TR-H-214

Diffpackを用いた2次元非圧縮性流体のシミュレーション —管楽器発音のモデル化にむけて—

藤坂洋一（工学院大／ATR-HIP），足立整治

1997.3.31

ATR人間情報通信研究所

〒619-02　京都府相楽群精華町光台2-2　TEL: 0774-95-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

Telephone: +81-774-95-1011

Fax : +81-774-95-1008

もくじ

1　はじめに

クラリネットなどのリード木管楽器では、リードの振動が、音源となって発音する。トランペットなどの金管楽器では、唇がリードの役目を果たす。両リード楽器では、リードや唇によって作られる狭めによって、気流が制御される。その制御の仕組みを決めるのは流体力学である。しかし、流体力学で解析的に解ける問題は限られており、実際のリード付近の気流の様子を解析的に求めることは困難である。現在までにモデル化にはベルヌーイの定理や、運動量保存の法則を用いてきた。おおざっぱな傾向を観測するうえでは、これらのモデル化は非常に有用であるが、正確なシミュレーションを行うためには不十分である。

エアリード楽器には、オルガンパイプのようにジェットがエッジの上下を揺動して駆動するものがある。オルガンパイプのようなジェット駆動楽器は、機械的に振動するリードを持たない。その代わりに、吹き付けられた空気ジェットを2分し、一方の空気を楽器内に導き、他方を楽器外に逃がすためのエッジを備えている。ジェットは、パイプの歌口付近の音響場に影響され、上下に揺動する。その結果、パイプ内に流入する空気の流量が変化する。しかし、音響場がジェットに与える影響は、理論的にも実験的にも完全には明らかにされていない。

上記の二つのように管楽器には現在モデル化の遅れている部分が存在する。管楽器の音を合成し、その音を正確に再現するためには、楽器内における流体の振る舞いをモデル化する必要がある。そのモデル化を図る有力な手段として、数値シミュレーションが挙げられる。数値シミュレーションは、近似的に様々な問題を解くことができる。その結果、問題が非常に複雑で解析的に解けないような場合においても、柔軟に対応が可能である。

本稿では、Diffpackを用いて非圧縮性流体解析を行う際の基本的な使用法について、インストールから、簡単なシミュレーションまでについて述べる。Diffpackは、ノルウェイのオスロ大学プロジェクトチームによって開発が進められている偏微分方程式(PDE)の解を求め、シミュレーションを行うためのC++オブジェクト志向のフリーなライブラリである。ライブラリの構成は、BasicTool,LaTool,DpKernel,DpUti l,DpApplから成り、これらのライブラリはそれぞれ以下のような役割を持っている。

・ BasicTool
メモリ管理、配列、メニュー、入出力などの基本ライブラリ。

・ LaTool
ベクトル、行列、逆行列や固有値解法を含む線形システム関連のライブラリ。

・ DpKernel
格子や要素の型、数値微分などを含む有限要素プログラミングのライブラリ。

・ DpUtil
格子や要素データの入出力、プリプロセッサ関連のライブラリ。

・ DpAppl
対流拡散方程式、NS 方程式、楕円形の初期値及び境界値問題など普遍的な上位のライブラリ。

Diffpack の入手は、Diffpack のホームページである
http://www.oslo.sintef.no/avd/33/3340/diffpack
から可能である。

2 Diffpack のインストール

2.1 インストール機種及び OS

Diffpack Public Access Release 1.0 のインストールは、以下のプラットホームで可能である。

hp9000s700 (Hewlett Packard/HP-UX),
iris4d (Silicon Graphics/IRIX),
sgi (Silicon Graphics/IRIX),
sparc (Sun Sparc/Solaris),
sun4 (Sun Sparc/SunOS),
rs6000 (IBM RS6000/AIX).

本稿では、インストールに使用する OS および HOSTTYPE は、それぞれ

OS : SunOs-4.1.3
HOSTTYPE: sun4

とする。HOSTTYPE は、シェルのコマンドライン上で、echo コマンドを使用することにより、確認できる。

```
UNIX> echo $HOSTTYPE
UNIX> sun4
```

2.2 必要なアプリケーション

gcc-2.7.2,
GNUmake-3.74.

上記の C コンパイラ及び make はインストールの際に絶対必要となる。
インストール時ではないが、Diffpack で用いることのできるアプリケーションは、
plotmtv,
gnuplot,
xgraph,
xplot,
xgen.
結果の出力に上記のいずれかが必要となる。(本稿では plotmtv を用いる。)
xmotif
GUI のメニューにそっていろいろ動かすときに必要だが、無くてもコマンドライン上で十分動かせるのでほとんど必要無い。(本稿では用いない。)

2.3 インストールの準備

2.3.1 ソースの解凍、展開

Diffpack の圧縮されたソース

demo.tar.Z - HTML ベースのデモ。
doc_ps.tar.Z - ポストスクリプト形式のチュートリアルレポート
doc_examples.tar.Z - チュートリアルイグザンプル
doc_ref.tar.Z - ポストスクリプト及び HTML 形式のリファレンスマニュアル。
* etc.tar.Z - セットアップファイル。
* bin.tar.Z - UNIX のシェルスクリプトファイル。
* bt.tar.Z - Basic Tools
* la.tar.Z - Linear Algebra Tools
* dp.tar.Z - DpKernel, DpUtil, DpAppl

(*: インストールの際に解凍、展開しておかなければならないファイル。)

を uncompress, tar する。

```
UNIX> uncompress file.tar.Z
UNIX> tar xvof file.tar
```

2.3.2 ソースの書き換え、BUG FIX

・書き換え

1. etc/setup/dpcshrc の内容を以下のように書き換える。

setenv TIMR /SI/diffpack/pub1.0 を
setenv TIMR "diffpack のソースを解凍、展開したディレクトリ（フルパスで指定。）"
にする。(以下 Diffpack のカレントディレクトリを $TIMR とする。)

2. $TIMR/bin/Make を書き換える。
ホストタイプが sun4 では、'gnumake' と Make file の中で書かれてしまっているので
gnumake を make に換える。

注）自分の使っている make の名前を gnumake に変えてもよい。

・ BUG FIX

$TIMR/bt/src/libs/bt2/menu/MenuSystem/MenuItem.C:354
$TIMR/bt/src/libs/bt2/menu/MenuSystem/MenuSub.C:169

'last_before_prompt' to 'int(last_before_prompt)'

$TIMR/dpsrc/libs/dpK/fem/GridFE.C:582

'isLattice()' to 'int(isLattice())'

$TIMR/dp/src/app/utilities/graphfilters/common/GBexport.inc:400

'String(argv[1])' を
String tmpf = String(args[1]); s_e << "FILENAME : " << tmpf << "\ n" ;
// displaying the inputfile
Is inputfile(tmpf.chars(),ASCII,INFILE); // ASCII/xdr inputfile
と入力すれば、gb2mtv などの実行ファイルでうまくアーギュメントの1番目（入力ファイル）が読み込まれる。

BooLean Colours::isBlack() return getBooLean(r==0 && g==0 && b==0);
BooLean Colours::isBlack() return getBooLean(r==g && g==b && b==r);
上の2行をコメントアウトする。

$TIMR/bt/src/libs/bt1/HandleId.h:36-39

#ifdef SUN4_Cplusplus
// SunOs 4.x requires char*, not void*, as input to free
inline void free (void* ptr) free((char*)ptr);)
#endif
38行目をコメントアウトする。

