Text presented in the ACDCA conference "Recent research on DERIVE/TI-92 supported mathematics education" in Goesing (Austria), 25-27.August 1999

Linear discrete least-square fitting assisted by CAS

Adam Marlewski
Institute of Mathematics, Poznañ University of Technology,
ul.Piotrowo 3a, 60-295 Poznañ, Poland; e-mail: amarlew@math.put.poznan.pl

Abstract. Least-square approximation is commonly used technique generating the best fitting to given function or to given set of points. This paper deals with the last case where the task may be reduced to solving a system of linear equations. Besides the polynomial and polynomial-reduced approximations there is considered in details a rational fitting where some traps appear (and for this reason it is called a problematically linearisable fitting). There is discussed the efficient assistance of a computer algebra system to the best fitting tasks, in particular definition of appropriate functions in *DERIVE* are given.

Key words: computer algebra systems, linear algebra, math education

1. General on the least-square fitting

Least-square fitting is one of the most popular approximation technique ([Akai], [Burden], [Chapra], [Scheid], [Venit]), so we recall it very briefly here. It is applied in both discrete and continuous cases, i.e. when there are given some points or there is given a function to be approximate by an (other) approximating function.

In the ***discrete approximation problem*** discussed in this paper we want to determine a function F which graph passes as close as possible to given n+1 points (x_k,y_k), k=0,1,...,n, of the real Cartesian plane Oxy. In aim to solve this problem we need to precise which is the form of the approximating function F and what does the best fitting mean.

If we opt for the function F being the polynomial in variable x, then we have so called ***polynomial approximation***. Now the coefficients of a polynomial are to be determined. Thus if we act within the class of polynomials of degree up to m (i.e. we look for the polynomial of degree m), m+1 coefficients $c_0, c_1, ..., c_m$ are to be found. Obviously, the uniqueness of the solution requires $m \le n$.

We talk about the ***least-square approximation*** (LSA) if the best fitting is defined via the minimisation of the following quantity (called the ***standard deviation*** or ***least-square error***):

(1) $$Q := \frac{1}{n+1} \cdot \sqrt{\sum_{k=0}^{n} \{F(x_k) - y_k\}^2}$$

So the deviation Q measures the average difference between given points (x_k,y_k) and the points $(x_k,F(y_k))$ laying on the approximating curve. We look for such function F for which the deviation Q is smallest possible.

Differentiating the quantity Q (or, to simplify the notations, the expression $(n+1)/2 \cdot Q^2$) with respect to unknown coefficients of the form F gives (by the Theorem on Extremes of a differentiable function) the following ***resolving system***

(2) $$\sum_{k=0}^{n} \{F(x_k) - y_k\} \cdot \frac{\partial F(x_k)}{\partial c_j} = 0 \text{ , j=0,1,...,m .}$$

Equations forming this system are commonly called ***normal equations of LSA***.

2. Linear least-square approximating

In polynomial case, i.e. when $F(x) = \sum_{j=0}^{m} c_j b_j(x)$, where $b_j(x)$ stays for j-th a priori fixed polynomial of degree exactly j, the resolving system takes (after simple rearrangement of the summation) form

(3) $$\sum_{i=0}^{m} < b_i, b_j > \cdot c_i = < y, b_j >,\ i=0,1,...,m,$$

where the symbol <g,h> stays for the inner product of functions g and h upon the points $\{x_0, x_1, ..., x_n)$, i.e.

(4) $$< g,h >:= \sum_{k=0}^{n} g(x_k)h(x_k),$$

and $y(x_k):=y_k$.

The resolving system (3) is composed of linear equations, so the task consisting in determining the coefficients $c_0, c_1, ..., c_m$ reduces to solving this system which can be rewritten in the matrix form as follows

(5) $$S\cdot c = r,$$

where $c:=[c_0, c_1, ..., c_m]^T$, $S:=P^T\cdot P$, $r:=P^T\cdot y$, $P:=[b_j(x_k)]$.

