# 3 Ausgleichs- und Interpolationsrechnung

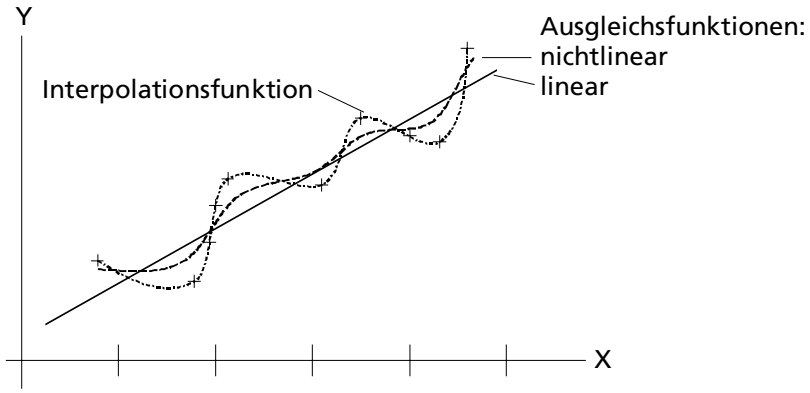
Die Aufgabe der Ausgleichsrechnung ist mit Hilfe einer stetigen Funktion *f(x)*eine bestimmte Menge von gegebenen Datenpunkten mit einem möglichst kleinen Fehler zu beschreiben.
Diese **Ausgleichsfunktion** ist im einfachsten Fall eine lineare Funktion. Wir sprechen dann von einer linearen Regression. Die Praxis erfordert aber oftmals nichtlineare Regressionsfunktionen, da diese die Datenpunkte in der Regel besser ausgleichen.

Die **Interpolationsfunktion** beschreibt mit einer stetigen Funktion eine Menge von Datenpunkten exakt. Der klassische Einsatzzweck ist das Gewinnen von Zwischenwerten zwischen den einzelnen Datenpunkten.

Wir zeigen nachfolgend die Herleitung und Anwendung gebräuchlichen Regressions- und Interpolationsfunktionen. Praktische Beispiele werden mit der Tabellenkalkulation EXCEL durchgerechnet. Ebenso werden programmierte Lösungen der Verfahren in der Sprache C und fallweise als EXCEL-Funktion (Visual-Basic) entwickelt.

## 3.1 Unterschiede Ausgleichsrechnung ↔ Interpolationsrechnung

Beide Begriffe werden (fälschlicherweise) oft synonym verwendet. Wir präzisieren deshalb beide Begriffe:




Ausgleichsrechnung: Bestimmen einer stetigen Ausgleichsfunktion *f(x),* die eine gegebene Menge von *n* Datenpunkten $(x_k,y_k)$, für $k$=1..$n$ optimal annähert (Summe kleinster Fehlerquadrate), in der Regel aber die Funktionswerte nicht exakt darstellen kann.

Interpolationsrechnung: Aus einer Klasse von Funktionen soll eine 'möglichst einfache' Interpolationsfunktion $f(x_k)=y_k$, $k$=1..$n$, bestimmt werden, die exakt durch eine gegebene Menge von *n* Datenpunkten verläuft.

Die Ausgleichsrechnung wird vorwiegend zur näherungsweisen Darstellung von Daten verwendet (Glättung). So können grosse Datenmengen mit einer einfachen stetigen Funktion beschrieben werden, wenn auch mit einem gewissen Fehler.

Mit der Interpolationsrechnung oft die Gewinnung von Zwischenwerten einer (tabellierten) Funktion *f(x)* bezweckt. Ferner wird die Interpolationsrechnung zur Herleitung von numerischen Verfahren (z. Bsp. numerischen Integrationsformeln) benutzt.

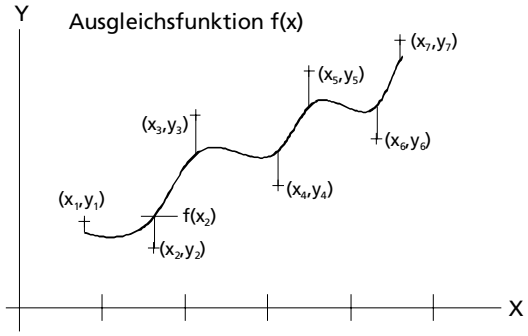
## 3.2 Prinzip der Ausgleichsrechnung

Die Ausgleichsrechnung bestimmt die Parameter einer Funktion, die eine Menge von gegebenen Datenpunkten optimal annähert.
Prinzipiell kann die ausgleichende Funktion eine beliebige stetige Funktion sein. In der Regel beschränkt man sich aber auf lineare Funktionen oder Polynomfunktionen niederen Grades, da diese mit wenig Aufwand zu entwickeln sind. Weniger häufig werden trigonometrische oder Exponentialfunktionen verwendet. Auch Kombinationen sind möglich.

Grundsätzlich wird bei der Ausgleichsrechnung mit einer einfachen Funktion eine grössere Menge an Datenpunkten näherungsweise beschrieben. Deshalb sind im Regelfall mit Ausgleichsfunktionen keine exakte Darstellung möglich.

Die Güte der Näherung wird **Korrelation** genannt und besagt wie gut die Ausgleichsfunktion die Datenwerte approximiert. Ein Mass für die Güte der Anpassung wird durch die Summe der Fehlerquadrate $d_1^2 + d_2^2 + \cdots + d_n^2$ definiert. Ist sie klein ist die Anpassung gut, ist sie gross, ist die Anpassung schlecht.
Für einen linearen Ausgleich wird die Güte in der Regel mit dem Korrelationskoeffizient $r_{xy}$ beschrieben.

Grundsätzlich wird aber bei allen Ausgleichsverfahren die beste Näherung als diejenige bestimmt, die die kleinste Fehlerquadratsumme $q$ liefert:




$$q = \sum_{i=1}^{n} \left( f(x_i) - y_i \right)^2$$

$y_i$: Gegebene Datenwerte
$f(x_i)$: Näherungswerte der Ausgleichsfunktion

**Fehlerquadratsumme** (3.1)

Bemerkung: Für Sonderanwendungen sind auch andere Kriterien möglich. Vgl. hierzu Kapitel Ausgleich mit relativer minimaler Fehlerquadratsumme.

Präzise kann also eine Ausgleichsaufgabe so formuliert werden:.
Zu einer gegebenen Menge von $n$ Datenpunkten $(x_i, y_i)$ ist aus einer vorgegebenen Klasse von Funktionen, diejenige zu bestimmen, deren **Fehlerquadratsumme minimal** ist. Funktionen , welche diese Bedingung erfüllen, heissen **Regressionsfunktionen**.

Für eine lineare Ausgleichsfunktion wird daher die folgende Fehlerquadratsumme minimiert:

$$q = q(a,b) = \sum_{i=1}^{n} \left( ax_i + b - y_i \right)^2 \qquad (3.2)$$

Die Fehlerquadratsumme $q$ ist hierbei von den beiden Parametern $a$, und $b$ abhängig. Ziel der Ausgleichsrechnung ist das Bestimmen der Parameter $a$ und $b$ so, dass die Fehlerquadratsumme minimal wird.

### 3.2.1 Funktionsklassen

Grundsätzlich strebt danach, die Datenpunkte mit möglichst einfachen Funktionen zu beschreiben. Deshalb haben sich folgende Grundfunktionen (Funktionsklassen) bewährt:

$$f(x;a,b) = ax + b \qquad \text{Lineare Funktionen} \tag{3.3}$$

$$f(x;a,b) = a e^{bx} \qquad \text{Exponentialfunktionen} \tag{3.4}$$

$$f(x;a,b) = a x^{b} \qquad \text{Potenzfunktionen} \tag{3.5}$$

Die Parameter *a, b* dieser drei Grundfunktionen können entweder direkt oder über eine geeignete Substitution mit den Formeln für die lineare Ausgleichsfunktion bestimmt werden.

Natürlich können Ausgleichsfunktionen auch mehr als zwei Parameter besitzen. Wichtige Vertreter sind :

$$f(x;a_i) = \sum_{0}^{r} a_i x^i \qquad \text{Polynome vom Grad } r \tag{3.6}$$

$$f(x;a_i,b_i) = \frac{a_0}{2} \sum_{i=1}^{r} \left(a_i \cos(ix) + b_i \sin(ix)\right) \qquad \text{trigonometrische Polynome} \tag{3.7}$$

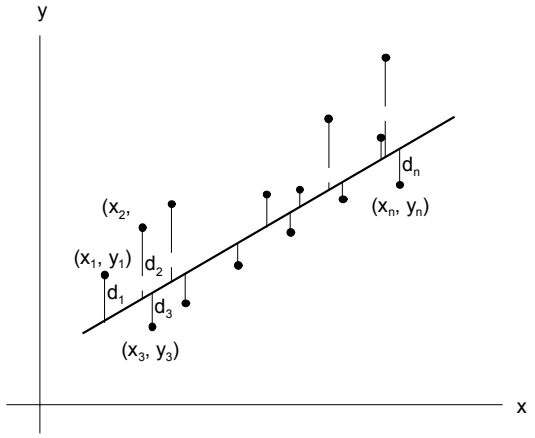
### 3.2.2 Lineare Ausgleichsrechnung

Die lineare Ausgleichsrechnung (lineare Regression) beschreibt mit einer linearen Funktion

$$y_i = ax_i + b \qquad (i = 1,\ldots,n) \tag{3.8}$$

**Lineare Ausgleichsfunktion**

eine Menge von *n* Datenpunkten optimal im Sinne der kleinsten Summe der Fehlerquadrate:




Die Fehlerquadrate werden in den **vorgegebenen Datenpunkten** $(x_i, y_i)$ als quadrierte Differenz bestimmt:

$$d_i = \left(f(x_i) - y_i\right)^2 = \left(ax_i + b - y_i\right)^2 \qquad (i = 1,\ldots,n) \tag{3.9}$$

Die Fehlerquadratsumme wird dann:

$$\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} \left(f(x_i) - y_i\right)^2 = \sum_{i=1}^{n} \left(\left(ax_i + b\right) - y_i\right)^2 \tag{3.10}$$

Die Parameter *a* und *b* werden so bestimmt, dass die Forderung nach Minimierung der Fehlerquadrate erfüllt wird. Dazu werden die partiellen Ableitungen $\frac{\partial q}{\partial a}, \frac{\partial q}{\partial b}$ null gesetzt und nach *a* und *b* aufgelöst:

$$\frac{\partial q}{\partial a} = \frac{\partial}{\partial a}\sum_{i=1}^{n}(ax_i + b - y_i)^2 = 2\sum_{i=1}^{n} x_i(ax_i + b - y_i) = 0 \qquad (n = \text{Anzahl Datenpunkte})$$

$$\rightarrow \qquad a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i$$

$$\frac{\partial q}{\partial b} = \frac{\partial}{\partial b}\sum_{i=1}^{n}(ax_i + b - y_i)^2 = 2\sum_{i=1}^{n}(ax_i + b - y_i) = 0$$

$$\rightarrow \qquad a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} 1 = \sum_{i=1}^{n} y_i$$

An Stelle der umständlichen Notation der Summen mit den Indizes schreiben wir:

$$\sum_{i=1}^{n} x_i = \sum X \qquad \sum_{i=1}^{n} x_i^2 = \sum X^2 \qquad \sum_{i=1}^{n} y_i = \sum Y \qquad \sum_{i=1}^{n} x_i y_i = \sum XY$$

Diese kompakte Schreibweise ist allgemein üblich. Sie vereinfacht die Darstellung der vorhergehenden Formeln:

$$\begin{aligned} b\sum 1 \; + a\sum X \; &= \sum Y \\ b\sum X + a\sum X^2 &= \sum XY \end{aligned} \qquad (3.11)$$

Die Summen in diesem Gleichungssystem verkörpern gegebene Zahlenwerte und stellen die Koeffizienten dar. Die Unbekannten *a, b* können durch geeignete Umformungen bestimmt werden.

**Lineare Ausgleichsfunktion**

$$b = \frac{\left(\sum Y\right)\left(\sum X^2\right) - \left(\sum X\right)\left(\sum XY\right)}{n\sum X^2 - \left(\sum X\right)^2} \qquad (3.12)$$

$$a = \frac{n\sum XY - \left(\sum X\right)\left(\sum Y\right)}{n\sum X^2 - \left(\sum X\right)^2} \qquad (3.13)$$

*a,b* verkörpern die Parameter der Ausgleichsgeraden $y_i \approx f(x_i) = ax_i + b$

**Beispiel:**
Zu bestimmen ist die lineare Regressionsfunktion für die Datenpunkte:

| x | 2 | 4 | 5 | 1 |
|---|---|---|---|---|
| y | 2 | 1 | 2 | 0 |

Wir führen die Rechnung von Hand durch und erhalten:

$$a = \frac{n\sum XY - (\sum X)(\sum Y)}{n\sum X^2 - (\sum X)^2} = \frac{4 \cdot 18 - 12 \cdot 5}{4 \cdot 46 - 12^2} = 0.3$$

$$b = \frac{(\sum Y)(\sum X^2) - (\sum X)(\sum XY)}{n\sum X^2 - (\sum X)^2} = \frac{5 \cdot 46 - 12 \cdot 18}{4 \cdot 46 - 12^2} = 0.35$$

$$\Rightarrow \quad \underline{\underline{y(x) = 0.3x + 0.35}}$$

Bemerkung:
Die lineare Ausgleichsfunktion kann auch über die Standardabweichung $s_x$ und Kovarianz $c_{xy}$ bestimmt werden. Diese Methode wird im Kapitel Statistik kurz vorgestellt.

Wir kennen nun die lineare Ausgleichsfunktion für die gegebenen Datenpunkte. Jedoch können wir keine Aussage machen wie 'gut' die Gerade ausgleicht, d.h. die Gerade die Datenpunkte beschreibt.

### 3.2.2.1 Korrelationskoeffizient

Der (lineare) Korrelationskoeffizient $r_{xy}$ dient zur Beurteilung des linearen Zusammenhanges von n Paaren $(x_i, y_i)$. Der Korrelationskoeffizient liegt immer in [-1,1] und ist wie folgt zu interpretieren:

Ist $r_{xy}=0$ so sind die Werte unkorreliert, d.h. zwischen $x_i$ und $y_i$ besteht keine lineare Beziehung. Ist hingegen $r_{xy}$ nahe bei $|1|$ so sprechen wir von starker Korrelation. Bei $r_{xy}=|1|$ liegen alle Punkte auf der Regressionsgeraden.

Wir definieren den Korrelationskoeffizienten:

$$r_{xy} = \frac{n\sum XY - (\sum X)(\sum Y)}{\sqrt{\left(n\sum X^2 - (\sum X)^2\right)\left(n\sum Y^2 - (\sum Y)^2\right)}}$$

**Korrelations-koeffizient** (3.14)

Eine Herleitung für $r_{xy}$ wird im Kapitel Statistik ausführlich gezeigt.

**Beispiel:**
Bestimmung des linearen Korrelationskoeffizienten $r_{xy}$ für die vorherige lineare Regressionsaufgabe $f(x) = 0.3x + 0.35$:

| x | 2 | 4 | 5 | 1 |
|---|---|---|---|---|
| y | 2 | 1 | 2 | 0 |

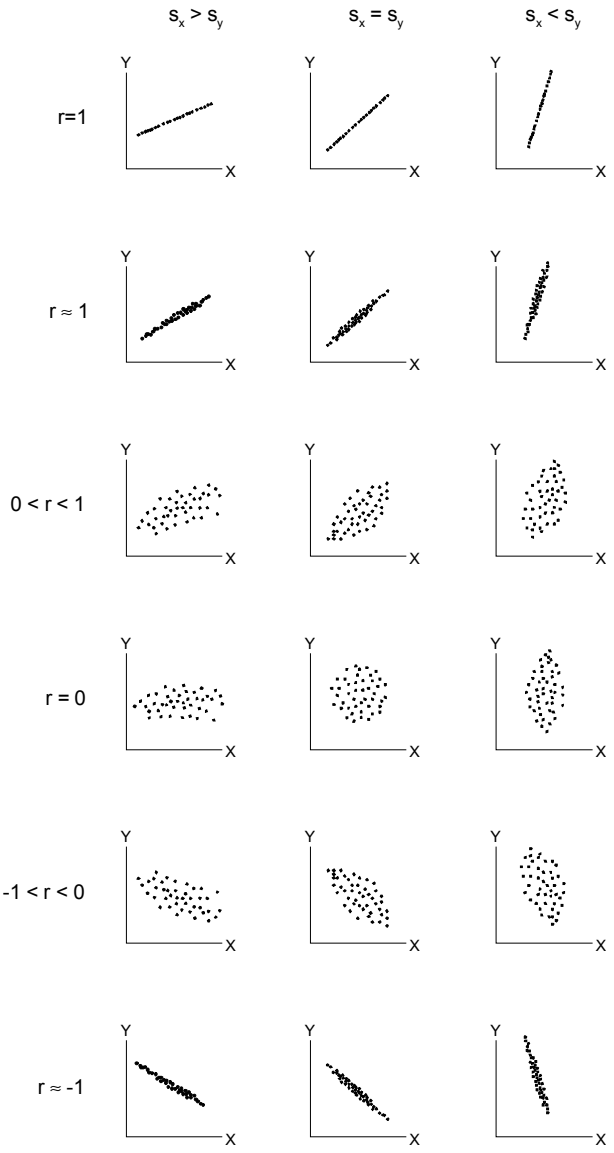
$$\sum Y^2 = 4 + 1 + 4 + 0 = 9$$

$$r_{xy} = \frac{n\sum XY - (\sum X)(\sum Y)}{\sqrt{\left(n\sum X^2 - (\sum X)^2\right)\left(n\sum Y^2 - (\sum Y)^2\right)}} = \frac{4 \cdot 18 - 12 \cdot 5}{\sqrt{\left(4 \cdot 46 - 12^2\right)\left(4 \cdot 9 - 5^2\right)}} = 0.57207...$$

Wir interpretieren diese Zahl als mässige bis schlechte Korrelation. Gute Korrelationskoeffizienten liegen bei nahe bei $|1|$.

### 3.2.2.2 Grafische Interpretation

Nachfolgend einige typische Punktediagramme mit ihren zugehörigen Korrelationskoeffizienten für verschieden verteilte Merkmalswerte $X$ und $Y$:




Typische Punktediagramme mit ihren zugehörigen Korrelationskoeffizienten für verschieden verteilte Merkmalswerte $X$ und $Y$

### 3.2.2.3 Algorithmus für lineare Regression

Wir machen einen Entwurf für ein einfaches Programm zur linearen Regression in C. Ausgehend von einer fiktiven Aufgabenstellung, welche eine bestimmte Menge Datenpunkte $(x_i, y_i)$ liefert und beschreiben zuerst auf einer höheren Ebene die auszuführenden Tätigkeiten:

Eingaben: Eingabe der Anzahl Datenpunkte $n$.
Sukzessive Eingabe von $n$ Wertepaaren $(x_i, y_i)$.

Berechnungen: Für die Koeffizientenformeln müssen bestimmt werden:
$\sum X, \sum X^2, \sum XY, \sum Y, \sum Y^2$
Nach dem Bestimmen dieser Summen werden $a$, $b$ und $r_{xy}$ berechnet.

Ausgaben: Koeffizienten $a$, $b$ und $r_{xy}$ ausgeben und Programm beenden.

