8 Tabellenkalkulation

Ulf Schreier

8.1 Einführung

Tabellenkalkulation

Tabellenkalkulation ist eine Verallgemeinerung des betriebswirtschaftlichen Rechnens auf Formularen. Ein typisches Beispiel hierfür ist das Berechnen von Steuern auf Formularen. Dabei werden Einzelwerte, die der Steuerberechnung zugrunde liegen, in Formularfeldern eingetragen. Der Inhalt dieser Felder wird dann in anderen Feldern aufsummiert, subtrahiert usw., um letztlich die genaue Steuerschuld zu berechnen. Dabei werden nicht nur die Grundrechenarten benötigt, sondern es müssen auch komplexe Formeln berechnet werden. Ferner sind bei vielen Steuerberechnungen auch Fallunterscheidungen wichtig nach dem Muster: „Wenn die Summe der Werbungskosten höher ist als der Pauschbetrag, dann trage die tatsächliche Höhe der Werbungskosten ein, ansonsten setze den Pauschbetrag ein." Systeme für die Tabellenkalkulation (die im folgenden kurz als TK-Systeme bezeichnet werden) vereinfachen das Rechnen von solchen Anwendungen erheblich, indem sie eine Computerunterstützung für diese Arbeit anbieten.

Verallgemeinerung des Rechnens auf Formularen

Alternative zum klassischen Programmieren

Formulare sind nicht nur für Steuerberechnungen nützlich, sondern auch für viele andere betriebswirtschaftliche Gebiete: für Marketing, Controlling, Buchhaltung, Projektmanagement, Kostenrechnung, Investitionsplanung etc. Aber auch für zahlreiche ingenieurwissenschaftliche Kalkulationen, vom Bauingenieurwesen bis zur Verfahrenstechnik, die neben den Grundrechenarten auch viele mathematische, z.B. trigonometrische, d.h. in bestimmter Weise programmierbarer Funktionen benötigen, bieten sich TK-Systeme als „intelligenter Taschenrechner" und als Alternative zum klassischen Programmieren an.

Geschichte der Tabellenkalkulation

Das Rechnen mit Formularen ist seit vielen Jahrhunderten bekannt. Die Geschichte der Tabellenkalkulation als EDV-Programm ist dagegen noch sehr kurz. Das erste Programm zur Tabellenkalkulation, das eine größere Verbreitung fand, war **VisiCalc**. Es wurde 1978 von David Bricklin erfunden, einem Studenten der Harvard Business School, der als Übungsaufgabe aufwendige Berechnungen unter Einsatz von tabellarischen Formularen durchführen musste. Er entwickelte mit Freunden ein Programm, das die Tabellen grafisch anzeigte. Es entstand ein Produkt, das auf den ersten PCs (z.B. Apple Computer) weite Verbreitung fand. Auf dem gerade entstehenden DOS-PC-Markt war jedoch ein anderes Produkt schneller verfügbar. **Lotus 1-2-3**, das von Mitch Kapor entwickelt wurde, der 1982 die Firma Lotus Development Corporation gründete. Bereits 1985 wurde VisiCalc von Lotus aufgekauft und nicht mehr weiterentwickelt. Mit der Etablierung einer neuen Systemplattform (Windows-PC) konnte sich schließlich das heute am meisten verbreitete System entfalten: **Excel** von Microsoft. Lange Zeit, von 1987 bis 1992, war Excel das einzige am Markt erhältliche Programm zur Tabellenkalkulation für das Betriebssystem Windows. Erst anschließend konnten sich eine Reihe von Konkurrenz-Produkten für das Betriebssystem Windows entwickeln. Eine weite Verbreitung erlangten neben Excel z.B. Lotus 1-2-3

und eine Reihe von „integrierten“ Systemen (sie besitzen noch weitere Komponenten wie Textprozessoren, Grafikwerkzeuge, Datenbank-Manager), z. B. StarOffice, WordPerfecht und Works (letzteres stammt ebenfalls von Microsoft und enthält eine vereinfachte Version von Excel).

Das Ziel dieses Kapitels soll sein, die reichhaltigen Möglichkeiten aufzuzeigen, die TK-Systeme bieten, um Anwendungen aufzubauen. Wir verwenden hierfür ein durchgängiges Beispiel aus dem ingenieurwissenschaftlichen Umfeld, an dem sich sehr viele Funktionen eines TK-Systems demonstrieren lassen. Nach dem Durcharbeiten dieses Beispiels sollte der Leser in der Lage sein, das hinzugewonnene Wissen umzusetzen, um eigene Anwendungen zu gestalten. Zur Realisierung des Beispieles verwenden wir das Produkt **Microsoft Excel**, das z. Z. der Marktführer auf dem Gebiet der TK-Systeme ist. Unabhängig von der Marktposition ist Excel vor allem deshalb interessant, weil es ein offenes System ist, das seine internen Schnittstellen freilegt. Diese Schnittstellen sind als Komponenten aufgebaut, auf die mit Hilfe der mitgelieferten und in das Produkt integrierten Programmiersprache VisualBasic zugegriffen werden kann. Dies ermöglicht die Entwicklung von eigenen maßgeschneiderten Lösungen in sehr flexibler Weise.

Microsoft Excel

Lotus 1-2-3

Ein anderes weitverbreitetes System ist **Lotus 1-2-3**, das in seiner Basisfunktionalität sehr viele Gemeinsamkeiten mit MS Excel hat. Ein wesentlicher Unterschied besteht bei der Programmierung, da statt einer kompletten Programmiersprache, wie Visual Basic, eine einfachere Makrosprache eingesetzt wird.

Dieses Kapitel enthält keine in die Breite gehende allgemeine Einführung in die elementare Bedienung von Excel; stattdessen geben wir Hinweise auf Online-Tutorials und auf die Online-Hilfe, die es ermöglichen, sich direkt am System sitzend einzuarbeiten. Das Kapitel ist so aufgebaut, dass wir in Abschnitt 8.2 die wichtigsten Eigenschaften von TK-Systemen im Allgemeinen und von Excel im Besonderen erläutern. In Abschnitt 8.3 werden wir ein umfangreiches Beispiel vorstellen, an dem sich sehr gut die Vorzüge von TK-Systemen demonstrieren lassen. Das Beispiel beschäftigt sich mit der Auswertung von naturwissenschaftlichen oder technischen Experimenten, wie sie in Praktika im ingenieurwissenschaftlichen Studium vorkommen. Das hat den Nebeneffekt, dass die Beispielanwendung auch direkt im Studium benutzt werden kann. In Abschnitt 8.4 entwickeln wir eine Lösung für das vorausgegangene Beispiel, die nur die generischen Funktionen von Excel verwendet. Das Wort „generisch“ benutzen wir hier im Sinne der Informatik: Es werden nur die allgemeinen Funktionen von Excel verwendet. Diese Funktionen sind für eine große Klasse von Anwendungen gedacht und nicht für ein Spezialproblem. In Abschnitt 8.5 stellen wir Excel als ein offenes System vor und entwerfen eine Programmskizze, mit welcher sich die Basislösung zu einer maßgeschneiderten Speziallösung ausbauen lässt, die die Bedienung auf wenige Handgriffe verkürzt und die nur noch sehr eingeschränktes Wissen über Excel vom Benutzer verlangt. Eine langwierige Einarbeitung in Excel entfällt dadurch also. Ferner werden wir zeigen, wie der Leser die Programmskizze zu einer kompletten Lösung weiterentwickeln kann.

Beispiel: Auswertung von Experimenten

8.2 Eigenschaften von Tabellenkalkulationssystemen

8.2.1 Kernfunktionalität

Tabelle, Zeile, Spalte

Den Kern eines jeden Systems zur Tabellenkalkulation bildet die Definition von Rechenformularen. Ein solches Formular wird als eine **Tabelle,** bestehend aus **Zeilen** und **Spalten,** definiert, die vom Benutzer beliebig formatiert, beschriftet und mit Rechenoperationen versehen werden kann.

Beispiel 8.1

Dieses Beispiel zeigt die Berechnung der Summe von Umsätzen, die in verschiedenen Ländern mit unterschiedlichen Währungen erzielt wurden:

	A	B	C	D
1	Umsatz	Währung	Wechselkurs	Umsatz in Euro
2	20 000	DM	2,00	=A2/C2
3	40 000	FF	8,00	=A3/C3
4	100 000	BF	40,00	=A4/C4
5		Gesamtumsatz		=Summe(D2:D4)

Tabellenzelle

konstanter Wert

Berechnungsformel

Die Tabelle besteht aus vier Spalten A, B, C, und D und fünf Zeilen, die von 1 bis 5 durchnummeriert sind. Durch die Aufteilung in Spalten und Zeilen entstehen **Tabellenzellen**, in denen Werte eingetragen werden können. Die Zellen der ersten drei Spalten enthalten **konstante Werte**, entweder Zahlen, die für Berechnungen benötigt werden, oder Texte, die nur zur besseren Lesbarkeit der Kalkulation beitragen. Die Tabellenzellen der Spalte D ab der 2. Zeile enthalten nicht direkt Werte, sondern **Berechnungsformeln**, die indirekt den Inhalt dieser Zellen bestimmen. Der Wert für die Zelle D5 ist sogar doppelt indirekt, da er zunächst von den Zellen D2 bis D4 abhängt, die wiederum von den Zellen in den Spalten A und C abhängig sind. Die Berechnungsformel kann direkt in eine Zelle eingegeben werden; nach Abschluss der Eingabe erfolgt sofort die Berechnung, und das Ergebnis wird in derselben Zelle angezeigt. Zum Bearbeiten der Formel schaltet der Systembenutzer gewöhnlicherweise in einen anderen Modus um. Bei Excel kann die Formel in einer speziellen Bearbeitungszeile verändert werden

Analogien zu Programmiersprachen

Jede Zelle besitzt einen Namen, der in standardisierter Weise vorgegeben ist (Spaltenbuchstabe und Zeilennummer). Jede Zelle besitzt ebenso einen Wert. Insoweit entsprechen Zellen den Variablen einer Programmiersprache. Jeder Zelle kann aber zusätzlich eine Berechnungsformel zugeordnet werden. Andere Berechnungsformeln können darauf wieder Bezug nehmen, wodurch eine Verschachtelung von funktionalen Ausdrücken im Sinne einer einfachen Programmiersprache entsteht. Ferner stellt die Formulierung D2:D4 eine Realisierung der Zählschleife dar (von D2 bis D4). Zusätzlich können den Zellen Layout-Eigenschaften zugeordnet werden. Dadurch ist es möglich, einfachen Berechnungsprogrammen eine sehr ansprechende Benutzeroberfläche zu geben.

Tabellenkalkulation verglichen mit herkömmlicher Programmierung

Für die Berechnungsformeln steht bei den marktgängigen Systemen eine große Fülle von Funktionen zur Verfügung: Grundrechenarten, Zeit-, Finanz-, technische, statistische Funktionen etc. Darüber hinaus bieten gute Systeme auch Möglichkeiten, eigene Funktionen zu definieren (durch Anbindung von sog. Makro- oder Programmiersprachen). Die Argumente der Funktionen können sich auf andere Einzelzellen oder Zellbereiche beziehen. Im Beispiel 8.1 bezieht sich die Formel D5 auf den Bereich D2, D3, D4.

Vergleicht man diese Kernfunktionalität der Tabellenkalkulation mit den Möglichkeiten, die ein Taschenrechner oder ein klassisches Programm (mit C, PASCAL oder ähnlichen Programmiersprachen geschrieben) bietet, so stechen bei der Tabellenkalkulation zwei Merkmale besonders hervor:

- Eine Tabelle ist eine **sehr einfach zu bedienende Benutzeroberfläche**, die sofort ohne Programmierung vom System dem Benutzer zur Verfügung gestellt wird, und gleichzeitig auch eine Dokumentation der Berechnung bietet, die sich leicht ausdrucken lässt. Ein Taschenrechner bietet für Bedienung und Drucken nur sehr eingeschränkte Möglichkeiten; bei der Verwendung einer Programmiersprache müssen die Benutzeroberfläche und das Drucken individuell implementiert werden.
- **Berechnungen werden inkrementell durchgeführt**, d.h., verändert der Systembenutzer einen Input-Wert einer Formel, werden alle direkt oder indirekt abhängigen Formeln ebenfalls neu (und sofort) berechnet. Zwischenwerte, die nicht betroffen sind, werden nicht notwendig neu kalkuliert. Möchte der Benutzer, z.B. im Rahmen einer Umsatzplanung, wissen, wie sich eine Umsatzsteigerung von 10 % in Belgien (mit BF als Währung) auf den Gesamtumsatz auswirkt, ist nur eine Änderung des Umsatzes für Belgien notwendig. Andere Euro-Umsätze müssen nicht neu berechnet werden, und die Wirkung auf den Gesamtumsatz ist sofort sichtbar. Ein Taschenrechner oder ein klassisches Programm führt eine komplette Neuberechnung mit eventueller Neueingabe aller Werte durch.

8.2.2 Geschäftsgrafiken

Diagrammtyp

Auf Rechenformularen entstehen oftmals lange Zahlenkolonnen, die nur mühsam lesbar sind. Besser verständlich sind Geschäftsgrafiken in Form von **Balken-**, **Säulen-**, **Torten-**, **Netz-** oder **Liniendiagrammen**. Deshalb bieten alle TK-Systeme Generatoren für Geschäftsgrafiken an. Das Grundprinzip ist die matrixförmige Anordnung der Daten für die Geschäftsgrafik.

