配管流路の多目的最適化
OpenFOAM + OpenMDAO

第28回オープンCAE勉強会@ 関西

2014. 03.08

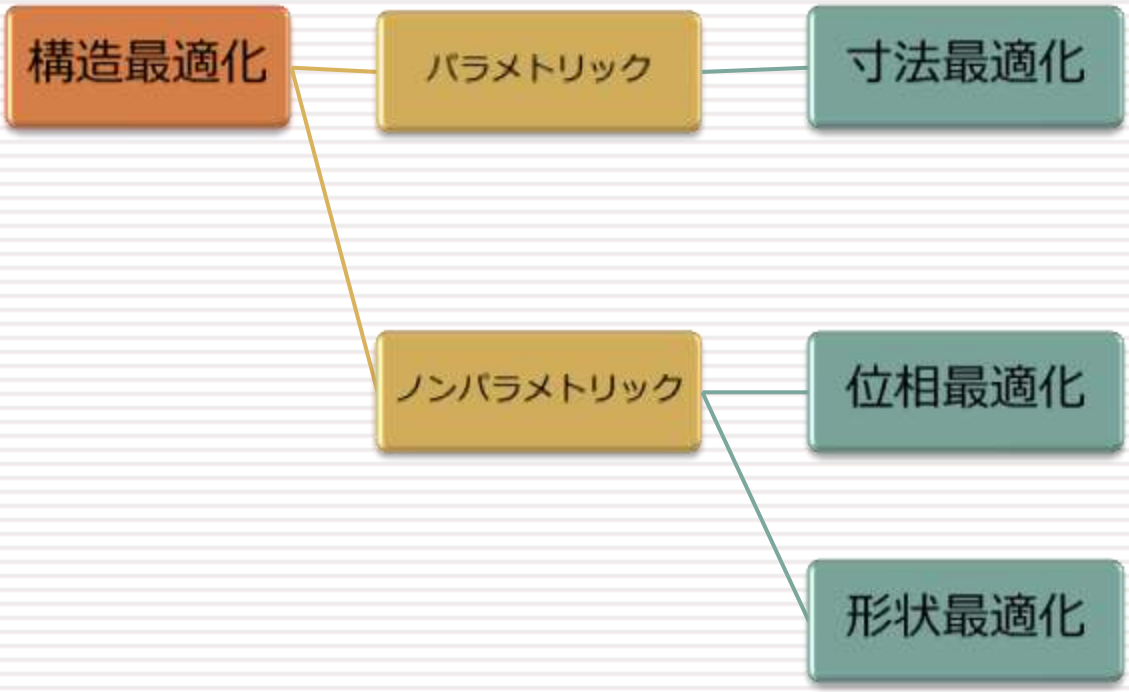
片山　達也

目次

- 多目的最適化
- 曲げ配管の流れ最適化
- 最適化問題
- 解析の自動化
- 応答曲面の作成(Kriging モデル)
- 多目的最適化(NSGA2)
- 自己組織化マップ(SOM)
- 付録