$TIMR/bt/src/libs/bt1/IOs_xdr.C:549

outputfile = fopen(filename.chars(),"rb+"); // modify binary file

を
outputfile = fopen(filename.chars(),"r+b"); // modify binary file

に書き換える。これを行わないとバイナリファイルへの書き込みが正しく行われないためシミュレーション結果を表示する際に不都合が生じる。

・置き換え以下のファイルは、Diffpack プロジェクトでバグが確認され、修正が行われたファイルである。これらのファイルのソースコードは、/home/mokusei1/diffpack を参照のこと。

$LAR/src/libs/arrays2/templates
MatSparse.C and MatDiag.C

$DPR/src/libs/dpU/envir/vis

DrawFE.h

$DPR/src/libs/dpU/prep/supel

GeometrySupElSet.C, PartitonSupElSet.C and PreproSupElSet.C

$TIMR/bin

InstallPackage

InstallPackage は、

```
chmod a+x $TIMR/bin/InstallPackage
```

を施して実行形式にする。

以上でインストールのための準備が完了である。

2.4 インストール

インストールは、$TIMR で

```
UNIX> source etc/setup/dpcshrc
```

```
UNIX> InstallDiffpack
```

を実行すれば4時間ぐらいで (Sparc Station5) コンパイルが終了する。いろいろ聞いてくるが、気にしないでデフォルトのまま実行して大丈夫である。ただし、この場合 test level でインストールされる。

3 開発環境

開発環境の設定は、$TIMR/etc/setup/dpcshrc をソースとして組み込むことにより実現できる。毎回コマンドを打ち込むのも面倒なので、以下の設定を.cshrc に書き込んでやれば、シェルを立ち上げるたびに自動的に Diffpack を使用できる環境になる。

```
source /home/user/diffpack/etc/setup/dpcshrc
```

(Diffpack をインストールしたディレクトリが /home/user/diffpack のとき)

実際に C++ のソースをコンパイルするには、Diffpack 独自のシェルスクリプト Mkdir でディレクトリを作成し、そのディレクトリ下にソースを置き make する必要がある。具体的には

```
UNIX> Mkdir -dp -test “ディレクトリ名”
UNIX> cd “ディレクトリ名”
UNIX> Make
```

とする。シェルスクリプト Mkdir を実行し、アプリケーションのディレクトリを作成すると、その中に Makefile などが自動的に作成される。あとは、ディレクトリの中で、サンプルコードやソースコードを編集し、make すればディレクトリ下の *.C ファイルがコンパイル、リンクされ app という実行ファイルが作成される。

4 流体解析（Navier-Stokes 方程式）

Diffpack では、Navier-Stokes （NS）方程式を有限要素法で解き、シミュレーションが行えるアプリケーションが存在する。数値解析手法は、ペナルティ法を用いている。NS のソースコードは、

$TIMR/dp/src/libs/dpA/NS

にある。ここにあるファイルを Mkdir で作成したディレクトリ下に置き、Make すれば、app という実行ファイルができる。このファイルが有限要素法で流体解析するためのアプリケーションとなる。

4.1 メッシュの作成

実際に実行するためには、観測領域の境界条件の設定及び要素分割をしてやる必要がある。そしてそれらのデータを app に渡してやる必要がある。まず、境界条件および要素分割には、makegrid コマンドを使う。ここでは 2 次元のパイプについてメッシュを作成する。具体的には、

```
makegrid +iscl +casename firstNS \
        -m PreproBox \
        -g 'd=2 [0,0.9]x[0,0.3]' \
        -p 'd=2 e=ElmB4n2D div=[30,30] g=[1,1]' \
        -r 'nb=4 names=inlet outlet u0 v0 1=(3),2=(1),3=(2 4),4=(2 4)'\
```

makegrid のオプション
-m プリプロセッサ。PreproBox, PreproSupElSet, PreproGeomPack などがある。
-g ジェオメトリの情報。
-p パーティッションの情報。
-r 境界の再指定。
-a 境界の追加。

とする。この場合、領域は 0.3[m]x0.1[m] の長方形となり、境界条件は上下の平らな境界で流速を 0 とし、左から流入し、右へ流出するように想定している。分割した要素を確認するためには drawgrid コマンドを使い *.gb ファイルを作成する。そして、plotmtv コマンドで表示できる *.mtv ファイルに変換するために gb2mtv コマンドを用いる。以下のようにコマンドライン上から入力すれば、要素の分割および境界条件を確認することが出来る。

・要素分割の表示

```
UNIX> drawgrid firstNS 0
UNIX> gb2mtv firstNS.grid.gb > firstNS.grid.mtv
UNIX> plotmtv first.grid.mtv
```

・流入部の境界表示

```
UNIX> drawgrid firstNS 1
UNIX> gb2mtv firstNS.boundary.gb > firstNS.boundary_inlet.mtv
UNIX> plotmtv first.boundary_inlet.mtv
```

・流出部の境界表示

```
UNIX> drawgrid firstNS 2
UNIX> gb2mtv firstNS.boundary.gb > firstNS.boundary_outlet.mtv
UNIX> plotmtv first.boundary_outlet.mtv
```

・横方向への流速 0 の境界表示

```
UNIX> drawgrid firstNS 3
UNIX> gb2mtv firstNS.boundary.gb > firstNS.boundary_u0.mtv
UNIX> plotmtv first.boundary_u0.mtv
```

・縦方向への流速 0 の境界表示

```
UNIX> drawgrid firstNS 4
UNIX> gb2mtv firstNS.boundary.gb > firstNS.boundary_v0.mtv
UNIX> plotmtv first.boundary_v0.mtv
```

4.2 インプットファイルの作成

NS のアプリケーションを実行するためには、インプットファイルを作成しなければならない。このファイルは、実行ファイル app にメッシュの情報や、様々なパラメータを渡す際に必要となる。ファイルはテキスト形式で以下のようになる。

```
!!
!! NavierStokes flow in a rectangular pipe
!!

! geometry, partition and boundary info
set gridfile=firstNS.grid

! boundary convention:
!   1: inlet boundary with prescribed velocity field
!   2: outlet boundary with both normal derivatives equal to zero
!   3: u=0 and dv/dn=dw/n=0
!   4: v=0 and du/dn=dw/n=0
!   5: w=0 and du/dn=dv/n=0
set redefine boundary indicators = nb=4 names=inlet outlet u0 v0 1=(1),2=(2),3=(3),4=(4)
```

```
! parameters of quation (MKS units)
set time integration parameters = dt=0 ! steady flow
set inlet profile = 2  ! 1: parabolic profile at x=0, 2: uniform plug flow
set characteristic inlet velocity = 1.0
set viscosity = 1.809e-5
set density = 1.205
set penalty parameter = 1.0e4

! parameters of nonlinear solver
sub prm(NonLinEqSolver)
set nonlinear iteration method = NewtonRaphson
set max nonlinear iterations = 15
set max estimated nonlinear error = 1.0e-3
ok

! parameters of linear solver
sub LinEqAdm
sub prm(Matrix)
set matrix type = MatBand
ok

sub prm(LinEqSolver)
set basic method = GaussElim
ok

ok
ok
```

4.3 実行及び結果の表示

インプットファイルを作成したら、以下のコマンドで解析が行える。

```
UNIX> app +casename firstNS < firstNS.i
```

うまく、計算が終了すると、

.firstNS.field

に解析結果がテキストで出力される。このファイルからある時間に対する結果を *.mtv ファイルにするためのフィルタとして、simres2mtv コマンドがある。firstNS での 0.1 秒後の結果を可視化するためには、

```
UNIX> simres2mtv -f firstNS -n p -t '0.1;' -s -a
```

```
UNIX> simres2mtv -f firstNS -n v -t '0.1;' -v -a
```

simres2mtv のオプション

```
 -f      ケース名。              -f case name
 -r      フィールド番号。        -r 'field number;'
 -n      圧力か流速              -n p or v
 -s/-v   スカラー / ベクター。
 -a/-b   アスキー / バイナリー。
```

