# パラメータ自動最適化ツール Dakotaの紹介

OpenCAE学会員

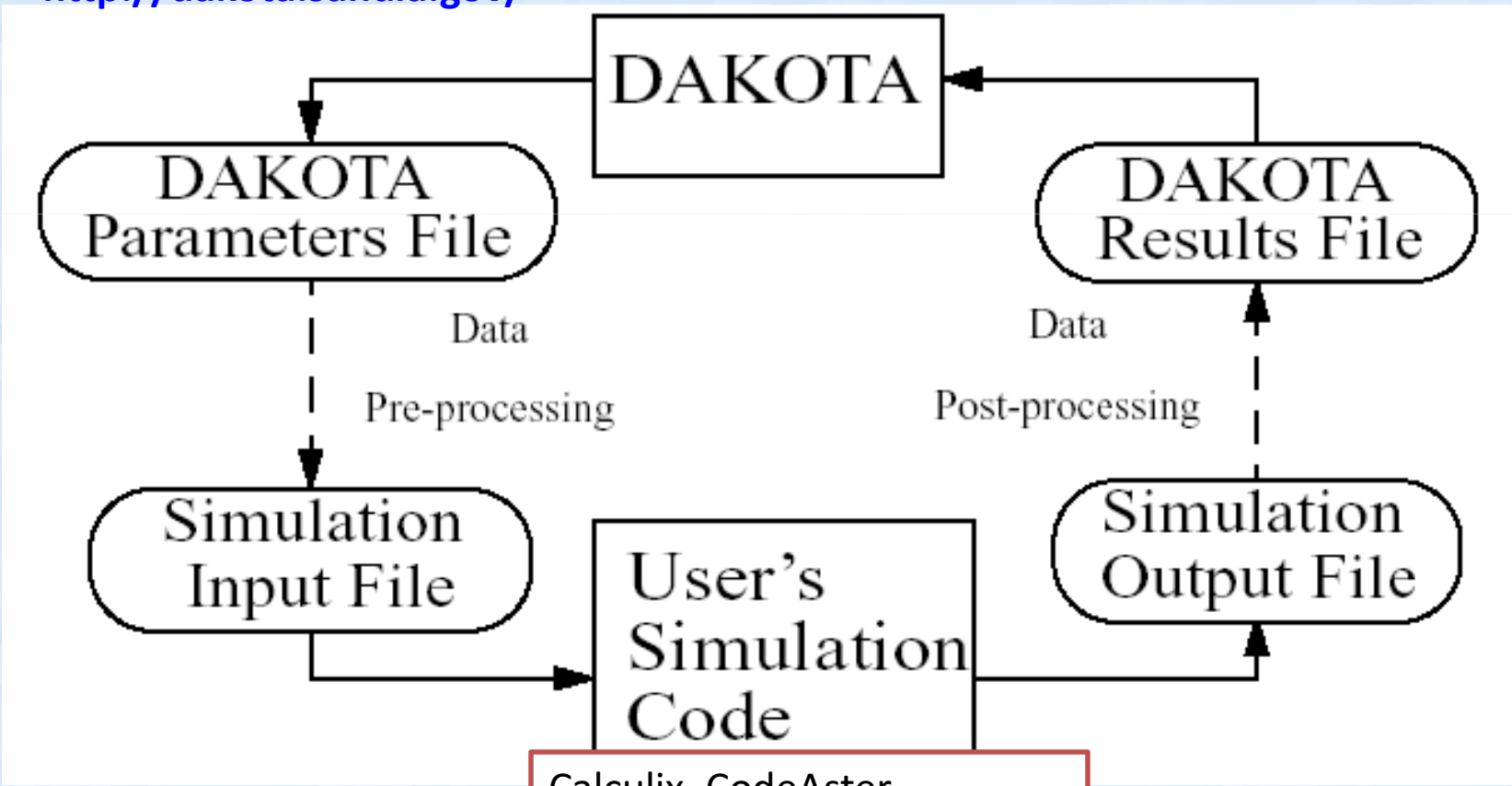
SH

# 本日の発表内容

- Dakotaとは？
- シミュレーション最適化手法の概要
- Dakotaインストール方法
  -Windows系(cygwinコンパイル済)
  -RedhatLinux系(binaryコンパイル済)
  -Dexcs2011系(コンパイル必須)
- Dakota 使用方法
  -標準例題
- Dakotaと各ツールとの連携
  -Excel連携-Calculix連携
- GUIツール:Jaguar について
- 参考資料(マニュアルなど)
- まとめ

# Dakotaとは①？

マニュアルのp.20に書かれているように自動的に解析に入力するパラメータを変更したパラメータSTUDYや目的となる指標（目的関数）を与えると自動的にそのような目的の値になるようなパラメータを見つけてくれるツールです。
一般的には総称して最適化ツールとか呼ばれています。
-ParaView と同じ　SandiaNational Laboで開発されている
**- http://dakota.sandia.gov/**




Calculix, CodeAster,
OpenFOAM,  etc

# Dakotaとは②？

- 同じような機能を持つ商用ソフトとしてはIsight, Optimus, modeFrontier, etc などがあります。

機能比較表は問題ありそうなので削除

# Dakotaとは③？

- この種のソフトウェアの特徴としてものによって違いはありますが、以下の機能のいくつかあるいは全ての機能を網羅しています。

-実験計画： Sampling 点の効率的選択
直交表， 多次元配置， 乱数，
数値実験計画(D-optimization)
-最適化計算： 数理計画法(古典)によるもの
ヒューリスティック手法(GA, EA, SA等)
-近似手法： 古典的モデル(多項式など)
確率論的モデル(Krigingなど)
-Simulationソフトとのインターフェース
-GUI： ツール間データフロー定義、結果可視化

商用ソフトはこの辺充実

# Dakotaとは④？

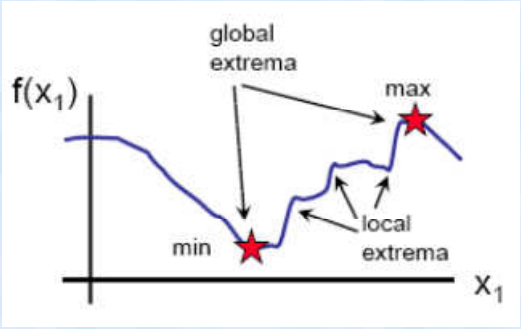
- DAKOTAは具体的には何に使えるのか？

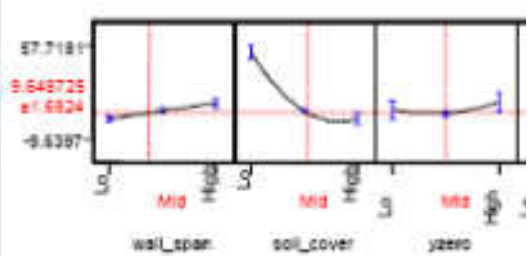
  -Optimization 設計最適化：
  最適寸法や最適パラメータ決定

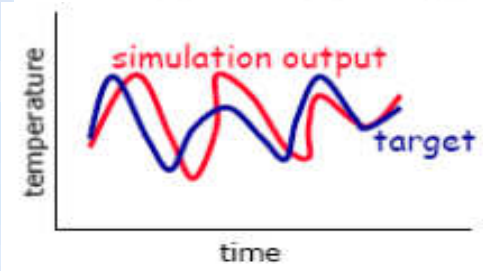
  -Sensitivity Analysis 感度解析：
  入力パラメータ変動に対する出力感度

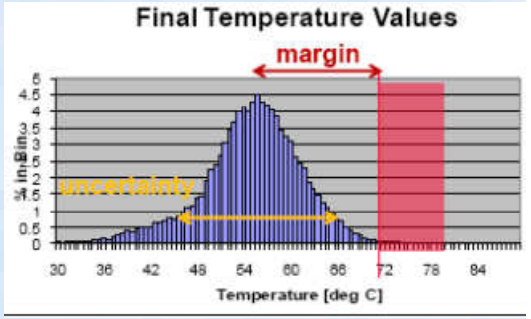
  -Calibration パラメータ同定：
  未知パラメータ決定（物性値同定など）

  -Uncertainty Quantification 信頼度：
  入力パラメータが変動した場合の応答変動の確率分布推定



# シミュレーション最適化手法の概要①

ー最適化問題あるいは問題の実用解の探索はあらゆる工学における基礎となっている。

ー応用数学、制御工学、情報工学、プログラミング、経営工学、機械工学などの分野ごとに個別扱われてきており、分野間の意志疎通は良好とは言えない。

ー構造設計などに関するシミュレーションにおける最適化手法も一般的な最適化手法を応用しているが、各種制約から独特の応用分野となっている。

# シミュレーション最適化手法の概要②
# 最適化手法全体の概論

ー最適化手法の例

制御工学：変分法、最大原理、動的計画法

応用数学：

線形計画法 LP（Simplex法，内点法）

非線形最適化法　NLP

-制約条件無し:最急降下法，Newton法,共役勾配法(CG法)

-制約条件有り:ペナルティ関数法，逐次2次計画法(SQP法)

経営工学：整数計画法，ミニマックス問題，ゲーム理論

情報工学：ニューラルネットワーク（NN)，進化戦略(ES)，

遺伝的アルゴリスム(GA)，遺伝的プログラミング(GP)