多目的最適化

- CAE/CFDにおける最適化

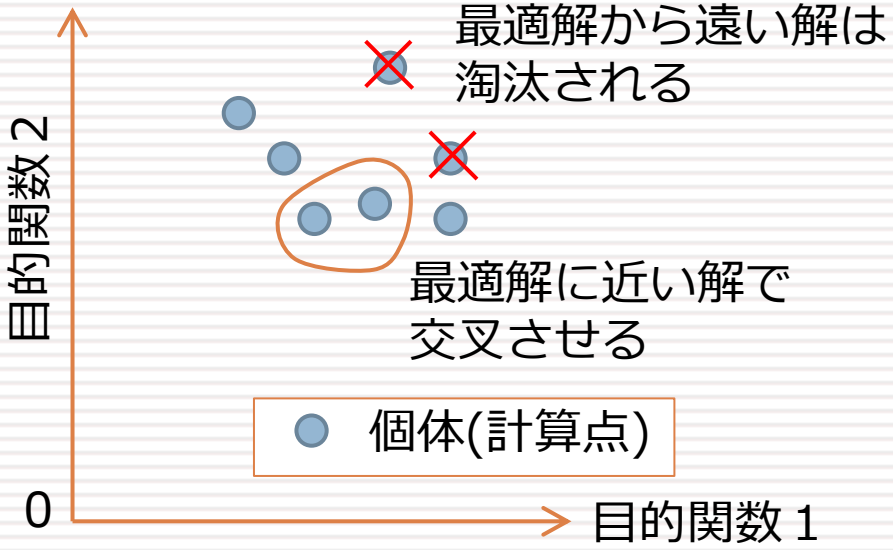


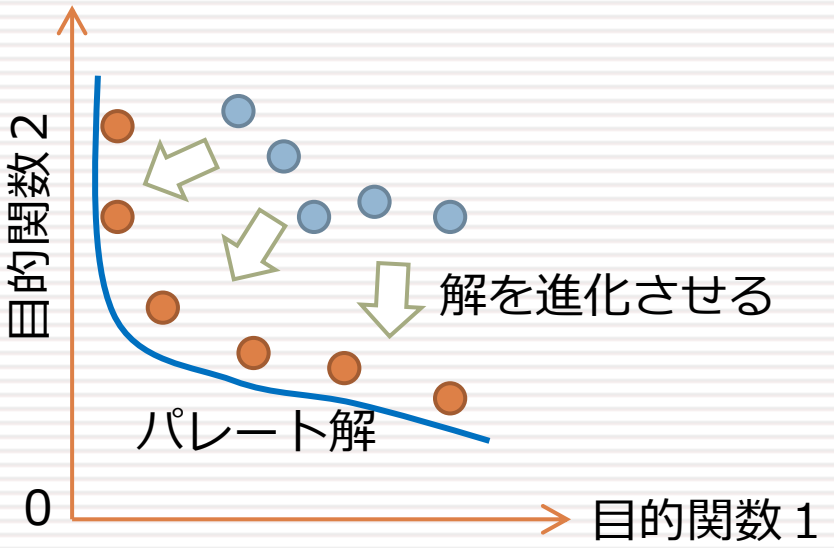
寸法などパラメトリックな値を設計変数とするため、工学的に利用しやすい？
多目的最適化も可能。
寸法のバラつきも考慮したロバスト最適化もある

形状のトポロジーや表面を設計変数とする。
随伴方程式を解くタイプの最適化。
単一の目的関数を最小とする形状の創出や滑らか形状を表面を計算できる

多目的最適化

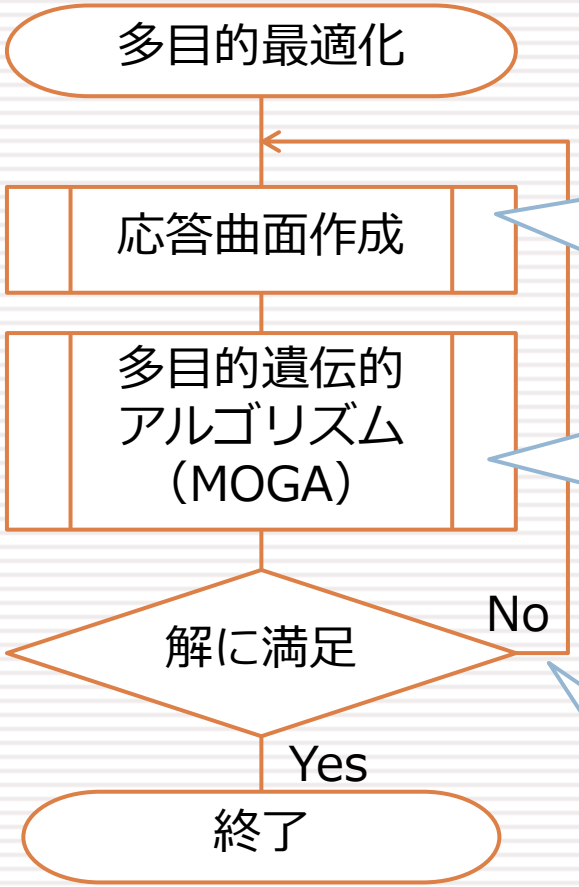
- パレート解と多目的遺伝的アルゴリズム
  - 複数の目的があればトレードオフの関係が生じうる
  - 淘汰、交叉、突然変異を繰り返し、よりよい個体を生存させる遺伝的アルゴリズムを用いてパレート解を求める。 ※CYBERNET解析講座　初めての最適化が分かりやすい





多目的最適化

- 多目的最適化の流れ



目的関数や制約関数の近似関数を作成する

$$f_1(\mathbf{x})\cdots f_i(\mathbf{x}),\ g_1(\mathbf{x})\cdots g_j(\mathbf{x})$$

設計変数**x**を与えると目的関数の近似値の計算が可能

遺伝的アルゴリズムの多目的版。
母集団内でパレート解以外を淘汰し、
パレート解同士の交配や突然変異により
新たなパレート解を生み出していく

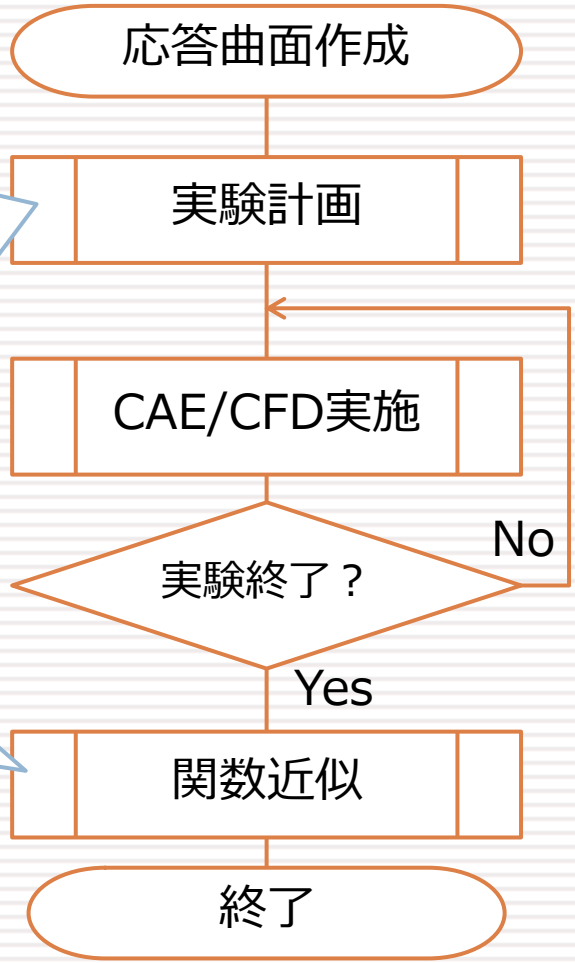
近似精度の低い応答曲面をループによる強化で
トータルの実験数を減らす
⇒逐次近似最適化アルゴリズム

