   4 / 200
  Control Parameter    =     6.999999999999957
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000015726073

  Step  /  MaxStep     =   400 / 400
  iter  /  MaxIter     =   5 / 200
  Control Parameter    =     6.999999999999957
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000001747026
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
  ...........
```

4. データの保存

   step1.2 で求めた分岐図のデータを SP_01 のディレクトリに保存する.

   ```
   $ ./aout␣SP_01 ↵
   ```

Step1.3 定常解の分岐計算

Step1.2 で求めた枝の続きを求める．その際，ニュートン法では解が収束しなくなる．
実際，初期値に step1.2 で求めた final.dat を用いて実行すると，

```
  ...........
  Step  /  MaxStep     =   8 / 400
  iter  /  MaxIter     =   115 / 200
  Control Parameter    =     7.039999999999957
  MachineEps           =     0.000000010000000
  CC_Norm              =  1252903.503148945979774

  Step  /  MaxStep     =   8 / 400
  iter  /  MaxIter     =   116 / 200
  Control Parameter    =     7.039999999999957
  MachineEps           =     0.000000010000000
  CC_Norm              =  252355404632210472960.000000000000000
-----------------------------------------------------------------
                      Not Convergence  !!
-----------------------------------------------------------------


-------------------------------------------------------------------
CC_Norm is not Converge !! (2.523554046322105E+20)
CC_Norm is not Converge !! (2.523554046322105E+20)
CC_Norm is not Converge !! (2.523554046322105E+20)
CC_Norm is not Converge !! (2.523554046322105E+20)
CC_Norm is not Converge !! (2.523554046322105E+20)
-------------------------------------------------------------------

  ...........
```

となってしまう．これは saddle-node 分岐により解が存在しないところで計算してしまっているためである．なので、ニュートン法から疑似弧長法に切り替えて計算を行う．

1. 実行前の設定
   (a) Initial.dat
   初期値は step1.2 で求めた final.dat を用いる．
   (b) parameter.dat

```
                ...........
        1   ;  KellerEqn       ;疑似弧長法
                ...........
     4000   ; MaxStep          ;最大ステップ数
                ...........
```

   と変更する．
2. 実行

   $ cp␣SP_01/middle_final.dat␣Initial.dat ↵

   $ ./aout␣1 ↵

デモプログラム **1.3**

Step1.3 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/03_SP02/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Exothermic-reaction_Model/AS/03_SP02 には Step1.3 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する.

3. 実行結果

```
.............

  Step  /  MaxStep     =   4000 / 4000
  iter  /  MaxIter     =   1 / 200
  Control Parameter    =     2.230401984086159
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000096142430039

  Step  /  MaxStep     =   4000 / 4000
  iter  /  MaxIter     =   2 / 200
  Control Parameter    =     2.230402182048909
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000023149141

  Step  /  MaxStep     =   4000 / 4000
  iter  /  MaxIter     =   3 / 200
  Control Parameter    =     2.230402182084287
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000013074
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------

  .............
```

4. データの保存

step1.3 で求めた分岐図のデータを SP_02 のディレクトリに保存する.

$ ./aout␣SP_02 ↵

Step1.4 定常解の分岐計算

parameter.dat の値を調節しながらニュートン法 (KellerEqn=-1) で計算を行い，ニュートン法で解が収束しない場合は疑似弧長法 (KellerEqn=1) で計算する，という作業を繰り返す．そのため詳しい解説は省略し，デモと操作方法のみ記しておく．

step1.4 のデモと操作方法

1. デモの読み込み
   $ cp␣../DEMO/Exothermic-reaction_Model/AS/(*1)/* ␣./ ↵
2. 実行
   $ ./aout␣1 ↵
3. 保存
   $ ./aout␣(*2) ↵

   上記の 1\UTF{FF5E}3 の操作を 2 回繰り返す．(*1)，(*2) は，
   1 回目　(*1):04_SP03　(*2):SP_03
   2 回目　(*1):05_SP04　(*2):SP_04
   と変更して実行する．

Step1.2 から 1.4 で求めた分岐図を見るには，gnuplot を開き，

$plot "SP_01/81_CP_TV_NR.dat" u 1:4, "SP_02/81_CP_TV_NR.dat"u 1:4,
 "SP_03/81_CP_TV_NR.dat" u 1:4, "SP_04/81_CP_TV_NR.dat" u 1:4 ↵
と実行する．分岐図は図 6.2.1 のようになり，saddle-node 分岐点が 2 つ存在する．

Step1.5 定常解の分岐計算

次に，図 6.2.1 の点 S から左側の分岐計算をする．

1. 実行前の設定
   (a) Initial.dat
   初期値として SP_INI の middle_final.dat を用いる．
   $ cp␣SP_INI/middle_final.dat␣Initial.dat ↵
   (b) parameter.dat

```
-0.005   ;  ds              ; パラメータの変化量 –0.005
          .....
  840    ; MaxStep         ;最大ステップ数
          .....
```

   と変更する．
2. 実行
   $ ./aout␣1 ↵
   で実行する．

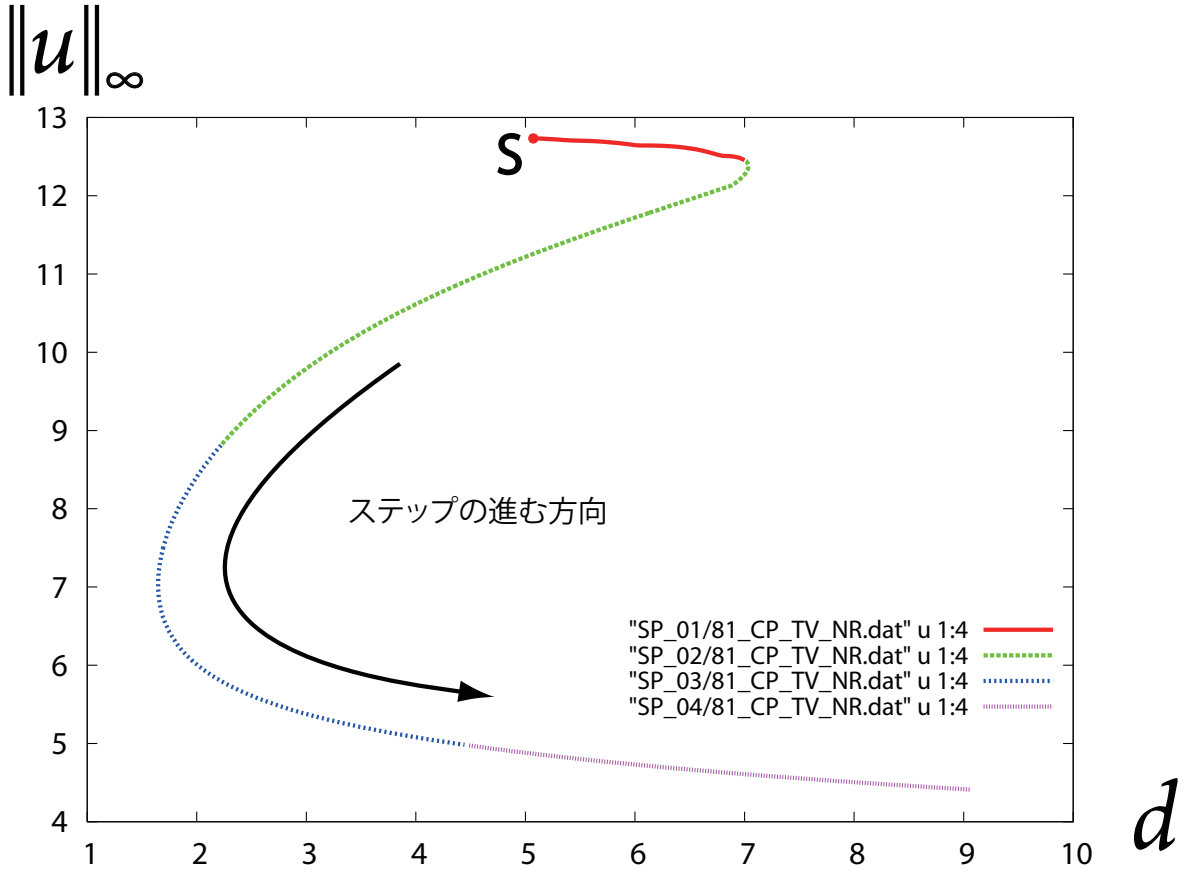


図 6.2.1: 定常パルス波の分岐図

デモプログラム **1.5**

Step1.5 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/06_SP05/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Exothermic-reaction_Model/AS/06_SP05 には Step1.5 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat

が存在する.

3. 実行結果

```
  ..........