By Gram Theorem the matrix S is non singular, so the system (5) provides the unique solution c.

The work with the function $F(x):=a+b\cdot x^c$, where a, b and c are unknown coefficients, shows that no every approximation problem can be reduced to solving a system of linear equations. A problem which can be is known as a ***linearisable fitting***.

There are always linearisable least-square approximation problems with the approximating function F being any linear combination of linearly independent functions b_j. Here we can take, for instance, Stevin (or standard, natural) basis, the standard cosine basis and Chebyshev basis, where $b_j(x):=x^{j-1}$, $b_j(x):=\cos(x)$, $b_j(x):=T_j(x)$, $b_j(x):=G_j(x)$, respectively ($T_j(x):=\cos(j\cdot \mathrm{arc}\cos(x))$ defines j-th Chebychev polynomial of first kind, G_j - the j-th polynomial of the Gram orthogonal system built on the abscissas of given points; for definition of these polynomials defined on the regular mesh see e.g. [Ralston], the arbitrary case is presented e.g. in [Blum], [Jankowscy]).

There exist functions F, different from linear combinations of some basic functions, which let to reduce the problem to solving a system of linear equations. Unfortunately, not always the resulting system provides the solution we looked for. That's why we have to distinguish two kinds of linearisation: a perfect one and a problematic one.

3. Perfectly linearised least-square fitting

We talk about the ***perfectly linearisable least-square fitting*** if it reduces (or can be reduced) to the resulting system (5) providing the coefficients c for which the best approximation is realised. Examples of such problems are that with the approximating function G of form, for instance,

(6) $$G(x):=\exp(c_0+c_1\cdot x+...c_m\cdot x^m),\ G(x):=a\cdot x^b.$$

In first case it is enough to apply the inverse functions to pass to the functions

$$F(x):=c_0+c_1\cdot x+...c_m\cdot x^m \text{ with } F(x):=\ln(G(x)),$$

and to deal next with points $(x_k, \ln(y_k))$.

Treating the second case, we represent

$$G(x)=e^{\ln(a)}\cdot e^{b\cdot \ln(x)}=e^{\ln(a)+b\cdot \ln(x)},$$

and it permits to work with the approximating function

$$F(x):=c_0+c_j\cdot x \text{ with } F(x):=\ln(G(x)),\ c_0:=\ln(a),\ c_1:=b$$

and transformed data points $(\ln(x_k), \ln(y_k))$. By the way let's say that we reduced our fitting to the classical regression problem (see e.g. [Eide], [Sobol]).

4. Problematically linearised least-square fitting

Differently than in perfectly linearisable fitting, in this approximation the formal transformations reduce the approximation task to the resolving system, but not always it provides the searched result. A typical situation of this case holds with rational fitting, where we seek a function of the form, let's say,

(7) $$G_{p,q}(x):=\frac{\sum_{j=0}^{p} a_j x^j}{\sum_{s=0}^{q} z_s x^s}$$

with a priori fixed degrees p of the nominator $a_0+a_1\cdot x+\ldots a_p\cdot x^p$ and q of the denominator $z_j+z_j\cdot x+\ldots z_q\cdot x^q$. In the sequel we consider essentially rational functions, i.e. we assume q>0 and $z_q\neq 0$. No generality is lost when we set $z_q:=1$. To simplify the description of the problem we restrict ourselves to p=q=1, so we discuss the ***(1,1)-rational approximation***, i.e. within the class composed of functions of the form

(8) $$G(x):=\frac{a_0+a_1\cdot x}{b_0+x}.$$

Multiplying both sides of (8) by the denominator we get the relation

$$a_0+a_1\cdot x-G(x)\cdot b_0=x\cdot G(x).$$

Thus we have

(9) $$c_0+c_1\cdot x+G(x)\cdot c_2=F(x)$$

where we denoted

$$F(x):=x\cdot G(x),$$
$$c:=[\ c_0,\ c_1,\ c_2\ ]^T := [\ a_1,\ a_0,\ -b_0\ ]^T.$$

