DirectSearch パッケージを用いた最適化問題

サイバネットシステム株式会社

▼背景

Maple 本体には Optimization パッケージと Statistics パッケージが標準で提供されています。
Optimization パッケージの Minimize コマンドを用いて、二乗誤差式の最小値を計算することで、フィッティングを行うことが可能です。
また、Statistics パッケージの NonlinearFit コマンドは Michaelis-Menton モデル($f(s) = \frac{a \cdot s}{b + s}$)等へのフィッティングも即時に行うことができます。

さらに、Maple 言語を用いてユーザによるカスタマイズパッケージ DirectSearch も開発されていて、Maplesoft 社の Application Center よりダウンロードの上でご利用いただけます。
このパッケージは非線形フィッティングの問題を処理する機能が強力です。詳細については、該当セクションで説明しています。
この資料では、モデルフィッティングを行うことを目的として、これらのパッケージ・機能について紹介します。

▼DirectSearch パッケージのセットアップ方法

DirectSearchパッケージを Maplesoft Application Center の以下のページからダウンロードします。

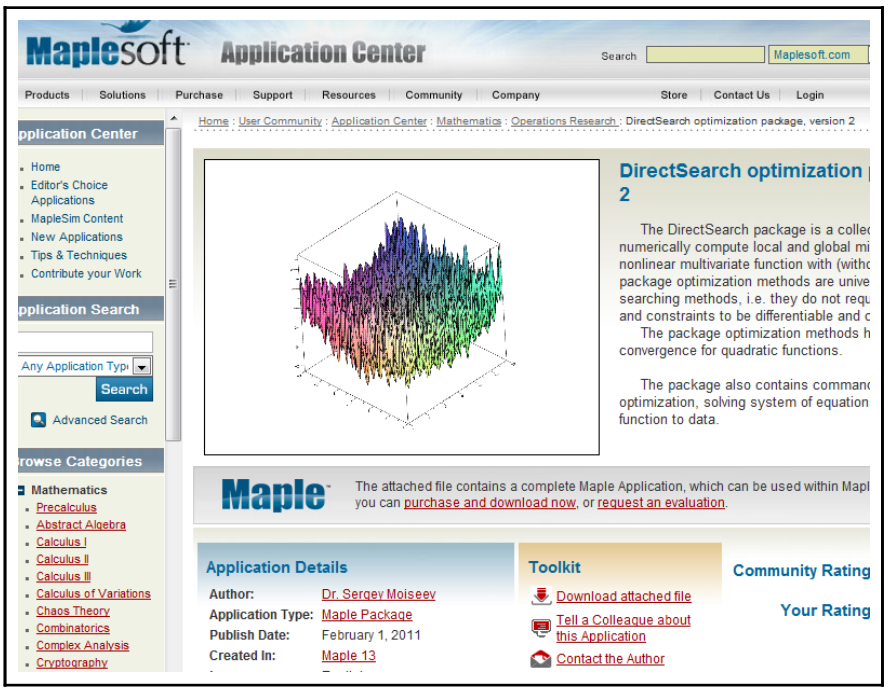
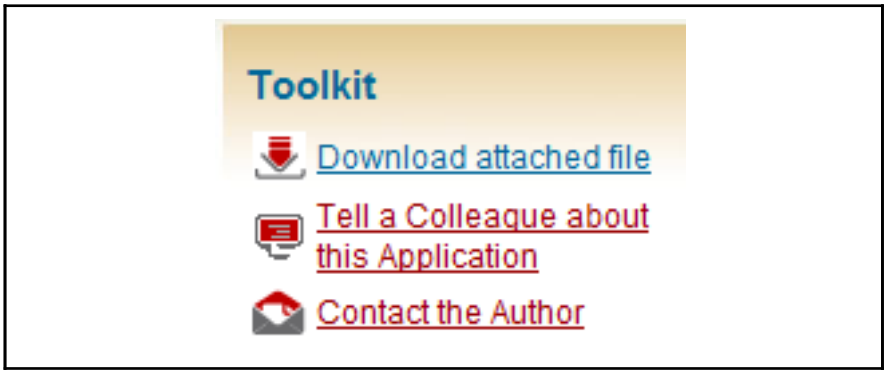


Table 1: http://www.maplesoft.com/applications/view.aspx?SID=101333

上記ページの [Toolkit]欄にある「Download attached file」をクリックして必要なファイル(DirectSearch2.zip)をダウンロードします。