人工生命，シュミレーテッドアニーリング(SA),PSO

# シミュレーション最適化手法の概要③
## 構造解析分野から見た最適化手法の分類①

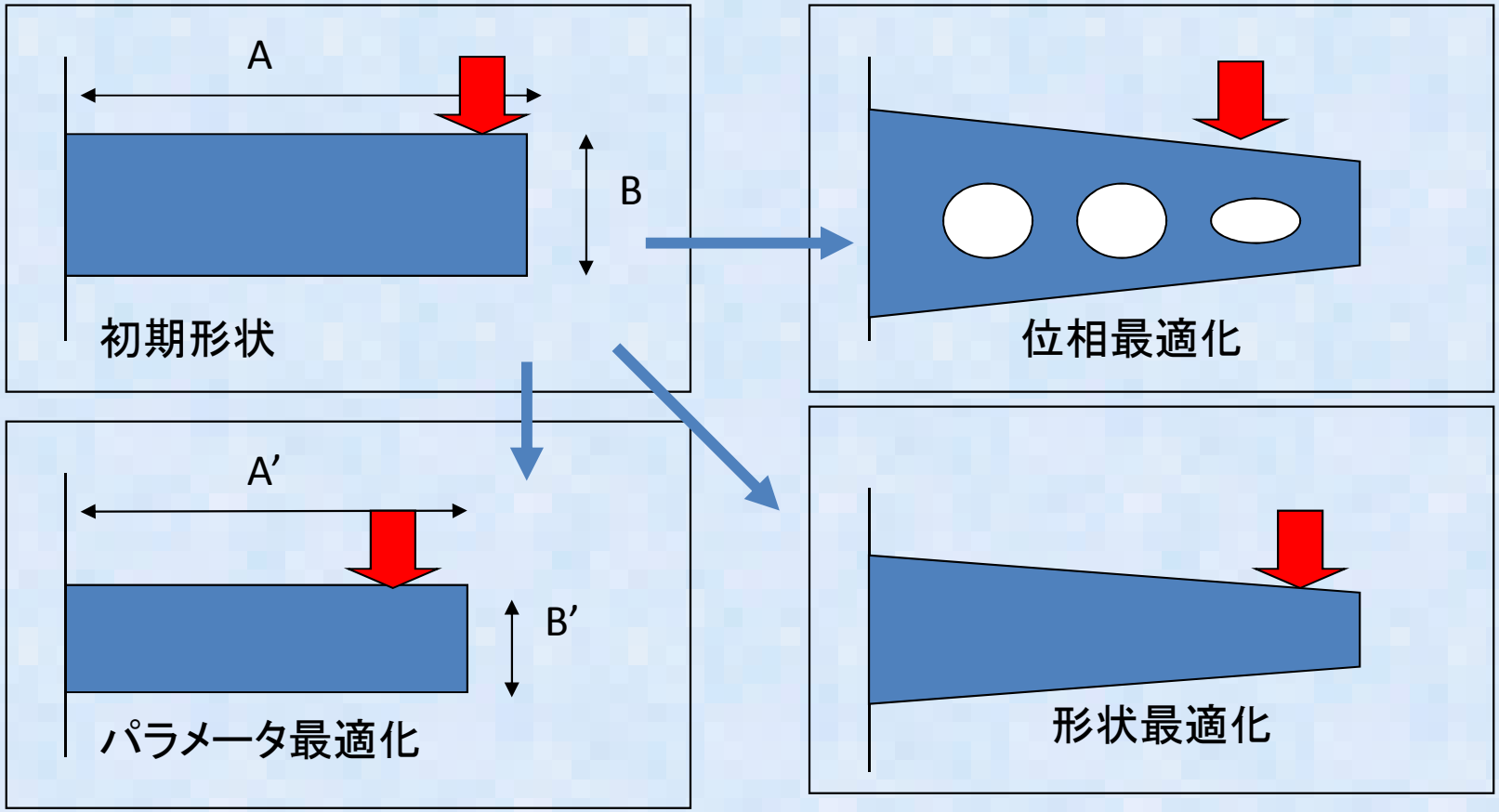
形態による分類：

－パラメータ（寸法）最適化問題：寸法、材料物性などを最適化

－形状最適化問題:位相を維持しつつ形状を最適化

－位相最適化問題:位相幾何学形状を最適化

ノンパラメトリック最適化 ⟺ DAKOTAはパラメータの最適化ソフトである



# シミュレーション最適化手法の概要④
# ー構造解析分野から見た最適化手法の分類②ー

具体的な解析手法：

ーパラメータ（寸法）最適化問題：

1)近似モデルを用いない

・GAなどの評価関数の勾配を用いないアルゴリズム

＋FEMプログラムの自動反復による最適化（汎用的）

2)近似モデルを用いる

・応答曲面法（RSM)，品質工学(タグチメソッド)

・感度解析，確率有限要素法，テーラ級数近似（線形問題）

ー形状最適化問題：

・Basis Vector法(Nastranなどのプログラムに装備)

ベーシスベクトル法は、設計が考えられる基本形状候補（ベーシスベクトル）の組み合わせにより最適形状を求める手法

・力法（Azegami et al.,1994）システム応力の一様化

ー位相最適化問題

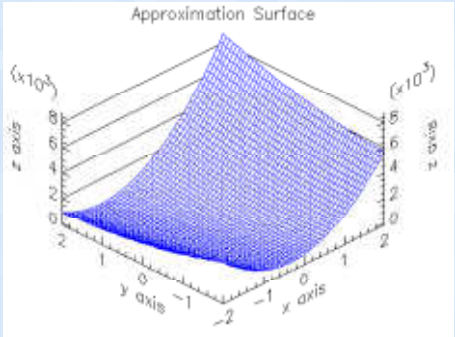
均質化法(菊池 他，Quint）平均コンプライアンスの最小化

[→類似手法:FSD(全応力設計法）構成部材応力を許容値以下]

# シミュレーション最適化手法の概要⑤
# ー構造解析分野から見た最適化手法の分類③ー

・非線形解析における最適化では近似モデル(RSM他)を用いる手法が多く用いられる。

ー理由：シミュレーションでは、ある設計条件が与えられた場合に1つの解(評価値,応力や変位や温度 その他)を返すが、設計条件(設計変数)が変化した場合の応答関数は数学一般における関数の最適化(最大化、最小化)問題のようにあらかじめ数式で与えられているわけではない。
　CAEの場合１つの設計変数に対する応答点を計算するのに膨大な計算時間を要する




多項式モデル

$$Y=\sum a_{ij}x_i x_j+\sum b_i x_i+c$$

幾つかの解析結果から、設計変数変化に対する応答を近似する
近似式を作成する事ができれば、その近似式の最適解を求める
事は従来の最適化技術で容易に実現できる。

$$Y = f(x_1, x_2, x_3, \ldots\ldots\ldots, x_n) \quad -> \quad max\ or\ min$$

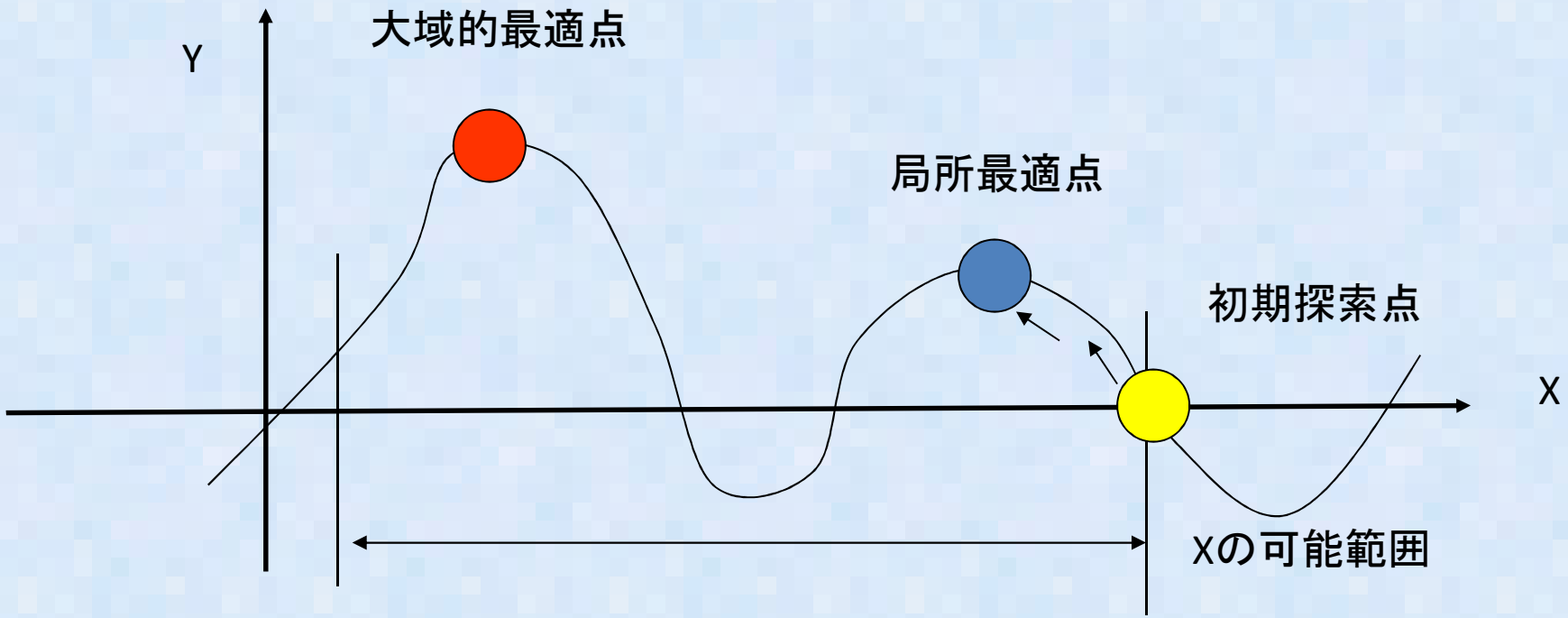
# (参考) 最適化で用いられる用語

ー最適化：構造、寸法、状態、方法、過程などを最も目的に良く合致させること

ー設計変数（説明変数）：最適化を行う際に変化可能な変数
（例：寸法、材料、荷重）
ー目的関数（評価関数）：目標の達成度合いを示す関数
（例：軽量化（質量）、最短化（時間）、応力最小化（応力））
ー制約条件：最適化問題に課せられる条件
（例：最大応力○ ○ 以下、金額、時間）

