ホウ素濃度計濃度推定アルゴリズム開発
ー非線形関数推定アルゴリズムとそのインプリメンテーションの手引きー

2004.2.12

■ 計算のフロー

実験データその１（濃度の低いケース：10 ppm程度のフルスケールと想定された場合）

```
data = {{1, 0.26}, {2, 0.31}, {3, 0.35}, {4, 0.40},
   {5, 0.45}, {6, 0.49}, {7, 0.53}, {8, 0.56}, {9, 0.60}, {10, 0.62}};
```

関数を定義する。

```
F[{a_, b_,c_}, x_] = a(1-Exp[-b x])+c
```

```
c + a (1 - ℯ^(-b x))
```

Jacobianの計算。実際には最後の結果である、{1 - ℯ^(-b x), a ℯ^(-b x) x, 1} だけを用いる。

```
JacobianMatrix[funs_List, vars_List] :=
    Outer[D, funs, vars]

dF[{a_, b_,c_}, x_] =
    JacobianMatrix[{F[{a,b,c}, x]}, {a,b,c}][[1]]
```

```
{1 - ℯ^(-b x), a ℯ^(-b x) x, 1}
```

非線形推定計算。この例の場合、4回のイタレーションで結果が得られた。

```
NonLinearRegression[data, F, dF, {0.8,0.1, 0.1}]
```

```
Iteration             Parameters

    1             {0.819052, 0.0681907, 0.201943}

    2             {0.893859, 0.0642488, 0.200478}

    3             {0.894443, 0.064593, 0.200398}

    4             {0.894575, 0.0645793, 0.200404}

Convergence obtained

{0.894575, 0.0645793, 0.200404}
```

得られた{a, b, c} である。

```
{0.894575, 0.0645793, 0.200404}
```

■ 最小二乗法によるパラメータ推定

ルーチンlinearRegressionは、Gauss-Newton アルゴリズムの中で、関数の新しいパラメータセットθ 1を推定するための古いパラメータセットθ 0からの増分を計算する。

$\theta_1 = \theta_0 + (X_0^T X_0)(X_0^T X_0)^{-1} X_0^T$ e

このルーチンは別途用意されている必要がある。

```
LinearRegression[dF_,beta_,x_,dy_]:=
(Z=Map[dF[beta,#]&,x];
    Inverse[Transpose[Z].Z].Transpose[Z].dy)
```

以下は途中計算例であり、逆行列ルーチンを別途用意した場合に、計算結果が同等であるかどうかを検証するために用いることができる。

```
Map[F[beta,#]&,x]
```

```
{0.17613, 0.245015, 0.307345, 0.363744,
 0.414775, 0.460951, 0.502732, 0.540537, 0.574744, 0.605696}
```

```
beta
```

```
{0.8, 0.1, 0.1}
```

```
dy
```

```
{0.0838699, 0.0649846, 0.0426546, 0.036256, 0.0352245,
 0.0290493, 0.0272682, 0.0194632, 0.0252557, 0.0143036}
```

```
Z//MatrixForm
```

$$\begin{pmatrix} 0.0951626 & 0.72387 & 1 \\ 0.181269 & 1.30997 & 1 \\ 0.259182 & 1.77796 & 1 \\ 0.32968 & 2.14502 & 1 \\ 0.393469 & 2.42612 & 1 \\ 0.451188 & 2.6343 & 1 \\ 0.503415 & 2.78088 & 1 \\ 0.550671 & 2.87571 & 1 \\ 0.59343 & 2.9273 & 1 \\ 0.632121 & 2.94304 & 1 \end{pmatrix}$$

```
Transpose[Z].Z//MatrixForm
```

$$\begin{pmatrix} 1.88457 & 10.1985 & 3.98959 \\ 10.1985 & 56.0614 & 22.5442 \\ 3.98959 & 22.5442 & 10. \end{pmatrix}$$

```
Inverse[Transpose[Z].Z]//MatrixForm
```

$$\begin{pmatrix} 62.6343 & -14.4023 & 7.4802 \\ -14.4023 & 3.50262 & -2.15045 \\ 7.4802 & -2.15045 & 1.96373 \end{pmatrix}$$

■ 非線形推定ルーチンのアルゴリズム

```
maxHalvings=10;
tolerance=.00001;

NonLinearRegression[data_, F_, dF_, guess_] := (

  presse = -1;
  beta = guess;
  n = 0;
  niter = 0;

  {x, y} = Transpose[data];
  Print["Iteration              Parameters"];


  (*while ループの開始*)
  While[True, dy = y - Map[F[beta, #] &, x];
   (* Map[F[beta,#]&,x] とは,
    関数Fの係数にbetaを適用してから，xのリストをこれに適用してyを求める作業 *)

   sse = Apply[Plus, dy^2]; (* dy^2 の総和を sseとする *)

   (* if 文の構文は If[condition, true, false] である*)
   If[presse < 0 || sse < presse,
    (*error did not increase*)

    (If[1 - sse / presse ≤ tolerance,
      (*error stabilized*)
      (Print["Convergence obtained"]; (*true*)
       Return[beta])
      , (*error decreased*)
      (presse = sse;
       dbeta = LinearRegression[dF, beta, x, dy];
       (*Return[];*)
       beta += dbeta; n = 0; niter++;

       Print["    ", niter, "              ", beta])]),

    (*error increased*)
    dbeta /= 2; beta -= dbeta; n++;
        If[n > maxhalvings,
         Print["Convergence not obtained after ", maxhalvings " halvings"];
         Print["Final parameters = ", beta];
         Return[Null],

     (*else*)
     Print["Halving... ", n, "       ", beta];]]]);
```

■ 参考

Map関数の意味は以下の通り。
Gなる関数は二つの引数を持つと定義する。

```
G[p_,w_]:=1+p w;
```

Gにrとjを与えれば、定義した関数が得られる。

```
G[r,j]
```

```
1 + j r
```

上記の例ではjは単独であったが、wというリストを与えれば、定義した関数のリストが得られる。

```
w={1,2,3};
```

```
Map[G[r,#]&,w]
```

```
{1 + r, 1 + 2 r, 1 + 3 r}
```