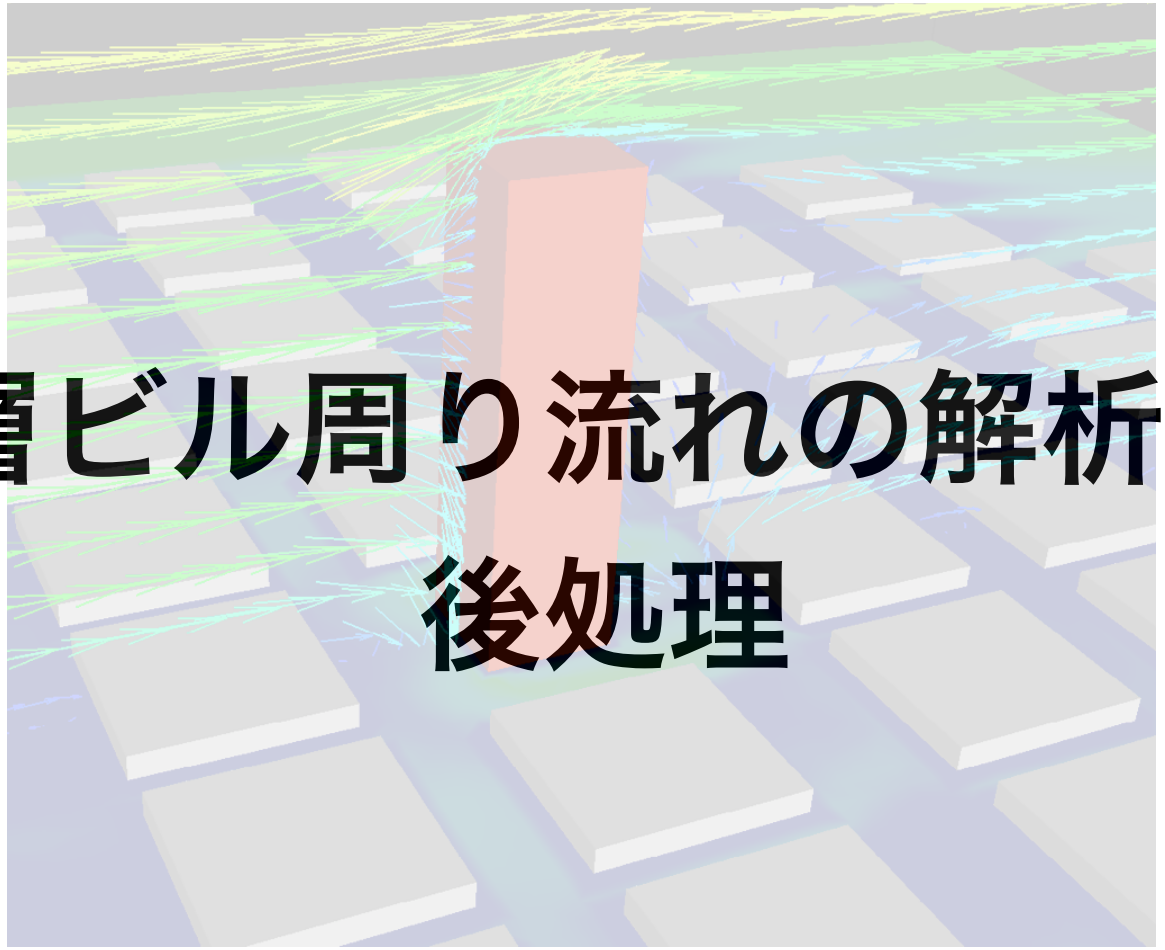
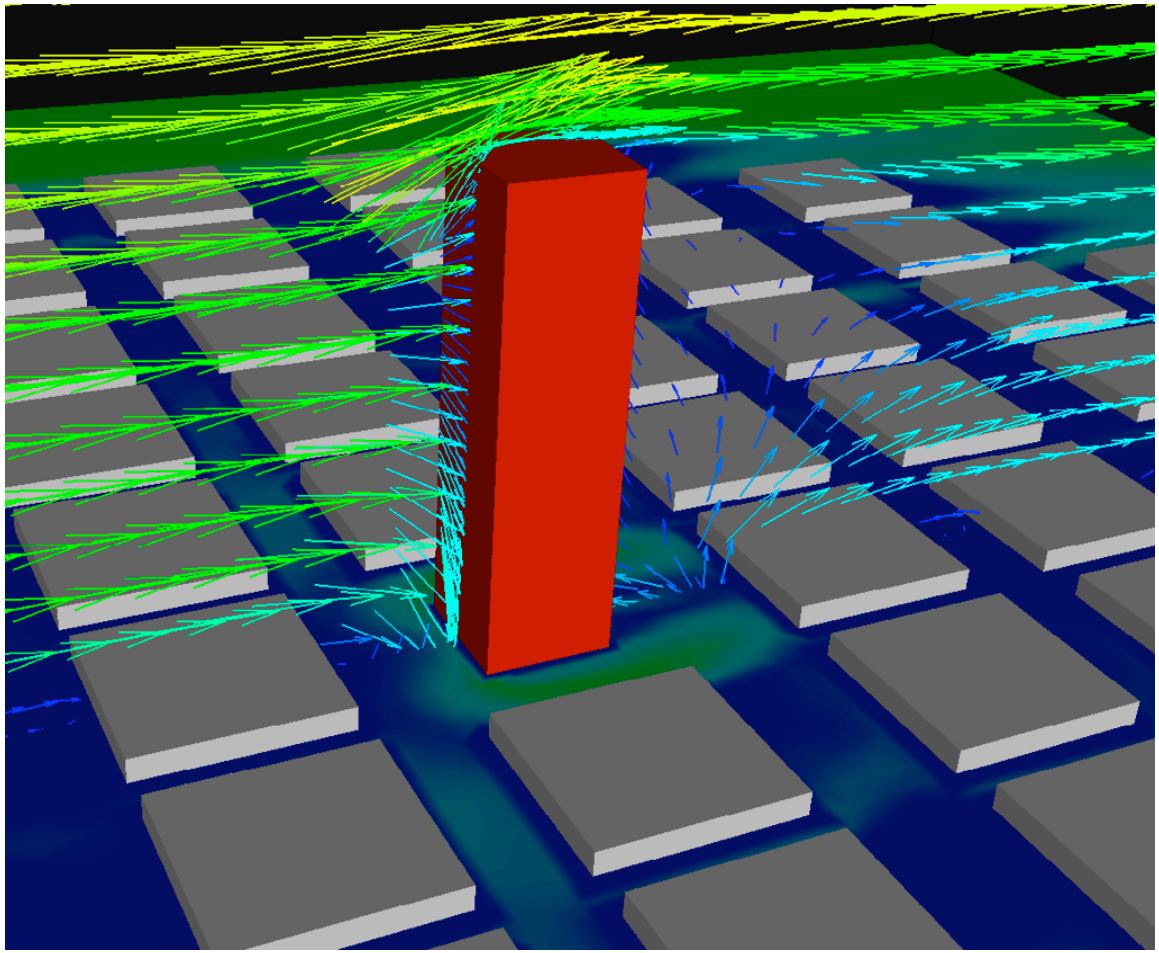
第1回OpenFOAM勉強会(2009年5月13日)