とコマンドライン上で入力する。スカラーデータは、

```
UNIX> plotmtv firstNS.scalar.mtv
```

として可視化できる。(Fig. 1) しかし、ベクターデータは、*.vector.mtv ファイルの中身を少しいじってやる必要がある。firstNS.vector.mtv の 5 行目付近の

```
#% vscale=
```

を

```
% vscale=0.02 # (見やすいように何回か調整する必要がある。)
```

に変更し、

```
UNIX> plotmtv firstNS.vector.mtv
```

とすれば、Fig. 2のように表示される。

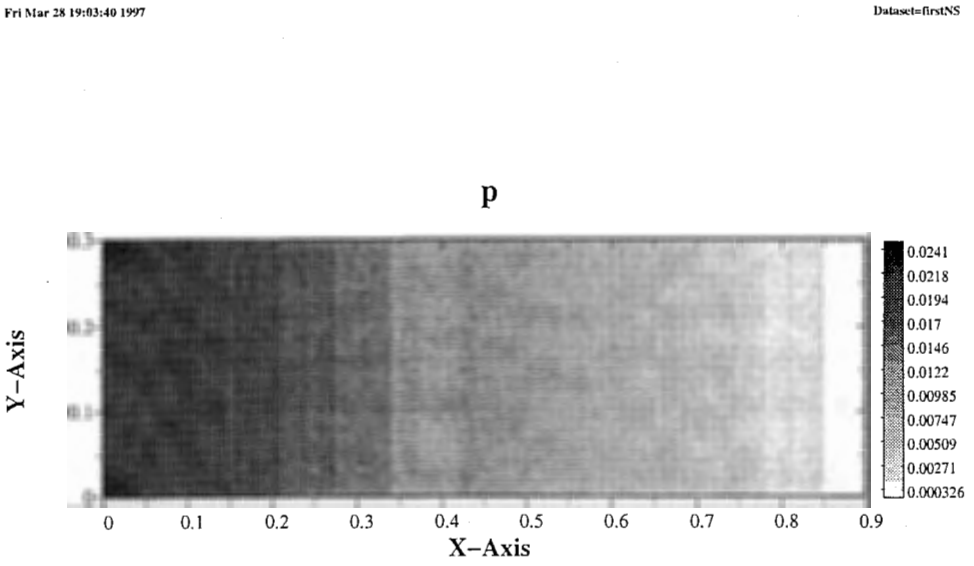

Fig. 1: firstNS の圧力分布

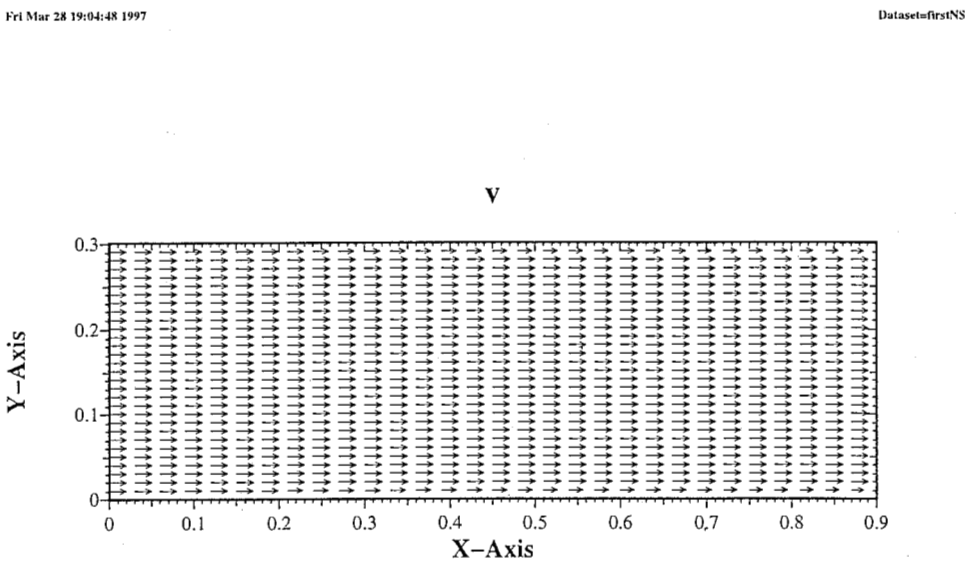

Fig. 2: firstNS の流速分布

5 境界および境界条件の自由な設定

メッシュを作成するプリプロセッサには、PreproBox, PreproSupElSet, PreproGeomPack などがある。しかし、ここでは自由に境界や境界条件を設定するために最低限必要な事柄について紹介する。よって PreproGeomPack については記述しない。

5.1 PreproBox

PreproBox は、四角形（3 次元の直方体も可能）の境界を作成するものである。たとえば、Fig. 3の様な壁を作成し左側の狭めから流体を流入させ右側の部分へ流出させたいなら、以下のように makegrid のオプションを定義する。

```
makegrid +iscl +casename jet \
        -m PreproBox \
        -g 'd=2 [0,0.9]x[0,0.3]' \
        -p 'd=2 e=ElmB4n2D div=[60,80] g=[1,1]' \
        -r 'nb=4 names=inlet outlet u0 v0 1=(),2=(1),3=(2 4),4=(2 4)'\
        -a 'n=5 b3=[0,0]x[0,0.1] b4=[0,0]x[0,0.1] \
                b1=[0,0]x[0.1,0.2] \
                b3=[0,0]x[0.2,0.3] b4=[0,0]x[0.2,0.3]'
```

-m: PreproBox で境界の形状を長方形（3 次元では直方体）と定義している。-g: 境界の領域を d=2 で 2 次元とし、その領域のスケールを 0.9x0.3[m] としている。
-p: 境界の領域は、2 次元。縦横の分割数は 60x80 とし、g=[1,1] により、メッシュの幅は均等としている。（後述の PreproSupElSet で用いる -grading と同等。） -r: 境界条件の名前（名前は何でもよい。）を inlet, outlet, u0, v0 とし、Fig. 4に示す辺をどの境界条件にするのか決定している。NS のアプリケーションでは、1 を流入部、2 を流出部、3 を横方向の流速 0[m/s]、4 を縦方向の流速 0[m/s] と定義してある。したがって、この場合には、上下の辺を流速 0 とし、右の辺を流出部としている。
-a: ここでは、境界条件を更に付加し、細かく設定することが出来る。b1 は流入部、b2 は流出部、b3 は横流速 0、b4 は縦流速 0 を示している。この場合には、x=0 において、流入部を y=0.1 から 0.2 の範囲とし、y=0 から 0.1, 0.2 から 0.3 の範囲を流速 0 と定義している。

