Fitting Data with Different Error Models

Béla Paláncz

A maximum likelihood estimator has been applied to find regression parameters of a straight line in case of different error models. Assuming Gaussian-type noise for the measurement errors, explicit results for the parameters can be given employing *Mathematica*. In the case of the ordinary least squares (OLS_y), total least squares (TLS), and least geometric mean deviation (LGMD) approaches, as well as the error model of combining ordinary least squares (OLS_x and OLS_y) in the Pareto sense, simple formulas are given to compute the parameters via a reduced Gröbner basis. Numerical examples illustrate the methods, and the results are checked via direct global minimization of the residuals.

Introduction

Generally, to carry out a regression procedure one needs to have a model $\mathcal{M}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = 0$, an error definition $e_{\mathcal{M}}(\mathbf{x}, \mathbf{y} : \boldsymbol{\theta})$, and the probability density function of the error $\mathrm{PDF}(e_{\mathcal{M}}(\mathbf{x}, \mathbf{y} : \boldsymbol{\theta}))$. Considering the set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ as measurement points, the maximum likelihood approach aims at finding the parameter vector $\boldsymbol{\theta}$ that maximizes the likelihood of the joint error distribution. Assuming that the measurement errors are independent, we should maximize (see eg. [1])

$$\mathrm{PDF}(e_{\mathcal{M}}((x_1, y_1) : \boldsymbol{\theta}),\ e_{\mathcal{M}}((x_2, y_2) : \boldsymbol{\theta}),\ \ldots,\ e_{\mathcal{M}}((x_n, y_n) : \boldsymbol{\theta})) = \prod_{i=1}^{n} \mathrm{PDF}(e_{\mathcal{M}}((x_i, y_i) : \boldsymbol{\theta})). \tag{1}$$

Instead of maximizing this objective, we minimize

$$\mathcal{L}(\boldsymbol{\theta}) \overset{\text{def}}{=} -\ln\left(\prod_{i=1}^{n} \mathrm{PDF}(e_{\mathcal{M}}((x_i, y_i) : \boldsymbol{\theta}))\right) = -\sum_{i=1}^{n} \ln(\mathrm{PDF}(e_{\mathcal{M}}((x_i, y_i) : \boldsymbol{\theta}))). \tag{2}$$

Consider the Gaussian-type error distribution as $\mathcal{N}(0, \sigma)$; then our estimator is

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \ln\left(\frac{e^{-\frac{e_{\mathcal{M}}((x_i,y_i):\boldsymbol{\theta})^2}{2\sigma^2}}}{\sqrt{2\pi}\,\sigma}\right) = n \ln\left(\sqrt{2\pi}\,\sigma\right) + \sum_{i=1}^{n} \frac{e_{\mathcal{M}}((x_i, y_i):\boldsymbol{\theta})^2}{2\sigma^2}. \tag{3}$$

In our case the model is a line,

$$\mathcal{L}(\boldsymbol{m}, \boldsymbol{b}) \overset{\text{def}}{=} \sum_{i=1}^{n} \frac{(e_{\mathcal{M}}((x_i, y_i):(\boldsymbol{m}, \boldsymbol{b})))^2}{2\sigma^2}. \tag{4}$$

It can be seen that (in the case of Gaussian-type measurement noise) only the type of the error model determines the parameter values, since we should always minimize the least squares of the errors. There are different error models, which can be applied to fitting a line in a least-squares sense. The error model frequently employed, assuming an error-free independent variable x, is the ordinary least squares model (OLS$_y$)

$$e_{\mathcal{M}}((x, y):(\boldsymbol{m}, \boldsymbol{b})) = y - \boldsymbol{m}\,x - \boldsymbol{b}. \tag{5}$$

Similarly, one may also consider an error-free dependent variable y. Then the error model (OLS$_x$) is

$$e_{\mathcal{M}}((x, y):(\boldsymbol{m}, \boldsymbol{b})) = x - \frac{-\boldsymbol{b} + y}{\boldsymbol{m}}. \tag{6}$$

These approaches are called the *algebraic approach*.

Another error model considers the geometrical distance between the data point and the line to be fitted. This type of fitting is also known as *orthogonal regression*, since the distances of the sample points from the line are evaluated by computing the orthogonal projection of the measurements on the line itself. The error in this case [2] is

$$e_{\mathcal{M}}((x, y):(\boldsymbol{m}, \boldsymbol{b})) = \frac{y - \boldsymbol{m}\,x - \boldsymbol{b}}{\sqrt{1 + \boldsymbol{m}^2}}. \tag{7}$$

This *geometrical approach* or *total least squares* (TLS) approach can also be considered as an optimization problem with constraints; namely, one should minimize the errors in both variables [3]:

$$\sum_{i=1}^{n} \Delta x_i^2 + \Delta y_i^2 \to \min \tag{8}$$

under the conditions

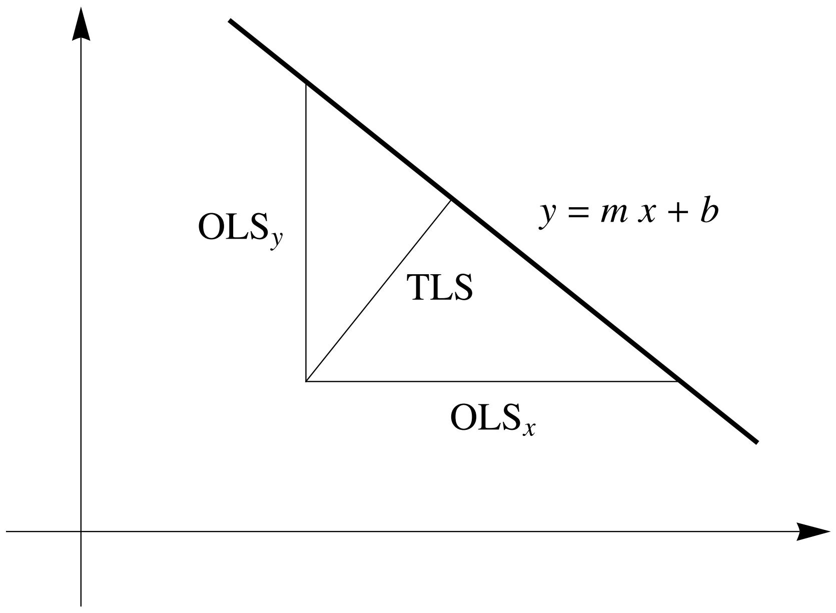
$$-\boldsymbol{b} - \boldsymbol{m}(x_i + \Delta x_i) + (y_i + \Delta y_i) = 0,\ i = 1, 2, \ldots, n. \tag{9}$$