高層ビル周り流れの解析結果
後処理

今野 雅　（東京大学）

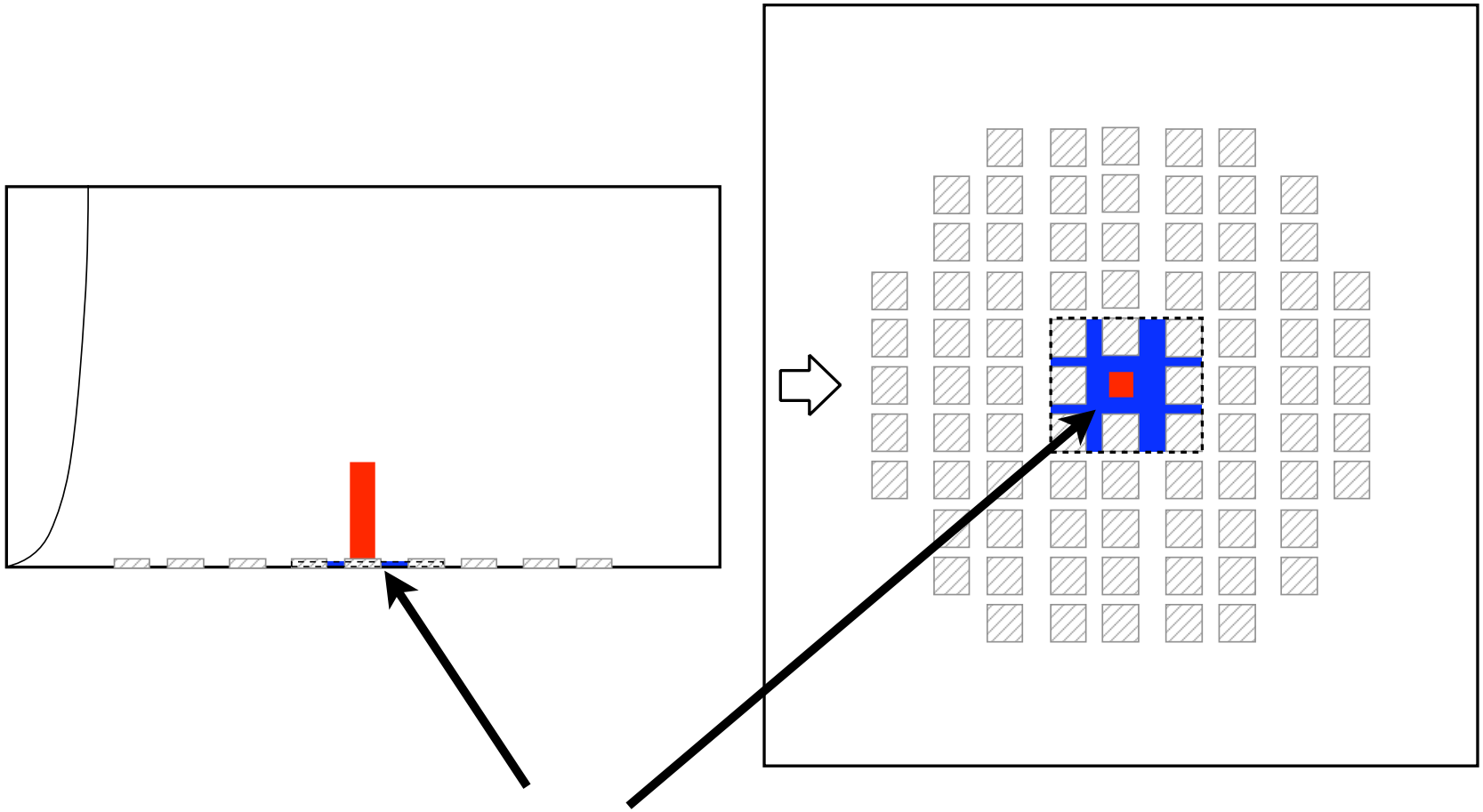
課題



街区内に建つ高層建物モデルの解析

(日本建築学会・流体数値計算による風環境評価ガイドライン作成WG)

