

2 Excel mit VBA für verfahrenstechnische Anwendungen

Mitautoren: ANKE PATT, STEFAN PINNOW, JAKOB SCHNEIDER, SIMON WITTENHORST

Zielsetzung

Einrichten von Berechnungsblättern in Excel und Erstellung von Vorlagen von der Wahl geeigneter Formatierungen über die Festlegung der signifikanten Stellen bis zur Formatierung der Druckausgabe. Behandlung der Grundlagen im Umgang mit der Entwicklungsumgebung von VBA: Aufruf des VBA-Editors, Sicherheitseinstellungen, Umgang mit Prozeduren, Funktionen, Makros, Schaltflächen und Schleifen. Hinweise zu Datentypen, Zellbezügen, der Verwendung von Namen für Zellen, zur Ausführung von Makros und zu Schutzfunktionen. Verwendung des Objektkatalogs, Einstellungen im VBA-Editor sowie Tipps zur Verwendung des Direktbereichs, des Lokalfensters, zu Haltepunkten, zum Debuggen und zur Verwendung digitaler Signaturen. Erstellung einfacher Anwendungsbeispiele und Anwendung des Solvers. Erstellung benutzerdefinierter Funktionen (UDFs) (siehe die nachfolgend aufgeführten Berechnungsbeispiele) einschließlich Kurzbeschreibungen. Erstellung von Add-Ins. Ermittlung von Ausgleichsfunktionen. Nullstellensuche mit dem Verfahren des zentralen Differenzenquotienten (ZDQ-Verfahren). Durchführung von Berechnungen unter Anwendung eines Zirkelbezugs. Behandlung ausgewählter Arbeitsblattfunktionen.

Empfohlene Literatur

Excel in Naturwissenschaften und Technik von FLEISCHHAUER [29], *Excel programmieren* von KOFLER u. a. [45], *Excel for Scientists and Engineers* von BILLO [4], *Numerical Methods for Engineers* von CHAPRA u. a. [9].

Anwendungen und Berechnungsbeispiele in Excel

- Einrichten eines Excel-Berechnungsblattes (Abb. 2.1).
- Einfache Anwendungsbeispiele in VBA
 - Addition über eine Schaltfläche (Beispiel 2.1).
 - Makros aufzeichnen und Befehle daraus verwenden (Beispiel 2.2).
 - Berechnungen mit Schleifen, **For**-Schleife (Beispiel 2.3).
 - Berechnungen mit Schleifen, **Do-While**-Schleife (Beispiel 2.4).
- Anwendung des Solvers
 - Nullstellensuche für eine Funktion unter Anwendung des Solvers (Abb. 2.7).
- Anwendungen benutzerdefinierter Funktionen

- Berechnung des Widerstandsbeiwerts für ein innendurchströmtes Rohr (Beispiel 2.6).
- Berechnung des Sättigungsdampfdrucks und der Sättigungstemperatur für einen Reinstoff (Abb. 2.8).
- Berechnung der Dichte eines idealen und eines realen Gases (Abb. 2.9).
- Berechnung der spezifischen Verdampfungsenthalpie (Abb. 2.10).
- Erstellung von Kurzbeschreibungen zu UDFs (Abschnitt 2.5.5).
- Ermittlung von Ausgleichsfunktionen
 - Nichtlineare Regression unter Verwendung des Solvers (Abb. 2.12).
 - Berechnung der Ausgleichsfunktion der Siedelinie eines realen binären Systems mit einer benutzerdefinierten Funktion (Abb. 2.13).
- Nullstellensuche für eine Funktion mit dem ZDQ-Verfahren (Abb. 2.14).
- Berechnungen unter Anwendung von Zirkelbezügen
 - Berechnung einer Rückführung (Abb. 2.17).
- Ausgewählte Arbeitsblattfunktionen
 - Berechnung der Ausgleichsfunktion der Siedelinie eines realen binären Systems mit der RGP-Funktion (Abb. 2.18).

2.1 Einrichten von Berechnungsblättern in Excel

Neben den wichtigen Fragen, *was* in einem Berechnungsblatt berechnet und *wie* diese Berechnung durchgeführt werden soll, stellt sich die mindestens ebenso wichtige Frage, welcher Zweck mit der Erstellung verfolgt wird oder an wen es gerichtet ist. Adressaten von Excel-Berechnungsblättern oder technischen Dokumenten sind üblicherweise Kollegen, Vorgesetzte oder Kunden.

Formatierung von Berechnungsblättern

Die optisch ansprechende Form eines in Excel erstellten Dokuments – unter sparsamem Einsatz von Formatierungen und Farbe – soll die Aufmerksamkeit des Adressaten oder Auftraggebers wecken. Schon das Layout soll den Eindruck vermitteln, dass die durchgeführten Berechnungen gewissenhaft und sorgfältig erstellt wurden. Wenn auch noch der Ablauf der Berechnung nachvollziehbar und die wesentlichen Ergebnisse erkennbar sind – z. B. in einem Diagramm mit typischen Verläufen der relevanten Größen und dem berechneten Arbeitspunkt – vergrößert dies die Chance, dass das geplante Projekt beachtet, umgesetzt oder zum Auftrag wird.

Standard für den Umgang mit Papier, aber auch elektronischen Dokumenten, ist das A4-Format (nach DIN 476). Dieses Format sollte grundsätzlich auch für die Druck- oder PDF-Ausgabe von Excel-Berechnungsblättern verwendet und deshalb bei deren Erstellung berücksichtigt werden.

Ein Berechnungsblatt muss einen aussagekräftigen Titel und ggf. Hinweise auf technische Randbedingungen (Annahmen oder Vereinfachungen) oder Datenquellen

enthalten. Dazu gehören evtl. die Bezeichnung des Projekts, das Druck- und das Erstelldatum, der Name des Verfassers (wenn die Berechnungen korrekt sind) und der Dateiname (evtl. mit Hinweis auf den Speicherort).

Die (sparsame) Verwendung von Farben erhöht bekanntermaßen die Aufmerksamkeit. Unter der Registerkarte **Seitenlayout** **Farben** können ein Farbschema gewählt oder eigene Farben für ein Farbschema festgelegt werden. Es empfiehlt sich die Verwendung kräftiger, unterschiedlicher Farben, also nicht hell-, mittel- und dunkelblau, sondern besser blau, grün und rot, damit auch Linien in Diagrammen unterscheidbar sind.

Schriftart, Formatierungen

Als Standardschriftart für die in diesem Buch gezeigten Excel-Berechnungsblätter wurde Arial und als Schriftgröße 10 pt gewählt. Die Standardschriftart lässt sich am einfachsten unter der Registerkarte **Seitenlayout** **Designs** **Schriftarten** wechseln oder verändern. Die Verwendung von Arial hat mehrere Vorteile. Sie ist auf den meisten Systemen installiert, nimmt – verglichen mit anderen Schriftarten – relativ wenig Platz ein und die Darstellung von zeitlichen Ableitungen mit einem Punkt, z. B. \dot{V} oder \dot{m} , die in der Verfahrenstechnik oft gebraucht werden, ist möglich. Mehr dazu nachfolgend.

Beim Anlegen von Berechnungstabellen bietet es sich an, den Namen der entsprechenden physikalischen Größe, das Symbol, die Dimension und den Zahlenwert aufzuführen. Dabei sollten *Formelschreibweise* und *Formelsatz* gemäß DIN 1338 [13] beachtet werden. Die wichtigsten Hinweise dazu:

- Formelzeichen für Variablen sowie für Größenwerte (z. B. Naturkonstanten) werden in kursiver oder geneigter Schrift gesetzt (z. B. $T = 300\text{ K}$). DIN 1338 verlangt hier die Verwendung von Serifen. Abweichend davon wird in den Excel-Berechnungsblättern in diesem Buch für Variablen, Indizes und Größenwerte Arial als serifenlose Schrift verwendet.
- Indizes werden – je nach Bedeutung – in geradestehender oder kursiver Schrift gesetzt (z. B. beim Flüssigkeitsvolumen V^L der Index L für „flüssig“ oder bei der Stoffmenge n_i der Index i für die Komponente).
- Zeichen für Dimensionen werden geradestehend gesetzt (z. B. $T = 300\text{ K}$).

Um die Struktur eines Berechnungsblattes besser erkennen zu können, bietet es sich an, *Eingaben*, *Berechnungen*, *verknüpfte Zellen* und *Ergebnisse* zu unterscheiden:

- Eingaben: Zellen, in die Daten für nachfolgende Berechnungen eingegeben werden, sind zur besseren Kennzeichnung grau unterlegt und mit schwarzer, fatter Schrift versehen. Gegebenenfalls ist eine Datenüberprüfung mit der Registerkarte **Daten** **Datentools** **Datenüberprüfung** sinnvoll, um unzulässige Eingaben abzufangen (siehe dazu auch Abschnitt 2.5.1).
- Berechnungen: Zellen, in denen Berechnungen stattfinden, bekommen keine besonderen Formatierungen (also schwarze Schrift).
- Verknüpfte Zellen: Zellen, in die einfach nur Werte aus anderen Zellen (auch aus anderen Arbeitsblättern oder Dateien) übernommen werden, sind mit grauer Schrift formatiert.

- Ergebnisse: Besondere Werte oder Ergebnisse können durch Formatierungen (Fettdruck, Farbe) hervorgehoben werden.

Sinnvoll ist die Erstellung einer Formatvorlage (Excel-Vorlage), siehe nachfolgend.

Symbole für zeitliche Ableitungen

Symbole für zeitliche Ableitungen oder molare Größen können in Excel relativ einfach formatiert werden, seit es die UTF-8-Zeichenkodierung gibt. Für z. B. die Schriftarten Arial, Times New Roman oder Calibri wird dazu wie folgt vorgegangen: Für das Symbol \dot{V} zunächst V in eine Zelle eingegeben und während der Cursor rechts neben das Symbol platziert ist, unter der Registerkarte **Einfügen** **Symbole** **Symbol** die **Schriftart** **Arial** wählen¹ und den Zeichencode „U+0307“ (kurz für *Zeichencode 0307 von Unicode (hex)*) eingeben, wodurch das Zeichen „Combining Dot Above“ im **Subset** **Diakritische Markierungen** gewählt wird. Ein Klick auf **Einfügen** fügt das Symbol ein und nach dem **Schließen** des Fensters kann das neue (kombinierte) Symbol kursiv gesetzt werden. Für das Symbol \dot{V} wird entsprechend vorgegangen und der Zeichencode „U+0304“ („Combining Macron“) verwendet.

Absolute Zellbezüge, Namen

Oft ist sinnvoll, Zellen in Berechnungsblättern Namen (oft einfach nur das Symbol oder eine Abkürzung) zu geben. Dies hat die Vorteile, dass Gleichungen in Zellen leichter eingegeben werden können, besser lesbar sind und beim Kopieren oder Verschieben von Zellen die Bezüge erhalten bleiben, da damit ein *absoluter Zellbezug* hergestellt wird. In VBA-Makros ist dies noch wichtiger, da sonst z. B. beim Einfügen von Zeilen im Arbeitsblatt jeder Bezug im VBA-Code manuell korrigiert werden muss. Siehe dazu die Anleitung im Abschnitt 2.2 „Absolute [Zellbezüge, Namen](#)“.

Signifikante Stellen

Excel rechnet intern mit 15 Stellen, aber die Ergebnisse von Berechnungen werden in der Regel gerundet angegeben. Für praktische Berechnungen ist die korrekte Angabe von Zahlenwerten – also die Anzahl der signifikanten Stellen – äußerst wichtig, siehe dazu DIN 1333 [12]. Nach dieser Norm sind alle Stellen von der ersten von null verschiedenen Stelle von vorn bis zur Rundungsstelle signifikant. Dazu zählen damit auch Nullen zwischen signifikanten Ziffern und End-Nullen in Nachkommastellen. Die wissenschaftliche Schreibweise von Zahlen ist eindeutiger und deshalb grundsätzlich vorzuziehen.

Rechnen mit signifikanten Stellen: Bei der Addition und der Subtraktion hat das Ergebnis so viele Nachkommastellen, wie die Zahl mit den wenigsten Nachkommastellen. Bei der Multiplikation oder Division hat das Ergebnis so viele signifikante Stellen, wie die Zahl mit den wenigsten signifikanten Stellen.

Signifikante Stellen von Zwischenergebnissen: Für Zwischenergebnisse werden meist sinnvoll zusätzliche Stellen angegeben. Zu den in diesem Buch enthaltenen Berechnungsbeispielen sei grundsätzlich angemerkt, dass – insbesondere auch zum besseren

¹Schriftart „Arial“, nicht „Arial (Überschriften)“ oder „Arial (Textkörper)“.

Vergleich der Ergebnisse von Berechnungen – oft entgegen der vorgenannten Regeln gehandelt und zusätzliche Stellen aufgeführt werden.

Einrichten von Arbeitsblättern und Vorlagen in Excel

Die Erstellung eines neuen Arbeitsblattes oder einer Vorlage kann – unter Beachtung der Hinweise oben – wie folgt ablaufen:

- Öffnen der Arbeitsmappe und Einrichten des Arbeitsblattes: Überschrift, Kopf- und Fußzeilen usw.
- Einrichten des Farbschemas.
- Umsetzung der Hinweise zur Standardschriftart und zur DIN 1338 (Schriftarten und Schriftgröße, Symbole, Indizes, Dimensionen).
- Erstellung der Berechnungen.
- Kontrolle der Druckansicht.
- Speichern als `Excel-Vorlage (*.xltx)` oder `Excel-Vorlage mit Makros (*.xltn)`.

Beispielhaftes Excel-Berechnungsblatt

Abbildung 2.1 enthält die Druckansicht für ein beispielhaftes Excel-Berechnungsblatt (zur Erstellung der Berechnungen siehe Abschnitt 2.5.3). Anmerkung: Die nachfolgenden Berechnungsblätter werden aus Platzgründen ausnahmslos ohne Kopf- und Fußzeilen dargestellt. Die meisten in diesem Buch dargestellten Berechnungsblätter sind im PDF-Format auf der Seite www.unit-operations.de abrufbar.

2.2 Grundlagen und Tipps zu VBA

VBA ist eine leistungsfähige Skriptsprache, Bestandteil der Programme von Microsoft® Office und lässt sich in diesen Anwendungen nutzen. Es ist nicht mit VB zu verwechseln, einer Programmiersprache, mit der ausführbare Programme erstellt werden können.

Excel besitzt als Entwicklungsumgebung den sogenannten VBA-Editor. Dieser ermöglicht umfangreiche Berechnungen, die in einer reinen „Tabellenkalkulation“ nicht oder nur umständlich durchführbar wären. Außerdem können hier Abläufe automatisiert werden. Beispiele dafür:

- Einfügen von Text oder den Ergebnissen von Berechnungen im Arbeitsblatt,
- Durchführung komplexer iterativer Berechnungen,
- Ausgeben von Meldungen im Arbeitsblatt.

In VBA erstellte Funktionen lassen sich als UDFs (siehe Abschnitt 2.5) oder als Add-Ins (siehe Abschnitt 2.6) abspeichern und damit für weitere Anwender verfügbar machen. Die nachfolgenden Betrachtungen, Beispiele und Tipps basieren auf der Entwicklungsumgebung von Microsoft Excel 2010 und 2013.

FH Aachen, Lehrgebiet Thermische Energietechnik

Prof. Dr.-Ing. Uwe Feuerriegel

Berechnung der Dichte und des spezifischen/molaren Volumens von idealen/realen Gasen

Berechnungen mit der Zustandsgleichung idealer Gase und der PENG-ROBINSON-Gleichung für reale Gase. Stoffdaten aus VDI-Wärmeatlas (2013) 11. Aufl. Springer, Abschnitt D3.1. Tripeldaten Ethen: Smukala et. al. (2000) J. Phys. Chem. Ref. Data 29 (5), S. 1053–1121.