In addition, one can also combine OLS_x and OLS_y to construct an error model. The first possibility is to consider the geometric mean of these two types of errors,

$$e_{\mathcal{M}}((x, y) : (m, b)) = \sqrt{(-b - m\,x + y)\left(x - \frac{-b + y}{m}\right)}\,. \tag{10}$$

These error models are illustrated in Figure 1.

```
Graphics[{
  Arrow[{{-1, 0}, {10, 0}}],
  Arrow[{{0, -1}, {0, 7}}],
  Line[{{3, 2}, {8, 2}}],
  Line[{{3, 2}, {3, 6}}],
  {Thick, Line[{{3, 6} - .2 {5, -4}, {8, 2} + .2 {5, -4}}]},
  Line[{{3, 2}, {3, 2} + {80 / 41, 100 / 41}}],
  Style[Text["OLS"Style["x",Italic], ({3, 2} + {8, 2}) / 2, {0, 2}],
   16],
  Style[Text["OLS"Style["y",Italic], ({3, 2} + {3, 6}) / 2,
    {1.5, 0}], 16],
  Style[Text["TLS", (2 {3, 2} + {80 / 41, 100 / 41}) / 2,
    {-1.8, 0}], 16],
  Style[
   Text[Row[{Style["y", Italic], " = ", Style["m", Italic],
      " ", Style["x", Italic], " + ", Style["b", Italic] }],
    ({3, 6} + {8, 2}) / 2, {-1.5, -1}], 16]
 }
]
```



▲ **Figure 1.** The different error models in the case of fitting a straight line.

This model is also called the *least geometric mean deviation* approach or LGMD model (see [4)]. As a second possibility, one may consider OLS$_x$ and OLS$_y$ as competing functions of the parameters and find their Pareto-front representing a set of optimal solutions for the parameters $\boldsymbol{\theta}$. Since this multi-objective problem is convex, the objective can be expressed as a linear combination of these error functions, namely

$$e_{\mathcal{M}}((x_i, y_i) : (m, b)) = \lambda\,(-b - m\,x + y) + (1 - \lambda)\left(x - \frac{-b + y}{m}\right), \tag{11}$$

where λ is a parameter, $0 \le \lambda \le 1$, and the set of optimal solutions of the parameters belonging to the Pareto-front is $\boldsymbol{\theta} = \boldsymbol{\theta}(\lambda)$. You can choose the value of λ depending on your trade-off preference between OLS$_x$ and OLS$_y$ [5].

■ Application of Symbolic Computation

□ `SuperLog` Function

Symbolic computation can be used to avoid direct minimization and to get an explicit formula for the estimated parameters. We apply the *Mathematica* function `SuperLog` developed in [6], which uses pattern matching that enhances *Mathematica*'s ability to simplify expressions involving the natural logarithm of a product of algebraic terms.

```
SuperLog[Q_] := Module[
{erk},
  Which[
   Q === On,
   Unprotect[Log];
   Clear[Log];
   Log[Product[x_, {k_. a_, b_}]] :=
    Log[Product[Times[erk, x], {k, a, b}]] /. erk → 1;
   Log[Product[HoldPattern[Times[x__]], {k_, a_, b_}]] :=
    Simplify[
     Map[Sum[#, {k, a, b}] &,
       Plus @@ Map[Expand[PowerExpand[Log[#]]] &,
         List[x]]] //.
      Sum[u_, w_, {kk_, aa_, bb_}] :→
       u Sum[w, {kk, aa, bb}] /; FreeQ[u, kk] == True];
   Protect[Log];
   Print["SuperLog is now On."],

   Q === Off,
   Unprotect[Log]; Clear[Log]; Protect[Log];
   Print["SuperLog is now Off."],

   True,
   Print["Please use SuperLog[On] or SuperLog[Off]!"]
  ]]
```

Let us activate this function.

```
SuperLog[On]
```

```
SuperLog is now On.
```

Then this is the ML estimator for Gaussian-type noise.

$$\mathcal{L} = \prod_{i=1}^{n} \mathrm{PDF}[\mathrm{NormalDistribution}[0, \sigma], e_{\mathcal{M}_i}]$$

$$\prod_{i=1}^{n} \frac{e^{-\frac{(e_{\mathcal{M}})_i^2}{2\sigma^2}}}{\sqrt{2\pi}\,\sigma}$$

□ Ordinary Least Squares (OLS_y)

Now let us consider the OLS_y problem.

```
Logℒ[OLSy] = Log[ℒ /. e𝓜i → yi - m xi - b] // Simplify
```

$$-\frac{b^2 n}{2\sigma^2}-\frac{1}{2}n\,\text{Log}[2]-\frac{1}{2}n\,\text{Log}[\pi]-n\,\text{Log}[\sigma]+\sum_{i=1}^{n}-\frac{b\,m\,x_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{m^2 x_i^2}{2\sigma^2}+\sum_{i=1}^{n}\frac{b\,y_i}{\sigma^2}+\sum_{i=1}^{n}\frac{m\,x_i\,y_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{y_i^2}{2\sigma^2}$$

Here are the necessary conditions for the optimum.

```
eq1[OLSy] = D[Logℒ[OLSy], m] == 0
```

$$\sum_{i=1}^{n}-\frac{b\,x_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{m\,x_i^2}{\sigma^2}+\sum_{i=1}^{n}\frac{x_i\,y_i}{\sigma^2}=0$$

```
eq2[OLSy] = D[Logℒ[OLSy], b] == 0
```

$$-\frac{b\,n}{\sigma^2}+\sum_{i=1}^{n}-\frac{m\,x_i}{\sigma^2}+\sum_{i=1}^{n}\frac{y_i}{\sigma^2}=0$$