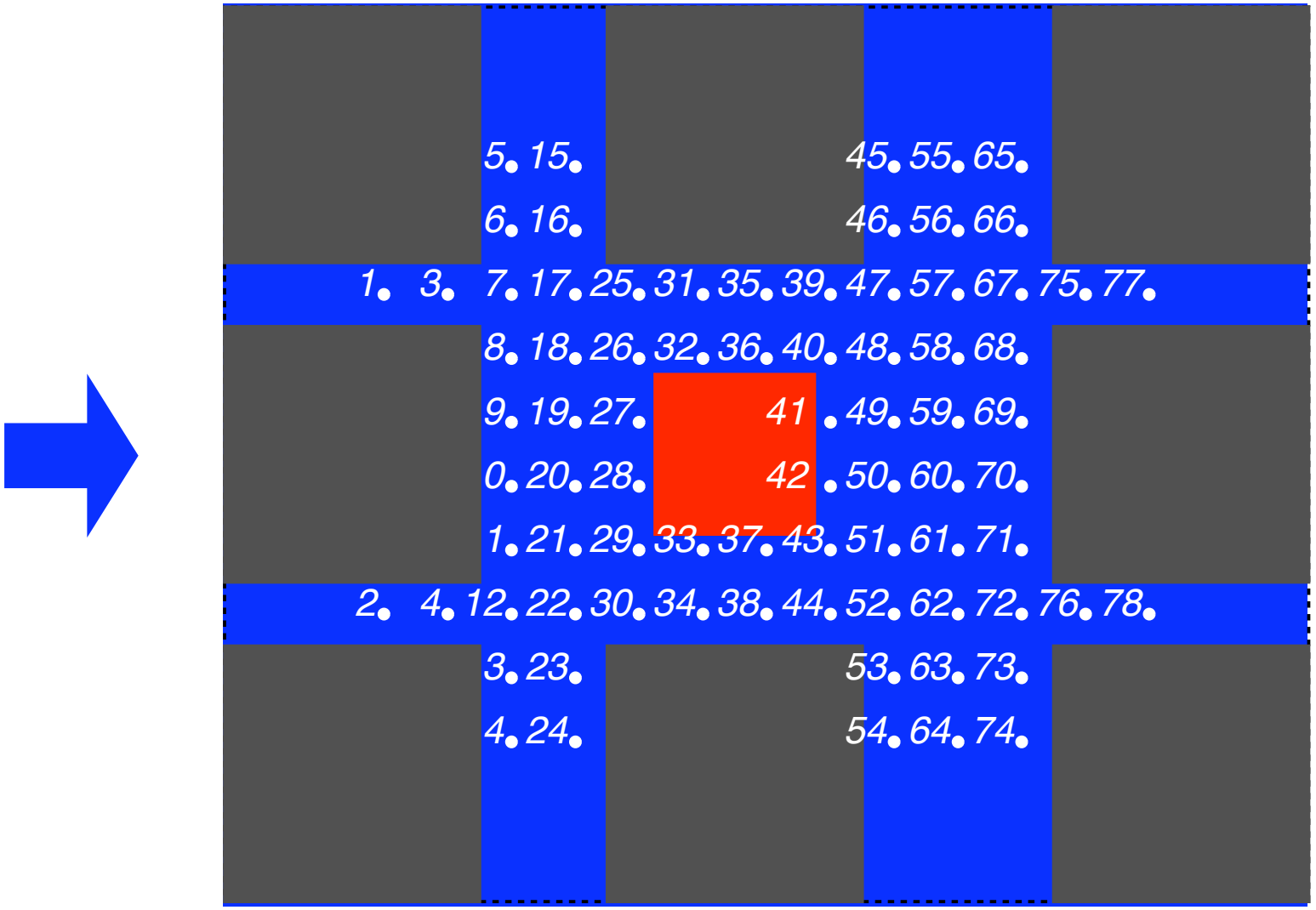
実験値との比較対象



高層建物による風害予測

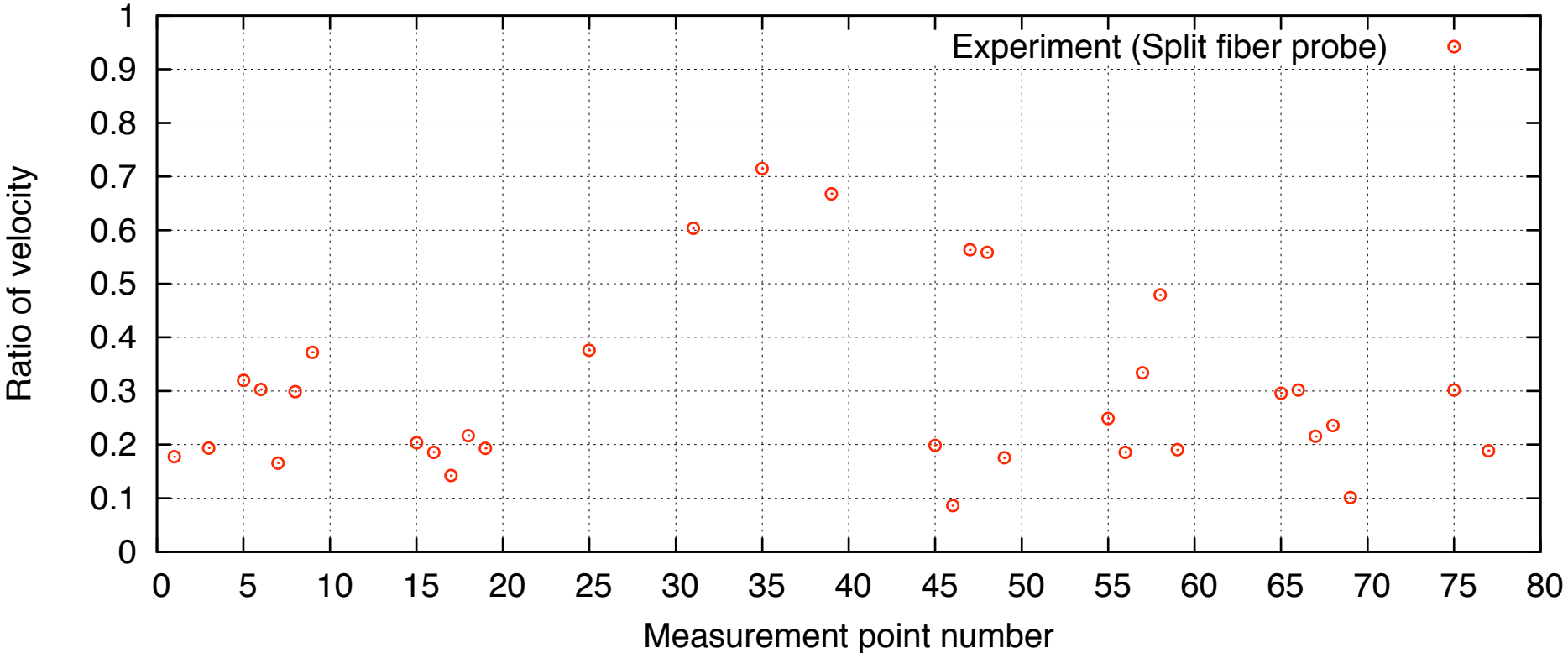
高層建物周辺の高さ2mの風速

風洞実験での風速測定点

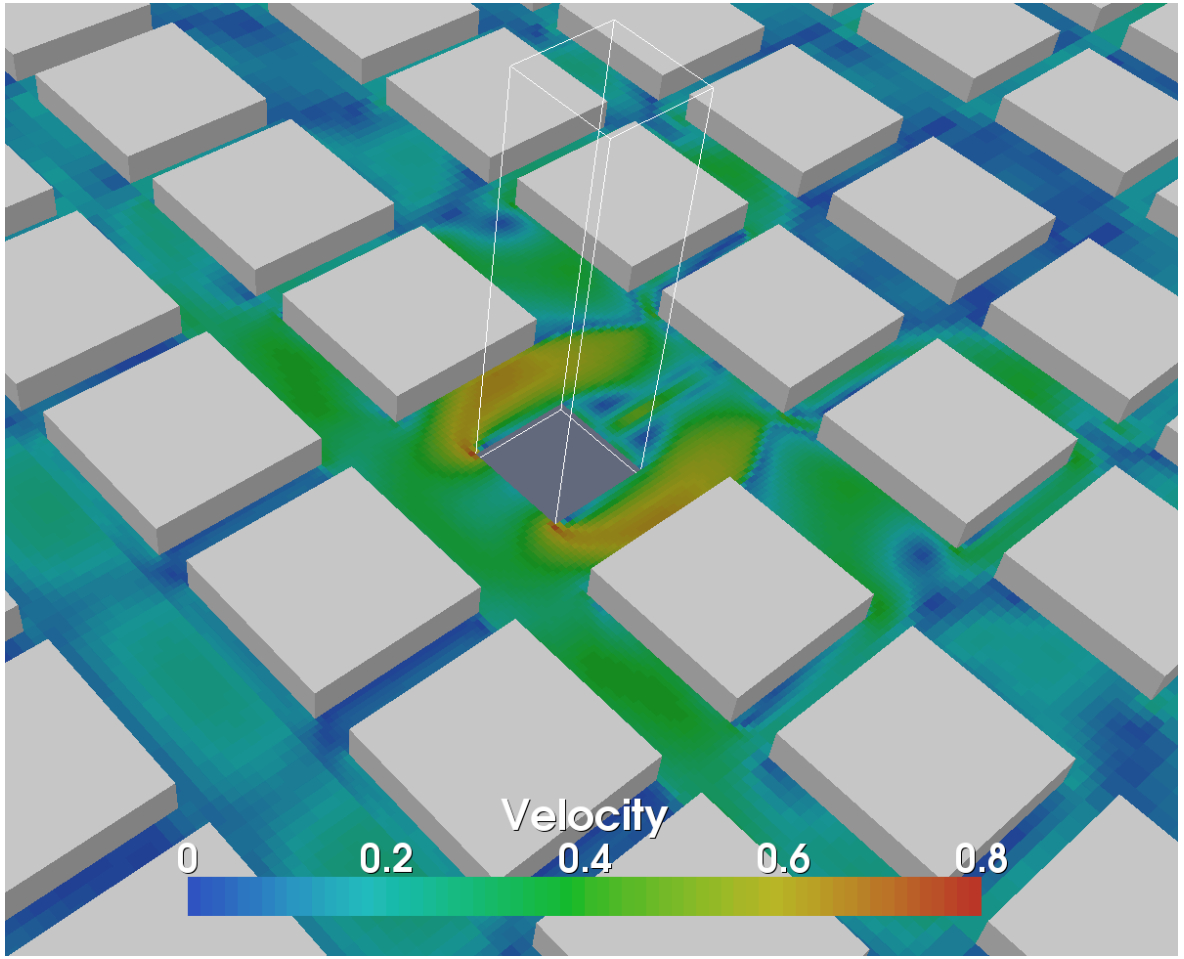


高さ5mm(実高さ2m相当)

風洞実験での風速測定結果



CFDでの風速解析結果



実験測定点での風速をサンプリングして、
実験結果と定量的に比較したい!

解析結果のサンプリング

- OpenFOAM標準のsampleユーティリティを使えば良い!
- ただし、1.4系と1.5系ではsampleの仕様が異なるので注意!
- 今回は、1.5系のsampleの使い方の説明を行う

sampleの設定

system/sampleDict で設定する。

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;
    class           dictionary;
    location        system;
    object          sampleDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Set output format : choice of
```

設定ファイルsampleDict

setFormat： 集合サンプリング出力形式

```
// Set output format : choice of
//        xmgr
//        jplot
//        gnuplot
//        raw
setFormat raw;
```

通常はASCII生データ形式のrawで良い

設定ファイルsampleDict

surfaceFormat： 面サンプリング出力形式

```
// Surface output format. Choice of
