1. 読み込んだデータの線形及び非線形近似

1.1 非線形最小二乗法によるデータ近似

"TestData1.csv" が "c:/data/"フォルダに保存されているとします。
与えられたデータを、線形最小二乗法を使い、曲線近似します。

```
> restart;
```

データ点:

```
> csvFile := "c:/data/TestData1.csv";
> Data := ImportMatrix(csvFile, source=csv);
> Data1:=convert(Data,listlist):
```

データ点のプロット:

```
> points:=plots[pointplot](Data1): points;
> with(CurveFitting):
```

1次の多項式で近似:

```
> eq:=LeastSquares(Data1,x,curve=a*x+b);
> CURVE:=plot(eq ,x=0..35):
> plots[display](points,CURVE);
```

２次の多項式で近似:

```
> eq:=LeastSquares(Data1,x,curve=a*x^2+b*x+c);
> CURVE:=plot(eq ,x=0..10):
> plots[display](points,CURVE);
```

３次の多項式で近似:

```
> eq:=LeastSquares(Data1,x,curve=a*x^3+b*x^2+c*x+d);
> CURVE:=plot(eq ,x=0..10):
> plots[display](points,CURVE);
```

非線形式での近似:

```
> eq:=evalf(LeastSquares(Data1,x,curve=a+b*exp(x)));
> CURVE:=plot(eq ,x=0..10):
> plots[display](points,CURVE);
```

1.2 非線形最小二乗法による線形近似

この例では、与えられたデータを非線形多項式で近似を行います。
ここでは、Mapleコマンドを使い、近似を行います。

```
> restart;
> with(LinearAlgebra):
```

関数の定義
以下のプロシージャを定義し実行します。

【アルゴリズムについて】
ここでは、パラメータを(a, b)の２つとした簡単な例で説明します。
パラメータの真値を(a0, b0)、初期値を(a0+da1, b0+db1)とします。
データの各点での関数を f_i(a,b) とし、真値でのテイラー展開を利用すると、

$$f_i(a0+da1, b0+db1) \fallingdotseq f_i(a0,b0) + \left(\frac{\partial}{\partial a} f_i(a0,b0)\right) da1 + \left(\frac{\partial}{\partial b} f_i(a0,b0)\right) db1$$

これらの線形式に 線形最小二乗法(leastsqrs コマンド)を使って、da1 と da2 を求めます。
この計算で求まる新しいパラメータ値(a0+da1, b0+db1)を新しい初期値として、繰り返し近似計算を行います。

```
> nonlinfit:=proc(Lx::list(numeric),Ly::list(numeric),eq::name=
  algebraic,x::name,initparam::set(name=numeric),u::{identical
  (iter)=posint,name},v::evaln)
  local y,n,fx,f,j,N,param,param2,p,S,dp,i,k,pnum,Ls,parseq,df,
  chg,MAXSTEP;
  global maxstep;
  n:=nops(Lx); if nops(Ly)<>n then ERROR(`The lists must have
  equal length`) fi;
  y:=lhs(eq);
  fx:=rhs(eq);
  param:= op(select(type,indets(fx) minus {x},name));
  param2:=map( t->lhs(t),initparam);
  if {param} minus param2 <> {} then ERROR(`Provide starting
  values for all parameters`) fi;
  pnum:=nops([param]);
  f:=unapply(fx,param,x);
  if nargs>5 and type(args[6],identical(iter)=posint) then N:=rhs
  (u) else N:=5 fi;

  for k to pnum do df[k]:=D[k](f); od;
  p[0]:=op(subs(initparam,[param]));
  for j from 0 to N do
  S:={seq( f(p[j],Lx[i])+add(df[k](p[j],Lx[i])*dp[k],k=1..pnum)=Ly
  [i],i=1..n)};

  #Ls:=linalg[leastsqrs](S,{seq(dp[k],k=1..pnum)});
  Ls:=LeastSquares(S,{seq(dp[k],k=1..pnum)});

  for k in Ls do if type(rhs(k),name) then Ls:=Ls minus {k} union
  {lhs(k)=0} fi od;
  if type(maxstep,procedure) then MAXSTEP:=signum*(maxstep@abs)
  else MAXSTEP:=s->s fi;
  # maxstep determines the absolute value of the damping to be
  used
  chg:=map(MAXSTEP,subs(Ls,[seq(dp[k],k=1..pnum)]));

  p[j+1]:=op(chg+[p[j]]);
  od;

  parseq:=seq( zip( (s,t)->s=t,[param],[p[j]]),j=0..N);
  if nargs>5 and type(args[nargs],name) then if nargs=6 then u:=
  parseq else v:=parseq; fi; fi;
  subs(parseq[N+1],eq);
  end:
```

データ点

```
> xdata := [0.47, 0.58, 1., 1.66, 3., 5.29]:
> ydata := [0.7, 0.43, 0.25, 0.14, 0.05, 0.03]:
> points:= plots[pointplot](zip((x,y)->[x,y],xdata,ydata)):
  points;
```

非線形最小二乗近似

上で定義したプロシージャを使い、データ近似を行います。
与える引数は左から順に、[データ点のx座標、データ点のy座標、近似式、近似式の独立変数、パラメータとその初期値、近似計算を繰り返す数、パラメータの履歴を保存する変数]です。

```
> nonlinfit(xdata,ydata,y=c/exp(a*x)+b,x,{a=1, b=1,c=1},iter=5,sq)
  ;
> assign(%);
```

sqから データの収束状態を確認します。

```
> sq;
```

結果のプロット

プロットにより、近似結果を確認します。
ブルーの線は、a, b, c の初期値でのプロット、赤の線が近似結果のプロットです。

```
> p0:=plot(subs({a=1, b=1,c=1}, c/exp(a*x)+b), x=0..6, color=blue)
  :
  p1:=plot(y,x=0..6, color=red):
> plots[display](p0,p1,points, title="非線形最小二乗法による近似結果
  ");
```

StatisticsパッケージのNonlinearFit関数による近似

```
> Statistics[NonlinearFit]((c/exp(a*x) + b), xdata, ydata, x);
> y := %;
```

結果のプロット

プロットにより、近似結果を確認します。

```
> points := plots[pointplot](zip((x, y) → [x, y], xdata, ydata)) :
      points:
> pf := plot(y, x=0..6, color=red) :

> plots[display](p0, pf, points, title
      ="非線形NonlinearFit関数による近似結果");
```