2: Download attached file をクリックして必要なファイルをダウンロード

ダウンロードした DirectSearch2.zip ファイルは適当なフォルダ（例：C:\temp 等）に保存し、展開します。
展開したフォルダの "EnglishVersion" フォルダに格納されている以下の２つのファイルを Maple 本体のインストールフォルダ内にある「lib」フォルダへコピーします。

コピーするファイル：DirectSearch.mdb, DirectSearch.hdb
コピーするフォルダ：C:\Program Files\Maple 15\lib
注：上記は Maple 本体を Windows 版の標準フォルダへインストールした場合です。適宜インストールフォルダは確認下さい。

２つのファイルのコピーが完了したら Maple 本体を再起動し、ワークシート上で「?DirectSearch」とタイプするか、またはヘルプナビゲータを起動して「DirectSearch」とタイプし検索してください。DirectSearch パッケージのヘルプドキュメントが表示されれば、正しくインストールされています。

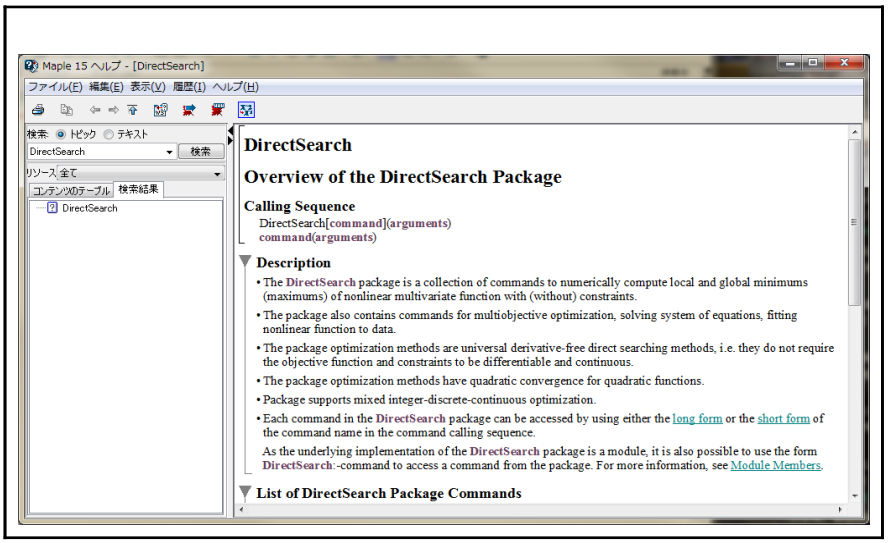


Table 3: DirectSearch パッケージのヘルプ画面

なお、展開したフォルダには、例題用のワークシート example.mw 及び DirectSearch パッケージで用いられている手法に関する参考論文の PDF が用意されています。必要に応じて参照して下さい。

注：なお、DirectSearch パッケージは Maplesoft の公認ではなく、あくまでユーザによって開発されたパッケージです。ご利用にあたってはユーザ様独自の判断に基づいてご利用下さい。本パッケージのご利用法にあたっての技術サポートは当社ではお

受け出来ませんのであらかじめご了承下さい。

Michaelis-Menton モデルへのフィッティング

```
> restart;
```

下記のような floating タイプの測定データが取得されているとして、それらのデータを Michaelis-Menton モデル式へフィッティングを行います。

まず、測定データを外部から Maple に取り込みます。
このデータは 46 x 2 の行列データです。

```
> data := ImportMatrix("biomeddata.csv", delimiter=",");
```

$$data := \begin{bmatrix} 46\ x\ 2\ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix} \tag{1}$$

上記 data の 1 列目を取り出して、変数 S に割り当てておきます。

```
> S := data[1..-1,1];
```

$$S := \begin{bmatrix} 1\ ..\ 46\ Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix} \tag{2}$$