//      null            : suppress output
//      foamFile        : separate points, faces and values file
//      dx              : DX scalar or vector format
//      vtk             : VTK ascii format
//      raw             : x y z value format for use with e.g.
gnuplot 'splot'.
//      stl             : ascii stl. Does not contain values!
surfaceFormat foamFile;
```

foamFile形式だと断面のポリンゴンの幾情報が詳しく得られるので、プロットツールで複雑な図を書くのに有利

設定ファイルsampleDict

surfaceFormat： 面サンプリング出力形式

```
// Surface output format. Choice of
//      null            : suppress output
//      foamFile        : separate points, faces and values file
//      dx              : DX scalar or vector format
//      vtk             : VTK ascii format
//      raw             : x y z value format for use with e.g.
gnuplot 'splot'.
//      stl             : ascii stl. Does not contain values!
surfaceFormat foamFile;
```

ただし、gnuplotなどで簡易にコンタ図を描きたい場合には、raw形式が便利

設定ファイルsampleDict

interpolationScheme： 補間方法

```
// interpolationScheme. choice of
// cell           : use cell-centre value only; constant over
cells (default)
// cellPoint      : use cell-centre and vertex values
// cellPointFace : use cell-centre, vertex and face values.

interpolationScheme cellPointFace;
```

通常は、格子中心、節点、界面の値を併用して補間する cellPointFaceのほうが、精度が良いと思われる。

設定ファイルsampleDict

interpolationScheme： 補間方法

ただし、snappyHexMeshで生成されるようなsplit faceを持つ格子では、cellPointFaceだと、断面での補間結果が滑らかにならない場合あるので注意する。

設定ファイルsampleDict

fields：サンプルするフィールド

```
// Fields to sample.
fields
(
    U
    magU
    k
);
```

設定ファイルsampleDict

sets：集合サンプリングの定義

```
sets
(
    measuringPoints
    {
        type cloud;

        axis xyz;

        points
        ((-0.55  0.25 0.02));
    }
);
```

←副辞書名

←サンプルタイプ

←座標の出力形式

←以下具体的な指定

sets：集合サンプリング

sampleによる集合サンプリングの結果は、以下のファイルに書き込まれる。