Beispiel 8.2

Die Umsätze aus Beispiel 8.1 seien in den drei Monaten Januar, Februar, März entstanden. Die Tabelle zeigt die Umsatzverteilung (in Euro) an:

	A	B	C	D
1		Januar	Februar	März
2	Deutschland	5 000	3 000	2 000
3	Frankreich	1 000	3 000	1 000
4	Belgien	500	500	1 500

Die Daten der Matrix bzw. Kreuztabelle können wir z.B. als ein mehrfarbiges Säulendiagramm wie in Bild 8.1 darstellen: Auf der *x*-Achse werden die Spaltenbeschriftungen aufgetragen. Die Zeilennamen können mit verschiedenen Farben oder Grau-

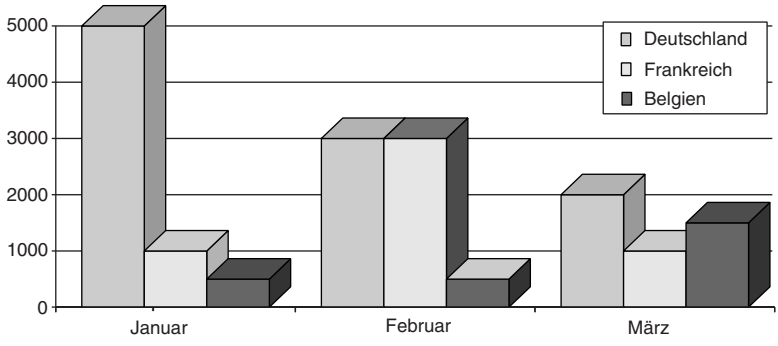


Bild 8.1
Ein Säulendiagramm für Matrixdaten

schattierungen dargestellt werden, die in einer *Legende* zugeordnet werden. Die *y*-Achse liefert eine Maßskala für die Werte der Matrixzellen. Üblicherweise bieten TK-Systeme mehrere Varianten für jeden Diagrammtyp an – für Säulendiagramme beispielsweise die Optionen „Säulen nebeneinander", „Säulen übereinander", „2-dimensionale Ansicht", „3-dimensionale Ansicht", „mit/ohne Hilfslinien" usw. Bei anderen Diagrammtypen (Balken-, Liniendiagramme etc.) erfolgt der eben skizzierte Übergang von der Matrix zur grafischen Darstellung nach dem gleichen Prinzip.

8.2.3 Kreuztabellen

Eine Matrix wird auch gerne als eine **Kreuztabelle** bezeichnet, da die jeweiligen Zahlenwerte (hier: die Umsätze) dort eingetragen werden, wo sich die zugehörige Zeile und Spalte der Merkmale (hier: Land und Monat) kreuzen. In Excel wird eine Kreuztabelle als Pivot-Tabelle bezeichnet.

Beispiel 8.3

relationale Darstellung von Umsatzzahlen

Häufig sind die Daten, die Grundlage für eine Geschäftsgrafik sind, noch nicht als Kreuztabelle angeordnet, sondern als lange Zahlenkolonnen nach dem Muster von relationalen Datenbanken gegeben:

	A	B	C
1	Deutschland	Januar	5 000
2	Deutschland	Februar	3 000
3	Deutschland	März	2 000
4	Frankreich	Januar	1 000
5	Frankreich	Februar	3 000
6	Frankreich	März	1 000
7	Belgien	Januar	500
8	Belgien	Februar	500
9	Belgien	März	1 500

automatische Umwandlung in eine Kreuztabelle

In vielen TK-Systemen ist eine **automatische Umwandlung in eine Kreuztabelle** möglich. Das ist vor allem dann interessant, wenn relationale Tabellen nicht nur Daten für zwei Kriterien (in unserem Beispiel Land und Monat) enthalten, sondern für viele Kriterien, die flexibel angeordnet werden sollen. Für viele Aufgabenstellungen müs-

sen auf die Daten auch sog. Verdichtungsfunktionen angewandt werden, um zeilen- und/oder spaltenweise Berechnungen durchzuführen. Typische Verdichtungsfunktionen sind Summe, Durchschnitt, Minimum und Maximum.

Auswertung von Umsatzzahlen

Beispiel 8.4

Wenn wir die Umsatzsummen aus Beispiel 8.3 pro Land, pro Monat und die Gesamtsumme mit aufnehmen, erhalten wir folgende Kreuztabelle:

	A	B	C	D	E
1		Januar	Februar	März	Summe
2	Deutschland	5 000	3 000	2 000	10 000
3	Frankreich	1 000	3 000	1 000	5 000
4	Belgien	500	500	1 500	2 500
5	Summe	6 500	6 500	4 500	17 500

Online Analytical Processing

Die Bereitstellung von Analyse-Daten aus Datenbanken, effizientes Berechnen von Kennzahlen und deren Darstellung als Kreuztabellen wird auch **Online Analytical Processing** (**OLAP**) genannt. Häufig kommen zu diesem Zweck eigenständige OLAP-Systeme zum Einsatz, welche die Verbindung zwischen Datenbank und TK-System herstellen.

8.2.4 Optimierung

Manchmal kennt der Anwender das Ergebnis einer Funktion, nicht aber alle Eingabewerte für diese Funktion. Ferner lässt sich die richtige Umkehrfunktion in vielen Fällen nicht so leicht bestimmen, wenn die Eingabewerte einer Funktion nicht direkt eingegeben werden, sondern wiederum von anderen Funktionen abhängen. Anstatt manuell und langwierig Eingabewerte auszuprobieren, kann der Anwender die **Zielwertsuche** einsetzen und dadurch das Ausprobieren automatisieren.

Zielwertsuche

Solver

Beispiel 8.5

Im Beispiel 8.1 (Umsätze aus mehreren Ländern) ist der DM-Umsatz herauszufinden, mit dem ein Gesamtumsatz von 20 000 Euro erreichbar ist. Dazu können wir in Excel den Befehl Zielwertsuche aus dem Menü Extras verwenden. Die Zelle mit dem Zielwert ist D5 (Gesamtumsatz) und die Zelle mit dem zu verändernden Wert ist A2 (Umsatz in DM).

Für den Fall, dass das Ergebnis einer Funktion vorgegeben ist, aber mehrere Eingabewerte unbekannt sind, bietet Excel das Zusatzprogramm **Solver** an. Es erlaubt das Aufstellen und Lösen von mathematischen Optimierungsproblemen, wie sie im Rahmen des Operations Research (dt. Unternehmensforschung) entwickelt wurden. Ähnlich wie bei der Zielwertsuche kann ein Zielwert für eine sog. Zielfunktion vorgegeben oder eines seiner Minima bzw. Maxima berechnet werden. Zusätzlich können für die Eingabezellen und die Zielzelle Nebenbedingungen in der Form von Werteeinschränkungen formuliert werden. Ist das Problem prinzipiell lösbar, variiert der Solver die Werte der Eingabezellen so lange, bis eine (Näherungs-) Lösung ge-

funden wurde, andernfalls wird die Suche nach einer vom Benutzer vorgegebenen Zeit abgebrochen. Ein konkretes Beispiel für die Verwendung von Solver werden wir in Abschnitt 8.4.6 kennenlernen.

8.2.5 Programmieren und Aufzeichnen von Makros

Automatisierung von Bedienungsfolgen

Zwar ist die Anzahl von angebotenen Menübefehlen und Tabellenfunktionen in TK-Systemen sehr groß, jedoch bleiben zwangsläufig viele individuelle Wünsche des Benutzers unerfüllt. Diese individuellen Anforderungen lassen sich grob in drei Klassen einteilen:

- Manuelle Bedienungsfolgen, die typisch sind für eine Anwendung und sich ständig wiederholen, sollen zusammengefaßt werden. Zum Beispiel kann es vorkommen, dass der Nutzer immer wieder die gleichen Einstellungen an Diagrammen vornimmt oder Tabellen immer wieder nach dem gleichen Muster aufbaut. Ideal wäre es, diese Bedienungsfolgen aufzuzeichnen und als eigenständige Menübefehle in das Excel-Menü aufzunehmen, damit sie jederzeit wieder aufrufbar sind.

funktionale Erweiterungen

- Hat ein Benutzer eine Tabellenkalkulation aufgebaut, die er auch anderen Personen zugänglich machen möchte, die keine oder nur geringe Vorkenntnisse bez. der Handhabung von TK-Systemen haben, so wäre es sinnvoll, Bedienungsfolgen aufzeichnen und als neue Menübefehle zur Verfügung stellen zu können.
- Systeme zur Tabellenkalkulation bieten eine große Anzahl von vorgefertigten Tabellenfunktionen. Manchmal ist dies aber nicht genug, weil spezielle Anwendungen spezielle Analysefunktionen erfordern können. Hier möchte der Benutzer die Möglichkeit haben, seine eigenen Funktionen als Programme zu entwikkeln und einzusetzen.

Programmiererweiterung VisualBasic (VB)

Gerade in Excel sind diese Wünsche in großer Konsequenz erfüllt worden, indem dem Benutzer eine interne Systemschnittstelle offen gelegt wurde, auf die mit Hilfe von **VisualBasic** zugegriffen werden kann. Einfache Prozeduren (häufig auch als Makros bezeichnet) muss der Benutzer gar nicht Buchstabe für Buchstabe eintippen; er kann vielmehr seine Bedienungsfolgen vom System „mitschreiben“ und anzeigen lassen. Diese aufgezeichneten Programme bestehen im Wesentlichen aus Aufrufen von Funktionen der offen gelegten Schnittstelle, welche die Arbeit auf Tabellen- oder Diagrammblättern durchführen. Für diese Programme steht eine eigene Programmierumgebung zur Verfügung, um Änderungen vornehmen zu können. Des Weiteren kann der Benutzer einen Menü-Editor, einen Symbol-Editor und einen Dialog-Editor anwenden, um diese neuen Funktionen benutzerfreundlich aufrufen zu können. Letztendlich lassen sich mit der Programmierumgebung eigenständige Windows-Anwendungen erstellen, die nur noch intern die Excel-Funktionen ausnutzen.

8.2.6 Weitere Möglichkeiten

Abschließend seien noch zwei weitere nützliche Eigenschaften erwähnt, die sich in vielen TK-Systemen wiederfinden:

- Geschäftsgrafiken werden gerne um grafische Elemente ergänzt; deshalb sollte auch ein Editor für Zeichnungen zum Programm gehören.
- Die Daten, die im Rahmen einer Tabellenkalkulation ausgewertet werden sollen, wird man sich oft erst aus einer oder mehreren Datenbanken von verschiedenen Rechnern zusammenholen müssen. Für diesen Zweck braucht man Tools zur Formulierung von Datenbankanfragen und zur Rechnerkommunikation.

8.3 Beispiel: Auswertung von Experimentdaten

Zur Demonstration der Möglichkeiten von Excel werden wir Schritt für Schritt ein Anwendungsbeispiel erarbeiten. Die Aufgabe soll darin bestehen, ein Auswertungssystem für physikalische Experimente zu entwickeln. Das System soll so aufgebaut werden, dass es in Praktika zur Physik, Elektronik, o.ä. während einer Ingenieurausbildung nützlich ist. Zunächst wollen wir die Aufgabenstellung eines Praktikums beschreiben.

Motivation des Beispiels

Eine Hauptaufgabe im Praktikum ist das Nachvollziehen von experimentellen Überprüfungen physikalischer Gesetze. Zu diesem Zweck bauen die Teilnehmer eines Praktikums Versuche auf, mit denen sie Messungen durchführen. Die Struktur des Versuchsaufbaus wird sehr stark vom Untersuchungsgegenstand abhängig sein, es liegt aber immer das folgende Muster zugrunde: Zum Versuch gehört eine Reihe von Geräten, an denen man physikalische Größen einstellen bzw. ablesen kann. Die Geräte können ganz einfacher Natur – wie bei einem Versuch zu idealen Gasen – oder komplexe Apparaturen sein – zum Beispiel in der Kernphysik. Eine der physikalischen Größen wird man variieren, um einen Effekt auf eine andere physikalische Größe zu messen, andere Größen wird man konstant halten.

Zeichnen von Kurven anhand von Messpunkten

Die Ergebnisse der Versuchsmessungen lassen sich in einer zweispaltigen Tabelle aufschreiben. Die erste Spalte enthält die zu verändernden Werte, im Folgenden x-Werte genannt, und die zweite Spalte die gemessenen Werte, die y-Werte. Zur Auswertung wird der Experimentator diese Werte zunächst als Punkte in ein xy-Diagramm auf ein Blatt Papier einzeichnen. Anschließend wird er mit möglichst ruhiger Hand versuchen, durch die Punkte eine Kurve zu legen, die möglichst nahe an den Punkten liegt. Dabei entsteht ein Kurventyp (Hyperbel, Parabel, Exponentialkurve, Sinuskurve etc.), der sich im nächsten Arbeitsschritt mit dem Kurventyp des zu prüfenden physikalischen Gesetzes vergleichen lässt. Der theoretisch richtige Kurventyp ist leicht zu bestimmen; es muss nur die Formel des Gesetzes nach der gemessenen physikalischen Größe aufgelöst werden. Dadurch wird eine Funktion abgeleitet, der ein bestimmter Kurventyp zugeordnet ist.

Ist die experimentell gewonnene Kurve mit dem theoretisch zu erwartenden Kurventyp im Einklang, so ist das physikalische Gesetz bestätigt (in dem Sinne, dass es durch das Experiment jedenfalls nicht widerlegt wurde). Falls nicht, muss das Experiment überprüft werden, um herauszufinden, ob sich Fehler in den Versuchsaufbau oder in die Messungen eingeschlichen haben oder ob gar die Theorie falsch war.

Fehlerabschätzung

Diese Vorgehensweise kann der Experimentator durch eine Fehlerabschätzung ergänzen, um Messungenauigkeiten in den Griff zu bekommen. Abhängig von der physikalischen Gesetzesformel lassen sich durch mathematische Ableitungen Fehlerfunktionen für die Berechnung der Fehlertoleranzen von y-Werten bestimmen. Die Fehlertoleranzen sind Maßzahlen, die angeben, wie weit Kurven von den y-Werten höchstens abweichen dürfen. Verläuft die eingezeichnete Kurve außerhalb des Toleranzbereiches, ist nicht eine Messungenauigkeit die Ursache, sondern entweder eine völlig falsche Messung, ein falscher Versuchsaufbau oder gar eine falsche Theorie. Falls solche großen Abweichungen nur vereinzelt vorkommen, wird der Experimentator die entsprechenden Einzelmessungen aus der Messreihe herausnehmen und die restliche Reihe weiterhin als gültig ansehen. Ansonsten ist der Versuchsaufbau zu überdenken.

