Regresii liniare

2.Liniarizarea expresiilor neliniare

(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed, ISBN-13:978-0-07-340110-2)

Există cazuri în care aproximarea datelor se face cu ajutorul unor curbe neliniare. Pentru a obţine curba de regresie se fac transformări de liniarizare a acestor curbe. Unele dintre cele mai folosite modele neliniare sunt:

- Modelul exponenţial (creşterea populaţiilor, dezintegrarea radioactivă)

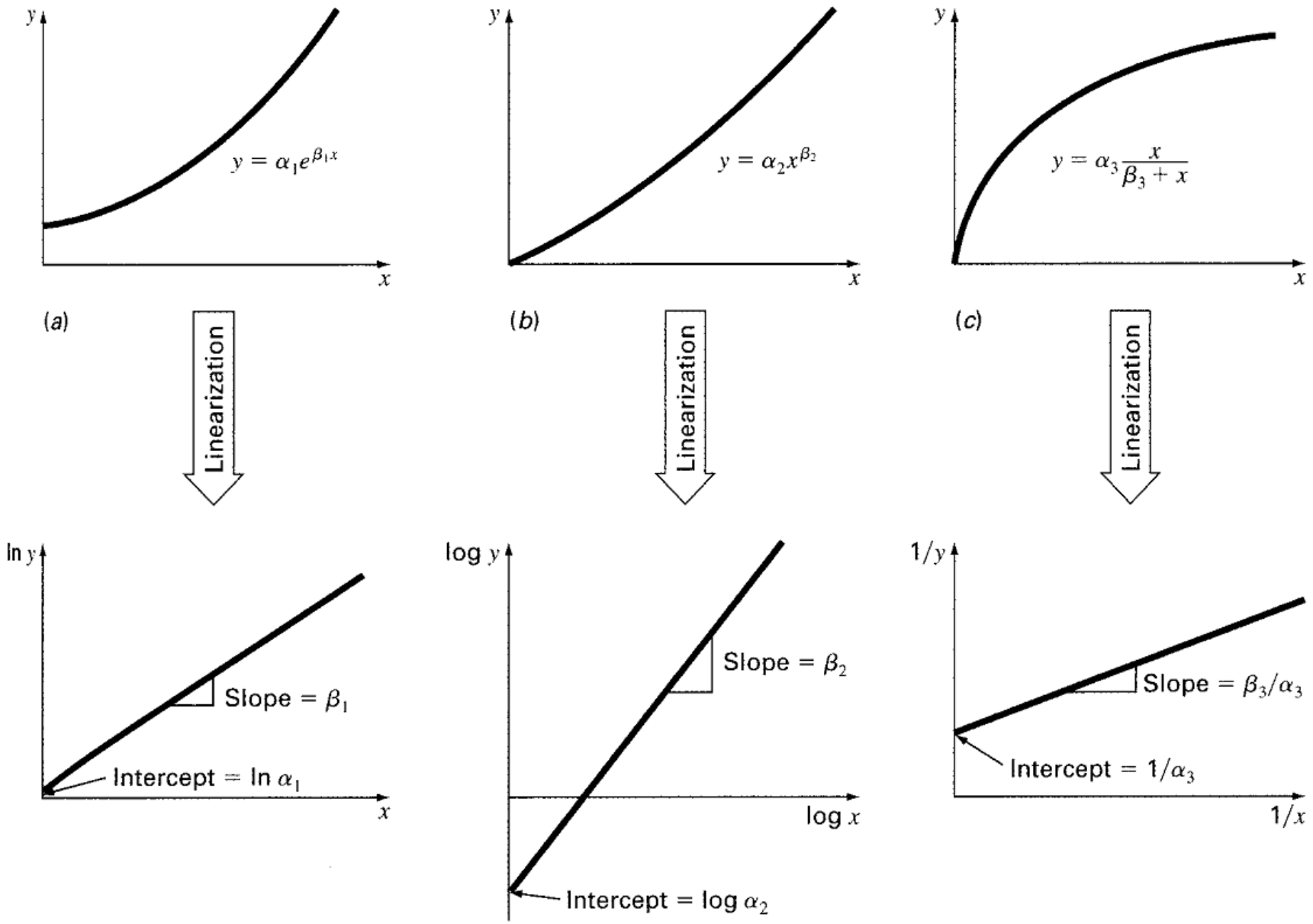
$$y = \alpha_1 e^{\beta_1 x} \longrightarrow \ln y = \ln \alpha_1 + \beta_1 x$$

- Modelul de tip putere (model general, utilizat când nu se ştie forma modelului)

$$y = \alpha_2 x^{\beta_2} \longrightarrow \log y = \log \alpha_2 + \beta_2 \log x$$

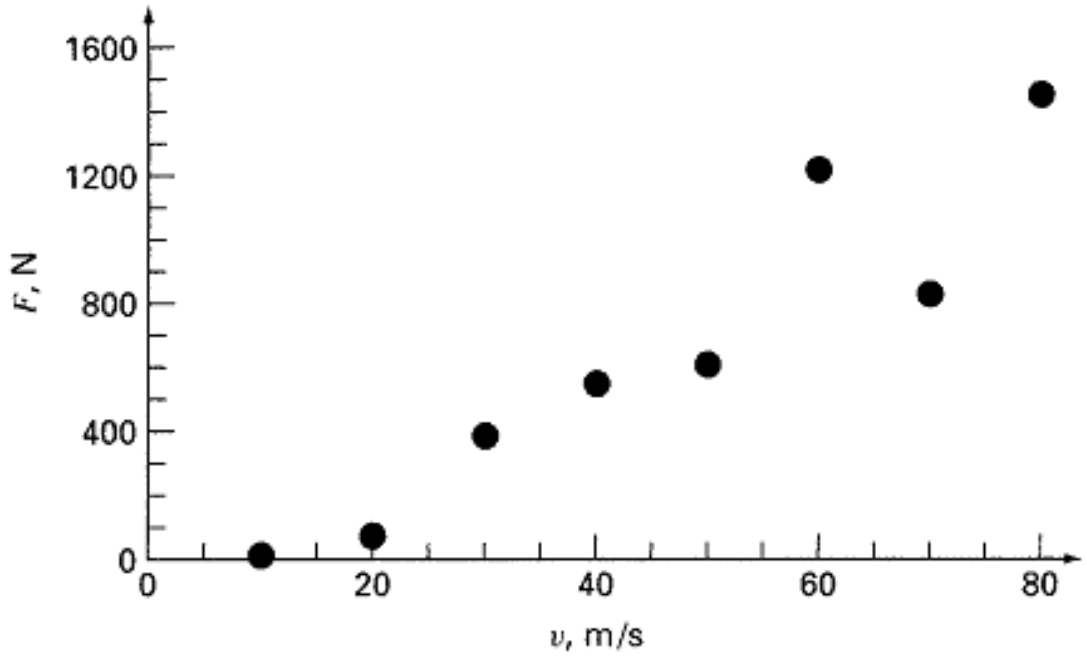
- Modelul de *saturation growth rate* (model utilizat în creşterea populaţiilor când există limitări ale creşterii)