```
sets/時刻/副辞書名_フィールド名.拡張子
```

例：

```
sets/472/measuringPoints_U.xy
sets/472/measuringPoints_magU_k.xy
```

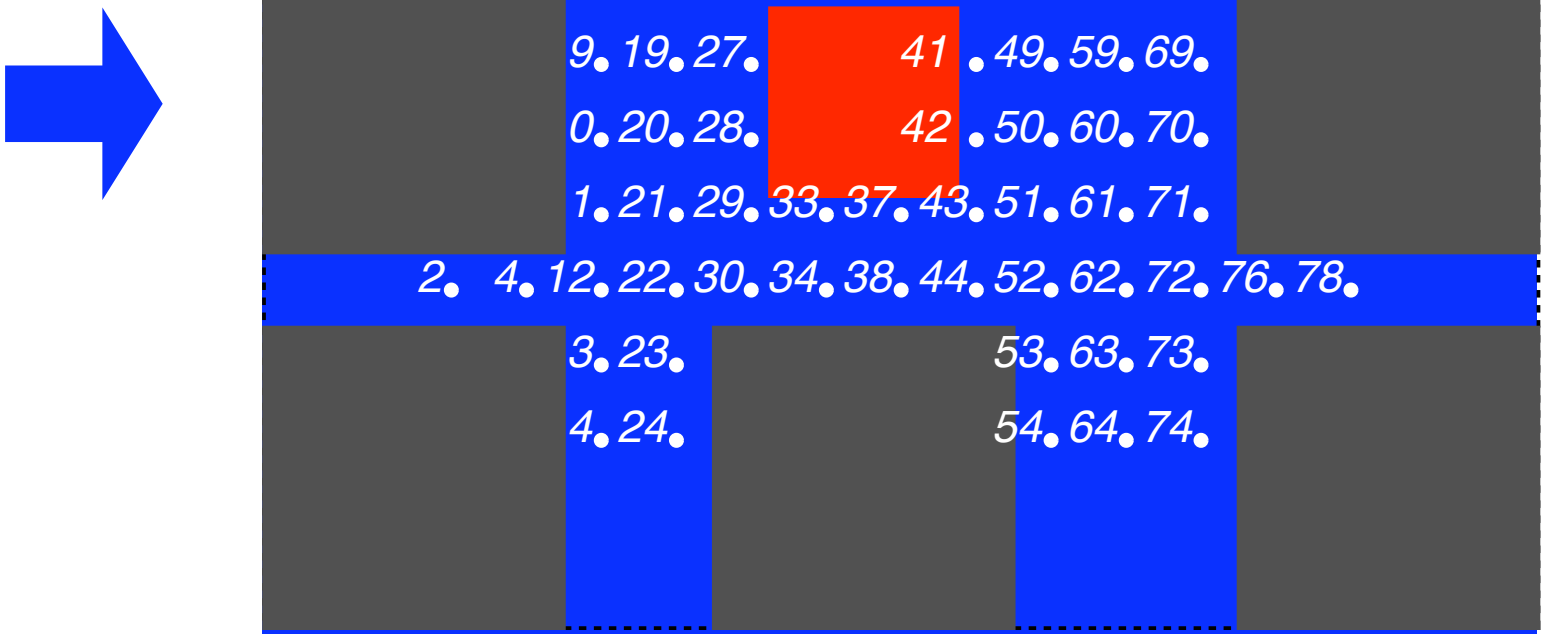
- 拡張子はsetFormatによって異なる
- フィールドの型毎に別々のファイル

sets：集合サンプリング

サンプルタイプ (type):

- uniform: 指定した線上に一様分布
- face: 指定した線と格子表面の交点
- midPoint: 格子表面の交点の中点
- midPointFace: faceとmidPoint
- curve: 曲線に沿って指定された点
- cloud: 指定された点群

実験での風速測定点



高さ5mm(実高さ2m相当)

sets：集合サンプリング

サンプルタイプ (type)：

- uniform: 指定した線上に一様分布
- face: 指定した線と格子表面の交点
- midPoint: 格子表面の交点の中点
- midPointFace: faceとmidPoint
- curve: 曲線に沿って指定された点
- cloud: 指定された点群

sets：集合サンプリング

```
sets
(
    measuringPoints
    {
        type cloud;
        axis xyz;
        points
        (
        (-0.55  0.25 0.02) //1
        (-0.55 -0.25 0.02) //2
        (-0.45  0.25 0.02) //3
        (-0.45 -0.25 0.02) //4
        (-0.35  0.45 0.02) //5
                :
```

←測定点のリスト

サンプリングの実践

端末で赤字のように打ってみましょう！

```
run

cd highRiseBuildingInCityBlocks/

cd 0deg/

more system/sampleDict
```

hの後にTabキーで補間

設定の確認

(more: スペースで次、bで前、qで終了)

サンプリングの実践

sampleを動かして、出力を確認

```
sample -latestTime
```

-latestTime：最終時点のデータをサンプリング

```
ls sets/
```

カーソルキー↑

```
ls sets/472/
```

カーソルキー↑で履歴を出し赤色の部分を追加・修正

カーソルキー↑

```
more sets/472/measuringPoints_magU_k.xy
```

長い名前はTabキーで補間

サンプリングの実践

スカラー型フィールドの場合

sets/472/measuringPoints_magU_k.xy：

X	Y	Z	magU	k
-0.55	0.25	0.02	0.077014039	0.0043491656
-0.55	-0.25	0.02	0.077249846	0.0044030344
-0.45	0.25	0.02	0.068113905	0.0042166839
-0.45	-0.25	0.02	0.068826129	0.004291114

サンプリング
点の座標
axis: xyzの場合

サンプリング
値(スカラー)

サンプリングの実践

速度ベクトルUのサンプリング結果を確認

カーソルキー↑

```
more sets/472/measuringPoints_U.xy
```

長い名前はTabキーで補間

サンプリングの実践

ベクトル型フィールドの場合

sets/472/measuringPoints_U.xy：

X	Y	Z	Ux	Uy	Uz
-0.55	0.25	0.02	0.076090407	-0.009761804	-0.0010983665
-0.55	-0.25	0.02	0.076308817	0.0098176701	-0.0012451114
-0.45	0.25	0.02	0.067512361	-0.0041875344	0.0036109745
-0.45	-0.25	0.02	0.068236725	0.0040790761	0.0034883398

サンプリング
点の座標
axis: xyzの場合

サンプリング
値(ベクトル値)

プロットの実践

汎用プロットツールgnuplotでプロット