Proceeding in the analogous way as before we take the quality function

$$Q=\frac{1}{n+1}\cdot\sqrt{\sum_{k=0}^{n}\{c_0\cdot x_k+c_1+c_2\cdot y_k-x_k\cdot y_k\}^2}$$

and we find that the coefficients c_0, c_1, c_2 have to satisfy the system (5), where

(10) $$P:=\begin{bmatrix} x_0 & 1 & y_0\\ x_1 & 1 & y_1\\ \vdots & \vdots & \vdots\\ x_n & 1 & y_n\end{bmatrix},\ y:=\begin{bmatrix} x_0\cdot y_0\\ x_1\cdot y_1\\ \vdots\\ x_n\cdot y_n\end{bmatrix}.$$

Thus the coefficient matrix S and the free term vector of the system (5) are

(11) $$S=\begin{bmatrix}\sum x_k^2 & \sum x_k & \sum x_k y_k\\ \sum x_k & n+1 & \sum y_k\\ \sum x_k y_k & \sum y_k & \sum y_k^2\end{bmatrix},\ r=\begin{bmatrix}\sum x_k^2\cdot y_k\\ \sum x_k\cdot y_k\\ \sum x_k\cdot y_k^2\end{bmatrix}$$

where the summation is expanded for k = 0, 1, ..., n.

This is not evident which composition of given numbers x_k, y_k results in the system (5) having an unique solution or being inconsistent (the conditions $\det(S)\neq 0$ and $\mathrm{rank}(S)\neq\mathrm{rank}(S\ \ r)$, respectively). Surprisingly, there is here even one more case: the system has a solution which can not be accepted.

All three cases are illustrated with data composed of 4 points having abscissas equal to 0, 1, 3 and 4. We deal with corresponding ordinates equal to a) 2, 2, 1, 1, b) 2, 1, 1, 1, c) 1, 1, 1, 1. There are produced resolving systems S·c = r where

$$S=\begin{bmatrix}26 & 8 & 8\\ 8 & 8 & 4\\ 8 & 4 & 4\end{bmatrix},\ r=\begin{bmatrix}26\\ 8\\ 8\end{bmatrix};\ S=\begin{bmatrix}26 & 8 & 8\\ 8 & 4 & 5\\ 8 & 5 & 7\end{bmatrix},\ r=\begin{bmatrix}26\\ 8\\ 8\end{bmatrix};\ S=\begin{bmatrix}26 & 8 & 9\\ 8 & 4 & 6\\ 9 & 6 & 10\end{bmatrix},\ r=\begin{bmatrix}27\\ 9\\ 11\end{bmatrix}$$

in cases a). b) and c), respectively. We investigate them below.

Case a). The resolving system has infinitely many solutions: $c = [\ 1, \alpha, -\alpha\ ]^T$, where α denotes an arbitrary parameter. Thus it is hard to correctly interpret the output function is $x \rightarrow (\alpha+x)/(x+\alpha)$, which equals to 1 for every $x \neq \alpha$. Moreover, this function (even if is made continuos at $x = \alpha$ by assigning the value 1) is not of desired form.

Case b). There exists exactly one solution $c = [\ 1, 0, 0\ ]^T$ to the system $S\cdot c = r$. This solution generates the expression $F(x) = (c1+c_0\cdot x)/(c_2+x) = x/x$. This formal answer can not be accepted because it is not defined at the point $x = 0$ (and the continuitisation at this point indicates the value 1, not 2). The analogous situations occur for data (-2,1), (0,1) and (4,2), and if given points are (-2,1), (0,2) and (4,1). We obtain now the systems

$$\begin{bmatrix} 20 & 2 & 6 \\ 2 & 3 & 4 \\ 2 & 4 & 6 \end{bmatrix} \cdot c = \begin{bmatrix} 36 \\ 6 \\ 14 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 20 & 2 & 2 \\ 2 & 3 & 4 \\ 2 & 4 & 6 \end{bmatrix} \cdot c = \begin{bmatrix} 20 \\ 2 \\ 2 \end{bmatrix},$$

