8.数式処理による最適化問題の例

8.1 Optimization package （標準パッケージ）による線形フィッティング

局所最適化での例

```
> restart;
```

まず、以下の線形モデルを考えます。

```
> model := t -> a + b*cos(t);
```

係数パラメータは、以下の 2 変数です。

```
> params := [a,b];
```

適当な乱数を元に、サンプル用のモデル式を作ります。

```
> randomize():
  f :=
  unapply(
    subs(
      {'params[i] = evalf(rand()/10^12)' $ i = 1..nops(params)},
      model(t)
    ),
  t);
```

上記で生成したモデルに、評価データ点を与えてサンプル点を生成します。

```
> r := [evalf(seq(2*Pi*t/99, t=0..99))]:
> data := [evalf(seq([t, f(t)], t=r))]:
> plots[pointplot](data);
  g1 := plots[pointplot](data):
```

このモデルに、元々のモデル式 model(t) の各係数を、Optimization パッケージの Minimize 関数により求めてみます。
まず、model(t) の式と上記サンプル点の各値における二乗誤差式を作ります。

```
> sqr :=
  expand(
    add((model(data[i,1]) - data[i,2])^2, i=1..nops(data))
  ):
```

この sqr で与えられた関数において、各係数 params を最小にする値を求めてみましょう。

```
> infolevel[Optimization]:=3:
> mincoefs := Optimization[Minimize](sqr, iterationlimit=1000,
  assume=nonnegative);
```

得られた係数を model(t) に割り当てます。

```
> assign(%[2]);
> model(t);
```

実際のモデルと係数を比較してみましょう。

```
> f(t);
> g_minimized := plot(model(t), t=r[1]..r[-1]):
> plots[display]([g1, g_minimized]);
```

Minimize 関数（または Maximize）関数は、内部で自動的に線形・2次・非線形の最適化関数を呼び出すことに注意してください。

8.2 Optimization package （標準パッケージ）による非線形フィッティング

局所最適化での例

```
> restart;
> model := t -> a + b*exp(t*c);
```

係数パラメータは、以下のように 3 変数です。

```
> params := [a,b,c];
```

適当な乱数を元に、サンプル用のモデル式を作ります。

```
> randomize():
```

```
f :=
unapply(
  subs(
    {'params[i] = evalf(rand()/10^12)' $ i = 1..nops(params)},
    model(t)
  ),
t);
```

上記で生成したモデルに、評価データ点を与えてサンプル点を生成します。

```
> r := [evalf(seq(2*Pi*t/99, t=0..99))]:
> data := [evalf(seq([t, f(t)], t=r))]:
```

サンプルされた点リストを描画してみましょう。

```
> plots[pointplot](data);
  g1 := plots[pointplot](data):
```

このモデルに、元々のモデル式 model(t) の各係数を、Optimization パッケージの Minimize 関数により求めてみます。
まず、model(t) の式と上記サンプル点の各値における二乗誤差式を作ります。

```
> sqr :=
  expand(
    add((model(data[i,1]) - data[i,2])^2, i=1..nops(data))
  ) :
```

この sqr で与えられた関数において、各係数 params を最小にする値を求めてみましょう。

```
> infolevel[Optimization]:=3:
  #係数の範囲と初期スタート点を指定します。
> mincoefs := Optimization[Minimize](sqr,a=0.1..10,b=0.1..10,c=
  0.1..10,initialpoint={a=0.8,b=0.5,c=0.65},iterationlimit=3000,
  assume=nonnegative);
```

得られた係数を割り当てます。

```
> assign(%[2]):
```

割り当て後のモデル式

```
> model(t);
> g_minimized := plot(model(t), t=r[1]..r[-1]):
> plots[display]([g1, g_minimized]);
```

何回も乱数を発生させてグラフで確認すると、きちんとフィッティングできない場合があります。
これは局所最適化 (Minimize 関数) における限界のひとつです。

8.3 Global Optimization Toolbox による非線形フィッティング

大域最適化での例

```
> restart;
```

まず、以下の非線形モデルを考えます。

```
> model := t -> a + b*cos(2*Pi*c*t) + d*sin(2*Pi*e*t);
```

係数パラメータは、以下のように 5 変数です。

```
> params := [a,b,c,d,e];
```

適当な乱数を元に、サンプル用のモデル式を作り上げます。

```
> randomize():
  f :=
  unapply(
    subs(
      {'params[i] = evalf(rand()/10^12)' $ i = 1..nops(params)},
      model(t)
    ),
  t);
```

上記で生成したモデルに、評価データ点を与えてサンプル点を生成します。
サンプルされた点リストを描画してみます。

```
> r := [evalf(seq(2*Pi*t/99, t=0..99))]:
> data := [evalf(seq([t, f(t)], t=r))]:
> plots[pointplot](data);
```

```
   g1 := plots[pointplot](data):
```
このモデルに、元々のモデル式 model(t) の各係数を、GlobalOptimization パッケージの GlobalSolve 関数により求めてみます。
まず、model(t) の式と上記サンプル点の各値における二乗誤差式を作ります。
```
> sqr :=
  expand(
    add((model(data[i,1]) - data[i,2])^2, i=1..nops(data))
  ):
```
この sqr で与えられた関数において、各係数 params を最小にする値を求めてみましょう。
```
> infolevel[GlobalOptimization]:=3:
> mincoefs := GlobalOptimization[GlobalSolve](sqr,a=-1..1,b=-1..1,
  c=-1..1,d=-1..1,e=-1..1);
```
得られた係数を model(t) に割り当てます。
```
> assign(%[2]);
> model(t);
```
実際のモデルと係数を比較してみましょう。
```
> f(t);
> g_minimized := plot(model(t), t=r[1]..r[-1]):
> plots[display]([g1, g_minimized]);
>
```

Copyright (c) 2009, Cybernet Systems, Co., Ltd. All rights reserved.

```
>
```