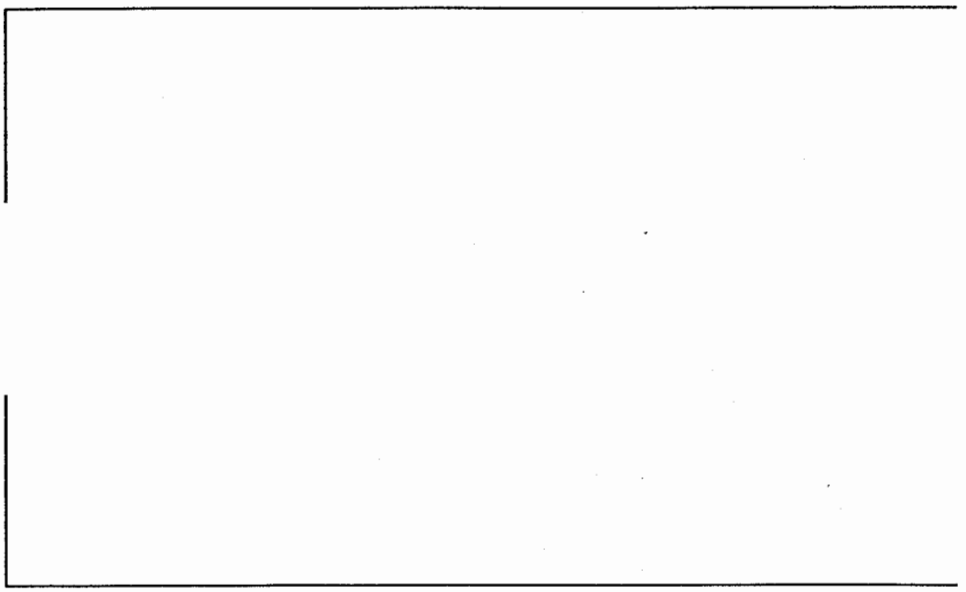

Fig. 3: 流速 0 の境界（壁）

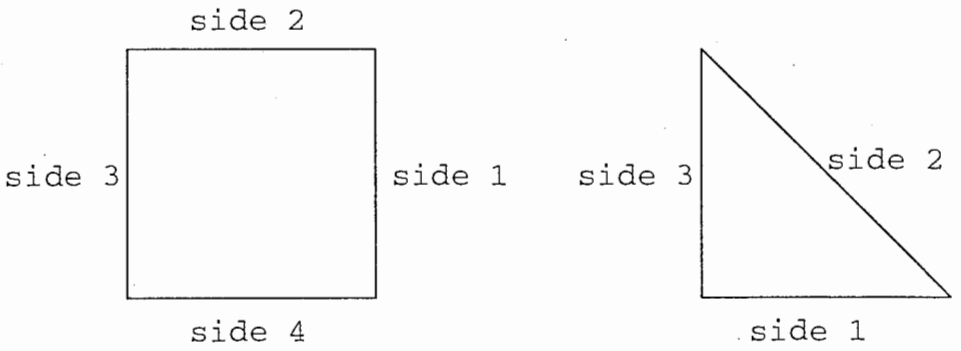

Fig. 4: 各辺の番号付

5.2 PreproSupElSet

PreproSupElSet は、-g, -p オプションの定義は PreproBox と違い *.geom, *.part ファイルを読み込ませる。これらのファイルはテキストで書き、そのフォーマットは以下のようになる。

・curve_pipe.geom

```
>no_of_dimensions = 2; > subdomains = 1; >no_of_supels = 7;
>no_of_ind = 4; >name b1 b2 b3 b4;

>SupEl; >subdomain_no = 1; >elementtype = ElmB8n2D;
>boundary= [1 (3)] [2 (4)] [4 (2)];
>nodes=[1(0 0.7)]+[2(0.15 0.7)]+[3(0 0.85)]+[4(0.15 0.85)];
>sides = ;
end SupEl1;

>SupEl; >subdomain_no = 1; >elementtype = ElmB8n2D;
>boundary= [2 (4)] [4 (2)];
>nodes=[2 (1.4 0.7)]+[4(1.4 0.85)];
>sides = [3 (1 1)];
end SupEl2;

>SupEl; >subdomain_no = 1; >elementtype = ElmB8n2D;
>boundary= [2 (4)] [4 (2)];
>nodes=[2 (1.506 0.656)]+[4(1.612 0.762)]+[5(1.457 0.689)]+[6(1.514 0.827)];
>sides = [3 (2 1)];
end SupEl3;

>SupEl; >subdomain_no = 1; >elementtype = ElmB8n2D;
>boundary= [2 (4)] [4 (2)];
>nodes=[2 (1.55 0.55)]+[4(1.7 0.55)]+[5(1.538 0.607)]+[6(1.677 0.665)];
>sides = [3 (3 1)];
end SupEl4;

>SupEl; >subdomain_no = 1; >elementtype = ElmB8n2D;
>boundary= [2 (4)] [4 (2)];
>nodes=[2 (1.506 0.444)]+[4(1.612 0.338)]+[5(1.538 0.493)]+[6(1.677 0.435)];
>sides = [3 (4 1)];
end SupEl5;

>SupEl; >subdomain_no = 1; >elementtype = ElmB8n2D;
```

```
>boundary= [2 (4)] [4 (2)];
>nodes=[2 (1.4 0.4)]+[4(1.4 0.25)]+[5(1.457 0.411)]+[6(1.514 0.273)];
>sides = [3 (5 1)];
end SupEl6;

>SupEl; >subdomain_no = 1; >elementtype = ElmB8n2D;
>boundary= [2 (4)] [4 (2)];
>nodes=[2(0.2 0.4)]+[4(0.2 0.25)];
>sides = [3(6 1)];
end SupEl7;
```

◎ Subdomain number: サブドメインの数。通常は1。
◎ Super element type: スーパーエレメントは通常の有限要素として定義し、2次元では、ElmB4n2D, ElmB8n2D をタイプとして選ぶことが出来る。エレメントのノードの構成は後述する。
◎ Boundary information: スーパーエレメントは、エレメントの辺に自由に境界条件を与えることが出来る。たとえば>boundary=[4 (1 2 4)] と定義したとすると、この意味は辺1, 2, 4の全てのノードがバウンダリーインディケータの4となることである。
◎ Nodal coordinates: スーパーエレメントは、エレメントの各ノードの座標値を自由に変更できる。
◎ Side connections: もし、すでにスーパーエレメントを定義していれば、となりにスーパーエレメントを接続していくことが出来る。たとえば>sides=[3 (1 1)] とスーパーエレメントの2番目に定義すれば、2番目のスーパーエレメントの辺3は1番目の辺1と同じと見なすということになる。

・ curve_pipe.part

```
>nsd = 2; >no_of_supels = 7;

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [2,8]; >grading = [1,1.3];

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [4,8]; >grading = [-1.5,1.3];

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [4,8]; >grading = [1,1.3];

>SupEl; >nsd = 2;
```

```
>elementtype = ElmB4n2D; >divisions = [4,8]; >grading = [1,1.3];

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [4,8]; >grading = [1,1.3];

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [4,8]; >grading = [1,1.3];

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [4,8]; >grading = [-0.6,1.3];
```

◎ Element type: 実際にメッシュを作成するときに次の要素タイプから選ぶことが出来る。ElmB4n2D, ElmB8n2D, ElmB9n2D, ElmT3n2D, ElmT6n2D など。エレメントのノード構成を Fig. 5に示す。

◎ Divisions for each space dimension: スーパーエレメントを縦横何分割するかを定義できる。たとえば¿divisions = [4,4] なら、そのスーパーエレメントを 4x4 のエレメントで構成することを示す。

◎ gradings: エレメントの密度を変化させるときに用いる。もし、grading が 1.0 より大きければ、エレメントの密度は境界に近付くにつれて増えて行く。逆に 1.0 より小さければ、領域の真ん中に近付くにつれ密度があがる。

*.geom, *.part ファイルが作成出来たら、makegrid コマンドを用いて以下のようにオプションを指定する。メッシュの作成結果は、Fig. 6に示す。

```
makegrid -m PreproSupElSet \
         -g FILE=curve_pipe.geom \
         -p FILE=curve_pipe.part \
         -c curve_pipe
```