  Step  /  MaxStep     =   840 / 840
  iter  /  MaxIter     =   2 / 200
  Control Parameter    =     0.800000000000085
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000974439979814

  Step  /  MaxStep     =   840 / 840
  iter  /  MaxIter     =   3 / 200
  Control Parameter    =     0.800000000000085
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000018493665

  Step  /  MaxStep     =   840 / 840
  iter  /  MaxIter     =   4 / 200
  Control Parameter    =     0.800000000000085
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000937121
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------

  ..........
```

4. データの保存

   step1.5 で求めた分岐図のデータを SP_05 というディレクトリに保存する.

   $ ./aout␣SP_05 ↵

Step1.6 定常解の分岐計算

step1.5 で求めた枝の続きを求める.

1. 実行前の設定

   (a) Initial.dat

   初期値として SP_05 の final.dat を用いる.

   ($ cp␣final.dat␣Initial.dat ↵)

   (b) parameter.dat

```
     -0.005   ;  ds              ; パラメータの変化量 −0.005
               ..........
           1  ;  KellerEqu     ; 疑似弧長法
               ..........
        3000  ; MaxStep         ; 最大ステップ数
               ..........
```

と変更する.

2. 実行

   $ ./aout␣1 ↵

   で実行する.

デモプログラム **1.6**

Step1.6 の設定から実行までの操作は

$ cp␣../DEMO/Exothermic-reaction_Model/AS/07_SP06/*␣./ ↵

$ ./aout␣1 ↵

で行える.

RD/DEMO/Exothermic-reaction_Model/AS/07_SP06 には Step1.6 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat

が存在する.

3. 実行結果

```
  ..........
  Step  /  MaxStep     =   3000 / 3000
  iter  /  MaxIter     =   0 / 6000
  Control Parameter    =     9.999085259217248
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.005000000000000

  Step  /  MaxStep     =   3000 / 3000
  iter  /  MaxIter     =   1 / 6000
  Control Parameter    =    10.004076269062283
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000001169675227

  Step  /  MaxStep     =   3000 / 3000
  iter  /  MaxIter     =   2 / 6000
  Control Parameter    =    10.004076275823657
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000584759
------------------------------------------------------------
                       Convergence  !!
------------------------------------------------------------
  ..........
```

4. データの保存

step1.6 で求めた分岐図のデータを SP_06 というディレクトリに保存する．

```
$ ./aout␣SP_06 ↵
```

Step1.5 と 1.6 で求めた分岐図を見るには，gnuplot を開き，

```
$plot "SP_05/81_CP_TV_NR.dat" u 1:4, "SP_06/81_CP_TV_NR.dat"u 1:4 ↵
```

と実行する．分岐図は図 6.2.2 のようになる．

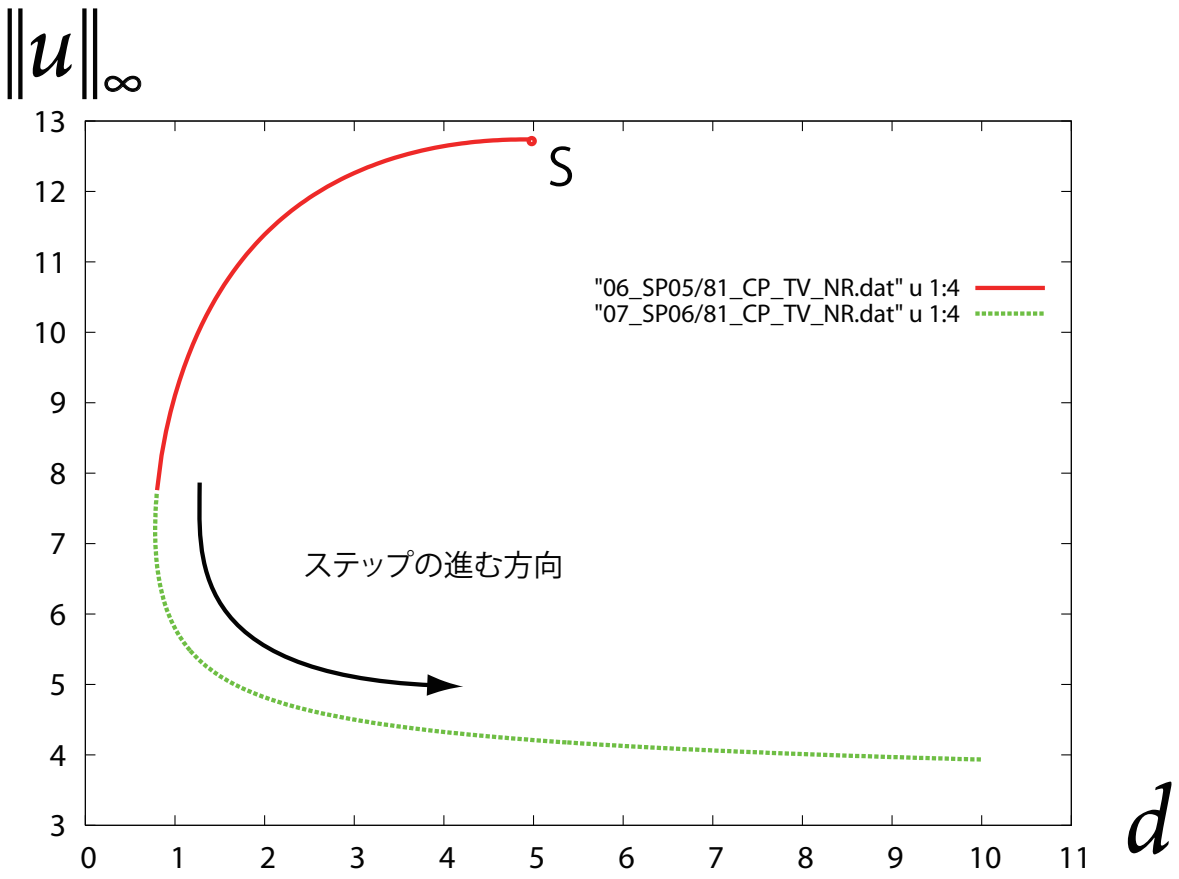


図 6.2.2: 定常パルス波の分岐図

Step1.7 定常解の分岐計算

step1.2 から step1.6 までのデータを結合する．その際，図 6.2.1 と図 6.2.2 をみるとわかるように分岐計算をすすめた向きがそれぞれ異なる．なので，ここでは step1.5 と step1.6 で求めたデータの向きを逆にして結合する．

1. step1.5 と step1.6 のデータを結合する．
   次のコマンドで，step1.5 と step1.6 の middle_final.dat のデータを結合し，middle_final.dat に保存する．

   ```
   $cat␣SP_05/middle_final.dat␣SP_06/middle_final.dat > middle_final.dat ↵
   ```

2. 結合したデータのステップ数を逆にする．

$./Sort ↵
(詳しい説明はデモ 1 の分岐計算 step1.5 を参照)

3. step1.2 から step1.4 までのデータとステップ数を逆にしたデータを結合し，middle_final.dat に保存する．

$cat␣middle_final.dat␣SP_01/middle_final.dat␣SP_02/middle_final.dat␣
SP_03/middle_final.dat␣SP_04/middle_final.dat > tmp.dat ↵

$mv␣tmp.dat␣middle_final.dat ↵

4. 結合したデータのステップ数を修正する
parameter.dat を次のように変更する．

```
                .............
          9    ;   Initial          ;   OP[17]
```

$ ./aout␣1 ↵
で実行する．
実行後，結合した middle_final.dat に対応する

- 81_CP_TV_NR.dat
- 82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat
- solution_valu.dat
- 1
- 11

が出力される．ただし，1 はステップ数が修正された middle_finel.dat，11 はステップ数が修正された 9middle_final.dat である．

5. データの出力
分岐図のデータを gnuplot で出力する (図 6.2.3)．
$ gnuplot ↵
$ p "81_CP_TV_NR.dat"␣u␣1:4␣w␣l ↵

6. データの保存
1 と 11 をそれぞれ middle_final.dat，　9middle_final.dat に上書きする．
$ cp␣1␣middle_final.dat ↵
$ cp␣11␣9middle_final.dat ↵

まとめたデータを SP_F に保存する．
$ ./aout␣SP_F ↵

Step1.8 定常解の分岐計算

ここでは Step1.7 で求めた定常解の安定性を求める方法について説明する．解の安定性は，

①解の枝を求めながら安定性を求める方法と

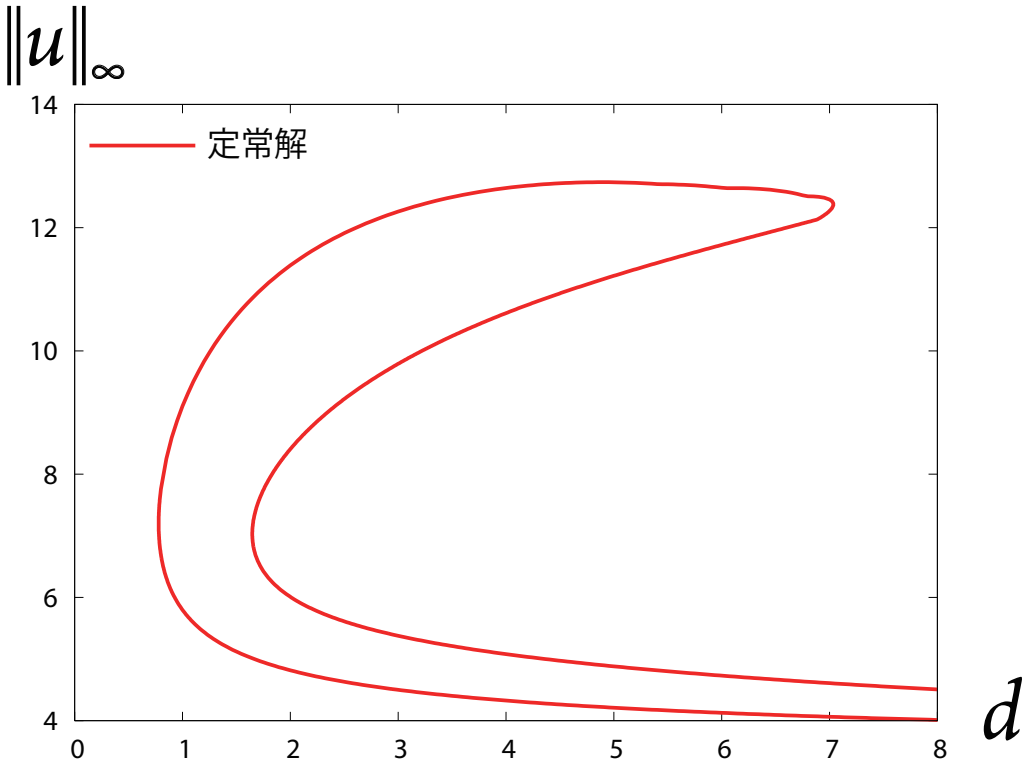


図 6.2.3: 定常解の分岐図

②解の枝を求め終わった後、それらの解の安定性を調べる方法

の 2 通りある．ここでは，定常解の枝を Step1.7 で既に求めたので 2 の方法を用いる

1. 実行前の設定
   (a) `Initial.dat`
   Step1.7 で保存したディレクトリ `SP_F` の中から安定性計算に必要とする `9middle_final.dat` を `Initial.dat` に変更して計算するディレクトリに持ってくる．
   `$ cp␣SP_F/9middle_final.dat␣Initial.dat`
   (b) `parameter.dat`