$$y = \alpha_3 \frac{x}{\beta_3 + x} \longrightarrow \frac{1}{y} = \frac{1}{\alpha_3} + \frac{\beta_3}{\alpha_3} \frac{1}{x}$$


y
x
$y = \alpha_1 e^{\beta_1 x}$
(a)
Linearization
ln y
Slope = β_1
Intercept = ln α_1
x
$y = \alpha_2 x^{\beta_2}$
(b)
Linearization
log y
Slope = β_2
log x
Intercept = log α_2
$y = \alpha_3 \frac{x}{\beta_3 + x}$
(c)
Linearization
1/y
Slope = β_3/α_3
Intercept = $1/\alpha_3$
1/x

Exemplu 1: Se ştie ca teoretic, forţa de rezistenţă ce o întampină un obiect la mişcarea prin aer este:

v, m/s	10	20	30	40	50	60	70	80
F, N	25	70	380	550	610	1220	830	1450



Interpolăm folosind o funcţie de tip putere:

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2}$$

$$a_0 = \bar{y} - a_1 \bar{x}$$

Unde mediile sunt (vezi tabelul de mai jos)

$$\bar{x} = \frac{12.606}{8} = 1.5757 \qquad \bar{y} = \frac{20.515}{8} = 2.5644$$

Atunci

$$a_1 = \frac{8(33.622) - 12.606(20.515)}{8(20.516) - (12.606)^2} = 1.9842$$

$$a_0 = 2.5644 - 1.9842(1.5757) = -0.5620$$

i	x_i	y_i	$\log x_i$	$\log y_i$	$(\log x_i)^2$	$\log x_i \log y_i$
1	10	25	1.000	1.398	1.000	1.398
2	20	70	1.301	1.845	1.693	2.401
3	30	380	1.477	2.580	2.182	3.811
4	40	550	1.602	2.740	2.567	4.390
5	50	610	1.699	2.785	2.886	4.732
6	60	1220	1.778	3.086	3.162	5.488
7	70	830	1.845	2.919	3.404	5.386
8	80	1450	1.903	3.161	3.622	6.016
Σ			12.606	20.515	20.516	33.622