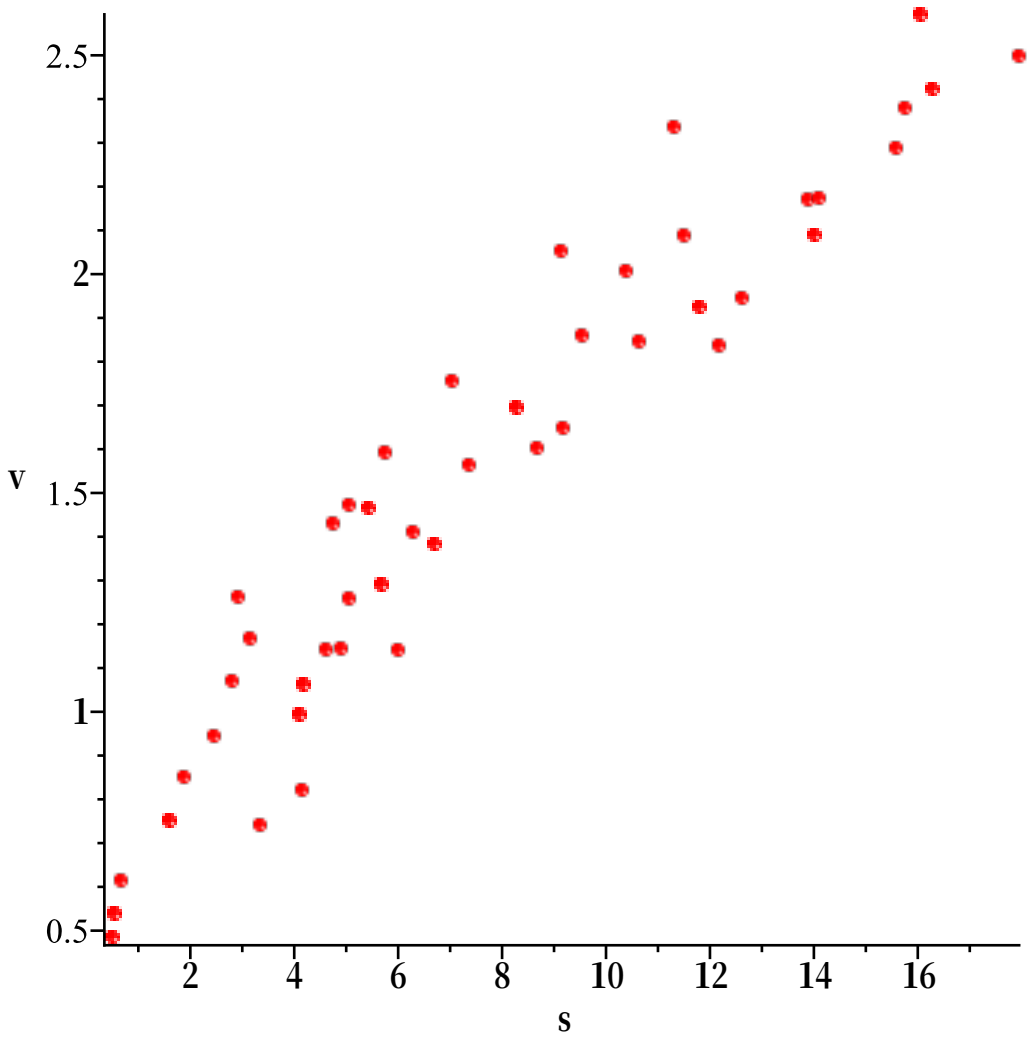
data の 2 列目を取り出して、変数 V に割り当てておきます。

```
> V := data[1..-1,2];
```

$$V := \begin{bmatrix} 1\ ..\ 46\ Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix} \tag{3}$$

データのプロットを描画し、確認します。

```
> plot(S, V, style=point, symbol=solidcircle, symbolsize=10,
  labels=[s,v]);
```



次に、上記 data を Michaelis-Menton モデル $f(s) = \frac{a \cdot s}{b + s}$ へフィッティングを行うことを考えます。
ここで、Maple の Optimization パッケージによる方法と Statistics パッケージによる方法をそれぞれ紹介します。