Stoff	Ethen
Temperatur ($T_u < T$)	250
	ϑ °C -23,15
Druck ($p < p_s$ für $T < T_{kr}$)	16,0
Stoffwerte aus VDI-Wärmeatlas, 11. Aufl., Abschnitt D3.1	
kritische Temperatur	T_{kr} K 282,35
kritischer Druck	p_{kr} bar 50,42
molare Masse	M kg kmol ⁻¹ 28,05
azentrischer Faktor	ω 1 0,087
Sättigungsdampfdruck (WAGNER-Gleichung)	$p_s(T)$ bar 23,288
Koeffizienten Sättigungsdampfdruck	A 1 -6,41327
	B 1 1,45469
	C 1 -1,24183
	D 1 -1,99446
Tripeltemperatur	T_{tr} K 103,989
Tripeldruck	p_{tr} bar 1,2265E-03
Berechnung mit der idealen Gasgleichung	
Dichte $\rho = p M / (R T)$	ρ kg m ⁻³ 21,59
spezifisches Volumen $v = 1 / \rho$	v m ³ kg ⁻¹ 0,04631
molares Volumen $\tilde{V} = v M$	\tilde{V} m ³ kmol ⁻¹ 1,299
Berechnung mit der PENG-ROBINSON-Gleichung	
	$p = \frac{RT}{\tilde{V} - b} - \frac{a(T)}{\tilde{V}^2 + 2b\tilde{V} - b^2}$
reduzierte Temperatur	$T_r = T / T_{kr}$ 1 0,8854
$\alpha(T) = [1 + (0,37464 + 1,54226\omega - 0,26992\omega^2)(1 - T_r^{0,5})]^2$	$\alpha(T)$ 1 1,0607
$a_{kr} = 0,45724 (R^2 T_{kr}^2) / p_{kr}$	a_{kr} Pa m ⁶ mol ⁻² 0,4998
$b = 0,0778 R T_{kr} / p_{kr}$	b m ³ mol ⁻¹ 3,6224E-05
$a(T) = a_{kr} \alpha(T)$	a Pa m ⁶ mol ⁻² 0,5301
Realgasfaktor Z , variable Zelle Solver	Z 1 0,8083
Zielfunktion $f(Z) = 0$, Zielzelle Solver	$f(Z)$ 1 -4,25E-07
	$f(Z) = Z^3 + Z^2 \left(\frac{bp}{RT} - 1 \right) + Z \left(\frac{ap}{R^2 T^2} - 3 \frac{p^2 b^2}{R^2 T^2} - 2 \frac{bp}{RT} \right) + \frac{p^3 b^3}{R^3 T^3} + \frac{p^2 b^2}{R^2 T^2} - \frac{abp^2}{R^3 T^3} = 0$
Dichte $\rho = p M / (Z R T)$	ρ kg m ⁻³ 26,713
spezifisches Volumen $v = 1 / \rho$	v m ³ kg ⁻¹ 0,03743
molares Volumen $\tilde{V} = v M$	\tilde{V} m ³ kmol ⁻¹ 1,050
Vergleich ideal – real	
relative Abweichung $(\rho_{ideal} - \rho_{real}) / \rho_{real}$	$\Delta \rho / \rho$ 1 -19,2%

Excel_mit_VBA_2015.xlsm Gasdichte ideal real Ethen P3_4 / UF / D 22.04.2016

UnitOperations.de

Abbildung 2.1: Druckansicht des Excel-Berechnungsblattes für die Berechnung der Dichte, des spezifischen sowie des molaren Volumens eines idealen oder realen Gases (siehe dazu Abschnitt 2.5.3)

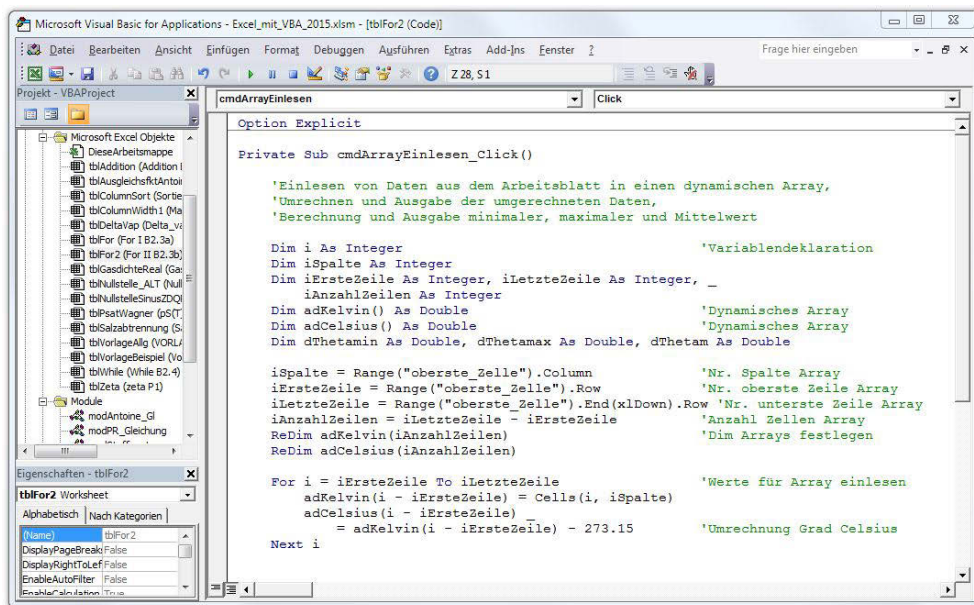


Abbildung 2.2: VBA-Editor mit Eingabebereich und Eigenschaftenfenster

Aufruf des VBA-Editors Um den VBA-Editor aufrufen zu können, siehe Abb. 2.2, muss die Registerkarte **Entwicklertools** in der Hauptansicht angezeigt werden. Dazu wird in der Registerkarte **Datei** **Optionen** **Menüband anpassen** im rechten Fenster unter **Hauptregisterkarten** das Kontrollkästchen **Entwicklertools** aktiviert. Über die nun sichtbare Registerkarte **Entwicklertools** der Hauptansicht kann mit der Schaltfläche **Visual Basic** der VBA-Editor aufgerufen werden. Zum schnellen Wechsel zwischen der Arbeitsmappe und dem VBA-Editor empfiehlt sich die Tastenkombination **[Alt] + [F11]**. Alternativ können – eine entsprechend große Monitoroberfläche vorausgesetzt – beide Fenster nebeneinander angeordnet werden.

Sicherheitseinstellungen in Excel Die Sicherheitseinstellungen für Makros müssen angepasst werden, da diese sonst nicht ausgeführt werden können. Dafür wird unter der Registerkarte **Entwicklertools** **Makrosicherheit** das folgende Kontrollkästchen aktiviert: **Alle Makros mit Benachrichtigung deaktivieren**.

Makros aufzeichnen Um Abläufe in der Arbeitsmappe zu automatisieren oder VBA-Befehle für bestimmte Aktionen auf einfache Art ermitteln zu können, wird der Makrorekorder genutzt, siehe Abschnitt 2.3.2. Nach Betätigen der Schaltfläche **Entwicklertools** **Makro aufzchn.** werden alle Aktionen in der Arbeitsmappe als VBA-Code gespeichert. **Aufzeichnung beenden** schließt die Aufzeichnung ab. Im VBA-Editor kann der aufgezeichnete Code unter der Registerkarte **Projekt-Explorer** **Module** angezeigt und angepasst werden.

Makros ausführen Die zuvor erzeugten Makros können z. B. über die Schaltfläche mit dem grünen Play-Symbol im VBA-Editor ausgeführt werden. Alternativ können Makros auch in Excel entweder manuell über das Dialogfenster unter **Entwicklertools** **Code** **Makros** oder über selbst erstellte Schaltflächen in der Arbeitsmappe (siehe nachfolgend) ausgeführt werden oder automatisch bei Änderungen im Arbeitsblatt.

Prozeduren Prozeduren werden durch Steuerelemente oder Ereignisse gestartet und dienen zur Automatisierung. Eine Prozedur **Sub** ist ein Unterprogramm in einem VBA-Programm. Prozeduren werden nach folgendem Muster in einem Modul erstellt:

```
Sub ProzedurName()  
    <Code>  
End Sub
```

Funktionen Elementarer Bestandteil von Programmen sind Funktionen. Benutzerdefinierte Funktionen erweitern die Excel-Bibliothek, siehe Abschnitt 2.5. Eine Funktion **Function** kann aus einer Prozedur ausgegliedert werden, um die Übersicht zu verbessern oder um die Verwendung in weiteren Prozeduren zu ermöglichen. Funktionen werden nach folgendem Muster in einem Modul erstellt:

```
Function FunktionName(Argument1 As Datentyp,...) As Datentyp  
    <Code>  
    FunktionName = <Ergebnis>  
End Function
```

Die Argumente in der Funktion sind die Variablen, von denen das Ergebnis der Funktion abhängt; der Datentyp am Ende der ersten Zeile ist der Datentyp des Ergebnisses. Die Funktion kann mit **FunktionName(Argument1,...)** mit den eingegebenen Variablen in einer Prozedur verwendet werden. Das Einfügen von Prozeduren und Funktionen erfolgt manuell oder durch Aufrufen des Menüs **Einfügen** **Prozedur einfügen**.

Public und Private Im VBA-Editor unter **Module** im **Projekt-Explorer** gespeicherte Prozeduren und Funktionen sind standardmäßig in der gesamten Arbeitsmappe verfügbar. Prozeduren und Funktionen unter **Microsoft Excel Objekte** sind nur in diesem Objekt verfügbar. Durch Voranstellen des Befehls **Public** (verfügbar in allen Modulen und Arbeitsblättern) oder des Befehls **Private** (verfügbar nur in diesem Objekt) vor **Sub** bzw. **Function** kann die Verfügbarkeit definiert werden.

Datentypen, Option Explicit In VBA sind u. a. die in Tabelle 2.1 aufgeführten Datentypen verwendbar. Standardmäßig kommt der Datentyp **Variant** zur Anwendung. Variablen werden nach dem Schema **Dim VarName [As Datentyp]** deklariert.

Mit der Eingabe **Option Explicit** am Anfang des VBA-Codes wird automatisch vor fehlenden Variablendeklarationen gewarnt (siehe auch nachfolgend). Hinweis: Die Zeile mit der Eingabe **Option Explicit** am Anfang der nachfolgenden VBA-Codes wird aus Platzgründen häufig nicht dargestellt.

Kommentare Zur näheren Beschreibung können im VBA-Code Kommentare eingegeben werden, die immer mit einem Apostroph (') beginnen. Über Kommentare lassen sich auch Code-Abschnitte zur Fehlersuche vorübergehend „ausschalten“.

Punkte, Kommata und Semikola Die Syntax ist in den Arbeitsblättern und im VBA-Editor teilweise unterschiedlich. In der deutschsprachigen Excel-Version werden im Arbeitsblatt Kommata als Dezimaltrennzeichen verwendet und Semikola zur Trennung von Parametern in einer Funktion. Der VBA-Editor versteht dagegen vorwiegend englisch. Daher werden dort Punkte als Dezimaltrennzeichen verwendet und Kommata zur Trennung von Parametern einer Funktion. Im VBA-Editor kommen zudem englische Funktionsnamen zur Anwendung.

Zellen und Zellbereiche Mit den Befehlen **Range** und **Cells** kann aus VBA auf Zellen oder Zellbereiche im Arbeitsblatt zugegriffen werden, z. B. **Range("E42")** oder **Range("A2:E42")**. **Cells** bietet Vorteile beim Programmieren, weil es numerische Werte erwartet und Zeilen- und Spaltennummern mit Variablen gebildet werden können. So lauten die entsprechenden Befehle für diese beiden Beispiele **Cells(42, 5)** (Zeile 42, Spalte 5) und **Range(Cells(2, 1), Cells(42, 5))**. Siehe dazu auch die Beispiele in den Abschnitten 2.3.1 und 2.3.3. Wie nachfolgend erläutert, können Zellen im Arbeitsblatt auch Namen gegeben werden, die dann mit **Range** angesprochen werden können.

Absolute Zellbezüge, Namen Werden in Arbeitsblättern Spalten oder Zeilen eingefügt oder gelöscht, ändern sich automatisch die Bezüge in den Formeln der von der Verschiebung betroffenen Zellen. Wird dagegen in VBA auf eine Zelle oder einen Zellbereich verwiesen und dann im Arbeitsblatt eine Spalte oder Zeile eingefügt oder gelöscht, steht in VBA nach wie vor der ursprüngliche Zellbereich. Dieses Problem kann umgangen werden, wenn Zellen oder Zellbereichen im Arbeitsblatt Namen zugewiesen und diese in VBA verwendet werden:

T = Range("A1").Value

T = Range("Temp").Value

"T" ist der Name einer Variablen in VBA

"Temp" ist der Name einer Zelle im Arbeitsblatt

Um einem Zellbereich – bestehend aus einer oder aus mehreren Zellen – im Arbeitsblatt einen Namen zuzuweisen, wird dieser Bereich markiert und nach einem Klick auf **Neu...**

Tabelle 2.1: Auswahl von Datentypen in VBA

Datentyp	Art	Wertebereich
Boolean	Logische Werte	True oder False
Integer	Ganze Zahlen	−32 768 bis 32 767
Double	Gleitkommazahlen	$-1,80 \cdot 10^{308}$ bis $-4,94 \cdot 10^{-324}$ für negative Werte und $4,94 \cdot 10^{-324}$ bis $1,80 \cdot 10^{308}$ für positive Werte
Date	Datum	01.01.1900 00:00:00 bis 31.12.9999 23:59:59
String	Zeichenketten	Zeichenkette kann 0 bis 63 000 Zeichen umfassen
Variant	Beliebige Daten	

im Namens-Manager unter **Formeln** **Definierte Namen** **Namens-Manager** im Feld **Name** der Name eingegeben. Unter **Bereich** kann gewählt werden, ob der Name für die gesamte Arbeitsmappe oder nur für ein bestimmtes Arbeitsblatt gelten soll. Häufig ist es sinnvoll, einen Namen nur für ein bestimmtes Arbeitsblatt zu vergeben. So besteht die Möglichkeit, diesen Namen auch in anderen Arbeitsblättern definieren zu können. Alternativ kann der Name direkt im **Namenfeld** links neben der **Bearbeitungsleiste** eingegeben werden, gilt dann allerdings für die gesamte Arbeitsmappe.

Makros bei Änderung ausführen Mit dem **Worksheet_Change**-Ereignis kann ein Makro bei Änderungen im Arbeitsblatt automatisch ausgeführt werden. Zur Erstellung wird im VBA-Editor im **Projekt-Explorer** ein Arbeitsblatt ausgewählt, in dem das Makro ausgeführt werden soll. Links über dem Codebereich wird im Dropdown-Menü **(Allgemein)** in **Worksheet** geändert und im Dropdown-Menü rechts daneben **Change** ausgewählt. In die entstandene Prozedur wird der VBA-Code eingefügt, der bei jeder Änderung im Arbeitsblatt ausgeführt werden soll. Die Ausführung auf Änderungen hin kann auf bestimmte Zellen beschränkt sein, z. B. auf die Zelle **A10**, siehe dazu Beispiel 2.6.

```
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Address = "$A$10" Then
        Range("B10").Value = "Test"
    End If
End Sub
```

Mit Änderung ist in diesem Fall die manuelle Änderung durch den Anwender und nicht die automatische Neuberechnung im Arbeitsblatt gemeint. Alternativ dazu ist auch die Ausführung von Makros nach dem Öffnen oder vor dem Speichern möglich.

Makros per Schaltfläche ausführen Die Verwendung von Schaltflächen erleichtert das Aufrufen von Makros. Dazu wird unter der Registerkarte **Entwicklertools** die Schaltfläche **Einfügen** **Befehlsschaltfläche (ActiveX-Steuerelement)** ausgewählt, wobei automatisch der Entwurfsmodus aktiviert ist – siehe dazu die farblich veränderte Schaltfläche **Entwurfsmodus** – und mit dem Cursor im Arbeitsblatt ein Rechteck aufgezogen. Der auszuführende VBA-Code kann entweder nach einem Doppelklick auf die Schaltfläche oder nach einem Rechtsklick darauf und Wahl von **Code anzeigen** an der aktuellen Cursorposition eingegeben werden.