Eine mögliche Implementierung wäre:

```
/* Lineare Regression:                                                    File: LINREG.C

    Bestimmen der linearen Regressionsfunktion yi=axi + b und des Korellationskoeffizienten.

    Autor: Gerhard Krucker
    Datum: 14.8.1995
    Sprache: MS Visual-C V1.5  (QuickWin Application)
*/

#include <stdio.h>
#include <math.h>           /* Fuer sqrt() */

main()
{  int n;                              /* Aktuelle Anzahl Datenpunkte */
   double x, y;                   /* Werte des aktuellen Datenpunktes */
   double sx, sxx, sxy, sy, syy;  /* Summenwerte */
   double a, b , rxy;                  /* Regressionskoeffizienten und Korellationskoeffizient */
   int i;

   printf("Bestimmen der linearen Regressionsfunktion aus n Datenwertepaaren:\n"
          "Anzahl Datenwerte (n): ");
   scanf("%d",&n);

   sx= sxx = sxy = sy = syy = 0.0;    /* Summen alle Nullsetzen */
   for (i=1; i <= n ; i++)
    { printf("\nPunkt %d x-Wert: ",i); scanf("%lf",&x);
      printf("Punkt %d y-Wert: ",i); scanf("%lf",&y);

      sx += x; sxx += x * x;         /* Wert aufsummieren */
      sy += y; syy += y * y;
      sxy += x * y;
    }

   a = (n * sxy - sx * sy) / (n * sxx - sx * sx);     /* Koeffizienten berechnen */
   b = (sy * sxx - sx * sxy)/ (n * sxx - sx * sx);
   rxy = (n * sxy - sx * sy)/ sqrt((n * sxx - sx * sx)*(n * syy - sy * sy));
   printf("\nAusgleichsgerade:\na = %g\nb = %g\nrxy = %4.3g\n",a, b, rxy);

   return 0;
}
```

**Beispiel:**
Wir wollen für die folgenden Datenpunkte $(x_i, y_i)$ eine lineare Ausgleichsfunktion bestimmen:

| x | 1.0 | 2.0 | 2.5 | 3.0 |
|---|---|---|---|---|
| y | 3.7 | 4.1 | 4.3 | 5.0 |

Wir erhalten mit diesen Werten folgenden Programmlauf:

```
Bestimmen der linearen Regressionsfunktion aus n Datenwertepaaren:
Anzahl Datenwerte (n): 4

Punkt 1 x-Wert: 1.0
Punkt 1 y-Wert: 3.7

Punkt 2 x-Wert: 2.0
Punkt 2 y-Wert: 4.1

Punkt 3 x-Wert: 2.5
Punkt 3 y-Wert: 4.3

Punkt 4 x-Wert: 3.0
Punkt 4 y-Wert: 5.0

Ausgleichsgerade:
a = 0.6
b = 3
rxy = 0.942
```

und somit die Ausgleichsgerade $y = f(x) = 0.6x + 3$. Der Korrelationskoeffizient $r_{xy}$=0.942 zeigt, dass die Ausgleichsgerade recht gut die Datenpunkte nähert. Eine grafische Interpretation des Resultates folgt in der Lösung mit der Tabellenkalkulation.

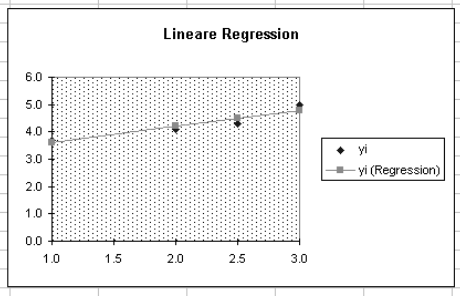
### 3.2.2.4 Linearer Ausgleich mit Tabellenkalkulationsprogrammen

In handelsüblichen Tabellenkalkulationsprogrammen, wie EXCEL, Lotus, etc. gehört die lineare Ausgleichsrechnung zum standardmässigen Funktionsumfang. Nach Eingabe der Daten kann direkt die Regressionsgerade bestimmt werden.

Wir möchten nun zeigen wie einerseits die Regressionsgerade direkt mit Standardfunktionen bestimmt

werden kann und wie wir ohne grossen Mehraufwand die Koeffizienten über die Formeln (3.12, 3.13) bestimmen:

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Lineare Regressionsfunktion** | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | |
| 3 | xi | 1.0 | 2.0 | 2.5 | 3.0 | | | | | | | | |
| 4 | yi | 3.7 | 4.1 | 4.3 | 5.0 | | | | | | | | |
| 5 | | | | | | | | | | | | | |
| 6 | Direkte Rechnung mit EXCEL-Funktion: | | | | | | | | | | | | |
| 7 | a: | 0.6 | | | | | | | | | | | |
| 8 | b: | 3 | | | | | | | | | | | |
| 9 | rxy: | 0.942 | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | |
| 12 | Formale Rechnung: | | | | | | | | | | | | |
| 13 | | | | | | Σ | | | | | | | |
| 14 | x | 1.0 | 2.0 | 2.5 | 3.0 | 8.5 | | | | | | | |
| 15 | x2 | 1 | 4 | 6.25 | 9 | 20.3 | | | | | | | |
| 16 | y | 3.7 | 4.1 | 4.3 | 5.0 | 17.1 | | | | | | | |
| 17 | y2 | 13.69 | 16.81 | 18.49 | 25 | 74.0 | | | | | | | |
| 18 | xy | 3.7 | 8.2 | 10.75 | 15 | 37.7 | | | | | | | |
| 19 | | | | | | | | | | | | | |
| 20 | n | 4.0 | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | |
| 22 | a: | 0.6 | | | | | | | | | | | |
| 23 | b: | 3 | | | | | | | | | | | |
| 24 | rxy: | 0.942 | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | |




Die Ausgangsdaten werden tabellarisch in B3:E4 aufgeführt. Sie dienen als Grundlage für alle Berechnungen.
Die Bestimmung der linearen Regressionsfunktion $f(x)=ax + b$ erfolgt in EXCEL über separate Funktionen zur Bestimmung von Achsenabschnitt $b$ und Steigung $a$. Damit werden die Zellinhalte für die Resultate:

```
B7:   =STEIGUNG(B4:E4;B3:E3)
B8:   =ACHSENABSCHNITT(B4:E4;B3:E3)
B9:   =KORREL(B3:E3;B4:E4)
```

Die Berechnung ohne Standardfunktionen erfolgt analog der Berechnung im C-Programm. Wir bestimmen zuerst die Summen $\sum X, \sum X^2, \sum Y, \sum Y^2, \sum XY$ und wenden nachher die Formeln (3.12, 3.13) an. Die hierzu notwendigen Formeln zur Berechnung der Formeln:

```
B20:   =ANZAHL(B14:E14)
B22:   =(B20 * F18 - F14 * F16) / (B20 * F15 - F14 * F14)
B23:   =(F16 * F15 - F14 * F18) / (B20 * F15 - F14 * F14)
B24:   =(B20 * F18 - F14 * F16) / WURZEL((B20 * F15 - F14 * F14)*(B20 * F17 - F16 * F16))
```

Beide Verfahren liefern natürlich dasselbe Resultat. Ein Blick in die Hilfestellung dieser zeigt, dass sie dieselben Formeln verwenden. Trotzdem ist die Verwendung der Standardfunktion bequemer, da wesentlich weniger Rechenaufwand anfällt und dadurch auch Fehlerquellen ausgeschaltet werden.

### 3.2.3 Exponentieller Ausgleich

Erreichen wir mit einem linearen Ausgleichsverfahren nur einen schlechten Korrelationskoeffizienten $r_{xy}$, so heisst dies nicht, dass die Datenwerte mit einer nichtlinearen Ausgleichsfunktion gut korrelieren.

Wenn schon zum vornherein ein nichtlinearer funktioneller Zusammenhang feststeht, ist eine lineare Regression unzweckmässig. Ein Beispiel dafür wäre die Messung von 10 Stromwerten im Bereich [0,0.8]V durch eine Diode in Vorwärtsrichtung.

Der wichtigste Vertreter der nichtlinearen Ausgleichsfunktion ist die exponentielle Ausgleichsfunktion

$$y = b\, e^{ax} \qquad \textbf{Exponentielle Ausgleichsfunktion} \qquad (3.15)$$

Die Bestimmung der Parameter $a$ und $b$ erfolgt durch eine logarithmische Transformation. Unter der Voraussetzung, dass alle $y_i > 0$ gilt:

$$Y = \ln y = \ln b e^{ax} = a \cdot x + \ln b = A \cdot X + B \qquad \begin{array}{l} B = \ln b \quad \rightarrow \quad b = e^B \\ A = a \\ X = x \end{array} \qquad (3.16)$$

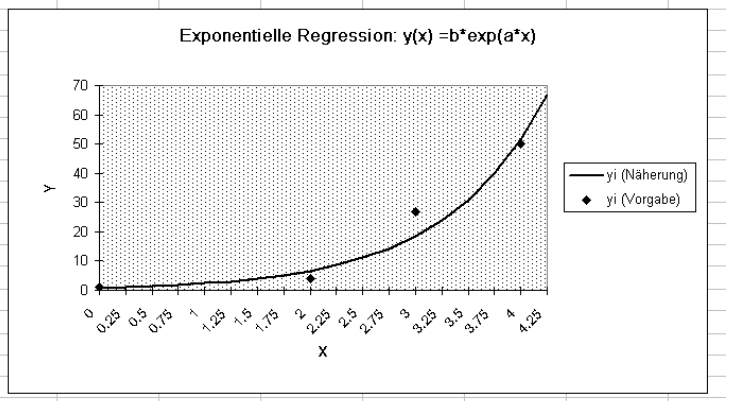
womit wir die Aufgabe in eine lineare Ausgleichsrechnung übergeführt haben: Die exponentielle Ausgleichskurve im *(x,y)*-System wurde in eine lineare Ausgleichskurve im *(x,* ln *y)*-System transformiert.

Nach der Transformation der *y*-Werte können wir das lineare System mit den bekannten Formeln (3.12, 3.13) problemlos lösen. Die erhaltenen Koeffizienten werden nachher zurück transformiert und wir erhalten die gesuchte exponentielle Ausgleichsfunktion.

**Beispiel:**
Die nachfolgenden 4 Datenpunkte $(x_i, y_i)$ sind durch eine exponentielle Ausgleichskurve $y_i = b\, e^{ax_i}$ optimal anzunähern:

| x | 0 | 2 | 3 | 4 |
|---|---|---|---|---|
| y | 1 | 4 | 27 | 50 |

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Exponentielle Ausgleichsfunktion | | | | |
| 2 | | | | | |
| 3 | Ausgangsdaten: | | | | |
| 4 | xi | 0 | 2 | 3 | 4 |
| 5 | yi | 1 | 4 | 27 | 50 |
| 6 | | | | | |
| 7 | Berechnungen: | | | | |
| 8 | ln y | 0.0000 | 1.3863 | 3.2958 | 3.9120 |
| 9 | | | | | |
| 10 | Lineare Regressionsfunktion (transformierte): | | | | |
| 11 | A: | 1.0253 | | | |
| 12 | B: | -0.1584 | | | |
| 13 | | | | | |
| 14 | Rücktransformation: | | | | |
| 15 | a: | 1.0253 | | | |
| 16 | b: | 0.8535 | | | |
| 17 | | | | | |
| 18 | | | | | |
| 19 | Diagrammwerte: | | | | |
| 20 | xi | 0 | 2 | 3 | 4 |
| 21 | yi (Vorgabe) | 1 | 4 | 27 | 50 |
| 22 | yi (Näherung) | 0.85 | 6.63 | 18.50 | 51.56 |
| 23 | | | | | |



### 3.2.4 Ausgleich mit Potenzfunktionen

Eine weitere wichtige Klasse von nichtlinearen Ausgleichsfunktionen wird durch die Potenzfunktionen dargestellt:

$$y = b \cdot x^a \tag{3.17}$$

Unter der Voraussetzung $x_i, y_i > 0$ können wir auch hier eine logarithmische Transformation durchführen und wir erhalten wiederum eine lineare Ausgleichsaufgabe:

$$Y = \ln y = a \ln x + \ln b = A \cdot X + B \qquad \begin{matrix} B = \ln b & \rightarrow & b = e^B \\ A = a & & \\ X = \ln x & & \end{matrix}$$

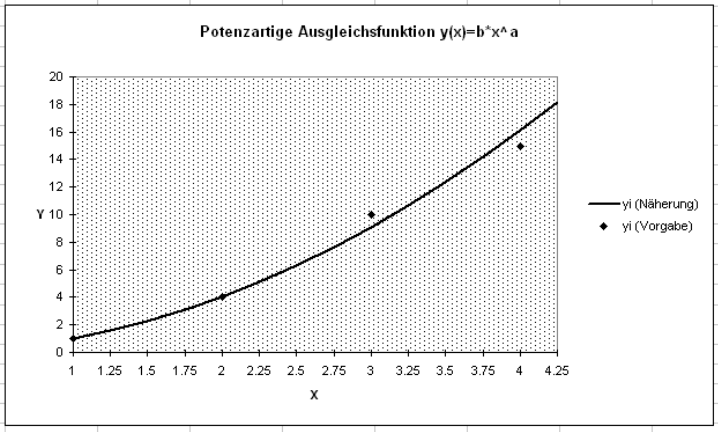
**Potenzartige Ausgleichsfunktion** (3.18)

Die durch die Potenzfunktion beschriebenen Punkte im $(x_i, y_i)$-System werden durch die Transformation über eine lineare Ausgleichsfunktion im $(\ln x_i, \ln y_i)$-System beschrieben.

**Beispiel:**
Die nachfolgenden 4 Datenpunkte $(x_i, y_i)$ sind durch eine potenzartige Ausgleichskurve $y_i = b\, x_i^a$ optimal anzunähern:

| x | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| y | 1 | 4 | 10 | 15 |

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Potenzartige Ausgleichsfunktion | | | | |
| 2 | | | | | |
| 3 | Ausgangsdaten: | | | | |
| 4 | xi | 1 | 2 | 3 | 4 |
| 5 | yi | 1 | 4 | 10 | 15 |
| 6 | | | | | |
| 7 | Berechnungen: | | | | |
| 8 | ln x | 0.0000 | 0.6931 | 1.0986 | 1.3863 |
| 9 | ln y | 0.0000 | 1.3863 | 2.3026 | 2.7081 |
| 10 | | | | | |
| 11 | Lineare Regressionsfunktion (transformierte): | | | | |
| 12 | A | 1.9943 | | | |
| 13 | B | 0.0147 | | | |
| 14 | | | | | |
| 15 | Rücktransformation: | | | | |
| 16 | a | 1.9943 | | | |
| 17 | b | 1.0148 | | | |
| 18 | | | | | |
| 19 | | | | | |
| 20 | Diagrammwerte: | | | | |
| 21 | xi | 1 | 2 | 3 | 4 |
| 22 | yi (Vorgabe) | 1 | 4 | 10 | 15 |
| 23 | yi (Näherung) | 1.01 | 4.04 | 9.08 | 16.11 |
| 24 | | | | | |




### 3.2.5 Ausgleich mit Polynomfunktionen

Prinzipiell können ausser der linearen Ausgleichsfunktion auch Polynomfunktionen höheren Grades verwendet werden. In der Regel erlauben Polynomfunktionen höheren Grades eine bessere Näherung als lineare Funktionen, jedoch wird die Berechnung der Ausgleichsfunktion entsprechend aufwendiger.

Das Haupteinsatzgebiet von polynomialen Ausgleichsfunktionen ist die **Glättung von Messwerten**. Prinzipiell lässt sich durch Steigerung des Polynomgrades eine beliebig genaue Ausgleichsfunktion bestimmen, bis für *n* Datenpunkte ein Polynom vom Grad *n-1* die Funktion exakt approximiert.

Zu beachten ist, dass Polynomfunktionen höheren Grades (>4) unüblich sind. Insbesondere neigen solche Ausgleichsrechnungen zu instabilen Verhalten: Die Funktion ist 'nervös', zeigt Überschwinger und ist ungeeignet für Zwischenwerte zu bestimmen (vgl. auch Kapitel "Interpolationsfunktionen").

Grundsätzlich ist polynomiale Ausgleichsfunktion als normale Polynomfunktion vom Grad *n* definiert:

$$y(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{3.19}$$

Diese Definition beinhaltet auch die lineare Regressionsfunktion wenn $n$=1 gesetzt wird.

Die Bestimmung der Koeffizienten $a_0,...,a_n$ erfolgt in bekannter Weise durch Minimieren der Fehlerquadratsumme. Für eine Polynomfunktion vom Grad *n* lautet die Fehlerquadratsumme:

$$q(a_0, a_1, \ldots, a_n) = \sum_{k=1}^{m} \left( \sum_{i=0}^{n} a_i x_k^i - y_k \right)^2 \tag{3.20}$$

*m*: Anzahl Datenpunkte
*n*: Grad des Ausgleichspolynoms

Die Koeffizienten $a_i$ für eine minimale Fehlerquadratsumme werden über Nullsetzen der partiellen Ableitungen $\frac{\partial q}{\partial a_i}$ für $i=0,..,n$ bestimmt.

$$\frac{\partial q}{\partial a_i} = 2 \sum_{k=1}^{m} x_k^i \left( \sum_{i=0}^{n} a_i x_k^i - y_k \right) = 0 \tag{3.21}$$

Wir erhalten dann durch Auflösen der *i*-Summen die Normalgleichungen:

$$\sum_k x_k^i a_0 + \sum_k x_k^{i+1} a_1 + \cdots + \sum_k x_k^{i+j} a_j + \cdots + \sum_k x_k^{i+n} a_n = \sum_k x_k^i y_k \qquad i: 0,\ldots,n \tag{3.22}$$

Dies ist ein lineares Gleichungssystem mit *n+1* Bestimmungsgleichungen und *n+1* Unbekannten. Die Unbekannten stellen hier die Koeffizienten $a_0,...,a_n$ dar.

Die Koeffizientenmatrix *G* hat die Elemente: $g_{ij} = \sum_k x_k^{i+j} \qquad i,j: 0,\ldots,n$ (3.23)

Die Elemente des Konstantenvektor $\underline{c}$ (rechte Seite des Systems): $c_i = \sum_k x_k^i y_k \qquad i: 0,\ldots,n$ (3.24)

Somit lautet das zu lösende Gleichungssystem:

$$G \cdot \underline{a} = \underline{c} \tag{3.25}$$

Dieses System wird mit den bekannten Verfahren gelöst und wir erhalten als Resultat die gesuchten Polynomkoeffizienten.

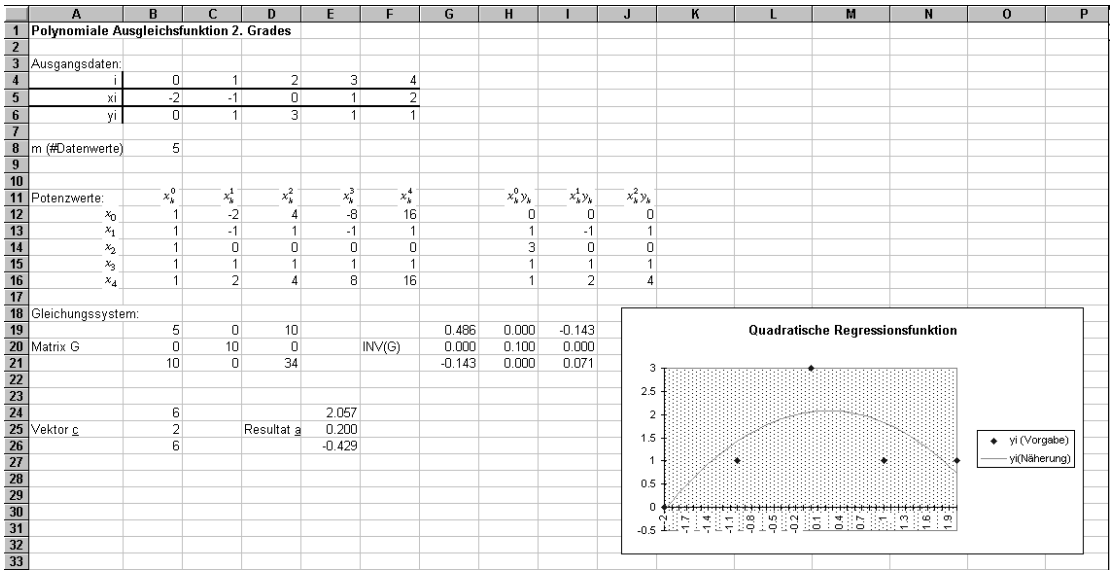
Auf den Entwurf einer programmierten Lösung wird hier verzichtet. Die Implementierung ist aber an sich problemlos. Nach dem Aufbauen der Koeffizientenmatrix wird das System mit einem geeigneten Verfahren (z.B. Gauss-Jordan) gelöst.

Stattdessen wollen wir ein Beispiel mit EXCEL durchrechnen. Sicherlich ist die Tabellenkalkulation für eine direkte Rechnung nicht die erste Wahl. Sie erlaubt aber alle berechneten Werte sauber darzustellen und die Werte grafisch zu präsentieren.

Im Rahmen einer Übung zeigen wir, wie man eine allgemeine EXCEL-Funktion für polynomialen Ausgleich in Visual BASIC definieren kann. Diese Lösung wird aus einer Lösung in der Sprache C abgeleitet.