まず、モデル式を定義します。

```
> f := s->a*s/(b+s);
```

$$f := s \rightarrow \frac{a\,s}{b + s} \tag{4}$$

▼1. Optimization パッケージによる方法

最初に、二乗誤差式 SS を作成します。

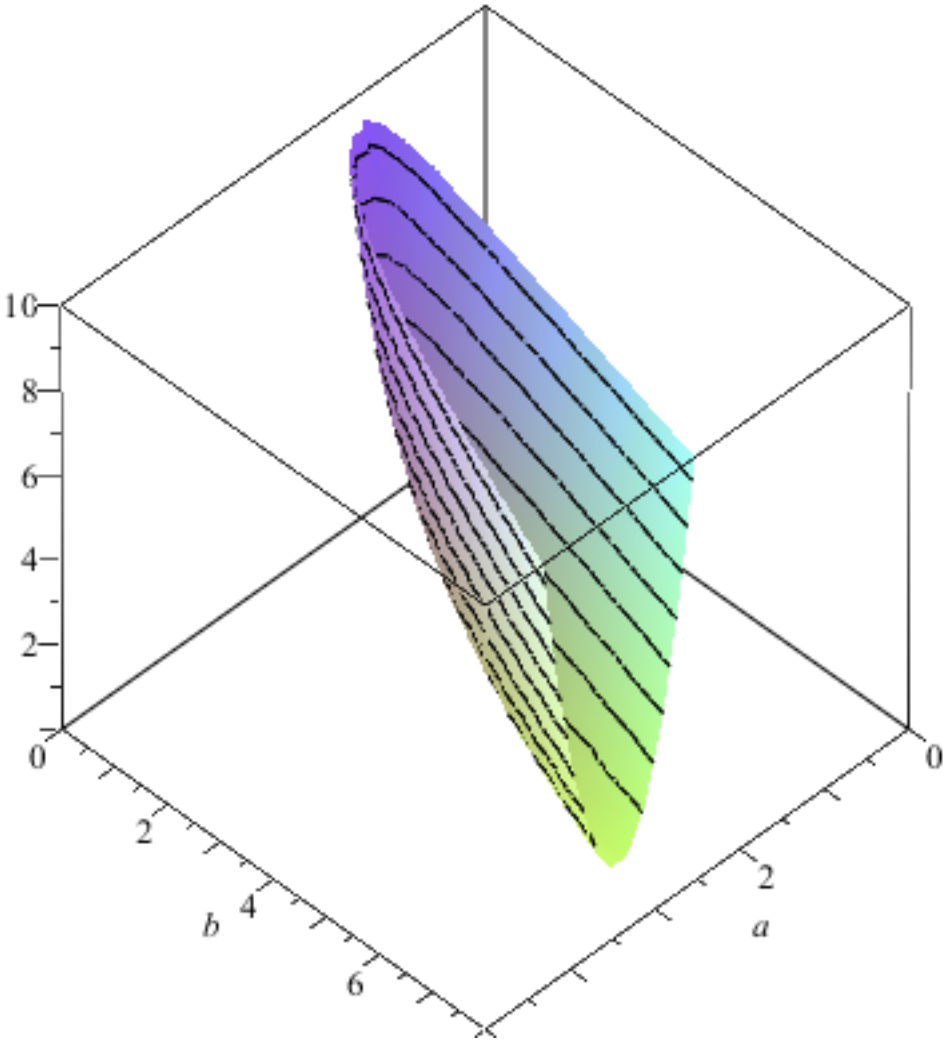
$$\sum_{i=1}^{n} \left(y_i - f\left(x_i\right)\right)^2$$

最小二乗法により、誤差式の f(x) の引数 x に S のデータ、y に V のデータをそれぞれ代入し、計算結果の総和を求めます。

```
> SS := add((f(S[k])-V[k])^2,k=1..46):
```

SS の計算でパラメータ a、b で構成されている式が得られ、以下のプロットにより二乗誤差の最小値を確認してみます。最小値はこの曲面の底部に存在していることがわかりますが、非常に狭い領域にあることもわかります。

```
> plot3d(SS, a=0..5, b=0..8, axes=boxed, style=patchcontour,
  view=[default,default,0..10]);
```



Optimizationパッケージの Minimize コマンドを用いて、二乗誤差 SS の最小値およびそのときのパラメータ a、b を計算してみます。

```
> with(Optimization):
> Minimize(SS);
```

$$[1.62257861139101212, [a = 3.47770700808553, b = 8.22462128995005]] \quad (5)$$

ここで得られた最小値は、実は正確ではありません。さらなる改良は、Optimizationパッケージの NLPSolve コマンドを用いることで得られます。
二乗誤差式 SS に NLPSolve コマンドを適用させて、再度、最小誤差およびパラメータ a、b の値を計算してみます。

なお、ここでは、非線形シンプレックス法を指定しており、許容誤差を 0.1e-14 に設定しています。

```
> NLPSolve(SS, method=nonlinearsimplex, evaluationlimit=200,
  optimalitytolerance=0.1e-14);
```

$$[1.62257861139007753, [a = 3.47770826399059, b = 8.22462700440175]] \quad (6)$$