Die Bearbeitung der Schaltfläche ist nur möglich, wenn der **Entwurfsmodus** auf der Registerkarte **Entwicklertools** aktiviert ist, wodurch die Ausführung von VBA-Code verhindert wird. Eine Änderung der Beschriftung der Schaltfläche ist nach Rechtsklick und Aktivieren von **Eigenschaften** unter **Caption** möglich, eine Änderung der Farbe unter **ForeColor** und **BackColor**. Der **Name** (Code-Name dieser Schaltfläche) wird unter **Eigenschaften** in **cmdTest** und nach Rechtsklick auf die Schaltfläche und Aktivieren von **Code anzeigen** ebenfalls in **cmdTest** verändert und abschließend im Arbeitsblatt unter der Registerkarte **Entwicklertools** der Entwurfsmodus wieder deaktiviert. Abschnitt 2.3 enthält mehrere Beispiele, die per Schaltfläche ausgeführt werden.

```
Private Sub cmdTest_Click()  
    Range("B10").Value = "Test"  
End Sub
```

Namenskonventionen: VBA-Objekte sollten mit einem einheitlichen Präfix benannt werden, um den Typ des Objekts einfacher erkennen zu können. Das Präfix „cmd“ steht beispielsweise für „Command Button“ (Befehlsschaltfläche).²

Schutzfunktionen Grundsätzlich ist es in Excel möglich, Zellen, Arbeitsblätter und Arbeitsmappen gegen Änderungen zu schützen und VBA-Code mit einem Passwortschutz zu verbergen. Die Schutzfunktionen können jedoch leicht von Tools zur Suche nach „vergessenen“ Passwörtern aufgehoben werden, sodass sie keine Sicherheit bieten. Ein sicherer Schutz ist über COM-Add-Ins möglich. Alternativ sind ggf. auch kommerzielle Programme wie LockXLS oder DoneEx in Betracht zu ziehen.

Sinnvoll ist der Schutz von Arbeitsblättern als Sicherheit gegen das versehentliche Überschreiben von Zellen. Dafür werden die Zellen markiert, welche nicht geschützt werden sollen. Nach einem Rechtsklick auf **Zellen Formatieren** **Schutz** wird der Haken bei **Gesperrt** entfernt und anschließend **Überprüfen** **Änderungen** **Blatt schützen** **OK** der Blattschutz aktiviert. Das Passwort ist optional.

Optionen und Einstellungen im VBA-Editor Bei der Arbeit mit VBA sind eine Reihe von Einstellungen sinnvoll, die im VBA-Editor unter **Extras** **Optionen** eingestellt werden:

- Unter **Editor** sollte die **Automatische Syntaxüberprüfung** deaktiviert werden. Die Syntaxüberprüfung wird weiterhin durchgeführt, jedoch unterbleibt die lästige Fehlermeldung.
- Ebenfalls unter **Editor** ist **Variablendeklaration erforderlich** zu aktivieren, wodurch an den Anfang des Codes automatisch die Zeile **Option Explicit** gesetzt wird und die explizite Deklaration der Variablen verlangt und nicht vergessen wird.
- Unter **Editor** sollte die **Tab-Schrittweite** auf einen sinnvollen Wert gesetzt werden, z. B. auf 2 oder 4.
- Unter **Editorformat** können Schriftart und Code-Farben festgelegt werden.
- Unter **Allgemein** ist unter **Kompilieren** der Punkt **Bei Bedarf** zu deaktivieren, damit immer der gesamte Code in der Arbeitsmappe und nicht nur die aufgerufene Prozedur oder Funktion überprüft wird.

Objektkatalog Im VBA-Editor wird unter **Ansicht** **Objektkatalog** oder mit der Taste **F2** der Objektkatalog aufgerufen. Der Objektkatalog enthält alle verfügbaren Objekte, die nach Bibliotheken geordnet sind – einschließlich der selbst definierten Funktionen insbesondere zu „Excel“, „VBA“ und mit „VBAProject“ zur geöffneten Arbeitsmappe.

²Zu den Namenskonventionen siehe auch http://de.wikibooks.org/w/index.php?title=VBA_in_Excel/_Namenskonventionen&oldid=699390.

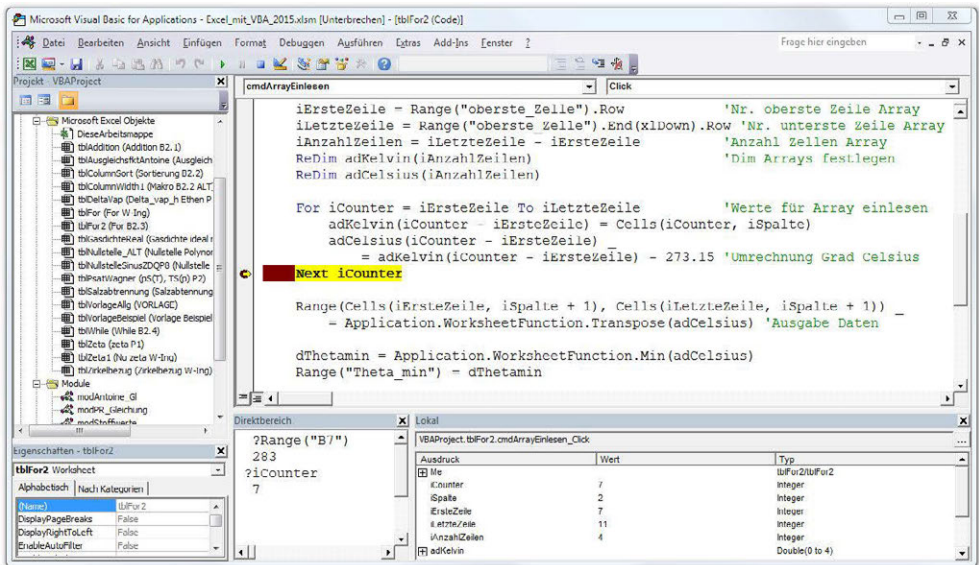


Abbildung 2.3: VBA-Editor mit Direktbereich und Lokalfenster

Direktbereich, Lokalfenster, Haltepunkte, Debuggen Der *Direktbereich* als Hilfsmittel zur Fehlersuche wird im VBA-Editor unter **Ansicht** **>** **Direktfenster** aufgerufen. Dort können u. a. Zellinhalte oder Inhalte von Variablen angezeigt werden, indem z. B. `?Range("B7")` oder `?iCounter` (für die Variable `iCounter`) im Direktbereich eingegeben und mit der **Enter**-Taste bestätigt wird, siehe Abb. 2.3. Zudem können Anweisungen ausgeführt werden, siehe dazu [45]. Mit dem Befehl **Debug.Print** kann beim Ausführen von Code innerhalb einer Prozedur oder Funktion auch in den Direktbereich geschrieben werden, siehe dazu die Beschreibungen in Abschnitt 2.8.

Im *Lokalfenster*, aufzurufen unter **Ansicht** **>** **Lokal-Fenster**, lassen sich die Inhalte und die Typen von Variablen anzeigen, siehe Abb. 2.3. Dazu werden am besten *Haltepunkte* verwendet, die sich durch Klicken direkt links vor dem Code auf die Leiste setzen und auch so wieder entfernen lassen, siehe die rotbraune Markierung in Abb. 2.3. Die Ausführung des Codes erfolgt bis zu diesem Haltepunkt, deutlich erkennbar an der zusätzlich erscheinenden gelben Markierung. Im Lokalfenster sind die Informationen zu den Variablen zu sehen. Durch Drücken von **F5** wird die Ausführung des Codes fortgesetzt.

Alternativ werden die aktuellen Werte der Variablen auch direkt im Editor als Tool-Tips angezeigt, wenn der Mauszeiger über die Variablen geführt wird (ohne Klicken). Diese Methoden eignen sich insbesondere zur Fehlersuche im Code, dem sogenannten *Debuggen*. Detailliertere Informationen dazu finden sich in der empfohlenen Literatur.

Digitale Signaturen Mit digitalen Signaturen können VBA-Projekte signiert werden, um ihnen anschließend „vertrauen zu können“. Dadurch unterbleibt – vor allem bei eigenen Projekten – die lästige Sicherheitswarnung, dass die Makros deakti-

Massenstrom 1	m_1	kg h ⁻¹	21	Addition ausführen
Massenstrom 2	m_2	kg h ⁻¹	21	
Massenstrom Summe	m_Σ	kg h ⁻¹	42	

Abbildung 2.4: Addition der Werte zweier Massenströme in VBA über eine Schaltfläche

viert wurden. Einfache digitale (Code-)Zertifikate können unter **Start** » **Alle Programme** » **Microsoft Office 2010** » **Microsoft Office 2010-Tools** » **Digitales Zertifikat für VBA-Projekte** erstellt werden. Dabei sollte dem Zertifikat ein aussagekräftiger Name gegeben werden wie z. B. <Vorname Nachname (Firmenname)>. Das Zertifikat wird im „Zertifikatsmanager“ (certmgr.msc) unter den „Eigenen Zertifikaten“ gespeichert.

Anschließend kann das Zertifikat einem eigenen Projekt im VBA-Editor unter **Extras** » **Digitale Signatur ...** » **Wählen** und der Auswahl des erstellten Zertifikates hinzugefügt werden. Beim nächsten Öffnen der Datei kann in den Signaturdetails „Allen Dokumenten von diesem Herausgeber vertrauen“ ausgewählt und bestätigt werden.

Wollen Sie Zertifikaten Fremder vertrauen, klicken Sie bei der Sicherheitswarnung auf **Makros wurden deaktiviert.** » **Inhalt aktivieren** » **Erweiterte Optionen** » **Signaturdetails anzeigen** » **Zertifikat anzeigen** » **Zertifikat installieren...**. Es ist wichtig, im „Zertifikatimport-Assistent“ nach einem Klick auf **Weiter** den Zertifikatspeicher nicht automatisch zu wählen, sondern diesen durch Klicken auf **Alle Zertifikate in folgendem Speicher speichern** » **Durchsuchen...** » **Vertrauenswürdige Stammzertifizierungsstellen** » **Fertig stellen** manuell zuzuweisen. Abschließend muss die Installation des Zertifikates bestätigt werden. Daraufhin öffnet sich erneut der Anfangsdialog (bei dem **Signaturdetails anzeigen** geklickt wurde), für den die Option **Allen Dokumenten von diesem Herausgeber vertrauen** hinzugekommen ist, die abschließend ausgewählt und bestätigt wird.

2.3 Anwendungsbeispiele mit VBA

In den nachfolgenden Abschnitten sollen die erarbeiteten Grundlagen angewendet werden. Wir starten mit anfangs „neutralen“ Beispielen, die zunehmend „verfahrenstechnischen“ Charakter bekommen. Es empfiehlt sich, die Beispiele in der vorgeschlagenen Reihenfolge nachzuarbeiten und dazu *eine* Arbeitsmappe mit mehreren Arbeitsblättern – auf Basis der in Abschnitt 2.1 erstellten Vorlage – anzulegen.

2.3.1 Addition über eine Schaltfläche

Beispiel 2.1

Im Arbeitsblatt sollen die Werte zweier Massenströme m_1 und m_2 eingegeben, die Summe bei der Zusammenführung der Ströme m_Σ in VBA berechnet, im Arbeitsblatt ausgegeben und der Ablauf über eine Schaltfläche gestartet werden. (Ergebnis im Excel-Berechnungsblatt in Abb. 2.4.)

1. Schaltfläche einfügen: Siehe Anleitung in Abschnitt 2.2 „Makros per Schaltfläche ausführen“.
2. Schaltfläche umbenennen: Zum Umbenennen der Schaltfläche wird ebenfalls wie in Abschnitt 2.2 beschrieben vorgegangen. In diesem Beispiel sollen bei `Name` „cmdAddition“ und `Caption` „Addition ausführen“ eingetragen werden, um den Unterschied zu verdeutlichen. `Name` und `Caption` müssen nicht unterschiedlich, sollten aber eindeutig sein. Zur Verwendung des Präfix „cmd“ siehe die Anmerkungen in Abschnitt 2.2.
3. Code anzeigen: Rechtsklick auf die erstellte Schaltfläche und den Menüpunkt `Code anzeigen` wählen. In einem neuen Fenster öffnet sich der VBA-Editor:

```
Private Sub cmdAddition_Click()

End Sub
```

4. Den Zellen im Arbeitsblatt mit den Zahlen Namen zuweisen (siehe dazu Abschnitt 2.2), VBA-Code vervollständigen: Die Zahlen aus den Zellen `m_1` und `m_2` sollen `addiert` und das Ergebnis in Zelle `m_Summe` geschrieben werden:

Listing 2.1: VBA-Code für die Addition über eine Schaltfläche

```
Private Sub cmdAddition_Click() 'Sub nur im Arbeitsblatt verfügbar

    Dim dMass1 As Double      'Variablendeklaration
    Dim dMass2 As Double
    Dim dMassSum As Double

    dMass1 = Range("m_1")      'Wert aus Zelle "m_1" in Variable 'dMass1'
    dMass2 = Range("m_2")      'Wert aus Zelle "m_2" in Variable 'dMass2'
    dMassSum = dMass1 + dMass2 'Werte addieren und in 'dMassSum' speichern
    Range("m_Summe") = dMassSum 'Wert von 'dMassSum' in Zelle "m_Summe" schreiben

End Sub
```

Auch hier wurden Präfixe vergeben, u. a. „d“ für die Kennzeichnung der verwendeten Variablen („d“ für Datentyp `Double`), siehe auch Abschnitt 2.2.

5. VBA-Editor schließen: Der VBA-Editor wird geschlossen und die Datei gespeichert. Falls die Arbeitsmappe zuvor noch nicht mit Makros gespeichert wurde, erfolgt die Aufforderung dazu. Im sich öffnenden Dialog ist der Dateityp `*.xlsm` zu wählen. Um das Makro über das Klicken der Schaltfläche aufrufen zu können, muss der Entwurfsmodus deaktiviert sein.

d_i / m	0,10530
\dot{V}	$w = \dot{V} / A$
$\text{m}^3 \text{h}^{-1}$	m s^{-1}
0,0	0,00
10,0	0,32
20,0	0,64
30,0	0,96
40,0	1,28
50,0	1,59
51,2	1,63
60,0	1,91
70,0	2,23

Abbildung 2.5: Datentabelle für die Anwendung einer Sortierung, vgl. Abb. 10.15

2.3.2 Makros aufzeichnen und Befehle daraus verwenden

Beispiel 2.2

Der VBA-Befehl für die automatische zeilenweise Sortierung der Tabelle in Abb. 10.15 soll gefunden und mittels einer Schaltfläche angewendet werden. Die Sortierung soll aufsteigend nach der Spalte A, also den Werten für den Volumenstrom \dot{V} , erfolgen. (Ergebnis im Excel-Berechnungsblatt in Abb. 2.5.)

Der Einfachheit halber erstellen wir hier eine vereinfachte Version der Tabelle in Abb. 10.15, die nur zwei Spalten enthält, siehe Abb. 2.5. Die Werte für den Volumenstrom \dot{V} in der ersten Spalte werden vorgegeben. Der Wert $\dot{V} = 51,2 \text{ m}^3 \text{h}^{-1}$ und damit die rot formatierte Zeile werden *nicht* passend einsortiert, sondern z. B. in die dritte Zeile der Tabelle geschrieben. Die Strömungsgeschwindigkeit w ergibt sich mit Gleichung (10.2) zu $w = \dot{V} / A$, wobei die Querschnittsfläche für die Rohrleitung mit $A = \pi d_i^2 / 4$ (unter Verwendung der Funktion `PI()` für die Kreiszahl π) berechnet wird.