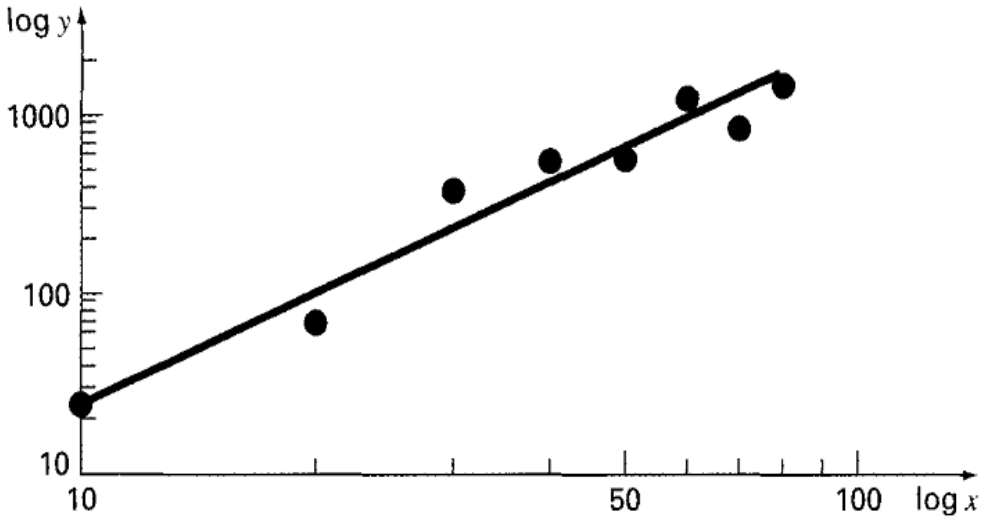
Obţinem

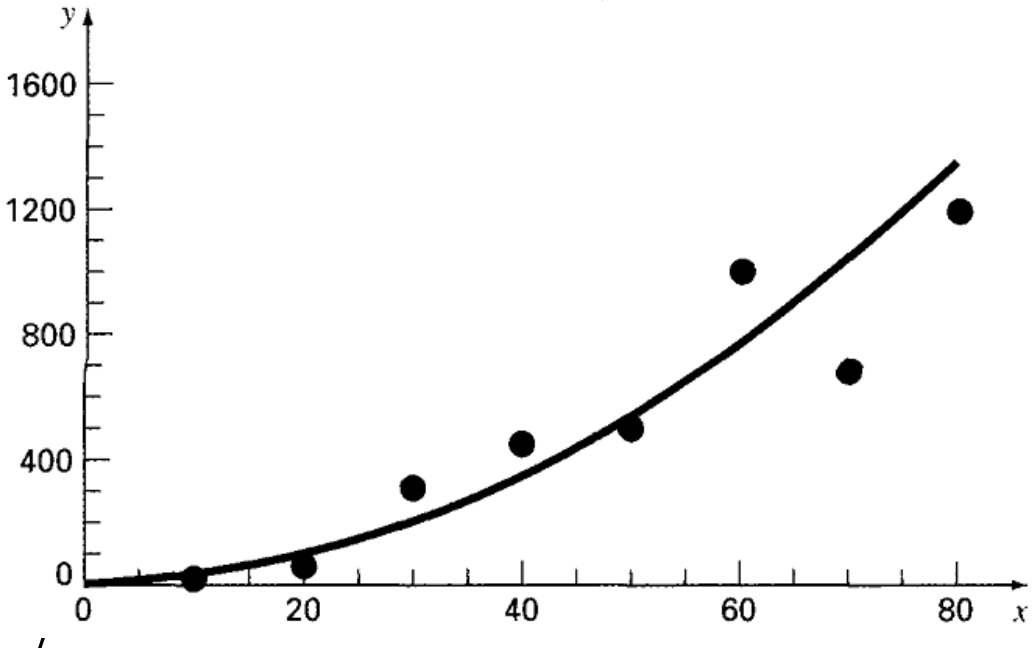
$$\log y = -0.5620 + 1.9842 \log x$$

$$\alpha_2 = 10^{-0.5620} = 0.2741 \text{ and } \beta_2 = 1.9842.$$

şi deci

$$F = 0.2741 v^{1.9842}$$

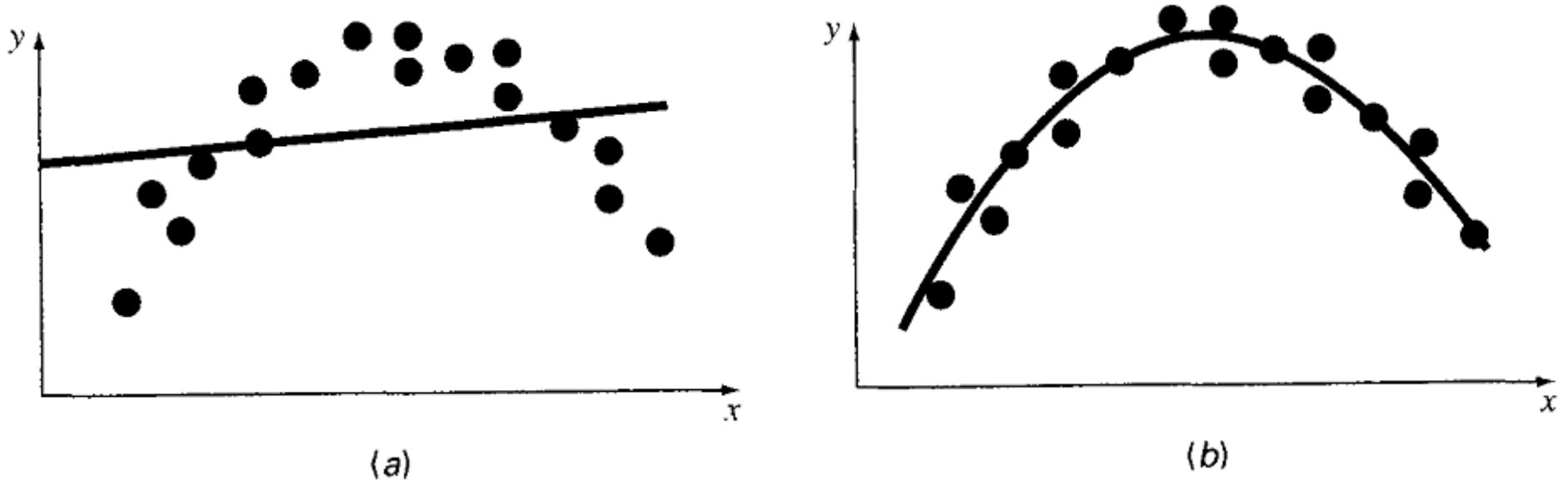

log y
1000
100
10
10
50
100
log x


y
1600
1200
800
400
0
0
20
40
60
80
x

Regresia liniară poate fi calculată cu

```
function [a, r2] = linregr(x,y)
% linregr: linear regression curve fitting
%    [a, r2] = linregr(x,y): Least squares fit of straight
%              line to data by solving the normal equations

% input:
%   x = independent variable
%   y = dependent variable
% output:
%   a = vector of slope, a(1), and intercept, a(2)
%   r2 = coefficient of determination

n = length(x);
if length(y)~=n, error('x and y must be same length'); end
x = x(:); y = y(:);     % convert to column vectors
sx = sum(x); sy = sum(y);
sx2 = sum(x.*x); sxy = sum(x.*y); sy2 = sum(y.*y);
a(1) = (n*sxy-sx*sy)/(n*sx2-sx^2);
a(2) = sy/n-a(1)*sx/n;
r2 = ((n*sxy-sx*sy)/sqrt(n*sx2-sx^2)/sqrt(n*sy2-sy^2))^2;
% create plot of data and best fit line
xp = linspace(min(x),max(x),2);
yp = a(1)*xp+a(2);
plot(x,y,'o',xp,yp)
```

3.Generalizarea metodei celor mai mici pătrate.Regresie neliniară. Regresie polinomială



(*a*) Data that is ill-suited for linear least-squares regression. (*b*) Indication that a parabola is preferable.