Let us introduce the following constants:

$$a=\sum_{i=1}^{n}x_i, \tag{12}$$

$$b=\sum_{i=1}^{n}x_i^2, \tag{13}$$

$$c=\sum_{i=1}^{n}y_i, \tag{14}$$

$$d=\sum_{i=1}^{n}y_i^2, \tag{15}$$

$$e=\sum_{i=1}^{n}x_i\,y_i. \tag{16}$$

In those terms, here are the necessary conditions for the optimum.

```
Eq1[OLSy] = -a b - b m + e == 0
```

$$e - a\,b - b\,m == 0$$

```
Eq2[OLSy] = -b n - a m + c == 0
```

$$c - a\,m - b\,n == 0$$

Then this is the optimal solution of the parameters.

```
solmbG = Solve[{Eq1[OLSy], Eq2[OLSy]}, {m, b}] // Flatten
```

$$\left\{m \to -\frac{-a\,c + e\,n}{a^2 - b\,n},\ b \to -\frac{b\,c - a\,e}{a^2 - b\,n}\right\}$$

◻ Total Least Squares (TLS)

Although the equation system for the parameters of OLS$_y$ is linear, for other error models we get a multivariable algebraic system. Now consider the TLS problem. Here is the maximum likelihood function.

$$\text{Log}\mathcal{L}[\text{TLS}] = \text{Log}\left[\mathcal{L}\ /.\ e_{\mathcal{M}_i} \text{->} \frac{y_i - m\,x_i\ - b}{\sqrt{1 + m^2}}\right]\ //\ \text{Simplify}$$

$$-\frac{b^2\,n}{2\left(1 + m^2\right)\sigma^2} - \frac{1}{2}\,n\,\text{Log}[2] - \frac{1}{2}\,n\,\text{Log}[\pi] - n\,\text{Log}[\sigma] + \sum_{i=1}^{n} -\frac{b\,m\,x_i}{\left(1 + m^2\right)\sigma^2} + \sum_{i=1}^{n} -\frac{m^2\,x_i^2}{2\left(1 + m^2\right)\sigma^2} + \sum_{i=1}^{n} \frac{b\,y_i}{\left(1 + m^2\right)\sigma^2} + \sum_{i=1}^{n} \frac{m\,x_i\,y_i}{\left(1 + m^2\right)\sigma^2} + \sum_{i=1}^{n} -\frac{y_i^2}{2\left(1 + m^2\right)\sigma^2}$$

Therefore here is the equation system to be solved.

```
eq1[TLS] = D[Logℒ[TLS], m] == 0 // Simplify
```

$$\frac{b^2 m n}{(1+m^2)^2 \sigma^2} + \sum_{i=1}^{n} \left(\frac{2 b m^2 x_i}{(1+m^2)^2 \sigma^2} - \frac{b x_i}{(1+m^2) \sigma^2} \right) + \sum_{i=1}^{n} \left(\frac{m^3 x_i^2}{(1+m^2)^2 \sigma^2} - \frac{m x_i^2}{(1+m^2) \sigma^2} \right) + \sum_{i=1}^{n} - \frac{2 b m y_i}{(1+m^2)^2 \sigma^2} + \sum_{i=1}^{n} \frac{m y_i^2}{(1+m^2)^2 \sigma^2} + \sum_{i=1}^{n} \left(- \frac{2 m^2 x_i y_i}{(1+m^2)^2 \sigma^2} + \frac{x_i y_i}{(1+m^2) \sigma^2} \right) == 0$$

```
eq2[TLS] = D[Logℒ[TLS], b] == 0 // Simplify
```

$$\sum_{i=1}^{n} - \frac{m x_i}{(1+m^2) \sigma^2} + \sum_{i=1}^{n} \frac{y_i}{(1+m^2) \sigma^2} == \frac{b n}{(1+m^2) \sigma^2}$$

Since $1 + m^2 \neq 0$, the conditions are as follows.

```
Eq1[TLS] =
  b^2 m n + a 2 b m^2 - a (1 + m^2) b + b m^3 - b m (1 + m^2) - 2 c b m +
    d m - 2 e m^2 + e (1 + m^2) == 0
```

$$d\,m - 2\,c\,b\,m - 2\,e\,m^2 + 2\,a\,b\,m^2 + b\,m^3 + e\,(1+m^2) - a\,b\,(1+m^2) - b\,m\,(1+m^2) + b^2 m n == 0$$

```
Eq2[TLS] = -b n - a m + c == 0
```

$$c - a\,m - b\,n == 0$$

A Gröbner basis solves this system, eliminating *b*.

```
gbm[TLS] = GroebnerBasis[{Eq1[TLS], Eq2[TLS]}, {m, b}, {b}] //
  Simplify
```

$$\{a^2 m - c^2 m + a\,c\,(-1+m^2) + (e - b\,m + d\,m - e\,m^2)\,n\}$$

```
solm[TLS] = Solve[gbm[TLS] == 0, m] // Flatten
```

$$\left\{m \to \frac{1}{2\ (\mathrm{a}\,\mathrm{c} - \mathrm{e}\, n)} \left(-\mathrm{a}^2 + \mathrm{c}^2 + \mathrm{b}\, n - \mathrm{d}\, n - \sqrt{\left(\mathrm{a}^2 - \mathrm{c}^2 - \mathrm{b}\, n + \mathrm{d}\, n\right)^2 - 4\ (\mathrm{a}\,\mathrm{c} - \mathrm{e}\, n)\ (-\mathrm{a}\,\mathrm{c} + \mathrm{e}\, n)}\right),\right.$$
$$\left. m \to \frac{1}{2\ (\mathrm{a}\,\mathrm{c} - \mathrm{e}\, n)} \left(-\mathrm{a}^2 + \mathrm{c}^2 + \mathrm{b}\, n - \mathrm{d}\, n + \sqrt{\left(\mathrm{a}^2 - \mathrm{c}^2 - \mathrm{b}\, n + \mathrm{d}\, n\right)^2 - 4\ (\mathrm{a}\,\mathrm{c} - \mathrm{e}\, n)\ (-\mathrm{a}\,\mathrm{c} + \mathrm{e}\, n)}\right)\right\}$$

Since the second equation is linear, it is reasonable to compute ***m*** first, then ***b***.

```
solb[TLS] = Solve[Eq2[TLS], b] // Flatten
```

$$\left\{b \to \frac{\mathrm{c} - \mathrm{a}\, m}{n}\right\}$$

◻ Least Geometric Mean Deviation (LGMD)