1. Der Bereich mit den Zahlenwerten in den *beiden Spalten* der Tabelle bekommt den Namen **TabelleWertebereich** und die Spalte mit den Werten der Volumenströme den Namen **Spalte_V**, siehe dazu Abschnitt 2.2.
2. Makro aufzeichnen: Die Aufzeichnung des Makros wird über **Entwicklertools** **Code** **Makro aufzeichnen** gestartet. Zunächst werden die beiden Spalten mit den Zahlenwerten markiert, die die Werte für die Volumenströme und Strömungsgeschwindigkeiten enthalten. Unter der Registerkarte **Daten** **Sortieren** wird nach der Spalte A nach Werten mit aufsteigender Reihenfolge sortiert. Aufzeichnung beenden: **Entwicklertools** **Code** **Aufzeichnung beenden**.
3. Schaltfläche einfügen, umbenennen, Code anzeigen: siehe Abschnitt 2.3.1.
4. Code kopieren: Im linken Fenster des VBA-Editors wird unter **Module** durch einen Doppelklick **Modul1** (oder das entsprechende Modul) ausgewählt. Der gesamte Code wird kopiert und der Schaltfläche zugeordnet, indem der Code der Schaltfläche durch einen Doppelklick im VBA-Editor auf **Tabelle1** (oder den

entsprechenden Namen des Arbeitsblattes) im linken Fenster aufgerufen und zwischen den bereits vorhandenen Zeilen eingefügt wird. Es bietet sich an, den Code wie folgt umzusortieren, zu vereinfachen und zu ergänzen:

Listing 2.2: VBA-Code für die Sortierung von Daten über eine Schaltfläche

```

Private Sub cmdTabelleSortieren_Click()

    With ActiveWorkbook.Worksheets("Sortierung").Sort
        .SortFields.Clear
5       .SortFields.Add Key:=Range("Spalte_V"), _
            SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
        .SetRange Range("TabelleWertebereich")
        .Header = xlNo
        .MatchCase = False
10      .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
15 End Sub

```

5. Speichern und VBA-Editor schließen, Makro testen.

2.3.3 Berechnungen mit Schleifen

Sollen Anweisungen mehrfach hintereinander durchgeführt werden, bietet sich die Verwendung von Schleifen an. Es gibt Zählschleifen (wie die **For**-Schleife) und prüfende Schleifen (wie die **Do-While**-Schleife). Eine Anwendung für die **For**-Schleife ist das Beispiel 2.3 oder auch die Nullstellensuche in Beispiel 2.13. Eine Anwendung für die **Do-While**-Schleife ist das Beispiel 2.4.

For-Schleife

Beispiel 2.3

Aus einer veränderbaren Anzahl von im Arbeitsblatt in einer Spalte vorgegebenen Werten für Temperaturen in Kelvin sollen die entsprechenden Temperaturen in °C berechnet und das Makro über eine Schaltfläche gestartet werden. Optional ist die Ermittlung der minimalen und der maximalen Celsius-Temperatur sowie des arithmetischen Mittelwertes durchzuführen. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.6.)

1. Tabellenkopf erstellen und Daten wie z. B. in Abb. 2.6 für die Temperatur in K eingeben. Schaltfläche einfügen, umbenennen, Code anzeigen: siehe Abschnitt 2.3.1.
2. **For**-Schleifen sind nach dem Schema

i	T_i / K	$\vartheta_i / ^\circ\text{C}$	$\vartheta_{\min} / ^\circ\text{C}$	$\vartheta_{\max} / ^\circ\text{C}$	$\vartheta_m / ^\circ\text{C}$	Berechnung
1	283,0	9,9	9,9	13,9	11,9	
2	284,0	10,9				
3	285,0	11,9				
4	286,0	12,9				
5	287,0	13,9				
6						
7						
8						

Abbildung 2.6: Excel-Berechnungsblatt für die Umrechnung von Temperaturen mit einem dynamischen Array

```

For iCounter = iStart To iEnd Step iStep
    <Code>
Next iCounter

```

aufgebaut. Ausgehend von einem Startwert wird bis zu einem Endwert gezählt und dabei ein Code ausgeführt. `iCounter` ist die Laufvariable, `iStart` ihr Startwert und `iEnd` ihr Endwert. `iStep` ist die Schrittweite und gibt an, um welchen Wert `iCounter` bei jedem Durchlauf erhöht (bzw. bei einem negativen Wert erniedrigt) wird. Wird `Step iStep` nicht angegeben, erhöht sich `iCounter` um den Wert 1.

- Die Werte für die Temperaturen werden in Arrays verarbeitet. Das sind z. B. Vektoren, Matrizen oder allgemein n -dimensionale Felder. Mit der Definition `Dim adVektor(100) as Double` wird ein Vektor mit 101 Zeilen definiert, da VBA mit dem Index 0 beginnt. Alternativ sind auch andere Datentypen möglich. Unser Beispiel erfordert ein dynamisches Array `Dim adVektor() as Double` (mit leerem Klammerausdruck), bei dem die Größe im Programmablauf mit dem Befehl `ReDim` festgelegt oder verändert werden kann. Der Code für die Aufgabenstellung lautet:

Listing 2.3: VBA-Code für Umrechnung von Temperaturen mit einem dynamischen Array

```

Private Sub cmdArrayForSchleife_Click()

    'Einlesen von Daten aus dem Arbeitsblatt in einen dynamischen Array,
    'Umrechnen und Ausgabe der umgerechneten Daten,
    'Berechnung und Ausgabe minimaler, maximaler und Mittelwert

    Dim iZaehler As Integer                                'Variablendeklaration
    Dim iSpalte As Integer
    Dim iErsteZeile As Integer, iLetzteZeile As Integer, _
    iAnzahlZeilen As Integer
    Dim adKelvin() As Double                                'Dynamisches Array
    Dim adCelsius() As Double                              'Dynamisches Array
    Dim dThetamin As Double, dThetamax As Double, dThetam As Double

    iSpalte = Range("oberste_Zelle").Column                'Nr. Spalte Array
    iErsteZeile = Range("oberste_Zelle").Row                'Nr. oberste Zeile Array

```

```

iLetzteZeile = Range("oberste_Zelle").End(xlDown).Row 'Nr. unterste Zeile Array
iAnzahlZeilen = iLetzteZeile - iErsteZeile           'Anzahl Zellen Array
ReDim adKelvin(iAnzahlZeilen)                       'Dim Arrays festlegen
ReDim adCelsius(iAnzahlZeilen)

20
For iZaehler = iErsteZeile To iLetzteZeile           'Werte für Array einlesen
    adKelvin(iZaehler - iErsteZeile) = Cells(iZaehler, iSpalte)
    adCelsius(iZaehler - iErsteZeile) = _
25        = adKelvin(iZaehler - iErsteZeile) - 273.15 'Umrechnung Grad Celsius
Next iZaehler

Range(Cells(iErsteZeile, iSpalte + 1), Cells(iLetzteZeile, iSpalte + 1)) _
    = Application.WorksheetFunction.Transpose(adCelsius) 'Ausgabe Daten
30

dThetamin = Application.WorksheetFunction.Min(adCelsius)
Range("Theta_min") = dThetamin
dThetamax = Application.WorksheetFunction.Max(adCelsius)
Range("Theta_max") = dThetamax
35
dThetam = Application.WorksheetFunction.Average(adCelsius)
Range("Theta_m") = dThetam

End Sub

```

4. Zunächst erfolgt die Deklaration der Variablen: der Laufvariablen **iZaehler**, der Variablen **iSpalte** (für die Nummer der Spalte im Arbeitsblatt) und den Variablen **iErsteZeile**, **iLetzteZeile**, **iAnzahlZellen** für die Nummern der obersten und der untersten Zellen sowie der Gesamtzahl der Zellen des Arrays. Danach werden die dynamischen Arrays **adKelvin()** und **adCelsius()** für die Temperaturfelder und die Variablen **dThetamin**, **dThetamax** und **dThetam** definiert.
5. Dem Element in der obersten Zelle des Temperaturfeldes wird im Arbeitsblatt der Name **oberste_Zelle** zugewiesen. Die Spaltennummer des Temperaturfeldes wird ermittelt und der Variablen **iSpalte** zugewiesen. Außerdem werden die Nummern der obersten und der untersten Zellen dieses Arrays an die Variablen **iErsteZeile** und **iLetzteZeile** sowie die Anzahl der Zellen des Arrays an die Variable **iAnzahlZellen** übergeben und die Größe der Arrays für die Temperaturfelder anhand der Anzahl der Zellen im Arbeitsblatt mit Temperaturwerten festgelegt.
6. In der **For**-Schleife werden an **adKelvin** die Werte aus dem Arbeitsblatt mittels **Cells**-Befehl übergeben, die Elemente des Arrays umgerechnet und an **adCelsius** gereicht. Anschließend erfolgt mit einem gekoppeltem **Range-Cells**-Befehl die Übergabe der Elemente an das Arbeitsblatt. Dabei ist zu beachten, dass eindimensionale Arrays aus VBA für eine zeilenweise Übergabe in das Arbeitsblatt transponiert werden müssen, deshalb der Befehl **Transpose**.
7. Abschließend erfolgt für **arKelvin** die Berechnung und Ausgabe der minimalen, der maximalen und der arithmetisch gemittelten Temperaturen **dThetamin**, **dThetamax** bzw. **dThetam** in die mit Namen zu versehenen Zellen im Arbeitsblatt.
8. Speichern und VBA-Editor schließen, Makro testen.

Do-While-Schleife

Beispiel 2.4

Die EULERSche Zahl e , die als Reihe dargestellt werden kann

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots, \quad (2.1)$$

soll mit einer **Do-While**-Schleife berechnet werden (mit $0! = 1$ per Definition). Die Berechnung soll durch eine Schaltfläche gestartet und das Ergebnis im Arbeitsblatt ausgegeben werden.

1. Schaltfläche einfügen, umbenennen, Code anzeigen: siehe Abschnitt 2.3.1.
2. Code einfügen: Zur Berechnung bietet sich hier eine **Do-While**-Schleife an, die nach dem Schema

```
Do While <Bedingung>
    <Code>
Loop
```

aufgebaut ist. Solange die Bedingung „wahr“ ist, wird der Code ausgeführt. Der Code für die Aufgabenstellung lautet:

Listing 2.4: VBA-Code für die Berechnung der Zahl e in einer Schleife

```
Private Sub cmdBerechnungE_Click()
    Range("Zahl_e") = e() 'Wert der Funktion e() in Zelle Zahl_e schreiben
End Sub

5 Function e() As Double 'Funktion ohne Argument

    Dim n As Integer 'Variablendeklaration
    Dim dDiff As Double
    Dim dSum As Double
10 Const cdAbbruchkrit As Double = 10 ^ (-14) 'Abbruchkriterium als Konstante

    n = 1 'Zuweisung Startwerte
    dDiff = 1
    dSum = 1

15 Do While dDiff > cdAbbruchkrit
        dDiff = dDiff / n
        dSum = dSum + dDiff
        n = n + 1
20 Loop

    e = dSum 'Zuweisung berechneter Funktionswert

End Function
```

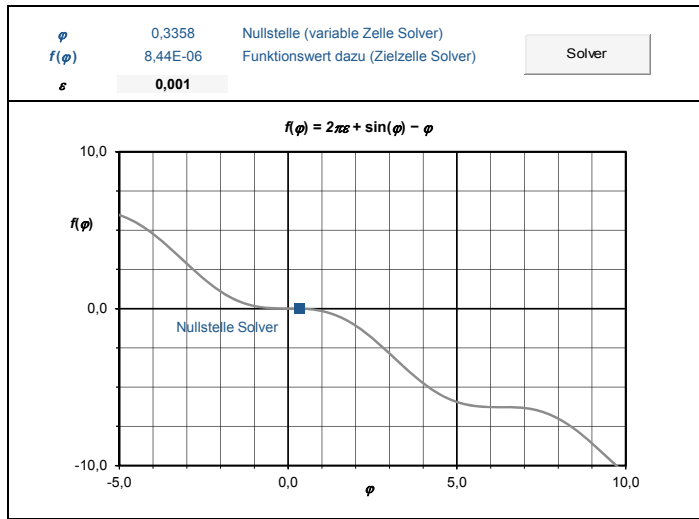


Abbildung 2.7: Nullstellensuche für eine Funktion unter Anwendung des Solvers

3. Speichern und VBA-Editor schließen, Makro testen.

In diesem Code wird mit Aufruf der Prozedur `cmdBerechnungE_Click()` die Funktion `e()` gestartet, in der die Zahl e berechnet und ihr Wert der Zelle `Zahl_e` im Arbeitsblatt übergeben wird.

Alternativ kann auch ein modifizierter Code unter Verwendung der Excel-Funktion für die Berechnung der Fakultäten verwendet werden. Der entsprechende Befehl im Arbeitsblatt lautet `FAKULTÄT(<Zahl>)`. Da VBA aber kein Deutsch versteht und den Hinweis erhalten muss, dass es sich um einen Befehl der Arbeitsmappe handelt, wird daraus in VBA der Befehl `Application.WorksheetFunction.Fact(<Zahl>)`.

2.4 Anwendung des Solvers

Der Solver ist ein leistungsfähiges Werkzeug zur Zielwertsuche und zum Lösen von Optimierungsproblemen oder mathematischen Gleichungen, z. B. zur Nullstellensuche. Der Solver ist ein Add-In und im Lieferumfang von Excel enthalten. In Abschnitt 2.8 wird alternativ zu der Vorgehensweise mit dem Solver das ZDQ-Verfahren zur Nullstellensuche vorgestellt.

Beispiel 2.5

Der Solver soll über eine Schaltfläche aufgerufen und die Nullstelle der Funktion $f(\varphi) = 2\pi\varepsilon + \sin(\varphi) - \varphi$ für $\varepsilon = 0,001$ bestimmt werden. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.7.)

Physikalischer Hintergrund für die Aufgabenstellung: Bei der Auslegung von horizontalen oder wenig geneigten Verdampferrohren nach *VDI-Wärmeatlas* ist die Strömungsform der Zweiphasenströmung in den Rohren zu ermitteln. Dafür ist die Nullstelle der im Beispiel genannten Funktion erforderlich (φ ist der unbenetzte Winkel der durchströmten Rohre im Bogenmaß, ε der Dampfvolumenanteil).³

1. x -Wert vorgeben: Die Zelle, in der im Arbeitsblatt der Startwert der Nullstelle von φ vorgegeben wird, bekommt den Namen **Nullstelle**, die Zelle $f(\varphi)$ den Namen **Funktionswert**. In einer weiteren Zelle mit den Namen **eps** wird der veränderbare Wert von ε eingegeben.
2. Funktionswert $f(\varphi)$ berechnen:

`=2 * PI() * eps + SIN(Nullstelle) - Nullstelle`

3. Solver installieren: Falls noch nicht geschehen, muss der Solver installiert werden. Dies erfolgt unter **Datei** **Optionen** **Add-Ins** **Gehe zu** durch Aktivieren des Kontrollkästchens für den **Solver**.
4. Makro unter Verwendung des Solvers aufzeichnen: Die Aufzeichnung wird gestartet (siehe Abschnitt 2.3.2) und der Solver unter **Daten** **Analyse** **Solver** aufgerufen. Als Zielzelle wird **Funktionswert**, als Zielwert 0 und als Variablenzelle **Nullstelle** ausgewählt und die iterative Berechnung anschließend mit der Schaltfläche **Lösen** gestartet. Die erscheinende Meldung kann bestätigt und die Aufzeichnung beendet werden.
5. Schaltfläche einfügen, umbenennen, Code anzeigen: siehe Abschnitt 2.3.1.
6. Code kopieren und einfügen: siehe Abschnitt 2.3.2. Das Ergebnis sollte wie folgt aussehen:

```
Private Sub cmdSolverNullstelle_Click()
    SolverOk SetCell:="$B$12", MaxMinVal:=3, ValueOf:=0, ByChange:="$B$11", _
        Engine:=1, EngineDesc:="GRG Nonlinear"
    SolverSolve
5 End Sub
```

Besser ist die nachfolgende Variante mit absoluten Zellbezügen, in der der Zelle **B11** im Arbeitsblatt der Name **Nullstelle** und der Zelle **B12** der Name **Funktionswert** gegeben und der VBA-Code entsprechend angepasst wird:

```
Private Sub cmdSolverNullstelle_Click()
    SolverOk SetCell:="Funktionswert", MaxMinVal:=3, ValueOf:=0, _
        ByChange:="Nullstelle", Engine:=1, _
        EngineDesc:="GRG Nonlinear"
5 SolverSolve
End Sub
```

³Siehe dazu *VDI-Wärmeatlas*, Abschnitt H3.1 „Strömungsformen in Verdampferrohren“, S. 901, Gl. (27) [91, 92]

Zeilenumbrüche im VBA-Code lassen sich mit einem Unterstrich „_“ am Zeilenende (mit einem Leerzeichen davor) realisieren. Informationen zum Solver sind auf der Webseite von Frontline Systems zu finden.⁴

7. Verweis auf Solver in VBA einrichten: Um den Code nutzen zu können, ist im VBA-Editor unter **Extras** > **Verweise** der **Solver** zu aktivieren. Im **Projekt-Explorer** (im linken Fenster des VBA-Editors) wird der Verweis unter dem Punkt **Verweise** angezeigt.
8. Speichern und VBA-Editor schließen, Makro testen.