多目的最適化

- 応答曲面の作成

近似関数を作るための実験計画。
近似モデルに適した実験計画がよい。
例えば、2次関数で近似したい場合中心複合化計画がよい。（たしか・・・）
逐次近似最適化を行う場合は、
近似モデルにKrigingモデルやRBFニューラルネットワーク、実験計画にラテン超方格を用いる（たぶん・・・）

KrigingモデルやRBFニューラルネットワークのような非線形モデルでは、オーバーフィッティングとならないように注意が必要。



曲げ配管の流れの最適化

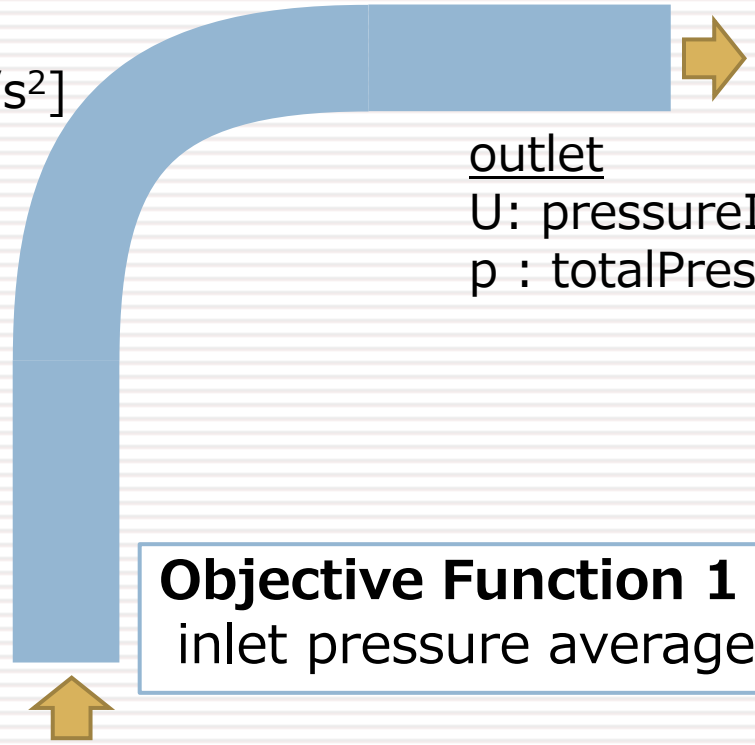
- 目的関数

目的関数１,2,3を最小化する

Objective Function 2
outlet velocity sum of squares

ソルバ　　：simpleFoam
動粘度　　：1.5e-05 [m²/s²]
乱流モデル：kOmegaSST

Objective Function 3
pipe Length



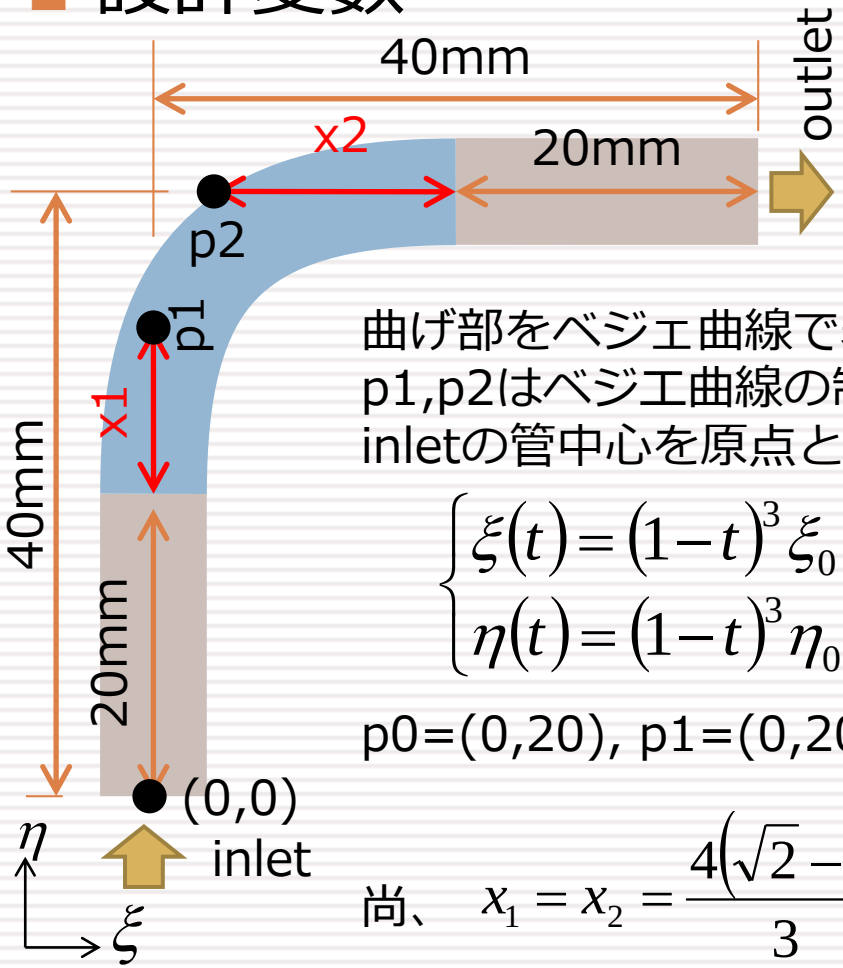
outlet
U: pressureInletOutletVelocity
p：totalPressure　p0=0