```
gnuplot

!ls sets/

カーソルキー↑
!ls sets/472

plot “sets/472/measuringPoints_magU_k.xy”
using 4
```

gnuplot内でのコマンドの実行は先頭に!を付ける

附属のgnuplotはファイル名の補間が効かない

プロットの実践

汎用プロットツールgnuplotでプロット

カーソルキー↑で履歴で出し、赤色の部分を書き足す

カーソルキー↑

```
plot “sets/472/measuringPoints_magU_k.xy”
using 4 with lp
```

with lp で線+マークでデータを描く

カーソルキー↑

```
plot “sets/472/measuringPoints_magU_k.xy”
using 4 with lp title “CFD”
```

凡例 を付ける

プロットの実践

gnuplotで実験値と比較

```
!ls exp/
カーソルキー↑

!more exp/split.txt        風速が何カラム目か確認


カーソルキー↑を何回か押して、黒字の文を出す

plot “sets/472/measuringPoints_magU_k.xy”
using 4 with lp title “CFD”,"exp/split.txt"
using 2 title “Exp.”
```

プロットの実践

横軸、縦軸の説明を入れましょう

```
set xlabel “Point no.”

カーソルキー↑

set ylabel “Velocity ratio”

replot
```

同じplotコマンドで書き直す場合
replotで良い

プロットの実践

図をEPS形式でファイルに保存しましょう

```
set terminal postscript eps

set output “test.eps”

replot

!ls

!gv test.eps     ←EPSファイルを確認
```

プロットの実践

設定をファイルに保存し、終了です

```
save “test.gp”

!ls

quit          ←Ctrl-d でも終了します
```

ちなみに、再度プロットするには、

```
gnuplot

load “test.gp”
```

プロットの実践

チュートリアルケースのgnuplot
の図をまとめてプロット&ビュー

```
make plotview
```

設定ファイルsampleDict

surfaces：面サンプリングの定義

```
surfaces
(
    x0          ←副辞書名
    {
        type            plane;      ←面タイプ

        basePoint       (1e-4 0 0);

        normalVector    (1 0 0);

        triangulate     false;  ←三角形分割の有無

        interpolate     false;  ←補間の有無

    }
);
```

surfaces：面サンプリング

setFormatがraw の場合、sampleによる面サンプリングの結果は、以下のファイルに書き込まれる。

surfaces/時刻/副辞書名/

faces ←面の節点番号
points ←節点の座標

以下サンプル値

scalarField/スカラー型フィールド名

vectorField/ベクトル型フィールド名

:

surfaces：面サンプリング

面タイプ (type):

- plane: 指定した平面上でサンプル

```
type            plane;
basePoint       (1e-4 0 0); ←平面上の基準点
normalVector    (1 0 0);    ←平面の法線ベクトル
```

- patch: パッチ上でサンプル

```
type            patch;
patchName       movingWall; ←パッチ名
```

サンプリングの実践

端末で赤字のように打ってみましょう！

```
run

cd highRiseBuildingInCityBlocks/0deg/
```

長い名前はTabキーで補間

```
more system/sampleDict
```

設定の確認

(more: スペースで次、bで前、qで終了)

サンプリングの実践

既にsampleは動かしたので、出力を確認

```
cd surfaces/

ls

cd 472/

ls

cd z0_02/
```

長い名前はTabキーで補間

サンプリングの実践

既にsampleは動かしたので、出力を確認

```
more points          ←節点座標

more faces           ←面の節点番号リスト

cd vectorField/

more U               ←ベクトルのリスト

cd ../scalarField/

more magU            ←スカラーのリスト
```

面データのプロット

複雑な面サンプリングデータを、高機能なプロットツールに渡し、論文で掲載するのに耐えうるような高品質な図を、できるだけ自動的に作成したい。

面データのプロット

高機能なプロットツールとして、ここでは
GMTを用いる。

python+matplotlibも有望だが今回は省略

とは?

- ハワイ大学の地球海洋工学科の学生が1988年から開発を初め、オープンソースで公開されているプロットツール
- PostScript形式の高品質な図が出力できる
- 地球海洋工学の研究者が開発したことや、名称がGeneric Mapping Toolsの略であることから、開発当初は、海洋観測・シミュレーションデータのプロット用だったが、現在は汎用性が高い

の欠点

- GUIは全く無い
- データの変換、処理のコマンドやプロットのコマンドをシェル上で複数実行する必要がある
- 図を重ね合わせることで複雑な図を仕上げるという、UNIX的ではあるが初心者には難解な構造
- コマンドのオプンショが多くて複雑
- GMTを実用的に使うには、個々の図に対して、シェルスクリプトを書く必要があった

とsample

- sampleの面サンプリング出力をGMTでプロットするには、GMT用入力データへの変換が必要
- 変換にはruby等のスクリプトが適当
- データ変換と、GMTコマンドの実行をrubyスクリプトで一括して行なえば良いのでないか?
- また、メッシュやベクトル、コンター図といった良く描く図は、簡単なオプション指定で自動的にプロットできるようにしたい!

rubyによるGMTのラッパーgmtFoam作成

gmtFoamとは?

- sampleユーティリティーのサンプリング出力を加工してGMTに渡し、メッシュ、ベクトル、コンター等の図をプロットするrubyスクリプト
- まだオプションの方針が固まっていないのと、オプションのヘルプも出ないので、公開はしていない
- ある程度整備ができれば、OFWikiJa等で公開予定

gmtFoamでプロットの実践

使用法: gmtFoam [オプション] 副辞書名

主なオプション:

- -a 引数 : 図の1軸2軸を指定 (例: -a yz)
- -b : 図に枠を付ける
- -m : メッシュを描く
- -v : ベクトル図を描く

gmtFoamでプロットの実践

使用法: gmtFoam [オプション] 副辞書名

主なオプション:

- -c : コンター図を描く
- -S 引数 : コンター図を描くフィールド
- -p 引数 : コンターラインの間隔
- -A 引数 : コンターラベルの間隔等

gmtFoamでプロットの実践

端末で赤字のように打ってみましょう！

```
run

cd highRiseBuildingInCityBlocks/0deg/

ls surfaces/472/
```

gmtFoamでプロットの実践

メッシュ図を作成してみましょう!