Die genaue Berechnung der Fehlertoleranzen per Hand und deren Einzeichnung in das *xy*-Diagramm sind allerdings eine sehr mühsame Sache, deshalb lässt man diese Berechnungen in der Praxis der Praktikumsübungen meist einfach weg und begnügt sich mit einer groben „visuellen" Abschätzung.

Ähnlich grob ist das Einzeichnen der Kurve durch die Messpunkte: Das Gesetz, das überprüft werden soll, ist ja bekannt! Die Versuchung ist daher natürlich sehr groß, nicht den am besten passenden Kurventyp einzuzeichnen, sondern denjenigen, den man braucht.

Dieser „grobe" Ansatz soll im Folgenden verbessert werden, indem wir ein Programm entwickeln, das die Rechen- und Zeichenarbeit unterstützt. Wir werden dabei sehen, dass der Ansatz der Tabellenkalkulation das Programmieren wesentlich vereinfacht. Als Beispiel – zur Demonstration des Prinzips – verwenden wir das Boyle-Mariottesche Gesetz (siehe /8.1/) für ideale Gase. Es besagt, dass bei gleichbleibender Temperatur das Produkt aus dem Druck p und dem Volumen V einer gegebenen Gasmenge konstant bleibt (d.h. mit einer Konstanten k identisch ist). Die zugehörige Formel lautet:

$$p\ V = k$$

Als Versuchsaufbau verwenden wir ein Glasrohr, in dem eine Stahlkugel ein bestimmtes Luftvolumen absperrt. Mit einem Manometer wird der Druck im Inneren des Rohres gemessen. Mittels einer Pumpe wird der Luftdruck auf einer Seite der Kugel im Rohr verändert. Die Kugel verschiebt sich, und zwar so lange, bis wieder Druckgleichgewicht im Glasrohr hergestellt ist. Das abgesperrte Luftvolumen kann dann leicht an einer Messskala abgelesen werden. Die Temperatur des Gases muss während des Experimentes konstant gehalten werden.

8.4 Problemlösung mit den generischen Excel-Funktionen

Zunächst wollen wir betrachten, inwieweit wir die Messdatenauswertung ohne Programmieren, nur mit den generischen Funktionen von Excel unterstützen können. Das Vorgehen lässt sich in sieben Schritte einteilen, die wir in den folgenden Unterabschnitten einzeln betrachten werden:

1. Eingeben der Messwerte
2. Zeichnen der Messpunkte
3. Berechnen des theoretischen Werteverlaufs mit vorgegebenen Parametern
4. Zeichnen der theoretischen Werteverlaufskurve
5. Interaktives Anpassen der Parameter der theoretischen Werteverlaufskurve
6. Bestimmung der optimalen Parameterwerte
7. Fehleranalyse

8.4.1 Eingeben der Messwerte

Es bietet sich an, für jedes Experiment eine Arbeitsmappe anzulegen, die ein Tabellenblatt mit den Messdaten enthält. In unserem Beispiel seien die in Bild 8.2 angegebenen Messwerte bekannt und in einer Arbeitsmappe GAS.XLS gespeichert.

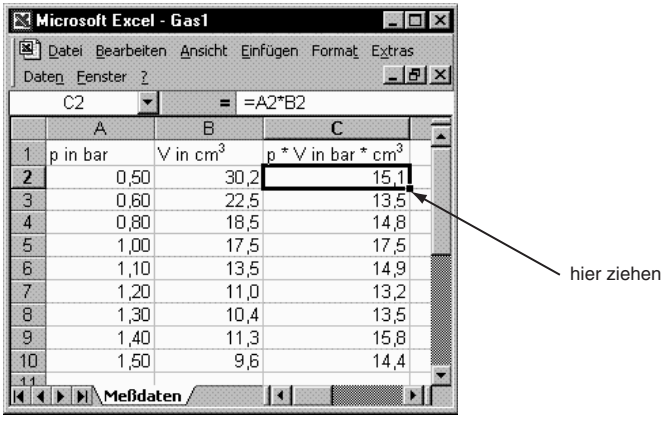


Bild 8.2 Messdaten-Tabelle Beispieldaten

In der ersten Spalte (*x*-Werte) ist der Druck angegeben, der bei unserem Versuch zum Boyle-Mariotteschen Gesetz eingestellt wurde. Die zweite Spalte (*y*-Werte) enthält die zugehörigen gemessenen Volumenwerte. In der dritten Spalte nutzen wir gleich die Möglichkeiten der Tabellenkalkulation aus und berechnen das Produkt der beiden Größen. Die Formel müssen wir nur für das erste Wertepaar angeben; für alle anderen Wertepaare können wir die Funktion *AutoAusfüllen* von Excel ausnutzen. Sie dient dazu, in einer Tabelle eine Datenreihe automatisch zu erstellen.

AutoAusfüllen

Der Aufruf erfolgt entweder über das Bearbeiten-Menü oder aber durch Ziehen des Ausfüllkästchens (erkenntlich am kleinen schwarzen Quadrat des fettgedruckten Auswahl-Rahmens) des markierten Bereichs mit der Maus über den Bereich, der ausgefüllt werden soll. Die Funktion *AutoAusfüllen* ist z.B. in der Lage, eine Zahlenreihe, von der die ersten zwei Werte (etwa 2 und 4) markiert wurden, richtig fortzuführen (in unserem Beispiel 6, 8, 10, wenn drei weitere Zellen ausgefüllt werden sollen). Eine ausführliche Erklärung für diese Funktion ist zum Beispiel in der Excel-Online-Hilfe unter dem Stichwort *AutoAusfüllen* zu finden. Es ist eine äußerst benutzerfreundliche Funktion, die dem Anwender viele wiederholende Eingaben abnimmt!

In unserem Falle markieren wir die Zelle C2, welche das Produkt für das erste Wertepaar enthält, und ziehen das Ausfüllkästchen mit der Maus über die nächsten 8 Zeilen in der gleichen Spalte. Dies bewirkt, dass die Formel in die darunter liegenden Zellen kopiert wird, wobei die Zeilennummern der Zellbezüge fortlaufend erhöht werden. Als Ergebnis erhalten wir die Produkte für alle Messwerte. Wie wir leicht sehen können, ist das Produkt von *p* und *V* recht starken Schwankungen unterworfen.

8.4.2 Zeichnen der Messpunkte

Aus den Messwerten kann eine Kurve generiert werden, indem wir ein Diagrammblatt einfügen. Folgende Schritte sind im Einzelnen durchzuführen, um das Diagramm von Bild 8.3 zu erzeugen (die Zuordnung von Excel-Befehlen zu Menüs schreiben wir in der Form *Menü* | *Untermenü* | *Befehl*, wenn Untermenüs vorhanden sind, und in der Form *Menü* | *Befehl*, wenn nur eine zweistufige Gliederung vorliegt):

1. Zunächst fügen wir ein neues Diagrammblatt mit dem Befehl *Einfügen* | *Diagramm* in die Arbeitsmappe ein. Zur Gestaltung des neuen Diagramms bekommen wir nun in den nächsten Schritten die Unterstützung durch den Diagrammassistenten.
2. Als Diagrammtyp selektieren wir *Punkt (XY)*, um unsere Daten in ein *xy*-Diagramm zu zeichnen. Durch Drücken der Schaltfläche *Weiter >* gelangen wir zum nächsten Schritt.
3. Wir wählen nun die Zellen aus, welche die Daten für das Diagramm enthalten. In unserem Falle sind dies die Zellen A2 bis B10 im Tabellenblatt Messdaten. Danach können wir auf die Schaltfläche *Weiter >* klicken, um zum nächsten Dialogblatt zu gelangen.
4. Die voreingestellten Gestaltungsoptionen sind für uns bereits brauchbar. Lediglich die Legendenanzeige schalten wir aus, da wir nur eine Datenreihe zeichnen. Zusätzlich schalten wir das Hauptgitternetz der *x*-Achse ein.
5. Im letzten Schritt des Assistenten bestimmen wir, das Diagramm in ein neues Blatt einzufügen. Durch Klicken auf *Fertig stellen* erzeugen wir das Diagramm.

Der Inhalt des Diagrammblattes kann nun durch kosmetische Operationen noch verschönert werden, die Arbeitsmappe besser organisiert werden:

- Durch Doppelklicken auf die Zeichnungsfläche können wir diese formatieren: Sinnvoll ist in unserem Beispiel die Hintergrundfarbe Weiß.
- Um das Diagramm größtmöglich im Fenster erscheinen zu lassen, rufen wir noch den Menübefehl *Ansicht* | *Fenstergröße angepasst* auf.
- Eine Umbenennung des Diagrammblattes auf den Namen *p-V*-Diagramm erreichen wir, indem wir das Kontextmenü von Blättern aufrufen und den Menübefehl *Umbenennen* verwenden. Das Kontextmenü erscheint, wenn wir den Mauspfeil über das Register des Blattes positionieren und die rechte Maustaste klicken.
- Letztendlich ziehen wir das Blattregister von *p-V*-Diagramm bei gedrückter linker Maustaste hinter Messdaten.

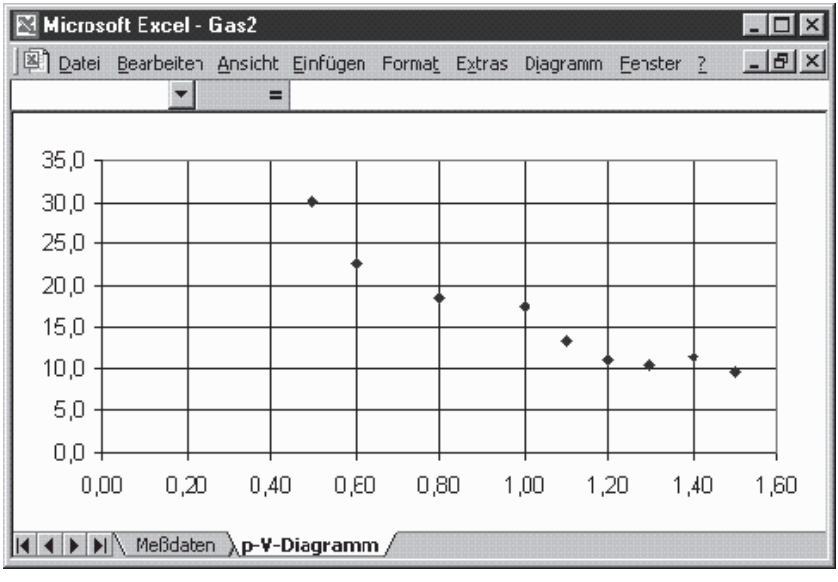


Bild 8.3
Aus Messdaten generiertes Messpunktdiagramm

Erzeugen eines Diagramms

8.4.3 Berechnen des theoretischen Werteverlaufs

Das *p*-*V*-Diagramm könnten wir nun ausdrucken und eine Kurve einzeichnen. Von der Theorie des Versuches her wissen wir, dass die Kurve eine Hyperbel mit Funktionsgleichung $V = k/p$ sein „muss“. Doch passt eine Hyperbel wirklich zu den Messpunkten? Passt nicht auch eine Gerade?

Zeichnen der theoretischen Kurve

Statt uns nun auf unser Augenmaß zu verlassen, wollen wir lieber die Möglichkeiten der Tabellenkalkulation zur genauen Klärung einsetzen. Hierfür legen wir uns ein neues Tabellenblatt an, das den theoretischen Werteverlauf enthält. Diesem Tabellenblatt geben wir den Namen Theorie. Nach der Berechnung des theoretischen Werteverlaufs werden wir ihn benutzen, um eine Kurve in unser *p*-*V*-Diagramm einzuzeichnen. Die theoretische Berechnung wird dadurch erschwert, dass üblicherweise die Konstante *k* dem Experimentator nicht bekannt ist. Wir müssen also die Konstante variieren, um eine möglichst passende Kurve zu erzeugen. Wie wir gleich kennenlernen werden, lässt sich dies optisch elegant in eine Tabellenkalkulation einbauen.

Auf dem neuen Tabellenblatt wählen wir eine Zelle aus, welche den Wert für die Konstante enthalten soll (die Zelle B2 in Bild 8.4) und tragen einen Schätzwert ein, den wir uns rein willkürlich setzen können. Für Bild 8.6 wählten wir den Wert 14,5 aufgrund der Kalkulationen der Spalte C auf dem Tabellenblatt Messdaten. Dort liegen die Werte zwischen 13,2 und 17,5.

Anschließend suchen wir uns eine Spalte, in die wir den zugehörigen theoretischen Werteverlauf eintragen wollen. In dieser Spalte berechnen wir für jeden *p*-Wert den theoretischen *V*-Wert. Dies kann mit wenigen Handgriffen auf dem Tabellenblatt erreicht werden:

Zunächst ist eine Berechnung für den ersten *p*-Wert auszuführen, was nach dem Muster von Bild 8.4 geschehen kann: Die Bearbeitungsleiste zeigt die Formel an, die für die Zelle C2 angegeben wurde. Die Formel hat die Struktur einer Hyperbelfunktion mit Bezug auf die Zelle für die Konstante *k* und auf die Zelle mit dem ersten *x*-Wert im Tabellenblatt Meßdaten. Für die Zelle selbst berechnet Excel sofort den richtigen Wert und zeigt ihn an.