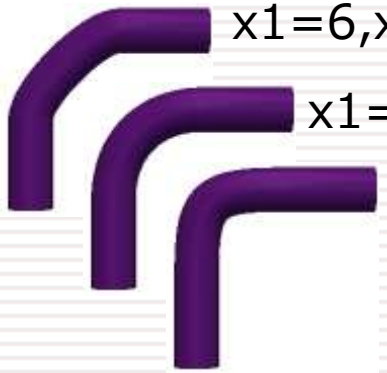
inlet
U:fixedValue (0 0 10)
p:zeroGradient

Objective Function 1
inlet pressure average

曲げ配管の流れの最適化

- 設計変数





曲げ部をベジェ曲線で表す。
p1,p2はベジエ曲線の制御点でp1,p2の位置x1,x2を設計変数とする
inletの管中心を原点としたとき曲げ部は以下のように表される

$$\begin{cases} \xi(t) = (1-t)^3\xi_0 + 3(1-t)^2 t\xi_1 + 3(1-t)t^2\xi_2 + t^3\xi_3 \\ \eta(t) = (1-t)^3\eta_0 + 3(1-t)^2 t\eta_1 + 3(1-t)t^2\eta_2 + t^3\eta_3 \end{cases}$$

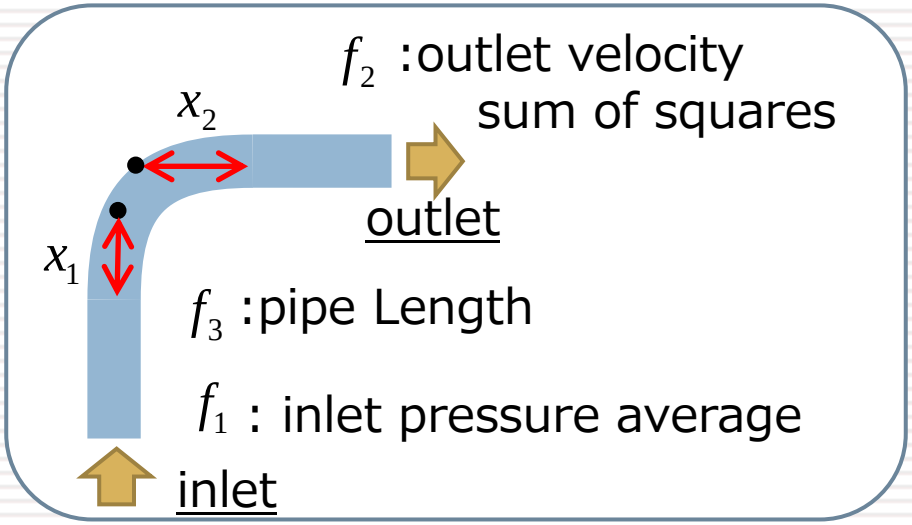
p0=(0,20), p1=(0,20+x1),p2=(20-x2,40),p3=(20,40)を代入

尚、 $x_1 = x_2 = \dfrac{4\left(\sqrt{2}-1\right)}{3}\cdot 20$ のとき最も円に近似される

最適化問題

- 多目的最適化問題

$$\min : f_1(x_1, x_2)$$
$$\min : f_2(x_1, x_2)$$
$$\min : f_3(x_1, x_2)$$
$$\text{subject to} : x_1 \in [6,18] \quad x_2 \in [6,18]$$
$$\text{find} : x_1, x_2$$



※ただし最適化には f_1, f_2, f_3 の近似似関数を用いる

解析の自動化

- Componentクラス
  - OpenMDAOでは計算部分をComponentクラスを継承して記述する必要がある
  - input-outputとなるクラス変数、計算を実行するクラスメソッドexecute()を定義する
  - execute内で、iotype=outのクラス変数に値を入力する
  - インスタンス生成後、iotype=inのクラス変数に値を入力し、クラスメソッドexecute()を実行すれば計算が行われ、iotype=outのクラス変数に値が入る

解析の自動化

- bendPipeクラス

```
38 class bendPipe(Component):
39     # set up interface to the framework
40     p1y = Float(0.0, iotype='in', desc='The variable of P1_y in bezie curve')
41     p2z = Float(0.0, iotype='in', desc='The variable of P2_z in bezie curve')
42
43     averagePressureHEAD = Float(iotype='out', desc='inlet_Pressure_Average')
44     sumSqVelocityOutlet = Float(iotype='out', desc='outlet_sumSq Velocity')
45     pipeLength = Float(iotype='out',desc="pipe Length from bezie Curve")
46
47     foamCase=SolutionDirectory("/opt/OpenFOAM/katayama-2.2.x/run/bendPipeMOGA/bendPipe")
48
49
50     def execute(self):
51         """CFD Analysis with OpenFOAM
52         """
53         print "¥ncalcate OpenFAOM :p1y=" + str(self.p1y) +",p2z="+str(self.p2z)
54         self.foamCase.clear()
55         #**pre-processing**
56         self.preprocessing()
57         #**main-processing**
58         self.mainprocessing()
59         #**post-processing
60         self.postprocessing()
```

設計変数(入力)

目的関数(出力)

executeメソッドに計算部分を書く

pre-main-post processingを
別のクラスメソッドで定義済み

解析の自動化

- クラスメソッド：preprocessing()
  1. blockMesh内曲げ部のedgeをspline定義
  2. ベジエ曲線からedgeのsplineを計算
  3. pyFoamを用いてblockMeshDictを編集　※1
  4. blockMeshの実行　※2