**Beispiel:**
Die nachfolgenden 5 Datenpunkte $(x_i, y_i)$ sind durch ein quadratisches Ausgleichspolynom $y_i = a_0 + a_1 x + a_2 x^2$ optimal anzunähern:

| $x$ | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| $y$ | 0 | 1 | 3 | 1 | 1 |




Lösen wir nun dieselbe Aufgabe mit Ausgleichspolynomen unterschiedlichen Grades, so erhalten wir folgende Werte:

| deg $p$ | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $q$ |
|---|---|---|---|---|---|---|
| 1 | 1.2 | 0.2 | - | - | - | 4.40 |
| 2 | 2.057 | 0.2 | -0.429 | - | - | 1.83 |
| 3 | 2.057 | -0.083 | -0.428 | -0.083 | - | 1.73 |
| 4 | 3 | -0.083 | -2.458 | -0.083 | -0.458 | 0 |

Das letzte Polynom hat exakt so viele Koeffizienten wie Datenpunkte. Dadurch kann das Polynom alle Datenpunkte $(x_i, y_i)$ exakt darstellen (Fehlerquadratsumme $q = 0$) und ist somit ein Interpolationspolynom.

Wie bereits erwähnt, spielen in der Praxis Ausgleichspolynome mit Grad > 4 kaum eine Rolle. Einerseits ist die Rechnung aufwendig, andererseits neigen Polynomfunktionen höheren Grades zum Überschwingen. Die Überschwinger bewirken, dass die Funktion nicht mehr geeignet ist zur Gewinnung von Zwischenwerten.

Sollen Funktionen dennoch mit Polynomfunktionen niedrigen Grades gut ausgeglichen werden, so kann dies durch stückweise Entwicklung stetiger Polynomfunktionen geschehen (Bsp.: Spline-Funktionen, Bezier-Kurven).
Hier wird das Intervall für die Berechnung aufgeteilt, so die Näherung mit mehreren Polynom-Teilfunktionen erfolgt. Jede dieser Teilfunktionen nähert eine kleine Menge von Datenpunkten in einem Teilintervall. Über geeignete Methoden sorgt man dafür, dass die Stossstellen der Teilfunktionen glatt verlaufen.

### 3.2.5.1 Rechenbeispiel zu polynomialer Ausgleichung

Bestimmen Sie die Ausgleichspolynome vom Grad 2 und 3, skizzieren Sie die Interpolationspolynome auf dem Intervall [-2,3] und bestimmen Sie die jeweiligen Fehlerquadratsummen.

| $x$ | -2 | -1 | 0 | 1 | 3 |
|---|---|---|---|---|---|
| $y$ | 0 | 0.5 | 1.5 | 0.4 | 0 |

Vorgehen:
Wir berechnen zuerst das Ausgleichspolynom 3. Grades. Hierbei fallen auch alle notwendigen Werte zur Bestimmung des Polynoms 2. Grades an.

Dabei werden die Koeffizientenmatrix erzeugt, indem die Summen gemäss (3.23) und (3.24) gebildet werden.
Anschliessend wird (mit dem Taschenrechner) die Koeffizientenmatrix invertiert und die Koeffizienten $a_0 ... a_3$ bestimmt. Durch Einsetzen der Werte in (3.20) wird die Fehlerquadratsumme bestimmt.

Dasselbe wird für das Polynom 2. Grades wiederholt, wobei die bereits berechneten Grössen benutzt werden.

Wir erstellen das Arbeitsblatt mit EXCEL indem wir zuerst die Vorgabedaten eintragen und nachher schrittweise die Grössen für $G$, $\underline{c}$ und $\underline{a}$ bestimmen:

### 3.2.5.2 Berechnung mit benutzerdefinierten EXCEL-Funktionen

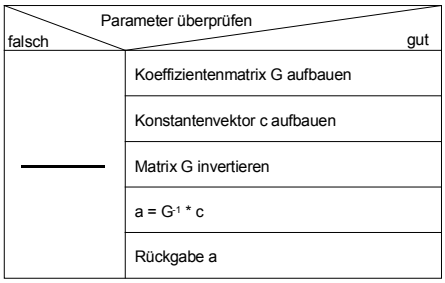
Wir haben gesehen, dass die Berechnung der Polynomkoeffizienten auch für ein einfaches Beispiel recht umfangreich wird.
EXCEL bietet neben den eigentlichen Kalkulationsfähigkeit auch die Möglichkeit benutzerdefinierte Funktionen in Visual BASIC for Applications (VBA) zu implementieren. Diese Funktionen können nachher wie normale EXCEL Funktionen benutzt werden.

Ohne näher auf die Sprache VBA einzugehen (das folgende Beispiel sollte in grossen Teilen intuitiv verständlich sein), wollen wir zeigen, wie eine Funktion zur Bestimmung der Koeffizienten für den polynomialen Ausgleich entwickelt wird, sowie eine Funktion zur Auswertung von Polynomen nach dem Horner-Schema:

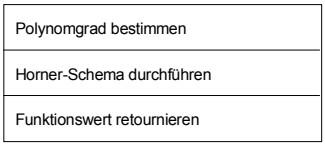
| | |
|---|---|
| Grundlage: | Formale Zusammenhänge (3.23) ,(3.24) und (3.25). |
| Anforderungsprofil: | Benutzerdefinierte Funktion in VBA mit dem Namen PolynomReg, die als Argumente zwei Listen mit den auszugleichenden $x$- und $y$-Werte erhält, sowie den Grad des zu bestimmenden Ausgleichspolynoms. Das Resultat soll als Array mit doppelt genauen Gleitkommazahlen retourniert werden (sog. Array-Funktion).<br><br>Benutzerdefinierte Funktion mit dem Namen PolynomEval, die als Argumente eine Liste (Array) mit den Koeffizienten $a_0..,a_n$ erhält, sowie die auszuwertende Stelle $x$. Alle Daten sind doppelt genaue Gleitkommazahlen. Das Resultat wird als Funktionswert in einer doppelt genauen Gleitkommazahl retourniert. |
| Datentypen: | Polynomgrad: Ganzzahl<br>Datenpunktelisten x, y: Eindimensionales Array mit doppelt genauen Gleitkommazahlen<br>Auszuwertende Stelle x: Doppelt genaue Gleitkommazahl |

## Ablaufdiagramme:

PolynomReg(x;y;Polynomgrad)




PolynomEval(a; x Als Doppelt)




## Der Code lautet in Visual Basic for Applications:

```
'Berechnen der Polynomkoeffizienten a0,..,an für eine polynomiale Ausgleichsfunktion     'File: AUSGL32.XLA'
' n-ten Grades für m Datenpunkte.
' Parameter: x = Array mit x-Werten (Anzahl: m, beliebig)
'            y = Array mit y-Werten (Anzahl: m, beliebig)
'            n = Grad des zu erzeugenden Ausgleichspolynoms
'
' Das Resultat wird als Funktionswert (Arrayfunktion) retourniert
' Autor: Gerhard Krucker
' Datum: 17.8.1995, 22. 9. 1996
' Sprache: VBA for EXCEL7
'
Function PolynomReg(x, y, Polynomgrad)
Dim AnzX, AnzY     ' Anzahl x- und y-Werte
Dim m              ' Anzahl auszugleichender Datenpunkte'
Dim Sxk()          ' Dynamisches Array fuer die Summe der Potenzen von xk '
Dim Sxkyk()        ' Dynamisches Array fuer die Summe der Potenzen von xk * yk '
Dim G(), g1        ' Dynamisches Array fuer die Koeffizientenmatrix G'
Dim c()            ' Dynamisches Array fuer den Konstantenvektor c'
Dim a()            ' Dynamisches Array fuer die Polynomkoeffizienten a0,..,an'
Dim i, j, k

  ' Parameterkontrollen'
  If (Polynomgrad < 1) And Polynomgrad <> "Integer" Then
     PolynomReg = CVErr(xlErrValue)
     Exit Function
  End If
  AnzX = x.Count  ' Anzahl Datenpunkte in den Arrays bestimmen '
  AnzY = y.Count
  If (AnzX <> AnzY) Then
     PolynomReg = CVErr(xlErrValue)
     Exit Function
  End If
  If AnzX <= Polynomgrad Then
     Exit Function
  End If
  m = AnzX

  'Summe der Potenzen xk und xk*yk berechnen und den entprechende Arrays abspeichern'
  ReDim Sxk(Polynomgrad * 2)     ' Arrays auf passende Groesse dimensionieren '
  ReDim Sxkyk(Polynomgrad)       ' Die Arrayindizes laufen von 0..Polynomgrad, resp 0..2*Polynomgrad'
  For i = 0 To 2 * Polynomgrad
     Sxk(i) = 0
     For k = 1 To m              ' Fuer jeden Datenpunkt'
       Sxk(i) = Sxk(i) + x(k) ^ i
     Next k
  Next i
  For i = 0 To Polynomgrad
     Sxkyk(i) = 0
     For k = 1 To m
       Sxkyk(i) = Sxkyk(i) + x(k) ^ i * y(k)
     Next k
  Next i

  'Koeffizientenmatrix G und Konstantenvektor c erzeugen'
  ReDim G(1 To Polynomgrad + 1, 1 To Polynomgrad + 1)  'Matrix mit Indizes 0..Polynomgrad,0..Polynomgrad
                                                        dimensionieren'
  ReDim g1(1 To Polynomgrad + 1, 1 To Polynomgrad + 1) 'Matrix fuer die Inverse von G (MINV kann nicht in G
                                                        zurueckschreiben)'
  ReDim c(1 To Polynomgrad + 1)
  ReDim a(0 To Polynomgrad)                            'Polynomkoeffizienten a0,..,an (a(0) = a0) '

  For i = 0 To Polynomgrad                             'Koeffizientenmatrix G und Konstantenvektor c aufbauen '
     For j = 0 To i
        G(i + 1, j + 1) = Sxk(i + j)
        G(j + 1, i + 1) = Sxk(i + j)
     Next j
     c(i + 1) = Sxkyk(i)
  Next i
```

```
   ' Gleichungssystem G * a = c loesen mit Matrixinversion'
  g1 = Application.MInverse(G)            'Koeffizientenmatrix G invertieren'
  For i = 1 To Polynomgrad + 1            'Matrixmultiplikation a = G1 * c'
    a(i - 1) = 0
    For j = 1 To Polynomgrad + 1
      a(i - 1) = a(i - 1) + g1(i, j) * c(j)
    Next j
  Next i

  PolynomReg = a   'Koeffizientenvektor a0,..,an retournieren'

End Function


' Auswerten einer Polynomfunktion n-ten Grades an der Stelle x
' Die Polynomfunktion wird mit Koeffizentenarray a0,..,an definiert und an der Stelle
' x ausgewertet und als Funktionswert retourniert.
' Die Berechnung erfolgt mit dem Horner-Schema.
'
' Autor: Gerhard Krucker
' Datum: 19.8.1995, 22. 9. 1996
' Sprache: VBA for EXCEL7
'
Function PolynomEval(a, x As Double) As Double
Dim px                          ' Polynomfunktionswert '
Dim Polynomgrad As Integer   ' Grad des Polynomes, das die Koeffizienten in a darstellen '
Dim i

  Polynomgrad = a.Count - 1  ' Grad des Polynoms bestimmen = Anzahl Koeffizienten-1 '

  px = a(Polynomgrad + 1)
  For i = Polynomgrad To 1 Step -1
      px = px * x + a(i)
  Next i

  PolynomEval = px       ' Funktionswert retournieren '

End Function
```

Damit haben wir jetzt zwei leistungsfähige Funktionen zur Verfügung, um die Aufgabe einfach zu lösen.
Wir fügen die Visual Basic Funktionen in unsere Arbeitsmappe ein und können nun unter "Einfügen/Funktion /Benutzerdefiniert" auf die Funktionen über den Funktionsassistenten zugreifen.

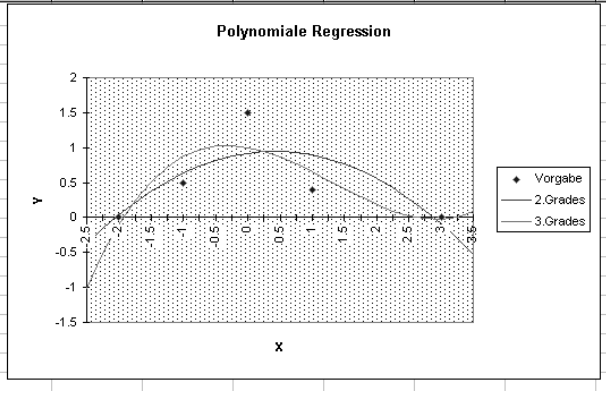
Da die Funktion PolynomReg eine sog. *Arrayfunktion* ist, geht die Berechnung etwas anders vonstatten als bei 'normalen' EXCEL-Funktionen:

In die Zelle B9 wird direkt oder mit Hilfe des Funktionsassistenten geschrieben:

B9: =PolynomReg($B$4:$F$4;$B$5:$F$5;2)

Die Eingabe wird normal abgeschlossen und wir erhalten in der Zelle B9 den Koeffizienten $a_0$. Nun wird das Feld für die Koeffizienten $a_0..,a_n$ durch Selektieren der Zellen B9: D9 aufgezogen. Dann wird die Eingabezeile mit der Formel oberhalb des Arbeitsblattes angeklickt und die Zeile mit <CTL><Shift><RET> abgeschlossen. Die restlichen Koeffizienten werden jetzt in die markierten Felder übertragen.
Für das Polynom 3. Grades ist das Vorgehen analog.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Ausgleichspolynom mit benutzerdefinierten EXCEL-Funktionen** | | | | | |
| 2 | | | | | | |
| 3 | **Vorgaben:** | | | | | |
| 4 | x | -2 | -1 | 0 | 1 | 3 |
| 5 | y | 0 | 0.5 | 1.5 | 0.4 | 0 |
| 6 | | | | | | |
| 7 | **Polynomkoeffizienten:** | | | | | |
| 8 | | a0 | a1 | a2 | a3 | |
| 9 | 2. Grades | 0.9181 | 0.1279 | -0.1546 | | |
| 10 | 3.Grades | 1.0013 | -0.1735 | -0.2397 | 0.0612 | |




Die Berechnung des Graphen kann jetzt ausserordentlich einfach mit der Funktion PolynomEval ausgeführt werden.

Besonders bequem wird die Arbeit, wenn die Funktionen als sog. XLA-Modul (EXCEL-Add-IN) erstellt werden. Dazu ist das Basic-Modul zu kompilieren (Add-In erstellen) und nachher mit dem Add-In Manager einzubinden. Dabei ist zu beachten, dass die Dateiinformation der Arbeitsmappe ausgefüllt wird, da diese die Namen enthält, die nachher im Add-In-Manager erscheinen.

## 3.2.6 Lösung überbestimmter Gleichungssysteme

Haben wir ein System mit mehr Gleichungen als Unbekannten, so sprechen wir von einem überbestimmten Gleichungssystem.

Für solche Systeme gibt es in der Regel keine Lösung im herkömmlichen Sinne. Vor allem dann, wenn der Rang der Koeffizientenmatrix maximal ist, d.h. die Gleichungen widersprüchlich sind und das System nicht reduziert werden kann. Hier wird ebenfalls eine Lösung nach der Methode der kleinsten Fehlerquadratsumme bestimmt.
Die erhaltene Lösung lässt sich umgangssprachlich interpretieren: 'Die Lösung passt etwa überall gleich schlecht...'

### 3.2.6.1 Anwendung auf ein allgemeines lineares Gleichungssystem

Die Lösung eines überbestimmten linearen Gleichungssystems soll nach der Methode der kleinsten Fehlerquadrate bestimmt werden. Wir setzen voraus, dass der Rang der Koeffizientenmatrix maximal ist, d.h. keine Zeile kann durch Linearkombination von anderen Zeilen dargestellt werden. Dies ist dann der Fall, wenn die Zeilen widersprüchlich sind. Wir betrachten stellvertretend für die Methode ein Beispiel.

**Beispiel:** Man bestimme *x, y* als Lösung des überbestimmten linearen Gleichungssystems

$$\begin{aligned} 2x + 3y &= 1 \\ x - 4y &= -9 \\ 2x - y &= -1 \end{aligned}$$

Die Lösung gestaltet sich in diesem Fall relativ einfach: Man bestimmt zuerst die Grundfunktion *f(x,y)=z*, die jeder einzelnen Zeile zugrunde liegt:

$$f(x,y) = z = (ax + by) \tag{3.26}$$

Von dieser Funktion wird nun das Gleichungssystem für die Parameter *a* und *b* durch Nullsetzen der partiellen Ableitungen der Fehlerquadratsumme q*(a,b)* bestimmt:

$$q(x,y) = \sum (ax + by - z)^2 \tag{3.27}$$

$$\frac{\partial}{\partial x}\sum (ax + by - z)^2 = 2a\sum (ax + by - z)$$

$$\frac{\partial}{\partial y}\sum (ax + by - z)^2 = 2b\sum (ax + by - z)$$

$$\begin{pmatrix} \sum a^2 & \sum ab \\ \sum ab & \sum b^2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sum az \\ \sum bz \end{pmatrix} \tag{3.28}$$

Das Gleichungssystem für *a, b* stellt ein 'normales' lineares Gleichungssystem mit zwei Unbekannten dar, welches mit den bekannten Methoden gelöst werden kann. Wir bestimmen zuerst die Summen,

dann lösen wir nach *a, b* auf mit Hilfe einer Matrixinversion und erhalten die Resultate:

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **Lösen eines überbestimmten linearen Gleichungssystems** | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | Gleichungssystem: | $2x+3y=1$ | | | Grundfunktion: | $f(y,x)=(ax+by)=z$ | | | |
| 4 | | $x-4y=-9$ | | | Ansatz: | $q(a,b)=\sum(ax+by-z)^2$ | | | |
| 5 | | $2x-y=-1$ | | | | $\frac{\partial}{\partial x}\sum(ax+by-z)^2=2a\sum(ax+by-z)$ | | | |
| 6 | | | | | | $\frac{\partial}{\partial y}\sum(ax+by-z)^2=2b\sum(ax+by-z)$ | | | |
| 7 | | | | | | $\begin{pmatrix}\sum a^2 & \sum ab\\ \sum ab & \sum b^2\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}=\begin{pmatrix}\sum az\\ \sum bz\end{pmatrix}$ | | | |
| 8 | | | | | | | | | |
| 9 | | | | | | | | | |
| 10 | | | | | Σ | | | | |
| 11 | a | 2 | 1 | 2 | 5 | | | | |
| 12 | $a^2$ | 4 | 1 | 4 | 9 | | | | |
| 13 | b | 3 | -4 | -1 | -2 | | | | |
| 14 | $b^2$ | 9 | 16 | 1 | 26 | | | | |
| 15 | ab | 6 | -4 | -2 | 0 | | | | |
| 16 | z | 1 | -9 | -1 | -9 | | | | |
| 17 | az | 2 | -9 | -2 | -9 | | | | |
| 18 | bz | 3 | 36 | 1 | 40 | | | | |
| 19 | | | | | | | | | |
| 20 | | 9 | 0 | | | -9 | | | |
| 21 | A = | 0 | 26 | | b = | 40 | | | |
| 22 | | | | | | | | | |
| 23 | | 0.11111 | 0 | | | -1 | | | |
| 24 | $A^{-1}$ = | 0 | 0.03846 | | $A^{-1}$ * b = | 1.53846 | | | |
| 25 | | | | | | | | | |
| 26 | | | | | | | | | |

Somit sind *x*=-1 und *y*=1.53846 die Lösungen des Gleichungssystems.

Auf eine programmierte Lösung des Verfahrens wird hier verzichtet. Das Vorgehen ist prinzipiell gleich wie bei den anderen gezeigten Methoden. Interessant und relativ einfach wird die Lösung, wenn der Algorithmus von Householder zur Lösung überbestimmter Gleichungssysteme eingesetzt wird (vgl. [3, S.101 und S.234]).

### 3.2.6.2 Anwendung auf eine Linearkombination beliebiger reeller Funktionen.

Für die Ausgleichsfunktion in Form einer Linearkombination reeller Funktionen

$$f(x)=\sum_{i=1}^{n} b_i g_i(x) \tag{3.29}$$

wird bei einer echten Ausgleichsaufgabe $m > n$ (mehr Punkte als Parameter) die Ausgleichskurve nicht exakt durch die Punkte $(x_i,y_i)$ verlaufen und i.a. nicht einmal einen Punkt genau treffen.