Regresia liniară descrisă anterior se poate generaliza în cazul polinoamelor. De exemplu, pentru un polinom de gradul doi avem:

$$y = a_0 + a_1 x + a_2 x^2 + e$$

Trebuie să minimizăm:

$$S_r = \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right)^2$$

adică

$$\frac{\partial S_r}{\partial a_0} = -2 \sum \left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right)$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum x_i \left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right)$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum x_i^2 \left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right)$$

Relaţiile de mai sus ne conduc la rezolvarea sistemului

$$\begin{aligned}
(n)a_0 + \left(\sum x_i\right) a_1 + \left(\sum x_i^2\right) a_2 &= \sum y_i \\
\left(\sum x_i\right) a_0 + \left(\sum x_i^2\right) a_1 + \left(\sum x_i^3\right) a_2 &= \sum x_i y_i \\
\left(\sum x_i^2\right) a_0 + \left(\sum x_i^3\right) a_1 + \left(\sum x_i^4\right) a_2 &= \sum x_i^2 y_i
\end{aligned}$$

din aflăm coeficienţii $a_0,\ a_1,\ a_2.$

Se poate face uşor o generalizare pentru:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m + e$$

Pentru un polinom de grad m (m+1 coeficienţi) eroarea standard este:

$$s_{y/x} = \sqrt{\frac{S_r}{n - (m+1)}}$$

iar coeficientul de determinare este dat de:

$$r^2 = \frac{S_t - S_r}{S_t} \quad \text{unde} \quad S_t = \sum (y_i - \bar{y})^2$$

Exemplu 2: Gasiţi polinomul de ordinul doi ce aproximează datele din tabelul de mai jos:

x_i	y_i	$(y_i - \bar{y})^2$	$(y_i - a_0 - a_1 x_i - a_2 x_i^2)^2$
0	2.1	544.44	0.14332
1	7.7	314.47	1.00286
2	13.6	140.03	1.08160
3	27.2	3.12	0.80487
4	40.9	239.22	0.61959
5	61.1	1272.11	0.09434
$\sum$	152.6	2513.39	3.74657

Avem:

$$
\begin{array}{lll}
m = 2 & \sum x_i = 15 & \sum x_i^4 = 979 \\
n = 6 & \sum y_i = 152.6 & \sum x_i y_i = 585.6 \\
\bar{x} = 2.5 & \sum x_i^2 = 55 & \sum x_i^2 y_i = 2488.8 \\
\bar{y} = 25.433 & \sum x_i^3 = 225 &
\end{array}
$$

Formăm sistemul:

$$
\begin{bmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 152.6 \\ 585.6 \\ 2488.8 \end{Bmatrix}
$$

Rezolvăm folosind Matlab

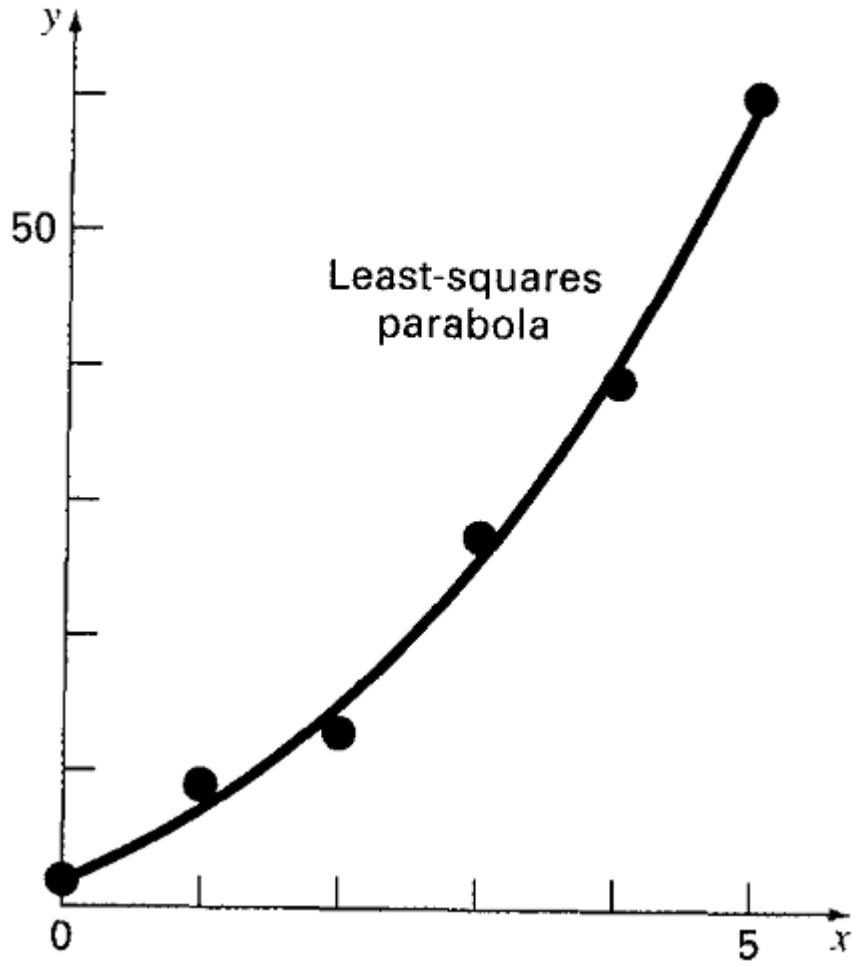
```
>> N = [6 15 55;15 55 225;55 225 979];
>> r = [152.6 585.6 2488.8];
>> a = N\r

a =
    2.4786
    2.3593
    1.8607
```

şi obţinem polinomul $y = 2.4786 + 2.3593x + 1.8607x^2$

pentru care

$$s_{y/x} = \sqrt{\frac{3.74657}{6-(2+1)}} = 1.1175 \qquad \text{şi} \qquad r^2 = \frac{2513.39 - 3.74657}{2513.39} = 0.99851$$