（ここでの -c オプションは、+casename オプションと機能は同じである。）

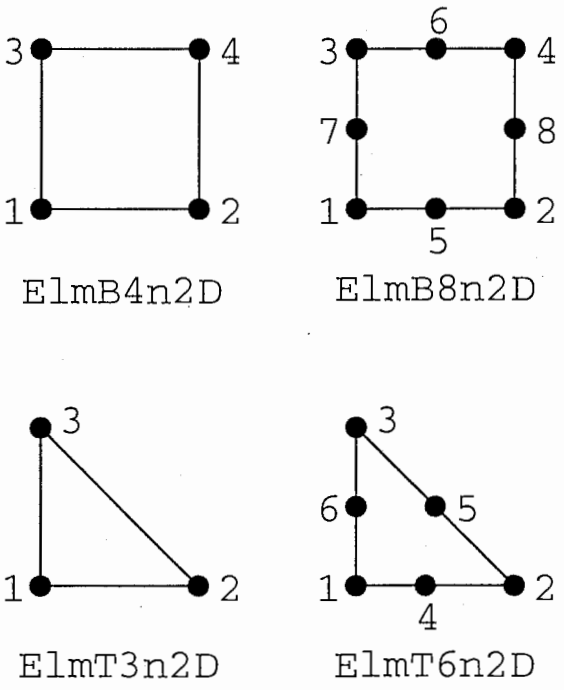

Fig. 5: エレメントのノード

Fri Mar 28 23:40:03 1997

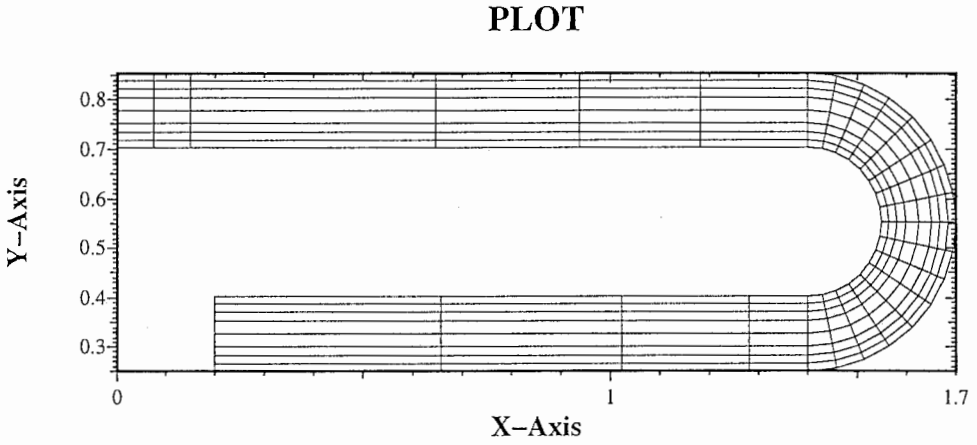

Fig. 6: PreproSupElSet を用いたメッシュの作成例

6 計算の妥当性

ここでは、2次元のパイプをモデルに用いて、シミュレーション結果の妥当性について解説する。

6.1 モデル

2次元のパイプにおいてパイプの長さが無限に長いと仮定すると、パイプ内の流速の分布は、y軸方向に変化する放物流（ハーゲン・ポアズイユ流れ）となる。このとき放物流とx方向に対する圧力勾配との間には、pを圧力、uをx方向の流速とすると、以下のような関係が成り立つ[1]。

$$\frac{dp}{dx} = \mu\frac{d^2u}{dy^2} = -P(constant)(-P\text{は、長さ1あたりの圧力降下}) \tag{1}$$

さらに、パイプの端で流速が0となる条件から、y方向に関する流速の分布は次式で表すことができる。

$$u = \frac{P}{2\mu}(h-y)y \tag{2}$$

(h: パイプの幅)

6.2 検証

実際にパイプの長さを無限にすることは数値計算では不可能である。したがって、観測領域は、x=0から0.5[m], y=0から0.05[m]としてシミュレーションを試みた。流入部は、y方向に一様な速度u=0.05[m/s]を与え、十分に時間が経ち、定常な状態になるまで計算を行った。圧力分布と速度分布を同時に表示した結果をFig. 7に示す。Fig. 8は、x=0から0.5における速度分布をEq. 2からの理論値（実線）とシミュレーション結果（○印）を示したものである。下流に行くにしたがって理論値に近付くのが観測できる。また、Fig. ??は、xに対する圧力勾配を示したものである。図中点線は、Eq. 2からの理論値（約0.00394）を示し、実線がシミュレーションによる結果である。Fig. 8において、理論値との差が無くなるにつれて、圧力勾配も理論値に近付くのが判る。これらの結果から、Diffpackによるシミュレーション結果は、妥当であると推測できる。

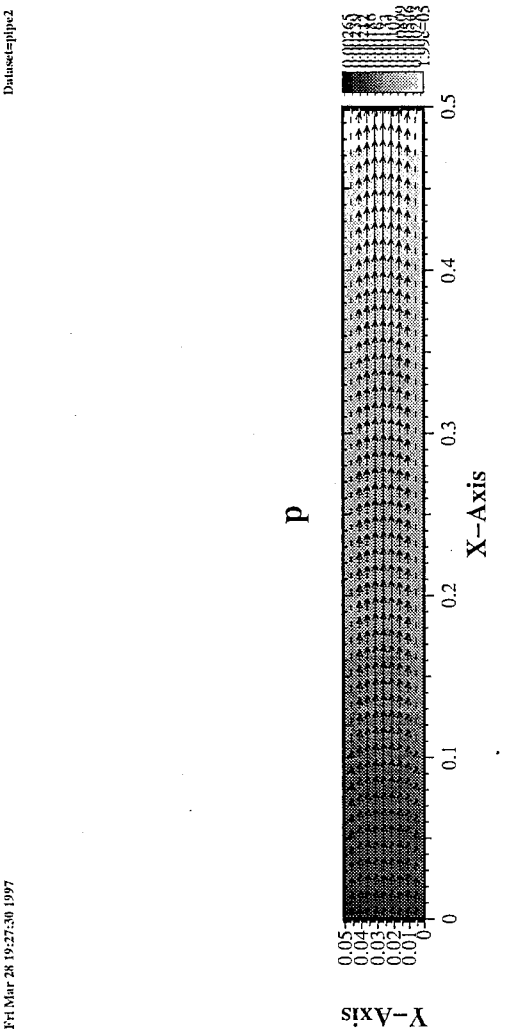

Fig. 7: シミュレーション結果 (plotmtv の -l オプション (ランドスケープにしたいとき用いる。) 及び -plotall オプション (圧力と速度の分布など同時に表示させたいとき用いる。) を用いて表示している。コマンドラインから、 plotmtv -l -plotall pipe2.scalar.mtv pipe2.vector.mtv)