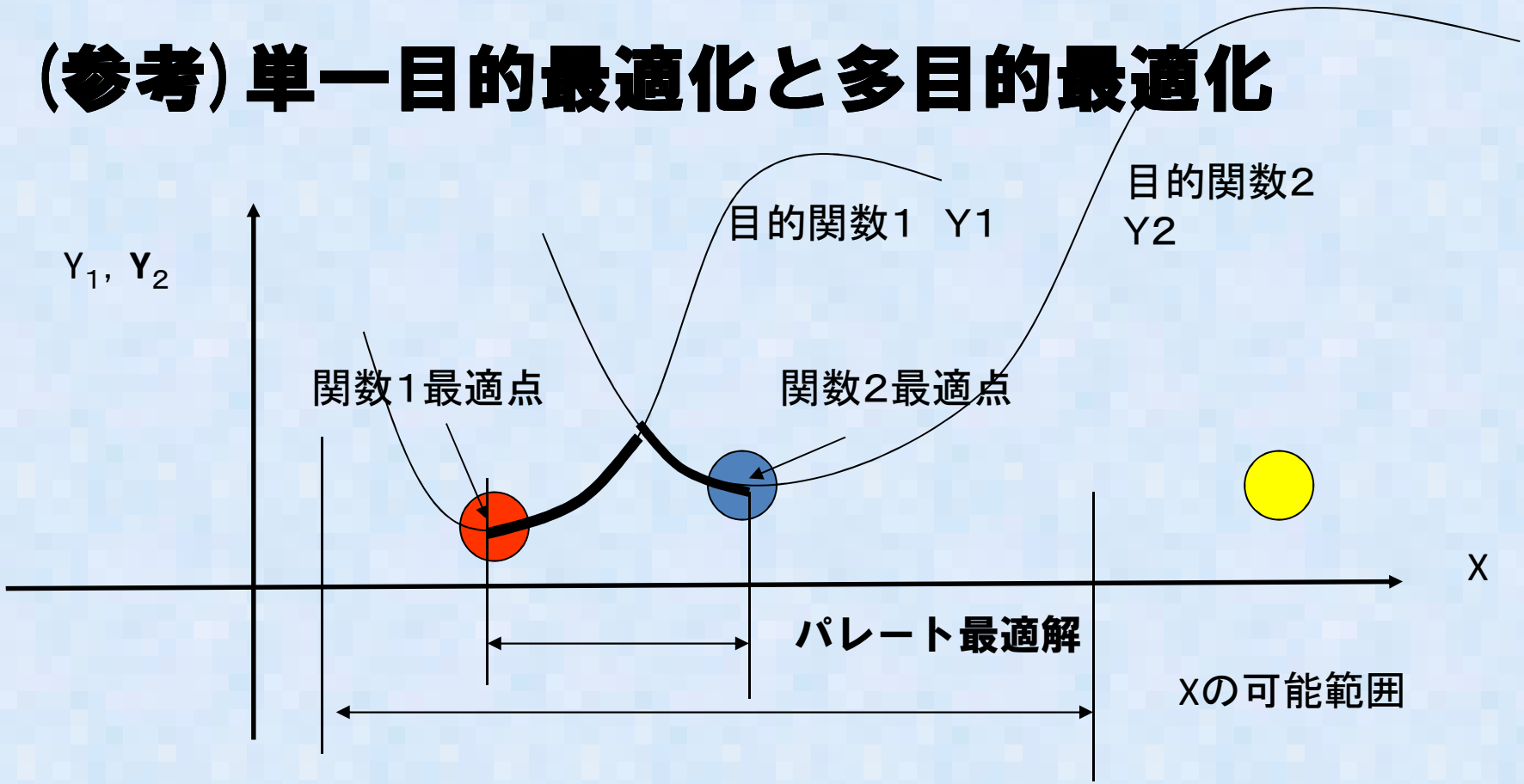
ー大域的最適化：解の全空間での最適化
ー局所最適化：ローカルな最適化を行う
ー単一目的最適化と多目的最適化

# (参考)大域的（Global)最適化と局所(Local)最適




－局所最適化：ある部分での局所的な最適化
　数理計画法などの古典的手法が確立
－大域的最適化：解の全空間での最適化
　＝＞　数理計画法などの古典的な局所的最適解を求める
アルゴリズムでは、大域的な最適解を得ることは難しい。
　＝＞　GA, SA などの手法が用いられる。

# (参考) 単一目的最適化と多目的最適化




ー単一の目的関数を持つ場合には唯一の最適解が存在する場合
でも、多目的最適化問題では、2つ以上の目的関数が単一の
最適解を持つことはあまり無い。
　　＝＞　多目的最適化問題では、他の部分より良いという意味で
最適解の集合を求め、これらをパレート最適解と呼ぶ。
　多目的最適化アルゴリズム
＝＞　スカラー化，優先順位法　などの手法が用いられる。

# Dakota インストール方法①

## ◆Windows　binary版①

**・Windows版はcygwin環境でコンパイル済みのモジュールが公開されているのでこれを展開して、パス設定をするだけで簡単。実行にはcygwinは不要**
**・しかし、試してみたところlinux環境と比較して実行速度が遅い.**
**おもに画面表示速度が遅い。Windows環境練習用と考えるべきであろう。**
**・グラフィックス機能（グラフ表示・作成など）は使えない**
**・Serial版 ：1CPUコアのみで実行可能**

**Home page** http://dakota.sandia.gov/
から　Dakota_5_2.cygwin.RC2.zip　を　Download
・どこかのフォルダに展開する。今回は”C:¥docs¥caetools¥Dakota_5_2¥Dakota”に展開した。
・パス設定する①
-Manual に従って展開フォルダPathの設定を行う
1) MS-DOSのコマンドプロンプト(DOS窓)を起動
2) 以下のコマンドでbinとtestにパス設定する

```
set PATH=C:¥docs¥caetools¥Dakota_5_2¥Dakota¥bin;C:¥docs¥caetools
¥Dakota_5_2¥Dakota¥test;%PATH%
```

しかし、この方法では、このDOS窓でしかPATH維持されないので面倒である

# Dakota インストール方法②

## ◆Windows　binary版②

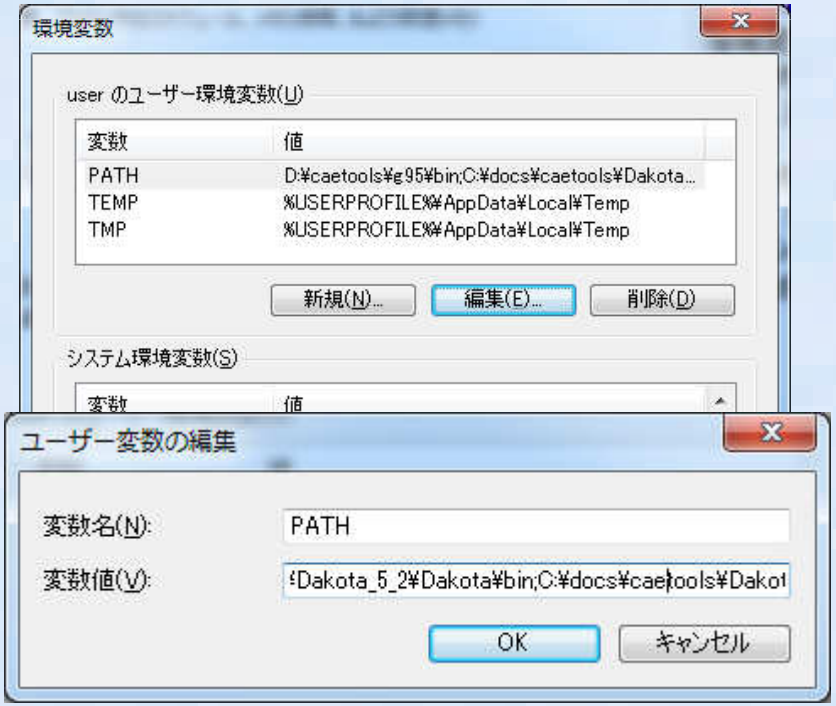
・Windowsの環境変数にPATH設定すれば、以降はこの設定が維持されるのでこちらの方が良いと思われる。Windowsの環境設定はXP,VISTA,7,8で微妙に違うので各自で確認ください。以下はWindows7の設定手順例です。

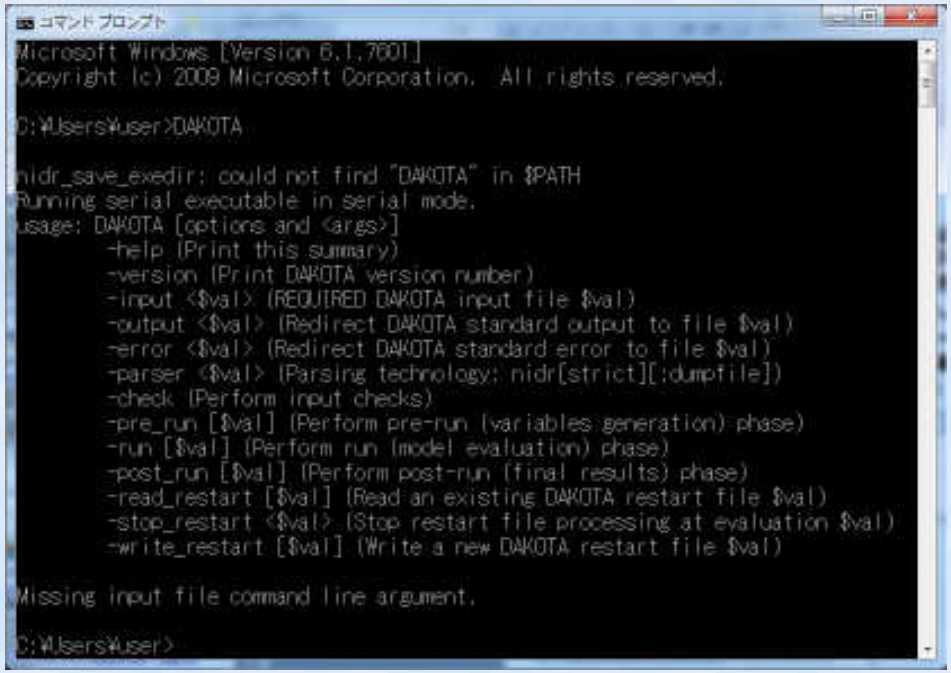
・コントロールパネルを起動し、システムとセキュリティを選択

システムを選択→　システムの詳細設定を選択　→　詳細設定のタグを選択

→　環境変数ボタンを押す　→　ユーザの環境編巣のPATH に”展開フォルダ名¥Dakota/bin, ～Dakota/test のPath　を追加する

・DOS窓で”dakota” とコマンドを入力してhelpが出れば成功



# Dakota インストール方法③
# -RedhatLinux系(binaryコンパイル済)-

- Dakota のバイナリ形式のモジュールはRedhat系のLinux版でしか公開されていないので、Redhat系のフリー版であるCentOSをVmwareでインストールしてみた。