respectively. They have the unique solutions $c = [\ 1, -4, -4\ ]^T$ and $c = [\ 1, 0, 0\ ]^T$ and they determine the approximating functions $F(x) = (x-4)/(x-4)$ and $F(x) = x/x$. Both these function are equal to 1 at every argument x but $x = 4$ and $x = 0$, respectively. Naturally, they can not be accepted as the solutions, because their graphs do not pass through given points (note that here the approximation reduces to the collocation by a rational function, comp. [Marlewski]). We exclude these solutions via the examining their behaviour for every abscissa of given points.

Case c). The vector $c = [\ 3/2, -15/4, 2\ ]^T$ is the only solution to the resolving system $S\cdot c = r$. The function $F(x) = (3/2\cdot x-15/4)/(-2+x) = 3/4\cdot(2x-5)/(x-2)$ is that we looked for, its graph passes as close (in the sense of LSA) as possible to given points (0,2), (1,2), (3,1), (4,1).

5. CAS assistance in linear least-square fitting

As it was outlined above, the linearisable least-square fitting reduces to solving a system (5) of linear equations. Even for few points (x_k,y_k) there are to be performed arduous, time-consuming summations and resolving systems of linear equations are to be solved. In general, these systems are very sensitive, so there is really need to facilitate this procedure. It is where a computer algebra system may efficiently assists. We will discuss it in case of ***DERIVE*** from Warehouse Inc., but the question is the same if one works with. let's say for instance, Maple from Waterloo Maple Software or Mathematica from Wolfram Research Inc.

The program ***DERIVE*** provides the built-in function `FIT` which returns the best least-square approximation within indicated class of function to given data. It is enough to simplify the evoking such as

```
FIT([x,c0+c1*x+c2*x^2],xy)
```

to get the answer

$$-x^2/3+11\cdot x/15+11/5$$

assuming that `xy:=[[0,2],[1,3],[3,1],[4,0]]` is the matrix collecting four given points (0,2), (1,3), (3,1) and (4,0).

This very efficient function has no big educative value because it does not show "bricks which build the final house". In this case these bricks are matrices P, S and the vector r presented in the system (5). In particular, it makes that a student coming to false result does not know where (s)he made an error (or, maybe, more errors). That's why we suggest that in lessons on least-square fitting an other function should be applied. For the simplicity we restrict ourselves only to the case of Stevin polynomial approximation. The function `LPO_` was supplied for students. We evoke it in the form `LPO_(xy,n)`, where xy is (identically as in the case of the built-in function `FIT`) the 2-column matrix listing given points, and n stays for the degree of the searched polynomial. Simplification yields the same expression as the function `FIT` does. But after this simplification we can get the value of the supplementary variable `lpo_all`. This displays the vector comprising 7 elements.

They are sequentially: 1) the matrix P, 2) the matrix S, 3) the vector r, 4) the vector c solving the system S ·c=r, 5) the approximating polynomial expression, 6) the matrix facilitating the marking of deviations at every given point, 7) the standard deviation. One easily note that the simplification of the calling **LPO_(xy,n)** exposes the fifth element of this list only.

In Fig.1 there is shown a sample evoking of **LPO_** function, its simplification and the values returned by the supplementary variable **lpo_all**. Its 5th and 6th components are plotted in Window 2, they are the parabolic arc and the thin bars starting on the axis Ox and visualisating the deviations at every given