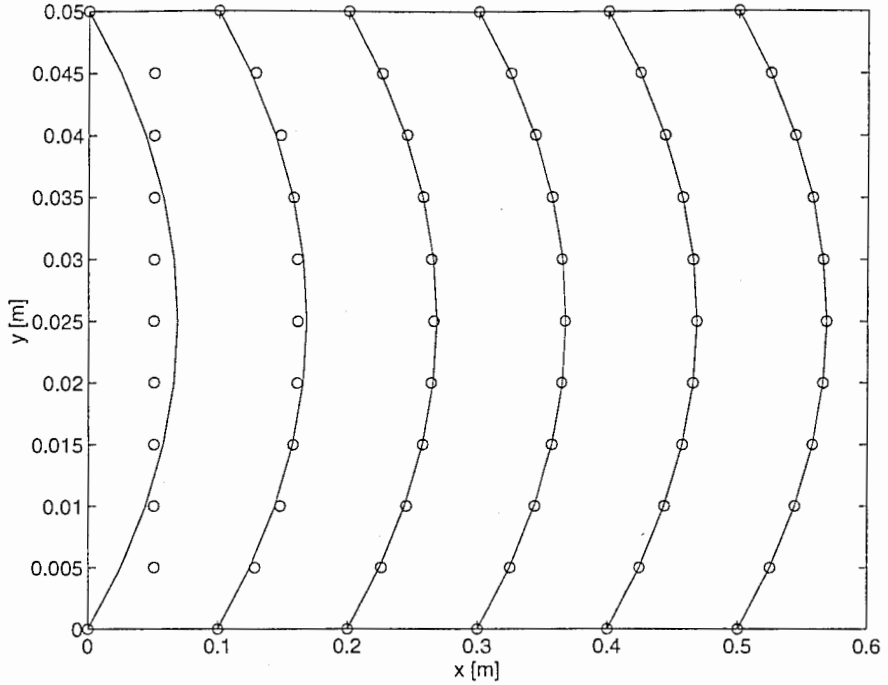

Fig. 8: 速度分布

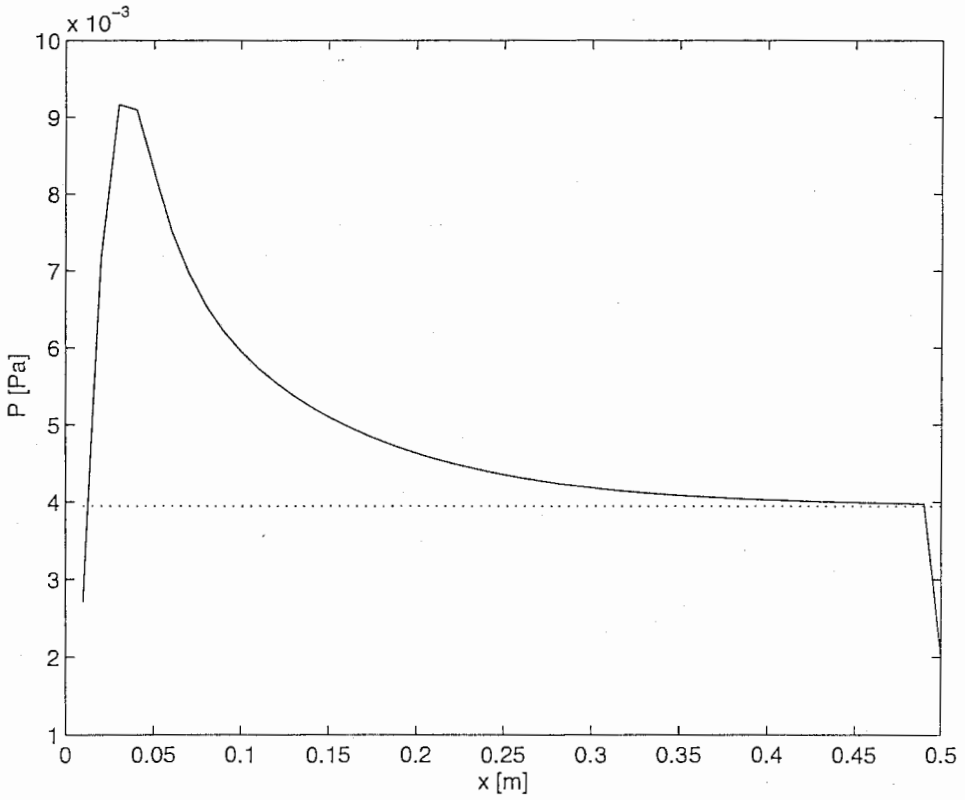

Fig. 9: 圧力降下

7 サンプル

7.1 流入部を自然な流れにするには

流入部の速度分布をハーゲン・ポアズイユ流れ [1] にするには、NS のアプリケーションのソースコード NavierStokes.C をハードコーディングする必要がある。0.9x0.3 の 2 次元パイプを考えた場合、流入部の速度分布をハーゲン・ポアズイユ流れ（パイプの端で流速 0）にするには、Eq. 2の速度分布を与えなければならない。しかし、デフォルトの NavierStokes.C の 293 行目には、以下のように記述されている。

```
v(1) = inlet_velocity * (1 - pow2(x(2)/0.01));
```

ここで、v(1) は x 方向の流速、inlet_velocity はインプットファイルに記述した characteristic inlet velocity、pow2 は () 内を自乗、x(2) は y 座標の値をそれぞれ示している。もちろんこれも x(2)=0 で最大値をとる放物流ではあるが、サンプル例としての目的とは異なる。したがって、ソースを書き換える作業が必要である。インプットファイルの characteristic inlet velocity をハーゲン・ポアズイユ流れの速度分布の平均値と考えると、293 行目は以下のように書き換えられる。

```
v(1) = inlet_velocity * (6 / pow2(0.3)) * (0.3 - x(2)) * x(2);
```

あとは、NavierStokes.C をコンパイル、リンクし、（Make をする。）実行すれば、Fig. 10,11のような定常状態の結果が得られる。実行の際には、インプットファイルの inlet profile を 1 にする。なお、メッシュの分割は 30x10、characteristic inlet velocity は 1[m/s] である。

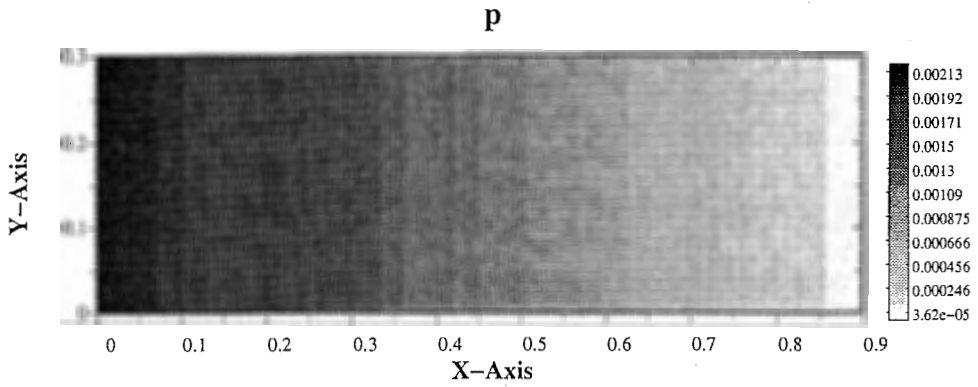

Fig. 10: 流入部をハーゲン・ポアズイユ流れにしたときの圧力分布

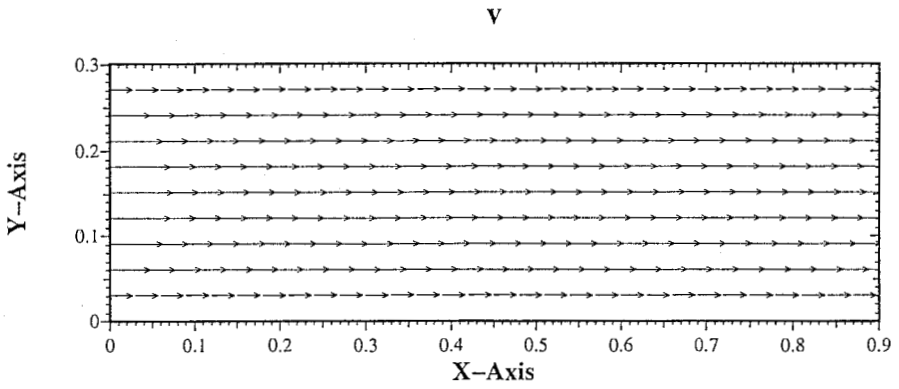

Fig. 11: 流入部をハーゲン・ポアズイユ流れにしたときの流速分布

7.2 上下対称形なら領域を半分にできる

もし、解析を行おうとしている境界がFig. 12に示すような上下に対称な形状で、レイノルズ数が低く乱れが生じない場合を考えるなら、観測領域を半分にして計算量の軽減をはかることが可能である。以下の *.geom, *.part ファイルから、PreproSupElSet を用いてメッシュを作成すると、Fig. 13のようになる。y=0 における境界条件は、x 方向の流速のみを考え、y 方向に関しては流速を 0 としている。シミュレーションの結果は、Fig. 14に示す。このシミュレーション結果は、流入部に y 方向に一様な速度 0.1[m/s] を与え、定常になったときの結果である。