The LGMD error model also leads to a second-order polynomial equation system. Now here is the ML estimator.

```
Log𝓛[LGMD] = Log[𝓛 /. e_M_i → Sqrt[(-b - m x_i + y_i) (x_i - (-b + y_i)/m)]] //
  Simplify
```

$$\frac{b^2 n}{2 m \sigma^2} - \frac{1}{2} n \operatorname{Log}[2] - \frac{1}{2} n \operatorname{Log}[\pi] - n \operatorname{Log}[\sigma] +$$
$$\sum_{i=1}^{n} \frac{b x_i}{\sigma^2} + \sum_{i=1}^{n} \frac{m x_i^2}{2 \sigma^2} + \sum_{i=1}^{n} -\frac{b y_i}{m \sigma^2} + \sum_{i=1}^{n} -\frac{x_i y_i}{\sigma^2} + \sum_{i=1}^{n} \frac{y_i^2}{2 m \sigma^2}$$

Consequently, here is the system to be solved for the parameters.

```
eq1[LGMD] = D[Logℒ[LGMD], m] == 0 // Simplify
```

$$\sum_{i=1}^{n} \frac{x_i^2}{2\,\sigma^2} + \sum_{i=1}^{n} \frac{b\,y_i}{m^2\,\sigma^2} + \sum_{i=1}^{n} -\frac{y_i^2}{2\,m^2\,\sigma^2} == \frac{b^2\,n}{2\,m^2\,\sigma^2}$$

```
eq2[LGMD] = D[Logℒ[LGMD], b] == 0
```

$$\frac{b\,n}{m\,\sigma^2} + \sum_{i=1}^{n} \frac{x_i}{\sigma^2} + \sum_{i=1}^{n} -\frac{y_i}{m\,\sigma^2} == 0$$

Assume $m \neq 0$.

```
Eq1[LGMD] = -b^2 n + m^2 b + 2 b c - d == 0
```

$$-\mathrm{d} + 2\,\mathrm{c}\,b + \mathrm{b}\,m^2 - b^2\,n == 0$$

```
Eq2[LGMD] = b n + m a - c == 0
```

$$-\mathrm{c} + \mathrm{a}\,m + b\,n == 0$$

Again a Gröbner basis gives a second-order system.

```
gbm[LGMD] =
 GroebnerBasis[{Eq1[LGMD], Eq2[LGMD]}, {m, b}, {b}] //
  Simplify
```

$$\left\{\mathrm{c}^2 - \mathrm{a}^2\,m^2 - \mathrm{d}\,n + \mathrm{b}\,m^2\,n\right\}$$

```
solmT = Solve[gbm[LGMD] == 0, m] // Flatten
```

$$\left\{m \to -\frac{\sqrt{\mathrm{c}^2 - \mathrm{d}\,n}}{\sqrt{\mathrm{a}^2 - \mathrm{b}\,n}},\ m \to \frac{\sqrt{\mathrm{c}^2 - \mathrm{d}\,n}}{\sqrt{\mathrm{a}^2 - \mathrm{b}\,n}}\right\}$$

When m is known, the other parameter can be computed.

```
solbT = Solve[Eq2[LGMD], b] // Flatten
```

$$\left\{b \to \frac{\mathrm{c} - \mathrm{a}\,m}{n}\right\}$$

□ Pareto Approach

In the case of the Pareto approach, the system is already fourth order.

```
LogL[Pareto] =
  Log[L /. {e_{M_i} → Sqrt[λ (-b - m x_i + y_i)^2 + (1 - λ) (x_i - (-b + y_i)/m)^2]}] //
   Simplify
```

$$-\frac{b^2 n}{2 m^2 \sigma^2}-\frac{b^2 n \lambda}{2 \sigma^2}+\frac{b^2 n \lambda}{2 m^2 \sigma^2}-\frac{1}{2} n \operatorname{Log}[2]-\frac{1}{2} n \operatorname{Log}[\pi]-n \operatorname{Log}[\sigma]+$$
$$\sum_{i=1}^{n}-\frac{b x_i}{m \sigma^2}+\sum_{i=1}^{n}\frac{b \lambda x_i}{m \sigma^2}+\sum_{i=1}^{n}-\frac{b m \lambda x_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{x_i^2}{2 \sigma^2}+\sum_{i=1}^{n}\frac{\lambda x_i^2}{2 \sigma^2}+$$
$$\sum_{i=1}^{n}-\frac{m^2 \lambda x_i^2}{2 \sigma^2}+\sum_{i=1}^{n}\frac{b y_i}{m^2 \sigma^2}+\sum_{i=1}^{n}\frac{b \lambda y_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{b \lambda y_i}{m^2 \sigma^2}+\sum_{i=1}^{n}\frac{x_i y_i}{m \sigma^2}+$$
$$\sum_{i=1}^{n}-\frac{\lambda x_i y_i}{m \sigma^2}+\sum_{i=1}^{n}\frac{m \lambda x_i y_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{y_i^2}{2 m^2 \sigma^2}+\sum_{i=1}^{n}-\frac{\lambda y_i^2}{2 \sigma^2}+\sum_{i=1}^{n}\frac{\lambda y_i^2}{2 m^2 \sigma^2}$$

Here is the system.

```
eq1[Pareto] = D[LogL[Pareto], m] == 0 // Simplify
```

$$\frac{b^2 n}{m^3 \sigma^2}+\sum_{i=1}^{n}\frac{b x_i}{m^2 \sigma^2}+\sum_{i=1}^{n}-\frac{b \lambda x_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{b \lambda x_i}{m^2 \sigma^2}+$$
$$\sum_{i=1}^{n}-\frac{m \lambda x_i^2}{\sigma^2}+\sum_{i=1}^{n}-\frac{2 b y_i}{m^3 \sigma^2}+\sum_{i=1}^{n}\frac{2 b \lambda y_i}{m^3 \sigma^2}+\sum_{i=1}^{n}-\frac{x_i y_i}{m^2 \sigma^2}+$$
$$\sum_{i=1}^{n}\frac{\lambda x_i y_i}{\sigma^2}+\sum_{i=1}^{n}\frac{\lambda x_i y_i}{m^2 \sigma^2}+\sum_{i=1}^{n}\frac{y_i^2}{m^3 \sigma^2}+\sum_{i=1}^{n}-\frac{\lambda y_i^2}{m^3 \sigma^2} == \frac{b^2 n \lambda}{m^3 \sigma^2}$$

```
eq2[Pareto] = D[LogL[Pareto], b] == 0 // Simplify
```

$$\frac{b n \lambda}{m^2 \sigma^2}+\sum_{i=1}^{n}-\frac{x_i}{m \sigma^2}+\sum_{i=1}^{n}\frac{\lambda x_i}{m \sigma^2}+\sum_{i=1}^{n}-\frac{m \lambda x_i}{\sigma^2}+$$
$$\sum_{i=1}^{n}\frac{y_i}{m^2 \sigma^2}+\sum_{i=1}^{n}\frac{\lambda y_i}{\sigma^2}+\sum_{i=1}^{n}-\frac{\lambda y_i}{m^2 \sigma^2} == \frac{b n \left(1+m^2 \lambda\right)}{m^2 \sigma^2}$$