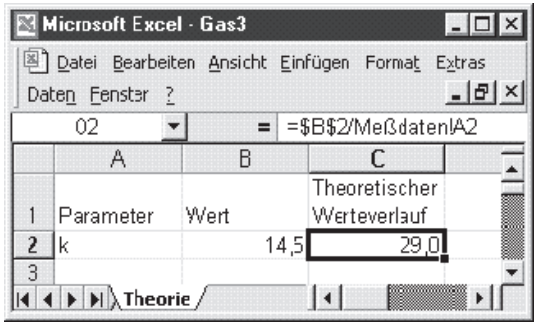
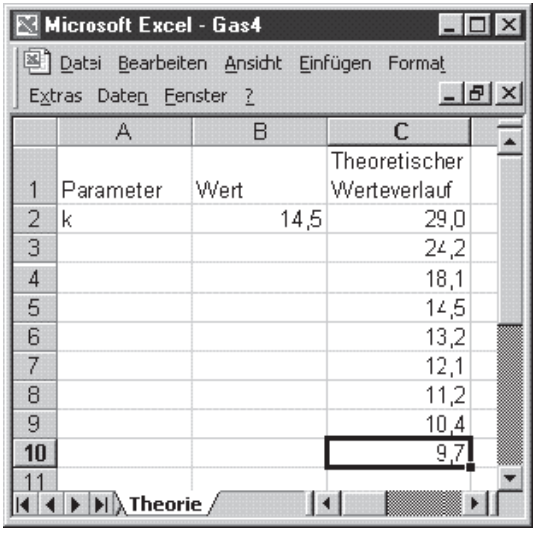


Bild 8.4 Berechnung des theoretischen V-Wertes

relativer Zellbezug

Der Verweis auf die eingestellten Gasdrücke ist als ein **relativer Zellbezug** angegeben, der Verweis auf *k* als absoluter Zellbezug (erkenntlich am Zeichen „$“) vor Spaltenbuchstabe und Zeilennummer. Die unterschiedliche Adressierung benötigen wir, um *AutoAusfüllen* richtig anwenden zu können. Das Ziehen des Ausfüllkästchens über einen Zellbereich bewirkt wieder, dass die Formel in alle Zellen des Bereichs kopiert wird, wobei der Zellbezug auf die Konstante des Gasgesetzes erhalten bleibt

(da es ein absoluter Bezug ist), die Zeilennummer des Zellbezugs für den Messwert aber fortlaufend erhöht wird (da der Bezug relativ angegeben ist). Als Ergebnis erhalten wir die berechneten Volumina (siehe Bild 8.5).



	A	B	C
1	Parameter	Wert	Theoretischer Werteverlauf
2	k	14,5	29,0
3			24,2
4			18,1
5			14,5
6			13,2
7			12,1
8			11,2
9			10,4
10			9,7
11			

Bild 8.5 Tabellenblatt mit Datenreihe nach Berechnung der theoretischen Werte durch AutoAusfüllen

Bei anderen physikalischen Formeln, die evtl. mehrere Funktionsparameter besitzen, gehen wir entsprechend vor und tragen auch die anderen Parameterwerte in die Spalte B ein.

8.4.4 Zeichnen der theoretischen Werteverlaufskurve

Als nächstes ist die Datenreihe der berechneten Werte als eine Linie in unserer Zeichnung auf dem Blatt *p-V*-Diagramm darzustellen. Dies lässt sich durch die folgenden beiden Schritte bewerkstelligen (vgl. Bild 8.6):

- Wenn das Diagrammblatt aktiv ist, können wir im Menü *Diagramm* den Befehl *Datenquelle* auswählen und auf dem nachfolgenden Dialogfeld den Bereich Theorie!C2:C10 angeben.
- Es erscheint eine neue Punktekette, die nun zu formatieren ist. Dies geschieht am einfachsten durch einen Doppelklick auf einen der Punkte in der Kette. Als Folge wird die Punktekette selektiert und ein Dialogfeld erscheint, das eine Reihe von Registern enthält. Auf dem Registerblatt *Muster* können wir die für uns beste Präsentationsform auswählen: eine geglättete Linie, deren Punkte nicht besonders hervorgehoben werden sollen, was wir durch Auswählen der Option „*ohne Markierung*“ erreichen können.

8.4.5 Interaktives Anpassen der Parameter der theoretischen Werteverlaufskurve

Das Ziel sollte sein, die Kurve so zu zeichnen, dass sie möglichst nahe an den Messpunkten vorbeiläuft. Die Kurve, die wir in Bild 8.6 erhalten haben, liegt nach Augenmaß schon recht gut. Können wir eine noch bessere Lösung finden? Zur Beantwortung dieser Frage kann man verschiedene Wege gehen. Eine einfach zu realisierende Möglichkeit, die zudem zur Demonstration der Excel-Vorzüge sehr lehrreich ist, besteht darin, die Parameterwerte interaktiv zu verändern und die Auswirkung auf die Kurve im Diagramm zu beobachten. Dieses Angleichen können wir sofort in

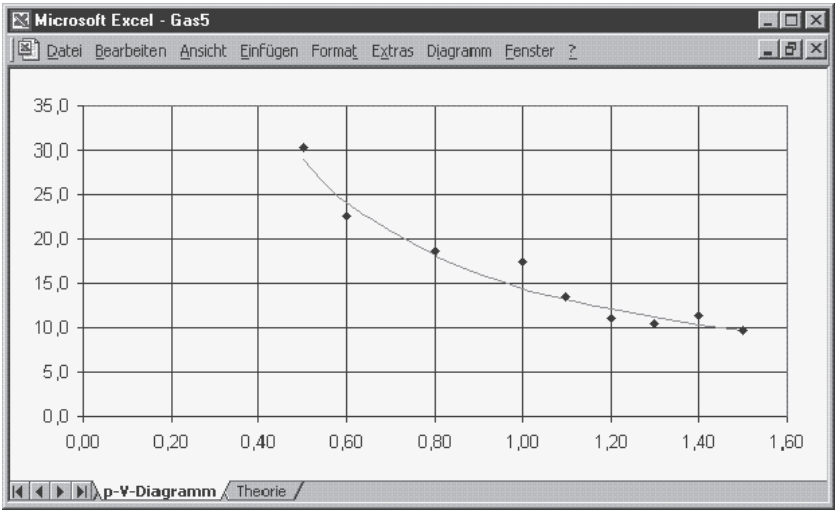


Bild 8.6 p-V-Diagramm nach Einfügen der berechneten Kurve

Angriff nehmen, indem wir die Gesetzeskonstante auf dem Tabellenblatt *Theorie* verändern. Unser TK-System bemerkt sofort die Veränderung, erkennt die Verknüpfungen mit Formeln in anderen Zellen – in unserem Falle die Spalte, welche die berechneten Werte enthält – und kalkuliert die Formeln neu. Da das System sich außerdem gemerkt hat, dass die Kurve im Diagramm von einer Datenreihe auf dem Blatt *Theorie* abhängt, ist es in der Lage, auch die Präsentation im Diagramm an die veränderten Daten anzupassen.

Gestaltung der Benutzeroberfläche

Für den Benutzer besteht jedoch noch das Problem, dass er ständig zwischen dem Blatt *Theorie* und dem Blatt *p-V-Diagramm* hin- und herspringen muss, um die Auswirkung einer Parameteränderung zu betrachten. Zur Lösung dieses Problems können wir die Möglichkeiten von Excel zur **Gestaltung der Benutzeroberfläche** ins Spiel bringen. Um diese Möglichkeiten kennenzulernen, schalten wir zunächst die Symbolleiste *Formular* ein (z.B. durch Ankreuzen des Kontrollkästchens für *Dialog* im Dialogfeld, das nach Auswahl des Befehls *Ansicht | Symbolleisten...* erscheint). Dort sind diejenigen Dialogelemente als Schaltflächen aufgelistet, die sich in Tabellenblättern, Diagrammblättern oder eigenständigen Dialogblättern, die wir später noch kennenlernen werden, verwenden lassen.

Für unsere Bedürfnisse wollen wir einen „Regler“, eine sog. Bildlaufleiste einbauen, mit deren Hilfe wir die Gesetzeskonstante verändern können. Eine solche interaktive Parametrisierung ist vor allem dann sinnvoll, wenn die physikalische Formel nicht nur von einem Parameter abhängt (in unserem Beispiel die Konstante k), sondern von mehreren (z.B. in Funktionen vom Typ „$y = ax^2 + bx + c$“).

Einbau von Reglern

Für den Einbau von einem oder mehreren Reglern sind folgende Aktionen notwendig (vgl. Bild 8.7):

1. Wir drücken in der Dialog-Symbolleiste die Schaltfläche für die Bildlaufleiste.
2. Wir ziehen mit der Maus ein Rechteck auf der Diagrammfläche auf, um die Position des Reglers festzulegen.
3. Wir wollen den Regler auch beschriften, um eine Zuordnung zu den Funktionsparametern zu haben. Dies können wir nach Drücken der Schaltfläche für Bezeichnungsfelder durchführen. Wie bei den Bildlaufleisten können wir anschließend ein Rechteck zur Positionierung angeben und unseren jeweiligen Text einfügen.

4. Zur Feinpositionierung verändern wir die geometrische Lage der Dialogelemente, indem wir sie bei gedrückter STRG-Taste anklicken und verschieben.
5. Es fehlt noch die Verknüpfung des Reglers mit der Zelle, deren Wert verändert werden soll. Das ist nach Auswahl des Dialogelements durch Drücken der Schaltfläche für *Steuerelement-Eigenschaften* möglich. Es erscheint ein Dialogblatt, das auf der Karte *Steuerung* ein Eingabefeld für die Ausgabeverknüpfung enthält. In unserem Falle ist die Zelle für *k* anzugeben.
6. Auf derselben Registerkarte lassen sich auch die passende Schrittweite, minimaler und maximaler Wert einsetzen, allerdings müssen Minimum und Maximum ganze Zahlen zwischen 0 und 30 000 sein, insbesondere können damit die Grenzen nicht negativ oder Dezimalbrüche sein. In unserem Beispiel wäre es sinnvoll, mit Zehntel-Schritten die Kurve zu verändern. Dies können wir durch einen kleinen Trick leicht erreichen: Wir bauen nicht einen, sondern zwei Regler ein: einen zur groben Bestimmung des Kurvenverlaufs und einen zur Feinabstimmung. Beide Regler erhalten als maximale Weite den Wert 100. Anschließend verknüpfen wir die Regler nicht direkt mit der Gesetzeskonstanten (in der Zelle *Theorie!B2*), sondern mit anderen Zellen, z.B. mit den direkt darunter liegenden Zellen (*Theorie!B3* und *Theorie!B4*). Unser Funktionsparameter in B2 berechnet sich nun aus der Formel „=B3+B4/10“.

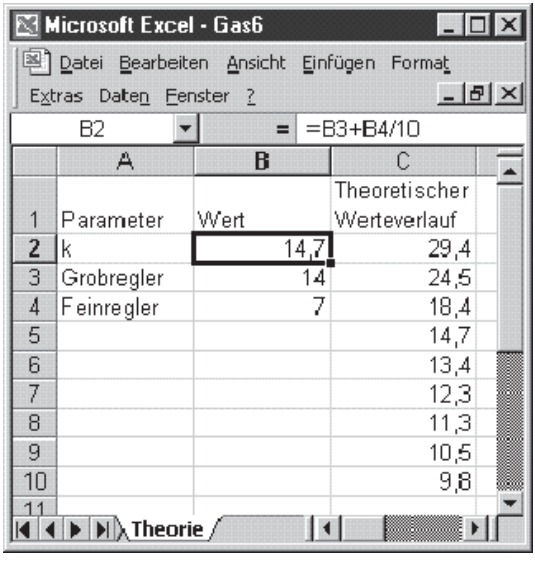


Bild 8.7 Indirekte Berechnung der Konstanten k

Nach diesen Vorbereitungen können wir nun die Regler einsetzen, um die Kurve näher an die Punkte heranzubringen. Dies geschieht am besten durch Klicken auf die Pfeiltasten, die am Regler angebracht sind. Nach jedem Klicken sehen wir sofort das Resultat: ob die Kurve besser oder schlechter „liegt“. In unserem Falle erhalten wir für $k = 14{,}7$ eine Lösung, die rein optisch gut passt. Das Ergebnis ist in Bild 8.8 zu sehen.

8.4.6 Bestimmung der optimalen Parameterwerte

Mit der Lösung des letzten Abschnittes wird man in vielen Fällen noch nicht zufrieden sein, da man zwar optisch eine Annäherung gefunden, aber eigentlich nicht weiß, wie gut sie ist, zumal durch die Bildschirmauflösung und durch den Bildmaßstab technische Grenzen gesetzt sind. Um an dieser Stelle weiterzukommen, müssen wir

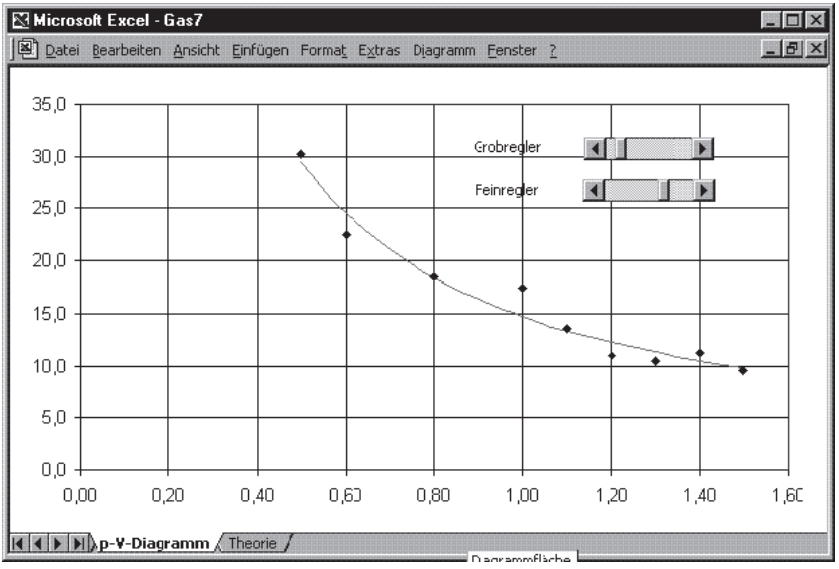


Bild 8.8 p-V-Diagramm nach Einbau von Bildlaufleisten und interaktiver Annäherung

uns die Mathematik zunutze machen, die für unsere Aufgabenstellung existiert. Mathematisch betrachtet, ist das Angleichen von Kurven an Messpunkte eine Regressionsanalyse zweiten Art und wird in der Statistik (siehe z.B. /8.2/) untersucht.