- クラスメソッド：　mainprocessing()
  1. simpleFoamの実行　※2

※1　PyFoam.RunDictionary.ParsedBlockMeshDict.ParsedBlockMeshDictを利用
※2　PyFoam.Execution.BasicRunner.BasicRunnerを利用

解析の自動化

- クラスメソッド： postprocessing()
  1. functionObjectを用いinlet Pressure Averageをログ出力。pyFoamを用いてログ解析 ※2
  2. 最終timeStepのUを読み込み(※3)、outlet velocity sum of squaresを計算
  3. pipe Lengthについてはpreprocessingにて取得

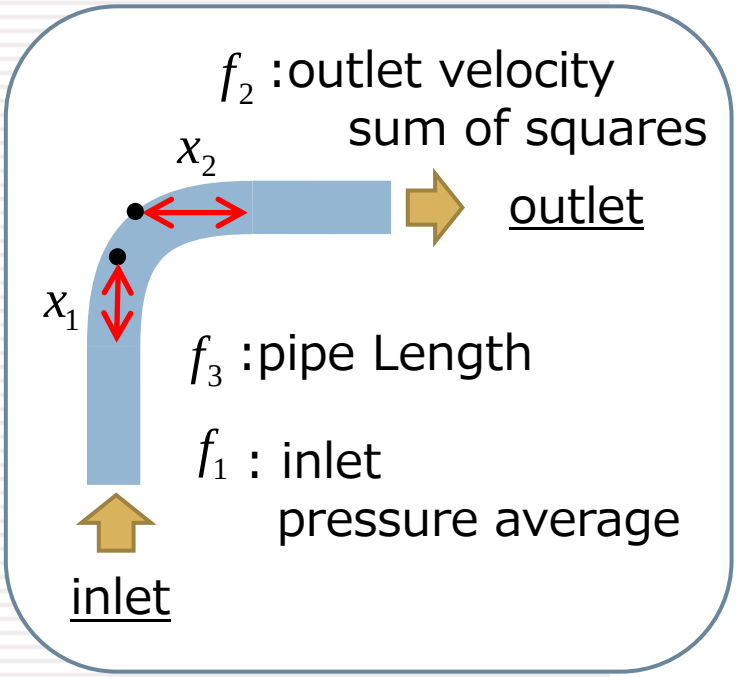
※1　PyFoam.RunDictionary.ParsedBlockMeshDict.ParsedBlockMeshDictを利用
※2　PyFoam.Execution.BasicRunner.BasicRunnerを利用