Here is the system in compact form.

```
Eq1[Pareto] =
 b^2 n - b^2 n λ + m b a - m^3 b λ a - m b λ a - m^4 λ b - 2 b c + 2 b λ c -
   m e + m^3 λ e + m λ e + d - d λ == 0
```

$$d-2\,c\,b-e\,m+a\,b\,m+b^2\,n-d\,\lambda+2\,c\,b\,\lambda+e\,m\,\lambda-a\,b\,m\,\lambda+e\,m^3\,\lambda-a\,b\,m^3\,\lambda-b\,m^4\,\lambda-b^2\,n\,\lambda==0$$

```
Eq2[Pareto] =
 -b n - m^2 b n λ + b n λ - m a + m λ a - m^3 λ a + c + m^2 λ c - λ c == 0
```

$$c-a\,m-b\,n-c\,\lambda+a\,m\,\lambda+c\,m^2\,\lambda-a\,m^3\,\lambda+b\,n\,\lambda-b\,m^2\,n\,\lambda==0$$

Here is the Gröbner basis for the first parameter.

```
gbm[Pareto] =
 GroebnerBasis[{Eq1[Pareto], Eq2[Pareto]}, {m, b}, {b}] //
  Simplify
```

$$\left\{\left(1+\left(-1+m^2\right)\lambda\right)\left(d\,n-e\,m\,n+c^2\,(-1+\lambda)+a^2\,m^4\,\lambda-d\,n\,\lambda+e\,m\,n\,\lambda+e\,m^3\,n\,\lambda-b\,m^4\,n\,\lambda-a\,c\,m\left(-1+\lambda+m^2\,\lambda\right)\right)\right\}$$

Assume that $1+\left(-1+m^2\right)\lambda\neq 0$.

```
Gbm = d n - e m n + c^2 (-1 + λ) + a^2 m^4 λ - d n λ + e m n λ + e m^3 n λ -
   b m^4 n λ - a c m (-1 + λ + m^2 λ) // Expand
```

$$-c^2+a\,c\,m+d\,n-e\,m\,n+c^2\,\lambda-a\,c\,m\,\lambda-a\,c\,m^3\,\lambda+a^2\,m^4\,\lambda-d\,n\,\lambda+e\,m\,n\,\lambda+e\,m^3\,n\,\lambda-b\,m^4\,n\,\lambda$$

After solving this polynomial for *m*, the other parameter can be solved from the second equation, which is linear in *b*.

```
solbP = Solve[Eq2[Pareto], b] // Flatten
```

$$\left\{b\to\frac{c-a\,m}{n}\right\}$$

■ Numerical Example

□ Data Samples

Consider some data on rainfall x (in mm) and the resulting groundwater level changes y (in cm) from a landslide along the Ohio River Valley near Cincinnati, Ohio [7].

```
xydata = {{1.94, 2.5}, {3.33, 1.89}, {3.22, 1.67},
   {5.67, 1.31}, {4.72, 1.02}, {3.89, 0.96}, {2.78, 1.1},
   {10.56, 0.15}, {9.44, 3.92}, {12.78, 5.23},
   {14.72, 4.22}, {13.61, 3.63}, {20.39, 4.32},
   {38.89, 5.89}};
```

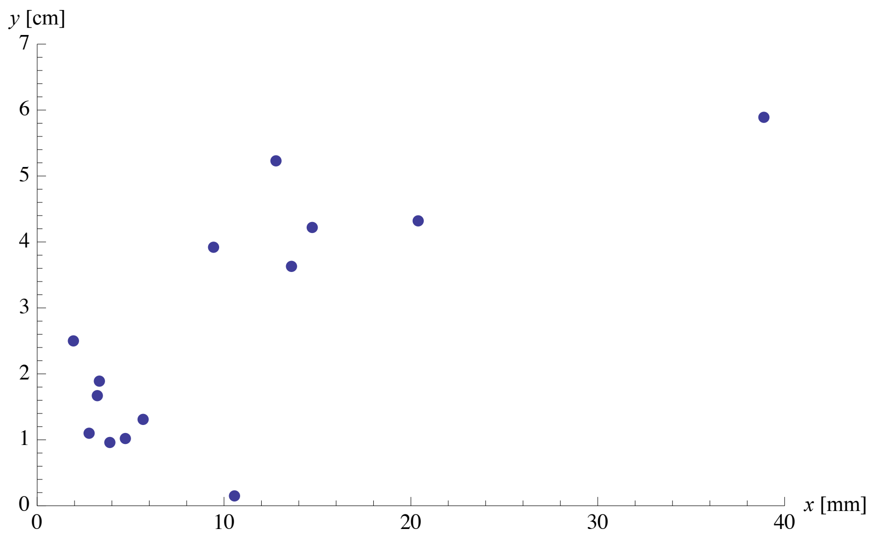
There are 14 measurements.

```
Length[xydata]
```

```
14
```

This displays the measured data.

```
ListPlot[xydata, AxesLabel → {"x [mm]", "y [cm]"},
 PlotStyle → PointSize[0.015], PlotRange → {{0, 40}, {0, 7}}]
```



▲ **Figure 2.** The measured data: rainfall versus water level change in dimensional form.

□ Computation of the Constants for the Equation Systems

The constants a, b, c, d, and e in equations (12) to (16) are needed.