Die optimale Ausgleichsfunktion soll jedoch mit einem möglichst kleinen Fehler alle Punkte $(x_i, y_i)$ beschreiben. Diese Forderung wird erfüllt, wenn die Summe der Fehlerquadrate minimal ist:

$$q(b_0,\ldots,b_n)=\sum_k \left(f(x_k)-y_k\right)^2=\sum_k\left[\sum_{i=1}^{n} b_i g_i(x_k)-y_k\right]^2 \to \min \tag{3.30}$$

Die Parameter $b_i$ werden durch Nullsetzen der partiellen Ableitungen $\frac{\partial q}{\partial b_j}$ bestimmt:

Allgemein: $$\frac{\partial q}{\partial b_j}=2\sum_k g_j(x)\left[\sum_{i=1}^{n} b_i g_i(x_k)-y_k\right]=0 \quad 0\le j\le n \tag{3.31}$$

Dieses System kann nun in ein allgemeines lineares Gleichungssystem umgeschrieben werden. Dieses stellt die Normalgleichungen dar, das mit den bekannten Verfahren aufgelöst werden kann. Für die Normalgleichung *j* wird dies:

$$\sum_{i=0}^{n} b_i \left[ \sum_{k=1}^{m} g_j(x_k) g_i(x_k) \right] = \sum_{k=1}^{m} y_k g_j(x_k) \qquad \begin{array}{l} m\text{: Anzahl Datenpunkte} \\ n\text{: Anzahl Gleichungen} \\ j\text{: } 0..n \end{array} \tag{3.32}$$

**Beispiel:**
Für die Ausgleichsfunktion:

$$y = b_1 \ln x + b_2 \cos x + b_3 e^x$$

erhalten wir die Fehlerquadratsumme:

$$q = \sum_k \left( b_1 \ln x_k + b_2 \cos x_k + b_3 e^{x_k} - y_k \right)^2$$

Wir setzen die partiellen Ableitungen $\frac{\partial q}{\partial b_1} = 0, \frac{\partial q}{\partial b_2} = 0, \frac{\partial q}{\partial b_3} = 0$ und erhalten nach dem Umstellen die drei Normalgleichungen:

$$\begin{array}{llll}
b_1 \sum_k (\ln x_k)^2 & + b_2 \sum_k (\ln x_k) \cos x_k & + b_3 \sum_k (\ln x_k) e^{x_k} & = \sum_k y_k \ln x_k \\
b_1 \sum_k (\ln x_k) \cos x_k & + b_2 \sum_k (\cos x_k)^2 & + b_3 \sum_k (\cos x_k) e^{x_k} & = \sum_k y_k \cos x_k \\
b_1 \sum_k (\ln x_k) e^{x_k} & + b_2 \sum_k (\cos x_k) e^{x_k} & + b_3 \sum_k \left(e^{x_k}\right)^2 & = \sum_k y_k e^{x_k}
\end{array}$$

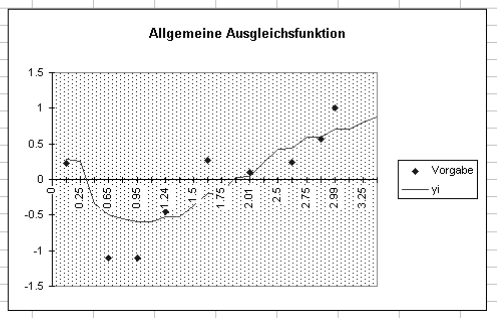
Auch hier erhalten wir immer eine symmetrische Koeffizientenmatrix.

Für die Datenpunkte

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *x* | 0.24 | 0.65 | 0.95 | 1.24 | 1.73 | 2.01 | 2.23 | 2.52 | 2.77 | 2.99 |
| *y* | 0.23 | -0.26 | -1.10 | -0.45 | 0.27 | 0.10 | -0.29 | 0.24 | 0.56 | 1.00 |

erhalten wir die Rechnung:

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Allgemeines Ausgleichsverfahren mit einr Linearkombination reeller Funktionen:** | | | | | | | | | | | |
| 2 | **Funktion:** | y=b1 ln x + b2 cos x + b3 exp x | | | | | | | | | | |
| 3 | | | | | | | | | | | | |
| 4 | **Vorgabewerte:** | | | | | | | | | | | |
| 5 | x | 0.24 | 0.65 | 0.95 | 1.24 | 1.73 | 2.01 | 2.23 | 2.52 | 2.77 | 2.99 | |
| 6 | y | 0.23 | -0.26 | -1.1 | -0.45 | 0.27 | 0.1 | -0.29 | 0.24 | 0.56 | 1 | |
| 7 | | | | | | | | | | | | |
| 8 | **Koeffizienten:** | | | | | | | | | | | |
| 9 | k | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Σ |
| 10 | (ln(xk))^2 | 2.037 | 0.186 | 0.003 | 0.046 | 0.300 | 0.487 | 0.643 | 0.854 | 1.038 | 1.200 | 6.794 |
| 11 | ln(xk) cos(xk) | -1.386 | -0.343 | -0.030 | 0.070 | -0.087 | -0.297 | -0.491 | -0.751 | -0.949 | -1.083 | -5.347 |
| 12 | ln(xk) exp(xk) | -1.814 | -0.825 | -0.133 | 0.743 | 3.092 | 5.210 | 7.459 | 11.487 | 16.259 | 21.780 | 63.259 |
| 13 | (cos (xk))^2 | 0.943 | 0.634 | 0.338 | 0.105 | 0.025 | 0.181 | 0.375 | 0.661 | 0.868 | 0.977 | 5.108 |
| 14 | cos(xk) exp(xk) | 1.525 | 1.504 | 1.122 | -0.894 | -3.174 | -5.696 | -10.104 | -14.869 | -19.658 | 1.000 | -49.243 |
| 15 | (exp(xk))^2 | 1.616 | 3.669 | 6.686 | 11.941 | 31.817 | 55.701 | 86.488 | 154.470 | 254.678 | 395.440 | 1002.507 |
| 16 | | | | | | | | | | | | |
| 17 | yk ln(xk) | -0.328 | 0.112 | 0.056 | -0.097 | 0.148 | 0.070 | -0.233 | 0.222 | 0.571 | 1.095 | 1.616 |
| 18 | yk cos(xk) | 0.223 | -0.207 | -0.640 | -0.146 | -0.043 | -0.043 | 0.178 | -0.195 | -0.522 | -0.989 | -2.383 |
| 19 | yk exp(xk) | 0.292 | -0.498 | -2.844 | -1.555 | 1.523 | 0.746 | -2.697 | 2.983 | 8.937 | 19.886 | 26.773 |
| 20 | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | |
| 25 | **Gleichungssystem G * b = c:** | | | | | | | | | | | |
| 26 | | | | | | | | | | | | |
| 27 | | 6.794 | -5.347 | 63.259 | | | 1.0678 | 0.8894 | -0.0237 | | | |
| 28 | G | -5.347 | 5.108 | -49.243 | | INV(G) | 0.8894 | 1.1126 | -0.0015 | | | |
| 29 | | 63.259 | -49.243 | 1002.507 | | | -0.0237 | -0.0015 | 0.0024 | | | |
| 30 | | | | | | | | | | | | |
| 31 | | 1.616 | | | | | -1.0276 | | | | | |
| 32 | c | -2.383 | | | | b | -1.2529 | | | | | |
| 33 | | 26.773 | | | | | 0.0300 | | | | | |
| 34 | | | | | | | | | | | | |
| 35 | | | | | | | | | | | | |
| 36 | | | | | | | | | | | | |
| 37 | | | | | | | | | | | | |
| 38 | | | | | | | | | | | | |




die gesuchte Ausgleichsfunktion lautet also:

$$y(x) = -1.0276\ \ln x\ -1.2529\ \cos\ x\ +\ 0.0300e^{x}$$

### 3.2.7 Beurteilung der Regressionsgüte bei nichtlinearen Regressionen

Bei nichtlinearen Regressionen wird die Güte anhand der Fehlerquadratsumme beurteilt, weil kein Korrelationskoeffizient wie bei der linearen Regression definiert ist. Werden mehrere verschiedene Regressionsfunktionen miteinander verglichen, so ist diejenige Funktion die beste Regressionsfunktion bezüglich der gegebenen Datenpunkte, welche die kleinste Fehlerquadratsumme hat.

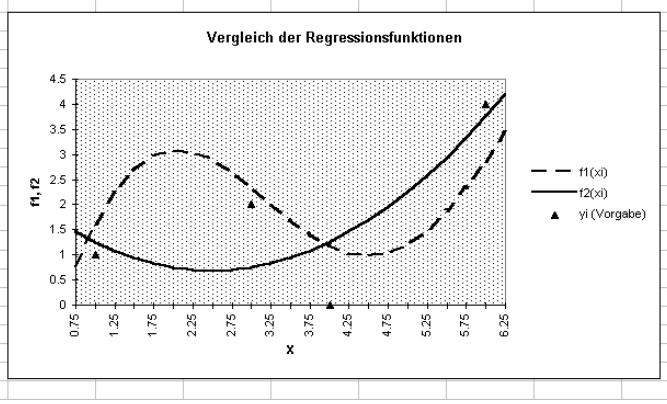
**Beispiel:**
Welche Regressionsfunktionen $f_1(x)$, $f_2(x)$ gleicht die folgenden Datenpunkte $(x_i, y_i)$ besser aus?

$$f_1(x) = 1.901\ln x + 1.928\sin x$$
$$f_2(x) = 0.25x^2 - 1.25x + 2.25$$

| $x_i$ | 1 | 3 | 4 | 6 |
|---|---|---|---|---|
| $y_i$ | 1 | 2 | 0 | 4 |

Wir bestimmen die Fehlerquadratsumme jeder Funktion bezüglich der Datenpunkte:

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Beurteilung der Regression durch Vergleich der Fehlerquadratsummen:** | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | |
| 3 | $f_1(x) = 1.901\ln x + 1.928\sin x$ | | | | | | | | | | | | | | |
| 4 | $f_2(x) = 0.25x^2 - 1.25x + 2.25$ | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | |
| 7 | $x_i$ | $f_1(x_i)$ | $f_2(x_i)$ | $y_i$ | $(f_1(x_i)-y_i)^2$ | $(f_2(x_i)-y_i)^2$ | | | | | | | | | |
| 8 | 1 | 1.6224 | 1.25 | 1 | 0.387 | 0.063 | | | | | | | | | |
| 9 | 3 | 2.3605 | 0.75 | 2 | 0.130 | 1.563 | | | | | | | | | |
| 10 | 4 | 1.1762 | 1.25 | 0 | 1.384 | 1.563 | | | | | | | | | |
| 11 | 6 | 2.8674 | 3.75 | 4 | 1.283 | 0.063 | | | | | | | | | |
| 12 | | | | Σ | 3.184 | 3.250 | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | |




Die Funktion $f_1(x)$ gleicht die Datenpunkte besser aus, weil sie die kleinere Fehlerquadratsumme hat.

### 3.2.8 Nichtlineare Regression mit Substitutionsverfahren

Ein ganze Reihe nichtlinearer Regressionsaufgaben kann durch geeignete Transformationen der Werte in eine äquivalente lineare Regressionsaufgabe übergeführt werden, welche dann mit den Standardformeln (3-12, 3-13) einfach zu lösen sind.

Wir betrachten nachfolgend Verfahren für zwei und drei Regressionskoeffizienten.

### 3.2.8.1 Regressionen mit zwei Regressionskoeffizienten

Viele nichtlineare Regressionsfunktionen können durch geeignete Transformationen in eine äquivalente lineare Regressionsaufgabe umgeformt werden ("linearisiert"). Diese ist dann sehr einfach zu lösen.

Bsp: Gesucht ist eine Ausgleichsfunktion der Art:

$$y(x) = be^{ax}$$

Diese Exponentialfunktion kann durch Logarithmieren linearisiert werden:

$$\ln(y(x)) = \ln(be^{ax}) = \ln(b) + \ln(e^{ax}) = \ln(b) + ax = B + AX$$

Dieses Prinzip kann genauso für weitere Funktionen angewandt werden:

| Typ | Funktion | Substitutionen $Y=$ | $X=$ | $B=$ | $A=$ | Bedingungen |
|---|---|---|---|---|---|---|
| 1 | $y(x) = b + ax^k$ | $y$ | $x^k$ | $b$ | $a$ | $k \neq 0$ |
| 2 | $y(x) = \frac{1}{b + ax^k}$ | $\frac{1}{y}$ | $x^k$ | $b$ | $a$ | $k \neq 0$<br>$y > 0$ |
| 3 | $y(x) = b + a\ln(x)$ | $y$ | $\ln(x)$ | $b$ | $a$ | $x > 0$ |
| 4 | $y(x) = \frac{1}{b + a\ln(x)}$ | $\frac{1}{y}$ | $\ln(x)$ | $b$ | $a$ | $x > 0$<br>$y > 0$ |
| 5 | $y(x) = bx^a + k$ | $\ln(y-k)$ | $\ln(x)$ | $\ln(b)$ | $a$ | $y-k > 0$<br>$x > 0$ |
| 6 | $y(x) = b\,a^{kx}$ | $\ln(y)$ | $k\,x$ | $\ln(b)$ | $\ln(a)$ | $k \neq 0$<br>$y > 0$ |
| 7 | $y(x) = be^{ax^k}$ | $\ln(y)$ | $x^k$ | $\ln(b)$ | $a$ | $k \neq 0$<br>$y > 0$ |

Der Begriff 'Typ' bezieht sich auf den Menüpunkt im Programm `NLREGWIN.EXE`. Es ist auf der zum Skript gehörenden Beispieldiskette verfügbar.

### 3.2.8.2 Regressionen mit drei Regressionskoeffizienten

Analog können hier auch über geeignete Substitutionen gewisse nichtlineare Regressionsfunktionen vereinfacht bestimmt werden, indem sie auf eine quadratische Regressionsaufgabe zurückgeführt werden.

$$y(x) = A_0 + A_1 x + A_2 x^2$$

Die Bestimmung der Polynomkoeffizienten geschieht über den Ansatz der Minimierung der Summe der Fehlerquadrate, analog Kap. 3. 2.5. Wir erhalten ein Gleichungssystem mit drei Unbekannten für die Regressionskoeffizienten $A_0..A_2$:

$$\begin{pmatrix} n & \sum X & \sum X^2 \\ \sum X & \sum X^2 & \sum X^3 \\ \sum X^2 & \sum X^3 & \sum X^4 \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} \sum Y \\ \sum XY \\ \sum X^2 Y \end{pmatrix}$$

Die Regressionsfunktionen, welche sich über eine quadratische Substitution lösen lassen sind nachfolgend tabelliert:

| Typ | Funktion | Substitutionen $Y=$ | $X=$ | $A_0=$ | $A_1=$ | $A_2=$ | Bedingungen |
|---|---|---|---|---|---|---|---|
| 8 | $y(x) = a_0 + a_1 x^k + a_2 x^{2k}$ | $y$ | $x^k$ | $a_0$ | $a_1$ | $a_2$ | $k \geq 1$ |
| 9 | $y(x) = \frac{1}{a_0 + a_1 x^k + a_2 x^{2k}}$ | $\frac{1}{y}$ | $x^k$ | $a_0$ | $a_1$ | $a_2$ | $y \neq 0$<br>$k \geq 1$ |
| 10 | $y(x) = a_0\, a_1^{x^k}\, a_2^{x^{2k}}$ | $\ln(y)$ | $x^k$ | $\ln(a_0)$ | $\ln(a_1)$ | $\ln(a_2)$ | $y > 0$<br>$k \geq 1$ |
| 11 | $y(x) = a_0 e^{a_1 (x - a_2)^2}$ | $\ln(y)$ | $x$ | $\ln(a_0) + a_1 a_2^2$ | $-2a_1 a_2$ | $a_1$ | $y > 0$ |

Ebenso lassen sich Substitutionen für vier Regressionskoeffizienten bestimmen, die dann in eine allgemeine Form mit vier Parameter gebracht wird.

### 3.2.8.3 Beispiele für Regressionen mit zwei Regressionskoeffizienten

Für folgende Messreihe soll eine Ausgleichsfunktion nach allen Verfahren aus Tabelle 1 bestimmt werden. *k* ist dem Wert 1.5 zu wählen:
Wir verwenden dazu das Programm NichtlineareRegression.EXE. Es bestimmt in der Konsolenversion die Regressionskoeffizienten für Regressionen mit zwei Regressionskoeffizienten. Mit der Win32- können auch Regressionen mit drei Regressionskoeffizienten durchgeführt werden. Aus Platzgründen werden hier die Listings nicht aufgeführt.

| $x$ | 1 | 2 | 4 | 8 | 12 | 14 | 18 | 22 |
|---|---|---|---|---|---|---|---|---|
| $y$ | 1.7 | 1.8 | 1.9 | 2.5 | 3.1 | 3.5 | 4.4 | 5.2 |

Resultate:

1: $1.678027 + 0.034666x + x^{1.5}$ $r_{xy} = 0.999$

2: $\frac{1}{0.539379 - 0.003952x^{1.5}}$ $r_{xy} = -0.954$

3: $1.065247 + 1.020555 \cdot \ln(x)$ $r_{xy} = 0.879$

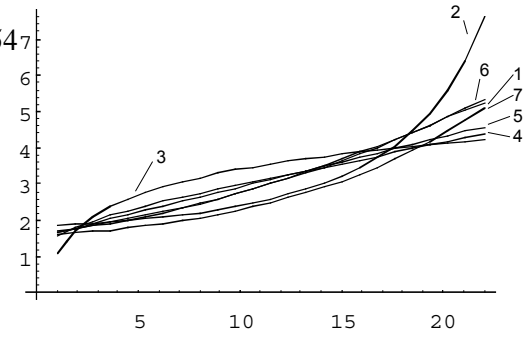
4: $\frac{1}{0.642606 - 0.133833 \cdot \ln(x)}$ $r_{xy} = -0.965$

5: $0.153601x^{0.969169} + 1.5$ $r_{xy} = 0.982$

6: $1.594536 \cdot 1.03776^{1.5x}$ $r_{xy} = 0.998$

7: $1.807221e^{0.011224x^{1.5}}$ $r_{xy} = 0.986$

Grafische Darstellung der einzelnen Regressionsfunktionen




Achtung:
$r_{xy}$ macht keine Aussage zur Korrelation der nichtlinearen Regressionsfunktion!
Besser ist hier die Aussage bezüglich der Fehlerquadratsumme.

## 3.2.9 Bestimmung von Kurvenfunktionen in logarith. Massstab mit Regression

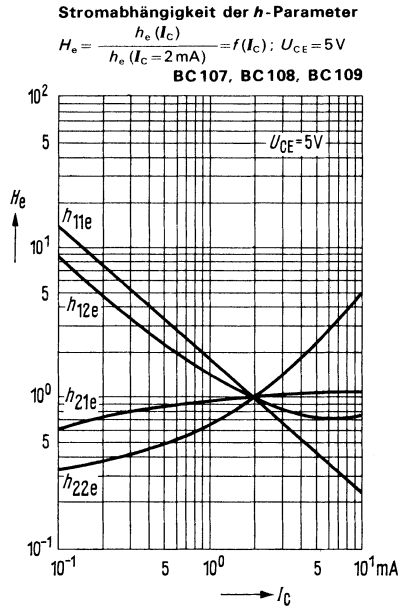
Die nachfolgenden Ausführungen beantworten einige Fragen, die sich beim Entwickeln einer Regressionsfunktion stellen können, wenn die Kurve in einem logarithmischen Bereich "gleichmässig gut passen" sollte. Das Prinzip das hier verfolgt wird ist, dass die Funktion nicht mehr bezüglich der kleinsten Fehlerquadrate optimiert wird, sondern dass eine relative Fehlerquadratsumme minimiert wird.

Wir betrachten dazu ein Fallbeispiel, wo für eine gegeben grafische Darstellung in logarithmischen Massstab ein gut passende analytische Funktion bestimmt werden soll. Dabei stellt sich die Frage:

Regressionsfunktionen bewerten nach der Methode der minimalen Fehlerquadratsumme. D.h. eine grosse Abweichung erzeugt demnach einen grossen Fehlerwert, der entsprechend berücksichtigt wird.

Dementsprechend stellen sich einige Fragen:
Wie sieht es aber aus, wenn die Regressionsfunktion sowohl kleine Werte, wie auch grosse möglichst gut abdecken muss? ("Kurve passt in einer logarithmischen Darstellung gut").
Erzeugt eine Abweichung der Kurve um z.B. 1mm beim Wert bei 10mA nicht einen wesentlich grösseren (quadratischen) Fehler als eine Abweichung um 1mm bei 100uA? Wäre es also nicht ev. besser mit einem relativen Fehler eine Regressionsfunktion zu bestimmen?

**Beispiel:**
Regression der h-Parameterumrechnung mittels Transformation und einer Regressionsfunktion vom Typ $y(x) = b + ax^k$. (Bild mit freundlicher Genehmigung der Firma Siemens AG.)