Se obţine un coeficient de determinare foarte bun. Buna aproximare este confirmată vizual şi de figura următoare:


y
50
Least-squares
parabola
0
5
x

În Matlab regresia polinomială poate fi calculată folosind funcţia `polyfit`

```
p = polyfit(x,y,n)
```

unde

$$p(x) = p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1}.$$

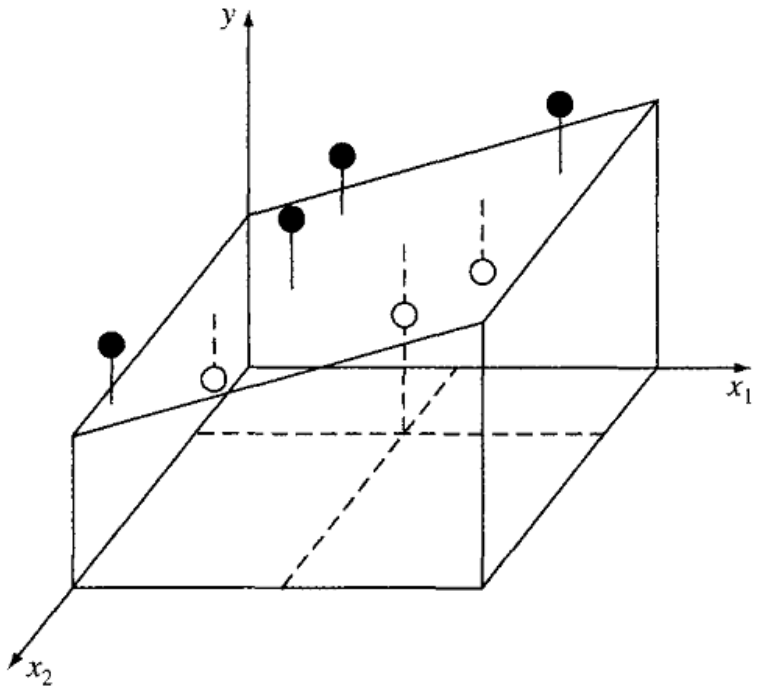
Exemplu 3

```
x = (0: 0.1: 2.5)';
y = erf(x);
p = polyfit(x,y,6)
p =
0.0084  -0.0983   0.4217   -0.7435  0.1471   1.1064  0.0004
```

Regresie liniară multidimensională

Înmod natural se poate face o generalizare a regresiei linare în mai multe dimensiuni. De exemplu, în $\mathbf{R}^2$ vom obţine un plan de regresie:

$$y = a_0 + a_1 x_1 + a_2 x_2 + e$$



Trebuie să minimizăm:

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_{1,i} - a_2 x_{2,i})^2$$

$$\frac{\partial S_r}{\partial a_0} = -2 \sum (y_i - a_0 - a_1 x_{1,i} - a_2 x_{2,i})$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum x_{1,i} (y_i - a_0 - a_1 x_{1,i} - a_2 x_{2,i})$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum x_{2,i} (y_i - a_0 - a_1 x_{1,i} - a_2 x_{2,i})$$

ceea conduce la rezolvarea sistemului

$$\begin{bmatrix} n & \sum x_{1,i} & \sum x_{2,i} \\ \sum x_{1,i} & \sum x_{1,i}^2 & \sum x_{1,i}x_{2,i} \\ \sum x_{2,i} & \sum x_{1,i}x_{2,i} & \sum x_{2,i}^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} \sum y_i \\ \sum x_{1,i}y_i \\ \sum x_{2,i}y_i \end{Bmatrix}$$

Exemplu 4: Găsiţi planul de regresie pentru:

y	x_1	x_2	x_1^2	x_2^2	x_1x_2	x_1y	x_2y
5	0	0	0	0	0	0	0
10	2	1	4	1	2	20	10
9	2.5	2	6.25	4	5	22.5	18
0	1	3	1	9	3	0	0
3	4	6	16	36	24	12	18
27	7	2	49	4	14	189	54
54	16.5	14	76.25	54	48	243.5	100

Avem de rezolvat

$$\begin{bmatrix} 6 & 16.5 & 14 \\ 16.5 & 76.25 & 48 \\ 14 & 48 & 54 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 54 \\ 243.5 \\ 100 \end{Bmatrix}$$

$$a_0 = 5 \qquad a_1 = 4 \qquad a_2 = -3$$

Regresia liniară multidimensională se poate generaliza astfel:

$$y = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_m x_m + e$$

În Matlab regresia liniară multidimensională poate fi calculată folosind funcţia `regress`

```
b = regress(y,X)
[b,bint] = regress(y,X)
[b,bint,r] = regress(y,X)
[b,bint,r,rint] = regress(y,X)
```

Exemplu 5

```
load carsmall
x1 = Weight;
x2 = Horsepower; % Contains NaN data
y = MPG;%mileage - numar de mile parcurse
```

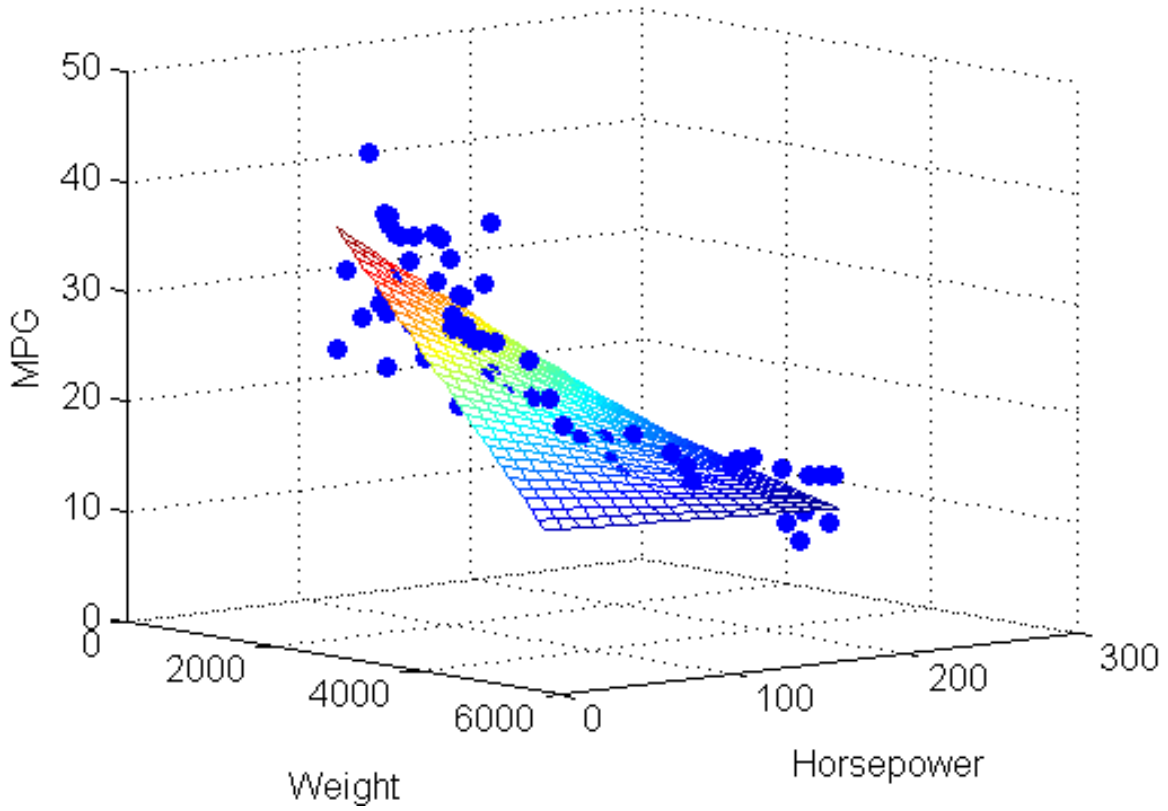
Compute regression coefficients for a linear model with an interaction term:

```
X = [ones(size(x1)) x1 x2 x1.*x2];
b = regress(y,X) % Removes NaN data
b =
  60.7104
  -0.0102
  -0.1882
   0.0000
```