```
                    .............
          8    ;   Initial          ; (*1)
(*1) 各 step における解の固有値と固有関数の計算を行う
```

   と変更する．
2. 実行
   `$ ./aout␣1` ↵ で実行する．

デモプログラム **1.8**

Step1.8 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/08_SP07/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Exothermic-reaction_Model/AS/08_SP07 には Step1.8 の設定で必要なファイル．

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

安定性の結果は 82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat のデータを参照する．gnuplot を開き，

```
$ gnuplot > plot␣"82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat"␣u␣1:8
```

と実行することで横軸にステップ数，縦軸に固有値実部を与えたグラフが表示される (図 6.2.4).

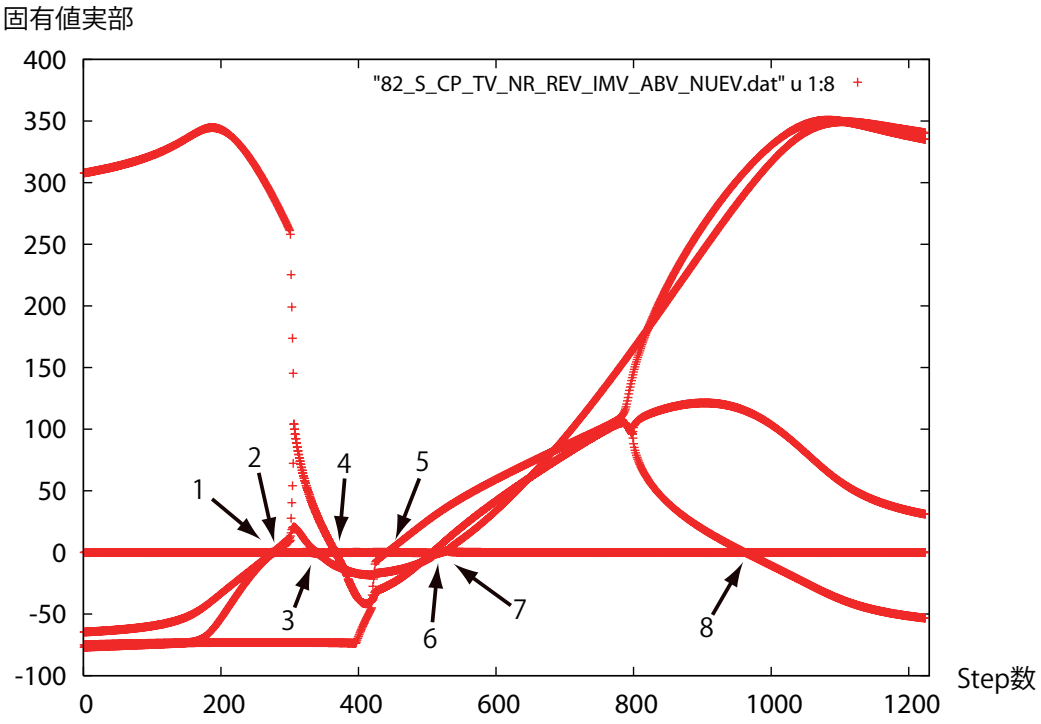


図 6.2.4:

step の変化と共に固有値の実部が 0 を横切るとき，分岐が起きる．今回の場合，8 つの分岐が存在する．

ここでひとつ問題があり，今回扱っているモデルでは，定常解の固有値を安定に求める

ことが困難である．なので，ここでは分岐の種類についての詳しい説明は省略する．
図 6.2.4 の 1〜8 では次の 3 つの種類の分岐が現れる．

- saddle-node 分岐 (1,5,8)
- Hopf 分岐 (4,6)
- pitch-fork 分岐 (2,3,7)

4. データの保存
   Step1.8 で求めた分岐図のデータを SP というディレクトリに上書きする．
   `$ ./aout␣SP_F` ↵
   このとき，ディレクトリを上書きしますかと尋ねられるので，`$ yes` ↵ と入力する．

Step2.1 進行解の数値計算

ここでは Step1.8 で求めた定常解の pitch-fork 分岐点における固有関数の情報を用いて進行解を求める．用いる固有関数の情報は，固有値の実部が正である step のものを用いる．Step1.8 でも示したように，pitch-fork 分岐点での固有値の情報は以下のようになっている．

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値．

```
                    ...........
 274  ..................  2.90284773E+02  0.00000000E+00  ...........
 274  ..................  8.78498815E-02  0.00000000E+00  ...........
 274  .................. -4.49039817E-01  0.00000000E+00  ...........
 274  .................. -1.46363103E+00  0.00000000E+00  ...........
 274  .................. -7.30440658E+01  0.00000000E+00  ...........
 275  ..................  2.89238921E+02  0.00000000E+00  ...........
 275  ..................  1.34657221E-01  0.00000000E+00  ...........
 275  ..................  9.15488035E-03  0.00000000E+00   ...........
 275  .................. -1.03611995E+00  0.00000000E+00  ...........
 275  .................. -7.30439543E+01  0.00000000E+00  ...........
                    ...........
```

1. 実行前の設定
   (a) Initial.dat
   pitch-folk 分岐点の固有関数を Initial.dat として使用する．固有ベクトルのデータは 6_EigenVector_Data に入っている．
   6_EigenVector_Data/ 275_CP ...[1]/o1 ...[2]のデータを Initial.dat に変更して持ってくる．
   (b) parameter.dat

[1] ここにはコントロールパラメータの値が入っている
[2] ここには固有値実部の値が入っている

```
-0.005000   ;  ds                ; パラメータの変化量なし
      -1    ;  KellerEqn        ; 擬似弧長法の設定なし
                ...........
       2    ;  SolSelect        ; 進行解を選択
                ...........
     155    ;  MaxStep           ; 最大 step 回数
                ...........
       2    ;  Initial          ; (*1)
(*1) 分岐点から伸びる解の枝を求める
```

と変更する.

(c) `0_BranchChangeParameter.dat`

分岐点における零固有値に対する固有関数の情報と `0_BranchChangeParameter.dat` のパラメータを用いて Newton 法に用いる初期値 $(\boldsymbol{u}_{ini}, c_{ini}, p_{ini})$ を構成する.

- 初期値の構成 $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon\boldsymbol{\Phi}_R$,
- 速度の構成 $c_{ini} = \delta_c$,
- パラメータの構成 $p_{ini} = \delta_p$,

ただし, $\boldsymbol{u}_0$ は定常解の pitch-fork 分岐点における解 $\boldsymbol{u}$, $\boldsymbol{\Phi}_R$ は分岐点における零固有値に対する固有関数の実部, $\varepsilon, \delta_c, \delta_p$ はそれぞれ `0_BranchChangeParameter.dat` における `eps` , `velocity_delta` , `par_delta` である.
ここでは,

```
      0.05  ;  eps             ; u_ini の構成に必要
      0.00  ;  period_delta    ; ここでは不必要な設定
       0.0  ;  velocity_delta  ; 速度の微小変量
0.00000000  ;  par_delta       ; パラメータの微小変量
     0.000  ;  time            ; ここでは不必要な設定
```

と変更する.

2. 実行

`$ ./aout␣1` ↵ で実行する

デモプログラム **2.1**

Step2.1 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/09_TP1_01/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Exothermic-reaction_Model/AS/09_TP1_01 には Step2.1 の設定で必要なファイル．

- Function.c
- Initial.dat
- parameter.dat
- 0_BranchChangeParameter

が存在する．

3. 実行結果

```
  ...........

  Step  /  MaxStep     =   155 / 155
  iter  /  MaxIter     =   2 / 200
  Control Parameter    =     0.003779154067403
  Velocity             =     5.773071012535138
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000037012342

  Step  /  MaxStep     =   155 / 155
  iter  /  MaxIter     =   3 / 200
  Control Parameter    =     0.003779154067403
  Velocity             =     5.773071012718584
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000004661
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
```

収束した解は速度を持っているため，進行解であるとわかる．

進行解の分岐図を表示すると図 6.2.5 のようになる．

図 6.2.5 を定常解とあわせて表示すると次のようになる．

4. データの保存

Step2.2 で求めた分岐図のデータを TP1_F というディレクトリに保存する．