Durch Anfügen des Befehls **True** direkt nach **SolverSolve** wird das Ergebnis des Solvers automatisch bestätigt, was aber nur eingerichtet werden sollte, wenn sicher ist, dass der Solver konvergiert.

2.5 Anwendung benutzerdefinierter Funktionen

Für häufig vorkommende Gleichungen bietet es sich an, benutzerdefinierte Funktionen (UDFs) in Excel zu erstellen, um Berechnungen zukünftig schneller durchführen zu können.

2.5.1 Berechnung des Widerstandsbeiwerts

Beispiel 2.6

Der Widerstandsbeiwert ζ ist für den Fall der ausgebildeten turbulenten Durchströmung eines Rohres mit der im *VDI-Wärmeatlas* gegebenen Gleichung nach KONAKOV in Abhängigkeit von der REYNOLDS-Zahl Re zu berechnen. Dafür ist eine benutzerdefinierte Funktion zu erstellen.

Um – wie im *VDI-Wärmeatlas* vorgegeben – den Wärmeübergangskoeffizienten in-
nendurchströmter Rohre bei einer voll ausgebildeten turbulenten Strömung⁵ berechnen zu können, ist die vorherige Berechnung des Widerstandsbeiwerts

$$\zeta = (1,8 \lg Re - 1,5)^{-2} \quad (2.2)$$

nach KONAKOV erforderlich [91, 92]. Mit dieser Gleichung kann der Druckverlust bei der Durchströmung technisch glatter Rohre im Bereich $10^4 \leq Re \leq 10^6$ ermittelt werden, siehe dazu auch Abschnitt 10.4.

1. Die Zelle, in der im Arbeitsblatt der Wert der REYNOLDS-Zahl eingegeben wird, bekommt den Namen **Re**.

⁴<http://www.solver.com/content/basic-solver-solveroptions-function>

⁵Siehe dazu *VDI-Wärmeatlas*, Abschnitt H1 „Wärmeübertragung bei turbulenter Strömung durch Rohre“, S. 788, Gl. (27) [91, 92]

2. VBA-Editor aufrufen, neues Modul einfügen: Benutzerdefinierte Funktionen werden in ein Modul eingetragen. Um ein Modul unter den Objekten der Arbeitsmappe einzufügen, wird in der linken Spalte des VBA-Editors (Projekt-Explorer) nach einem Rechtsklick **Einfügen** **Modul** gewählt (analog in oberer Menüleiste).
3. Neues Modul umbenennen: Das Modul sollte einen aussagekräftigen Namen bekommen (z. B. „modZeta“). Dafür wird das Modul durch Anklicken markiert, in der Menüleiste die Schaltfläche **Eigenschaftenfenster** (alternativ **F4**-Taste) gewählt und der Name angepasst.
4. Im Arbeitsblatt ist der dekadische Logarithmus und der Logarithmus zu einer Basis verfügbar, in VBA nur der natürliche Logarithmus. Mit dem Befehl **Application.WorksheetFunction.Log10(Re)** kann der dekadische Logarithmus auch in VBA aufgerufen werden, siehe dazu die entsprechende benutzerdefinierte Funktion im nachfolgenden Code:

Listing 2.5: VBA-Code für die Berechnung des Widerstandsbeiwerts

```
Function zeta_Re(Re As Double) As Double
    zeta_Re = (1.8 * Application.WorksheetFunction.Log10(Re) - 1.5) ^ -2
End Function
```

Die erstellte UDF kann nun in der gesamten Arbeitsmappe eingesetzt werden. Ihr Aufruf erfolgt über die Befehlsschaltfläche **Funktion einfügen** links neben der **Bearbeitungsleiste** in der Kategorie **Benutzerdefiniert**. Zur Erstellung einer Kurzbeschreibung, die bei Aufruf dieser Funktion und Eingabe der Parameter erscheint, siehe Abschnitt 2.5.5.

5. Optional kann eine „Sicherheitsabfrage“ bereits bei der Eingabe der REYNOLDS-Zahl erfolgen, damit der Gültigkeitsbereich von Gleichung (2.2) zwingend eingehalten wird. Dazu wird die Zelle für die Eingabe von Re angeklickt und unter **Daten** **Datentools** **Datenüberprüfung** **Datenüberprüfung** die Eingabe von Dezimalzahlen sowie von $10^4 \leq Re \leq 10^6$ zugelassen.
6. Speichern und VBA-Editor schließen, Funktion testen.

2.5.2 Berechnung des Sättigungsdampfdrucks und der Sättigungstemperatur

Beispiel 2.7

Für Ethen ist ein Excel-Berechnungsmodul zu erstellen, mit dem – bei vorgegebener Temperatur (z. B. -42°C) – der Sättigungsdampfdruck $p_s(T)$ und – bei vorgegebenem Druck (z. B. 2,0 bar) – die Sättigungstemperatur $T_s(p)$ mit der WAGNER-Gleichung aus dem *VDI-Wärmeatlas* berechnet werden können. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.8.)

Die Berechnung von Sättigungsdampfdrücken oder Dampf-Sättigungspartialdrücken kann mit der ANTOINE- oder der WAGNER-Gleichung durchgeführt werden. In der

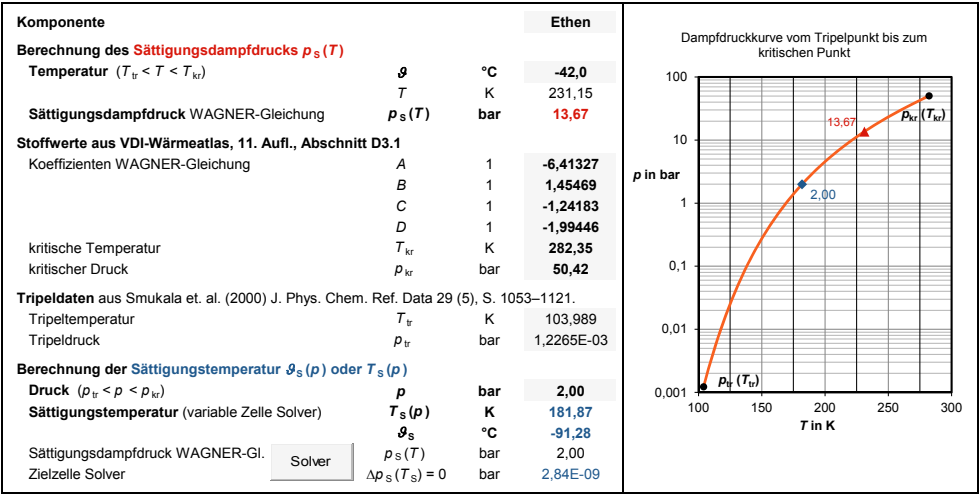


Abbildung 2.8: Excel-Berechnungsblatt für die Berechnung des Sättigungsdampfdrucks und der Sättigungstemperatur

ANTOINE-Gleichung

$$\lg \left(\frac{p_s(T)}{\text{hPa}} \right) = A + \frac{B}{T/\text{K} + C} \tag{2.3}$$

sind die Parameter A , B und C dimensionslos; ihre Werte hängen jedoch von den Einheiten des Druckes und der Temperatur ab. Es existieren unterschiedliche Formen oder Erweiterungen dieser Gleichung [64]. Nachteil ist ihr relativ enger Gültigkeitsbereich.

Die WAGNER-Gleichung ist dagegen in der Lage, den Bereich von der Tripeltemperatur T_{tr} bis zur kritischen Temperatur T_{kr} mit trotzdem hoher Genauigkeit zu beschreiben. Deshalb ist diese Gleichung für viele Anwendungen die bessere Wahl. Diesen Gleichungstyp gibt es mit vier bis sechs Koeffizienten. Im *VDI-Wärmeatlas* [91, 92] ist ein Gleichungstyp mit vier Koeffizienten

$$\ln \frac{p_s(T)}{p_{kr}} = \frac{T_{kr}}{T} \left(A\Theta + B\Theta^{1,5} + C\Theta^{2,5} + D\Theta^5 \right) , \tag{2.4}$$

mit

$$\Theta = 1 - \frac{T}{T_{kr}} \tag{2.5}$$

angegeben. Dort sind auch die kritische Temperatur T_{kr} und der kritische Druck p_{kr} sowie die Koeffizienten A bis D für 275 Stoffe aufgeführt. Gegenüber der ANTOINE-Gleichung hat die WAGNER-Gleichung den Nachteil, dass sie nicht nach der Temperatur aufgelöst werden kann, weshalb die Berechnung von Sättigungstemperaturen – wie in Beispiel 2.7 gefordert – bei vorgegebenem Druck nur iterativ möglich ist.

Auf Basis der WAGNER-Gleichung (2.4) wird ein Excel-Berechnungsblatt wie in Abb. 2.8 angelegt:

1. Der obere Teil des Berechnungsblattes dient zur Berechnung des Sättigungsdampfdrucks $p_S(T)$. Es wird ein Eingabefeld für die Temperatur in °C und ein Ergebnisfeld für den Sättigungsdampfdruck erstellt und die Temperatur in Kelvin umgerechnet.
2. Für die Berechnung des Sättigungsdampfdrucks für Ethen sind die Werte der Koeffizienten A bis D der WAGNER-Gleichung sowie der kritischen Daten T_{kr} und p_{kr} aus dem *VDI-Wärmeatlas* erforderlich, siehe Abb. 2.8.
3. Tripeltemperatur und -druck von Ethen wurden [79] entnommen, da diese Daten nicht im *VDI-Wärmeatlas* aufgeführt sind, siehe Abb. 2.8.
4. VBA-Editor aufrufen, neues Modul einfügen: Siehe Abschnitt 2.5.1.
5. Neues Modul umbenennen: Das Modul sollte einen aussagekräftigen Namen bekommen (z. B. „modStoffwerte“), siehe Abschnitt 2.5.1.
6. Der VBA-Code kann entsprechend der Gleichungen (2.4) und (2.5) entweder als Version erstellt werden, bei der die Koeffizienten A bis D einzeln angegeben werden

Listing 2.6: VBA-Code für die Berechnung des Sättigungsdampfdrucks nach *VDI-Wärmeatlas* (Parameter A bis D einzeln anzugeben)

```
'VDI-Wärmeatlas 11. Auflage Abschnitt D3.1
'Gl. (7) Sättigungsdampfdruck WAGNER-Gleichung
'[T] = K , [p_S] = bar
Function p_S_VDI_11(
5   T As Double, _
   p_kr As Double, _
   T_kr As Double, _
   A As Double, _
   B As Double, _
10  C As Double, _
   D As Double _
   ) As Double

   Dim Theta As Double

15  Theta = 1 - T / T_kr
   p_S_VDI_11 = p_kr * Exp((T_kr / T) * (A * Theta + B * Theta ^ 1.5 _
   + C * Theta ^ 2.5 + D * Theta ^ 5))

20 End Function
```

oder als Version, bei der die Koeffizienten als Array angegeben werden:^{6,7}

⁶Wie im Listing beschrieben, benötigt diese Version zusätzlich das Modul „modArraySupport“, welches unter <http://www.cpearson.com/excel/VBAArrays.htm> heruntergeladen werden kann.

⁷Ein umfassender VBA-Code, der die Korrelationsgleichungen für die Stoffwerte nach dem Abschnitt D3.1 aus dem *VDI-Wärmeatlas* einschließlich der Kurzbeschreibungen gemäß Abschnitt 2.5.5 enthält, steht unter <http://www.unit-operations.de> zum Download bereit.

Listing 2.7: VBA-Code für die Berechnung des Sättigungsdampfdrucks nach *VDI-Wärmeatlas* (Parameter *A* bis *D* als Array anzugeben)

```

Option Explicit

'=====
'benötigt das 'modArraySupport'-Modul von Pearson Software Consulting Services
5 '<http://www.cpearson.com/excel/VBAArrays.htm>'
'=====

'VDI-Wärmeatlas 11. Auflage Abschnitt D3.1
'Gl. (7) Sättigungsdampfdruck WAGNER-Gleichung
10 '[T] = K , [p_S] = bar
Function p_S_VDI_11_arr( _
    T As Double, _
    p_kr As Double, _
    T_kr As Double, _
15 a As Variant _
    ) As Double

    Dim Theta As Double
    Dim b() As Double

20

'=====
'Anzahl Parameter 'a'
Const N As Integer = 4
'=====

25

'Falls 'a' ein Range-Object ist, konvertiere es zu einem Array
a = a

'Extrahiere den Vektor 'b' aus 'a', wenn 'a' "richtig" gegeben ist
30 If Not ExtractNumericVector(a, b, N) Then Exit Function

'berechne p_S
Theta = 1 - T / T_kr
p_S_VDI_11_arr = p_kr * Exp((T_kr / T) * (b(1) * Theta + b(2) * Theta ^ 1.5 _
35 + b(3) * Theta ^ 2.5 + b(4) * Theta ^ 5))

End Function

'=====
40

'Extrahiere die erste Spalte/Zeile aus Array 'a' und speichere den Vektor in 'b'.
'Anschließend prüfe Vektor 'b', ob er 'N' Elemente enthält und schließlich, ob er
'numerische Daten enthält
Private Function ExtractNumericVector( _
45 a As Variant, _
    b As Variant, _
    N As Integer _
    ) As Boolean

50

'Initialisiere 'ExtractNumericVector'
ExtractNumericVector = False

'Wenn die Parameter in 'a' vertikal gegeben sind, extrahiere die erste
'Spalte des Arrays, sind die Parameter in 'a' horizontal gegeben,
55 'extrahiere die erste Zeile des Arrays

```

```

        If UBound(a, 1) > 1 And UBound(a, 2) = 1 Then
            If Not GetColumn(a, b, 1) Then Exit Function
        ElseIf UBound(a, 1) = 1 And UBound(a, 2) > 1 Then
            If Not GetRow(a, b, 1) Then Exit Function
60      Else
            Debug.Print "Parameter array 'a' is given non-valid"
            Exit Function
        End If

65      '---
      'Stelle sicher, dass der Vektor 'b' die Elemente 1 bis 'N' enthält
      If LBound(b) <> 1 Or UBound(b) <> N Then
          Debug.Print "Bounds of 'b' are not 1 and " & N
          Exit Function
70      End If
      'Stelle sicher, dass der Vektor 'b' numerische Daten enthält
      If Not IsNumericDataType(b) Then
          Debug.Print "Array is not numeric."
          Exit Function
75      End If
      '---

      'Wenn kein Fehler aufgetreten ist, retourniere 'WAHR'
      ExtractNumericVector = True
80
End Function

```

Die benutzerdefinierten Funktionen können nun in der gesamten Arbeitsmappe eingesetzt werden. Zur Erstellung einer Kurzbeschreibung, die bei Aufruf dieser Funktion und Eingabe der Parameter erscheint, siehe Abschnitt 2.5.5. Es bietet sich an, diese und weitere benutzerdefinierte Funktionen in die in Abschnitt 2.1 erwähnte Vorlage zu übernehmen oder daraus ein Add-In zu erstellen, siehe Abschnitt 2.6.