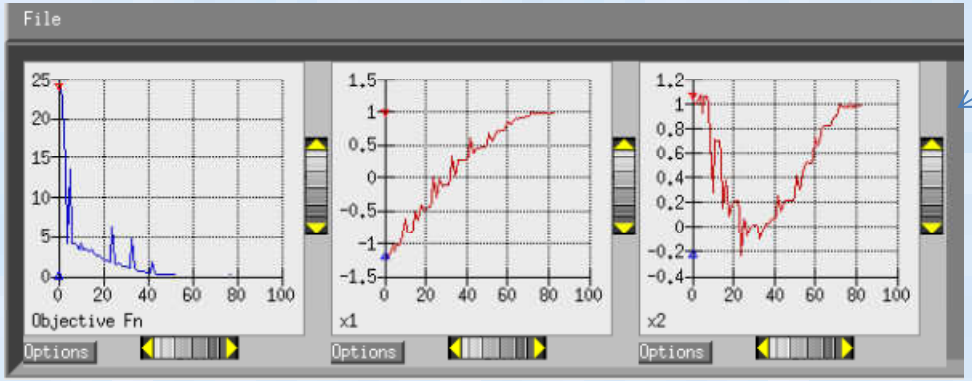
- したがって手順は以下の２段階
  1) CentOSのVmware へのInstall
  2) CentOSへのDakota のInstall

- CentOS版のinstall のメリットは コンパイルの必要なく、Graphicsを含めDakota全機能が使えること

# Dakota インストール方法④
# -RedhatLinux系(binaryコンパイル済)-

1) CentOSのVmware へのInstall
http://www.centos.org/modules/tinycontent/index.php?id=30 からCenOSのisoファイルをダウンロード 今回は“CentOS-6.3-x86_64-LiveCD.iso”をDownload した。
2) Vmwareに他isoと同様にinstall
3) DakotaのDakota_5_2.rhel6_x86_64.tar.gzをhttp://dakota.sandia.gov/download.phpからDownload
4) CentOSをVmwareで起動し、DakotaのモジュールをWorkDirectoryにコピー
5) Su コマンドでSuper userになる。
6) /opt/Dakota 下にDakota_5_2.rhel6_x86_64.tar.gzを展開
# cd /opt; tar -xvzf Dakota_5_2.rhel6_x86_64.tar.gz;

7) Path と LibraryPATHの設定する(しないと動かなかったので意外に重要)
(以下はcshでの設定例, bashrc系の場合は少し違うので注意)
setenv PATH ～既存のPATH～:/opt/Dakota/bin:/opt/Dakota/test
setenv LD_LIBRARY_PATH ～既存～:/opt/Dakota/bin:/opt/Dakota/lib
8) Dakota を起動 super user を抜けて、dakotaを起動
% **dakota -i dakota_rosenbrock_grad_opt.in**




Windows版で出てこなかったグラフィックス画面が出てきた

# Dakota インストール方法⑤
# -Dexcs2011系（コンパイル必須）

- DakotaのBinaryモジュールはRedhat(CentOS)系以外は公開されていないので、DexcsのUbuntsで実行するためにはコンパイルが必要である。
- 最低限の事前準備としてGfortranのインストールが必要で、後は各自の好みで追加モジュールを追加する。以下その手順

# Dakota インストール方法⑥

## -Dexcs2011系 (コンパイル必須)

1) Proxy 設定(普通の人は不要. 会社などLANから外部Internetに接続する場合にProxy設定の必要がある人)

export http_proxy="http://proxy.xxxxxx(proxyserver name) :8080/“

export ftp_proxy="http:// proxy.xxxxxx(proxyserver name) :8080 /“

2) Gfortran install

apt-get install gfortran

apt-get install libgfortran3

3) Dakota のSource file を任意Directory に展開 (例 /opt/Dakota)

cd /opt

(Super user になる)

sudo su

tar -xvzf /tmp/Dakota_5_2.src.tar.gz

/tmp に仮置き

cd Dakota

./configure --without-graphics --with-blas=/usr/lib/libblas.so.3gf.0 --with-lapack=/usr/lib/liblapack.so.3gf.0

-Graphics Optionではconfigure できなかった

make

make install

場所を指定しないと /usr/local/dakota に置かれる

-blas package , lapack package を指定する。Dexcs にはinstall 済みだったが 場所を明示する必要があるようだ

# Dakota インストール方法⑦ -Dexcs2011系（補足）

- OCSE^2 野村さんのホームページにDakota　Install の記事が最近紹介されてました。詳しくはそちらをご参照ください。
- http://mogura7.zenno.info/~et/wordpress/ocse/?p=871

- 野村さん指摘されているのですが、よく読んでみるとBoostパッケージが必要とのことですが、boostが無くてもinstall はできました。しかし、boost をinstall してからconfigure するとboost を参照しているようなので、心配な人は入れてからconfigure した方が良いと思います。

# Dakota 使用方法 -標準例題①-

## ◆標準例題 (User Manual P.27- DAKOTA Tutorial)

**・User Manual P.27- Tutorial に簡単なinstall 方法も記載**
**実行方法は P.29～記載**

**◆実行方法**
dakota dakota.in または
dakota -i dakota.in にて実行 (dakota.in はDakota のコマンド実行ファイル)

**◆標準例題** ( Rosenbrock function ローゼンブロック関数)
この例題の説明がマニュアル中で非常に多いので簡単に説明しておきます。

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$-2 \le x_1 \le 2$$

$$-2 \le x_2 \le 2$$

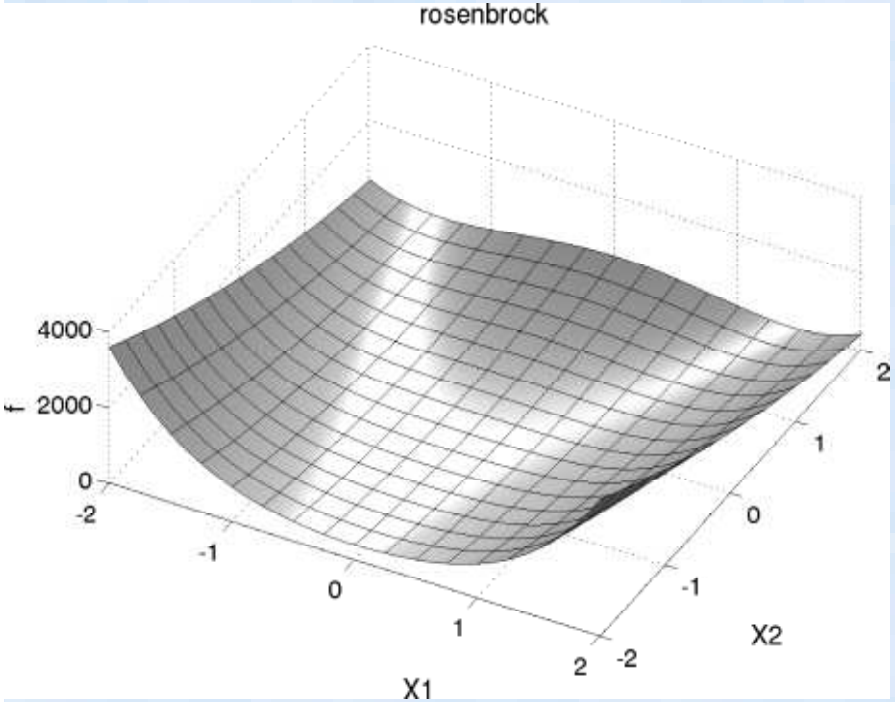
目的関数(評価関数) :f(x1,x2) → minimize
設計変数(制御変数): x1, x2
制約条件: -2<=x1,x2<=2

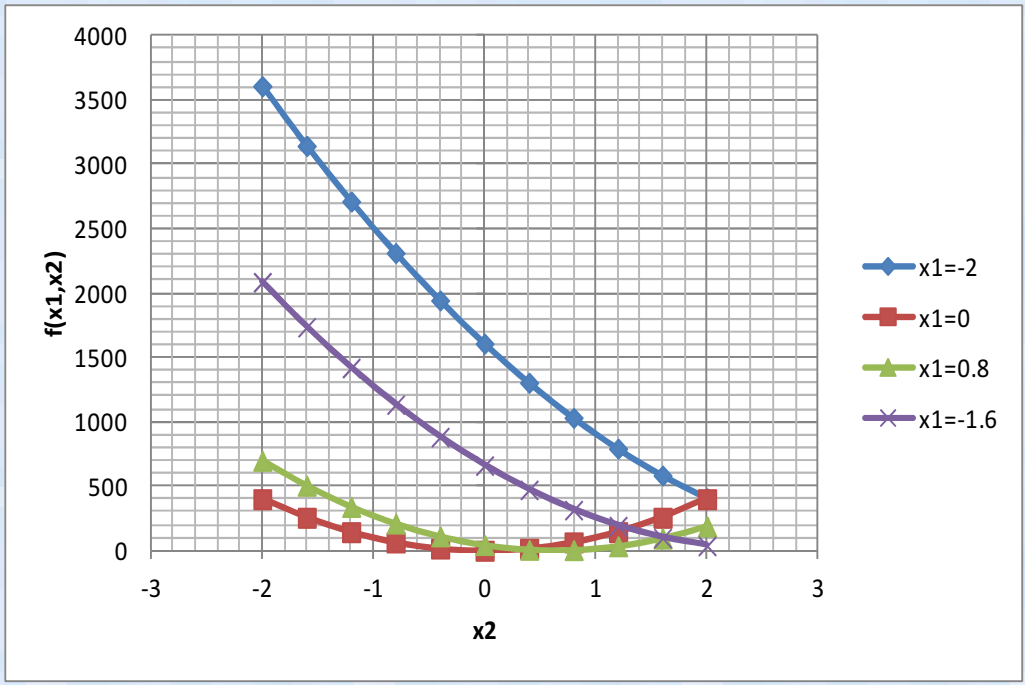
問題:f(x1,x2) を 最小化する
その時のx1,x2を求める。

# Dakota 使用方法 -標準例題②-

## ◆Rosenbrock 関数の概要

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2; -2 \le x_1 \le 2, -2 \le x_2 \le 2$$




**最大値： f=3609 （x1=−2, x2=−2）**
**最小値： f=0 （x1=1, x2=1）**

最小値の理論解は
f＝0 である。
(直観でわかるがちゃんと求めようとすると結構めんどうくさい)

# Dakota 使用方法 -標準例題③-

## ◆Dakota を使ったRosenbrock 関数最小化の計算例

Dakota により　Rosenbrock 関数の最小値を求める Dakota のSample プログラムは Example/tutorial directoryにあるので、自分の計算につかう作業フォルダにコピーしておくとよい.