Plot the data and the model:

```
scatter3(x1,x2,y,'filled')
hold on
x1fit = min(x1):100:max(x1);
x2fit = min(x2):10:max(x2);
[X1FIT,X2FIT] = meshgrid(x1fit,x2fit);
YFIT = b(1) + b(2)*X1FIT + b(3)*X2FIT + b(4)*X1FIT.*X2FIT;
mesh(X1FIT,X2FIT,YFIT)
```

```
xlabel('Weight')
ylabel('Horsepower')
zlabel('MPG')
view(50,10)
```



Regresia liniară multiplă poate fi folosită pentru determinarea unor relaţii de tipul:

$$y = a_0 x_1^{a_1} x_2^{a_2} \cdots x_m^{a_m}$$

$$\log y = \log a_0 + a_1 \log x_1 + a_2 \log x_2 + \cdots + a_m \log x_m$$

Regresie neliniară

Există cazuri în care funcţia cu care vrem să aproximăm datele nu se poate liniariza (de exemplu):

$$y = a_0(1 - e^{-a_1 x}) + e$$

În acest caz se poate face o liniarizare folosind dezvoltări în serie Taylor sau se poate face minimizarea direct, rezolvând un sistem neliniar.

Funcţia ce trebuie minimizată este

$$f(a_0, a_1) = \sum_{i=1}^{n} [y_i - a_0(1 - e^{-a_1 x_i})]^2$$

De exemplu, în Matlab există funcţia `fminsearch (Optimization Toolbox)` ce minimizează o funcţie.

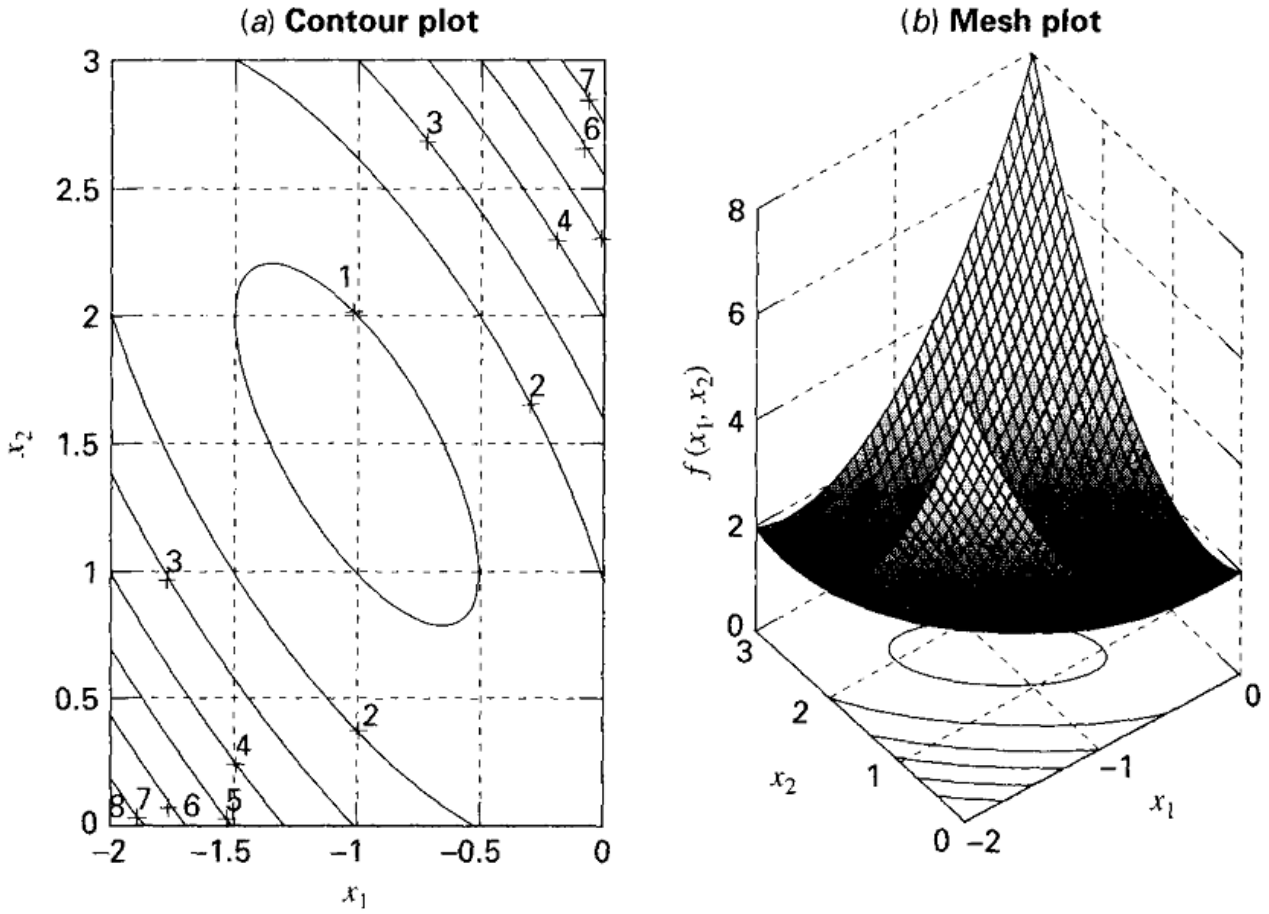
```
[x, fval] = fminsearch(fun,x0,options,p1,p2,...)
```