7. Der untere Teil des Berechnungsblattes dient zur Berechnung der Sättigungstemperatur $T_S(p)$. Es wird ein Eingabefeld für den Druck erstellt und die Sättigungstemperatur iterativ unter Verwendung des Solvers ermittelt. In der vom Solver veränderbaren oder variablen Zelle wird ein geeigneter Startwert vorgegeben, z. B. 200 K. Dieser Wert kann zusätzlich in °C umgerechnet werden.
8. Der Sättigungsdampfdruck wird in Abhängigkeit des eben in der variablen Zelle eingegebenen Startwertes berechnet. In der Zielzelle für den Solver wird die Differenz $\Delta p_S(T_S) = p - p_S(T)$ berechnet.
9. Wie in Abschnitt 2.4 beschrieben, wird ein Makro für den Aufruf und die Anwendung des Solvers aufgezeichnet. Der Wert in der Zielzelle soll gleich null werden. Dabei darf der Wert in der variablen Zelle verändert werden.
10. Schaltfläche für den Aufruf des Solvers einfügen, umbenennen, Code anzeigen: siehe Abschnitt 2.3.1.
11. Code kopieren und einfügen: siehe Abschnitt 2.3.2. Das Ergebnis sollte wie folgt aussehen:

```

Private Sub cmdSolver_Click()
    SolverOk SetCell:="Zielzelle", MaxMinVal:=3, ValueOf:=0, _
        ByChange:="variable_Zelle", Engine:=1, EngineDesc:="GRG Nonlinear"
    SolverSolve
5 End Sub

```

Im Arbeitsblatt wurde **Zielzelle** als Name für die Zielzelle und **variable_Zelle** als Name für die veränderbare oder variable Zelle im Solver gewählt.

12. Speichern und VBA-Editor schließen, Funktion testen.

Zusätzlich kann wie in Abb. 2.8 für die Darstellung der Dampfdruckkurve vom Tripelpunkt bis zum kritischen Punkt ein Diagramm erstellt werden, für das eine Wertetabelle $p_S(T)$ unter Verwendung der neuen benutzerdefinierten Funktion angelegt wird. Es bietet sich an, den Tripelpunkt, den kritischen Punkt und die berechneten Datenpunkte explizit im Diagramm darzustellen.

2.5.3 Berechnung der Dichte eines idealen und eines realen Gases

Nachfolgend werden Berechnungen für die Dichte eines idealen und eines realen Gases sowie weiterer Größen unter Verwendung benutzerdefinierter Funktionen erstellt.

Berechnung der Dichte eines idealen Gases

Beispiel 2.8

Für die Berechnung der Dichte ϱ , des spezifischen Volumens v und des molaren Volumens \bar{V} idealer Gase ist ein Berechnungsblatt zu erstellen. Darin sind beispielhaft die vorgenannten Größen für Ethen für eine Temperatur von 250 K und einen Druck von 16 bar zu berechnen. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.9.)

Es gilt die thermische Zustandsgleichung idealer Gase

$$pV = nRT \quad (2.6)$$

mit der universellen Gaskonstante $R = 8,314\,462\,1\,\text{kJ kmol}^{-1}\,\text{K}^{-1}$ [59]. Gleichung (2.6) und davon abgeleitete Gleichungen sollten – wenn überhaupt – nur für niedrige Drücke (bis etwa 5 bar im unterkritischen Temperaturbereich) und nicht zu tiefe Temperaturen verwendet werden. Besondere Vorsicht gilt bei Berechnungen realer Gase in der Nähe des Sättigungszustands. Dort kann der Fehler schon bei Umgebungsdruck mehrere Prozent betragen [1].

Die Dichte des idealen Gases folgt aus Gleichung (2.6) mit

$$\varrho = \frac{pM}{RT}, \quad (2.7)$$

woraus sich das spezifische Volumen als Kehrwert ergibt

$$v = \frac{1}{\rho} . \quad (2.8)$$

Für den Zusammenhang zwischen spezifischem und molarem Volumen folgt

$$\bar{V} = vM . \quad (2.9)$$

Bevor diese Berechnungen durchgeführt werden, ist zu klären, ob die eingegebenen Werte für Temperatur und Druck überhaupt zu einem Zustand führen, der in der Gasphase liegt. Da aus Abschnitt 2.5.2 nur eine benutzerdefinierte Funktion für die Dampfdruckkurve, und nicht für die Sublimationsdruckkurve vorliegt, ist auch nur eine Abschätzung für den Bereich $T_{\text{tr}} < T < T_{\text{kr}}$ – dem Gültigkeitsbereich von Gleichung (2.4) – möglich. Für diesen Bereich ist die Bedingung $p < p_{\text{S}}$ einzuhalten. Im Bereich $T \geq T_{\text{kr}}$ entfällt die Unterscheidung zwischen Gas- und Flüssigphase und die Anwendung der Zustandsgleichung idealer Gase (und grundsätzlich auch der PENG-ROBINSON-Gleichung (2.13) für reale Gase) ist uneingeschränkt möglich.

Auf Basis dieser Gleichungen und Betrachtungen wird ein Excel-Berechnungsblatt angelegt, siehe dazu den oberen Teil von Abb. 2.9:

1. Die Berechnung erfolgt in Abhängigkeit von Druck und Temperatur. Optional ist die Berechnung der Temperatur ϑ in °C möglich.
2. Erforderliche Daten sind die kritische Temperatur T_{kr} , der kritische Druck p_{kr} , die molare Masse M und der azentrische Faktor ω aus dem *VDI-Wärmeatlas*. Der azentrische Faktor wird erst für die nachfolgende Erweiterung des Berechnungsblattes in Beispiel 2.9 benötigt. Für die Berechnung des Sättigungsdampfdrucks mit der benutzerdefinierten Funktion nach Abschnitt 2.5.2 (mit den Gleichungen (2.4) und (2.5)) werden die Werte der Koeffizienten A bis D aus Abb. 2.8 übernommen, ebenso die Werte für die Tripeltemperatur und den Tripeldruck von Ethen.
3. Für die Berechnungen im Arbeitsblatt wird empfohlen, für die relevanten Zellen Namen zu definieren, z. B. **T** für die Temperatur, **T_{kr}** für die kritische Temperatur oder **rho** für die Gasdichte.
4. Für dieses Beispiel soll eine „Sicherheitsabfrage“ bereits bei der Eingabe von Temperatur und Druck erfolgen. Dazu wird die Zelle für die Eingabe der Temperatur angeklickt und unter Daten Datentools Datenüberprüfung Datenüberprüfung die Eingabe von Dezimalzahlen sowie von Temperaturen größer als die Tripeltemperatur (Verweis auf die entsprechende Zelle) zugelassen. Für die Zelle mit der Eingabe des Druckes erfolgt für das Minimum die Eingabe des Wertes 0 und für das Maximum die Eingabe von `=WENN(T<Tkr;pS;1000)` (mit **T** als der Eingabezelle für die Temperatur, **T_{kr}** als der Zelle mit der kritischen Temperatur, **p_S** als der Zelle mit dem berechneten Sättigungsdampfdruck und 1000 als willkürlich festgelegtem Maximalwert für den Druck).

Da Gleichung (2.7) für die Berechnung der Dichte des idealen Gases auch in den nachfolgenden Kapiteln zur Anwendung kommt, bietet es sich an, eine benutzerdefinierte

Stoff		Ethen	
Temperatur ($T_{tr} < T$)		T	K
	ϑ		$^{\circ}\text{C}$
Druck ($p < p_S$ für $T < T_{kr}$)		p	bar
Stoffwerte aus VDI-Wärmeatlas, 11. Aufl., Abschnitt D3.1			
kritische Temperatur	T_{kr}	K	282,35
kritischer Druck	p_{kr}	bar	50,42
molare Masse	M	kg kmol ⁻¹	28,05
azentrischer Faktor	ω	1	0,087
Sättigungsdampfdruck (WAGNER-Gleichung)	$p_S(T)$	bar	23,288
Koeffizienten Sättigungsdampfdruck	A	1	-6,41327
	B	1	1,45469
	C	1	-1,24183
	D	1	-1,99446
Tripeltemperatur	T_{tr}	K	103,989
Tripeldruck	p_{tr}	bar	1,2265E-03
Berechnung mit der idealen Gasgleichung			
Dichte $\rho = p M / (R T)$	ρ	kg m ⁻³	21,59
spezifisches Volumen $v = 1 / \rho$	v	m ³ kg ⁻¹	0,04631
molares Volumen $\bar{V} = v M$	\bar{V}	m ³ kmol ⁻¹	1,299
Berechnung mit der PENG-ROBINSON-Gleichung		$p = \frac{RT}{\bar{V} - b} - \frac{a(T)}{\bar{V}^2 + 2b\bar{V} - b^2}$	
reduzierte Temperatur	$T_r = T / T_{kr}$	1	0,8854
$\alpha(T) = [1 + (0,37464 + 1,54226\omega - 0,26992\omega^2)(1 - T_r^{0,5})]^2$	$\alpha(T)$	1	1,0607
$a_{kr} = 0,45724 (R^2 T_{kr}^2) / p_{kr}$	a_{kr}	Pa m ⁶ mol ⁻²	0,4998
$b = 0,0778 R T_{kr} / p_{kr}$	b	m ³ mol ⁻¹	3,6224E-05
$a(T) = a_{kr} \alpha(T)$	a	Pa m ⁶ mol ⁻²	0,5301
Realgasfaktor Z, variable Zelle Solver	Z	1	0,8083
Zielfunktion $f(Z) = 0$, Zielzelle Solver	f(Z)	1	-4,25E-07
$f(Z) = Z^3 + Z^2 \left(\frac{bp}{RT} - 1 \right) + Z \left(\frac{ap}{R^2 T^2} - 3 \frac{p^2 b^2}{R^2 T^2} - 2 \frac{bp}{RT} \right) + \frac{p^3 b^3}{R^3 T^3} + \frac{p^2 b^2}{R^2 T^2} - \frac{abp^2}{R^3 T^3} = 0$			
Dichte $\rho = p M / (Z R T)$	ρ	kg m ⁻³	26,713
spezifisches Volumen $v = 1 / \rho$	v	m ³ kg ⁻¹	0,03743
molares Volumen $\bar{V} = v M$	\bar{V}	m ³ kmol ⁻¹	1,050
Vergleich ideal – real			
relative Abweichung $(\rho_{ideal} - \rho_{real}) / \rho_{real}$	$\Delta \rho / \rho$	1	-19,2%

Abbildung 2.9: Excel-Berechnungsblatt für die Berechnung der Dichte und des spezifischen/molaren Volumens eines idealen und eines realen Gases (vgl. Abb. 2.1)

Funktion zu erstellen.

5. VBA-Editor aufrufen: Die neue benutzerdefinierte Funktion kann zu bereits erstellten Funktionen der Arbeitsmappe in der linken Spalte des VBA-Editors (Projekt-Explorer) hinzugefügt werden. Oder es wird ein neues Modul angelegt, wie bereits in Abschnitt 2.5.1 beschrieben.
6. Code einfügen:

Listing 2.8: VBA-Code für die Berechnung der Gasdichte mit der Zustandsgleichung idealer Gase

```
'Gl. (0) Zustandsgleichung idealer Gase: Dichte rho = p M / (R T)
'[rho] = kg m^-3, [T] = K, [p] = bar, [M] = kg kmol^-1
Function rho_ideal(T As Double, p As Double, M As Double) As Double
5    Const R As Double = 8.3144621 'univ. Gaskonstante, [R] = kJ kmol^-1 K^-1

    rho_ideal = (p * M * 100) / (R * T)

End Function
```

7. Berechnung des spezifischen und des molaren Volumens mit den Gleichungen (2.8) und (2.9).
8. Speichern und VBA-Editor schließen, Funktion testen.

Das Ergebnis der Berechnung für das ideale Gas nach Beispiel 2.8 ist – insbesondere wegen des im Beispiel gewählten hohen Druckes – nur mit Vorbehalt zu verwenden.

Berechnung der Dichte eines realen Gases

Für das nachfolgende Beispiel wird das Berechnungsblatt zu Beispiel 2.8 erweitert und die reale Gasdichte mit der PENG-ROBINSON-Gleichung berechnet [63].

Beispiel 2.9

Aufbauend auf Beispiel 2.8 soll ein Berechnungsblatt für die Dichte ρ , das spezifische Volumen v und das molare Volumen \bar{V} eines *realen* Gases auf Basis der PENG-ROBINSON-Gleichung erstellt werden. Darin sind beispielhaft die vorgenannten Größen für Ethen für eine Temperatur von 250 K und einen Druck von 16 bar zu berechnen. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.9.)

Historische Zustandsgleichungen sind die VAN-DER-WAALS-Gleichung

$$\left(p + \frac{a}{\bar{V}^2}\right) (\bar{V} - b) = RT \quad (2.10)$$

oder die Virialform der Zustandsgleichung

$$Z = (1 + B'p + C'p^2 + \dots) \quad (2.11)$$

mit dem so genannten Realgasfaktor

$$Z = \frac{p\bar{V}}{RT} . \quad (2.12)$$

Für zeitgemäße Berechnungen eignen sich in besonderer Weise kubische Zustandsgleichungen wie die PENG-ROBINSON-Gleichung [63]

$$p = \frac{RT}{\bar{V} - b} - \frac{a(T)}{\bar{V}^2 + 2b\bar{V} - b^2} \quad (2.13)$$

oder auch die SOAVE-REDLICH-KWONG-Gleichung [80]

$$p = \frac{RT}{\bar{V} - b} - \frac{a(T)}{\bar{V}^2 + 2b\bar{V}} , \quad (2.14)$$

da einfach handhabbar und hinreichend genau. Diese Gleichungen sind für sehr polare oder für assoziierende Stoffe nicht geeignet.

Die Gleichungen (2.10) und (2.13) sind kubische Funktionen des spezifischen Volumens. Bei gegebenem Druck und gegebener Temperatur liefern sie im unterkritischen Bereich drei reale Lösungen, wobei die größte Lösung für die Gasphase und die kleinste Lösung für die Flüssigkeit gilt. Allerdings sind diese Gleichungen für die Berechnung von Flüssigkeiten nicht geeignet. Im überkritischen Bereich existieren eine reale und zwei komplexe Lösungen. Die reale Lösung beschreibt die überkritische Fluidphase [91, 92].

Um mit der PENG-ROBINSON-Gleichung die Aufgabenstellung in Beispiel 2.9 bearbeiten zu können, ist der azentrische Faktor

$$\omega = -1 - \lg \left(\frac{p_s}{p_{kr}} \right)_{T/T_{kr}=0,7} \quad (2.15)$$

erforderlich, der mit den kritischen Daten berechnet werden kann oder im *VDI-Wärmeatlas* gegeben ist. Zusammen mit der reduzierten Temperatur

$$T_r = \frac{T}{T_{kr}} \quad (2.16)$$

gelten für die Koeffizienten in Gleichung (2.13) die folgenden Beziehungen [91, 92]:

- Für die PENG-ROBINSON-Gleichung

$$a(T) = a_{kr}\alpha(T) \quad (2.17)$$

$$\alpha(T) = \left[1 + (0,374\,64 + 1,542\,26\omega - 0,269\,92\omega^2)(1 - T_r^{0,5}) \right]^2 \quad (2.18)$$

$$a_{kr} = 0,457\,24 \frac{R^2 T_{kr}^2}{p_{kr}} \quad (2.19)$$

$$b = 0,0778 \frac{RT_{kr}}{p_{kr}} . \quad (2.20)$$

- Und für die SOAVE-REDLICH-KWONG-Gleichung

$$a(T) = a_{kr}\alpha(T) \quad (2.21)$$

$$\alpha(T) = \left[1 + (0,48 + 1,574\omega - 0,176\omega^2)(1 - T_r^{0,5})\right]^2 \quad (2.22)$$

$$a_{kr} = 0,42748 \frac{R^2 T_{kr}^2}{p_{kr}} \quad (2.23)$$

$$b = 0,08664 \frac{RT_{kr}}{p_{kr}}. \quad (2.24)$$

Diese Gleichungen können mit den Mischungsregeln auch für Gasgemische angewendet werden.