```
$ ./aout␣TP1_F ↵
```

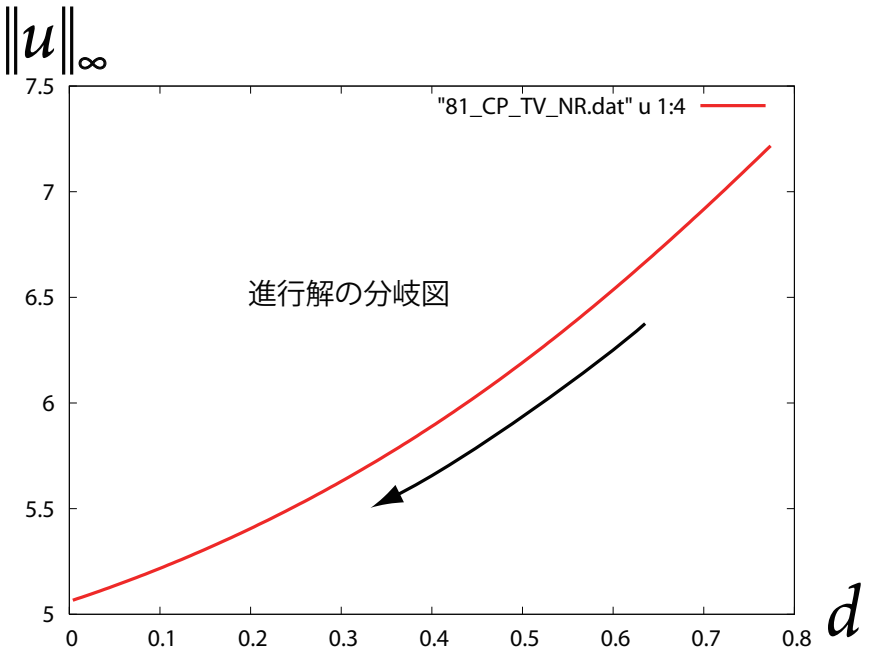


図 6.2.5: 進行解の分岐図

Step3.1 進行解の分岐計算

7.1 節で求めた Step2 の解を用いて，進行解の分岐図を求める．

1. 実行前の設定

   (a) `Initial.dat`

   7.1 節の Step で求めた解を Newton 法の初期値に与える．

   `$ cp␣../TE/TP/final.dat␣Initial.dat` ↵

   (b) `parameter.dat`

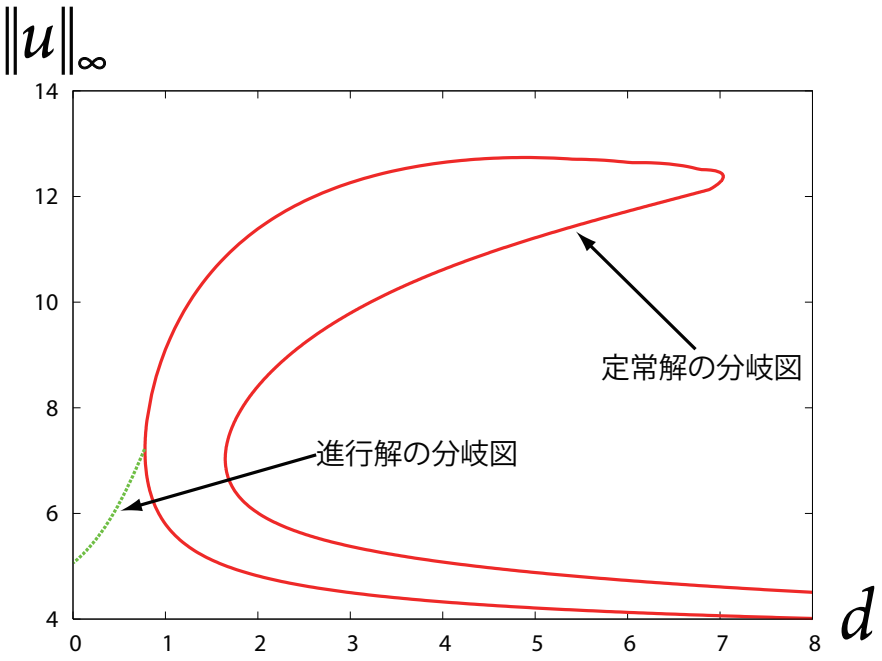


図 6.2.6: 進行解の分岐図

```
0.000000   ;  ds              ;  OP[1]
0.000101   ;  PseudoNewton    ;  OP[2]
      -1   ;  KellerEqn       ;  OP[3]
      30   ;  MaxIter         ;  OP[4]
 1.0E-03   ;  dt              ;  OP[5]
       1   ;  N_MSM           ;  OP[6]
      -1   ;  PeriodFix       ;  OP[7]
       2   ;  ConPar          ;  OP[7]
       2   ;  SolSelect       ;  OP[8]
       3   ;  BC_Select       ;  OP[9]
       6   ;  EE_Select       ;  OP[10]
      -1   ;  EigenValue      ;  OP[11]
      -1   ;  EigenVector     ;  OP[12]
 1.0E-08   ;  MachineEpsion   ;  OP[13]
       1   ;  StepCut         ;  OP[14]
       1   ;  MaxStep         ;  OP[15]
       0   ;  Gnuplot         ;  OP[16]
       1   ;  Initial         ;  OP[17]
```

2. 実行

   上記の設定が終わったら，

   $ ./aout␣1 ↵

   で実行する．

デモプログラム **3.1**

Step3.1 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/10_TP2_INI/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Exothermic-reaction_Model/AS/10_TP2_INI には Step3.1 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

```
  ...........

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   3 / 30
  Control Parameter    =     2.000000000000000
  Velocity             =    12.928757323597416
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.032783248608212


  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   4 / 30
  Control Parameter    =     2.000000000000000
  Velocity             =    12.928384371030262
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000002199030632


  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   5 / 30
  Control Parameter    =     2.000000000000000
  Velocity             =    12.928384353630330
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000231466
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------

  ...........
```

4. データの保存

   Step3.1 で求めた定常解のデータを TP2_INI というディレクトリに保存する.

   次のコマンドを実行することでデータをディレクトリ TP2_INI に保存できる.

   $ ./aout␣TP2_INI ↵

Step3.2 進行解の分岐計算

次に，Step3.1 で求めた解を用いてパラメータを変化させながら進行解の枝を求める.

1. 実行前の設定

   (a) Initial.dat

   初期値は Step3.1 で求めた解を用いる.

   (b) parameter.dat

```
0.002   ;  ds              ; パラメータの変化量 0.005
           ...........
    1   ;  StepCut        ; 10step ごとにデータを出力する
  330   ;  MaxStep        ; 最大 step 回数
           ...........
```

と変更する.

2. 実行

```
$ cp␣TP2_INI/final.dat␣Initial.dat ↵
$ ./aout␣1 ↵
```

デモプログラム **3.2**

Step3.2 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/11_TP2_01/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Exothermic-reaction_Model/AS/11_TP2_01 には Step3.2 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する.

3. 実行結果

```
  ..........

  Step   /   MaxStep      =    330 / 330
  iter   /   MaxIter      =    6 / 30
  Control Parameter       =      2.659999999999927
  Velocity                =     -0.090468446226469
  MachineEps              =      0.000000010000000
  CC_Norm                 =      0.000003118381549

  Step   /   MaxStep      =    330 / 330
  iter   /   MaxIter      =    7 / 30
  Control Parameter       =      2.659999999999927
  Velocity                =     -0.090468666535193
  MachineEps              =      0.000000010000000
  CC_Norm                 =      0.000000001455116
------------------------------------------------------------------
                          Convergence   !!
------------------------------------------------------------------

  ..........
```

進行解の分岐図を表示すると図 6.2.7 のようになる.

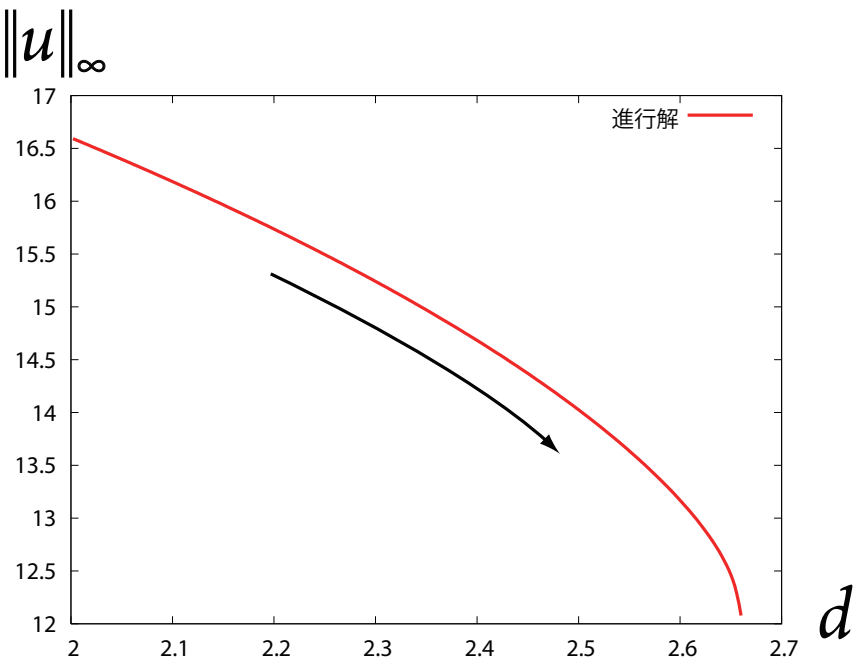


図 6.2.7: 進行解の分岐図

図 6.2.7 を定常解とあわせて表示すると図 6.2.8 のようになる.

4. データの保存

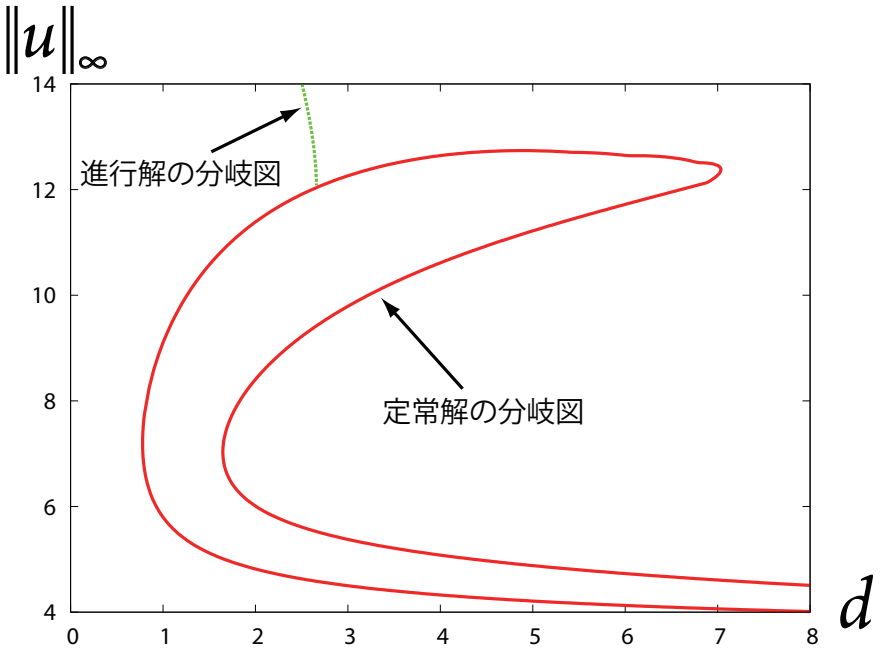


図 6.2.8: 進行解の分岐図

step3.2 で求めた分岐図のデータを TP2_F のディレクトリに保存する.