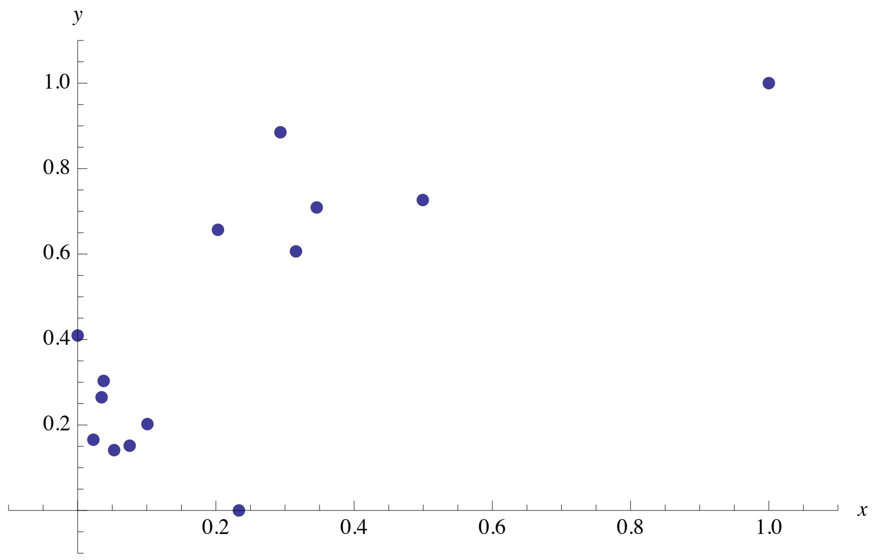
This separates the data.

```
{xdata, ydata} = Transpose[xydata];
```

This transforms the data into dimensionless form.

```
xS = (xdata - Min[xdata]) / (Max[xdata] - Min[xdata]);
yS = (ydata - Min[ydata]) / (Max[ydata] - Min[ydata]);

p2 = ListPlot[Transpose[{xS, yS}], AxesLabel → {x, y},
  PlotStyle → PointSize[0.015],
  PlotRange → {{-0.1, 1.1}, {-0.1, 1.1}}]
```



▲ **Figure 3.** The measured data: rainfall versus water level change in dimensionless form.

Now the constants can be computed.

```
abcden =
 Join[
  Thread[{a, b, c, d, e} →
    Total /@ {xS, xS^2, yS, yS^2, xS yS}], {n → Length[xydata]}
 ]

{a → 3.21461, b → 1.67216,
 c → 6.22125, d → 4.05349, e → 2.25602, n → 14}
```

□ Computation of the Parameters of the Fitted Line

■ OLS_y *Model*

Here are the estimated parameters employing the explicit solutions.

```
solmbG /. abcden
```

```
{m → 0.885963, b → 0.240945}
```

This checks the result.

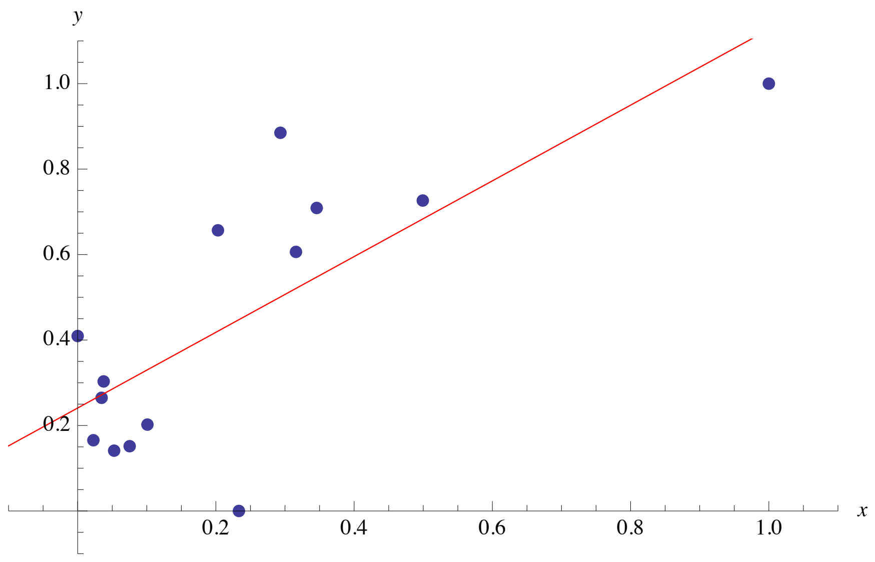
```
Fit[Transpose[{xS, yS}], {1, t}, t]
```

```
0.240945 + 0.885963 t
```

Figure 4 shows the estimated line with the sample points.

```
p3 = Plot[%, {t, -0.1, 1.1}, PlotStyle → Red];
```

```
Show[{p2, p3}]
```



▲ **Figure 4.** The sample points with the line estimated with OLS_y.

■ TLS *Model*

Here are the first and second parameters.

```
solm[TLS] /. abcden
```

```
{m → 1.23716, m → -0.808306}
```

```
solb[TLS] /. solm[TLS] /. abcden
```

```
{b → 0.160305}
```

Here is a check of this result on the basis of the TLS definition. Equation (8) gives the objective function.

```
obj[TLS] = Total[Table[Δxᵢ² + Δyᵢ², {i, Length[xydata]}]]
```

$$\Delta x_1^2 + \Delta x_2^2 + \Delta x_3^2 + \Delta x_4^2 + \Delta x_5^2 + \Delta x_6^2 + \Delta x_7^2 + \Delta x_8^2 + \Delta x_9^2 + \Delta x_{10}^2 + \Delta x_{11}^2 + \Delta x_{12}^2 + \Delta x_{13}^2 + \Delta x_{14}^2 + \Delta y_1^2 + \Delta y_2^2 + \Delta y_3^2 + \Delta y_4^2 + \Delta y_5^2 + \Delta y_6^2 + \Delta y_7^2 + \Delta y_8^2 + \Delta y_9^2 + \Delta y_{10}^2 + \Delta y_{11}^2 + \Delta y_{12}^2 + \Delta y_{13}^2 + \Delta y_{14}^2$$

The constraints are $y_i + \Delta y_i - m\,(x_i + \Delta x_i) - b = 0,\ i = 1, \ldots, n$.

```
(cons = Table[yS[[i]] + Δyᵢ - m (xS[[i]] + Δxᵢ) - b == 0,
    {i, Length[xydata]}]) // Column
```