“install directory”¥example¥tutorial¥dakota_rosenbrock_grad_opt.in

コピーが完了したら、作業フォルダに移動し、以下のコマンドを端末エミュレータ (Windows の場合は　DOS コマンド）　上で実行する

**dakota -i dakota_rosenbrock_grad_opt.in　>　grad_opt.out**

Dakota の計算が完了すると grad_opt.out に最適化計算がテキストで出力される。
(graphics が正しくinstall されているとグラフ画面も表示)
Dakota の計算による　最小値の出力結果

```
FINAL OPTIMIZATION INFORMATION
 OBJ =   0.749435E-04
 DECISION VARIABLES (X-VECTOR)
   1)   0.99342E+00  0.98744E+00
```

grad_opt.out の結果出力の最後に
最適解が出力される。
（正解とはほんの少しずれている。）

# Dakota 使用方法 -標準例題④-

## ◆dakota_rosenbrock_grad_opt.in の中身

```
## DAKOTA INPUT FILE - dakota_rosenbrock_grad_opt.in
strategy,
        single_method
          graphics,tabular_graphics_data
method,
        conmin_frcg
          max_iterations = 100
          convergence_tolerance = 1e-4
model,
        single
variables,
        continuous_design = 2
          initial_point   -1.2        1.0
          lower_bounds    -2.0    -2.0
          upper_bounds     2.0     2.0
          descriptors     'x1'    "x2"
Interface,
        direct

          analysis_driver = 'rosenbrock'
responses,
        objective_functions = 1
#       analytic_gradients
        numerical_gradients
          method_source dakota
          interval_type forward
          fd_gradient_step_size = 1.e-5
        no_hessians
```

Dakota の入力File に6個のブロックがある。
6個のブロック:
strategy(戦略)
method(方法)
model(モデル)
variables(変数)
interface(インターフェース)
responses(応答)

Strategy: 全体の目的
～ 単一目的関数の最適化(single), パレート最適　など

method: 最適やSamplingなどを行う手段・手法 NLP勾配法、GA, ランダムサンプル, 実験計画法 etc.

model: Surrogate Models=近似モデル（多項式など）、single(近似しない)

Variable: 変数設定
Interface: 外部インターフェース
Response: 目的関数の設定

# Dakota 使用方法 -標準例題⑤-

Conmin : Constrained Minimization (CONMIN) Library
制約付き最小化のライブラリ　frcg　はFletcher-Reeves conjugate gradient　(共役勾配法) のこと。

Method について補足

conmin_frcg

max_iterations = 100

convergence_tolerance = 1e-4

共役勾配法:
補足説明参照

最適化手法は別手法に
さしかえることも可能

**EA ( 進化アルゴリズム) に置き換えて計算を行ってみる**

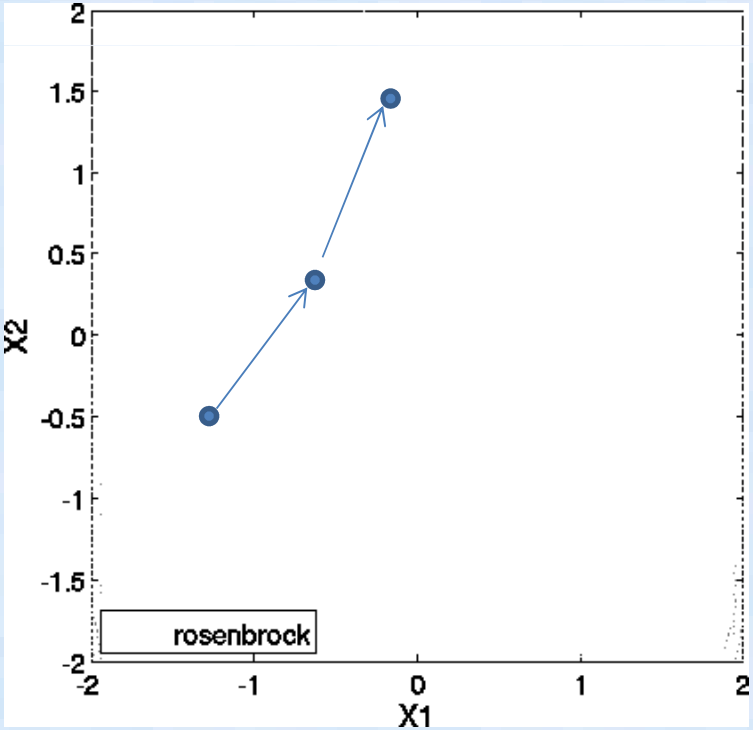
Tutorial に置き換えた例題が
既に準備されていた。
**dakota_rosenbrock_ea_opt.in**

Dakota は　オリジナルのプログラムを使って最適化などの処理を行っているわけではなく別の最適化処理を行うオープンソースのライブラリを呼び出している。 Conmin はその一つ。 その他はManual の7章を参照する

**HOPSPACK Library:**
**-asynch_pattern_search**
**SCOLIB Library:**
**-coliny direct, coliny solis wets, etc.**
**Design Optimization Tools (DOT) Library:**
**-dot_bfgs, dot sqp etc.**
**JEGA**
**- moga , soga**
**OPT++ Library:**
**- optpp_q_newton**　など多数

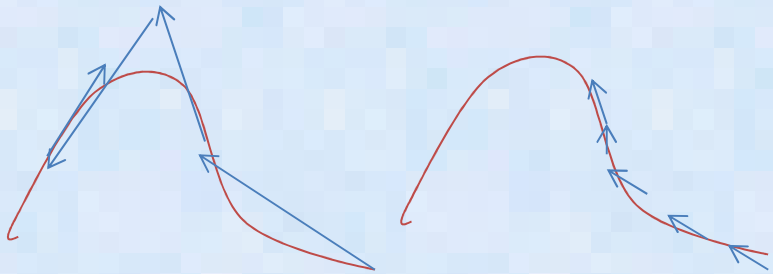
# Dakota 使用方法 -補足-

- 共役勾配法（CG法）： 非線形計画法の勾配法の一種で最適化手法の他に行列計算（Matrix Solver）の反復ソルバとして有名。類似方法に Newton法, 最大勾配法などがある。詳細は参考文献を参照。ここでは類似手法で最も簡単な最大勾配法の紹介のみ実施
- 最大勾配法：目的関数最大勾配方向に次の探索点を設ける。




$$\mathbf{x}_{n+1} = \mathbf{x}_n + K\nabla f$$

-K は定数：Kが各Stepで　固定のものが狭い意味での最大勾配法、KをStepごとに変更する方法は最適勾配法と呼ぶことがある
-Kの値が大きいと最適解の周辺で収束しにくくなる。Kが小さいと収束に時間がかかる。

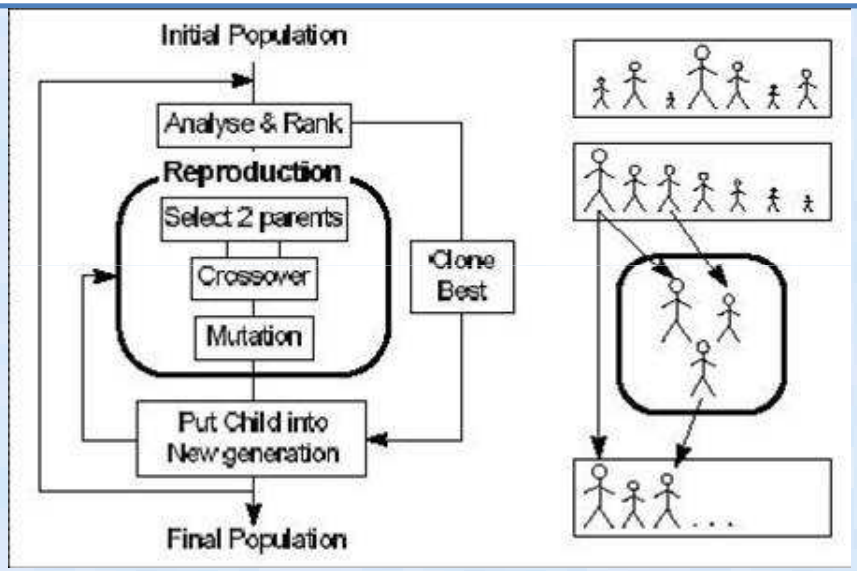


# Dakota 使用方法 -標準例題⑦-
# 最適化Option でEA (進化アルゴリズム) を用いた場合

Coliny_ea : Common Optimization Library Interface (COLIN)
共通最適化ライブラリ: 勾配法以外の最適化LIBRARY
EA はEvolutional Algorism 進化アルゴリズム(中身はGA)

Method 説明

```
method, coliny_ea
    max_iterations = 100
    max_function_evaluations = 2000
     seed = 11011011
     population_size = 50
    fitness_type merit_function
    mutation_type offset_normal
     mutation_rate 1.0
    crossover_type two_point
     crossover_rate 0.0
    replacement_type chc = 10
```




```
<<<<< Best parameters  =
 1.1817540107e+00 x1,  1.4035612403e+00 x2
<<<<< Best objective function  =  3.7960733143e-02
<<<<< Best data captured at function evaluation 218
```

ea.out の結果出力
X1=1.18, X2=1.40 ; F = 0.038
Ref. gradopt.out (勾配法)の結果
X1=0.99, X2=0.987; F=0.00749
EA(GA) の方が時間のかかる割に
精度は悪い