Prinzip der kleinsten Abstands-quadrate

Nach dem Gaußschen Prinzip der kleinsten vertikalen Abstandsquadrate erhält man die am besten angepaßte Kurve dann, wenn die Summe der vertikalen Abstandsquadrate aller Meßpunkte von der Kurve minimal ist. Ist z.B. y_i der berechnete Wert und m_i der gemessene Wert für die i-te Messung, dann ist das vertikale Abstandsquadrat $(y_i - m_i)^2$. Die Summe

$$\sum_{i=1}^{n}(y_i - m_i)^2$$

ist ein Maß für die Genauigkeit unserer Parametrisierung und wird im Folgenden von uns als Abweichungsgrad bezeichnet. y_i ist im allgemeinen Fall das Ergebnis einer Funktion $f(a, b, c, ..., x)$, d.h. einer Funktion, die nicht nur von x-Werten abhängig ist, sondern auch von Funktionsparametern a, b, c, … Beim Gasgesetz gibt es nur einen Funktionsparameter, nämlich k.

Anwendung einer Excel-Tabellen-funktion

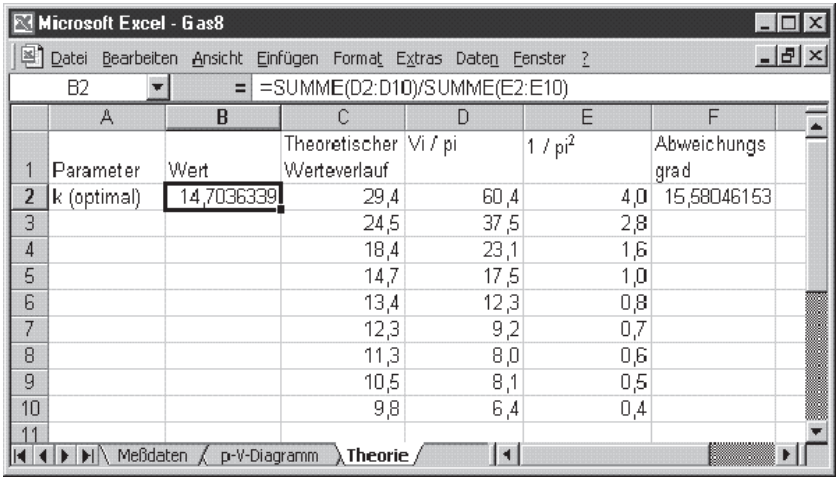
Die Summe der kleinsten Abweichungsquadrate wird durch die Excel-Tabellenfunktion *SUMMEXMY2* berechnet. Die Berechnung können wir auf dem Tabellenblatt Theorie mit unterbringen (siehe Bild 8.9). Der Zelle F2 ist die Formel

= *SUMMEXMY2*(C2:C10;Meßdaten!B2:B10)

zugeordnet. Dadurch haben wir für das interaktive Anpassen eine Möglichkeit, die Qualität einer gezeichneten Kurve zu messen.

Die optimale Lösung ist bestimmt durch den minimalen Abweichungsgrad. Das bedeutet für das Gasgesetz, dass das Minimum der Funktion

$$g(k) := \sum_{i=1}^{n}(k / p_i - V_i)^2$$


Microsoft Excel - Gas8

Datei Bearbeiten Ansicht Einfügen Format Extras Daten Fenster ?

B2 = =SUMME(D2:D10)/SUMME(E2:E10)

	A	B	C	D	E	F
1	Parameter	Wert	Theoretischer Werteverlauf	Vi / pi	1 / pi²	Abweichungsgrad
2	k (optimal)	14,7036339	29,4	60,4	4,0	15,58046153
3			24,5	37,5	2,8	
4			18,4	23,1	1,6	
5			14,7	17,5	1,0	
6			13,4	12,3	0,8	
7			12,3	9,2	0,7	
8			11,3	8,0	0,6	
9			10,5	8,1	0,5	
10			9,8	6,4	0,4	
11						

Meßdaten / p-V-Diagramm / Theorie /

Bild 8.9 Berechnung des optimalem Wertes für k

zu berechnen ist, wenn wir mit p_i bzw. V_i den Gasdruck bzw. das Gasvolumen der i-ten Messung bezeichnen. Die Berechnung des Minimums kann mit zwei verschiedenen Strategien erfolgen: durch die exakte Berechnung der Funktionsparameterwerte mittels Differentialrechnung oder durch den Einsatz eines numerischen Näherungsverfahrens.

Bei der ersten Strategie ist die erste Ableitung der Funktion $g(k)$ nach k gleich Null zu setzen. Daraus erhalten wir für die Konstante k des Gasgesetzes die folgende Lösung:

$$k = \frac{\sum_{i=1}^{n} \frac{V_i}{p_i}}{\sum_{i=1}^{n} \frac{1}{p_i^2}}$$

Zur Berechnung in Excel benutzen wir zwei freie Spalten auf dem Tabellenblatt *Theorie*, in die wir die Werte für V_i/p_i und für $1/p_i^2$ eintragen. Aus diesen beiden Reihen können wir k berechnen, wie Bild 8.9 zeigt.

Wie wir sehen, war unsere interaktive Abschätzung mit $k = 14{,}7$ gar nicht so schlecht! Zumal wir ja zusätzlich noch die Möglichkeit haben, den Abweichungsgrad zu berechnen und beim Verändern der Parameter zu überprüfen, ob eine Verbesserung eingetreten ist oder nicht.

Das Problem bei der exakten Parameterberechnung besteht darin, dass für viele physikalische Formeln die Herleitungen für die Funktionsparameter sehr kompliziert werden. Zum Beispiel sind bei der allgemeinen Hyperbel-Funktion $y = a/(x + b) + c$ die partiellen Ableitungsfunktionen für drei Parameter zu berechnen und gleich Null zu setzen. Es ergibt sich ein System von drei Gleichungen mit drei Unbekannten, das es zu lösen gilt.

symbolische Formelmanipulation

Durch Anbindung eines Systems zur **symbolischen Formelmanipulation** könnten wir versuchen, diese Berechnungen in den Griff zu bekommen. Ein solches System ist in der Lage, symbolische Berechnungen, insbesondere Ableitungen, automatisch durchzuführen. Ein solches System ist aber nicht direkt an Excel angebunden, und daher soll dieser Ansatz hier auch nicht weiter verfolgt werden.

Zusatzprogramm Solver

Bestimmung von Näherungslösungen mit dem Solver

Wir wenden uns deshalb der zweiten Strategie zu – dem Einsatz eines numerischen Näherungsverfahrens. Hierfür gibt es eine Unterstützung durch Excel in Form des **Zusatzprogramms Solver.** Dieses Programm bietet numerische Näherungsverfahren zur Bestimmung von Nullstellen, Minima und Maxima von Funktionen an.

Für unser Beispiel können wir den Solver einsetzen, um den Abweichungsgrad, das Minimum der Summe der Abstandsquadrate, zu bestimmen. Zur Vorbereitung müssen wir zunächst den Solver über den Add-In-Manager aktivieren. Dies geschieht folgendermaßen:

1. Auswahl des Menübefehls *Extras* | *Add-In-Manager*.
2. Ankreuzen des Kontrollkästchens für den *Solver* in der Liste der vorhandenen Add-Ins.

Nach der Aktivierung steht uns im Menü „*Extras*“ ein neuer Menübefehl *Solver...* zur Verfügung. Der Aufruf bewirkt zunächst, dass ein Dialogfeld mit Titel *Solver-Parameter* erscheint (Bild 8.10).

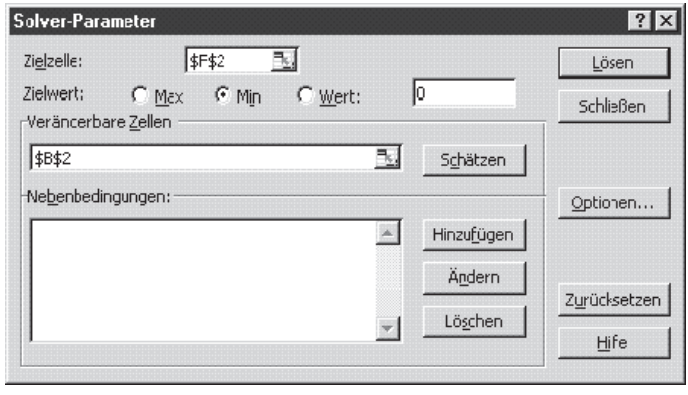


Bild 8.10 Einstellung der Solver-Parameter zur Berechnung der Summe der Abweichungsquadrate

Im Eingabefeld „Zielzelle“ geben wir die Zelle an, die den Abweichungsgrad enthält. Als Zielwert stellen wir „Minimum“ ein, da wir das Minimum des Abweichungsgrades berechnen wollen. Im Eingabefeld „Veränderbare Zellen“ tragen wir die Zelle ein, in welcher die Konstante *k* steht. Prinzipiell können wir uns auf einen ganzen Bereich von veränderbaren Zellen beziehen; z.B. für die drei Parameter *a*, *b*, *c* der allgemeinen Hyperbelfunktion.
Nach Einstellung der Parameter brauchen wir nur noch die Schaltfläche für „Lösen“ zu drücken, und nach einigen Sekunden erhalten wir vom Solver eine Antwort zurück. In unserem Falle ist das Problem für den Solver lösbar und liefert genau denselben Wert, der auch bei der exakten Berechnung der Funktionsparameter gefunden wurde.

8.4.7 Fehleranalyse

Mit den Methoden, die in den letzten beiden Abschnitten geschildert wurden, erhalten wir die bestmögliche Hyperbel für unsere Messpunkte. Noch nicht geklärt ist die Frage, ob die Messwerte die Theorie stützen oder ob im Gegenteil das Experiment fehlerhaft war, oder gar die Theorie in Frage zu stellen ist. Um dieses Problem zu klären, müssen wir eine Fehleranalyse durchführen. Sie liefert eine Beurteilung der Vertrauenswürdigkeit der Messergebnisse.

Wenn wir uns das *p-V*-Diagramm mit eingezeichneter Kurve ansehen, müssen wir feststellen, dass einige Meßpunkte mehr oder weniger stark von der Kurve abweichen, zunächst also keine Bestätigung der Theorie möglich ist. Wenn wir voraussetzen, dass die Theorie korrekt ist, müssen wir die Ursache in Ungenauigkeiten des Versuchsaufbaus und der Einstellungen suchen. Für den Gasversuch können wir davon ausgehen, dass sich in der Konstanten k Temperaturschwankungen, bedingt durch Gasverdichtung oder Gasverdünnung, während der Versuchsdurchführung bemerkbar machen. Ferner lässt sich die Größe des Drucks p am Manometer nicht exakt ablesen. Die Ungenauigkeit von k und p bewirkt natürlich, dass auch das Volumen V keine exakte Größe ist. Dies können wir berücksichtigen, indem wir eine Fehlertoleranz ΔV_i für jede Messung V_i angeben. Abhängig von der physikalischen Formel, können wir eine entsprechende Fehlerfortpflanzungsformel herleiten.

Fehlerfortpflanzungsgesetz

Allgemein gilt zunächst (siehe /8.3/ oder /8.4/): Hängt eine physikalische Größe y von mehreren fehlerbehafteten Größen a, b, c, ... ab, so gilt das folgende **Fehlerfortpflanzungsgesetz,**

$$\Delta y = \pm \sqrt{\left(\frac{\partial y}{\partial a}\Delta\ a\right)^2 + \left(\frac{\partial y}{\partial b}\Delta\ b\right)^2 + \left(\frac{\partial y}{\partial c}\Delta\ c\right)^2 + \ldots}$$

d.h., die Fehlerabweichung Δy ist abhängig von den partiellen Ableitungen nach a, b, c, ... Für unser Gasgesetz ergibt sich die folgende Fehlerabweichung:

$$\Delta V_i = \pm\sqrt{(V_i \times \Delta k/k)^2 + (V_i \times \Delta p/p)^2}$$

Die Ausdrücke $\Delta p/p$, $\Delta k/k$ sind Bruchzahlen zwischen 0 und 1. Sie beziehen sich auf die Ungenauigkeit der Einstellung bzw. der Versuchsapparatur. Zum Beispiel bedeutet 1/100 für $\Delta p/p$, dass die Ungenauigkeit für die Druckeinstellung 1% beträgt.

Die obige Formel bauen wir wieder in unser Theorie-Tabellenblatt per *AutoAusfüllen* ein und erhalten damit die Struktur, die in Bild 8.11 zu sehen ist. Für unser Beispiel nehmen wir an, dass die Ungenauigkeit für unseren Versuch mit 7% sehr hoch ist.