Mit Hilfe einer Transformation gemäss Kap. 3. 2. 8. 1 finden wir die Ausgleichsfunktion bezüglich der 5 gewählten Referenzpunkte.

$f(x) = 0.31966 + 0.29408x^{1.2}$ ($x$: Strom in mA [0.1,10])

Der Exponent $k$=1.2 wird durch "Probieren" ermittelt. Die Referenzpunkte und Abweichungen mit der obigen Ausgleichsfunktion werden:

| *I [mA]* | *H22* | *ε *10–3* | *δ [%]* |
|---|---|---|---|
| 0.1 | 0.33 | 8.21 | 2.49 |
| 0.5 | 0.48 | -32.34 | -6.74 |
| 2 | 1 | -4.73 | -0.47 |
| 5 | 2.3 | 48.39 | 2.1 |
| 10 | 5 | -19.52 | -0.39 |

Diese Funktion wird im Regelfall bereits eine genügende Ausgleichsgenauigkeit aufweisen. Trotzdem stellt sich die Frage, ob nicht gerade bei den kleinen Funktionswerten ein besserer Ausgleich erreicht werden kann.

### 3.2.9.1 Prinzip der minimalen relativen Fehlerquadratsumme

Wir definieren für den Ansatz eine relative quadratische Fehlersumme und bestimmen das Gleichungssystem für die Parameter *a, b*:

$$\Delta(a,b) = \sum_i \left( \frac{a\,x_i + b - y_i}{y_i} \right)^2 \tag{3.33}$$

$$\frac{\partial \Delta(a,b)}{\partial a} = 2\sum_i \frac{x_i(ax_i + b - y_i)}{{y_i}^2} = 2\left( a\sum \frac{X^2}{Y^2} + b\sum \frac{X}{Y^2} - \sum \frac{X}{Y} \right) == 0$$

$$\frac{\partial \Delta(a,b)}{\partial b} = 2\sum_i \frac{(ax_i + b - y_i)}{{y_i}^2} = 2\left( a\sum \frac{X}{Y^2} + b\sum \frac{1}{Y^2} - \sum \frac{1}{Y} \right) == 0$$

$$a\sum \frac{X^2}{Y^2} + b\sum \frac{X}{Y^2} = \sum \frac{X}{Y} \tag{3.34}$$

$$a\sum \frac{X}{Y^2} \quad + b\sum \frac{1}{Y^2} = \sum \frac{1}{Y} \tag{3.35}$$

Die Lösung für *a, b* werden unter Verwendung der Cramerschen Regel:

$$a = \frac{\sum \frac{X}{Y}\sum \frac{1}{Y^2} - \sum \frac{1}{Y}\sum \frac{X}{Y^2}}{\sum \frac{X^2}{Y^2}\sum \frac{1}{Y^2} - \left( \sum \frac{X}{Y^2} \right)^2} \tag{3.36}$$

$$b = \frac{\sum \frac{X^2}{Y^2}\sum \frac{1}{Y} - \sum \frac{X}{Y^2}\sum \frac{X}{Y}}{\sum \frac{X^2}{Y^2}\sum \frac{1}{Y^2} - \left( \sum \frac{X}{Y^2} \right)^2} \tag{3.37}$$

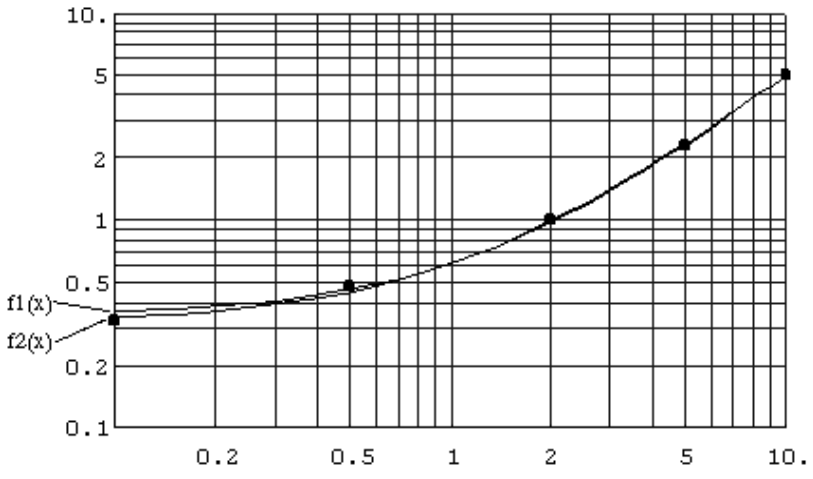
Dieser Ansatz wird nun mit der normalen linearen Regression nach der Methode der kleinsten Fehlerquadrate verglichen. Wir entwickeln die Lösungen mit EXCEL. Dies hat insofern den Vorteil, dass mit Hilfe des Solvers der optimale Exponent *k* bestimmt werden kann.

Wir erhalten die Lösungen:

$f_1(x) = 0.35033 + 026305x^{1.2474}$ (kleinste Fehlerquadratsumme)

$f_2(x) = 0.32578 + 0.28873x^{1.2092}$ (kleinste relative Fehlerquadratsumme)

Ein grafischer Vergleich zeigt, dass die nach der relativen Methode entwickelte Kurve besser bei den kleinen *x*-Werten annähert. Bezüglich absolutem Fehler ist sie aber etwas schlechter.




Regressionsfunktion nach der Methode der kleinsten relativen Fehlerquadratsumme $f_2(x)$, verglichen mit der Methode der kleinsten absoluten Fehlerquadratsumme $f_1(x)$.

Wir erkennen bei logarithmischen Darstellung eine bessere Näherung bezüglich der relativen Fehler in jedem Punkt.

## 3.3 Interpolation

Bereits im vorgehenden Kapitel wurde der Begriff *Interpolation* erklärt und mit der Ausgleichsaufgabe verglichen:

Eine Interpolationsfunktion *f(x)* ist eine Funktion, die exakt durch jeden der *m* Datenpunkte verläuft, wobei wir voraussetzen, dass alle Datenpunkte verschieden sind.

Die Interpolation kann also als Spezialfall der Ausgleichsrechnung betrachtet werden. Jedoch wird unter dem Bestimmen einer Interpolationsfunktion häufig das Bestimmen einer Polynomfunktion mit speziellen Verfahren verstanden. Diese Verfahren sind weniger rechenintensiv für eine grössere Menge von Datenpunkten als der Ansatz über die normale Ausgleichsrechnung.

Die meist verwendeten Methoden sind:

- Newton-Interpolation
- Lagrange-Interpolation
- Spline-Interpolation

In der Interpolationsrechnung werden die Begriffe unterschieden:

- Stützpunkte: $(x_i, y_i)$
- Stützstellen: $x_i$
- Stützwerte: $y_i$

### 3.3.1 Lagrange-Interpolation

Das Interpolationsverfahren wurde nach dem französischen Mathematiker Joseph Louis Lagrange (1736-1813) benannt. Mit dieser Methode lassen sich mit wesentlich weniger Rechenaufwand Interpolationspolynome und Interpolationswerte berechnen als mit den vorher gezeigten Verfahren.

Grundelemente der Methode sind die sog. Lagrange-Polynome:

$$L_i(x) := \prod_{\substack{j=0 \\ j\neq i}} \frac{x - x_j}{x_i - x_j} \qquad (i = 0.,\ldots,n) \tag{3.38}$$

Sie sind als Hilfsfunktionen zur Bestimmung des Interpolationspolynomes zu verstehen. Sie haben eine Reihe spezieller Eigenschaften, die nachfolgend betrachtet werden. Mit diesen speziellen Polynomfunktionen lässt sich das Interpolationspolynom wie folgt berechnen:

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x) \tag{3.39}$$

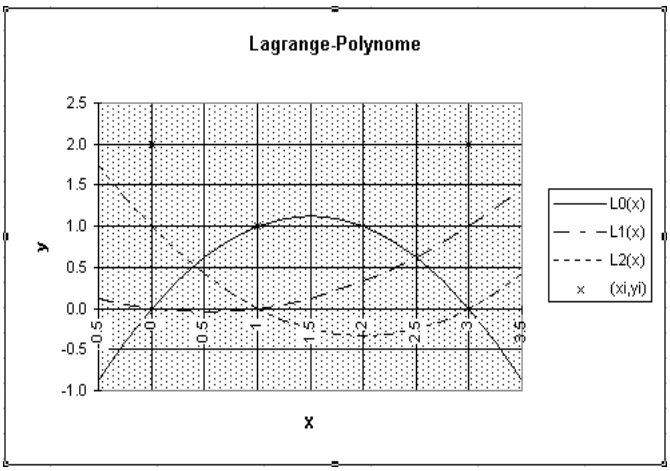
**Lagrange Interpolation**

Wir betrachten nachfolgend die Eigenschaften der Lagrange-Polynome anhand des Beispiels:

| $i$ | 0 | 1 | 2 |
|---|---|---|---|
| $x$ | 1 | 3 | 0 |
| $y$ | 1 | 2 | 2 |

Gemäss Definition werden die zugehörigen Lagrange-Polynome:

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(3-x)x}{2}$$
$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-1)x}{6}$$
$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{x^2-4x+3}{3}$$




Die nebenstehende grafische Darstellung zeigt die speziellen Eigenschaften der Lagrange-Polynome:

Es gilt grundsätzlich: $L_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ (3.40)

Also läuft $L_0$ bei der Stelle $x_0$ durch den Punkt ($x_0$,1) und bei den restlichen Stellen $x_1$, $x_2$ durch Null. Die anderen verlaufen gemäss Definition analog.
Damit $L_i(x)$ nun durch den Punkt $(x_i, y_i)$ läuft (Interpolationsbedingung!) muss $L_i(x)$ mit $y_i$ multipliziert werden. Anschliessend werden alle $L_i(x)$ addiert.
Die $L_i(x)$ beeinflussen sich gegenseitig nicht, da sie in allen anderen Punkten $L_i(x_{\neq i}) = 0$ sind und somit die Interpolationsbedingung erhalten bleibt.

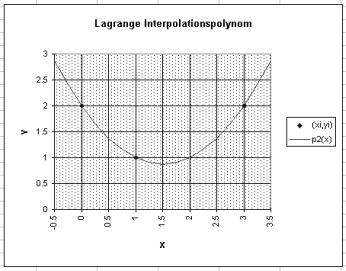
Zusammenfassend:
Sind *n* verschiedene Datenpunkte gegeben, so existiert immer ein Lagrange-Interpolationspolynom vom Grad *n-1*. **Dieses Interpolationspolynom ist eindeutig**, d.h. es gibt kein anderes Polynom diesen Grades, das dieselben Eigenschaften aufweist.

Das Interpolationspolynom wird nun durch Einsetzen der Lagrange-Polynome $L_i(x)$ bestimmt:

$$p_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) = 1\frac{(3-x)x}{2} + 2\frac{(x-1)x}{6} + 2\frac{(x-1)(x-3)}{3} = \frac{1}{2}x^2 - \frac{3}{2}x + 2$$

Die numerische Auswertung des Polynoms und die grafische Darstellung zeigen, dass die Interpolationsvorgaben für alle Stützpunkte $(x_i, y_i)$ erfüllt werden:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Lagrange Interpolation:** | | | |
| 2 | | | | |
| 3 | Interpolationsvorgaben: | | | |
| 4 | i | 0 | 1 | 2 |
| 5 | x | 1 | 3 | 0 |
| 6 | y | 1 | 2 | 2 |
| 7 | | | | |
| 8 | x0 | -0.5 | | |
| 9 | xn | 2.5 | | |
| 10 | h | 0.25 | | |
| 11 | | | | |
| 12 | i | xi | (xi,yi) | p2(x) |
| 13 | 0 | -0.5 | | 2.875 |
| 14 | 1 | -0.25 | | 2.40625 |
| 15 | 2 | 0 | 2 | 2 |
| 16 | 3 | 0.25 | | 1.65625 |
| 17 | 4 | 0.5 | | 1.375 |
| 18 | 5 | 0.75 | | 1.15625 |
| 19 | 6 | 1 | 1 | 1 |
| 20 | 7 | 1.25 | | 0.90625 |
| 21 | 8 | 1.5 | | 0.875 |
| 22 | 9 | 1.75 | | 0.90625 |
| 23 | 10 | 2 | | 1 |
| 24 | 11 | 2.25 | | 1.15625 |
| 25 | 12 | 2.5 | | 1.375 |
| 26 | 13 | 2.75 | | 1.65625 |
| 27 | 14 | 3 | 2 | 2 |
| 28 | 15 | 3.25 | | 2.40625 |
| 29 | 16 | 3.5 | | 2.875 |
| 30 | | | | |



### 3.3.2 Implementierung der Lagrange-Interpolation

Für die Handrechnung einzelner Funktionswerte (wozu die Interpolationsrechnung eigentlich gedacht ist) ist die Grundform der Lagrange-Formel kaum geeignet. Eine programmierte Lösung gestaltet sich aber einfach:

```
/* Lagrange Interpolation                                    File: LAGRANG1.C

   Bestimmen von Interpolationswerten mit Lagrange-Interpolation.
   Die Koeffizienten des Interpolationspolynomes werden jedoch nicht explizit berechnet.

   Grundformel:
              p(xk) = y0 L0(xk) + y1 L1(xk) + ... + yn Ln(xk)

   Autor: Gerhard Krucker
   Datum: 21.8.1995
   Sprache: MS Visual-C V1.52  (QuickWin App.)
*/

#include <stdio.h>

#define MAX 10           /* Maximale Anzahl Datenpunkte (kann angepasst werden) */
main()
{ double x[MAX]; /* Vektor mit x-Werten der Datenpunkte */
  double y[MAX];    /* Vektor mit y-Werten der Datenpunkte */
  double Li;
  double xk;        /* Aktueller x-Wert */
  double px;             /* Interpolationswert */
  int n;                 /* Anzahl Datenpunkte */
  int i,k;

  printf("Berechnen von Interpolationswerten mit Lagrange-Interpolation:\n");
  printf("Anzahl der Datenpunkte? "); scanf("%d",&n);

  /* Einlesen der Datenpunkte */
  for (k=0; k < n; k++)
    {   printf("Punkt %d: x = ",k+1); scanf("%lg",&x[k]);
        printf("          y= "); scanf("%lg",&y[k]);
        putchar('\n');
    }

 printf("Berechnen der Interpolationswerte (Beenden mit <CTL>C)\n"  );
 while (1)
   {  px = 0.0;
      printf("x = "); scanf("%lg",&xk);

      for (i = 0; i < n; i++)
        { Li = 1.0;
          for (k = 0; k < n; k++)
             if (k != i) Li *= (xk - x[k])/(x[i] - x[k]);
          px += y[i] * Li;
        }
       printf("px(%g) = %g\n",xk,px);
    }
  return 0;
  }
```

Für das obige Interpolationspolynom $p_2(x) = \frac{1}{2}x^2 - \frac{3}{2}x + 2$ erhalten wir die interpolierten Werte im Intervall [0,1]:

| $x$ | Interpolationswert | |
|---|---|---|
| 0.0 | 2 | (Vorgabe) |
| 0.2 | 1.72 | |
| 0.4 | 1.48 | |
| 0.6 | 1.28 | |
| 0.8 | 1.12 | |
| 1.0 | 1 | (Vorgabe) |

Die Polynomkoeffizienten von $L_i(x)$ werden üblicherweise über die Stützkoeffizienten $\lambda_i$ und ein Produkt von Linearfaktoren bestimmt.

Die $\lambda_i$ verkörpern eine Hilfsgrösse, die nur von den $x_i$ abhängig ist. Sie entsteht durch Extrahieren des Nennerterms aus dem Lagrange-Polynom $L_i$:

$$\lambda_i = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{1}{x_i - x_j} \tag{3.41}$$

Die Lagrange-Interpolationsformel mit den Stützkoeffizienten lautet dann:

$$p_n(x) = \sum_{i=0}^{n} y_i \; \lambda_i \prod_{j=0}^{n} (x - x_j) \tag{3.42}$$

Soll nun eine programmierte Lösung entwickelt werden, so werden zweckmässigerweise die Grössen $\lambda_i$ und $\prod_j (x - x_j)$ in separaten Funktionen bestimmt und nachher in einer Hauptfunktion verrechnet.

Dazu für EXCEL eine Lösung in Visual-Basic for Application:

```
Option Explicit         'Variablendefinitionen explizit erzwingen
'
' Berechnen des Lagrange-Stuetzkoeffizienten bezueglich i.
' Parameter: x Array mit den x-Werten
'            i Index, fuer den der Stuetzkoeffizient bestimmt werden soll.
'              Mathematisch ist der erste Index 0. Dieser entspricht dem
'              Arrayindex 1 in Visual Basic.
' Resultat:  lambda_i Stuetzkoeffizient
'
' Autor: Gerhard Krucker
' Datum: 21.8.1995, 22.9.1996
' Sprache: MS Visual BASIC for Applications EXCEL 7.0
'
Function Lambda(x, ByVal i As Integer) As Variant
Dim j As Integer
Dim Anz As Integer          ' Anzahl Datenwerte im Array x '
Dim lambda_i As Double      ' Lagrange Stuetzkoeffizient bezueglich i '

   Anz = x.Count

   If (i >= Anz) Or (i < 0) Then         ' Fehler: Index > Anzahl Datenpunkte! '
      Lambda = CVErr(xlErrValue)
    Else
     lambda_i = 1
     i = i + 1                           ' EXCEL Visual Basic Index beginnt bei 1 '
     For j = 1 To Anz
       If i <> j Then
         lambda_i = lambda_i * 1 / (x(i) - x(j))
       End If
     Next j

     Lambda = lambda_i      ' Funktionswert retournieren '
   End If
End Function
'
' Bestimmen der Lagrange Produktes als Polynom a0 + a1 * x + a2 x^2 + .. + an * x^n bezueglich i '
' als Arrayfunktion.
' Diese Funktion wird nur von der Funktion LagrangeInterPol aufgerufen und steht nicht der Allgemeinheit
' zur Verfuegung.
' Formal:
'      p = (x-x0)(x-x1)...(x-xj)...(x-xn)   fuer j <> i und j=0...n
'
' Referenz: Skript Numerik I, H.P. Blau Uni Bern
' Parameter: x : Liste mit den xj
'            i : Stelle i, fuer die das Produkt entwickelt werden soll
' Resultat:  Array mit den Polynomkoeffizienten a0,...,an als Funktionswert retourniert.
'
' Datum: 21.8.1995, 22.9.1996
' Sprache: MS Visual BASIC for Applications EXCEL 7.0
'
Function LagrangeProdukt(x, ByVal i As Integer) As Variant
Dim Anz As Integer     ' Anzahl xj's in der Liste '
Dim j As Integer, k As Integer
Dim a                  ' Dynamisches Array fuer die Polynomkoeffizienten a0,..,an '
Dim startIndex         ' Index wo der 2. Term der Multiplikation ist (Beginn der Kettenmultplikation)'
Dim Agrad              ' Aktueller Grad des erzeugten Polynoms im Durchlauf '

  ' Parameterkontrolle '
  Anz = x.Count     ' Anzahl xj's in der Liste bestimmen '

  If i > Anz - 1 Then ' Index i ungueltig?  (Index groesser als Anzahl x-Werte) '
      LagrangeProdukt = CVErr(xlErrValue)
      Exit Function
  End If
  If Anz < 2 Then     ' weniger als 2 Datenpunkte (Keine multiplikation moeglich) '
     LagrangeProdukt = CVErr(xlErrValue)
     Exit Function
  End If

  ReDim a(0 To Anz - 1)    'Array fuer die benoetigte Anzahl Polynomkoeffizieten dimensionieren '
```