# （参考） Excel で同様の最適化計算をおこなう方法

## ◆Excel のSolver 機能を使う

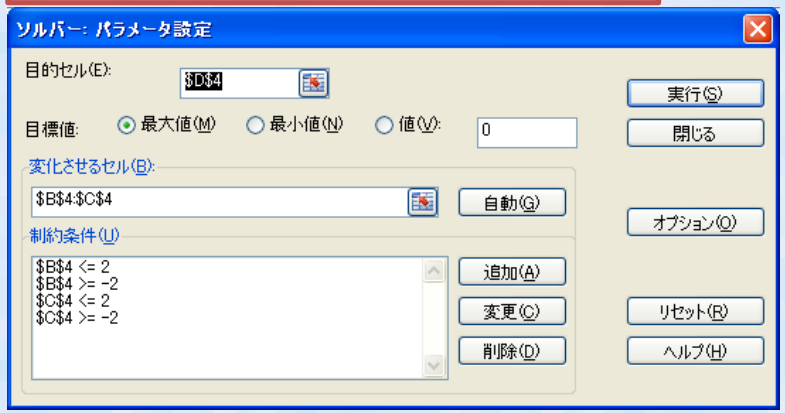
・Excel をinstall している段階ではSolver アドインがinstall されていない事があるので Solver アドイン をinstall する。
-Excel 上左端のリボンボタンを押し、Excel のオプションを選択→ アドインを選択 → ソルバーアドインを選択 → 設定 を押す → ポップアップメニューのソルバを選択→ OK (INSTALL が完了されるまで待つ。)

- Excel のシートに以下データ入力

| x1 | x2 | f |
|---|---|---|
| −2 | −2 | 3609 |

B4セル X1 として -2
C4セル X2として -2
D4セル Fとして以下数式
=100*(C4-B4^2)^2+(1-B4)^2

Excelのソルバを起動
-データ → 分析 → ソルバ
目的セル（目的関数） D4 セル
変化させるセル：自動
x1: b4>=-2, b4<=2
x2: c4 >=-2, c4<=2




Excelの計算による 最小値の出力結果

| x1 | x2 | f |
|---|---|---|
| 0.923626 | 0.848194 | 0.008225 |

Excel アドオン ソルバーを使用すると数式で入力した任意関数の最大・最小を求めることができる。

# Dakota と各ツールとの連携①

- Rosenbrock関数の例題は標準でDakotaに組み込まれているようで関数式を別のものに差し替えたり式を編州はできないようだ。
- Dakotaは任意のプログラムや解析ツールと連成する場合、スクリプトなど外部インターフェースを使って自動実行させる。
- 幾つかのスクリプトやbinary 連携ツールがexamples Directory　以下に提供されているのでこれを使って連携実行させる compiled interface 以下にはmatlab 以外はつかえそうもなくscript_interfacesの下のサンプルを利用することにした。
- ABAQUS, NASTRANなど解析ツールの他、Python script, Excel  Visual Basic script と　連携する例のsample がある。最初に簡単な例題としてWindows環境のVisual Basic script の例題を実施

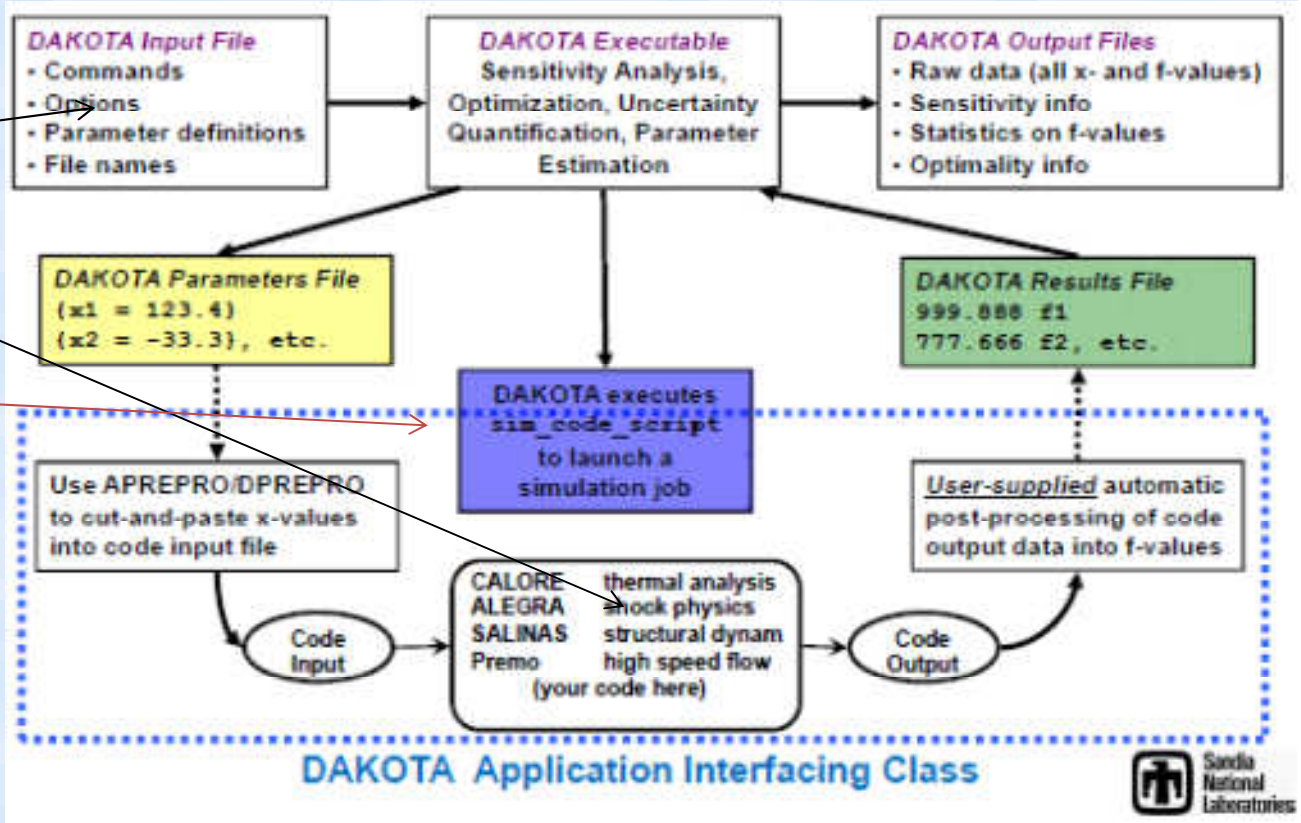
# Dakota と各ツールとの連携②

- Excel Visual Basic script例題
  -ファイルの場所：“Dakota install Directory”¥examples¥script_interfaces¥VisualBasic¥の下
  - 実行はWindows 環境, Excel がInstall されている必要がある

・実行ファイルは以下の3つ
-dakota_rosenbrock_vbs.in
( dakota input file)
-Rosenbrock.xls
(excel worksheet)
-RunExcelRosenbrock.vbs
(VisualBasic script)

・実行はdakota を
dakota_rosenbrock_vbs.in
を入力ファイルにして実行

%dakota -i
dakota_rosenbrock_vbs.in >
vbs.out



# Dakota と各ツールとの連携③

- 最適化の目的関数・制約条件はRosenbrock の例題と同じ。

Rosenbrock.xls を読み込み

Excel起動

この辺りはRosenbrock の例題とほぼ同じ

ここでVisual Basic Script を指定

```
# DAKOTA with Excel for Rosenbrock
strategy
  tabular_graphics_data
  single_method
method
  conmin_frcg
variables,
  continuous_design = 2
    initial_point   -1.0     1.0
    lower_bounds    -2.0    -2.0
    upper_bounds     2.0     2.0
    descriptors     'x1'    'x2'

interface
  analysis_drivers = 'cscript RunExcelRosenbrock.vbs //NoLogo'
  fork
    parameters_file = 'params.in'
    results_file = 'results.out'

responses,
  num_objective_functions = 1
  numerical_gradients
    fd_gradient_step_size = .000001
  no_hessians
```

```
REM VisualBasic example of computing Rosenbrock in Excel

REM Create a shell object for filesystem commands
Set wshShell = WScript.CreateObject("WScript.shell")
mainDirectory = wshShell.CurrentDirectory

REM Open the excel workbook and select Sheet1
Set ExcelApp = CreateObject("Excel.Application")
～　省略　～

REM Cleanup activities
REM ExcelApp.Workbooks(1).Save
ExcelWB.Close(false)
ExcelApp.Quit
Set ExcelApp = Nothing
```

# Dakota と各ツールとの連携③

- 最適化の目的関数・制約条件はRosenbrock の例題と同じ。

Rosenbrock.xls を読み込み

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Compute Rosenbrock function in B4 for return to DAKOTA | | | | |
| 2 | x1 | 1.00 | | | |
| 3 | x2 | 1.00 | | | |
| 4 | f(x1,x2) | 0.00 | | | |
| 5 | | | | | |

Excel起動

In File 中身

```
# DAKOTA with Excel for Rosenbrock
strategy
  tabular_graphics_data
  single_method
method
  conmin_frcg
variables,
  continuous_design = 2
    initial_point   -1.0     1.0
    lower_bounds    -2.0    -2.0
    upper_bounds     2.0     2.0
    descriptors     'x1'    'x2'

interface
  analysis_drivers = 'cscript RunExcelRosenbrock.vbs //NoLogo'
  fork
    parameters_file = 'params.in'
    results_file = 'results.out'

responses,
  num_objective_functions = 1
  numerical_gradients
    fd_gradient_step_size = .000001
  no_hessians
```

この辺りはRosenbrock の例題とほぼ同じ

ここでVisual Basic Script を指定

```
REM VisualBasic example of computing Rosenbrock in Excel

REM Create a shell object for filesystem commands
Set wshShell = WScript.CreateObject("WScript.shell")
mainDirectory = wshShell.CurrentDirectory

REM Open the excel workbook and select Sheet1
Set ExcelApp = CreateObject("Excel.Application")
～ 省略 ～

REM Cleanup activities
REM ExcelApp.Workbooks(1).Save
ExcelWB.Close(false)
ExcelApp.Quit
Set ExcelApp = Nothing
```