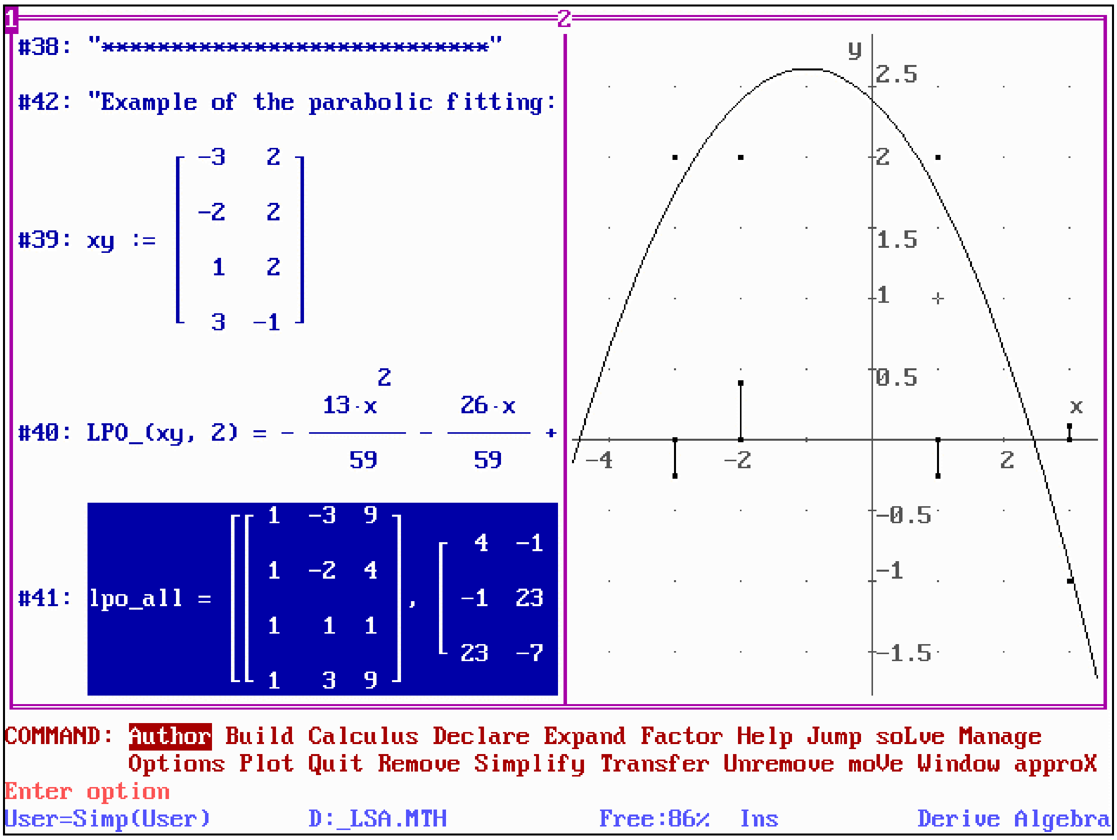

Fig.1. Parabolical fitting discussed in Section 5

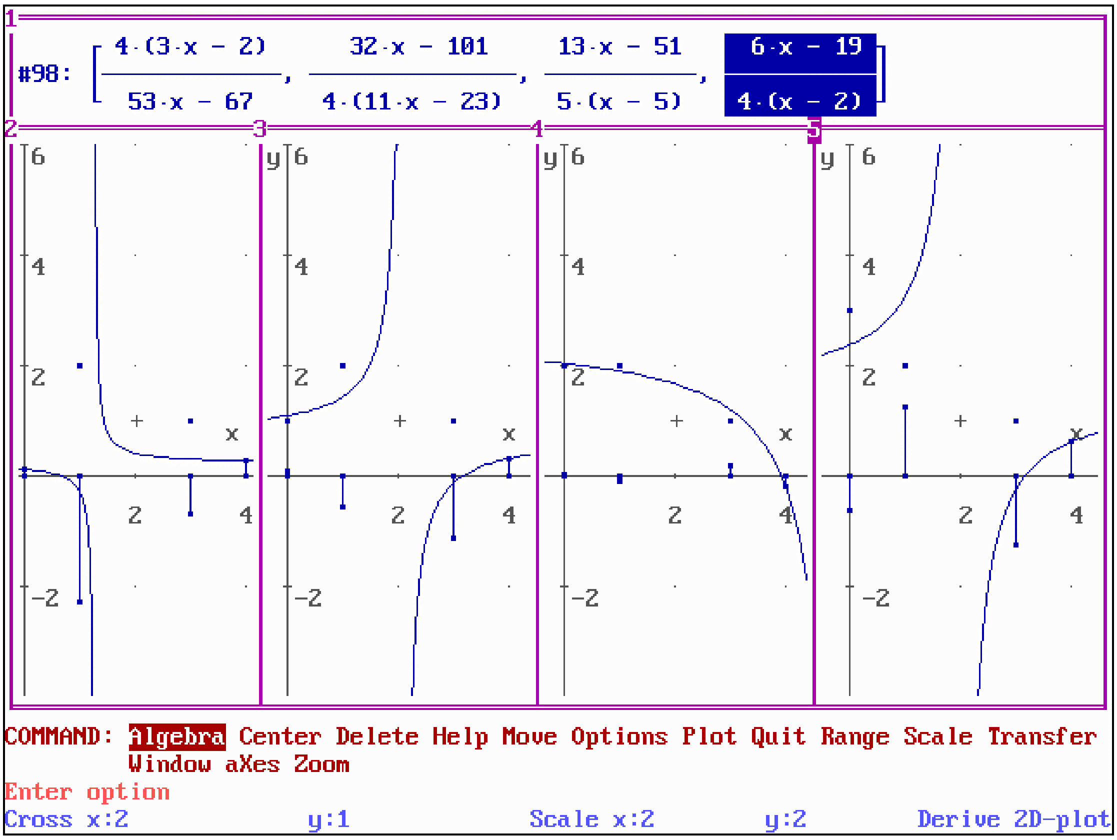

Fig.2. Four sample approximations by functions of the form (a·x+b)/(x-c).
We simplified the calls **ARA([[0,z],[1,2],[3,1],[4,0]])** for z = 0, 1, 2 and 3

point. The whole picture is enriched with the image of given points (the value **Discrete** has to be assigned to the field **Options State Mode:** now, while it has to be switched to **Connected** when the deviations are to be plotted).

6. CAS assistance to rational least-square fitting

There is no build-in or unit-provided function in DERIVE which may be applied to produce the best discrete least-square rational approximation. This problem is not discussed also in books on approximation (see Bibliography), although the continuous case is discussed. That's why here we had to construct our own function. In this paper we do it in case of the strict (1,1)-rational fitting, i.e. when the approximating function is of the form (9). In presence of some complications mentioned in Chapter 4, much more than in Chapter 6 we need here to use the function which does not expose only the final output (we know, it may be the expression describing the approximating function, or the statement that such a function does not exist). That's why we follow the approach applied in case of the function **LPO**. We supply students with the definition of the own-defined function **ARA**. The syntax of the evoking is **ARA(xy)**, where **xy** stays for the same matrix as above. The simplification results in the expression describing found approximation function of the form (8) or in the statements on the singularity or non equivalence. By the way there is assigned the value of the supplementary variable **ara_all** and we can see it if interested in the details of the process leading to the final conclusion. **ara_all** is the vector of the length depending on the case. If the searched is determined, it is of the form as **all_lpo** is. If the quested function does not exist, all_lpo comprises only 1) the matrix P, 2) the matrix S, 3) the vector r, 4) the inscription **"SINGULAR CASE"** (if the system has infinitely many solutions and no one of them can not be acceptable) or **"NO EQUIVALENCE"** (if the resolving system is uniquely solvable, but the obtained function can not be accepted).