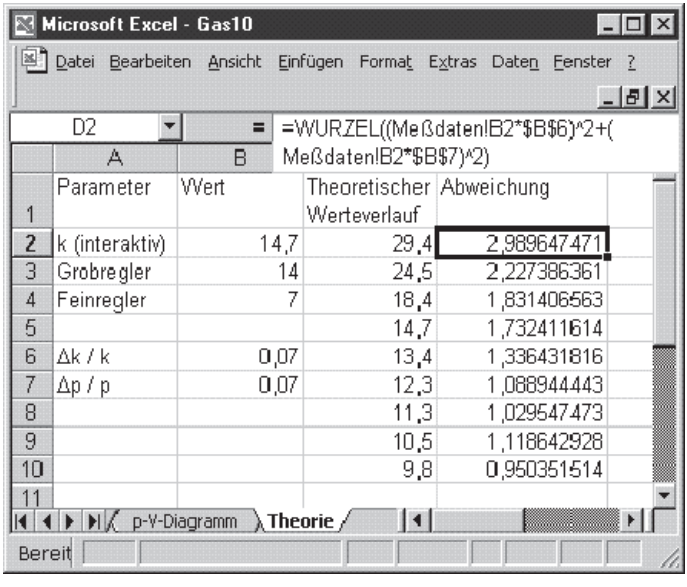


Bild 8.11 Fehlerabweichung für alle Messungen

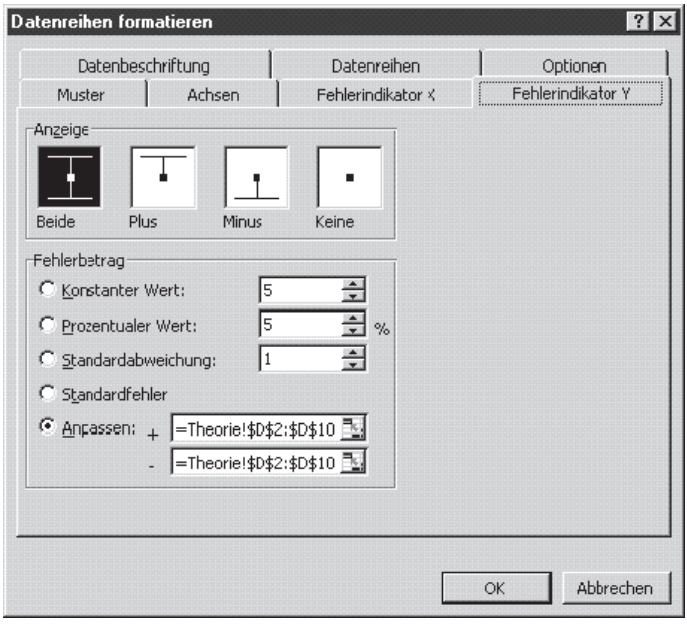


Bild 8.12 Einstellung der Fehlerbalken

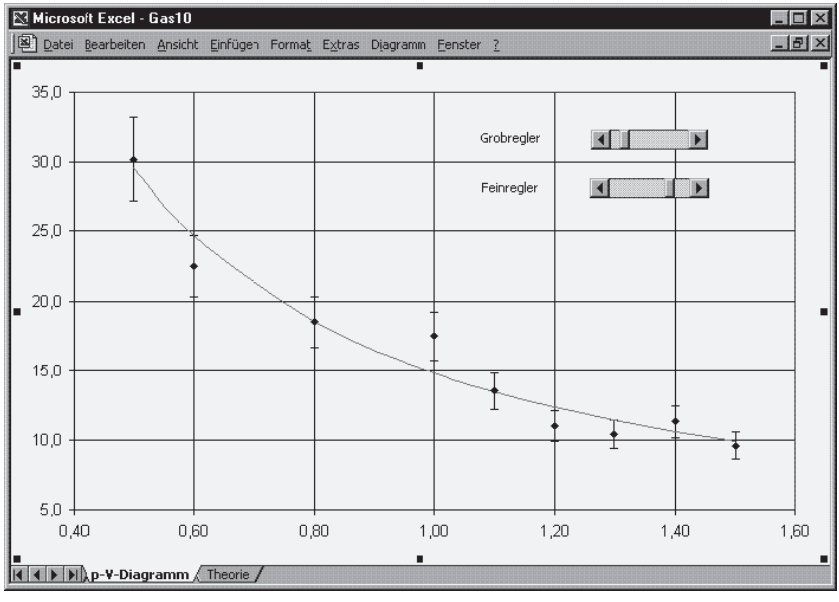


Bild 8.13 Endgültiges p-V-Diagramm mit Fehlerbalken

Die Fehlerabweichungen können wir nun als letzten Arbeitsschritt in unser *p-V*-Diagramm als Fehlerbalken einzeichnen. Hierzu gehen wir zum Diagrammblatt und aktivieren dort die Punktekette per Doppelklick. Wir erhalten das Dialogblatt *Datenreihen formatieren*, das mehrere Registerkarten enthält. Auf der Karte *Fehlerindikator Y* können wir unsere Fehlerberechnung einbauen, indem wir als Darstellung den Punkt *Beide* auswählen, um einen Fehlerbalken nach unten und nach oben zu erhalten. Die Berechnung des Fehlerbetrags erfolgt nach unserer eigenen Methode, wir müssen deshalb den Punkt *Anpassen* anklicken. In den beiden Eingabefeldern dahinter bestimmen wir die Abweichung nach oben und unten durch Angabe der Datenreihe, wel-

che die Fehlerberechnung für jede einzelne Messung enthält. Excel ist dadurch in der Lage, jedem einzelnen Element in der Punktekette seine individuelle Fehlerabweichung zuzuordnen.

Grafisch wird die Fehlerinformation als Balken im *p-V*-Diagramm dargestellt. Damit können wir leicht im Diagramm ablesen, ob unser Experiment korrekt abgelaufen ist, oder nicht. Bild 8.13 zeigt uns, dass die Messungen für $p = 1{,}00$ und $p = 1{,}20$ außerhalb des Toleranzbereichs liegen. Der Versuch lief also nicht korrekt ab. Als Reaktion müssen wir korrekterweise den gesamten Versuch überprüfen und wiederholen.

8.4.8 Aufgaben

8.1 Versuchen Sie, eine Gerade durch die Punkte zu legen, indem Sie einen weiteren theoretischen Werteverlauf berechnen und entsprechend als Linie zum Diagramm hinzufügen. Führen Sie, wie bei der Hyperbel, eine Fehlerabschätzung durch! Liegen alle Messpunkte innerhalb der Fehlertoleranzen?

8.2 Ergänzen Sie das Experiment, indem Sie Messungen für verschiedene Temperaturen durchführen. Für jede Temperatur entsteht dadurch eine eigene Kennlinie, die Isothermen. Fügen Sie die Isothermen in das *p-V*-Diagramm ein und schätzen Sie die Fehlerabweichungen ab!

8.3 Führen Sie die Experimentauswertung mit anderen Versuchen durch!

8.4 Für bestimmte Kurventypen (linear, polynomial, logarithmisch, ...) bietet Excel eine vorgefertigte Möglichkeit, eine Näherungskurve einzuzeichnen. Sie können diese Möglichkeit nach Anklicken einer Punktekette testen. Nach der Auswahl durch Anklicken steht Ihnen im Menü „*Diagramm*“ der Befehl „*Trendlinie hinzufügen*“ zur Verfügung. Wählen Sie einen geeigneten Kurventyp aus und kreuzen Sie auf der Registerkarte „*Optionen*“, „*Gleichung im Diagramm darstellen*“ sowie „*Bestimmtheitsmaß im Diagramm darstellen*“ an.

8.5 Programmieren mit Excel

8.5.1 Einführung

Nach den bisherigen Ausführungen können wir in Excel zwei Ebenen unterscheiden:

Ebene der Tabellenkalkulation

- Auf der **Ebene der Tabellenkalkulation** verknüpften wir auf Tabellenblättern Daten mit Funktionen, die wiederum Werte für andere Funktionen lieferten, und auf Diagrammblättern Diagramme mit Datenreihen. Es entstanden einfache nichtrekursive, funktionale Programme, die bei vorgegebenem Input einen Output berechnen. Dabei dienten die Tabellenblätter nicht nur der Programmbeschreibung, sondern lieferten auch eine grafische Benutzeroberfläche für die Eingabe von Werten und für die Präsentation von Ergebnissen.

Ebene des Editors

- Auf der **Ebene des Editors** lernten wir eine Reihe von Excel-Menübefehlen kennen, die wir für den Aufbau von Kalkulationen und zum Formatieren der Benutzeroberflächen nutzten.

Diese beiden Ebenen sollten jedoch einige weitere Anforderungen erfüllen, die wir in Abschnitt 8.4 noch nicht realisieren konnten:

Fehleranalyse

- Unsere *Fehleranalyse* ist bislang noch unvollständig; sie basiert nur auf einer optischen Kontrolle, indem wir überprüfen, ob die Kurve inner- oder außerhalb der Fehlerbalken verläuft. Solange die Messabweichungen und die Fehlertoleranzen, wie in unserem Beispiel, recht hoch sind, funktioniert die optische Kontrolle recht gut. Bei guten Versuchsaufbauten wird der Spielraum wesentlich klei-

ner sein, so dass das visuelle Abschätzen schwieriger wird. Unter diesen Bedingungen wäre es besser, einen exakten Vergleich der Zahlenreihen durchzuführen. Auf der Ebene der Tabellenkalkulation fehlt uns aber eine vorgefertigte Tabellenfunktion, welche einen solchen Zahlenvergleich vornimmt.

Zeichnen von Messpunkten

- Zum *Zeichnen der Messpunkte* auf dem Diagrammblatt muss der Benutzer zur Auswahl der Menübefehle und ihrer Parameter 10 verschiedene Einzelaktionen ausführen. Da dieser Ablauf bei jeder Experimentauswertung nach demselben Muster erfolgt, wäre es schön, wenn es einen speziellen Menübefehl gäbe, der das Zeichnen ohne weitere Benutzer-Interaktion erledigt.

Funktionalität des Excel-Editors

- Der *Excel-Editor* bietet sehr vielfältige Möglichkeiten zur Bedienung des Systems. Eine solch umfangreiche Funktionalität ist allerdings auch mit dem Nachteil verbunden, dass sie logischerweise eine umfangreiche und zeitaufwendige Einarbeitung erforderlich macht! Die direkte Benutzung von Excel für die Auswertung von Experimenten setzt diese Einarbeitung voraus. Besser wäre es, wenn für den Benutzer nur solche Menübefehle sichtbar sind, die er für die Experimentausarbeitung benötigt. Er hätte dann eine für seine Aufgabe maßgeschneiderte Benutzeroberfläche und Funktionalität.

Um Anforderungen dieser Art zu erfüllen – die ja nicht nur bei unserer Experimentauswertung auftreten, sondern ein allgemeines Phänomen sind –, bietet Excel eine Schnittstelle zu den Excel-Funktionen und eine vollständige Programmiersprache an. Durch die Offenlegung der Schnittstelle erhält Excel die in Bild 8.14 skizzierte Architektur.

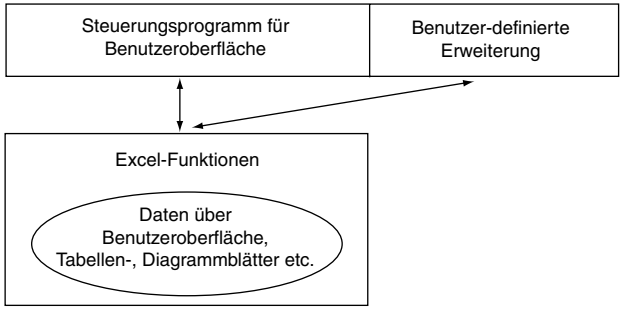


Bild 8.14 Architektur von Excel

Trennung von Oberfläche und Funktion

Excel trennt die Steuerung der Benutzeroberfläche von den Funktionen ab, welche die Elemente von Excel manipulieren und evtl. visualisieren. Jeder Menübefehl entspricht dem Aufruf von einer oder mehreren Excel-Funktionen. Elemente von Excel sind die globalen Optionen des Arbeitsbereiches, Angaben zu Arbeitsmappen, Tabellenblätter, Diagrammblätter, deren Inhalte usw. Zu den Elementen gehören ferner auch alle Menüs und Symbolleisten. Diese Elemente sind Daten im Hauptspeicher, die durch die Excel-Funktionen abgekapselt werden. Dadurch ist eine direkte Änderung der Daten nicht mehr möglich; es geht nur noch über die Excel-Funktionen, die in einer Programmbibliothek versammelt sind und den Anwendern für die eigene Programmierung zur Verfügung stehen. Da auch Menü- und Symbolleisten in die öffentliche Schnittstelle einbezogen sind, kann das Kernsteuerungsprogramm von Excel um neue Funktionalitäten erweitert werden. Mit diesem Mittel kann die Benutzeroberfläche letztlich so weit verändert werden, dass Excel nur noch für geübte Augen erkennbar bleibt!

VisualBasic

Zur Entwicklung von anwendungsspezifischen Erweiterungen steht uns in Excel **VisualBasic** zur Verfügung. Die Programmierumgebung ist in Excel integriert und ermöglicht dadurch eine leichte Einbindung von Benutzerprogrammen.

8.5.2 Die Objektbibliothek

Bevor wir mit dem Programmieren anfangen, verschaffen wir uns einen Überblick über die Excel-Funktionen. Das ist unbedingt notwendig, da die reichhaltige Funktionalität von Excel auch in der Fülle der aufrufbaren Funktionen ihren Ausdruck findet. Die Funktionen gruppieren sich um Objekttypen, welche Excel strukturieren. Den Objekttypen sind Eigenschaften und Methoden zugeordnet. Für alle wichtigen Konzepte in Excel existieren entsprechende Objekttypen, deren Eigenschaften Daten über die Konzepte enthalten und deren Funktionen das Verhalten der Objekte modellieren.

Objekttyp

Bild 8.15 zeigt einen Ausschnitt aus der Objektbibliothek mit den wichtigsten **Objekttypen**, die wir bei der Erweiterung der Experimentauswertung auch verwenden werden. Die Objekttypen sind hierarchisch gegliedert, da eine Anwendung (Application) aus mehreren geöffneten Arbeitsmappen (Workbooks), eine Arbeitsmappe aus mehreren Tabellenblättern (Worksheets) und Diagrammen (Charts) usw. bestehen kann. Die Hierarchiestufen sind in Bild 8.15 durch Einrückungen wiedergegeben.