```
$ ./aout␣TP2_F ↵
```

Step3.3 進行解の分岐計算

ここでは Step3.2 で求めた進行解の安定性を求める方法について説明する.

1. 実行前の設定
   (a) Initial.dat
   まず, Step3.2 で保存したディレクトリ TP2_01 の中から固有値計算に必要なデータ 9middle_final.dat を Initial.dat に変更して計算するディレクトリに持ってくる.

   ```
   $ cp␣TP2_01/9middle_final.dat␣Initial.dat
   ```

   (b) parameter.dat

   ```
                  .............
           8    ;   Initial          ; 安定性の計算を選択
   ```

   と変更する.
2. 実行

   ```
   $ ./aout␣1 ↵
   ```

   で実行する.

デモプログラム **3.3**

Step3.3 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/12_TP2_02/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Exothermic-reaction_Model/AS/12_TP2_02 には Step3.3 の設定で必要なファイル

- Function.c
- Initial.dat
- parameter.dat

が存在する．

3. 実行結果

82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat のデータから横軸にステップ数，縦軸に固有値実部をグラフで示すと図 6.2.9 となる．

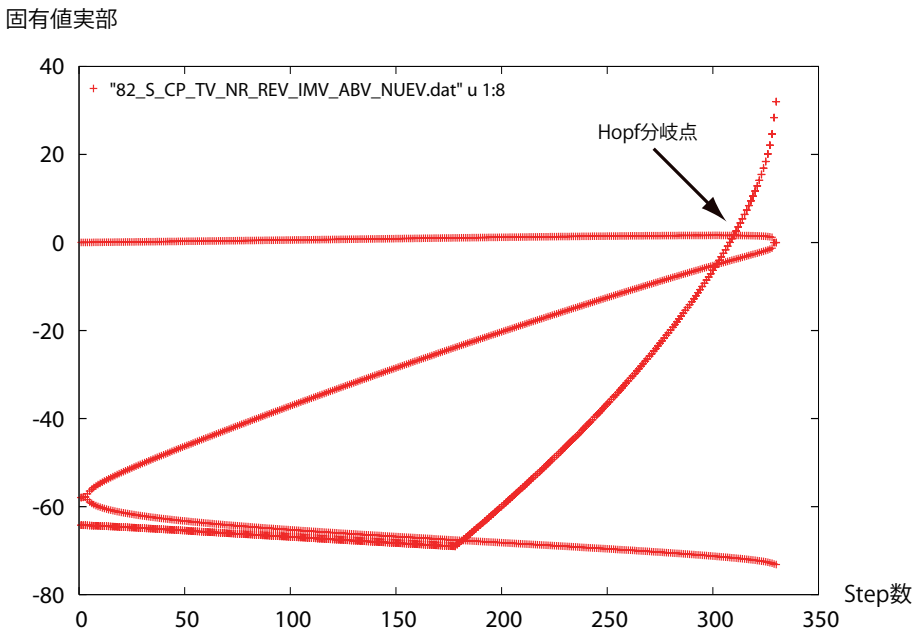


図 6.2.9:

図 6.2.9 をみると，step300 付近で固有値実部が 0 を横切っているのがみてとれる．そこで，82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat を開いてみてみると，308step で固有値の実部が 2 つ 0 を横切っている．つまりここで Hopf 分岐が起こっていることがわかる．Hopf 分岐点からは脈動進行解の枝が存在する．

左から 1 列目の step 数，8 列目の固有値実部，9 列目の固有値虚部の値.

```
                      …………
307 ……………………   1.61302794E+00  0.00000000E+00   …………
307 ……………………  -8.27802402E-01 -1.44360682E+02  …………
307 ……………………  -8.27802402E-01  1.44360682E+02   …………
307 ……………………  -4.29835612E+00  0.00000000E+00   …………
307 ……………………  -7.15261082E+01  0.00000000E+00   …………
308 ……………………   1.60629017E+00  0.00000000E+00   …………
308 ……………………   1.07230184E-02 -1.43964738E+02  …………
308 ……………………   1.07230184E-02  1.43964738E+02   …………
308 ……………………  -4.15896757E+00  0.00000000E+00   …………
308 ……………………  -7.15701018E+01  0.00000000E+00   …………
                      …………
```

4. データの保存

   Step3.3 で求めた分岐図のデータを TP2_F というディレクトリに上書きする.

   `$ ./aout␣TP2_F` ↵

Step4 進行解の数値計算

Step4 では Step1.8 で求めた定常解の pitch-fork 分岐点から分岐する進行解の分岐図を求める. その際，分岐点における固有関数の情報を用いるという今までのやり方ではここで求めたい枝を求めるのは困難である. そこで，このでは違う方法を用いて進行解の枝を求めていく.

Step4.1 進行解の数値計算

ここではまず，コントロールパラメーターを ϵ に変更して，$\epsilon = 0.001025$ のときの定常解の分岐図を求める.

1. 実行前の設定

   (a) `Initial.dat`

   初期値は SP_INI にある `middle_final.dat` のデータを用いる.

   `$cp␣SP_INI/middle_final.dat␣Initial.dat` ↵

   (b) `parameter.dat`

```
-0.000005   ;  ds              ;  OP[1]
 0.000101   ;  PseudoNewton    ;  OP[2]
       -1   ;  KellerEqn       ;  OP[3]
       30   ;  MaxIter         ;  OP[4]
  1.0E-04   ;  dt              ;  OP[5]
        1   ;  N_MSM           ;  OP[6]
       -1   ;  PeriodFix       ;  OP[7]
        3   ;  ConPar          ;  OP[7]
        1   ;  SolSelect       ;  OP[8]
        3   ;  BC_Select       ;  OP[9]
        6   ;  EE_Select       ;  OP[10]
       -1   ;  EigenValue      ;  OP[11]
       -1   ;  EigenVector     ;  OP[12]
  1.0E-08   ;  MachineEpsion   ;  OP[13]
        1   ;  StepCut         ;  OP[14]
       35   ;  MaxStep         ;  OP[15]
        0   ;  Gnuplot         ;  OP[16]
        1   ;  Initial         ;  OP[17]
```

と変更する.

2. 実行

   $ ./aout␣1 ↵

   で実行する.

デモプログラム **4.1**

Step4.1 の設定から実行までの操作は

$ cp␣../DEMO/Exothermic-reaction_Model/AS/13_TP3_01/*␣./ ↵

$ ./aout␣1 ↵

で行える.

RD/DEMO/Exothermic-reaction_Model/AS/13_TP3_01 には Step4.1 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat

が存在する.

Step4.2 進行解の数値計算

Step4.1 で求めた解を用いて定常解の分岐図を求める.

1. 実行前の設定

(a) Initial.dat
先ほど求めた解を初期値として用いる．
$cp␣final.dat␣Initial.dat ↵

(b) parameter.dat

```
0.0050000   ;  ds
             ...........
          2  ; ConPar
             ...........
          1  ; EigenValue    ;固有値を計算する
          1  ; EigenCector   ;固有ベクトルを計算する
             ...........
          1  ;  StepCut
        200  ;  MaxStep
             ...........
          1  ;    Initial
```

と変更する．Step4.2 では固有値の計算をしながら定常解の枝を追っていく．

2. 実行

$ ./aout␣1 ↵

デモプログラム **4.2**

Step4.2 の設定から実行までの操作は

$ cp␣../DEMO/Exothermic-reaction_Model/AS/14_TP3_02/*␣./ ↵

$ ./aout␣1 ↵

で行える．

RD/DEMO/Exothermic-reaction_Model/AS/14_TP3_02 には Step4.2 の設定で必要なファイル．

- Function.c
- Initial.dat
- parameter.dat

が存在する．

82_S_CP_TV_NR_REV_IMV_ABV_NUEV.dat を開いてみてみると，step166 で固有値実部の値が 0 を横切っている．詳しい説明は省略するが，ここでは pitch-fork 分岐により，進行解が分岐する．

```
                    ...........
165  ..................    8.54593778E-02  0.00000000E+00    ...........
165  ..................   -2.39380306E+00  0.00000000E+00    ...........
165  ..................   -3.76071247E+01  1.66351451E+02    ...........
165  ..................   -3.76071247E+01 -1.66351451E+02    ...........
165  ..................   -4.31666931E+01  2.24716640E+01    ...........
166  ..................   -3.89203220E-02  0.00000000E+00    ...........
166  ..................   -2.25037219E+00  0.00000000E+00    ...........
166  ..................   -3.74751869E+01  1.66530530E+02    ...........
166  ..................   -3.74751869E+01 -1.66530530E+02    ...........
166  ..................   -4.29838236E+01  2.18762624E+01    ...........
                    ...........
```

Step4.3 進行解の数値計算

ここでは，Step4.2 で求めた pitch-fork 分岐点の固有関数の情報を用いて，進行解を求める．

1. 実行前の設定

   (a) Initial.dat

   step165 では正の値だった 1 行目の固有値実部が，step166 では負の値になっている．したがって，初期値として step165 の固有関数の情報を用いる．

   $cp␣6_EigenVector_Data/165_CP.../o1...␣Initial.dat ↵

   (b) parameter.dat