7. Conclusion

Looking for best linear least-square approximation is one of most frequently used techniques in both theoretical researches (based on the experiment observations) and practice applications. There is no university program in mathematics which does not treat this question, specially because out of its wide applicability it is an excellent area to exercise the skills gained in courses in linear algebra. On the other side, this part of mathematics is not deeply explored, and usually it is limited to deal with the standard polynomial approximation. In this paper we showed how to adopt the least-square method to linearisable problems, which sometimes reveal some unexpected features. An essentially aid furnished by computer algebra systems (such as ***DERIVE***) is outlined here. This aid does not serve only to speed up the calculations (students do not have to obtain matrices and solve systems of equations) and to observe how data influence the result (see Fig.2). Thanks to constructions **`lpo_all`** and **`ara_all`** it helps to better understand the algorithm and control its performation.

Appendix.

We give here the definitions of functions described in the paper, as well as examples of their use (for data considered in Chapters 4, too).

```
DIM(v):=DIMENSION(v)
    ;standard deviation of the function f (of variable x) on data
     collected in vector xy:
OSTA_(xy,f):=SQRT(SUM((LIM(f,x,xy SUB j SUB 1)-xy SUB j SUB 2)^2,
                      j,DIM(xy)))/DIM(xy)
     ;matrix to see the deviations of the function f (of variable x)
      at points listed in the vector xy (keep Option State Mode: Connected):
BAR_VIS(xy,f):=VECTOR([[xj_:=ELEMENT(xy,j,1),0],
                       [xj_,LIM(f,x,xj_)-ELEMENT(xy,j,2)]],j,DIM(xy))
"------------------------------------------------------------------"
     ;matrix of powers of Stevin base (1,x,x^2,...,x^m):
LPO_POW(xx,m):=VECTOR(VECTOR(xx SUB j^k,k,0,m),j,DIM(xx))
     ;best LSA to data xy by the standard polynomial of degree m:
LPO_(xy,m):= (lpo_all:=[lpo_p:=LPO_POW(xy` SUB 1,m),
              lpo_s:=lpo_p`.lpo_p,lpo_r:=lpo_p`.xy` SUB 2,
              lpo_c:=lpo_r/lpo_s,lpo_f:=lpo_c*VECTOR(x^(j-1),j,m+1),
              BAR_VIS(xy,lpo_f),lpo_q:=OSTA_(xy,lpo_f)]) SUB 5
     ;sample data:
xy:=[[-3,2],[-2,2],[1,2],[3,-1]]
     ;sample simplification resulting in searched function:
LPO_(xy,2)=-13*x^2/59-26*x/59+142/59
     ;displaying the value of the supplementary variable lpo_all
      (5th and 6th components, as well as the argument xy, are shown at Fig.1):
lpo_all=[[[1,-3,9],[1,-2,4],[1,1,1],[1,3,9]],[[4,-1,23],[-1,23,-7],[23,-7,179]],
         [5,-11,19],[142/59,-26/59,-13/59],-13*x^2/59-26*x/59+142/59,
         [[[-3,0],[-3,-15/59]],[[-2,0],[-2,24/59]],
          [[1,0],[1,-15/59]],[[3,0],[3,6/59]]],3*SQRT(118)/236]
"------------------------------------------------------------------"
     ;matrix and vector (10):
[ARA_A(xy):=VECTOR([xy SUB j SUB 1,1,xy SUB j SUB 2],j,DIM(xy)),
 ARA_B(xy):=VECTOR(xy SUB j SUB 1*xy SUB j SUB 2,j,DIM(xy))]
     ;shortnames for matrices (10) and creation of matrices in (11):
ara_i:=[ara_p:=ARA_A(xy),ara_v:=ARA_B(xy),
        ara_s:=ara_p`.ara_p, ara_r:=ara_p`.ara_v]
     ;answer in case the fitting is okey:
ara_o:=[ara_c, ara_f:=(ara_c↓1*x+ara_c↓2)/(x-ara_c↓3),
        BAR_VIS(xy,ara_f), ara_q:=OSTA_(xy,ara_f)]
     ;examing for 0 in the denominator (NO EQUIVALENCE case):
ara_j:=IF(IS_IN_DATA((ara_c:=ara_r/ara_s) SUB 3,
          xy`SUB 1),["NO EQUIVALENCE"],ara_o)
     ;augmenting initial part with NO EQUIVALENCE announcement:
ara_e:=APPEND(ara_i,ara_j)
     ;augmenting initial part with SINGULAR CASE announcement:
```

```
ara_0:=APPEND(ara_i,["SINGULAR CASE"])
     ;collapsing 3 cases into one bunch:
ara_:=IF(DET(ara_i↓3)=0,(ara_all:=ara_0)↓5,
                         (ara_all:=ara_e) SUB IF(DIM(ara_j)=1,5,6))
     ;function yielding the best LSA or the appropriate announcement:
ARA(m):=0*(xy:=m) SUB 1 SUB 1+ara_
     ;Case a) in Chapter 4: example of use - no fitting is found
      (there is a parameter in the solution of resolving system):
ARA([[0,1],[1,1],[3,1],[4,1]])="SINGULAR CASE"
ara_all=[[[0,1,1],[1,1,1],[3,1,1],[4,1,1]],[0,1,3,4],
         [[26,8,8],[8,4,4],[8,4,4]],[26,8,8],"SINGULAR CASE"]
     ;Case b) in Chapter 4: example of use - no fitting is found
      (the denominator vanishes for the abscissa of a given point):
ARA([[0,2],[1,1],[3,1],[4,1]])="NO EQUIVALENCE"
ara_all=[[[0,1,2],[1,1,1],[3,1,1],[4,1,1]],[0,1,3,4],[[26,8,8],[8,4,5],[8,5,7]],
         [26,8,8],"NO EQUIVALENCE"]
     ;Case c) in Chapter 4:- best LSA is furnished:
ARA([[0,2],[1,2],[3,1],[4,1]])=3*(2*x-5)/(4*(x-2))
     ;displaying the value of the supplementary variable ara_all (5th and 6th
      components, as well as the argument xy,are shown in Window 5 in Fig.2):
ara_all=[[[0,1,2],[1,1,2],[3,1,1],[4,1,1]],[0,2,3,4],[[26,8,9],[8,4,6],[9,6,10]],
         [27,9,11],[3/2,-15/4,2],3*(2*x-5)/(4*(x-2)),[[[0,0],[0,-1/8]],
         [[1,0],[1,1/4]],[[3,0],[3,-1/4]],[[4,0],[4,1/8]]],SQRT(10)/32]
```

