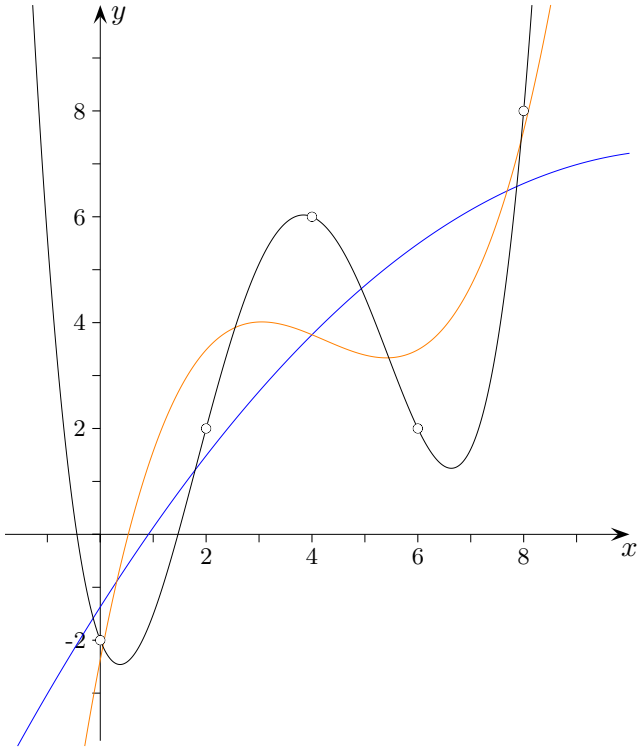
Regression mit Polynomen

x	x_1	x_2	x_3	$\dots$	x_m
y	y_1	y_2	y_3	$\dots$	y_m



Wir gehen von $m = 5$ Punkten aus:

x	y
0	-2
2	2
4	6
6	2
8	8

Eine Ausgleichsfunktion

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

ist zu ermitteln, die gegebene Datenpunkte (x_k, y_k), $k = 1 \ldots m$, optimal annähert, d. h. hier, deren Fehlerquadratsumme

$$q(a_0, a_1, \ldots, a_n) = \sum_{k=1}^{m} \big(f(x_k) - y_k\big)^2 = \sum_{k=1}^{m} \big(a_0 + a_1 x_k + a_2 x_k^2 + \ldots + a_n x_k^n - y_k\big)^2$$

minimal ist.

Die Polynome vom Grad 2, 3 und 4 wurden gezeichnet.

Die Koeffizienten a_i werden über Nullsetzen aller partiellen Ableitungen $\dfrac{\partial}{\partial a_i}$ von q bestimmt.

$$\frac{\partial q}{\partial a_i} = 2 \sum_{k=1}^{m} x_k^i \big(a_0 + a_1 x_k + a_2 x_k^2 + \ldots + a_n x_k^n - y_k\big) = 0, \qquad i = 0 \ldots n \qquad \text{Kettenregel}$$

Wir erhalten durch Auflösen von $\displaystyle\sum_{k=1}^{m} x_k^i \big(a_0 + a_1 x_k + a_2 x_k^2 + \ldots + a_n x_k^n - y_k\big)$ die Gleichungen:

$$\sum_{k=1}^{m} x_k^i \, a_0 + \sum_{k=1}^{m} x_k^{i+1} a_1 + \ldots + \sum_{k=1}^{m} x_k^{i+n} a_n = \sum_{k=1}^{m} x_k^i \, y_k, \qquad i = 0 \ldots n$$

Programm zum Lösen von Gleichungssystemen in Python

```
def zeilen_zusammenfassen(zeile1, zeile2):
    # Das geeignete Vielfache einer Zeile wird zur 2. Zeile addiert.
    # Die an der ersten Stelle stehende Null entfällt durch [1:].
    c = - zeile2[0]/zeile1[0]
    return [c*a+b for a,b in zip(zeile1, zeile2)][1:]



def variablen_eliminieren(lgs):
    # Nach dem Gauss-Verfahren wird das LGS schrittweise jeweils um eine Variable verringert.
    # Zum Schluss entsteht die Gleichung ax=b, deren Lösung b/a zurückgeliefert wird.
    for i in range(len(lgs)):
      erste_zeile = lgs[0]
      lgs_neu = []
      for zeile in lgs[1:]:
           zeile_neu =zeilen_zusammenfassen(erste_zeile, zeile)
           lgs_neu.append(zeile_neu)
           lgs=lgs_neu
    return lgs[0][1]/lgs[0][0]



def einsetzen(lgs, wert):
    # In das Gleichungssystem wird für die ganz rechts stehende Variable die Lösung ein-
    # gesetzt. Die Zahlen werden zusammengefasst. Die rechts stehenden Zahlen entfallen
    # durch [:-1], so dass ein Gleichungssystem mit einer Variablen weniger entsteht.
    return [zeile[:len(lgs)-1] + zeile[len(lgs):-1] + [zeile[-1] - zeile[len(lgs)-1]*wert]
                                                                    for zeile in lgs][:-1]



def loese(lgs):
    loesungen = []
    for i in range(len(lgs)):
        z = variablen_eliminieren(lgs)
        loesungen = [z]+ loesungen
        lgs = einsetzen(lgs, z)
    return loesungen


lgs = [[2,3,-1,11],
      [1,-1, 2,3 ],
      [3,-2, 3,8 ]]

print(loese(lgs))
```

Bei diesem Algorithmus wird jeweils nach Einsetzen einer Lösung
das Gauss-Verfahren erneut angewandt, siehe loese(lgs).