```
gmtFoam -m -a xy z0_02

gv -watch gmtFoam.eps &
```

&を付けるとバック・グラウンド・ジョブとなり、
他のコマンドを続けて実行できる。

カーソルキー↑ x2回

```
gmtFoam -m -a xz y0
```

gmtFoamでプロットの実践

ベクトル図を作成してみましょう！

カーソルキー↑

```
gmtFoam -v -a xz y0
```

gmtFoamでプロットの実践

速度のコンター図を作成してみましょう！

```
gmtFoam -c -S magU -a xz y0
```

カーソルキー↑

```
gmtFoam -c -S magU -a xz y0 -p 0.2
```

カーソルキー↑　　コンター間隔を0.2に

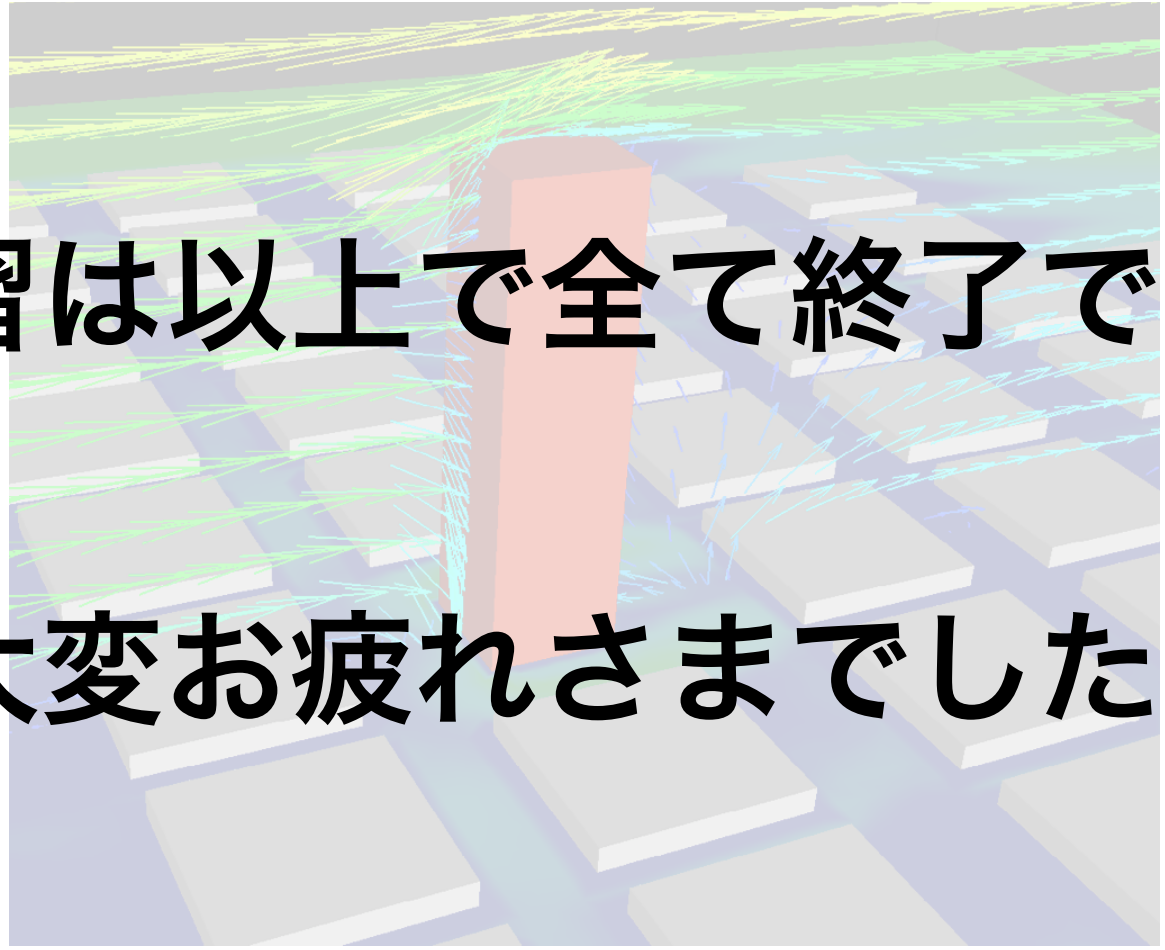
```
gmtFoam -c -S magU -a xz y0 -p 0.2 -A 0.2
```

数字を間隔0.2毎に入れる

gmtFoamでプロットの実践

チュートリアルケースのgmtFoamの図をまとめてプロット&ビュー

```
make figview
```



講習は以上で全て終了です!

大変お疲れさまでした!