$0.409408 - b - m\,(0. + \Delta x_1) + \Delta y_1 == 0$
$0.303136 - b - m\,(0.0376184 + \Delta x_2) + \Delta y_2 == 0$
$0.264808 - b - m\,(0.0346414 + \Delta x_3) + \Delta y_3 == 0$
$0.202091 - b - m\,(0.100947 + \Delta x_4) + \Delta y_4 == 0$
$0.151568 - b - m\,(0.0752368 + \Delta x_5) + \Delta y_5 == 0$
$0.141115 - b - m\,(0.052774 + \Delta x_6) + \Delta y_6 == 0$
$0.165505 - b - m\,(0.0227334 + \Delta x_7) + \Delta y_7 == 0$
$0. - b - m\,(0.233288 + \Delta x_8) + \Delta y_8 == 0$
$0.656794 - b - m\,(0.202977 + \Delta x_9) + \Delta y_9 == 0$
$0.885017 - b - m\,(0.293369 + \Delta x_{10}) + \Delta y_{10} == 0$
$0.709059 - b - m\,(0.345873 + \Delta x_{11}) + \Delta y_{11} == 0$
$0.606272 - b - m\,(0.315832 + \Delta x_{12}) + \Delta y_{12} == 0$
$0.726481 - b - m\,(0.499323 + \Delta x_{13}) + \Delta y_{13} == 0$
$1. - b - m\,(1. + \Delta x_{14}) + \Delta y_{14} == 0$

The unknown variables are not only the parameters, but the adjustments (Δx_i, Δy_i) as well.

```
vars = Join[Table[{Δxᵢ, Δyᵢ}, {i, Length[xydata]}], {m, b}] //
  Flatten
```

```
{Δx1, Δy1, Δx2, Δy2, Δx3, Δy3, Δx4, Δy4, Δx5, Δy5,
 Δx6, Δy6, Δx7, Δy7, Δx8, Δy8, Δx9, Δy9, Δx10, Δy10,
 Δx11, Δy11, Δx12, Δy12, Δx13, Δy13, Δx14, Δy14, m, b}
```

This uses a built-in global optimization method. (This takes a long time to compute.)

```
Timing[
 sol = NMinimize[{obj[TLS], cons}, vars,
    Method → {"RandomSearch", "SearchPoints" → 200}];]
```

```
{257.582130, Null}
```

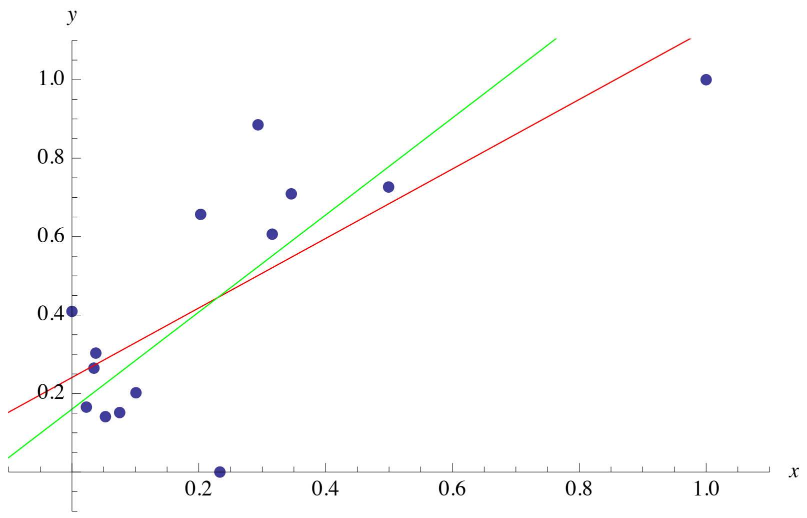
```
Drop[sol[[2]], {1, 2 Length[xydata]}]
```

```
{m → 1.23716, b → 0.160305}
```

The TLS estimation gives a result quite different from the OLS_y model; see Figure 5.

```
p4 = Plot[m t + b /. %, {t, -0.1, 1.1}, PlotStyle → Green];
```

```
Show[{p2, p3, p4}]
```



▲ **Figure 5.** The lines estimated with the OLS_y (red) and TLS (green) models.

Since the constraints are linear, the optimization can be written in unconstrained form, reducing the original number of variables $2n+2$ to $n+2$.

■ LGMD *Model*

Now here is the first parameter.

```
solmT /. abcden // Chop
```

```
{m → -1.17471, m → 1.17471}
```

This uses the result.

```
solbT /. solmT[[2]] /. abcden // Chop
```

```
{b → 0.174644}
```

Here is a numerical check of the objective.

```
obj[LGMD] =
  Total[
   Table[√((-b - m xS[[i]] + yS[[i]])^2 (xS[[i]] - (yS[[i]] - b)/m)^2),
    {i, Length[xydata]}]];
```

$$\text{Table}\left[\sqrt{\left((-b - m\,\text{xS[[i]]} + \text{yS[[i]]})^2 \left(\text{xS[[i]]} - \frac{\text{yS[[i]]} - b}{m}\right)^2\right)},\right.$$

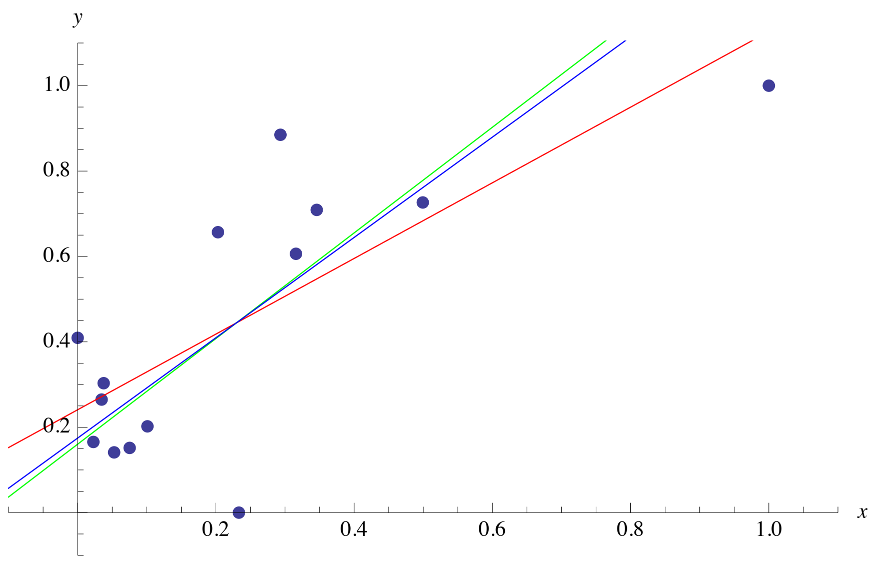
```
solT = NMinimize[obj[LGMD], {m, b}]
```

```
{0.5394, {m → 1.17471, b → 0.174644}}
```

Figure 6 shows this result together with the OLS$_y$ and TLS models.

```
p5 = Plot[m t + b /. solT[[2]], {t, -0.1, 1.1},
   PlotStyle → Blue];
```

```
Show[{p2, p3, p4, p5}]
```



▲ **Figure 6.** The lines estimated with the OLS_y (red), TLS (green), and LGMD (blue) models.