▼2．Statisticsパッケージによる方法

```
> with(Statistics):
```

Statistics パッケージの NonlinearFit コマンドにより、非線形フィッティングを行います。
これにより、フィッティング後のモデル式が即時に得られます。

```
> NonlinearFit(f(s),S,V,s);
```

$$\frac{3.47770837575172\,s}{8.22462695881459+s} \tag{7}$$

DirectSearch カスタマイズパッケージによるフィッティング

概要

Maple は数式処理のソフトウェアではありますが、数値数式処理のみならず、ユーザ独自の関数・パッケージを開発する環境も提供しております。

この最適化パッケージは、Maple の数式処理機能を用いて、Maple 言語により開発された（サードパーティの）カスタマイズパッケージです。

Maple 本体に実装されている Optimization パッケージの NLPSolve コマンドは、Nelder- Mead による逐次シンプレックス法をサポートしております。
一方、DirectSearch パッケージには derivative-free direct search（微分情報を用いない直接探索）法をサポートしております。この方法は微分不可能な関数や不連続な関数の場合でも、最適化問題を解決することができます。

さらに、多目的最適化問題には、７つほどのコマンドが用意されております。
また、複素方程式モデルや連立方程式モデルによる最適化問題に対処するためのコマンド SolveEquations も開発されております。
ポイントデータにフィッティングするためには DataFit コマンドが用意されており、このコマンドでは、９つのフィッティング方法をサポートしております。
その９つのフィッティング方法は、下記表にまとめてあります。

method オプション	手法の概要	目的関数
lms	Least mean squares method. This method is effective but not robust to outliers.	$F(\mathbf{a})=\frac{1}{k}\sum_{i=1}^{k} w_i\cdot\left[Y_i-f\left(\mathbf{X}_i;\mathbf{a}\right)\right]^2$
lad	Least absolute deviations method. This method is more robust to outliers than LMS method.	$F(\mathbf{a})=\frac{1}{k}\sum_{i=1}^{k} w_i\cdot\left\|Y_i-f\left(\mathbf{X}_i;\mathbf{a}\right)\right\|$
lts	Least trimmed squares method. Robustness of this method depends on upper percentile value p ($p=95$ by default).	$F(\mathbf{a})=\mathrm{TrimmedMean}(\mathbf{r},0,p)$, $r_i=w_i\cdot\left\|Y_i-f\left(\mathbf{X}_i;\mathbf{a}\right)\right\|^2,\ i=1..k$, where TrimmedMean is trimmed mean
lws	Least Winsorized squares method. Robustness of this method depends on upper percentile value p ($p=95$ by	$F(\mathbf{a})=\mathrm{WinsorizedMean}(\mathbf{r},0,p)$, $r_i=w_i\cdot\left\|Y_i-f\left(\mathbf{X}_i;\mathbf{a}\right)\right\|^2,\ i=1..k$, where WinsorizedMean is Winsorized mean

	default).	
minimax	Minimax method. This method is very sensitive to outliers.	$F(\mathbf{a}) = \max_i \left(w_i \cdot \left\| Y_i - f(\mathbf{X}_i; \mathbf{a}) \right\| \right),\ i = 1 .. k$
median	Least median squares method. This method is most robust to outliers.	$F(\mathbf{a}) = \mathrm{me}\left(w_i \cdot \left\| Y_i - f(\mathbf{X}_i; \mathbf{a}) \right\|^2 \right),\ i = 1 .. k,$ where $\mathrm{me}(\ )$ is sample median
quantile	Least quantile squares method. Robustness of this method depends on quantile order p ($p = 0.75$ by default).	$F(\mathbf{a}) = \mathrm{q}_p\left(w_i \cdot \left\| Y_i - f(\mathbf{X}_i; \mathbf{a}) \right\|^2 \right),\ i = 1 .. k,$ where $\mathrm{q}_p(\ )$ is sample quantile of order p
trimean	Trimean method. This method is robust to outliers.	$F(\mathbf{a}) = \frac{1}{2}\left(\mathrm{me}(r_i) + \frac{q_{0.25}(r_i) + q_{0.75}(r_i)}{2} \right),$ $r_i = w_i \cdot \left\| Y_i - f(\mathbf{X}_i; \mathbf{a}) \right\|,\ i = 1 .. k,$ where $\mathrm{me}(\ )$, $\mathrm{q}_p(\ )$ is sample median and quantile of order p
lfad	Least function of absolute deviations method. Robustness of this method depends on the function g specified (by default, $g(x) = \ln(1 + x)$).	$F(\mathbf{a}) = \frac{1}{k} \sum_{i=1}^{k} w_i \cdot g\left(\left\| Y_i - f(\mathbf{X}_i; \mathbf{a}) \right\| \right)$