Die PENG-ROBINSON-Gleichung (2.13) ergibt mit Gleichung (2.12) sowie den Gleichungen (2.17) bis (2.20) die Zielfunktion

$$f(Z) = Z^3 + Z^2 \left(\frac{bp}{RT} - 1 \right) + Z \left(\frac{ap}{R^2 T^2} - 3 \frac{p^2 b^2}{R^2 T^2} - 2 \frac{bp}{RT} \right) + \frac{p^3 b^3}{R^3 T^3} + \frac{p^2 b^2}{R^2 T^2} - \frac{abp^2}{R^3 T^3} = 0, \quad (2.25)$$

mit welcher der Realgasfaktor Z iterativ unter Verwendung des Solvers berechnet werden kann, siehe die nachfolgende Beschreibung. Grundsätzlich können kubische Gleichungen auch mit der cardanischen Formel analytisch gelöst werden, was für unsere Anwendung jedoch aufwendiger wäre.

Für die Berechnung der Realgasdichte wird das Berechnungsblatt in Abb. 2.9 erweitert:

1. VBA-Editor aufrufen: Die neue benutzerdefinierte Funktion $f(Z)$ mit Gleichung (2.25) kann zu den bereits erstellten Funktionen der Arbeitsmappe in der linken Spalte des VBA-Editors (Projekt-Explorer) hinzugefügt werden. Oder es wird ein neues Modul angelegt, siehe Abschnitt 2.5.2.
2. Code einfügen:

Listing 2.9: VBA-Code für die Zielfunktion nach der PENG-ROBINSON-Gleichung

```
'Gleichung f(Z) zur iterativen Berechnung des Realgasfaktors Z
'mit der realen Zustandsgleichung nach PENG und ROBINSON (1976)
'[T] = K , [p] = bar
Function Z_Funktion_PR(Z As Double, T As Double, p As Double, _
5   T_kr As Double, p_kr As Double, omega As Double) As Double

    Dim T_red As Double           'Variablendeklaration
    Dim A As Double
    Dim alpha As Double
    Dim a_kr As Double
10   Dim B As Double

    '=====
```

```

15  Const R As Double = 8.3144621      '[R] = kJ kmol^-1 K^-1
    '=====

    p = p * 100000                      'Umrechnung von bar in Pa
    p_kr = p_kr * 100000

20  T_red = T / T_kr
    alpha = (1 + (0.37464 + 1.54226 * omega - 0.26992 * omega ^ 2) _
        * (1 - T_red ^ 0.5)) ^ 2
    a_kr = 0.45724 * R ^ 2 * T_kr ^ 2 / p_kr
    B = 0.0778 * R * T_kr / p_kr
25  A = a_kr * alpha

    Z_Funktion_PR = Z ^ 3 + Z ^ 2 * ((B * p) / (R * T) - 1) _
        + Z * ((A * p) / (R ^ 2 * T ^ 2) - 3 _
            * (p ^ 2 * B ^ 2) / (R ^ 2 * T ^ 2) _
30         - 2 * (B * p) / (R * T)) _
            + (p ^ 3 * B ^ 3) / (R ^ 3 * T ^ 3) _
            + (p ^ 2 * B ^ 2) / (R ^ 2 * T ^ 2) _
            - (A * B * p ^ 2) / (R ^ 3 * T ^ 3)

35  End Function

```

Anmerkung: Mit dieser UDF ist nicht die explizite Berechnung der Terme gemäß der Gleichungen (2.16) bis (2.20) möglich. Diese Terme wurden nur für die Darstellung in Abb. 2.9 berechnet, um – falls erforderlich – die Fehlersuche bei der Berechnung von $f(Z)$ zu erleichtern.

3. Wie in Abschnitt 2.4 beschrieben, wird ein Makro für den Aufruf und die Anwendung des Solvers aufgezeichnet. Der Wert in der Zielzelle mit der benutzerdefinierten Funktion $f(Z)$ soll gleich null werden. Dabei darf der Wert in der variablen Zelle verändert werden.
4. Schaltfläche einfügen, umbenennen, Code anzeigen: siehe Abschnitt 2.3.1.
5. Code kopieren und einfügen: siehe Abschnitt 2.3.2.
6. Mit dem berechneten Realgasfaktor Z folgt die Dichte des realen Gases mit

$$\varrho = \frac{pM}{ZRT} \quad (2.26)$$

und daraus das spezifische sowie das molare Volumen mit den Gleichungen (2.8) und (2.9). Optional kann die relative Abweichung zwischen idealer und realer Gasdichte ermittelt werden.

7. Speichern und VBA-Editor schließen, Funktion testen.

In Abb. 2.9 wird auch die relative Abweichung zwischen den Dichten des idealen und des realen Gases berechnet, die bei den eingegebenen Werten von Temperatur und Druck erheblich ist – zuungunsten der mit der idealen Zustandsgleichung berechneten Dichte. Der mit FLUIDCAL [28] auf Basis einer hochgenauen Zustandsgleichung ermittelte Wert beträgt $\varrho = 26,389 \text{ kg m}^{-3}$ für 250 K und 16 bar.

2.5.4 Berechnung der spezifischen Verdampfungsenthalpie

Beispiel 2.10

Für Ethen ist für eine Temperatur von $T = 250 \text{ K}$ die spezifische Verdampfungsenthalpie $\Delta_{\text{vap}}h$ unter Verwendung der PENG-ROBINSON-Gleichung zu berechnen. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.10.)

Die Ableitung der Dampfdruckkurve $p_{\text{S}}(T)$

$$\frac{dp_{\text{S}}}{dT} = \frac{h'' - h'}{T(v'' - v')} \quad (2.27)$$

wird als CLAUSIUS-CLAPEYRON-Gleichung bezeichnet. Darin kennzeichnet der hochgestellte Index ' den Zustand der siedenden Flüssigkeit und '' den Zustand des gesättigten Dampfes. Diese Gleichung gilt analog für die Schmelzdruckkurve und die Sublimationsdruckkurve.

Gleichung (2.27) liefert einen Zusammenhang zwischen der spezifischen Verdampfungsenthalpie und den einfach zugänglichen Größen Dampfdruck, Temperatur sowie den spezifischen Volumina von siedender Flüssigkeit und gesättigtem Dampf und kann damit zur Berechnung der spezifischen Verdampfungsenthalpie verwendet werden

$$\Delta_{\text{vap}}h = h'' - h' = T(v'' - v') \frac{dp_{\text{S}}}{dT} . \quad (2.28)$$

Siehe dazu auch die Hinweise zur Genauigkeit von Berechnungen im *VDI-Wärmeatlas* [91, 92], insbesondere die Empfehlung, die CLAUSIUS-CLAPEYRONsche Gleichung nicht bei Dampfdrücken $p_{\text{S}} < 1 \text{ hPa}$ anzuwenden.

Für die Ableitung der WAGNER-Gleichung (2.4) folgt mit der reduzierten Temperatur T_{r} gemäß Gleichung (2.16) (siehe [91, 92])

$$\frac{dp_{\text{S}}}{dT} = -\frac{p_{\text{S}}}{T} \left(\ln \frac{p_{\text{S}}}{p_{\text{kr}}} + A + 1,5B(1 - T_{\text{r}})^{0,5} + 2,5C(1 - T_{\text{r}})^{1,5} + 5D(1 - T_{\text{r}})^4 \right) . \quad (2.29)$$

Alternativ zur (analytischen) Ableitung soll hier eine einfache numerische Methode unter Verwendung des zentralen Differenzenquotienten (ZDQ) verwendet werden. Beim zentralen Differenzenquotienten

$$\frac{\Delta y}{\Delta x} = \frac{f(x + \frac{1}{2}\Delta x) - f(x - \frac{1}{2}\Delta x)}{\Delta x} \quad (2.30)$$

liegen die zur Differenzbildung verwendeten Stellen symmetrisch um den x -Wert, für den die Ableitung angenähert werden soll.

Es wird ein neues Berechnungsblatt in Abb. 2.10 auf Basis von Abb. 2.9 erstellt:

1. Für die Berechnung der spezifischen Verdampfungsenthalpie $\Delta_{\text{vap}}h$ ist in dieser Tabelle nur die Vorgabe der Temperatur erforderlich. Der Druck p ist gleich dem

Stoff			Ethen
Temperatur ($T_{tr} < T < T_{kr}$)	T	K	250
	g	°C	-23,15
Druck	p = p_s(T)	bar	23,288
Stoffwerte aus VDI-Wärmeatlas, 11. Aufl., Abschnitt D3.1			
kritische Temperatur	T_{kr}	K	282,35
kritischer Druck	p_{kr}	bar	50,42
kritische Dichte	ρ_{kr}	kg m ⁻³	214
molare Masse	M	kg kmol ⁻¹	28,05
azentrischer Faktor	ω	1	0,087
Sättigungsdampfdruck (WAGNER-Gleichung)	$p_s(T)$	bar	23,288
Koeffizienten Sättigungsdampfdruck	A	1	-6,41327
	B	1	1,45469
	C	1	-1,24183
	D	1	-1,99446
Flüssigkeitsdichte (Dichte der gesättigten Flüssigkeit $\rho^l = \rho'$)	$\rho^l(T)$	kg m ⁻³	421,8
Koeffizienten Flüssigkeitsdichte	A	1	364,3614
	B	1	211,7762
	C	1	-203,5475
	D	1	188,4411
Trippeltemperatur	T_{tr}	K	103,989
Trippeldruck	p_{tr}	bar	1,2265E-03
Berechnung mit der PENG-ROBINSON-Gleichung			
	$p = \frac{RT}{\bar{V} - b} - \frac{a(T)}{\bar{V}^2 + 2b\bar{V} - b^2}$		
reduzierte Temperatur	$T_r = T / T_{kr}$	1	0,8854
$\alpha(T) = [1 + (0,37464 + 1,54226\omega - 0,26992\omega^2)(1 - T_r^{0,5})]^2$	$\alpha(T)$	1	1,0607
$a_{kr} = 0,45724 (R^2 T_{kr}^2) / p_{kr}$	a_{kr}	Pa m ⁶ mol ⁻²	0,4998
$b = 0,0778 R T_{kr} / p_{kr}$	b	m ³ mol ⁻¹	3,6224E-05
$a(T) = a_{kr} \alpha(T)$	a	Pa m ⁶ mol ⁻²	0,5301
Realgasfaktor Z, variable Zelle Solver	Z	1	0,6913
Zielfunktion $f(Z) = 0$, Zielzelle Solver	$f(Z)$	1	-1,21E-07
$f(Z) = Z^3 + Z^2 \left(\frac{bp}{RT} - 1 \right) + Z \left(\frac{ap}{R^2 T^2} - 3 \frac{p^2 b^2}{R^2 T^2} - 2 \frac{bp}{RT} \right) + \frac{p^3 b^3}{R^3 T^3} + \frac{p^2 b^2}{R^2 T^2} - \frac{abp^2}{R^3 T^3} = 0$			
Dichte gesättigter Dampf $\rho'' = p M / (Z R T)$	ρ''	kg m ⁻³	45,458
spezifisches Volumen gesättigter Dampf $v'' = 1 / \rho$	v''	m ³ kg ⁻¹	0,02200
Berechnung spezifische Verdampfungsenthalpie			
	$\Delta_{vap} h = h'' - h' = T(v'' - v') \frac{dp_s}{dT}$		
Dichte siedende Flüssigkeit	ρ'	kg m ⁻³	421,8
spezifisches Volumen $v' = 1 / \rho'$	v'	m ³ kg ⁻¹	0,00237
gewählte Differenz	ΔT	K	1,00E-03
Ableitung Dampfdruckkurve (aus Differenzenquotient)	dp_s/dT	bar K ⁻¹	0,612
Ableitung Dampfdruckkurve (analytisch)	dp_s/dT	bar K ⁻¹	0,612
spezifische Verdampfungsenthalpie (aus Differenzenquotient)	$\Delta_{vap} h(T)$	kJ kg ⁻¹	300,27

Abbildung 2.10: Excel-Berechnungsblatt für die Berechnung der spezifischen Verdampfungsenthalpie

Sättigungsdampfdruck $p_S(T)$ (gemäß der WAGNER-Gleichung (2.4)).

2. Für die einzugebende Temperatur gilt $T_{tr} < T < T_{kr}$. Auch in diesem Beispiel ist deshalb eine „Sicherheitsabfrage“ sinnvoll, vergleiche Abschnitt 2.5.3. Dazu wird die Zelle für die Eingabe der Temperatur angeklickt und unter Daten Datentools Datenüberprüfung Datenüberprüfung die Eingabe von Dezimalzahlen sowie von Temperaturen größer als der Tripeltemperatur und kleiner als der kritischen Temperatur zugelassen.
3. Die Berechnung des Realgasfaktors Z erfolgt wie in Abb. 2.9 (und entsprechend der Erläuterungen zu Beispiel 2.9) mit der PENG-ROBINSON-Gleichung (2.13) unter Verwendung des Solvers. Daraus werden die Dichte ϱ'' und das spezifische Volumen v'' des gesättigten Dampfes mit den Gleichungen (2.26) und (2.8) ermittelt.
4. Für die Berechnung des spezifischen Volumens v' wird die Dichte der siedenden Flüssigkeit $\varrho' = \varrho^L$ anhand der im *VDI-Wärmeatlas* [91, 92] gegebenen Funktion für die Flüssigkeitsdichte

$$\frac{\varrho^L}{\text{kg/m}^3} = \frac{\varrho_{kr}}{\text{kg/m}^3} + A\Theta^{0,35} + B\Theta^{2/3} + C\Theta + D\Theta^{4/3} \quad (2.31)$$

mit Θ gemäß Gleichung (2.5) berechnet. Dafür werden oben in der Tabelle der Wert für die kritische Dichte ϱ_{kr} und die Werte der Koeffizienten A bis D aus dem *VDI-Wärmeatlas* aufgenommen.

5. Das spezifische Volumen der siedenden Flüssigkeit v' folgt aus der Dichte der siedenden Flüssigkeit mit Gleichung (2.8).
6. Um die spezifische Verdampfungsenthalpie $\Delta_{vap}h$ mit Gleichung (2.28) ermitteln zu können, ist die Ableitung dp_S/dT erforderlich, wofür hier vereinbarungsgemäß der zentrale Differenzenquotient in Gleichung (2.30) verwendet wird. Dafür wird eine Differenz ΔT vorgegeben und für die Differenzbildung in Gleichung (2.30) die Werte $p_S(T + \Delta T/2)$ und $p_S(T - \Delta T/2)$ mit der WAGNER-Gleichung (2.4) berechnet. Unter Beachtung der Dimensionen folgt die Ableitung dp_S/dT und mit Gleichung (2.28) die gesuchte spezifische Verdampfungsenthalpie $\Delta_{vap}h(T)$.
7. Optional ist die Berechnung von $\Delta_{vap}h(T)$ mit Gleichung (2.29) sowie auch mit der Korrelationsgleichung (3.71) aus dem *VDI-Wärmeatlas* möglich.

Der mit FLUIDCAL [28] auf Basis einer hochgenauen Zustandsgleichung ermittelte Wert für Ethen beträgt $\Delta_{vap}h(250 \text{ K}) = 304,303 \text{ kJ kg}^{-1}$.

2.5.5 Kurzbeschreibungen zu benutzerdefinierten Funktionen

Beim Aufrufen und Einfügen von Funktionen in Arbeitsblättern z. B. über die Registerkarte Formeln Funktionsbibliothek Funktionen einfügen erscheinen Kurzbeschreibungen der entsprechenden Funktion. Um diese Kurzbeschreibungen auch beim Aufruf benutzerdefinierter Funktionen (UDFs) zu erhalten, sind zwei Methoden möglich:

- Die Beschreibung im *Objektkatalog* des VBA-Editors und
- die Beschreibung per *Workbook_Open()* mit *Application.MacroOptions*.

Beschreibung im Objektkatalog des VBA-Editors

In Abschnitt 2.2 wurde bereits auf den Objektkatalog hingewiesen, der unter der Registerkarte **Ansicht** » **Objektkatalog** im VBA-Editor aufgerufen wird. Durch Markierung der entsprechenden Funktion im Objektkatalog, Anwahl mit rechter Maustaste und Auswahl von **Eigenschaften** » **Elementoptionen** kann die Kurzbeschreibung der Funktion eingegeben werden. Sollen auch die Parameter der Funktion beschrieben werden, empfiehlt sich die nachfolgende Methode.