■ *Pareto Approach*

The first parameter is a fourth-order polynomial.

```
Gbm /. abcden
```

```
18.0448 - 11.5853 m - 18.0448 λ +
 11.5853 m λ + 11.5853 m^3 λ - 13.0765 m^4 λ
```

The best trade-off between OLS_y and OLS_x is to let $\lambda = 0.5$.

```
solP = NSolve[Gbm /. abcden /. λ → 1 / 2, m]
```

```
{{m → -1.0602}, {m → 0.404042 - 0.99016 i},
 {m → 0.404042 + 0.99016 i}, {m → 1.13808}}
```

This is the real positive solution.

```
solP[[4]]
```

```
{m → 1.13808}
```

Using this value gives the second parameter.

```
solbP /. solP[[4]] /. abcden // Flatten
```

```
{b → 0.183054}
```

We compute the solution using direct global minimization. Here is the objective.

```
obj[Pareto] =
  Total[
   Table[
    (λ (-b - m xS[[i]] + yS[[i]])^2 +
       (1 - λ) (xS[[i]] - (yS[[i]] - b)/m)^2), {i, Length[xydata]}]];
```

This gives the result.

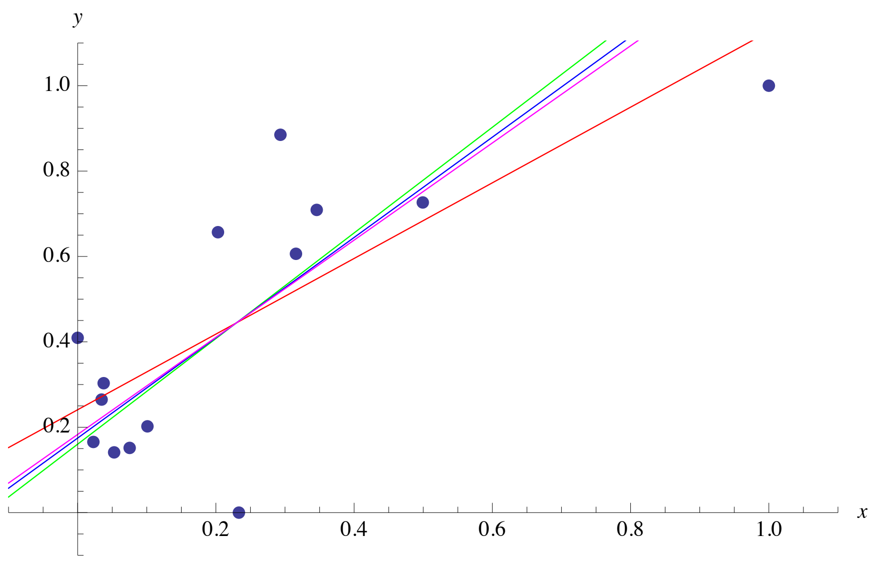
```
solP = NMinimize[{obj[Pareto] /. λ → 1 / 2}, {m, b},
  Method → {"RandomSearch", "SearchPoints" → 200}]
```

```
{0.545029, {m → 1.13808, b → 0.183054}}
```

Figure 7 shows this solution with the results of the other models.

```
p6 = Plot[m t + b /. solP[[2]], {t, -0.1, 1.1},
   PlotStyle → Magenta];
```

```
Show[{p2, p3, p4, p5, p6}]
```



▲ **Figure 7.** The lines estimated with the OLS_y (red), TLS (green), and LGMD (blue) models, and the Pareto approach with $\lambda = 1/2$ (magenta).

Conclusion

The numerical computations show that the formulas developed by an ML estimator via symbolic computation to determine the parameters of a straight line to be fitted provide correct results and require considerably less computation time than the direct methods based on global minimization of the residuals. Our examples also illustrate that the TLS, LGMD, and Pareto approaches give more realistic solutions than the traditional OLS_y, since Figure 7 shows there are at least two outliers in the sample set.

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed., Cambridge: Cambridge University Press, 1992.

[2] M. Zuliani. “RANSAC for Dummies.” (Jan 10, 2014) vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf.

[3] B. Schaffrin, “A Note on Constrained Total Least-Squares Estimation,” *Linear Algebra and Its Applications*, **417**(1), 2006 pp. 245–258. doi:10.1016/j.laa.2006.03.044.

[4] C. Tofallis, “Model Fitting for Multiple Variables by Minimising the Geometric Mean Deviation,” in *Total Least Squares and Errors-in-Variables Modeling: Analysis, Algorithms and Applications* (S. Van Huffel and P. Lemmerling, eds.), Dordrecht: Kluwer, 2002.

[5] B. Paláncz and J. L. Awange, “Application of Pareto Optimality to Linear Models with Errors-in-All-Variables,” *Journal of Geodesy*, **86**(7), 2012 pp. 531–545. doi:10.1007/s00190-011-0536-1.

[6] C. Rose and M. D. Smith, “Symbolic Maximum Likelihood Estimation with *Mathematica*,” *The Statistician*, **49**(2), 2000 pp. 229–240. www.jstor.org/stable/2680972.

[7] W. C. Haneberg, *Computational Geosciences with Mathematica*, Berlin: Springer, 2004.

B. Paláncz, “Fitting Data with Different Error Models,” *The Mathematica Journal*, 2014. dx.doi.org/doi:10.3888/tmj.16-4.

About the Author

Béla Paláncz received his D.Sc. degree in 1993 from the Hungarian Academy of Sciences and has wide-ranging experience in teaching and research (RWTH Aachen, Imperial College London, DLR Köln, and Wolfram Research). His main research fields are mathematical modeling and symbolic-numeric computation.

Béla Paláncz
Department of Photogrammetry and Geoinformatics,
Budapest University of Technology and Economics
1521 Budapest, Hungary
palancz@epito.bme.hu