Table 1 DirectSearch パッケージの DataFit コマンドで利用可能な手法とその概要

以下は、 DirectSearch パッケージを用いたアプリケーション事例となります。

▼事例1

事例１では、上記「Michaelis-Menton モデルへのフィッティング」セクションで用いられたデータおよび Michaelis-Menton モデル $f(s) = \frac{a \cdot s}{b + s}$ を使い、DirectSearch パッケージの DataFit コマンドでフィッティングします。

DirectSearch パッケージに最小二乗法が実装されており、ここで、最小二乗法を指定してフィッティングを行います。
そのためには、fitmethod オプションに lms を指定しています。

```
> with(DirectSearch);
```

[*BoundedObjective*, *CompromiseProgramming*, *DataFit*, *ExponentialWeightedSum*, *GlobalOptima*, *GlobalSearch*, *Minimax*, *ModifiedTchebycheff*, *Search*, *SolveEquations*, *WeightedProduct*, *WeightedSum*] (8)

```
> DataFit(f(s), S, V, s, fitmethod = lms);
```

[0.0352734480736973, [$a = 3.47770827198454$, $b = 8.22462700463589$], 96] (9)

上記結果の１つ目の値 0.0352734480736973 は、Table1 の１行目 lms アルゴリズムの

関数 $F(\mathbf{a}) = \frac{1}{k}\sum_{i=1}^{k} w_i \cdot \left[Y_i - f\left(\mathbf{X}_i; \mathbf{a}\right) \right]^2$ で計算された最小値です。

この関数 F(a) は、セクション1 で使われた SS の関数と比較して、係数 1/k がある違いにご注意ください。
この事例の場合は、k が 46 になっているので、下記処理で Optimization パッケージによる計算と同様の結果を確認することができます。
(9)[1] は式(9) の 1 番目の要素を取り出すことになります。その要素は値 0.0352734480736973 になります。

```
> 46*(9)[1];
```

$$1.62257861139008 \tag{10}$$

▼ 事例2

ここでは、次の 3 つの式からなる連立方程式系の解を求める問題を考えます。

```
> restart;
```

各 g[i] は以下の非線型な連立方程式で表されるとします：

```
> g[1] := sqrt(a^2+(1-b)^2)*(arcsin((beta-a)/sqrt(a^2+(1-b)
  ^2))+arcsin(a/sqrt(a^2+(1-b)^2)))-3/2 = 0;
  g[2] := beta^4-beta^2*b+beta^2-2*beta*a-2*a*beta^3+2*a*
  beta*b-beta^3+beta*b-1+2*b-b^2 = 0;
  g[3] := (beta-a)^2+(beta^2-b)^2-a^2-(1-b)^2 = 0;
```

$$g_1 := \sqrt{a^2+1-2b+b^2}\left(\arcsin\left(\frac{\beta-a}{\sqrt{a^2+1-2b+b^2}}\right) + \arcsin\left(\frac{a}{\sqrt{a^2+1-2b+b^2}}\right)\right) - \frac{3}{2} = 0$$

$$g_2 := \beta^4 - \beta^2 b + \beta^2 - 2\beta a - 2a\beta^3 + 2a\beta b - \beta^3 + \beta b - 1 + 2b - b^2 = 0$$

$$g_3 := (\beta-a)^2 + \left(\beta^2-b\right)^2 - a^2 - (1-b)^2 = 0 \tag{11}$$

ただし、a, b, β はいずれも非負であるとします。

上記 3 つの連立方程式を fsolve コマンドにより、適当な初期値を与えて数値解を求めてみます。

```
> fsolve({g[1], g[2], g[3]}, {a=0.3, b=2, beta=1.2});
```

$$\{a = 0.2998750339,\ b = 2.003187464,\ \beta = 1.252187766\} \tag{12}$$

しかし、2 番目以降の解は fsolve では見つけ出すことが出来ません。ここで、fsolve の avoid オプションは以前に見つかった解以外の解を求めるためのオプションです。

```
> fsolve({g[1], g[2], g[3]}, {a=0.3, b=2, beta=1.2}, avoid=
  (12)) ;
```

$$fsolve\left(\left\{\sqrt{a^2+1-2b+b^2}\left(\arcsin\left(\frac{\beta-a}{\sqrt{a^2+1-2b+b^2}}\right) + \arcsin\left(\frac{a}{\sqrt{a^2+1-2b+b^2}}\right)\right) - \frac{3}{2} = 0,\ (\beta-a)^2 + \left(\beta^2-b\right)^2 - a^2\right. \tag{13}$$