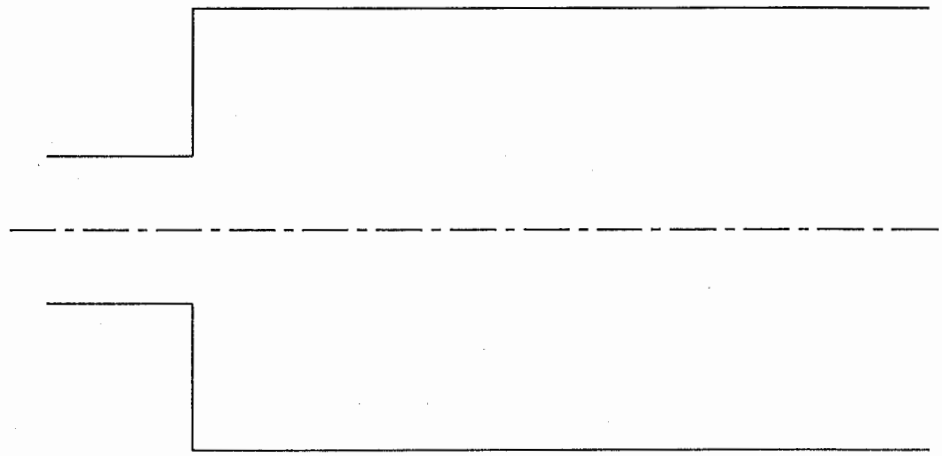

Fig. 12:

· tr_half2.geom

```
>no_of_dimensions = 2; > subdomains = 1; >no_of_supels = 3;
>no_of_ind = 4; >names b1 b2 b3 b4;

>SupEl; >subdomain_no = 1; >elementtype = ElmB4n2D;
>boundary= [1 (3)] [3 (2)] [4 (2 4)];
>nodes=[1(0 0)]+[2(0.005 0)]+[3(0 0.0015)]+[4(0.005 0.0015)];
>sides = ;
end SupEl1;

>SupEl; >subdomain_no = 1; >elementtype = ElmB4n2D;
>boundary= [2 (1)] [4 (4)];
>nodes=[1(0.005 0)]+[2(0.03 0)]+[3(0.005 0.0015)]+[4(0.03 0.0015)];
>sides = [3(1 1)];
end SupEl2;
```

```
>SupEl; >subdomain_no = 1; >elementtype = ElmB4n2D;
>boundary= [2 (1)] [3 (2 3)] [4 (2 3)];
>nodes=[1(0.005 0.0015)]+[2(0.03 0.0015)]+[3(0.005 0.0075)]+[4(0.03 0.0075)];
>sides = [4 (4 1)] ;
end SupEl3;
```

· tr_half.part

```
>nsd =2; >no_of_supels = 3;

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [5,3]; >grading = [1,1];

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [25,3]; >grading = [1,1];

>SupEl; >nsd = 2;
>elementtype = ElmB4n2D; >divisions = [25,12]; >grading = [1,1];
```

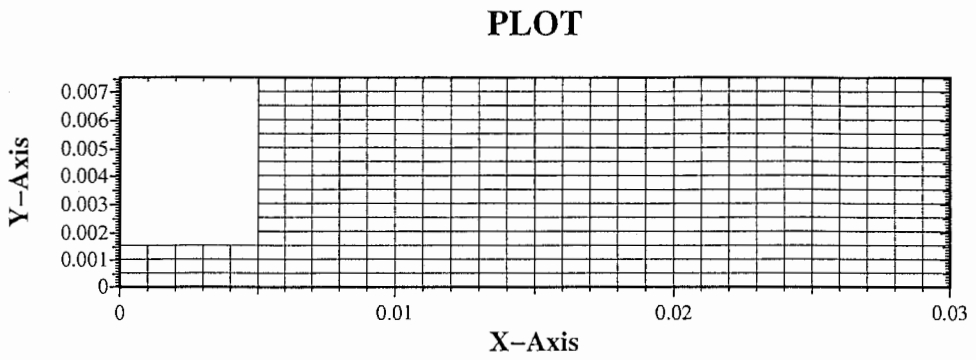

Fig. 13: 観測領域を半分にしたメッシュ

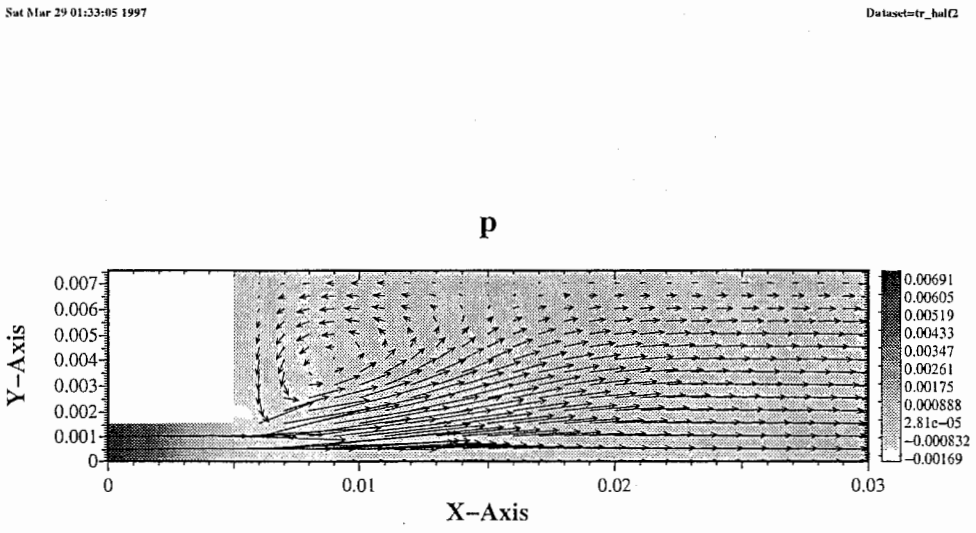

Fig. 14: 観測領域を半分にしたときのシミュレーション結果

7.3 カルマン渦

ここでは、流れが上下非対称になるカルマン渦 [2, 3, 4] の発生シミュレーションを試みる。以下のメッシュ作成のシェルスクリプトとインプットファイルによりシミュレーションを行うことができる。カルマン渦を発生させるためには、双子渦が崩れるまで計算を続けなければならない。そのためには、タイムステップ間隔を十分に小さくする必要がある。このサンプルでは、タイムステップ間隔を 0.005[s] とした。

・pipe_slit2_mkg.sh

```
#!/bin/sh
makegrid +iscl +casename pipe_slit2 \
        -m PreproBox \
        -g 'd=2 [0,0.9]x[0,0.3]' \
        -p 'd=2 e=ElmB4n2D div=[100,80] g=[1,1]' \
        -r 'nb=4 names=inlet outlet u0 v0 1=(),2=(1),3=(),4=(2 4)'\
        -a 'n=4 b3=[0,0]x[0.1,0.2] b4=[0,0]x[0.1,0.2] \
                b1=[0,0]x[0,0.1] \
                b1=[0,0]x[0.2,0.3]'
```