# Dakota と各ツールとの連携④

- Excel シートを使う利点は 独立変数と目的関数を自由に設定できること
- 例としてX1 1 変数の例題に書き換えてみた

In File 中身

```
# DAKOTA with Excel for Rosenbrock

strategy
  tabular_graphics_data
  single_method

method
  conmin_frcg

variables,
  continuous_design = 1
    initial_point   -1.0
    lower_bounds    -2.0
    upper_bounds     2.0
    descriptors      'x1'

interface
  analysis_drivers = 'cscript RunExcelRosenbrock2.vbs //NoLogo'
  fork
    parameters_file = 'params.in'
    results_file = 'results.out'

responses,
  num_objective_functions = 1
  numerical_gradients
    fd_gradient_step_size = .000001
  no_hessians
```

設計変数(独立変数) を1つにする

この辺りは同じ

Excel を書きかえる

B4 セル(目的関数); b2セル x1; b3セル x2

=100*(B3-B2*B2)*(B3-B2*B2)+(1-B2)*(1-B2)

Rosenbrock.xls

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Compute Rosenbrock function in B4 for return to DAKOTA | | | | |
| 2 | x1 | 1.00 | | | |
| 3 | x2 | 1.00 | | | |
| 4 | f(x1,x2) | 0.00 | | | |
| 5 | | | | | |

Rosenbrock2.xls

B4 セル(目的関数) b2 = x1 のみに変更

=100*(10-B2*B2)*(10-B2*B2)+(1-B2)*(1-B2)

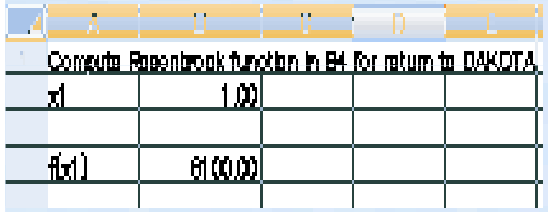
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Compute Rosenbrock function in B4 for return to DAKOTA | | | | |
| 2 | x1 | 1.00 | | | |
| 3 | | | | | |
| 4 | f(x1) | 8100.00 | | | |
| 5 | | | | | |

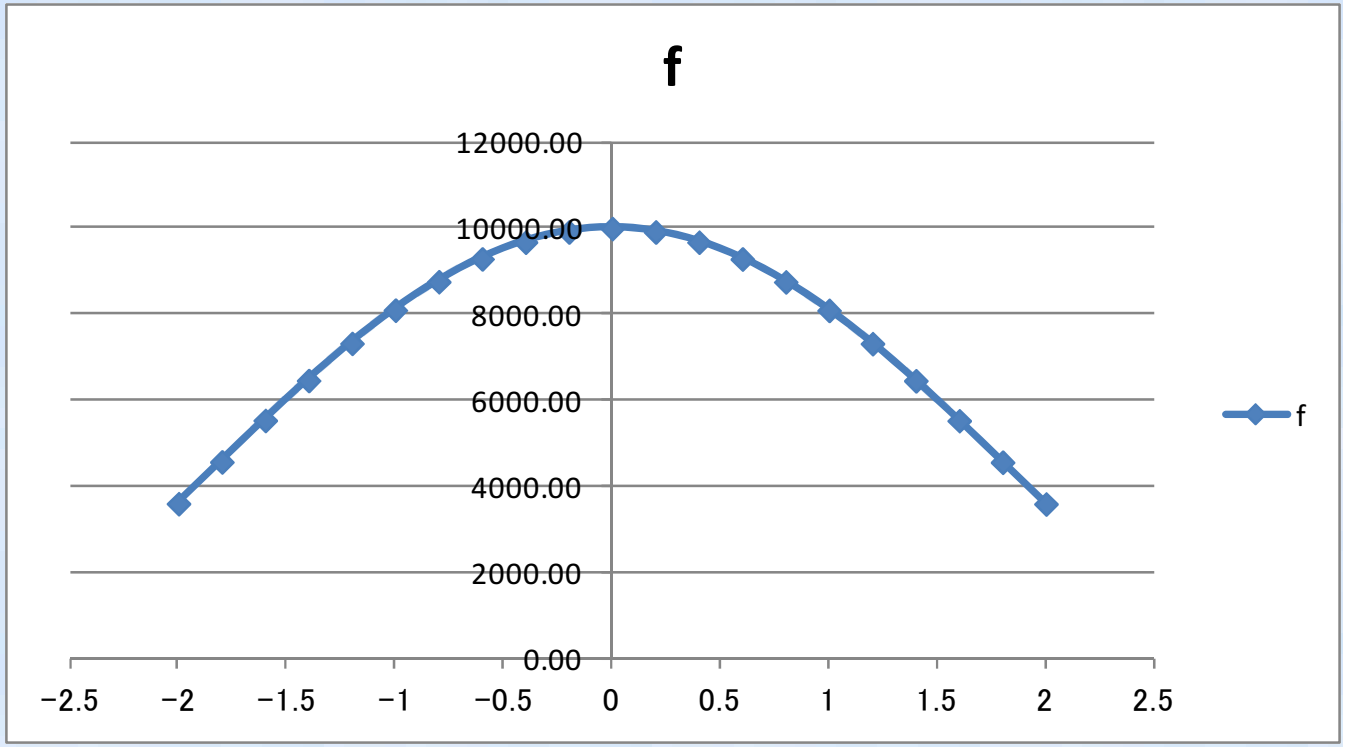
# Dakota と各ツールとの連携⑤

- dakota -i dakota_rosenbrock_vbs2.in > vbs.out (dos 窓で実行)
- vbs.out に結果が書き込まれる。

```
FINAL OPTIMIZATION INFORMATION
OBJ =   0.360900E+04     DECISION
VARIABLES (X-VECTOR)
 1)   -0.20000E+01
```

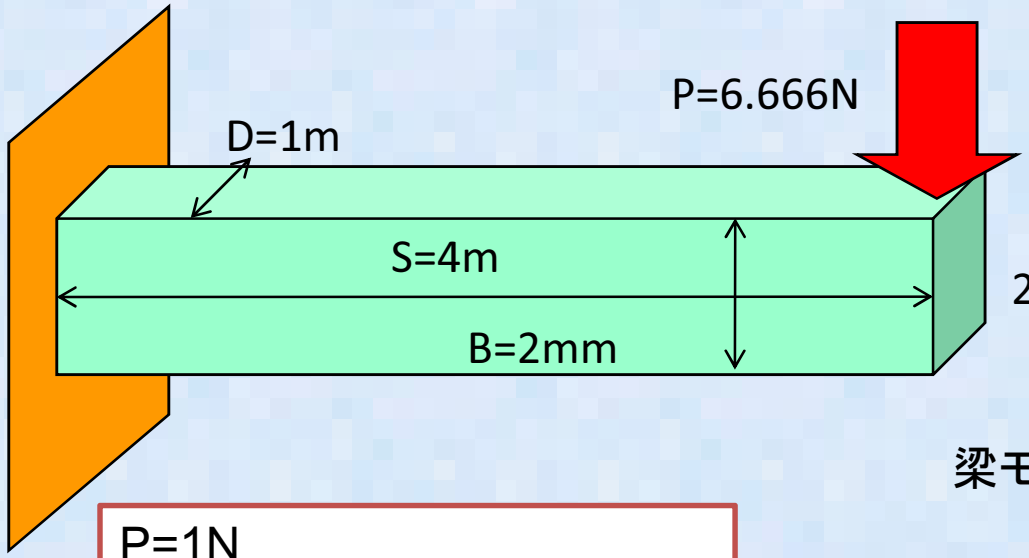
変数の端部で最小値を
とったのでこれを最適解と
判断したようだ。



# Dakota と各ツールとの連携⑥

- 片持ちはりの反り計算の例題でABAQUSのSample が入っていたので、これをOSS有限要素解析ソフト　Calculix　で動くように書き換えて実行してみる
- 前提:Linux 環境 (Ubuntsu)　Calculix linux版　install 済み
- Calculix V2.1 を使用




設計パラメータ

S=2～6m
B=1～3m
P=4～10N?

2次元平面応力要素を利用

P=1N
S= 4m
B= 2m
D= 1m
E=30e3Pa
I= $bt^3/12$

梁モデルの最大反り理論解

$$\delta = \frac{WL^3}{3EI}$$

# Dakota と各ツールとの連携⑦

- ①dakota_abaq_opt.in の中身を書き換える

```
strategy,
        single_method graphics
method,
        dot_mmfd
variables,
        continuous_design = 3
         cdv_initial_point =  4.0  2.0  6.66
         cdv_lower_bounds  =  2.0  1.0  4.0
         cdv_upper_bounds  =  6.0  3.0  10.0
         cdv_descriptor    =  'S' 'B' 'P'
interface,
        application system #asynch evaluation_concurrency = 5
         analysis_driver = 'abaq_driver'
         parameters_file = 'params.in'
         results_file = 'results.out'
         aprepro
responses,
        num_objective_functions = 1
        numerical_gradients
         method_source dakota
         interval_type central
         fd_step_size = .01
        no_hessians
```

今はこのモジュールなし。とりあえず下記入れ替え

```
method,
    conmin_frcg
      max_iterations = 100
      convergence_tolerance = 1e-4
```

Application は現在認識しないので　消す system だけ残す

Dakotaに今入っていない
Internet から　Source
Download しinstall
aprepro-2.01.tar.gz
(他のVersionはmake に失敗)