$$-(1-b)^2=0,\ \beta^4-\beta^2 b+\beta^2-2\beta a-2a\beta^3+2a\beta b-\beta^3+\beta b-1+2b$$

$$-b^2=0\},\ \{a=0.3,\ b=2,\ \beta=1.2\},\ avoid=\{a=0.2998750339,\ b=2.003187464,$$

$$\beta=1.252187766\}\Big)$$

異なる初期値を与えても、やはり２番目以降の解は見つけられません。

```
> fsolve({g[1], g[2], g[3]}, {a=0.7, b=1.2, beta=2.2}, avoid=
  (12)) ;
```

$$fsolve\left(\left\{\sqrt{a^2+1-2b+b^2}\left(\arcsin\left(\frac{\beta-a}{\sqrt{a^2+1-2b+b^2}}\right)\right.\right.\right. \tag{14}$$

$$\left.+\arcsin\left(\frac{a}{\sqrt{a^2+1-2b+b^2}}\right)\right)-\frac{3}{2}=0,\ (\beta-a)^2+(\beta^2-b)^2-a^2$$

$$-(1-b)^2=0,\ \beta^4-\beta^2 b+\beta^2-2\beta a-2a\beta^3+2a\beta b-\beta^3+\beta b-1+2b$$

$$-b^2=0\},\ \{a=0.7,\ b=1.2,\ \beta=2.2\},\ avoid=\{a=0.2998750339,\ b$$

$$=2.003187464,\ \beta=1.252187766\}\Big)$$

そこで、より詳しく解の存在をグラフで確認するために、下記の処理を行います。

１．g[2] は変数 a に関して1次であることから、g[2]式を変数 {b, β} で表せるよう式を変形します。

```
> isolate(g[2],a);
```

$$a=\frac{-\beta^4+\beta^2 b-\beta^2+\beta^3-\beta b+1-2b+b^2}{-2\beta-2\beta^3+2\beta b} \tag{15}$$