```
0.0050000   ;  ds              ; パラメータの変化量なし
       -1   ;  KellerEqn       ; 擬似弧長法の設定なし
            ...........
        2   ;  SolSelect       ; 進行解を選択
            ...........
       -1   ; EigenValue
       -1   ; EigenCector
            ...........
      200   ;  MaxStep
            ...........
        2   ;  Initial
```

   と変更する．

2. 実行

   $ ./aout␣1 ↵

   で実行する

デモプログラム **4.3**

Step4.3 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/15_TP3_03/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Exothermic-reaction_Model/AS/15_TP3_03 には Step4.3 の設定で必要なファイル．

- Function.c
- Initial.dat
- parameter.dat

が存在する．

Step4.4 進行解の数値計算

ここでは Step4.1 と同様の操作で，$\epsilon = 0.001025$ のときの進行解の分岐図を求める．

1. 実行前の設定
   (a) Initial.dat
   初期値は Step4.3 で求めた final.dat のデータを用いる．
   $cp␣final.dat␣Initial.dat ↵
   (b) parameter.dat

```
0.000005  ;  ds
             ...........
       3   ;  ConPar
             ...........
      35   ;  MaxStep
             ...........
       1   ;  Initial
```

   と変更する．
2. 実行
   $ ./aout␣1 ↵
   で実行する．

デモプログラム **4.4**

Step4.4 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/16_TP3_04/*␣./ ↵
$ ./aout␣1 ↵
```

で行える．

RD/DEMO/Exothermic-reaction_Model/AS/16_TP3_04 には Step4.4 の設定で必要なファイル．

- Function.c
- Initial.dat
- parameter.dat

が存在する．

Step4.5 進行解の数値計算

Step4.1〜Step4.4 までの操作により，ϵ=0.0012 のときの進行解の枝を見つけることができた．ここでは Step4.4 で求めた解を用いて進行解の分岐図を求める．

1. 実行前の設定
   (a) Initial.dat
   初期値は Step4.4 で求めた final.dat のデータを用いる．
   $cp␣final.dat␣Initial.dat ↵
   (b) parameter.dat

```
-0.005    ;  ds
   -1     ;  KellerEqn
             ...........
    2     ;  ConPar
             ...........
  114     ;  MaxStep
             ...........
    1     ;  Initial
```

   と変更する．
2. 実行
   $ ./aout␣1 ↵ で実行する

デモプログラム **4.5**

Step4.5 の設定から実行までの操作は

```
$ cp␣../DEMO/Exothermic-reaction_Model/AS/17_TP3_05/*␣./ ↵
$ ./aout␣1 ↵
```

で行える.

RD/DEMO/Exothermic-reaction_Model/AS/17_TP3_05 には Step4.5 の設定で必要なファイル.

- Function.c
- Initial.dat
- parameter.dat

が存在する.

求めた進行解の分岐図を定常解のものとあわせて表示すると次のようになる (※はデモでは求めていない部分の進行解の枝).

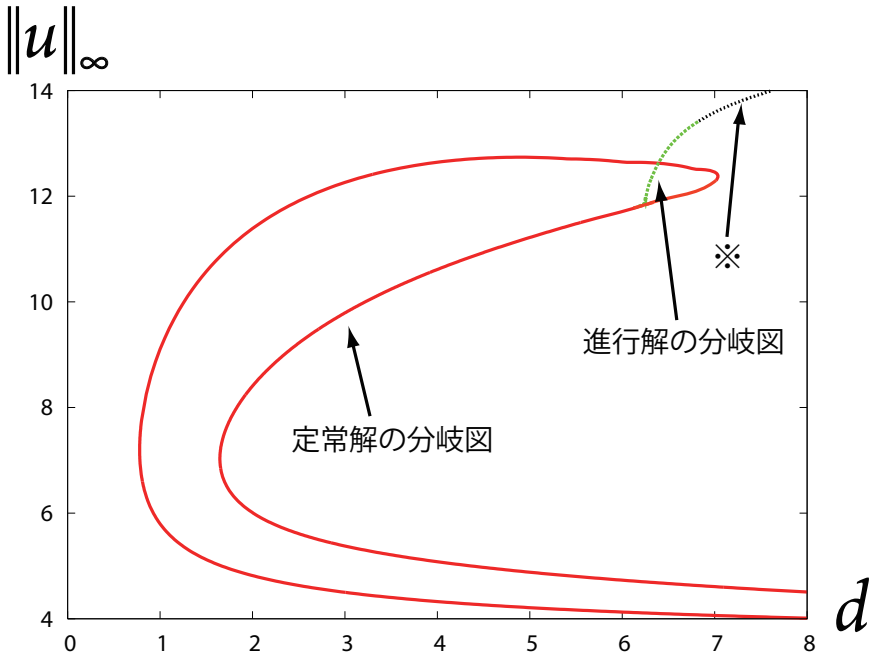


図 6.2.10: 定常解と進行解の分岐図

3. データの保存
   Step4.5 で求めた分岐図のデータを TP3_F というディレクトリに保存する.
   `$ ./aout␣TP3_F ↵`

Step5.1 脈動進行解の分岐計算

ここでは Step3.3 で求めた進行解の Hopf 分岐点の固有関数の情報を用いて脈動進行解を求める.

1. TP2_F の固有関数の情報

TP2_F で求めた固有関数の情報は保存されていないので，求めなおす必要がある．
次のコマンドで保存した TP2_F のデータをもってくる．
$ cp␣TP2_F/*␣./ ↵

実行する．
$ ./aout␣1 ↵

2. 実行前の設定

(a) Initial.dat
Hopf 分岐点の固有関数を Initial.dat として使用する．固有ベクトルのデータは 6_EigenVector_Data に入っている．6_EigenVector_Data/ 308_CP ...[3]/o1 ...[4]のデータを Initial.dat に変更して持ってくる．
$cp␣6_EigenVector_Data/308_CP.../o1...␣Initial.dat ↵

(b) parameter.dat

```
0.000000   ;  ds              ; パラメータの変化量なし
               ............
       4   ;  SolSelect       ; 脈動進行解を選択
               ............
       1   ;  StepCut         ; 毎 step ごとにデータを出力する
       1   ;  MaxStep         ; 最大 step 回数
               ............
       2   ;  Initial         ; (*1)
(*1) 分岐点から伸びる解の枝を求める．
```

と変更する．

(c) 0_BranchChangeParameter.dat
分岐点における零固有値に対する固有関数の情報と 0_BranchChangeParameter.dat のパラメータを用いて Newton 法に用いる初期値 $(\boldsymbol{u}_{ini}, T_{ini}, c_{ini}, p_{ini})$ を構成する．

- 初期値の構成 $\boldsymbol{u}_{ini} = \boldsymbol{u}_0 + \varepsilon(\sin(2\pi t)\boldsymbol{\Phi}_R + \cos(2\pi t)\boldsymbol{\Phi}_I)$,
- 周期の構成 $T_{ini} = \dfrac{2\pi}{\beta} + \delta_T$
- 速度の構成 $c_{ini} = \delta_c$,
- パラメータの構成 $p_{ini} = \delta_p$,

ただし，$\boldsymbol{u}_0$ は定常解の pitch-fork 分岐点における解 $\boldsymbol{u}$，$\boldsymbol{\Phi}_R, \boldsymbol{\Phi}_I$ はそれぞれ分岐点における零固有値に対する固有関数の実部と虚部，c は解の速度，T は解の周期 p はコントロールパラメータ，c_0 は進行解の Hopf 分岐点における速度，$\varepsilon, \delta_T, \delta_c, \delta_p, t$ はそれぞれ 0_BranchChangeParameter.dat における eps , period_delta , velocity_delta , par_delta , time である．
ここでは，

[3]ここにはコントロールパラメータの値が入っている
[4]ここには固有値実部の値が入っている

```
      0.05  ;  eps             ;    SS_2  &  SS_3  &  SS_4
      0.00  ;  period_delta    ;          &  SS_3  &  SS_4
       0.0  ;  velocity_delta  ;    SS_2  &        &  SS_4
0.00000000  ;  par_delta       ;    SS_2  &  SS_3  &  SS_4
     0.000  ;  time            ;          &  SS_3  &  SS_4
```

と変更する.

3. 実行

   $ ./aout␣1 ↵
   で実行する

> デモプログラム **5.1**
>
> Step5.1 の設定から実行までの操作は
>
> $ cp␣../DEMO/Exothermic-reaction_Model/AS/18_TB_INI/*␣./ ↵
>
> $ ./aout␣1 ↵
>
> で行える.
>
> RD/DEMO/Exothermic-reaction_Model/AS/18_TB_INI には Step5.1 の設定で必要なファイル.
>
> - Function.c
> - Initial.dat
> - parameter.dat
> - 0_BranchChangeParameter
>
> が存在する.