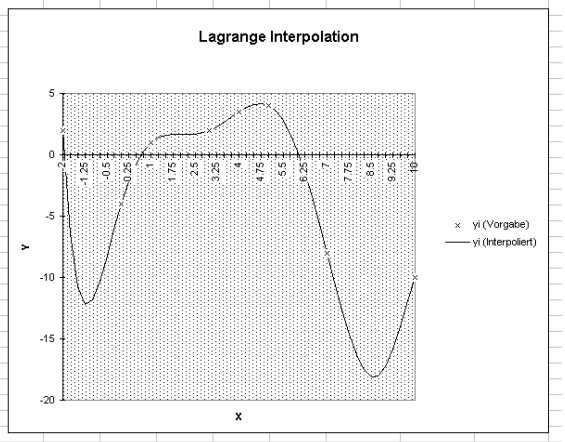
```
  If i <> 0 Then           ' Ersten Linearfaktor uebertragen '
     a(0) = -x(1)          ' i ist != 0, dann ist erster Faktor (x-x0)[EXCEL beginnt standardmaessig Indizierung bei 1!]'
     a(1) = 1
     startIndex = 2
    Else
     a(0) = -x(2)          ' i war == 0, dann ist erster Faktor (x-x1) '
     a(1) = 1
     startIndex = 3
  End If
  Agrad = 1


  k = startIndex
  While k <= Anz
    If (i + 1) <> k Then              ' (x-xj) anmultiplizieren, wenn i <> j '
      For j = Agrad To 0 Stepp -1      ' Koeffizienten um eine Stelle nach rechts schieben'
        a(j + 1) = a(j)
      Next j
      a(0) = 0
      For j = 0 To Agrad
        a(j) = a(j) - a(j + 1) * x(k)
      Next j
      Agrad = Agrad + 1
    End If
    k = k + 1
  Wend

  LagrangeProdukt = a

End Function
'
' Berechnen der Koeffizienten des Lagrange-Interpolationspolynoms fuer
' n Datenpunkte (xi, yi). Die Stuetzstellen duerfen hierbei auch ungleichmaessig
' verteilt sein.
'
' Parameter: x: Liste mit den Stuetzstellen xi '
'            y: Liste mit den Stuetzwerten yi '
' Resultat:  Array mit den Koeffizienten a0,..,an des Lagrange-Interpolationspolynomes '
'
' Referenz: F. Scheid, Numerische Analysis, Seite 57-61, McGraw-Hill (Schaum Serie)
'           H.P. Blau, Numerik I, Skript zur Vorlesung UNI Bern, Seite 33
'           H.R. Schwarz, Numerische Mathematik, Seite 95-104, Verlag Teubner
' Datum: 21.8.1995, 22.9.1996
' Sprache: MS Visual BASIC for Applications EXCEL 7.0
'
Function LagrangeInterPol(x, y) As Variant
Dim AnzX          ' Anzahl x-Werte in der Liste'
Dim AnzY          ' Anzahl y-Werte in der Liste'
Dim pn            ' Dynamisches Array fuer die Koeffizienten des Lagrange Polynoms n-ten Grades '
Dim Polynomgrad   ' Grad des Interpolationspolynomes '
Dim yili          ' Produkt yi * lambda(i) '
Dim pni           ' Lagrange-Polynom i / yili'
Dim i, j

  'Parameterkontrolle '
  AnzX = x.Count
  AnzY = y.Count

  If AnzX <> AnzY Then   'Fehler: Anzahl x-Werte ungleich Anzahl y-Werte '
     LagrangeInterPol = CVErr(xlErrValue)
     Exit Function
  End If
  If AnzX < 2 Then       'Fehler: Anzahl Datenpunkte müssen >= 2 sein! '
     LagrangeInterPol = CVErr(xlErrValue)
     Exit Function
   End If

  'Bestimmen der Koeffizienten des Interpolationspolynomes '
  Polynomgrad = AnzX - 1         ' Grad des erzeugten Interpolationspolynomes '
  ReDim pn(0 To Polynomgrad)     ' Array fuer benoetigte Anzahl Koeffizienten dimensionieren '
  ReDim pni(0 To Polynomgrad)    ' Array fuer Lagrange-Teilpolynom (x-x0)(x-x1)..(x-xj)..(x-xn)  i<>j '

  For i = 0 To AnzX - 1          ' Für jede Stuetzstelle ... '
    yili = y(i + 1) * Lambda(x, i)
    pni = LagrangeProdukt(x, i)
    For j = 0 To Polynomgrad     'Neue Koeffizienten dazuaddieren '
      pn(j) = pn(j) + pni(j) * yili
    Next j
  Next i

  LagrangeInterPol = pn

End Function
```

Mit diesen Funktionen gestaltet sich die Bestimmung der Polynomkoeffizienten äusserst einfach.

Wir bestimmen das Interpolationspolynom nach Lagrange für die Datenpunkte:

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $x$ | 1 | 3 | 0 | 4 | 7 | 5 | 10 | -2 |
| $y$ | 1 | 2 | 2 | 2 | 0 | 4 | -4 | 2 |

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **Beispiel: Lagrange Interpolation mit benutzerdefinierter Funktion** | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | **Datenpunkte:** | | | | | | | | |
| 4 | x | 1 | 3 | 0 | 4 | 7 | 5 | 10 | -2 |
| 5 | y | 1 | 2 | -4 | 3.5 | -8 | 4 | -10 | 2 |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | **Polynomkoeffizienten:** | | | | | | | | |
| 9 | a0 | a1 | a2 | a3 | a4 | a5 | a6 | a7 | |
| 10 | -4.0000 | 7.8011 | -2.2905 | -1.2081 | 0.8694 | -0.1884 | 0.0170 | -0.0006 | |
| 11 | | | | | | | | | |
| 12 | | | | | | | | | |
| 13 | **Graphische Darstellung:** | | | | | | | | |
| 14 | x0 | -2 | | xn | 11 | | | | |
| 15 | h | 0.25 | | | | | | | |
| 16 | | | | | | | | | |
| 17 | i | xi | yi (Vorgabe) | | yi (Interpoliert) | | | | |
| 18 | 0 | -2 | 2 | | 2.000 | | | | |
| 19 | 1 | -1.75 | | | -6.429 | | | | |
| 20 | 2 | -1.5 | | | -10.743 | | | | |
| 21 | 3 | -1.25 | | | -12.206 | | | | |
| 22 | 4 | -1 | | | -11.808 | | | | |
| 23 | 5 | -0.75 | | | -10.307 | | | | |
| 24 | 6 | -0.5 | | | -8.262 | | | | |
| 25 | 7 | -0.25 | | | -6.071 | | | | |
| 26 | 8 | 0 | -4 | | -4.000 | | | | |
| 27 | 9 | 0.25 | | | -2.209 | | | | |
| 28 | 10 | 0.5 | | | -0.774 | | | | |
| 29 | 11 | 0.75 | | | 0.286 | | | | |
| 30 | 12 | 1 | 1 | | 1.000 | | | | |
| 31 | 13 | 1.25 | | | 1.423 | | | | |
| 32 | 14 | 1.5 | | | 1.626 | | | | |
| 33 | 15 | 1.75 | | | 1.685 | | | | |
| 34 | 16 | 2 | | | 1.675 | | | | |
| 35 | 17 | 2.25 | | | 1.659 | | | | |
| 36 | 18 | 2.5 | | | 1.688 | | | | |
| 37 | 19 | 2.75 | | | 1.796 | | | | |
| 38 | 20 | 3 | 2 | | 2.000 | | | | |
| 39 | 21 | 3.25 | | | 2.298 | | | | |




Die interpolierende Kurve zeigt bei $x$-Werten um -1 und -7.5 recht grosse Schwinger. Hier kann nicht mehr von 'Interpolation' im Sinne des Bestimmens von Zwischenwerten gesprochen werden. Dies ist der Grund, warum in der Regel keine Interpolationspolynome vom Grad > 4 verwendet werden.

### 3.3.3 Newton-Interpolation

Die Newton-Interpolation ist eine weitere Polynominterpolation mit den folgenden Eigenschaften:

1. Das Interpolationspolynom $p_n(x)$ wird als Summe von Linearfaktoren dargestellt.

2. Die Koeffizienten von $p_n(x)$ werden durch schrittweise Erhöhung der Anzahl Stützpunkte rekursiv bestimmt.

3. Die Berechnung von $p_n(x)$ erfolgt rekursiv.

Die Newton-Interpolation liefert dieselben Ergebnisse wie die Lagrange-Interpolation, ist aber effizienter in der Durchführung.

Das Newton-Interpolationspolynom ist folgendermassen definiert:

$$\begin{aligned} p_n(x) &= \sum_{k=0}^{n} c_k \prod_{i=0}^{k-1} (x - x_i) \\ &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned} \tag{3.43}$$

Die Koeffizienten $c_i$ können der Reihe nach durch Einsetzen der $x_k$ und den Interpolationsbedingungen $p(x_k) = y_k$ bestimmt werden:

$$
\begin{aligned}
y_0 = p(x_0) = c_0 &\qquad \rightarrow c_0 = y_0 \qquad \text{(Rekursionsanfang)} \\
y_1 = p(x_1) = c_0 + c_1(x_1 - x_0) &\qquad \rightarrow c_1 = \frac{y_1 - c_0}{(x_1 - x_0)} \\
y_2 = p(x_2) = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) &\qquad \rightarrow c_2 = \frac{y_2 - c_0 - c_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)}
\end{aligned}
\tag{3.44}
$$

*etc.*

Es wird sofort klar, dass bei dieser Methode beliebig neue Punkte $(x_i, y_i)$ zugefügt werden können, ohne dass sich an den bereits berechneten Koeffizienten $c_i$ etwas ändert. Dieser Umstand ist der grosse Vorteil des Newton-Interpolationsverfahren.

Das Interpolationspolynom liegt nicht als Polynomfunktion Form $p_n(x) = \sum_{i=0}^{n} a_i x^i$ vor, sondern als Produkt von Linearfaktoren. Dies ist allerdings für die Berechnung von Interpolationswerten nicht von Bedeutung.

Bei Bedarf kann durch sukzessive Multiplikation das Produkt von Linearfaktoren in ein Standardpolynom umgerechnet werden, analog der Funktion '`LagrangeProdukt`' auf Seite 3-27.

### 3.3.4 Implementierung der Newton-Interpolation

Kernstück des Verfahrens ist die rekursive Berechnung der Koeffizienten $c_i$. Die Rekursionsformel lautet

$$
\begin{aligned}
c_0 &= y_0 \\
c_i &= \frac{y_i - \left[c_0 + c_1(x_i - x_0) + \cdots + c_{i-1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})\right]}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})}
\end{aligned}
\tag{3.45}
$$

In Visual-Basic wird die Interpolation zweckmässigerweise mit zwei Funktionen implementiert:

| | |
|---|---|
| NewtonInterpolWert(x;y;xi): | Bestimmen des Interpolationswertes an der Stelle $x_i$ für die Datenpunkte in der Liste $x$ und $y$ mit Hilfe der Newton-Interpolation. |
| NewtonC(x;y): | Bestimmen der Newton-Polynomkoeffizienten $c_i$ |

wobei die erste Funktion intern auf die zweite Funktion zurückgreift. Die Aufteilung in zwei Funktionen bringt den Vorteil, dass die Funktion zur Bestimmung der Polynomkoeffizienten $c_i$ auch unter EXCEL direkt verfügbar ist.

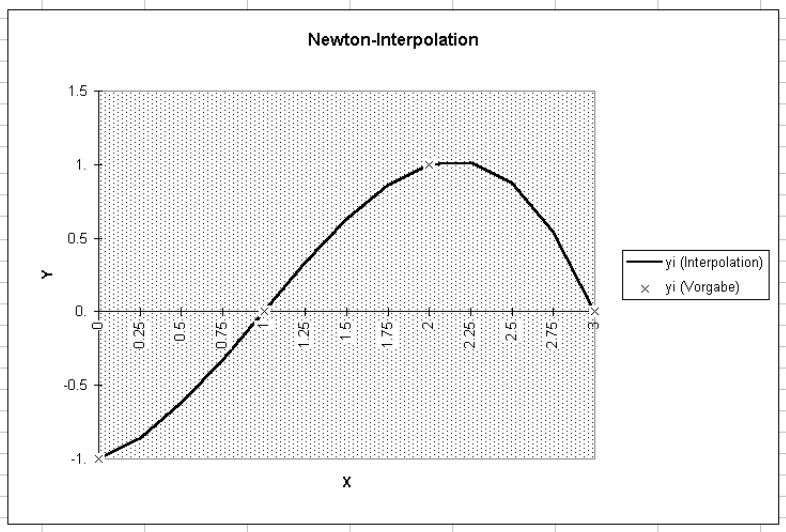
```
Option Explizit
'
' Berechnen des Newton-Interpolationswertes fuer die in x, y uebergebenen Datenpunkte
' an der Stelle xi
' Die Interpolation erfolgt durch das Bestimmen der Newton-Koeffizienten ci und
' anschliessendes Ausmultiplizieren des Polynoms
'
' Datum: 21.8.1995, 22.9.1996
' Sprache: MS Visual BASIC for Applications EXCEL 7.0
'
Function NewtonInterpolWert(x, y, xi) As Variant
Dim AnzX            ' Anzahl Stuetzstellen in x '
Dim AnzY            ' Anzahl Stuetzwerte in y '
Dim c               ' Dynamisches Array für die Koeffizienten ci '
Dim pn              ' Nennerprodukt (xi-x0)(xi-x1)..(xi-xi-1) '
Dim pz              ' Zaehlerterm '
Dim Polynomgrad     ' Grad des Newton-Interpolationspolynomes '
Dim k

  ' Parameter pruefen '
  AnzX = x.Count
  AnzY = y.Count
  If AnzX <> AnzY Then    ' Anzahl x-Wert <> Anzahl y-Werte: -> Fehler '
     NewtonInterpolWert = CVErr(xlErrValue)
     Exit Function
  End If
  If AnzX < 2 Then        ' Fuer eine Interpolation braucht es mindestens 2 Datenpunkte sonst: Fehler '
      NewtonInterpolWert = CVErr(xlErrValue)
      Exit Function
  End If
  Polynomgrad = AnzX - 1

  ' Interpolation durchfueren '
  ReDim c(0 To AnzX)         ' 0 .. AnzX Polynomkoeffizienten '
  c = NewtonC(x, y)          ' ci's bestimmen '
  pz = c(0): pn = 1          ' Initialisierung '
  For k = 1 To Polynomgrad
    pn = pn * (xi - x(k))    ' pn = pn(xi -x[k1]) (EXCEL beginnt mit ersten Element bei 1) '
    pz = pz + c(k) * pn
  Next k

  NewtonInterpolWert = pz

End Function

'
' Bestimmen der Newton-Koeffizienten ci fuer das NewtonInterpolationspolynom
' Das Resultat wir als Array mit allen Koeffizeinen c0,..,cn retounrniert.
' (EXCEL Array-Funktion)
'
' Referenz: Haemmerlin, Numerische Mathematik, Seite 111, Verlag Teubner
' Datum: 21.8.1995, 22.9.1996
' Sprache: MS Visual BASIC for Applications EXCEL 7.0
'
Function NewtonC(x, y) As Variant
Dim AnzX            ' Anzahl Stuetzsellen in x '
Dim AnzY            ' Anzahl Stuetzwerte in y '
Dim c               ' Dynamisches Array für die Koeffizienten ci '
Dim pn              ' Nennerprodukt (xi-x0)(xi-x1)..(xi-xi-1) '
Dim pz              ' Zaehlerterm '
Dim k, i

  ' Parameter pruefen '
  AnzX = x.Count
  AnzY = y.Count
  If AnzX <> AnzY Then    ' Anzahl x-Wert <> Anzahl y-Werte: -> Fehler '
     NewtonC = CVErr(xlErrValue)
     Exit Function
  End If
  If AnzX < 2 Then        ' Fuer ein Interpolation braucht es mindestens 2 Datenpunkte sonst: Fehler '
      NewtonC = CVErr(xlErrValue)
      Exit Function
  End If

  ' Koeffizienten bestimmen '
  ReDim c(0 To AnzX)       ' 0 .. AnzX Polynomkoeffizienten '
  c(0) = y(1)              ' c0 = y0 (in EXCEL ist y0 als y(1) indiziert '
  For k = 1 To AnzX - 1
     pn = 1: pz = c(0)     ' Initialisierung'
     For i = 1 To k - 1    ' Sukzessiv ck's aufbauen '
       pn = pn * (x(k + 1) - x(i - 1 + 1))
       pz = pz + c(i) * pn
     Next i
     pn = pn * (x(k + 1) - x(k - 1 + 1))
     c(k) = (y(k + 1) - pz) / pn
   Next k

   NewtonC = c

End Function
```

Ein Test für die Datenpunkte

| $x$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $y$ | -1 | 0 | 1 | 0 |

liefert die Werte:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Newton Interpolation:** | | | | |
| 2 | | | | | |
| 3 | xi | 0 | 1 | 2 | 3 |
| 4 | yi | -1 | 0 | 1 | 0 |
| 5 | | | | | |
| 6 | **Newton ci's:** | | | | |
| 7 | c0 | c1 | c2 | c3 | |
| 8 | -1 | 1 | 0 | -0.333 | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | **Interpolationswerte:** | | | | |
| 12 | $x_0$ | 0 | | | |
| 13 | h | 0.25 | | | |
| 14 | | | | | |
| 15 | i | xi | yi (Vorgabe) | yi (Interpolation) | |
| 16 | 0 | 0 | -1 | -1.0000 | |
| 17 | 1 | 0.25 | | -0.8594 | |
| 18 | 2 | 0.5 | | -0.6250 | |
| 19 | 3 | 0.75 | | -0.3281 | |
| 20 | 4 | 1 | 0 | 0.0000 | |
| 21 | 5 | 1.25 | | 0.3281 | |
| 22 | 6 | 1.5 | | 0.6250 | |
| 23 | 7 | 1.75 | | 0.8594 | |
| 24 | 8 | 2 | 1 | 1.0000 | |
| 25 | 9 | 2.25 | | 1.0156 | |
| 26 | 10 | 2.5 | | 0.8750 | |
| 27 | 11 | 2.75 | | 0.5469 | |
| 28 | 12 | 3 | 0 | 0.0000 | |
| 29 | | | | | |
| 30 | | | | | |




## 3.3.5 Spline-Interpolation

Während die bisher gezeigten Interpolationsverfahren darauf beruhen, die Datenpunkte mit einer einzigen stetigen Funktion zu beschreiben, geht die Spline-Interpolation einen anderen Weg: Die Interpolationsfunktion wird stückweise aus Polynomfunktionen niedrigen Grades zusammengesetzt. Diese Teilfunktionen beschreiben jeweils für ein Teilintervall die Interpolationsfunktion.

Dadurch kann das Überschwingen, wie bei Polynomen höheren Grades, weitgehend vermieden werden.

Je nachdem welchen Grades die verwendeten Polynomfunktionen sind, spricht man von linearen , , quadratischen, kubischen oder Splines 4. Grades. In der Praxis haben kubische Splines die grösste Verbreitung. Sie bieten ein gutes Interpolationsverhalten mit wenig Überschwingen.

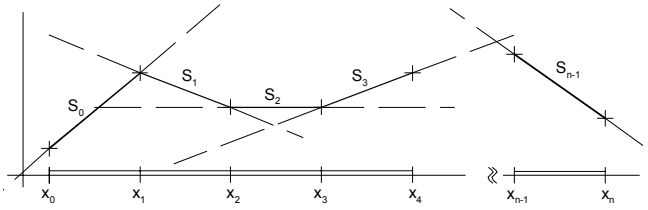
Der Begriff *Splines* ist eigentlich eine Zusammenfassung von verschiedenen Spline-Typen. Je nachdem mit welchem Ansatz gearbeitet wird unterscheidet man:

- natürliche Splines
- B-Splines
- NURBS (Non Uniform Rational B-Splines), u.a.

Sie unterscheiden sich zum Teil stark im Rechenaufwand, Überschwingverhalten und anderen Vorgaben.
Wir beschränken uns nachfolgend auf lineare, quadratische und natürliche kubische Splines. Andere Typen werden analog entwickelt und können in [5] referenziert werden.

### 3.3.5.1 Lineare Splines

Die zu interpolierende Menge von *(n+1)* Datenpunkten wird mit *n* einzelnen linearen Funktionen interpoliert. Ausgehend vom Startpunkt $(x_0,y_0)$ wird für ein Intervall zwischen zwei Stützstellen die Interpolation mit einem Polynom 1. Grades vorgenommen. Daher ist keine knickfreie Beschreibung möglich.




Interpolation mit Splines ersten Grades.
Die Interpolation ist stetig, aber nicht knickfrei.

Konventionsgemäss werden die inneren Datenpunkte zwischen $x_0$ und $x_n$ *Knoten* genannt.

Die Punktemenge wird stückweise mit Funktionen ersten Grades beschrieben:

$$S(x)=\begin{cases} S_0(x) & x_0 \le x \le x_1 \\ S_1(x) & x_1 \le x \le x_2 \\ S_2(x) & x_2 \le x \le x_3 \\ \quad\cdots & \\ S_{n-1}(x) & x_{n-1} \le x \le x_n \end{cases} \tag{3.46}$$

Daher sind die Eigenschaften der linearen Spline-Interpolation:

1. Der Gültigkeitsbereich für *S(x)* ist $[x_0,x_n]$.
2. *S(x)* ist stetig in $[x_0,x_n]$
3. $[x_0,x_n]$ ist in Teilintervalle zerlegt der Art $x_0 < x_1 < x_2 < ... < x_n$. Für jedes Teilintervall ist ein eigenes Polynom ersten Grades definiert.