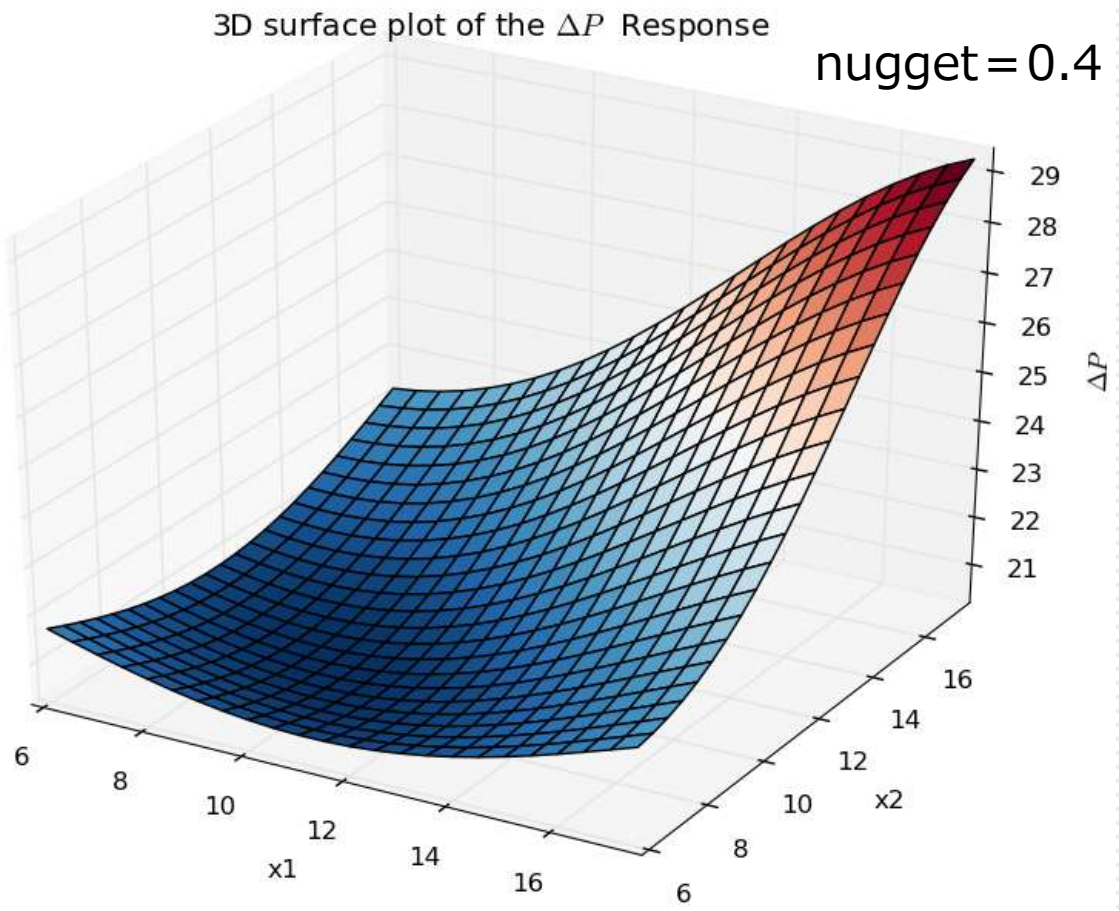
応答曲面の作成（Kriging モデル）

- 実験計画
  - ２変数について各５水準の計５^{2}回実験する
- メタモデル(近似関数)
  - 近似関数
    - Pressure Average の近似モデル：Kriging
      - nugget(平滑化パラメータ)の使用
    - outlet velocity sum of squaresの近似モデル：Kriging
      - nugget(平滑化パラメータ)の使用
    - pipe Lengthの近似モデル：Kriging
      - nugget(平滑化パラメータ)の使用
  - モデル
    - bendPipeインスタンス