4. 実行結果実行すると，Hopf 分岐点から振動進行解の枝を求めるための初期値が得られる.

```
  ...........
  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   8 / 30
  Control Parameter    =     2.641842624371586
  Period               =     0.043900163639821
  Velocity             =     4.887823615525028
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000002305873995

  Step  /  MaxStep     =   1 / 1
  iter  /  MaxIter     =   9 / 30
  Control Parameter    =     2.641842624544382
  Period               =     0.043900163647529
  Velocity             =     4.887823617448827
  MachineEps           =     0.000000010000000
  CC_Norm              =     0.000000000632522
-----------------------------------------------------------------
                        Convergence  !!
-----------------------------------------------------------------
  ...........
```

ただし，プログラムのバグで「Convergence !!」と表示されていても，周期 (Period) が負になっていたり，極端に大きな値になっていたりする場合は，求めたい解に収束していないことに注意する．

5. データの保存

   step5.1 で求めた分岐図のデータを TB2_INI のディレクトリに保存する．

   $ ./aout␣TB2_INI ↵

Step5.2 進行解の分岐計算

Step5.2 では，Step5.1 で求めた解を用いて，振動進行解の枝を求める．
実行と保存を繰り返すため，詳しい説明は省略してデモプログラムを記しておく．

step5.2 のデモと操作方法

1. デモの読み込み

   $ cp␣../DEMO/Exothermic-reaction_Model/AS/(*1)/* ␣./ ↵

2. 実行

   $ ./aout␣1 ↵

3. 保存

   $ ./aout␣(*2) ↵

   上記の 1\UTF{FF5E}3 の操作を 2 回繰り返す．(*1)，(*2) は，
   1 回目　(*1):19_TB_01　(*2):TB_01
   2 回目　(*1):20_TB_02　(*2):TB_02

3 回目　(*1):21_TB_03　(*2):TB_03
4 回目　(*1):22_TB_04　(*2):TB_04
5 回目　(*1):23_TB_05　(*2):TB_05
と変更して実行する．

分岐図を出力するには，それまでに保存したデータを結合する．

```
$ cat␣TB_01/81_CP_TV_NR.dat␣TB_02/81_CP_TV_NR.dat␣TB_03/81_CP_TV_NR.dat
␣TB_04/81_CP_TV_NR.dat␣TB_05/81_CP_TV_NR.dat ＞ tmp.dat ↵
$ mv␣tmp.dat␣81_CP_TV_NR.dat ↵
```

次のコマンドで定常解と Step3 で求めた進行解の分岐図と一緒に分岐図を出力できる．

```
$ gnuplot ↵
$ p "SP_F/81_CP_TV_NR.dat"␣u␣1:4␣w␣l, "TP2_F/81_CP_TV_NR.dat"␣u␣1:4␣w␣l
, "81_CP_TV_NR.dat"␣u␣1:4␣w␣l ↵
```

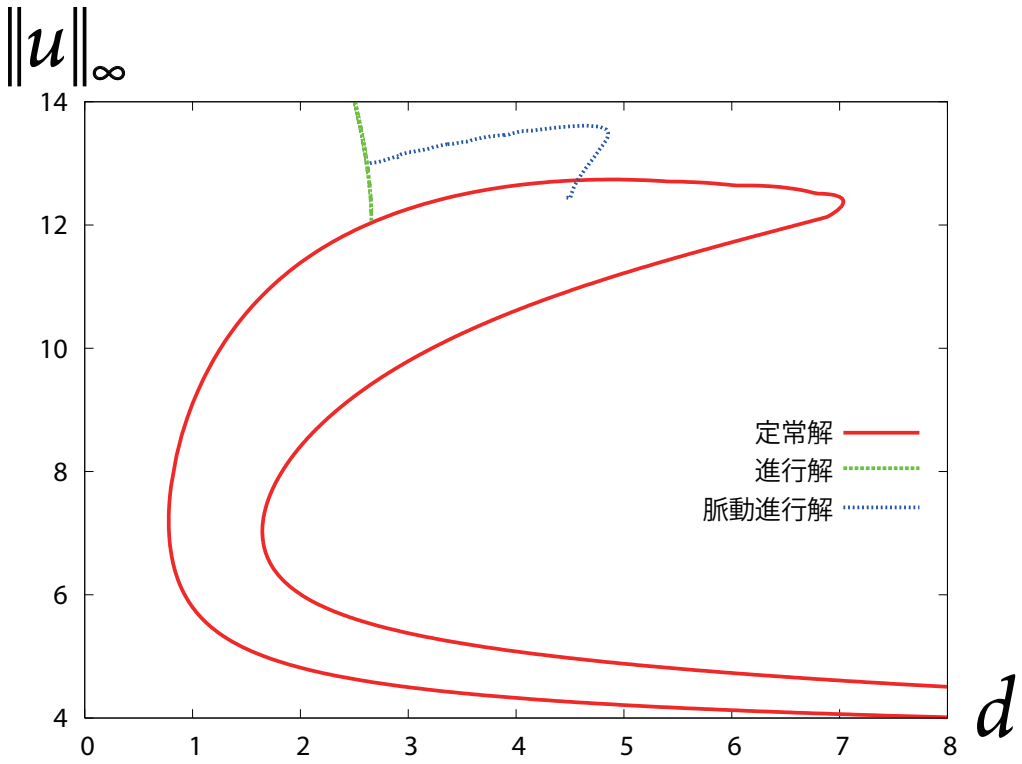


図 6.2.11:

Step5.2 で求めたデータを TB_F というディレクトリに保存する．

```
$ ./aout␣TB_F ↵
```

Step6 脈動定常解の分岐計算

Step6 では，定常解の Hopf 分岐点より分岐する振動定常解の分岐図を求める．
しかし，脈動定常解の枝を追うのは非常に難しいため，このマニュアルでは取り扱わない．
分岐図は以下のようになる．

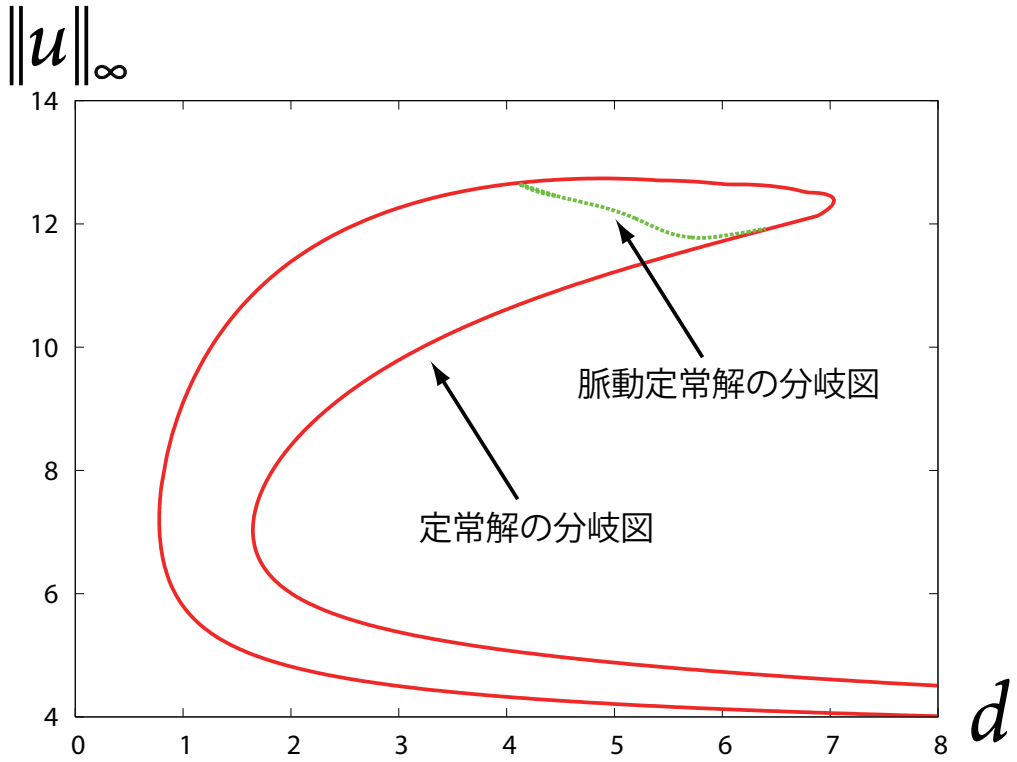


図 6.2.12:

Step7 分岐図

今まで求めた分岐図をまとめて出力すると図 6.2.13 のようになる.

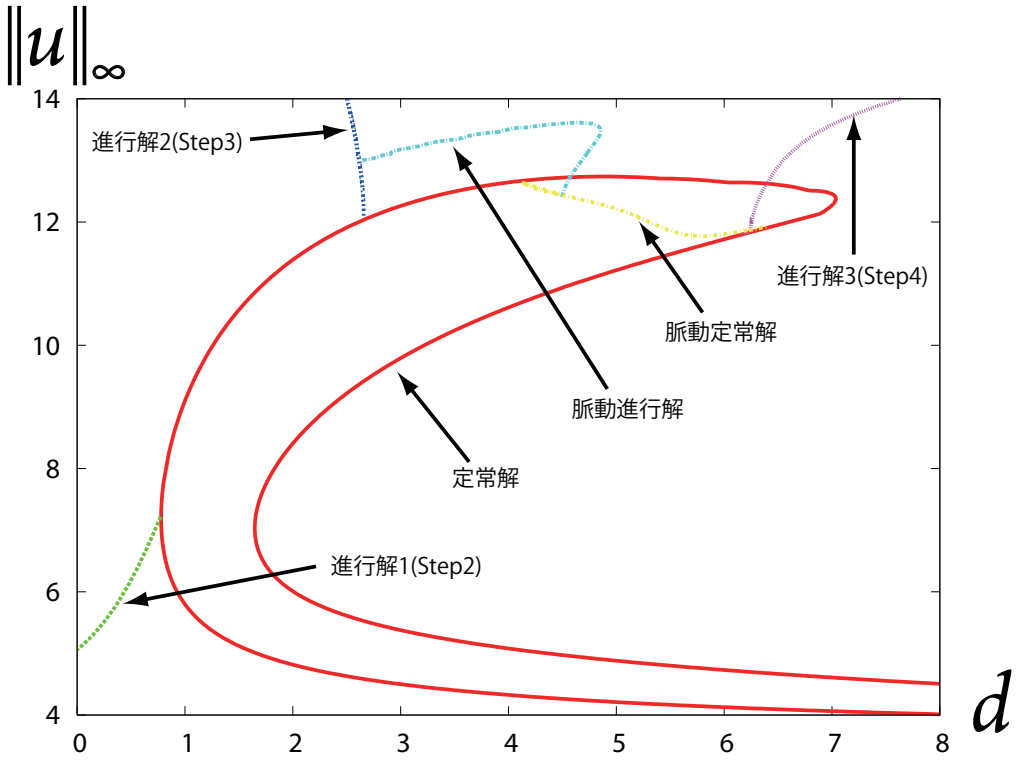

$\|u\|_\infty$
14
12
10
8
6
4
0
1
2
3
4
5
6
7
8
d
進行解2(Step3)
進行解3(Step4)
脈動定常解
脈動進行解
定常解
進行解1(Step2)