Exemplu 6: Fie funcţia

$$f(x_1, x_2) = 2 + x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2 \quad , \quad -2 \le x_1 \le 0 \text{ and } 0 \le x_2 \le 3$$

```
x=linspace(-2,0,40);y=linspace(0,3,40);
[X,Y] = meshgrid(x,y);
Z=2+X-Y+2*X.^2+2*X.*Y+Y.^2;
subplot(1,2,1);
cs=contour(X,Y,Z);clabel(cs);
xlabel('x_1');ylabel('x_2');
title('(a) Contour plot');grid;
subplot(1,2,2);
cs=surfc(X,Y,Z);
zmin=floor(min(Z));
zmax=ceil(max(Z));
xlabel('x_1');ylabel('x_2');zlabel('f(x_1,x_2)');
title('(b) Mesh plot');
```

Din figura de mai jos minimul se afla în vecinătatea punctului $x_1 = -1$ and $x_2 = 1.5$.



```
>> f=@(x) 2+x(1)-x(2)+2*x(1)^2+2*x(1)*x(2)+x(2)^2;
>> [x,fval]=fminsearch(f,[-0.5,0.5])

x =
   -1.0000    1.5000
fval =
    0.7500
```

Exemplu 7: Considerăm din nou exemplul cu mişcarea unui obiect în aer:

***v*, m/s**	10	20	30	40	50	60	70	80
***F*, N**	25	70	380	550	610	1220	830	1450

Scriem o funcţie generală ce calculează suma pătratelor

```
function f = fSSR(a,xm,ym)
yp = a(1)*xm.^a(2);
f = sum((ym-yp).^2);
```

Minimizăm

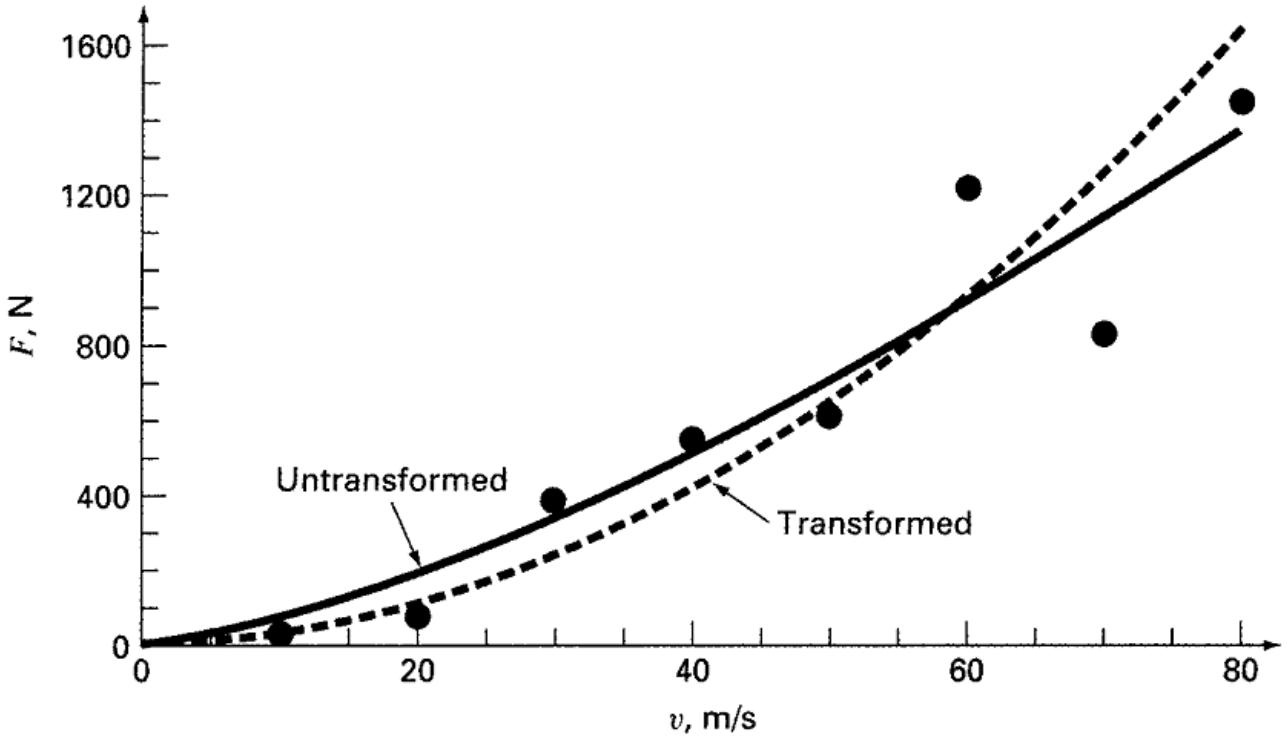
```
>> x = [10 20 30 40 50 60 70 80];
>> y = [25 70 380 550 610 1220 830 1450];

>> fminsearch(@fSSR, [1, 1], [], x, y)
ans =
    2.5384    1.4359
```

şi obţinem

$$F = 2.5384v^{1.4359}$$

Reamintim că prin liniarizare am obţinut

$$F = 0.2741 v^{1.9842}$$



Note that although the model coefficients are very different, it is difficult to judge which fit is superior based on inspection of the plot.