・pipe_slit2.i

```
!!
!! NavierStokes flow in a rectangular pipe
!!

! geometry, partition and boundary info
set gridfile=pipe_slit2.grid

! boundary convention:
!   1: inlet boundary with prescribed velocity field
!   2: outlet boundary with both normal derivatives equal to zero
!   3: u=0 and dv/dn=dw/n=0
!   4: v=0 and du/dn=dw/n=0
!   5: w=0 and du/dn=dv/n=0
set redefine boundary indicators=nb=4 names=inlet outlet u01 u02 1=(1),2=(2),3=(3),4=(4)

! parameters of quation (MKS units)
set time integration parameters = dt=0.005 t in [0.0,5.0]
set inlet profile = 2  ! 1: parabolic profile at x=0, 2: uniform plug flow
set characteristic inlet velocity = 5.0
set viscosity = 1.809e-5
set density = 1.205
```

```
set penalty parameter = 1.0e4

! parameters of nonlinear solver
sub prm(NonLinEqSolver)
set nonlinear iteration method = NewtonRaphson
set max nonlinear iterations = 15
set max estimated nonlinear error = 1.0e-3
ok

! parameters of linear solver
sub LinEqAdm
sub prm(Matrix)
set matrix type = MatBand
ok

sub prm(LinEqSolver)
set basic method = GaussElim
ok

ok
ok
```

シミュレーション結果をFig. 15,16に示す。流入部の流速0の境界の幅を基準の長さl(0.1[m])とすると、次式からレイノルズ数は、約33,306となる。

$$Re = \frac{Ul\rho}{\mu} \tag{3}$$

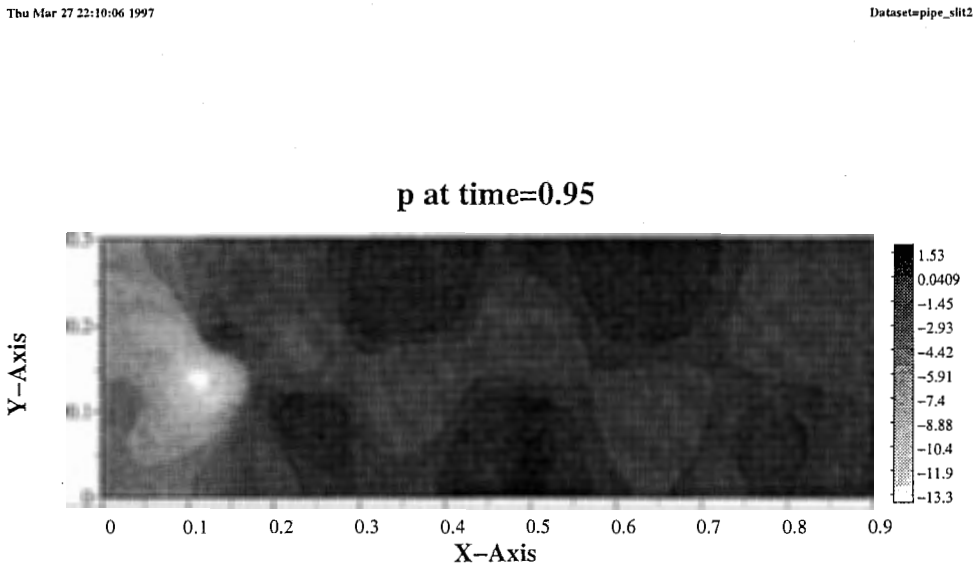

Fig. 15: カルマン渦（圧力分布）

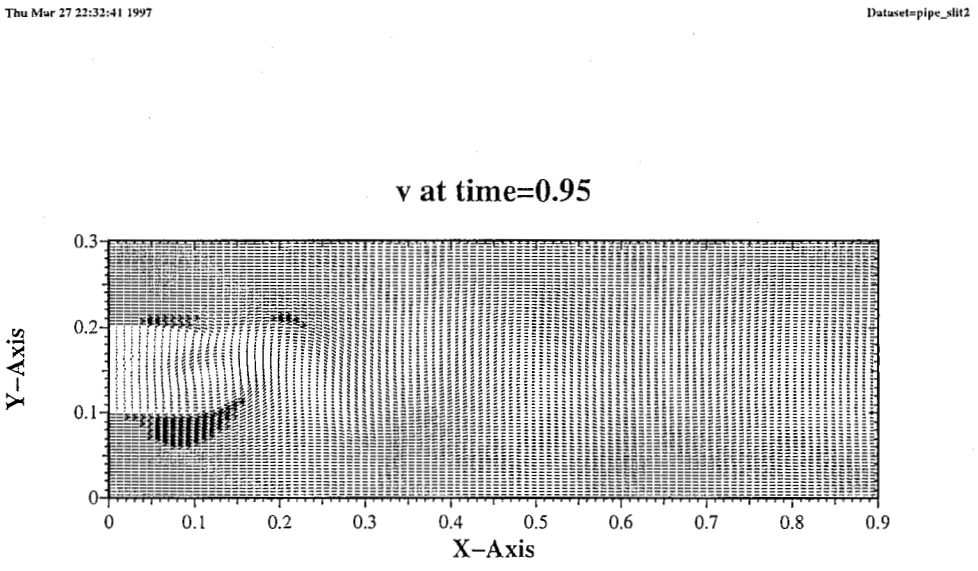

Fig. 16: カルマン渦（速度分布）

8 まとめ

本稿では、非圧縮性粘性流体の簡単なシミュレーションを Diffpack を用いて試みた。ここで取り上げた事象は非常に簡単なケースに限られているが、管楽器の流体シミュレーションを行う上で、 Diffpack における解析の妥当性を確認することができた。さらに、カルマン渦の発生シミュレーションの再現が可能であり、その結果非常に不安定な流体にも適用することが可能である事も実証できた。

数値流体解析を行う際、以下のような注意点および問題点があることを承知しておかなければならない。

1. 観測領域を広くとる必要がある。

たとえば、カルマン渦の発生シミュレーションでは、流体の流れを妨げる物体があるが、観測する上下の領域は本来、その物体の幅に対して８倍から１０倍とる必要がある。なぜなら上下の境界条件によって流体の振る舞いが影響されてしまう危険性があるからである。

2. 非定常な流体を解析するにあたっては、タイムステップの間隔を小さくとる必要がある。

その理由は、 NS のアプリケーションでは、ある程度の誤差の範囲内であったら、それを結果として次の計算を行っていくという点にある。これにより観測時間が長くなるにつれて誤差が増大してしまう。タイムステップの間隔を小さくすることによりこの傾向を軽減することができる。

3. 流速が大きいときには、観測領域を細かく要素分割をしなければならない。

これは、要素数が少ないと、有限要素法解析の分解能が悪くなるためであり、定常解を求める際にも同様なことが言える。

２や３の要因を解決しておかないと非定常解析では、ある程度の時間を観測すると、後に誤差が発散してしまい、途中で計算を打ち切らざるを得ないことがある。

参考文献

[1] 笠原英司 他, "図解 流体力学の学び方", オーム社 (1986).

[2] 河村哲也 他, "非圧縮性流体解析", 東京大学出版会 (1995).

[3] 吉澤徹 他, "乱流解析", 東京大学出版会 (1995).

[4] 巽 友正, "流体力学", 培風社 (1982).