図 6.2.13: 分岐図

第7章　付録

7.1　鳥瞰図

この節では，第 6 章で求めた解の鳥瞰図を描く工程を説明する．
この節では，GLSC がインストールされていることを前提としている．
ここでは例として，RD/DEMO/Allee-EFFECT_Model/TE/04_bird のデータを用いて鳥瞰図を描く．

Step1 必要なデータを作成する

時間発展方程式を解くディレクトリに移動 ($ cd␣RD/TE ↵) する．
RD/DEMO/Allee-EFFECT_Model/TE/04_bird のデータを計算するディレクトリに持ってくる．

```
$ cp␣../DEMO/Allee-Effect_Model/TE/04_bird/*␣./ ↵
```

1. MaxStep と StepCut を決める．

   MaxStep と StepCut は好きなように指定してよいが，MaxStep/StepCut の値によって作成される鳥瞰図のメッシュが異なる．
   MaxStep/StepCut が大きくなると，線が密になる．図 7.1.1
   MaxStep/StepCut が小さくなると，線が疎になる．図 7.1.2

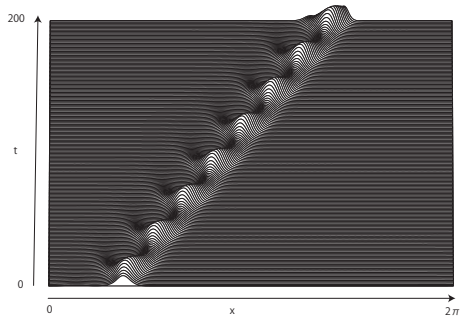


図 7.1.1: MaxStep/StepCut = 200

図 7.1.2: MaxStep/StepCut = 40

ここでは，MaxStep/StepCut = 200 となるように以下のように設定する．

```
    ............
 10   ;  StepCut                  ;  OP[12]
2000  ;  MaxStep                  ;  OP[13]
    ............
 2    ;  Initial_Select           ;  OP[15]
```

2. 実行

   $ ./aout ↵

3. 実行結果

   図 7.1.3 のような振動進行パルスが出る.

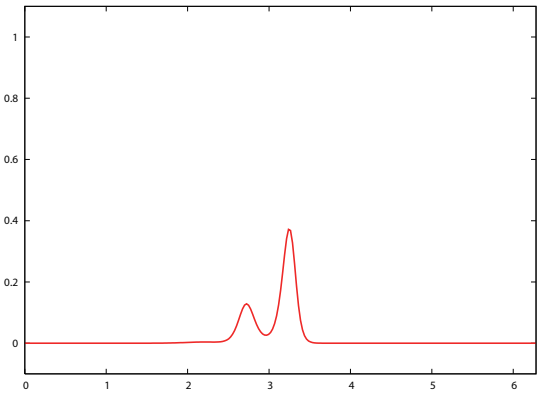


図 7.1.3: 振動進行パルス

実行すると，鳥瞰図を描くために必要なデータ (solution_value.dat) が作られる.

Step2 鳥瞰図を描く

鳥瞰図は RD/TE で描く.

1. glsc の設定

   glsc を次のように設定する.

```
        .....
#-----------------------------------------------------------------
  $Initial_Select        = 2  ; #(*1)
  $OutPut_Cut_Data       = 1  ; #この数字ごとのデータが出力される.
  $U_SolutionValue_Point = 3  ; #solution_value.dat の 3 列目を u とする.
  $V_SolutionValue_Point = 4  ; #solution_value.dat の 4 列目を v とする.
#-----------------------------------------------------------------
  $Dx_Zbottom_U   =  0.0;       #出力するときの u の 0 の位置.
  $Dx_Ztop_U      =  0.0;       #出力するときの u の高さに関係する.
        .....
  $Graph_X        =  30.0;      #出力するときの横幅.
  $Graph_Y        =  20.0;      #出力するときの縦幅.
        .....
```

(*1) Initial_Select の値によって出力されるものが違う.

- InitialSelect = 1 の時，u と v のそれぞれの鳥瞰図が並んで出力される.
- InitialSelect = 2 の時，u の鳥瞰図が出力される.
- InitialSelect = 3 の時，v の鳥瞰図が出力される.
- InitialSelect = 4 の時，最初に u と v のそれぞれの鳥瞰図が並んで出力され，画像をクリックするたびに u の鳥瞰図，v の鳥瞰図，と出力されていく.

2. 実行

次のコマンドで鳥瞰図が作成される.
$ ./glsc ↵

3. 実行結果

図 7.1.4 のような鳥瞰図が出力される．また，画像を消すことでディレクトリ TE/DATA が作成され、画像 TE/DATA/U.i00.epsi が作られる.
Initial_Select = 1 の時は，U_V.i00.epsi が作られる.
Initial_Select = 3 の時は，V.i00.epsi が作られる.
Initial_Select = 4 の時は，U.i00.epsi，V.i00.epsi，U_V.i00.epsi が作られる.

次のコマンドで画像が作られていることを確認できる.
$ display␣DATA/U.i00.epsi ↵

4. 視点を変える

次のコマンドで出力される画像の視点が変わる.
$ ./glsc␣pXX ↵

ただし，XX には 00 から 90 までの数字が入り，デフォルト ($ ./glsc ↵) では XX = 80 として出力される．図 7.1.4

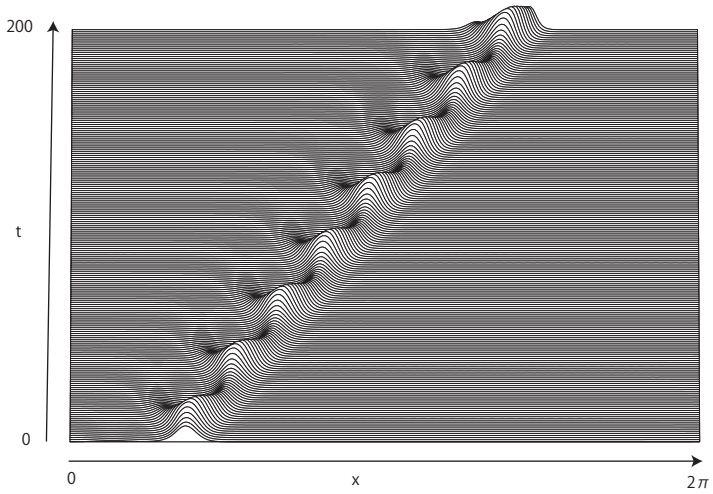


図 7.1.4: 鳥瞰図 p80

`XX = 90` では真上からの視点になる．図 7.1.5
`XX = 00` では真横からの視点になる．図 7.1.8

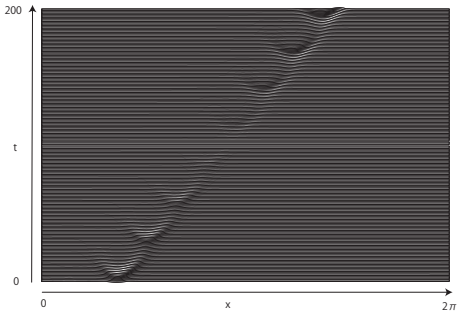


図 7.1.5: 鳥瞰図 p90

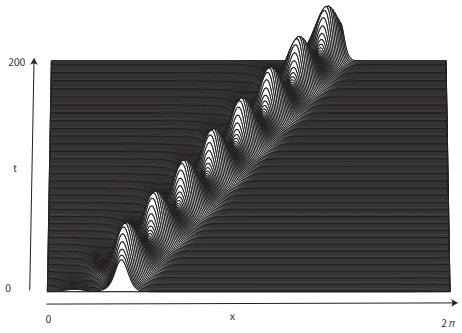


図 7.1.6: 鳥瞰図 p60

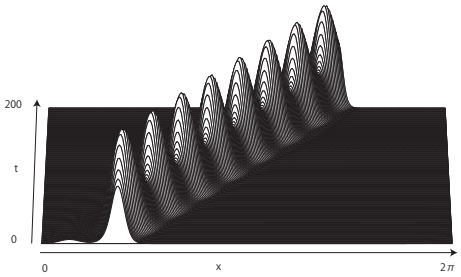


図 7.1.7: 鳥瞰図 p30

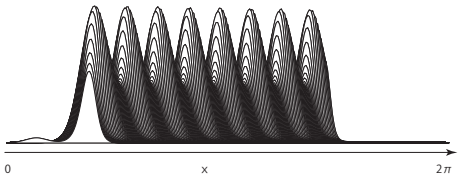


図 7.1.8: 鳥瞰図 p00

7.2 実行時に起きるエラーの対処法

$./aout 1 実行時に以下のようなエラーが起きた場合に対する対処法を記載する.

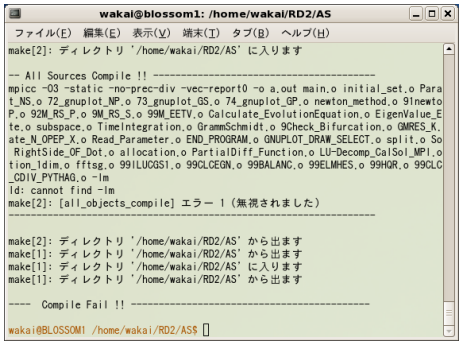


図 7.2.1: エラー時のキャプチャ

上の図のようなエラーが発生した場合，以下のコマンドを使うことで解決できる可能性がある.

```
$ yum search glibc-static ↵
$ su ↵
# yum install glibc-static.x86_64 ↵
```