This example illustrates how different best-fit equations result when fitting the same model using nonlinear regression versus linear regression employing transformations. This is because the former minimizes the residuals of the original data whereas the latter minimizes the residuals of the transformed data.

3.Metoda generalizată

(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed, ISBN-13:978-0-07-340110-2)

In the preceding pages, we have introduced three types of regression: simple linear, polynomial, and multiple linear. In fact, all three belong to the following general linear least-squares model:

$$y = a_0 z_0 + a_1 z_1 + a_2 z_2 + \cdots + a_m z_m + e \tag{1}$$

where $z_0, z_1, \ldots, z_m$ are $m + 1$ basis functions. It can easily be seen how simple linear and multiple linear regression fall within this model—that is, $z_0 = 1, z_1 = x_1, z_2 = x_2, \ldots, z_m = x_m$. Further, polynomial regression is also included if the basis functions are simple monomials as in $z_0 = 1, z_1 = x, z_2 = x^2, \ldots, z_m = x^m$.

Note that the terminology "linear" refers only to the model's dependence on its parameters—that is, the a's. As in the case of polynomial regression, the functions themselves can be highly nonlinear. For example, the z's can be sinusoids, as in

$$y = a_0 + a_1 \cos(\omega x) + a_2 \sin(\omega x)$$

Such a format is the basis of *Fourier analysis*.

Equation (1) can be expressed in matrix notation as

$$\{y\} = [Z]\{a\} + \{e\} \tag{2}$$

where [Z] is a matrix of the calculated values of the basis functions at the measured values of the independent variables:

$$[Z] = \begin{bmatrix} z_{01} & z_{11} & \cdots & z_{m1} \\ z_{02} & z_{12} & \cdots & z_{m2} \\ \vdots & \vdots & & \vdots \\ z_{0n} & z_{1n} & \cdots & z_{mn} \end{bmatrix}$$

where m is the number of variables in the model and n is the number of data points. Because $n \geq m + 1$, you should recognize that most of the time, $[Z]$ is not a square matrix.

The column vector $\{y\}$ contains the observed values of the dependent variable:

$$\{y\}^T = \lfloor y_1 \quad y_2 \quad \cdots \quad y_n \rfloor$$

The column vector $\{a\}$ contains the unknown coefficients:

$$\{a\}^T = \lfloor a_0 \quad a_1 \quad \cdots \quad a_m \rfloor$$

and the column vector $\{e\}$ contains the residuals:

$$\{e\}^T = \lfloor e_1 \quad e_2 \quad \cdots \quad e_n \rfloor$$

The sum of the squares of the residuals for this model can be defined as

$$S_r = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} a_j z_{ji} \right)^2 \tag{3}$$

This quantity can be minimized by taking its partial derivative with respect to each of the coefficients and setting the resulting equation equal to zero. The outcome of this process is the normal equations that can be expressed concisely in matrix form as

$$[[Z]^T[Z]]\{a\} = \{[Z]^T\{y\}\} \tag{14.10}$$

The coefficient of determination and the standard error can also be formulated in terms of matrix algebra. Recall that r^2 is defined as

$$r^2 = \frac{S_t - S_r}{S_t} = 1 - \frac{S_r}{S_t}$$

Substituting the definitions of S_r and S_t gives

$$r^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

where $\hat{y}$ = the prediction of the least-squares fit. The residuals between the best-fit curve and the data, $y_i - \hat{y}$, can be expressed in vector form as

$$\{y\} - [Z]\{a\}$$

Reluăm Exemplul 2 folosind metoda generalizată descrisă mai sus

Exemplu 2: Gasiţi polinomul de ordinul doi ce aproximează datele din tabelul de mai jos:

x_i	y_i
0	2.1
1	7.7
2	13.6
3	27.2
4	40.9
5	61.1
$\sum$	152.6

```
>> x = [0 1 2 3 4 5]';
>> y = [2.1 7.7 13.6 27.2 40.9 61.1]';
```

```
>> Z = [ones(size(x)) x x.^2]

Z =
     1     0     0
     1     1     1
     1     2     4
     1     3     9
     1     4    16
     1     5    25
```

```
>> Z'*Z

ans =
     6    15    55
    15    55   225
    55   225   979
```

```
>> a = (Z'*Z)\(Z'*y)

ans =
    2.4786
    2.3593
    1.8607
```

```
>> Sr = sum((y-Z*a).^2)

Sr =
    3.7466
```

```
>> r2 = 1-Sr/sum((y-mean(y)).^2)

r2 =
    0.9985
```

4.Aproximarea sinusoidelor

Prin funcţie sinusoidă vom înţelege toate funcţiile trigonometrice. Dorim să aproximăm fenomenele oscilatorii cu funcţii de forma:

$$f(t) = A_0 + C_1 \cos(\omega_0 t + \theta)$$

unde A_0 este valoarea medie, C_1 este amplitudinea, ω_0 este frecvenţa unghiulară, iar θ este diferenţa de fază.

Folosind identitatea

$$C_1 \cos(\omega_0 t + \theta) = C_1[\cos(\omega_0 t)\cos(\theta) - \sin(\omega_0 t)\sin(\theta)]$$

rescriem funcţia sinusoidă în forma:

$$f(t) = A_0 + A_1 \cos(\omega_0 t) + B_1 \sin(\omega_0 t)$$

$$A_1 = C_1 \cos(\theta)$$

$$B_1 = -C_1 \sin(\theta)$$

apoi se poate aplica teoria generalizată.

(*a*) A plot of the sinusoidal function $y(t) = A_0 + C_1\cos(\omega_0 t + \theta)$. For this case, $A_0 = 1.7$, $C_1 = 1$, $\omega_0 = 2\pi/(1.5\text{ s})$, and $\theta = \pi/3$ radians. (*b*) An alternative expression of the same curve is $y(t) = A_0 + A_1 + A_1\cos(\omega_0 t) + B_1\sin(\omega_0 t)$. The three components of this function are depicted in (*b*), where $A_1 = 0.5$ and $B_1 = -0.866$. The summation of the three curves in (*b*) yields the single curve in (*a*).