Koeffizientenmatrix für die Regression

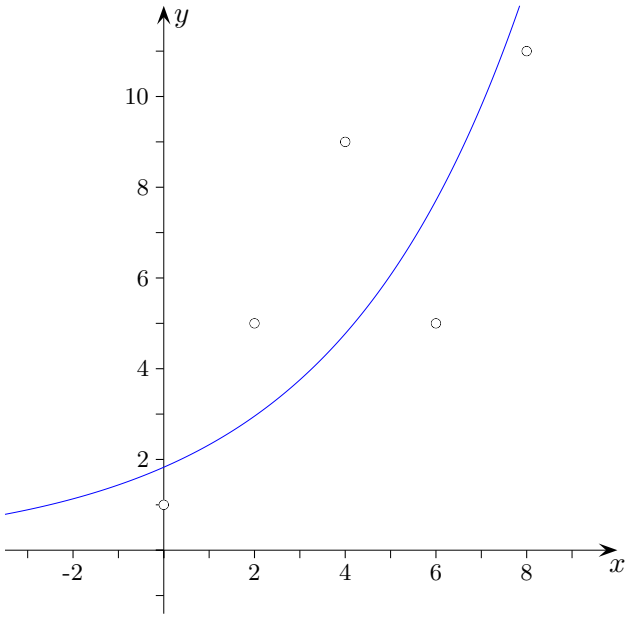
```
m=5    # m Anzahl der Datenpunkte
xWerte=[ 0,2,4,6,8]
yWerte=[-2,2,6,2,8]

def s(i):
    summe=0
    for k in range(m):
        summe+=xWerte[k]**i
    return summe

def t(i):
    summe=0
    for k in range(m):
        summe+=xWerte[k]**i*yWerte[k]
    return summe

# Polynom 3. Grades
lgs = [[s(0),s(1),s(2),s(3),t(0)],
       [s(1),s(2),s(3),s(4),t(1)],
       [s(2),s(3),s(4),s(5),t(2)],
       [s(3),s(4),s(5),s(6),t(3)]]

print(loese_lgs(lgs))
print(loese_lgs(lgs)[0],'+',loese_lgs(lgs)[1],'*x+' ,
           loese_lgs(lgs)[2],'*x^2+',loese_lgs(lgs)[3],'*x^3')

# Polynom 4. Grades
lgs = [[s(0),s(1),s(2),s(3),s(4),t(0) ],
       [s(1),s(2),s(3),s(4),s(5),t(1) ],
       [s(2),s(3),s(4),s(5),s(6),t(2) ],
       [s(3),s(4),s(5),s(6),s(7),t(3) ],
       [s(4),s(5),s(6),s(7),s(8),t(4) ]]
print(loese_lgs(lgs))
print(loese_lgs(lgs)[0],'+',loese_lgs(lgs)[1],'*x+',loese_lgs(lgs)[2],'*x^2+',
          loese_lgs(lgs)[3],'*x^3+',loese_lgs(lgs)[4],'*x^4')

# Polynom 5. Grades
lgs = [[s(0),s(1),s(2),s(3),s(4),s(5), t(0)],
       [s(1),s(2),s(3),s(4),s(5),s(6), t(1)],
       [s(2),s(3),s(4),s(5),s(6),s(7), t(2)],
       [s(3),s(4),s(5),s(6),s(7),s(8), t(3)],
       [s(4),s(5),s(6),s(7),s(8),s(9), t(4)],
       [s(5),s(6),s(7),s(8),s(9),s(10),t(5)]]

print(loese_lgs(lgs))
print(loese_lgs(lgs)[0],'+',loese_lgs(lgs)[1],'*x+' ,loese_lgs(lgs)[2],'*x^2+',
          loese_lgs(lgs)[3],'*x^3+',loese_lgs(lgs)[4],'*x^4+',loese_lgs(lgs)[5],'*x^5')
```

Regression mit $f(x) = be^{ax}$

Wir gehen von $m = 5$ Punkten aus:

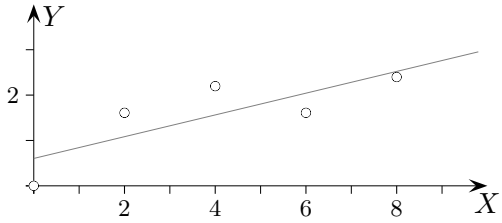
x	y
0	1
2	5
4	9
6	5
8	11

x	y
0	$\ln 1$
2	$\ln 5$
4	$\ln 9$
6	$\ln 5$
8	$\ln 11$



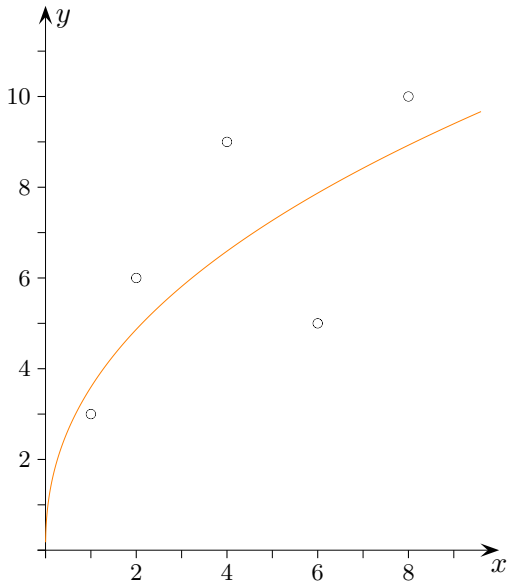
a und b werden mit einer logarithmischen Transformation T bestimmt, $y_i > 0$.

$$
\begin{aligned}
y &= be^{ax} && |\ \ln \\
\ln y &= ax + \ln b \\
T\colon (x, y) &\to (x, \ln y) = (X, Y) \\
Y &= AX + B && |\ T^{-1} \\
\ln y &= Ax + B \\
a &= A && \text{Vergleich 2. und 5. Zeile} \\
\ln b &= B \implies && b = e^B \\
y &= e^B e^{Ax}
\end{aligned}
$$




```
# linear

from math import *
xWerte=[0,2,4,6,8]
yWerte=[log(1),log(5),log(9),log(5),log(11)]

lgs = [[s(0),s(1),t(0)],
       [s(1),s(2),t(1)]]
print(loese_lgs(lgs))
print(loese_lgs(lgs)[0],'+',loese_lgs(lgs)[1],'*x')
print(exp(loese_lgs(lgs)[0]),'* exp(',loese_lgs(lgs)[1],'*x)')
```

Regression mit $f(x) = bx^a$



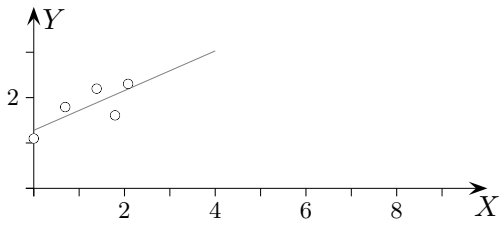
Wir gehen von $m = 5$ Punkten aus:

x	y
1	3
2	6
4	9
6	5
8	10

x	y
$\ln 1$	$\ln 3$
$\ln 2$	$\ln 6$
$\ln 4$	$\ln 9$
$\ln 6$	$\ln 5$
$\ln 8$	$\ln 10$

a und b werden mit einer logarithmischen Transformation T bestimmt, $x_i, y_i > 0$.

$$
\begin{aligned}
y &= bx^a && |\ln \\
\ln y &= a \ln x + \ln b \\
T\colon (x, y) &\to (\ln x, \ln y) = (X, Y) \\
Y &= AX + B && |T^{-1} \\
\ln y &= A \ln x + B \\
a &= A && \text{Vergleich 2. und 5. Zeile} \\
\ln b &= B \implies b = e^B \\
y &= e^B x^A
\end{aligned}
$$




```
from math import *
# Ausgleich mit einer Potenzfunktion
xWerte=[log(1),log(2),log(4),log(6),log(8)]
yWerte=[log(3),log(6),log(9),log(5),log(10)]

lgs = [[s(0),s(1),t(0)],
       [s(1),s(2),t(1)]]
print(loese_lgs(lgs))

print(loese_lgs(lgs)[0],'+',loese_lgs(lgs)[1],'*x')
print(exp(loese_lgs(lgs)[0]),'* x^',loese_lgs(lgs)[1])
```

Ergänzung zum Programm zum Lösen von Gleichungssystemen

Zur Vermeidung der Division durch Null wird das LGS geordnet.
Die 1. Zeile beginnt dann mit einer Null.

```
def lgs_ordnen(lgs):
    for zeile in lgs:
       if zeile[0]!= 0:
          lgs.remove(zeile)
          lgs.insert(0, zeile)
    return lgs
```

alternativ

```
def lgs_ordnen(lgs):
    return sorted(lgs, key = lambda a: a[0], reverse=True)
```

```
def variablen_eliminieren(lgs):
    lgs=lgs_ordnen(lgs)
    ...
```