応答曲面の作成（Krigingモデル）

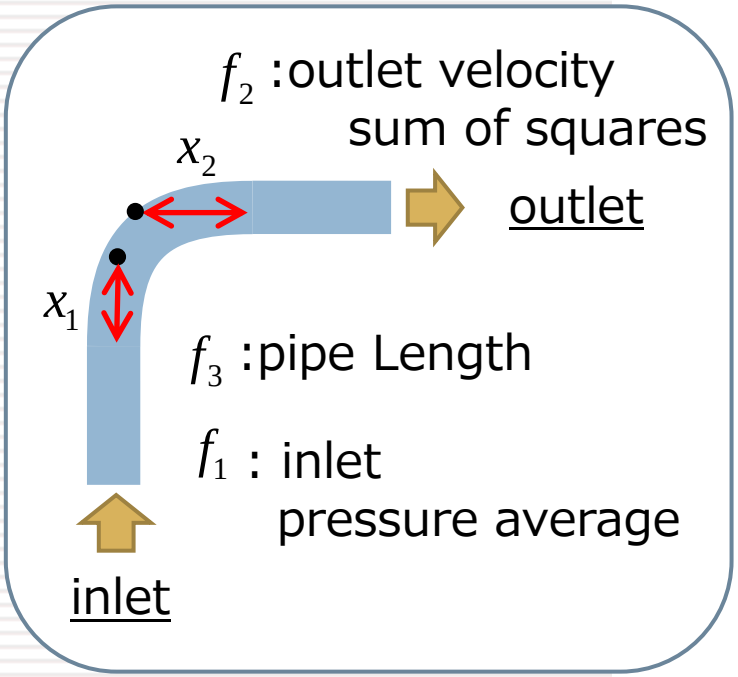
- inlet pressure average の応答曲面

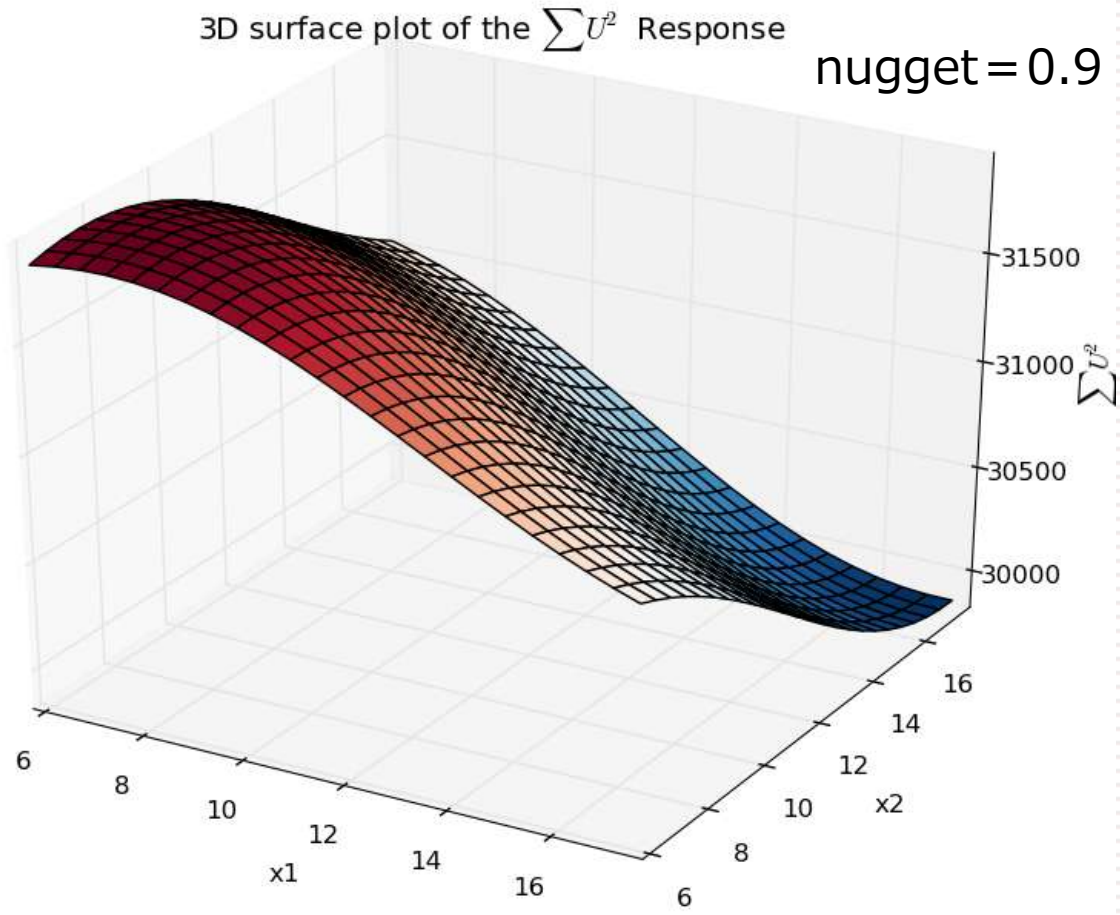




応答曲面の作成（Krigingモデル）

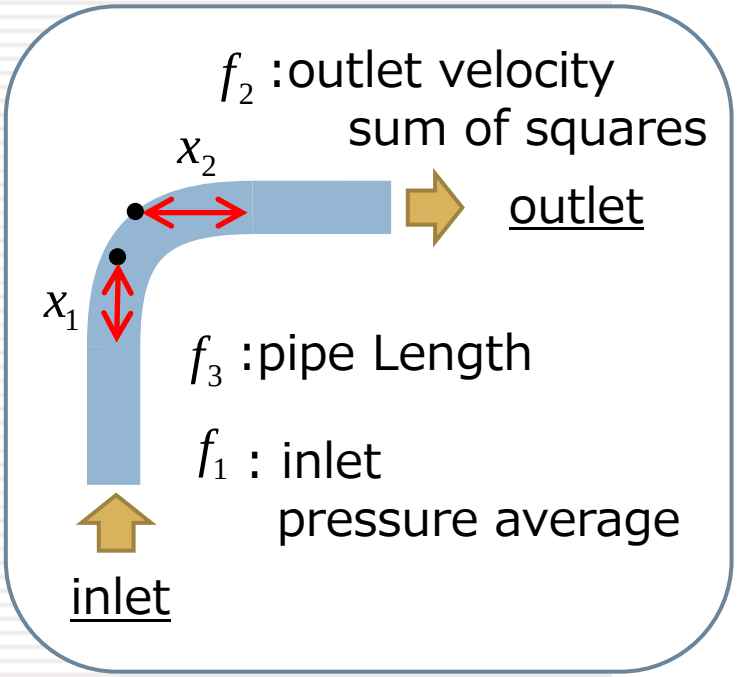
- outlet velocity sum of squares の応答曲面

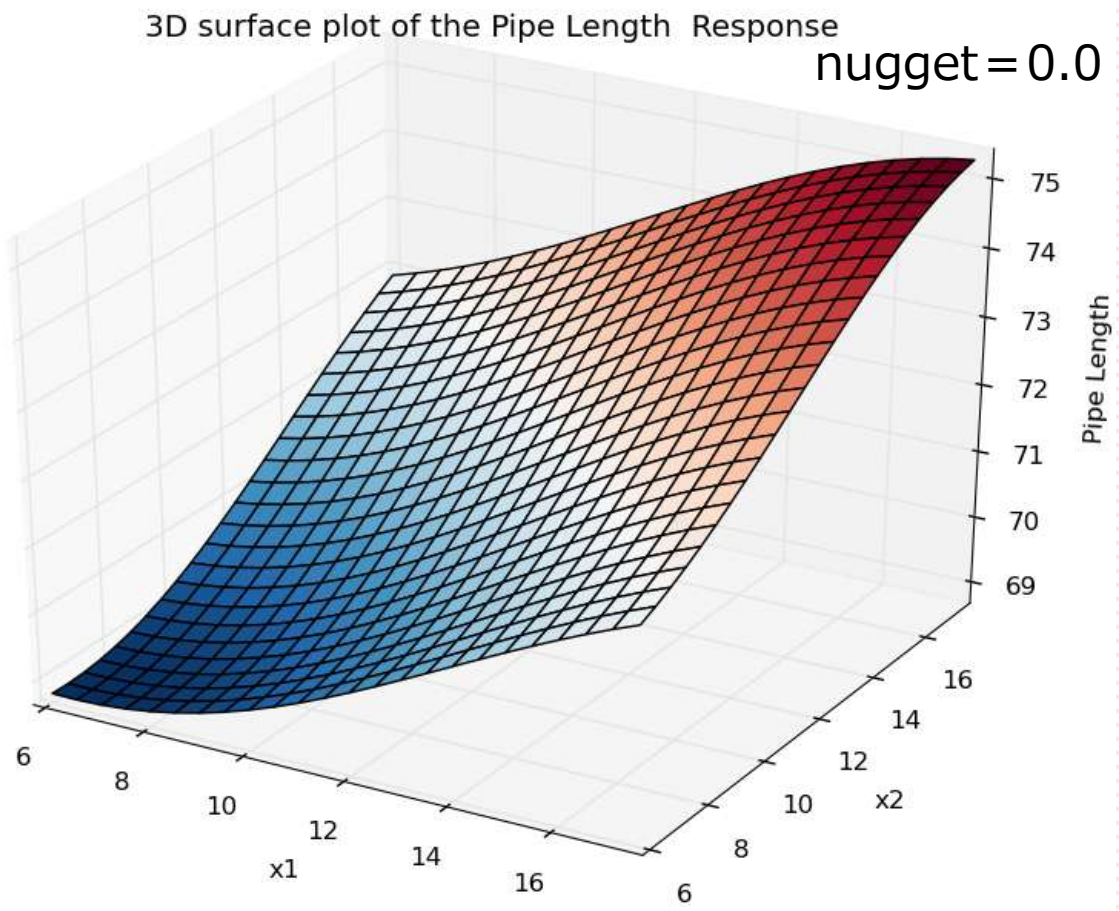




応答曲面の作成（Krigingモデル）

- pipe Length の応答曲面





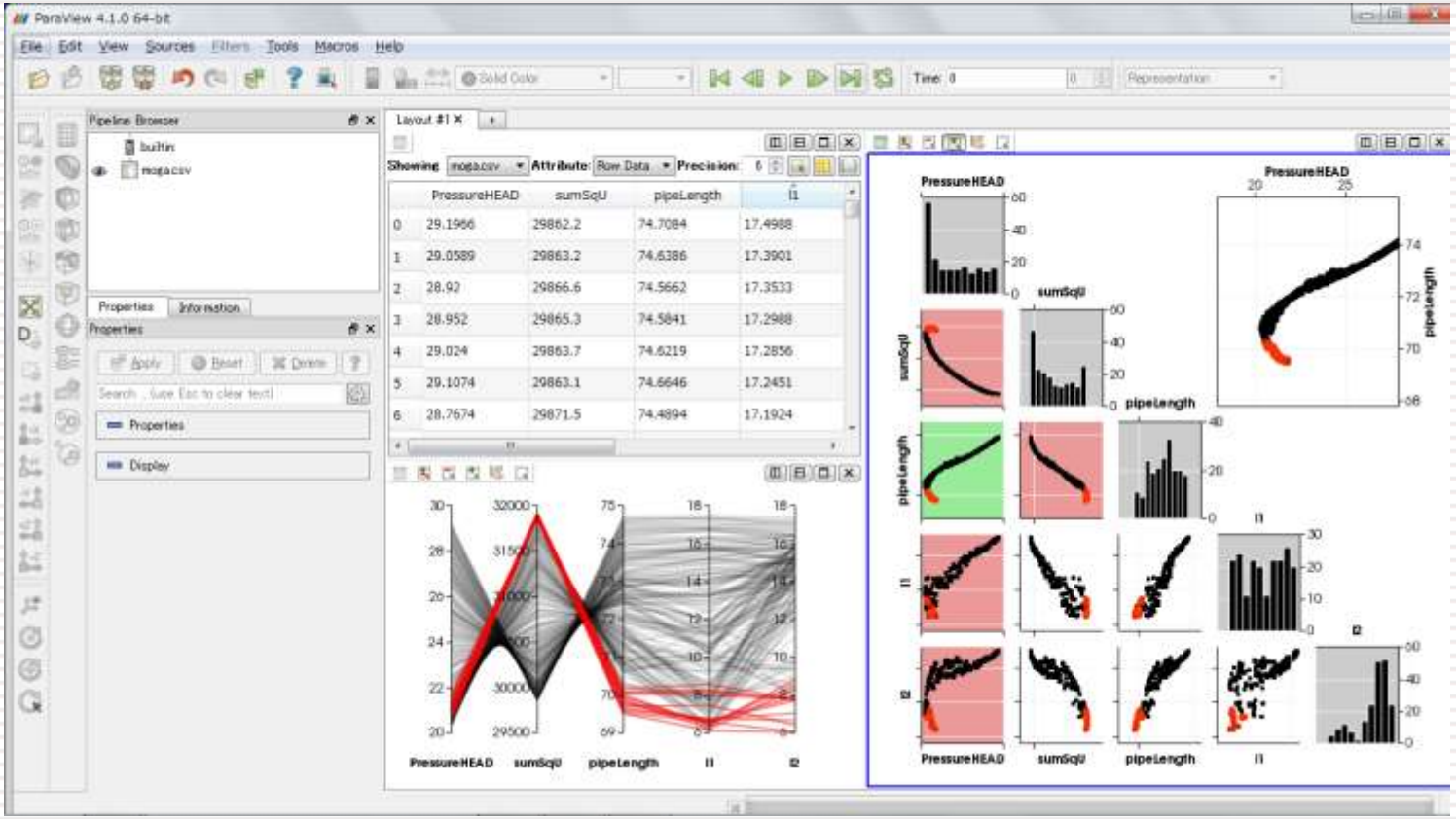
多目的最適化(NSGA2)

- NSGA2
  - 特徴
    - 高速優越ソート
    - シェアリングの代用
    - エリート主義
  - Optimizer Options
    - 母集団：２００
    - 世代数：１０
  - 詳細資料
    - 同志社大学 理工学部 インテリジェント情報工学科 知的システムデザイン研究室の資料

その他のオプションは
pyOpt NSGA2の
Optimizer Options参照

多目的最適化(NSGA2)

- パレート解の可視化(paraview-4.1)



今後

- OpenMDAOの展開
  - 構造寸法最適化(Salome-Meca)
  - ロバスト寸法最適化
  - ノンパラメトリック最適化
  - 自己組織化マップ
  - 逐次近似最適化
  - OpenMDAOをlibraryとして使う
  - 並列化

付録

- 使用ツール
  - OpenFOAM　(ver:2.2.x)
    - blockMeshによる形状変更
    - simpleFoamによる解析
  - pyFoam　(ver:0.6.1)
    - python libraryとして使用しCFD解析を自動化
  - OpenMDAO (ver:0.9.2)
    - 応答曲面作成(実験計画/近似モデル)
    - 多目的最適化(NSGA2 @pyOpt (ver:1.1.0))