# Dakota と各ツールとの連携⑧

- fe.inp.appの書き換え(fe.inp.appはCalculix（abaqus）の入力ファイルのテンプレート

```
{ECHO(OFF)}
{include(params.in)}
{ECHO(ON)}
*HEADING
*NODE
    1,        0.0,        0.0
    2,      {S/4},        0.0
    3,      {S/2},        0.0
    4,        0.0,      {B/2}
    5,      {S/4},      {B/2}
    6,      {S/2},      {B/2}
    7,        0.0,        {B}
    8,      {S/4},        {B}
    9,      {S/2},        {B}
*ELEMENT,TYPE=CPS8,ELSET=BEAM
 1, 1, 3, 9, 7, 2, 6, 8, 4
**ELEMENT,TYPE=CPS4,ELSET=BEAM
** 1, 1, 2, 5, 4
** 2, 2, 3, 6, 5
** 3, 4, 5, 8, 7
** 4, 5, 6, 9, 8
*SOLID SECTION,ELSET=BEAM,MATERIAL=AL2024T3
1.0
*MATERIAL,NAME=AL2024T3
*ELASTIC
30.000000e3,     0.3
*BOUNDARY
 1, 2, , 0.0
 3, 1, , 0.0
 6, 1, , 0.0
 9, 1, , 0.0
*STEP
*STATIC
*CLOAD
** 9, 2,    3.3333333,
 9, 2,    {P/2},
*EL PRINT, POSITION=AVERAGED AT NODES, ELSET=BEAM
S
*END STEP
```

S, B, P はDakotaとAPREPROで変更する Parameter

Calculix　V2.1 ではCPS4 が無いのでCPS8 に変更した

Calculix　ではELSET 指定が必要

# Dakota と各ツールとの連携⑨

- fe.inp.appの書き換え(fe.inp.appはCalculix（abaqus）の入力ファイルのテンプレート

```
{ECHO(OFF)}
{include(params.in)}
{ECHO(ON)}
*HEADING
*NODE
    1,        0.0,        0.0
    2,      {S/4},        0.0
    3,      {S/2},        0.0
    4,        0.0,      {B/2}
    5,      {S/4},      {B/2}
    6,      {S/2},      {B/2}
    7,        0.0,        {B}
    8,      {S/4},        {B}
    9,      {S/2},        {B}
*ELEMENT,TYPE=CPS8,ELSET=BEAM
 1, 1, 3, 9, 7, 2, 6, 8, 4
**ELEMENT,TYPE=CPS4,ELSET=BEAM
** 1, 1, 2, 5, 4
** 2, 2, 3, 6, 5
** 3, 4, 5, 8, 7
** 4, 5, 6, 9, 8
*SOLID SECTION,ELSET=BEAM,MATERIAL=AL2024T3
1.0
*MATERIAL,NAME=AL2024T3
*ELASTIC
30.000000e3,     0.3
*BOUNDARY
 1, 2, , 0.0
 3, 1, , 0.0
 6, 1, , 0.0
 9, 1, , 0.0
*STEP
*STATIC
*CLOAD
** 9, 2,    3.3333333,
 9, 2,    {P/2},
*EL PRINT, POSITION=AVERAGED AT NODES, ELSET=BEAM
S
*END STEP
```

S, B, P はDakotaとAPREPROで変更する Parameter

Calculix　V2.1 ではCPS4 が無いのでCPS8 に変更した

Calculix　ではELSET 指定が必要

# Dakota と各ツールとの連携⑩

- Abaqus_driver を編集

```
#!/bin/csh -f

# $argv[1] is params.in FROM Dakota
# $argv[2] is results.out returned to Dakota

# Workdir setup for running in parallel (file_tag option turned on)
# In this simple case, all templatedir would contain is fe.inp.app
#set num = `echo $argv[1] | cut -c 11-`
#cp -r templatedir workdir.$num
#mv $argv[1] workdir.$num/params.in
#cd workdir.$num

# Pre-processing

aprepro --nowarning -q fe.inp.app fe.inp

# Run ABAQUS

#rm -f *.dat *.sta
/usr/bin/ccx_2.1 -i fe
####>>&! abaqus.out

# Post-processing

grep '   1  27' fe.dat | head -n 1 | awk '{print $3}' > $argv[2]

# Results file move :and workdir cleanup for running in parallel
#mv $argv[2] ../.
#cd ..
#rm -rf workdir.$num
```

ここをCalculix 用に編集

ここをCalculix 用に編集

# Dakota と各ツールとの連携⑪

Dakotaの最適化Reort
S=2, B=3, P=3 の時最適解が得られている。

- FINAL OPTIMIZATION INFORMATION

  OBJ = 0.674360E+00

  DECISION VARIABLES (X-VECTOR)

  1) 0.20000E+01 0.30000E+01 0.40000E+01

Sは梁の長さ、Bは梁の高さ(厚さ), Pは荷重なので
長さが短く、厚さは厚く、荷重はなるべく小さくすると
梁の曲げ応力が最小になるという、ある意味当たり前の
結果が得られている。

発表の時はこのページが欠けてましたすみません。

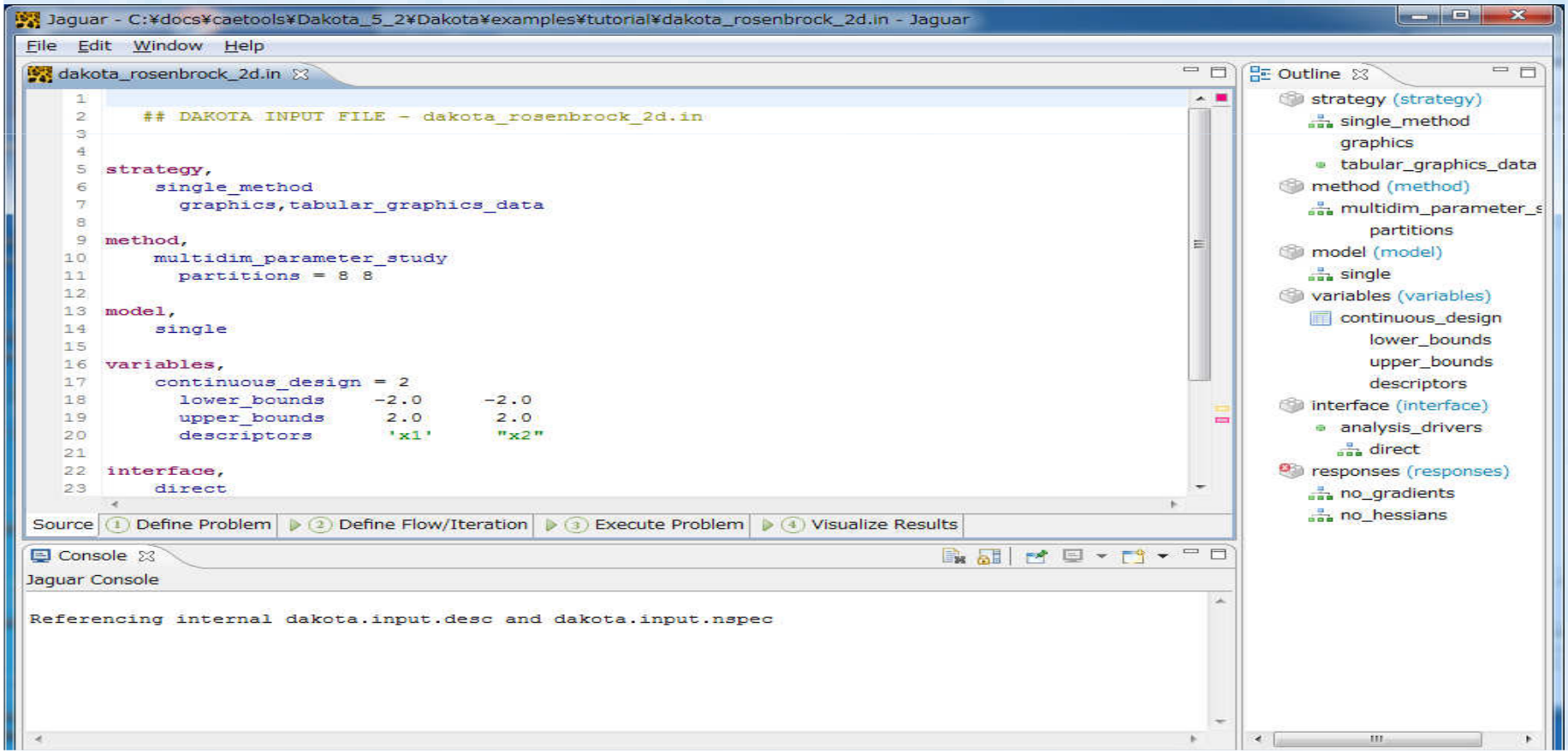
# GUIツール:Jaguar について

-Jaguar とは：Dakota 専用のGUI tool

-JAVAにて書かれている

ある程度 input file 作成をSupportする。

結果の分析機能もあるようだが未確認



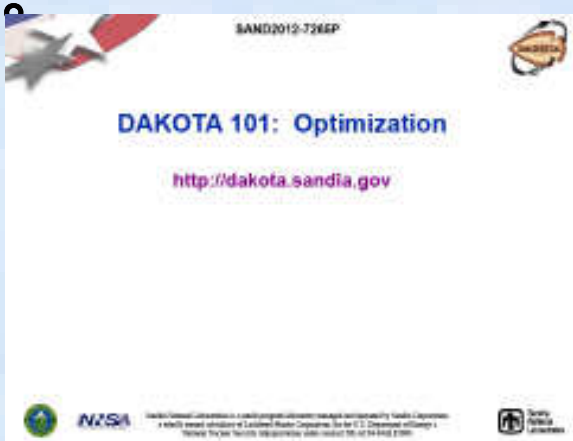
# 参考資料

① マニュアル類：User Manual , Reference Manual, Theory Manual, Developers Manual 4種類マニュアルがある。いずれもSandia national laboratory のホームページからDownload 可能.

http://dakota.sandia.gov

初心者は User's Manual, Chapter 2: Tutorialから読む

② Dakota の教育コースをSandia Labo. で実施している模様でこの Presentation 資料もホームページで公開されている。
Optimization（最適化), Calibration(パラメータチューニング），
Uncertain Qualification(信頼性)
などがある。マニュアルよりは分かりやすい。




③ Support(連絡先):
–dakota-users@software.sandia.gov
–dakota-help@sandia.gov

# まとめ

Dakotaの機能について調査し、Calculixと連携してはり曲げ問題で寸法最適化の計算を実施した。

今後他のツール(OpenFOAM等)との連携や最適化手法の調査を継続して実施する。