２．(15)式を g[1] に代入します。

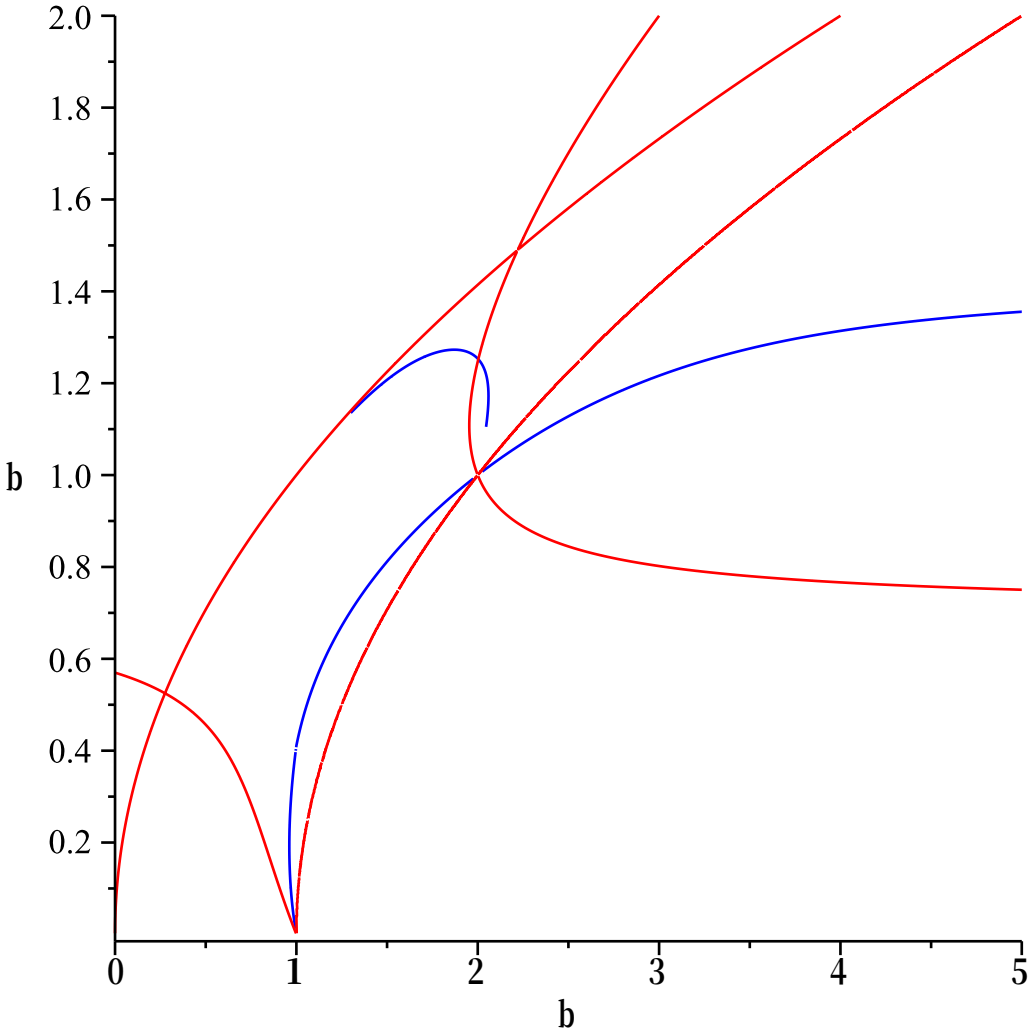
```
> g2:=eval(g[1],(15)) :
```

同様に、g[3] にも代入します。

```
> g3:=eval(g[3],(15)) :
```

３．上記処理により、変数 b と β のみの２つの方程式 {g2, g3} が用意できました。
そこで、陰関数描画の implicitplot コマンドを用いて、２つの方程式のグラフを描画し、交点を確認してみます。

```
> with(plots):
> implicitplot(
    [g2,g3], b=0..5, beta=0..2,
    color=[blue,red], gridrefine = 5,crossingrefine = 2) ;
```



ここで、(12) の結果を改めて確認してみます：

```
> (12);
```

$$\{a = 0.2998750339,\ b = 2.003187464,\ \beta = 1.252187766\} \tag{16}$$

すなわち、fsolve による (12) の値は、グラフ中の真ん中に位置している交点のみを探しだしていることがわかります。

一方、DirectSearch パッケージの SolveEquations コマンドを用いて、この系の解を計算してみます。

```
> with(DirectSearch):
> M:=SolveEquations([g[1],g[2],g[3]],AllSolutions,
  evaluationlimit=100000);
```

$$M := \begin{bmatrix} 86\ x\ 4\ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix} \tag{17}$$

SolveEquations コマンドは $R = \sum_{k=1}^{n} g_k^2$ の残差式の極小値を計算し、戻り値として m × 4 の行列を出力します。

行列データの 各行は求まった極値及び関連するデータが格納されています。各列は、１列目：残差 R、２列目：各式の残差（ここでは g[i], i=1..3の残差）、３列目：

求まったパラメータ値、４列目：反復回数、の各データが格納されています。

今回の例において、３行目までの結果は以下のようになっています：

```
> M[1,1..-1];
```

$$\left[8.60871060845570\ 10^{-22}, \begin{bmatrix} 2.41477948748070\ 10^{-11} \\ 1.38351552436689\ 10^{-11} \\ -9.29212262690271\ 10^{-12} \end{bmatrix}, [a = 0.299875033920208, b = 2.00318746379434, \beta = 1.25218776615005], 569\right] \quad (18)$$

```
> M[2,1..-1];
```

$$\left[1.90338069846921\ 10^{-10}, \begin{bmatrix} -3.70444055430852\ 10^{-8} \\ 1.63638915751108\ 10^{-7} \\ -0.0000137952861465136 \end{bmatrix}, [a = 0.489459117954082, b = 0.961265937303055, \beta = 0.980434433292770], 1935\right] \quad (19)$$

```
> M[3,1..-1];
```

$$\left[0.0000169029659446024, \begin{bmatrix} 0.000563252656696100 \\ 0.000439901355975492 \\ -0.00404872809488833 \end{bmatrix}, [a = 0.518917167300393, b = 1.08921253440662, \beta = 1.04157030838497], 2741\right] \quad (20)$$

デフォルトで、ここでの極値計算は絶対許容誤差が 10^{-6} であり、３行目以降の極値は 1.7×10^{-5} になっているため、誤差の有意性から判断して実際の結果は１行目と２行目のみになります。

微分不可能な関数モデルや連立方程式系モデル、複素方程式モデルによる最適化問題の場合は、カスタマイズされたDirectSearchパッケージを利用することで最適値を求められる可能性があります。
DirectSearchパッケージには、本資料で紹介したDataFit, SolveEquationsコマンド以外にも、いくつかの最適化手法が用意されていますので、最適化問題に対する別のアプローチを利用することが可能です。