Application

Oberster Objekttyp ist **Application**. Diesem Objekttyp sind alle Eigenschaften zugeordnet, die für das gesamte Excel-System gelten, z. B. alle einstellbaren Optionen (siehe den entsprechenden Menübefehl im Menü *Extras*). Ebenso sind alle Methoden zugeordnet, die sich auf die gesamte Anwendung auswirken, z. B. die Methoden

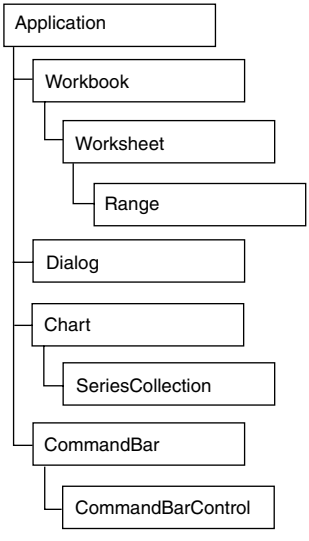


Bild 8.15 Ausschnitt aus der Objekthierarchie der Excel-Bibliothek

Quit (der Aufruf dieser Methode beendet Excel), Undo (macht die letzte Benutzeraktion rückgängig; unabhängig davon, ob sich der Benutzer in einem Diagramm-, Tabellenblatt oder einer anderen Umgebung befindet) und Repeat (wiederholt die letzte Benutzeraktion). Eine Anwendung besteht aus mehreren geöffneten Arbeitsmappen, mehreren Menü- und Symbolleisten.

Wenn die Excel-Anwendung aufgerufen wurde und evtl. eine oder mehrere Arbeitsmappen geöffnet sind, führt die Objekttypenhierarchie zum Aufbau einer Objekthierarchie im Hauptspeicher. Der Zugriff auf alle Objekte erfolgt über eine globale, bereits vorgegebene Variable, die, wie ihr Typ, *Application* heißt. Der Zugriff auf Eigenschaften und Methoden erfolgt in der üblichen Punkt-Notation: *Application.Repeat* ist z. B.

der Aufruf der Prozedur *Repeat* des Objektes *Application*, während sich der Aufruf *Application.ActiveWorkbook* auf die aktive Arbeitsmappe einer Anwendung bezieht (d.h. ein Fenster dieser Arbeitsmappe ist aktiv, und der Benutzer kann Eingaben für diese Arbeitsmappe durchführen).

Arbeitsmappe

Eine **Arbeitsmappe** wird als Datei abgespeichert, deshalb hat sie auch entsprechende Eigenschaften, wie *Name* (Datei-Name), *Path* (Verzeichnis-Pfad), *Readonly* etc., und entsprechende Methoden, wie *Activate* (eine Arbeitsmappe ist aktiv, wenn der Eingabe-Fokus auf eines ihrer Fenster gesetzt ist; gewöhnlich sind aktive Fenster durch einen blauen Titelbalken erkennbar) und *Save*. Eine Arbeitsmappe besteht u.a. aus Tabellenblättern und Diagrammblättern, die wiederum eigenständige Objekttypen sind.

Tabellenblatt

Jedes **Tabellenblatt** besteht aus einer Menge von Zellen, für die allerdings kein Objekttyp existiert. Statt dessen gibt es einen Objekttyp **Bereich** (Range), der eine Menge von Zellen umfassen kann. Als Bereich bezeichnet man in Excel eine Gruppe von Zellen, die zusammen ausgewählt und markiert wurden (optisch gewöhnlicherweise durch eine schwarz gefärbte, rechteckige Fläche gekennzeichnet). Da viele Eigenschaften für Zellengruppen gesetzt und viele Methoden auf Zellengruppen angewandt werden können, existiert nur der Objekttyp **Bereich.** Dieser Objekttyp hat z.B. die Eigenschaften *Value* (um Zahlenwerte oder Texte in eine oder mehrere Zellen einzutragen), *Formula* (um eine Berechnungsformel einzufügen), *Count* (Anzahl der Zellen in einem Bereich) und *Font* (ein Objekttyp, dem wiederum die Eigenschaften von Schriftarten zugeordnet sind, z.B. Größe). Mit dem Objekttyp Bereich sind auch wichtige Methoden assoziiert, z.B. *Sort*, *Find*, *Copy* etc.

Bereich

Sowohl Tabellenblätter als auch Diagrammblätter können zudem eine Reihe von Dialog- und Bildelementen enthalten. Unter Dialogelementen verstehen wir eine ganze Gruppe von Objekttypen, wie Bildlaufleiste, Schaltfläche, u.ä., die auch in Diagrammblättern vorkommen können (s. Abschnitt 8.4.5). Bildelemente sind Kreise, Rechtecke, Linien, Bitmaps etc., die ebenfalls eigenständige Objekttypen der Excel-Objektbibliothek sind.

Diagramm

Diagrammblätter sind vom Objekttyp **Diagramm** (Chart). Jedes Diagramm ist die Visualisierung einer Zahlenmatrix, bei der für jede Zeile oder Spalte eine Gruppe von gleichfarbigen Grafikobjekten (gleichfarbige Linie oder gleichfarbige Gruppe von Balken, Säulen etc.) angelegt wird. Jede solche Gruppe ist vom Typ *Datenreihe* (Series) mit einem Verweis auf eine Zeile oder Spalte der Zahlenmatrix (gespeichert in der Eigenschaft Werte, die eine Angabe über ein Bereichsobjekt enthält). Wie wir schon in Abschnitt 8.5 gesehen haben, können Diagramme außerdem eine Reihe von Dialog- und Bildelementen besitzen. In ähnlicher Weise haben auch Symbolleiste und Menüleiste eine Binnenstruktur von Subobjekten.

Der Zugriff auf die einzelnen Hierarchiestufen erfolgt über die sog. Listenmethoden. Wenn wir z.B. das Tabellenblatt *Theorie* einer aktiven Arbeitsmappe ausdrucken wollen, können wir zunächst auf die Methode *TabellenblattListe* (*Worksheets*) des Objekttyps Arbeitsmappe zurückgreifen. Einzelne Objekte innerhalb einer Liste können wir mit dem einzigen Argument der Listenmethoden identifizieren. Die komplette Anweisung zum Ausdrucken ist daher:

```
Application-ActiveWorkbook.Worksheets("Theorie").Print
```

8.5.3 Erweiterung der Experimentauswertung

Die Excel-Objektbibliothek und VisualBasic (VB) wollen wir nun benutzen, um unsere Experimentauswertung zu erweitern. Wir realisieren zuerst eine exakte Fehlerkontrolle, bevor wir das Zeichnen der Messpunkte vereinfachen. Abschließend ergänzen wir die Experimentauswertung um ein neues Menü, das Befehle für die beiden nachfolgenden benutzerdefinierten Funktionen enthält.

Beispiel 8.6

VB-Programm zur exakten Fehlerkontrolle

Die exakte Fehlerkontrolle basiert auf den folgenden Überlegungen: Jedem experimentellen Wert e entspricht ein berechneter, theoretischer Wert t. Für jeden experimentellen Wert haben wir bereits seine Fehlertoleranz f bestimmt. Zur exakten Fehlerkontrolle müssen wir überprüfen, ob für alle Werte die Ungleichung $|t - e| \leq f$ erfüllt ist. Für die Ausnahmen soll eine einfache Warnung am Bildschirm ausgegeben werden. Ein Programm, das diese Anforderungen realisiert, muss der Reihe nach die Zellen der Spalte B des Tabellenblattes *Messdaten*, der Spalten C und D von *Theorie* miteinander vergleichen. Wie dies geschehen kann, zeigt das folgende VB-Programm:

```
1  Option Explicit
2  Sub Fehlerkontrolle()
3  Dim TheorieBlatt As Object
4  Dim MeßdatenBlatt As Object
5
6  Set TheorieBlatt = _
7     Application.ActiveWorkbook.Worksheets("Theorie")
8  Set MeßdatenBlatt = _
9     Application.ActiveWorkbook.Worksheets("Meßdaten")
10
11 Dim i As Integer
12 i = 2
13
14 Dim ExperimentWert As Object
15 Dim TheorieWert As Object
16 Dim FehlerWert As Object
17
18 Set ExperimentWert = MeßdatenBlatt.Cells(2, 2)
19 Set TheorieWert = TheorieBlatt.Cells(2, 3)
20 Set FehlerWert = TheorieBlatt.Cells(2, 4)
21
22 Do Until IsEmpty(ExperimentWert.Value)
23    If Abs(TheorieWert.Value - ExperimentWert.Value) >
         FehlerWert.Value Then
24          MsgBox „Die Abweichung für p = „ & _
25          MeßdatenBlatt.Cells(i, 1).Value & _
26          " ist größer als die statistische _
27          Fehlerabweichung!"
28    End If
29    i = i + 1
30    Set ExperimentWert = MeßdatenBlatt.Cells(i, 2)
31    Set TheorieWert = TheorieBlatt.Cells(i, 3)
32    Set FehlerWert = TheorieBlatt.Cells(i, 4)
33 Loop
34 End Sub
```

Das VB-Programm werden wir nun Zeile für Zeile erklären. Die Anweisung in Zeile 1 besagt, dass alle Variablen des Programms deklariert werden müssen, d. h., allen Variablen muss ein Datentyp zugeordnet werden. Würden wir diese Anweisung weglassen, ergäbe sich der Datentyp der Variablen aus ihrer ersten Verwendung. Die Zeile 2 enthält die Prozedurdefinition, die mit dem Schlüsselwort *Sub* beginnt. Innerhalb der Klammern nach dem Prozedurnamen können Argumente definiert werden. Außer Prozeduren können auch Funktionen mit dem Schlüsselwort *Function* definiert werden.

Variablendeklaration

Die Zeilen 3 und 4 deklarieren zwei Variablen, die sich auf Tabellenblattobjekte beziehen werden. **Variablendeklarationen** beginnen mit dem traditionellen Basic-Schlüsselwort *Dim*; es folgt der Variablenname und hinter dem Schlüsselwort *As* ist der Datentyp angegeben.

In den Zeilen 6 bis 9 initialisieren wir die beiden Objektvariablen mit Bezügen zu den Tabellenblättern *Theorie* und *Messdaten*. Bezüge zu Objekten werden durch den Wertzuweisungsoperator „=“ hergestellt, dem das Schlüsselwort *Set* vorangesetzt wird. Für die Wertzuweisung müssen wir das richtige Objekt in der Objekthierarchie über eine Listenmethode lokalisieren. Der Unterstrich „_“ wird in diesen Programmzeilen verwendet, um eine Anweisung in einer weiteren Zeile fortzusetzen.

In den Zeilen 11 und 12 wird eine Laufschleifenvariable vorbereitet. Zunächst wird die Variable *i* mit dem Standarddatentyp *Integer* als eine 2 Byte lange ganze Zahl deklariert. Weitere Standarddatentypen können in der Online-Hilfe, im VisualBasic-Sprachverzeichnis nachgeschlagen werden. Die folgende Zeile zeigt eine Wertzuweisung für eine Variable eines Standarddatentyps, die ohne das Schlüsselwort *Set* auskommt.

Zugriff auf Zellen

Anschließend deklarieren wir in den Zeilen 14 bis 16 die Zellen, die verglichen werden sollen, als Objekte, und initialisieren sie in den Zeilen 18 bis 20. Dabei greifen wir auf eine weitere Listenmethode, *Cells*, zurück, um die Variablen auf Zellen der jeweiligen Tabellenblätter zu beziehen. Innerhalb eines Tabellenblattes ist eine Zelle eindeutig durch ihre Zeilen- und Spaltennummer lokalisierbar. Die Methode *Cells* hat daher zwei diesbezügliche Argumente. Alle Wertelisten beginnen in der zweiten Zeile (in der ersten Zeile sind die Überschriften eingetragen). Deshalb hat der erste Parameter der Methode *Cells* den Wert $i = 2$. Der zweite Parameter richtet sich nach der jeweiligen Spaltennummer.

Laufschleife

Die Zellenwerte können nun Zeile für Zeile miteinander verglichen werden. Dies geschieht mit einer **Laufschleife**, die sich über die Zeilen 22 bis 33 erstreckt. Die gezeigte Laufschleife beginnt mit den Schlüsselwörtern *Do Until IsEmpty* und endet mit dem Schlüsselwort *Loop*. Sie testet zu jedem Beginn eines Schleifenlaufes, ob die Bedingung *IsEmpty(Experimentwert.Value)* erfüllt ist. Trifft dies zu, wird die Laufschleife abgebrochen, ansonsten werden die Anweisungen im Rumpf der Laufschleife ausgeführt. In unserem Falle prüft die Bedingung mittels der VB-Funktion *IsEmpty*, ob die Zelle *Experimentwert* einen Eintrag enthält oder leer ist.

Danach erfolgt in der Zeile 23 die Überprüfung, ob der Absolutbetrag der Differenz zwischen dem theoretischen Wert und dem experimentellen Wert größer als die erlaubte Fehlerabweichung *f* ist. Ist dies der Fall, erhält der Benutzer eine entsprechende Meldung, die durch die Funktion *Msgbox* ausgegeben werden kann. Der Meldungstext ist eine Zeichenkette, die mit dem sog. Konkatenationsoperator „&“ zusammengesetzt wird. Beim Zusammensetzen können Texte und Zahlen gemischt werden; VB wandelt Zahlen automatisch in Text um.

Nach der Überprüfung sind die Zellen in der nächsten Zeile an der Reihe. Die Bereichsobjekte *Experimentwert*, *Theoriewert*, *Fehlerwert* müssen deshalb auf Zellen in der nächsten Zeile positioniert werden. Dies geschieht durch die Anweisungen in den Zeilen 29 bis 32. Nach dem Schleifenende beenden wir die gesamte Prozedur mit den Schlüsselwörtern *End Sub*.

Zu klären bleibt nur noch der Aufruf unserer Prozedur. Zuvor wollen wir jedoch noch ein zweites Beispiel für ein VB-Programm betrachten.