Damit *S(x)* eine Spline Interpolation ersten Grades ist, muss daher gelten:

1. Interpolationsbedingung $S_i(x_i) = y_i$ erfüllt für alle *i*.
2. Stetigkeit $S_i(x_i)=S_{i+1}(x_i)$ erfüllt für alle *i=1..n-1* (innere Knoten).

**Beispiel:**
Ist *f(x)* eine lineare Spline Interpolation?

$$f(x)=\begin{cases} x & x \in [-1,0] \\ 1-x & x \in [0,1] \\ 2x-2 & x \in [1,1] \end{cases}$$

Antwort:
Nein, obwohl *f(x)* stückweise linear ist, ist *f(x)* im Definitionsbereich [-1,2] nicht stetig, denn:

$$\lim_{f(x)\to 0+} f(x) = 1 \neq \lim_{f(x)\to -0} f(x) = 0$$

#### 3.3.5.1.1 Konstruktion linearer Splines

Ausgangslage bildet eine Tabelle mit den zu interpolierenden Datenpunkten:

| $x_i$ | $x_0$ | $x_1$ | $x_2$ | $x_3$ | … | $x_n$ |
|---|---|---|---|---|---|---|
| $y_i$ | $y_0$ | $y_1$ | $y_2$ | $y_3$ | … | $y_n$ |
| $i$ | 0 | 1 | 2 | 3 | … | $n$ |

Aus der Funktionenlehre folgt für jede einzelne Teilfunktion der linearen Spline-Interpolation:

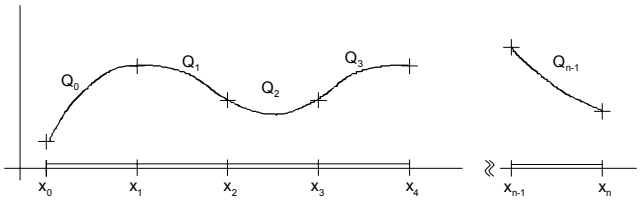
$$S_i(x) = y_i + m_i(x - x_i) \qquad x \in [x_i, x_{i+1}] \tag{3.47}$$

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{3.48}$$

$S_i(x)$ beschreibt dabei die einzelne lineare Funktion für das Intervall $[x_i, x_{i+1}]$ und $m_i$ die zugehörige Steigung.

### 3.3.5.2 Quadratische Splines

Die zu interpolierende Menge von $(n+1)$ Datenpunkten wird mit $n$ einzelnen quadratischen Polynomfunktionen interpoliert. Die Interpolationsfunktion ist knickfrei, jedoch nicht biegungsfrei. Ausgehend vom Startpunkt $(x_0, y_0)$ wird für ein Intervall zwischen zwei Stützstellen die Interpolation mit einem Polynom 2. Grades vorgenommen.




Interpolation mit quadratischen Splines.
Die Interpolation ist stetig, und knickfrei , jedoch nicht biegungsfrei, dh. die Krümmung in den inneren Knoten ist ungleich.

Die Punktemenge wird stückweise mit quadratischen Funktionen beschrieben:

$$Q(x) = \begin{cases} Q_0(x) & x_0 \le x \le x_1 \\ Q_1(x) & x_1 \le x \le x_2 \\ Q_2(x) & x_2 \le x \le x_3 \\ & \dots \\ Q_{n-1}(x) & x_{n-1} \le x \le x_n \end{cases} \tag{3.49}$$

Daher sind die Eigenschaften der quadratischen Spline-Interpolation:

1. Der Gültigkeitsbereich für $Q(x)$ ist $[x_0, x_n]$.
2. $Q(x)$ ist stetig in $[x_0, x_n]$, d.h. $Q_i(x_i) = Q_{i+1}(x_i) \qquad (i{:}1..n-1)$
3. $Q(x)$ ist knickfrei in $[x_0, x_n]$, d.h. $Q'_i(x_i) = Q'_{i+1}(x_i) \qquad (i{:}1..n-1)$
4. $[x_0, x_n]$ ist in Teilintervalle zerlegt der Art $x_0 < x_1 < x_2 < ... < x_n$. Für jedes Teilintervall ist ein eigenes Polynom zweiten Grades definiert.

### 3.3.5.2.1 Konstruktion quadratischer Splines

Es existieren zahlreiche Verfahren zur Berechnung von quadratischen Spline-Funktionen.
Eine einfache Methode zur Berechnung der einzelnen quadratischen Teilfunktionen ist wie folgt:

$$Q_i(x) = \frac{z_{i+1} - z_i}{2(x_{i+1} - x_i)}(x - x_i)^2 + z_i(x - x_i) + y_i \tag{3.50}$$

$z_0 =$ frei wählbar

$$z_{i+1} = -z_i + 2\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) \tag{3.51}$$

Die $z_i$ verkörpern hierbei die Steigungen in den einzelnen Knoten. Die Anfangssteigung ist frei wählbar. Daher ist die Darstellung dieser quadratischen Spline-Interpolation nicht eindeutig. Je nachdem was für eine Anfangssteigung gewählt wurde erhält man einen anderen Satz an Teilfunktionen.

**Beispiel:**
Folgende Datenpunkte sollen mit einer quadratischen Spline-Interpolation beschrieben werden:

| $x_i$ | 0 | 2 | 3 | 4 |
|---|---|---|---|---|
| $y_i$ | 1 | 4 | 5 | 5 |
| $i$ | 0 | 1 | 2 | 3 |

Lösung: $z_0$ wird frei gewählt mit dem Wert $z_0$=0. Damit werden die einzelnen Steigungen:

$$z_1 = -z_0 + 2\left(\frac{y_1 - y_0}{x_1 - x_0}\right) = -0 + 2\left(\frac{4-1}{2-0}\right) = 3$$

$$z_2 = -z_1 + 2\left(\frac{y_2 - y_1}{x_2 - x_1}\right) = -3 + 2\left(\frac{5-4}{3-2}\right) = -1$$

$$z_3 = -z_2 + 2\left(\frac{y_3 - y_2}{x_3 - x_2}\right) = 1 + 2\left(\frac{5-5}{4-3}\right) = 1$$

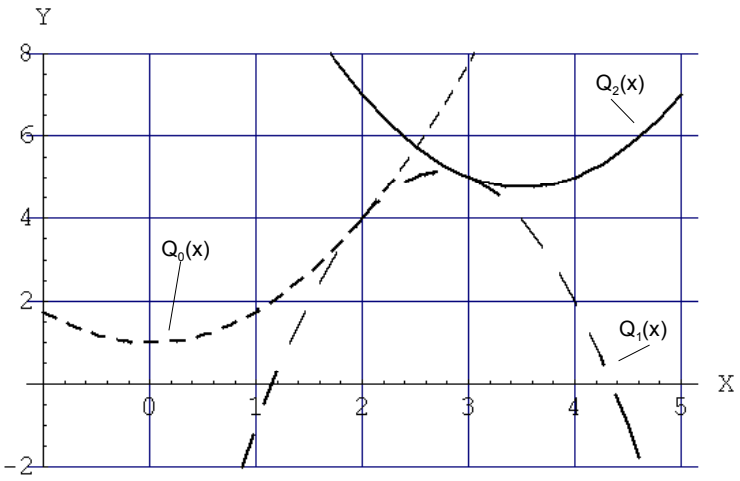
Die Teilpolynome werden demnach:

$$Q_0(x) = \frac{z_1 - z_0}{2(x_1 - x_0)}(x - x_0)^2 + z_0(x - x_0) + y_0 = \frac{3-0}{2(2-0)}(x-0)^2 + 0(x-0) + 1 = \frac{3}{4}x^2 + 1$$

$$Q_1(x) = \frac{z_2 - z_1}{2(x_2 - x_1)}(x - x_1)^2 + z_1(x - x_1) + y_1 = \frac{-1-3}{2(3-2)}(x-2)^2 + 3(x-2) + 4 = -2x^2 + 11x - 10$$

$$Q_2(x) = \frac{z_3 - z_2}{2(x_3 - x_2)}(x - x_2)^2 + z_2(x - x_2) + y_2 = \frac{-5-(-1)}{2(4-3)}(x-3)^2 + -1(x-3) + 5 = x^2 - 7x + 17$$

Die graphische Darstellung der drei Polynomfunktionen zeigt, wie sich die Einzelfunktionen in den Knoten knickfrei anschmiegen:




Plot der Einzelfunktionen gemäss Beispiel zur Interpolation mit quadratischen Splines.

### 3.3.5.3 Kubische Splines

Die Interpolationsfunktion wird hierbei stückweise aus Polynomfunktionen 3. Grades zusammengesetzt. Ausgehend vom Startpunkt $(x_0 y_0)$ wird für ein Intervall zwischen zwei Stützstellen die Interpolation mit einem Polynom 3. Grades vorgenommen. Somit werden $(n+1)$ Datenpunkte durch $n$ kubische Spline-Funktionen interpoliert.

Diese Form der Spline-Interpolation ist vom Ansatz her aufwendiger als eine Polynominterpolation nach Newton oder Lagrange. Wir erhalten aber im Gegenzug wesentlich bessere Interpolationsresultate.

Wir betrachten das anhand des Beispiels der Interpolation von 7 Datenpunkten:

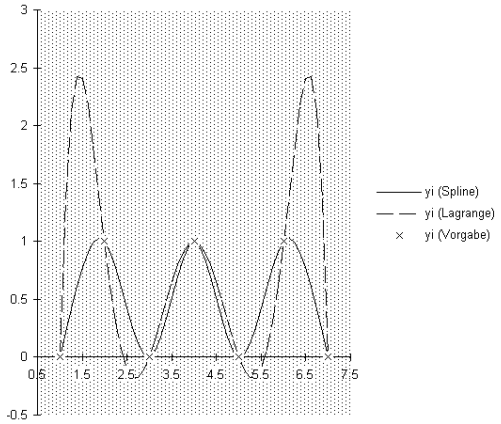
| x | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| y | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

Das Lagrange-Interpolationspolynom lautet:

$$p_6(x) = -63 + 142.933x - 118.84x^2 + 48x^3 - 10.111x^4 + 1.067x^5 - 0.044x^6$$

Die entsprechenden kubischen Splines beschreiben die Interpolation jeweils für ein Teilintervall:

$$f_0(x) = 1.731(x-1) - 0.731(x-1)^3 \qquad x \in [1,2)$$
$$f_1(x) = 1 - 0.462(x-2) - 2.192(x-2)^2 + 1.654(x-2)^3 \qquad x \in [2,3)$$
$$f_2(x) = 0.115(x-3) + 2.769(x-3)^2 - 1.885(x-3)^3 \qquad x \in [3,4)$$
$$f_3(x) = 1 - 2.885(x-4)^2 + 1.885(x-4)^2 \qquad x \in [4,5)$$
$$f_4(x) = -0.115(x-5) + 2.769(x-5)^2 - 1.654(x-5)^3 \qquad x \in [5,6)$$
$$f_5(x) = 1 + 0.462(x-6) - 2.192(x-6)^2 + 0.731(x-6)^3 \qquad x \in [6,7]$$




Aus der Grafik ersieht man, dass die Spline-Interpolation wesentlich bessere Zwischenwerte liefert als die Lagrange-Interpolation.

### 3.3.5.4 Natürliche kubische Splines

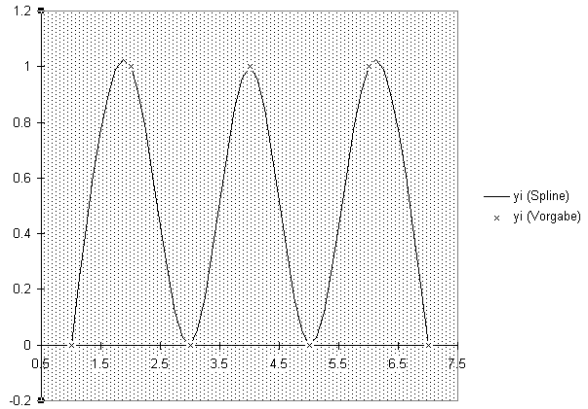
Wir gehen von einer Menge *(n+1)* verschiedener Datenpunkten $(x_k, y_k)$, $k=0..n$ aus und verlangen, dass diese Punkte nach aufsteigenden *x*-Werten sortiert sind:

$$x_0 < x_1 < x_2 < ... < x_n$$

Auf jedem Teilintervall $[x_k,x_{k+1}]$ für $k=0,..,n-1$ definieren wir ein Polynom 3. Grades durch

$$p_k(x) := d_k + c_k(x - x_k) + b_k(x - x_k)^2 + a_k(x - x_k)^3 \tag{3.52}$$

Die Interpolation im gesamten Bereich wird dann durch *n* Spline-Polynome beschrieben:




Für jeden inneren Punkt $x_k$ , $k=1,..,n-1$ verlangen wir, dass die Anschlussbedingung für die nachfolgende Kurve erfüllt ist, d.h. die nachfolgende Polynomfunktion muss 'glatt' auf die Vorgängerfunktion folgen.

Wir haben einen glatten Übergang, wenn folgende Bedingungen in den inneren Punkten erfüllt sind:

| | | | |
|---|---|---|---|
| Funktionswerte stimmen überein: | $p_k(x_k) = p_{k+1}(x_k)$ | (Stetigkeit) | (3.53) |
| Tangentensteigungen stimmen überein: | $p_k'(x_k) = p_{k+1}'(x_k)$ | (Knickfrei) | (3.54) |
| Krümmungen stimmen überein: | $p_k''(x_k) = p_{k+1}''(x_k)$ | (Biegungsfrei) | (3.55) |

Ausserdem sollen die Interpolationsbedingungen erfüllt sein:

$$y_k = p_k(x_k) \tag{3.56}$$

Für die sog. *natürliche Spline-Interpolation* wird festgelegt, dass die Krümmung in den Anfangs- und Endpunkten 0 betragen soll:

$$p_0''(x_0) = p_{n-1}''(x_n) = 0 \tag{3.57}$$

**Beispiel**
Die Funktion *S(x)* soll die Punkte als natürliche kubische Spline-Interpolation interpolieren:

| $x$ | -1 | 0 | 1 |
|---|---|---|---|
| $S(x)$ | 1 | 2 | -1 |

$$S(x)=\begin{cases} ax^3+bx^2+cx+d & -1\le x\le 0 \\ ex^3+fx^2+gx+h & 0\le x\le 1 \end{cases}$$

Damit die folgende Funktion *S(x)* eine kubische Spline-Interpolation darstellt, müssen die Parameter *a,b,c,d,e,f,g,h* so beschaffen sein, dass die Bedingungen (3.53) bis (3.55) erfüllt sind.

Aus der Interpolationsbedingung (3.54) folgt:

$$d=h=2$$
$$-a+b-c=-1$$
$$e+f+g=-3$$

Weil im Anschlusspunkt $x_1=0$ die Tangentensteigungen übereinstimmen müssen, gilt ferner:

$$S'(x)=\begin{cases} 3ax^2+2bx+c \\ 3ex^2+2fx+g \end{cases}$$
$$\Rightarrow c=g$$

Ebenso müssen im Anschlusspunkt die Krümmungen übereinstimmen. Die Auswertung bei $x_1=0$ liefert die Werte der Parameter *b* und *f*:

$$S''(x)=\begin{cases} 6ax+2b \\ 6ex+2f \end{cases}$$
$$\Rightarrow b=f$$

Aus der Bedingung (3.57) für die natürliche kubische Spline-Interpolation, folgt eine Anfangs- und Endkrümmung von Null. Wir erhalten daraus zwei zusätzliche Gleichungen:

$$3a=b$$
$$3e=-f$$

Nach Auflösen aller Gleichungen erhalten wir das Resultat *a=-1, b=-3, c=-1, d=2, e=1, f=-3* und *h=2*.

### 3.3.5.5 Systematische Konstruktion der Spline-Polynome

Grundlage für die Berechnung der Spline-Polynomkoeffizienten bilden generell die Krümmungen. Wir definieren die Krümmung $z_i$ im Punkt *i*:

$$z_i=S''(x_i) \qquad (0\le i\le n,\quad n\text{: Anzahl Datenpunkte}-1) \tag{3.58}$$

Für eine natürliche kubische Spline-Interpolation sind einzig die Werte $z_0=z_n=0$ definiert. Die Krümmungen $z_1,..,z_{n-1}$ müssen berechnet werden.

Würden wir alle Krümmungen kennen, könnten die Spline-Polynome konstruiert werden. Wir haben im Beispiel gesehen, dass das kubische Polynom in der zweiten Ableitung ein lineares Polynom darstellt. Es nimmt an den Endpunkten eines Intervalls $[x_i,x_{i+1}]$ die Krümmungswerte $z_i$, resp. $z_{i+1}$ an. Somit gilt für die Krümmung im Intervall $[x_i,x_{i+1}]$ :

$$S_i''(x) = \frac{z_{i+1}}{h_i}(x - y_i) + \frac{z_i}{h_i}(y_{i+1} - x) \qquad h_i = x_{i+1} - x_i \qquad (0 \le i \le n-1) \tag{3.59}$$

Wir integrieren $S_i''(x)$ zweimal und erhalten so die Gleichung für das natürliche kubische Spline-Polynom $S_i(x)$ bei gegebenen Krümmungen an den Anfangs- und Endpunkten:

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - y_i)^3 + \frac{z_i}{6h_i}(y_{i+1} - x)^3 + cx + d \qquad (0 \le i \le n-1) \tag{3.60}$$

Die Grössen *c* und *d* sind Integrationskonstanten. Ihre Werte können unter Zuhilfenahme der Interpolationsbedingung $S_i(x_i) = y_i$ und $S_i(x_{i+1}) = y_{i+1}$ bestimmt werden und wir erhalten:

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - y_i)^3 + \frac{z_i}{6h_i}(y_{i+1} - x)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x - y_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(y_i - x) \qquad (0 \le i \le n-1) \tag{3.61}$$

Sind die Krümmungen $z_0..z_n$ bekannt, können mit dieser Gleichung die kubischen Splines $S_0(x),..,S_{n-1}(x)$ bestimmt werden.

Die Krümmungen werden auch dadurch bestimmt, dass die Tangentensteigung im Anschlusspunkt übereinstimmen muss, d.h. $S_i'(x_i) = S_{i-1}'(x_i)$ für $1 \le i \le n\text{-}1$, also für alle inneren Knoten der Spline-Interpolation (Knickfreiheit im Übergang).