Bibliography

[Akai] Applied numerical methods for engineers", John Wiley & Sons, Inc., New York 1993

[Blum] E.K.Blum, "Numerical analysis and computation", Addison Wesley, Reading (Mass.) 1972

[Burden] R.L.Burden, J.D.Faires, 'Numerical analysis", Prindle, Webber & Schmidt, Boston 1985

[Chapra] S.C.Chapra, R.P.Canale, "Numerical methods foe engineers", McGraw-Hill International Editions, 1990

[Eide] A.R.Eide, R.D.Jenioson, L.H.Mashaw, L.L.Northup, "Engineering fundamentals and problem solving", McGraw-Hill Book Company, New York 1979

[Jankowscy] J. i M. Jankowscy, "Przegl• d metod i algorytmów numerycznych. Cz• • • 1", WNT Warszawa 1981

[Marlewski] A.Marlewski, "Rational collocation", DERIVE Newsletters 8/1992, 10-14

[Nicholson] W.K.Nicholson, "Elementary linear algebra with applications", Prindle, Webber & Schmidt, Boston 1986

[Ralston] A.Ralston, "A first course in numerical analysis", McGraw-Hill Book Company, New York 1965; Polish edition: PWN Warszawa 1983

[Scheid] F.Scheid, "Análise numérica", Editora McGraw-Hill de Portugal, 1991

[Sobol] M.S.Sobol, M.K.Starr, "Statistics for buisiness and economics. An action learning approach", McGraw-Hill Book Company, New York 1983

[Venit] S.Venit, W.Bishop, "Elementary linear algebra", Prindle, Webber & Schmidt, Boston 1986