Verwendung des Makro-Rekorders

Das Besondere an diesem neuen Beispiel wird sein, dass wir große Teile dieses Programms nicht Buchstabe für Buchstabe eintippen müssen. Statt dessen setzen wir den Makro-Rekorder ein. Dieses Werkzeug eignet sich hervorragend dafür, mehrere Benutzerinteraktionen zu einer neuen Prozedur zusammenzufassen. Zur Demonstration greifen wir auf das Zeichnen von Messpunkten in ein Diagramm zurück (siehe Abschnitt 8.4.2). Wir wollen eine neue Prozedur definieren, in welcher die einzelnen Editorbefehle automatisch aufgerufen werden. Zu diesem Zweck wiederholen wir die Eingaben, die wir in Abschnitt 8.4.2 kennengelernt haben, aber nun bei eingeschaltetem Makro-Rekorder. Dies bewirkt, dass Excel die Benutzereingaben in Form von Aufrufen der zugehörigen Excel-Funktionen auf einem neuen Modul-Blatt „mitschreibt“ und automatisch eine Prozedur anlegt.

Um dies zu erreichen, rufen wir zunächst den Befehl *„Extras|Makro|Aufzeichnen...“* auf. Im eingeblendeten Dialogfeld können wir die Einstellungen unverändert übernehmen. Anschließend wartet der Makro-Rekorder auf unsere Befehlseingaben. Nach Abschluss der Eingaben beenden wir wieder die Aufzeichnung, entweder über das Menü „*Extras*“ oder durch Drücken der einzigen Schaltfläche auf der Symbolleiste „*Aufzeichnen beenden*“ (sie wird beim Aufzeichnen automatisch aktiviert).

Das Ergebnis der Aufzeichnung bezieht sich genau auf die Randbedingungen, die beim Gasversuch gegeben sind, z. B., dass die Experimentwerte im Bereich A2:B10 stehen. Für den allgemeinen Gebrauch müssen wir also die Prozedur noch etwas verallgemeinern, wie dies in folgendem Beispiel zu sehen ist.

Beispiel 8.7

Prozedur zum automatischen Zeichnen von Messpunkten

```
1  Option Explicit
2
3  Sub Zeichnen()
4     Dim Count As Integer
5     Dim Meßdaten As Object
6
7     Set Meßdaten = ActiveWorkbook.Sheets(“Meßdaten”)
8     Count = Meßdaten.Columns(1).Find(Empty).Row – 1
9
10    Charts.Add
11    ActiveChart.ChartWizard _
12       Source:=Meßdaten.Range(“A2:B” & Count),  _
13       Gallery:=xlXYScatter, _
14       Format:=3, _
15       PlotBy:=xlColumns, _
16       CategoryLabels:=1, _
17       SeriesLabels:=0,  _
18       HasLegend:=2, Title:=””, _
19       CategoryTitle:=””,  _
```

```
20       ValueTitle:="", _
21       ExtraTitle:=""
22    ActiveChart.Selection.Interior. _
23       ColorIndex = xlNone
24    ActiveChart.Name = "p-V-Diagramm"
25    ActiveChart.SizeWithWindow = True
26    ActiveChart.Move Sheets(3)
27 End Sub
```

Die neue Prozedur nennen wir *Zeichnen* (siehe Zeile 3). Die ersten 8 Zeilen der Prozedur enthalten Variablendeklarationen und -Initialisierungen. Eine erste Besonderheit ist in Zeile 7 zu finden, denn es fehlt bei der Eigenschaft *Active Workbook* die Angabe der globalen Variablen *Application*. Da diese Variable sehr häufig verwendet wird, erlaubt VB in vielen Situationen die Variable wegzulassen. Für viele Eigenschaften und Methoden ist sie nur eine optionale Angabe; manchmal ist sie allerdings doch notwendig. Wann sie weggelassen werden kann und wann nicht, kann der Benutzer nur herausfinden, indem er die Hilfe-Informationen der verwendeten Eigenschaft oder Methode durchliest. Dem Programmierer wird dadurch zwar Schreibarbeit erspart; besonders für den Anfänger entsteht dadurch aber eine gefährliche Fehlerquelle!

optionale Angabe des Objektes

Die Zeile 8 enthält eine zweite Besonderheit, die für das Verallgemeinern von Makro-Aufzeichnungen wichtig ist. Im allgemeinen Fall müssen wir von einer variablen Anzahl von Messungen ausgehen; es werden nicht gerade 9 Messungen sein. Deshalb muss die Prozedur feststellen, wie viele Einträge sich auf dem Messblatt befinden. Dank der zur Verfügung stehenden Methoden genügt hierfür eine einzige Anweisung. Wir wenden einfach die Methode *Find* des Objekts *Range* auf die erste Spalte des Messdatenblattes an und suchen nach der ersten leeren Zelle (mit dem Schlüsselwort *Empty*). Als Ergebnis erhalten wir die erste leere Zelle in dieser Spalte (ein Bereichsobjekt), deren Eigenschaft *Row* uns die Zeilennummer liefert. Vom Ergebnis müssen wir eine 1 abziehen, um die Nummer der letzten gefüllten Zelle zu erhalten.

Für die Zeile 10 verzichtete die Makro-Aufzeichnung ebenfalls auf optionale Angaben.

programmiertes Erzeugen eines Diagramms

Die Anweisung ist ein Beispiel für das Erzeugen von neuen Objekten in Excel – in unserem Fall für das Erzeugen eines Diagramms. Dies geschieht über die Listen, die eine Doppelrolle spielen: Einerseits sind sie Methoden, andererseits Objekte. So wird in der obigen Anweisung die *Charts* als ein Listenobjekt verwendet, dem selbst wieder Methoden zugeordnet sind. Dazu gehört auch die Methode *Add*, mit deren Hilfe der Liste ein neues Diagrammobjekt hinzugefügt werden kann. Nach demselben Schema besitzen auch andere Listenobjekte eine *Add*-Methode.

Mit dem Editorbefehl für das Einfügen des Diagramms wurde der Diagrammassistent aktiviert, der in mehreren Schritten durch die Diagrammauswahl führte. Analog hierzu wird in unserer Prozedur die Methode *ChartWizard* für das Objekt *Active Chart* aufgerufen, die eine Vielzahl von optionalen Parametern besitzt. Um die Parametereingabe in einem solchen Fall zu erleichtern, bietet uns VB die Möglichkeit, beim Aufruf die Parameternamen zu verwenden. Bei anderen Programmiersprachen entscheidet üblicherweise beim Prozeduraufruf die Position eines Wertes über die Zuordnung zu einem Parameter. Die allgemeine Syntax für den Prozedur- oder Methodenaufruf entspricht der Form:

```
<Prozedur><Parameter-Name 1>:=<Wert 1>;<Parameter-Name 2>:=<Wert 2>…
```

Eine weitere Besonderheit von VB ist, dass die Parameter nur dann geklammert werden, wenn sie Parameter einer Funktion (mit einem Ausgabe-Parameter) sind, aber nicht bei einer Prozedur!

Am wichtigsten ist beim obigen Aufruf der Parameter *Source*, dessen Wert sich auf den Zellenbereich mit der Diagrammdatenquelle bezieht. Dies geschieht mit Hilfe der Methode *Range*, deren Argument einen Zellenbereich adressiert. Da wir variabel bleiben wollen, beziehen wir uns auf die Variable *Count*. Die weiteren Parameter entscheiden über die Präsentation des Diagrammes. Dies geschieht mittels Zahlen, Texten oder Konstanten (z. B. ist *xlXYScatter* eine vordefinierte Konstante, die für Punktdiagramme steht).

In der Anweisung der Zeilen 22 und 23 wird die Farbe der Zeichnungsfläche gesetzt; in unserem Fall hatten wir uns für die Hintergrundfarbe entschieden. In den anschließenden Zeilen erfolgt das Umbenennen des Diagrammblattes, das Anpassen an die Fenstergröße, und das Umstellen des Blattes an die dritte Stelle im Register. Das Ende der Prozedur ist wieder durch *End Sub* gekennzeichnet.

Aufruf von Prozeduren

Damit sind unsere Beispielprozeduren vollständig beschrieben. Als nächstes müssen wir die beiden Prozeduren mit der Benutzeroberfläche verknüpfen, denn bis jetzt sind sie für den Benutzer noch nicht aufrufbar. Hierfür bietet Excel einige Möglichkeiten an. Die Prozeduren können u. a. mit einer Tastenkombination, mit einer Schaltfläche auf einem Blatt, mit einem Symbol auf einer Symbolleiste oder mit einer Menüleiste verknüpft werden. Um eine Menü- oder eine Symbolleiste zu ändern, rufen wir den Befehl *„Anpassen“* im Menü *„Extras“* auf. Beispielsweise könnten wir ein neues Menüelement „*Experiment*“ mit den beiden Befehlen „*Zeichnen*“ und „*Fehlerkontrolle*“ einbauen. Dies geschieht mit Hilfe der Registrierkarte „*Befehl*“ und den beiden Kategorien „*Neues Menü*“ und „*Makros*“. Nach der Auswahl in der Liste „*Befehle*“ können wir die Objekte mit der Maus in die gewählte Menüleiste hineinziehen und ein Makro zuordnen.

Das Schließen des Anpassen-Dialogs ist der letzte Arbeitsschritt, den wir benötigen, um unsere kleine Anwendung zu beenden. Wenn wir nun ein Tabellenblatt aktivieren, ist das neue Menü schon aktiv und für einen ersten Test bereit.

8.5.4 Ausblick

weitere Werkzeuge, Bibliotheken und Funktionen

Die Programmiersprache VB, die Objektbibliothek und der Menü-Editor bilden den Kern der Excel-Programmierumgebung. Darüber hinaus stehen dem Excel-Programmierer eine Reihe von weiteren Werkzeugen, Bibliotheken und Funktionen zur Verfügung, die für die Anwendungsentwicklung sehr wichtig sind.

- Der Benutzer ist in der Lage, sich neue Symbolleisten zu definieren, deren Symbole mit einem Pixel-Editor selbst bearbeitet und denen selbstdefinierte Prozeduren zugewiesen werden können.
- Ein Dialogeditor bietet die Möglichkeit, das Layout von Dialogmasken interaktiv zu gestalten. Dazu gehört auch eine Programmierschnittstelle, um die Dialogmasken in VB-Programmen zu verwenden.
- Es ist möglich, eigene Hilfe-Systeme im Stile von professionellen Windows-Anwendungen aufzubauen, und diese von VB-Programmen aus aufzurufen.
- Mit der Schnittstelle ODBC (Open Database Connectivity) kann von VB-Programmen aus auf Datenbanken zugegriffen werden. Die Architektur von ODBC ist dabei so ausgelegt, dass der Zugriff auf unterschiedlich formatierte Datenquellen in einheitlicher Weise erfolgt.

- Da das OLE-Protokoll (Object Linking and Embedding) bereitgestellt wird, kann ein VB-Programm auch auf Objekte zugreifen, die von anderen Prozessen verwaltet werden.
- Der Benutzer kann Prozeduren schreiben, die bei Eintreten von vorgegebenen Ereignissen automatisch ausgeführt werden (z.B. soll beim Öffnen einer Arbeitsmappe auf eine benutzerdefinierte Menüleiste umgeschaltet werden). Aus einer Menge von VB-Prozeduren kann der Entwickler zudem ein Add-In erzeugen, dass von anderen Benutzern zu ihren Systemen zugeladen werden kann.

8.5.5 Aufgaben

8.6 Führen Sie die Automatisierung der Benutzeraktionen fort und implementieren Sie Prozeduren, die das Zeichnen von Näherungskurven, Abschätzen von Fehlern und das Drucken von Diagrammen übernehmen.

8.7 Entwickeln Sie eine eigene Menüleiste und eigene Menüs, die sämtliche Befehle enthält, welche zur Experimentauswertung notwendig sind.

8.8 Entwerfen Sie eine Symbolleiste, die Schaltflächen für häufig verwendete Auswertungsbefehle enthält.

8.9 Erweitern Sie die Prozedur Fehlerkontrolle so, dass die Zellen mit Messwerten, die außerhalb der Fehlertoleranz liegen, rot eingefärbt werden.

8.10 Hinterlegen Sie auf einem Tabellenblatt die Formeln für wichtige Kurventypen, deren Formeln für den kleinsten Abweichungsgrad und deren Formeln für das Fehlerfortpflanzungsgesetz. Schreiben Sie eine Prozedur, damit der Benutzer in einer Dialogmaske die Auswahl des Kurventyps vornehmen kann. Alle weiteren Auswertungsprozeduren sollen diese Auswahl anschließend berücksichtigen.

8.6 Literatur

Literatur zu Physik und Statistik:

/8.1/ F. Bergmann, et al.: *Einführung in die Physik*. – Frankfurt: Diesterweg, 1984

/8.2/ Bosch, Karl: *Elementare Einführung in die angewandte Statistik*. – 2. Auflage. Braunschweig: Vieweg Studium 27, 1982

/8.3/ Hartwig, G.: *Einführung in die Fehler- und Ausgleichsrechnung*. – München: Hanser, 1967

/8.4/ Lichten, W.: *Skriptum Fehlerrechnung*. – Berlin: Springer, 1988

Literatur über das Programmieren mit Excel:

/8.5/ Buhl, A.: *Programmierung mit VisualBasic Excel 97*. – Oldenbourg, 1999

/8.6/ Kofler, M.: *VBA – Programmierung mit Excel 97*. – Addison-Wesley, 1997

/8.7/ Brudermanns, B.; Tiemeyer, E.: *Excel für Profis. Programmierung mit VBA*. – Rowohlt, 1997

Literatur über die Geschicht der Tabellenkalkulation:

/8.8/ Power, O. J.: *A Brief History of Spread-Sheets*. – `http://www.dssresources.com/history/sshistory.html`