Wir leiten (3.61) ab und erhalten:

$$S_i'(x) = \frac{z_{i+1}}{2h_i}(x - y_i)^2 + \frac{z_i}{2h_i}(y_{i+1} - x)^2 + \frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1} - \frac{y_i}{h_i} + \frac{h_i}{6}z_i \tag{3.62}$$

Die Auswertung der Steigung bei $x_i$ ergibt:

$$\begin{aligned} S_i'(x_i) &= \frac{z_{i+1}}{2h_i}(x_i - y_i)^2 + \frac{z_i}{2h_i}(y_{i+1} - x_i)^2 + \frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1} - \frac{y_i}{h_i} + \frac{h_i}{6}z_i \\ &= -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + \frac{1}{h_i}(y_{i+1} - y_i) \end{aligned} \tag{3.63}$$

Ebenso wird $S_{i-1}'(x_i)$:

$$S_{i-1}'(x_i) = -\frac{h_i}{6}z_{i-1} - \frac{h_i}{6}z_i + \frac{1}{h_{i-1}}(y_i - y_{i-1}) \tag{3.64}$$

Wir setzen beide Steigungen gleich und erhalten das Gleichungssystem für die Krümmungen:

$$h_{i-1}z_{i-1} + 2(h_{i-1} - h_i)z_i + h_i z_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_i}(y_i - y_{i-1}) \qquad \begin{pmatrix} 1 \le i \le n-1 \\ n\text{: Anzahl Datenpunkte} - 1 \end{pmatrix} \tag{3.65}$$

Dieses tridiagonale Gleichungssystem ist immer und leicht lösbar. Zusammengefasst lassen sich die gesuchten Teilfunktionen für die natürliche kubische Spline-Interpolation bestimmen, wobei wir die Krümmungen aus dem vorherigen Ansatz nun mit $y_0'',..,y_n''$ beschreiben:

1. Vorgaben für Anfangskrümmung: $y_0'' := 0, y_n'' := 0$ *n*: Anzahl Datenpunkte - 1

2. Schrittweite zwischen den Datenpunkten: $h_i := x_{i+1} - x_i \qquad i: 0,\ldots,n-1$ (3.66)

3. Berechnung der $y''_i$ aus dem tridiagonalen LGS:

$$h_{i-1}y''_{i-1} + 2(h_{i-1}+h_i)y''_i + h_i y''_{i+1} = \frac{6}{h_i}(y_{i+1}-y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \quad i: 1,..,n-1 \tag{3.67}$$

4. Polynomkoeffizienten für den Spline *i*:

$$\begin{aligned} a_i &= \frac{1}{6h_i}(y''_{i+1} - y''_i) \qquad\qquad i: 0,\ldots,n-1 \\ b_i &= \frac{y''_i}{2} \\ c_i &= \frac{1}{h_i}(y_{i+1}-y_i) - \frac{h_i}{6}(y''_{i+1} + 2y''_i) \\ d_i &= y_i \end{aligned} \tag{3.68}$$

**Beispiel:**
Bestimmen des ersten kubischen Spline-Polynoms $p_0(x)$ für die Datenpunkte:

| $i$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $x$ | -2 | -1 | 0 | 1 | 2 |
| $y$ | 1 | -2 | 0 | -2 | 5 |

- Bestimmen der Schrittweiten $h_i$ und setzen der Anfangskrümmungen:

$$h_i = x_{i+1} - x_i \qquad (0 \le i \le 3)$$
$$y''_0 = y''_4 = 0$$

| $i$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $h_i$ | 1 | 1 | 1 | 1 |

- Lineares Gleichungssystem für die Krümmungen $y''_i$:

$$\begin{array}{rrrcl} 4y''_1 & + y''_2 & & = & 30 \\ y''_1 & + 4y''_2 & + y''_3 & = & -24 \\ & y''_2 & 4y''_3 & = & 54 \end{array} \qquad \rightarrow \qquad \begin{array}{l} y''_1 = 150/14 \\ y''_2 = -180/14 \\ y''_3 = 234/14 \end{array}$$

- Bestimmen der Polynomkoeffizienten für $p_0(x)$:

$i = 0$

$$a_0 = \frac{25}{14}, \quad b_0 = 0, \quad c_0 = \frac{-67}{14}, \quad d_0 = 1$$

$$\begin{aligned} p_0(x) &= \frac{25}{14}(x+2)^3 - \frac{-67}{14}(x+2) + 1 \\ &= \frac{1}{14}\left(25x^3 + 150x^2 + 233x + 80\right) \qquad x \in [-2,-1] \end{aligned}$$

### 3.3.5.6 Implementierung als EXCEL-Funktion

Eine Lösung in Visual-Basic für EXCEL, die als EXCEL-Array-Funktion die kubischen Splines für *n* Datenpunkte bestimmt.
Beim Entwurf wurde Wert darauf gelegt, dass die Berechnung genau nach den vorgestellten Formeln erfolgt, obwohl fallweise gewisse Vereinfachungen hätten vorgenommen werden können.

```
'
' Natuerliche Spline Interpolation2
' Bestimmen der natuerlichen Spline-Interpolation fuer eine beliebige Anzahl n (>2)
' Datenpunkte. Die Datenpunkte muessen bezueglich x-Werte in aufsteigender Reihenfolge
' sortiert sein.
' Das Resulat wird als 2dimensionales Array retourniert (EXCEL Array - Funktion)
' Die Eintraege jeder Zeile enthalten eine Spline-Funktion in der Notation:
'   a0 + a1 x + a2 x^2 + a3 x^3
' Es werden n-1 Spline-Funktionen berechnet, wobei jeweils eine Polynomfunktion zwischen zwei
' Datenpunkten gilt.
'
' Parameter: x: Liste mit den Stuetzstellen
'            y: Liste mit den Stuetzwerten
' Referenz: H.P. Blau, Skript Vorlesung Numerik I, Uni Bern
' Datum: 21.8.1995, 22.9.1996, 15.9.1997
' Sprache: MS VBA EXCEL97
'
Function KSpline2(x, y)
Dim h               ' Dynamisches Array fuer Schrittweite '

Dim SplinePoly      ' Dynamisches Array mit Koeffizienten fuer Spline-Polynome '
Dim AnzX            ' Anzahl Stuetzstellen in der Liste '
Dim AnzY            ' Anzahl Stuetzwerte in der Liste '
Dim G               ' Koeffizientenmatrix fuer lin. Gleichungssystem '
Dim y2i             ' Kruemmungen an den inneren Datenpunkten '
Dim Y2              ' Kruemmungen an den Datenpunkten (+ aeussere Datenpunkte) '
Dim B               ' Konstantenvektor (rechte Seite des LGS) '
Dim N               ' Anzahl benoetigte Splines fuer Interpolation (Anzahl Datenpunkte -1) '
Dim i, j
Dim xi

  ' Parameter kontrollieren '
  AnzX = x.Count
  AnzY = y.Count
  If AnzX <> AnzY Then     ' Keine Interpolation wenn Anzahl x-Werte <> Anzahl y-Werte! )
     KSpline2 = CVErr(xlErrValue)
     Exit Function
  End If
  If AnzX < 2 Then         ' Mindestens 2 Datenpunkte fuer Interpolation benötigt '
     KSpline2 = CVErr(xlErrValue)
     Exit Function
  End If

  ' Spline Berechnung Vorbereitung '
  N = AnzX - 1                            ' hoechster Index (mathematisch) '
  ReDim h(0 To N - 1)                     ' Array fuer die Schrittweiten '
  ReDim SplinePoly(0 To N - 1, 0 To 3)

  For i = 0 To N - 1                        ' Schrittweiten bestimmen, a(i) setzen '
     h(i) = x(i + 2) - x(i + 1)             ' h0 = x(2)-x(1), h1=x(3)-x(2),... '
  Next i

  ' Lin. Gleichungssystem für die Kruemmungen aufbauen. Es ensteht ein tridiagoinales Gleichungssystem fuer
  ' die inneren Kruemmungen. Der Einfachheit halber mit einer Matrixinversion geloest. '
  ReDim G(1 To N - 1, 1 To N - 1)         ' Koeffizientenmatrix fuer benoetigte Azahl dimensionieren '
  ReDim c(1 To N - 1)                     ' Konstantenvektor fuer das Gleichungssystem (rechte Seite) '
  ReDim y2i(1 To N - 1)                   ' Kruemmungen an den inneren Punkten '
  ReDim Y2(0 To N)                        ' Alle Kruemmungen '


  For i = 1 To N - 1       ' Matrix mit 0 initialisieren '
    For j = 1 To N - 1
       G(i, j) = 0
    Next j
  Next i
  For i = 1 To N - 1        ' Koeffizienten bestimmen und einschreiben '
     If i > 1 Then G(i, i - 1) = h(i - 1)
     G(i, i) = 2 * (h(i - 1) + h(i))
     If i < N - 1 Then G(i, i + 1) = h(i)
     c(i) = 6 / h(i) * (y(i + 2) - y(i + 1)) - 6 / h(i - 1) * (y(i + 1) - y(i))
  Next i
  G = Application.MInverse(G)
  y2i = Application.MMult(c, G)

  ' Kruemmungsvektor aufbauen: Innere Kruemmungen + Kruemmung an den aeusseren Punkten '
  Y2(0) = 0: Y2(N) = 0   ' Kruemmung am Anfang und Ende = 0 (Def. natuerlicher Spline) '
  For i = 1 To N - 1
     Y2(i) = y2i(i)
  Next i

  ' Polynomkoeffizienten ak,..,dk fuer jeden Spline aufbauen und einschreiben '
  For i = 0 To N - 1
      SplinePoly(i, 0) = y(i + 1)       'ak '
      SplinePoly(i, 1) = 1 / h(i) * (y(i + 2) - y(i + 1)) - h(i) / 6 * (Y2(i + 1) + 2 * Y2(i))  'bk'
      SplinePoly(i, 2) = Y2(i) / 2      'ck'
      SplinePoly(i, 3) = 1 / (6 * h(i)) * (Y2(i + 1) - Y2(i))  'dk'
  Next i
```

```
  'Polynomnotation a0 + a1 (x-xi) + a2 (x-xi)^2 + a3 (x-xi)^3 in a0' + a1' x + a2' x^2 + a3' x^3 ueberfuehren
  For i = 0 To N - 1
    xi = x(i + 1)     'Startwert des Definitionintervalles fuer pi(x)
    SplinePoly(i, 0) = SplinePoly(i, 0) + (((-SplinePoly(i, 3) * xi + SplinePoly(i, 2)) * xi) - SplinePoly(i, 1)) * xi
    SplinePoly(i, 1) = SplinePoly(i, 1) - 2 * xi * SplinePoly(i, 2) + 3 * xi * xi * SplinePoly(i, 3)
    SplinePoly(i, 2) = SplinePoly(i, 2) - 3 * xi * SplinePoly(i, 3)
  Next i

  KSpline2 = SplinePoly

End Function
```

Ein Test für vorheriges Beispiel

| x | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| y | 1 | -2 | 0 | -2 | 5 |

liefert die Polynomkoeffizienten:

**Interpolation mit kubischen Splines**

Vorgaben:

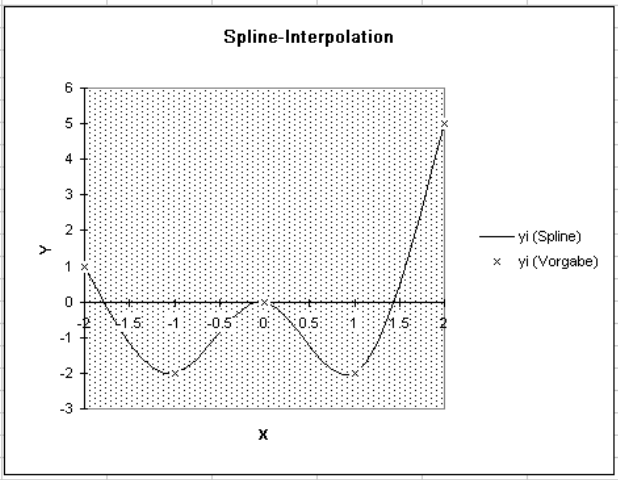
| x | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| y | 1 | -2 | 0 | -2 | 5 |

Koeffizienten der Spline-Polynome ax^3+bx^2+cx+d:

| | d | c | b | a |
|---|---|---|---|---|
| $p_0(x)$ | 5.7143 | 16.6429 | 10.7143 | 1.7857 |
| $p_1(x)$ | 0.0000 | -0.5000 | -6.4286 | -3.9286 |
| $p_2(x)$ | 0.0000 | -0.5000 | -6.4286 | 4.9286 |
| $p_3(x)$ | 7.7143 | -23.6429 | 16.7143 | -2.7857 |

Grafische Darstellung:

| i | $x_i$ | $y_i$ (Spline) | $y_i$ (Vorgabe) |
|---|---|---|---|
| 0 | -2 | 1.0000 | 1 |
| 1 | -1.875 | 0.4053 | |
| 2 | -1.75 | -0.1685 | |
| 3 | -1.625 | -0.7005 | |
| 4 | -1.5 | -1.1696 | |
| 5 | -1.375 | -1.5551 | |
| 6 | -1.25 | -1.8359 | |
| 7 | -1.125 | -1.9912 | |
| 8 | -1 | -2.0000 | -2 |
| 9 | -0.875 | -1.8525 | |
| 10 | -0.75 | -1.5837 | |
| 11 | -0.625 | -1.2395 | |
| 12 | -0.5 | -0.8661 | |
| 13 | -0.375 | -0.5093 | |
| 14 | -0.25 | -0.2154 | |
| 15 | -0.125 | -0.0303 | |
| 16 | 0 | 0.0000 | 0 |
| 17 | 0.125 | -0.1533 | |
| 18 | 0.25 | -0.4498 | |
| 19 | 0.375 | -0.8316 | |
| 20 | 0.5 | -1.2411 | |
| 21 | 0.625 | -1.6204 | |
| 22 | 0.75 | -1.9118 | |
| 23 | 0.875 | -2.0576 | |
| 24 | 1 | -2.0000 | -2 |
| 25 | 1.125 | -1.6963 | |
| 26 | 1.25 | -1.1641 | |
| 27 | 1.375 | -0.4360 | |
| 28 | 1.5 | 0.4554 | |
| 29 | 1.625 | 1.4773 | |
| 30 | 1.75 | 2.5971 | |



## 3.4 Aufgaben

### Ausgleichsrechnung:

1. Wie wird bei einer gegebenen Ausgleichsfunktion und den Datenpunkten die Fehlerquadratsumme bestimmt? Zeigen Sie das Vorgehen konkret am Beispiel:

2. Hat eine polynomiale Ausgleichsfunktion höheren Grades immer eine kleinere oder gleiche Fehlerquadratsumme als eine Ausgleichsfunktion niedrigeren Grades?

3. Weshalb werden polynomiale Ausgleichsfunktionen selten höher als bis zum 4. Grad entwickelt? Wann werden polynomiale Ausgleichsfunktionen besonders kritisch?

4. Welche Ausgleichsfunktion ist zu bevorzugen?

   $f_1(x) = 1.8 \cdot 1.7^x$

   $f_2(x) = 0.68x^2 + 0.69x + 1.36$

| $x$ | -1 | 0 | 2 | 4 |
|---|---|---|---|---|
| $y$ | 1 | 2 | 5 | 15 |

5. An einer Diode werden die Ströme und Spannungen gemessen:

| $U_D$ | 0.4 | 0.5 | 0.6 | 0.65 | 0.7 | 0.75 | 0.8 | 0.85 |
|---|---|---|---|---|---|---|---|---|
| $I_D$ | 4.3E-7 | 6.2E-6 | 8.8E-5 | 3.3E-4 | 1.2E-3 | 4.5E-3 | 1.5E-2 | 4.9E-2 |

   Zu Bestimmen ist die exponentielle Ausgleichsfunktion für die obigen Datenwerte bezüglich $I_D$.

6. Bestimmen Sie die Parameter $b_p b_2$ und $b_3$ für die Ausgleichsfunktion vom Typ $y = b_1 + b_2 x + b_3 e^x$ für die Datenwerte:

| $x$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $y$ | 0 | 1 | 3 | 4 |

7. Gegeben sind die Datenpunkte:

| $x$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $y$ | 2 | 1 | 0.9 | 0.5 |

   Durch diese Punkte lege man die Ausgleichsfunktionen vom Typ

   a.) $y = a + bx + ce^x$

   b.) $y = a + bx + cx^2$

   c.) $y = a + bx + \frac{c}{x}$

   Man ermittle nachher welche der drei Funktionen die Vorgabewerte am besten annähert.

**Polynom-Interpolation:**

8. Mit Hilfe der Lagrange'schen Interpolationsformel ist ein Interpolationspolynom zu bestimmen, dass die Werte interpoliert:

| $x$ | 0 | 1 | 4 | 6 |
|---|---|---|---|---|
| $y$ | 1 | -1 | 1 | -1 |

9. Zeigen Sie, dass die beiden Polynome

$$p(x) = 3 + 2(x-1) + 4(x-1)(x+2) \qquad q(x) = 4x^2 + 6x - 7$$

die Punkte

| $x$ | 1 | -2 | 0 |
|---|---|---|---|
| $y$ | 3 | -3 | -7 |

interpolieren und nicht im Widerspruch zum Eindeutigkeitssatz stehen.

10. Zeigen Sie, dass die beiden Polynome

$$p(x) = 5x^3 - 27x^2 + 45x - 21 \qquad q(x) = x^4 - 5x^3 + 8x^2 - 5x + 3$$

die Punkte

| $x$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $y$ | 2 | 1 | 6 | 47 |

interpolieren und nicht im Widerspruch zum Eindeutigkeitssatz stehen.

11. Bestimmen Sie das Interpolationspolynom, welches die Funktionsdaten liefert:

| $x$ | 1.0 | 2.0 | 2.5 | 3.0 | 4.0 |
|---|---|---|---|---|---|
| $f(x)$ | -1.5 | -0.5 | 0.0 | 0.5 | 1.5 |

12. Man nimmt an, dass die nachfolgende Tabelle von einem kubischen Polynom stammt. Wie kann dies geprüft werden? Erklären Sie das Vorgehen!

| $x$ | -2 | -1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| $y$ | 1 | 4 | 11 | 16 | 13 | -4 |

13. Sie haben als Ausgangslage folgenden Datensatz:

| $x$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $y$ | 2 | 2 | 4 | 5 |

Prüfen Sie, ob nachfolgende Funktionen Interpolations- oder Ausgleichsfunktionen bezüglich $x$ sind:

| *Ausgleichs -funktion* | *Interpol.- Funktion* | *Weder Interpol.- noch Ausgl.-Funktion* | *Funktion* |
|---|---|---|---|
| ☐ | ☐ | ☐ | $f(x) = 0.5x + 1.0$ |
| ☐ | ☐ | ☐ | $f(x) = 0.25x^2 - 0.15x + 1.75$ |
| ☐ | ☐ | ☐ | $f(x) = 0.125x^2 + 0.525x + 1.125$ |
| ☐ | ☐ | ☐ | $f(x) = -0.5x^3 + 4x^2 - 8.5x + 7$ |
| ☐ | ☐ | ☐ | $f(x) = -0.75x^3 + 5.75x^2 - 12x + 9$ |

**Splines:**

14. Bestimmen Sie die natürliche Spline Interpolationsfunktion, die $f(x) = x^6$ im Intervall [0,2] mit den Knoten bei 0,1,2.
    Anmerkung: Knoten sind die *x*-Werte der Punkte in der Interpolationsfunktion, wo ein Wechsel zu der nächsten Polynomfunktion erfolgt, also das nächste Teilintervall interpoliert wird.

15. Zeigen Sie, dass

$$f(x) = \begin{cases} 28 + 25x + 9x^2 + x^3 & -3 \le x - 1 \\ 26 + 19x + 3x^2 - x^3 & -1 \le x \le 0 \\ 26 + 19x + 3x^2 - 2x^3 & 0 \le x \le 3 \\ -163 + 208x - 60x^2 + 5x^3 & 3 \le x \le 4 \end{cases}$$

    eine natürliche kubische Spline-Interpolation darstellt.

16. Bestimmen Sie *a,b* und c so, dass *S(x)* eine kubische Spline-Interpolation darstellt:

$$S(x) = \begin{cases} x^3 & 0 \le x \le 1 \\ \frac{1}{2}(x-1)^3 + a(x-1)^2 + b(x-1) + c & 1 \le x \le 3 \end{cases}$$

17. Bestimmen Sie eine natürliche kubische Spline-Interpolation, welche die nachfolgenden beiden Punkte interpoliert:

| $x$ | $x_0$ | $x_1$ |
|---|---|---|
| $y$ | $y_0$ | $y_1$ |

18. Sie kennen die Vorgaben *f(0)=0, f(1)=1.1752, f'(0)=1* und *f'(1)=1.5431*. Bestimmen Sie ein kubisches Interpolationspolynom $p_3(x)$ für diese Punkte! Ist es auch ein natürliches Spline-Interpolationspolynom?

19. Bestimmen Sie mit einer Handrechnung die natürliche kubische Spline-Interpolation für die Datenpunkte:

| $x$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $y$ | 0 | 1 | 0 | 1 | 0 |

## Korrelation

20. Die Preisentwicklung eines Artikels wird mit Hilfe einer Stichprobe untersucht. Man hat zehn Lieferanten zufällig ausgewählt und den Preis dieses Artikels in der Mitte und am Ende des Jahres festgestellt. Man bestimme $r_{xy}$.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Preis ende Juni | 1.86 | 1.57 | 2.41 | 2.26 | 2.31 | 1.77 | 2.49 | 1.54 | 1.66 |
| Preis ende Dezember | 1.93 | 1.61 | 2.48 | 2.34 | 2.32 | 1.85 | 2.55 | 1.58 | 1.69 |

21. Man bestimme den linearen Korrelationskoeffizienten für die Mathematik- und Physiknoten aus der Tabelle:

| | | Mathematiknoten | | | | | |
|---|---|---|---|---|---|---|---|
| | | 40-49 | 50-59 | 60-69 | 70-79 | 80-89 | 90-99 |
| Physiknoten | 90-99 | | | | 2 | 4 | 4 |
| | 80-89 | | | 1 | 4 | 6 | 5 |
| | 70-79 | | | 5 | 10 | 8 | 1 |
| | 60-69 | 1 | 4 | 9 | 5 | 2 | |
| | 50-59 | 3 | 6 | 6 | 2 | | |
| | 40-49 | 3 | 5 | 4 | | | |