Beschreibung per *Workbook_Open()* mit *Application.MacroOptions*

Mit dem **Workbook_Open()**-Ereignis werden Anweisungen beim Öffnen der Arbeitsmappe ausgeführt. Für die Erstellung einer Beschreibung wird wie folgt vorgegangen:

1. VBA-Editor öffnen und Doppelklick auf die Arbeitsmappe, damit der Code angezeigt wird (sollte noch leer sein).
2. Links über dem Codebereich wird im Dropdown-Menü **(Allgemein)** in **Workbook** geändert und im Dropdown-Menü rechts daneben **Open** ausgewählt. Es erscheint der folgende Code, der um die mittlere Zeile ergänzt wird:

```
Private Sub Workbook_Open()
    Call modStoffwerte.AddUDFDescription
End Sub
```

3. Nachfolgend wird beispielhaft eine Beschreibung für die bereits in Abschnitt 2.5.2 verwendete benutzerdefinierte Funktion zur Berechnung des Sättigungsdampfdrucks auf Basis der WAGNER-Gleichung (2.4) erstellt, siehe dazu Abb. 2.11. Dazu wird der in Listing 2.10 aufgeführte Code *vor* der Funktion in Listing 2.6 eingefügt:⁸

Listing 2.10: VBA-Code für die Erstellung der Kurzbeschreibung der benutzerdefinierten Funktion gemäß Listing 2.6

```
Sub AddUDFDescription()
    With Application
        .MacroOptions _
            Category:="Stoffwerte", _
5         Macro:="p_S_VDI_11", _
            Description:="Berechnung des Sättigungsdampfdrucks mit Gl. (7) " & _
                "S. 357 aus dem VDI-Wärmeatlas (2013) 11. Aufl. Abschn. D3.1", _
            ArgumentDescriptions:=Array( _
10         "Temperatur in K", _
            "kritischer Druck in bar", _
            "kritische Temperatur in K", _
            "Parameter A aus VDI-Wärmeatlas", _
            "Parameter B aus VDI-Wärmeatlas", _
            "Parameter C aus VDI-Wärmeatlas", _
```

⁸Ein umfassender VBA-Code, der die Korrelationsgleichungen für die Stoffwerte nach dem Abschnitt D3.1 aus dem *VDI-Wärmeatlas* einschließlich der Kurzbeschreibungen enthält, steht unter <http://www.unit-operations.de> zum Download bereit.

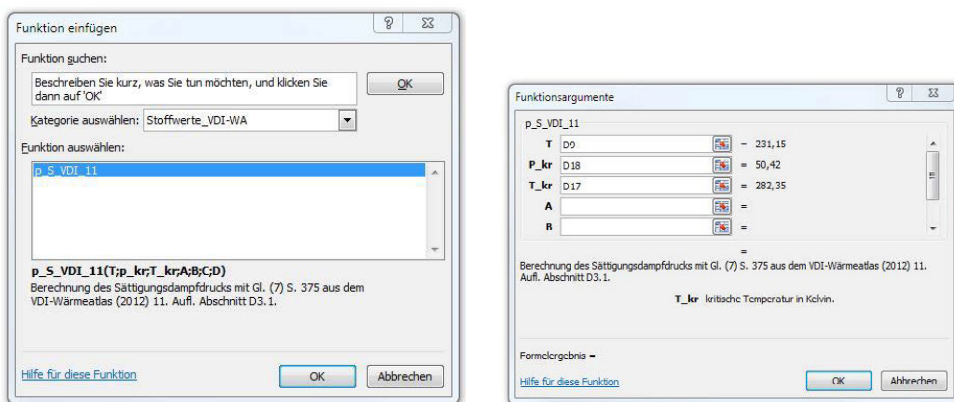


Abbildung 2.11: „Funktion einfügen“-Dialog. Links Funktionsaufruf, rechts Funktionsargumente.

```

15      "Parameter D aus VDI-Wärmeatlas")
      End With
End Sub

```

Erläuterung:

- *Category*: Definiert eine neue Funktionskategorie oder wählt eine der von Excel bereits vorgegebenen Kategorien. Diese Definition ist optional.
- *Macro*: Definiert die Funktion, deren Beschreibung erstellt wird.
- *Description*: Hier wird die Kurzbeschreibung eingegeben.
- *ArgumentDescriptions*: Hier werden die Hinweise zu den Parametern in Form eines Arrays eingegeben.

2.6 Erstellung von Add-Ins

Aus einer Arbeitsmappe mit benutzerdefinierten Funktionen kann ein *Add-In* erstellt werden. Add-Ins sind an der Endung **.xlam** erkennbar. Aktivierte Add-Ins können in allen Arbeitsmappen verwendet werden. Ein Beispiel für ein vorhandenes Add-In ist der Solver. Vorgehensweise:

- Arbeitsmappe als Add-In speichern: **Datei** > **Speichern unter** Dateityp *Excel-Add-In* (*.xlam). Speicherort wählen.
- Excel schließen, neu öffnen. Hinweis auf das Add-In unter **Datei** > **Optionen** > **Add-Ins** einrichten.
- Um auch in VBA Funktionen aus dem Add-In aufrufen zu können, ist ein Verweis auf das Add-In im VBA-Editor unter **Extras** > **Verweise** hinzuzufügen.
- Der Aufruf einer benutzerdefinierten Funktion aus dem Add-In kann nun auf die übliche Weise erfolgen.

2.7 Ermittlung von Ausgleichsfunktionen

Bei einer Regression oder Ausgleichsrechnung werden auf der Basis von Daten oder Messdaten die Parameter einer vorgegebenen Funktion ermittelt. Excel bietet hier zwei Möglichkeiten an. Die Ermittlung von Ausgleichsfunktionen mit der Trendlinienfunktion geht den Umweg über vorher zu erstellende Diagramme und ist bezüglich des Handlings mit den zu ermittelnden Parametern umständlich. Die in Abschnitt 2.10.2 beschriebene **RGP**-Funktion (sowie die **RKP**-Funktion) ist etwas kryptisch. Nachteil bei der Methoden ist, dass sie nur auf polynomische, logarithmische und exponentielle sowie Potenzfunktionen anwendbar sind. Nachfolgend werden zwei alternative Methoden vorgestellt.

2.7.1 Nichtlineare Regression unter Verwendung des Solvers

Die Ermittlung selbst definierter Ausgleichsfunktionen ist durch eine nichtlineare Regression unter Verwendung des Solvers auf Basis einer Ausgleichung nach der GAUSSschen *Methode der kleinsten Quadrate* möglich [62].

Beispiel 2.11

Für die ANTOINE-Gleichung (2.3) sollen die Parameter A , B und C an die im *VDI-Wärmeatlas* gegebenen Sättigungsdaten von Wasser für den Temperaturbereich von $0,01\text{ °C}$ bis $60,0\text{ °C}$ angepasst werden. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.12.)

Es wird ein Berechnungsblatt wie in Abb. 2.12 erstellt:

1. Anlegen der Wertetabelle: Aus dem *VDI-Wärmeatlas* [91, 92] werden die Sättigungsdaten für den vorgegebenen Temperaturbereich übernommen und die entsprechenden Datenpunkte in einem Diagramm dargestellt.
2. Berechnung der Funktionswerte: Unter Vorgabe sinnvoller Startwerte für die ANTOINE-Parameter A , B und C (z. B. 1) werden in der Spalte p_S' die Sättigungsdampfdrücke mit Gleichung (2.3) – auch hier am besten unter Verwendung einer benutzerdefinierten Funktion – berechnet. Der Tripeldruck p_{tr} von Wasser ist in der Spalte p_S für $\vartheta = 0,01\text{ °C}$ aufgeführt.
3. In der nächsten Spalte werden die vertikalen Abweichungsquadrate $(p_S' - p_S)^2$ und daraus die Summe der Abweichungsquadrate am unteren Ende der Spalte ermittelt.
4. Berechnung der relativen Abweichungen: In der letzten Spalte stehen die relativen Abweichungen $\Delta p_S/p_S = (p_S' - p_S)/p_S$.
5. Aufruf des Solvers: Zielzelle für den Solver ist die Summe der Abweichungsquadrate, die minimiert werden soll. Die veränderbaren oder variablen Zellen sind die Zellen mit den ANTOINE-Parametern A , B und C . Als Lösungsmethode für den Solver empfiehlt sich hier **GRG-nichtlinear** mit der Option **Ableitungen zentral**. Die Option **Nicht eingeschränkte Variablen als nicht-negativ festlegen** ist zu deaktivieren.

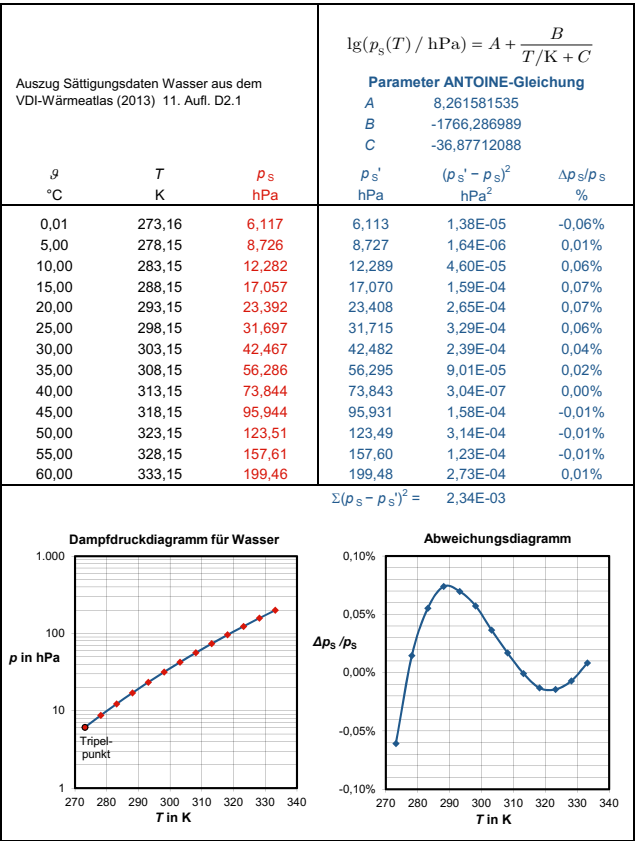


Abbildung 2.12: Anpassung der ANTOINE-Parameter an die Sättigungsdaten für Wasser (Sättigungsdaten entnommen aus [91, 92])

Da die relative Abweichung im Bereich um den Tripelpunkt recht groß ist, wird dieser Datenpunkt bei der Berechnung der Summe der Abweichungsquadrate z. B. mit dem Faktor 25 gewichtet. Die vorgegebenen Dampfdrücke p_S werden als Punkte und die berechneten Dampfdrücke p_S' als Linie (ohne Datenpunkte) im Dampfdruckdiagramm mit logarithmisch skaliertem Ordinate aufgetragen. Da auch relativ große Abweichungen in dieser Diagrammdarstellung kaum erkennbar sind, wird zusätzlich ein „Abweichungsdiagramm“ mit den temperaturabhängigen relativen Abweichungen erstellt.

2.7.2 Polynomregression unter Verwendung einer benutzerdefinierten Funktion

Alternativ zu Abschnitt 2.7.1 soll eine Regression unter Anwendung der Polynomfunktion

$$y = a_0 + a_1x + a_2x^2 + \dots = \sum_{i=0}^n a_i x^i . \quad (2.32)$$

auf Basis einer benutzerdefinierten Funktion, die keiner Hilfszellen bedarf und direkt die gewünschten Koeffizienten zurückgibt, erstellt werden. Grundlage hierfür ist ebenfalls die *GAUSSsche Methode der kleinsten Quadrate*.

Beispiel 2.12

Für das reale binäre System Ethanol-Wasser soll für einen konstanten Druck $p = 1,013$ bar die Siedelinie durch ein Polynom als Ausgleichsfunktion auf Basis einer benutzerdefinierten Funktion beschrieben werden (vergleiche hierzu Beispiel 2.15 in Abschnitt 2.10.2). (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.13.)

Für die Lösung der Aufgabenstellung verwenden wir den dankenswerterweise von KRUCKER [46] erstellten VBA-Code der benutzerdefinierten Funktion **PolynomReg**, siehe den zweiten Teil des nachfolgend aufgeführten Listings 2.11.⁹ Zusätzlich ist in diesem Listing die Funktion **Polynom** enthalten, die eine Erweiterung der hier nicht gelisteten Funktion **PolynomEval** von KRUCKER ist. Diese neue Funktion ermöglicht – unter Angabe des optionalen Argumentes **NV** – auch die Verarbeitung von Zellen mit dem Fehlerwert **#NV** und damit die weitgehend flexible Vorgabe unterschiedlicher Grade der Polynomfunktion entsprechend Gleichung (2.32).¹⁰

⁹siehe dazu auch <http://www.krucker.ch/skripten-uebungen/IAMSkript/IAMKap3.pdf> [46]

¹⁰Die längeren VBA-Codes aus diesem Buch stehen unter <http://www.unit-operations.de> zum Download bereit.

Listing 2.11: VBA-Code für die Ermittlung der Koeffizienten einer Polynomregression

```

'Polynomregression als Erweiterung der Funktion PolynomEval von G. Krucker
Public Function Polynom(Coefficients As Variant, x As Double, Optional NV As Variant) _
    As Variant

5   Dim i As Integer
    Dim sum As Double
    Dim bCoeffRange As Boolean

    'Falls 'Coefficients' als Range gegeben sind
10  If TypeName(Coefficients) = "Range" Then
        'setze 'bCoeffRange' auf 'Wahr'
        bCoeffRange = True
        'und überprüfe die Dimensionen der Koeffizientenmatrix
        If Coefficients.Rows.Count >= 1 And Coefficients.Columns.Count = 1 Then
15          'sind die Koeffizienten spaltenweise gegeben, ist alles in Ordnung
            Coefficients = Coefficients.Value2
        ElseIf Coefficients.Rows.Count = 1 And Coefficients.Columns.Count >= 1 Then
            'sind sie zeilenweise gegeben, werden sie transponiert
            Coefficients = Application.WorksheetFunction.Transpose(Coefficients.Value2)
20        Else
            'hier ist weder ein Skalar noch ein Vektor aber eine Matrix gegeben
            '--> beende die Funktion
            Polynom = "wrong"
            Exit Function
        End If
    End If

    'ist der Parameter 'NV' gegeben und 'Wahr', dann ersetze alle 'NV's durch Nullen
    On Error Resume Next
30    If Not IsMissing(NV) Then
        'dies macht nur Sinn, wenn mehr als 1 Koeffizient (Skalar) gegeben ist
        If NV = True And IsArray(Coefficients) = True Then
            For i = UBound(Coefficients) To LBound(Coefficients) Step -1
                'wenn die Koeffizienten als "Range" gegeben sind, ist
35                'Coefficients' ein zweidimensionales Array
                If bCoeffRange = True Then
                    'sind die gegebenen Koeffizienten keine Zahlen, beende die Schleife
                    If Not IsNumeric(Coefficients(i, 1)) Then
                        'ist der aktuelle Koeffizient der Fehlerwert 'NV',
                        'ersetze ihn durch 0
                        If Coefficients(i, 1) = CVErr(xlErrNA) Then
                            Coefficients(i, 1) = 0
                        End If
                    Else
45                        Exit For
                    End If
                'wird die Funktion von einer anderen Funktion aufgerufen,
                'dann ist 'Coefficients' als Vektor mit nur einer Dimension gegeben
                Else
50                'sind die gegebenen Koeffizienten keine Zahlen, beende die Schleife
                If Not IsNumeric(Coefficients(i)) Then
                    'ist der aktuelle Koeffizient der Fehlerwert 'NV',
                    'ersetze ihn durch 0
                    If Coefficients(i) = CVErr(xlErrNA) Then
80                        Coefficients(i) = 0
                    End If
                End If
            End For
        End If
    End If

```

```

        End If
    Else
        Exit For
    End If
End If
Next
End If
End If
On Error GoTo 0

'ist 'Coefficients' ein Skalar, entspricht der Koeffizient dem Polynom
If IsArray(Coefficients) = False Then
    sum = Coefficients
'andernfalls wird das Polynom berechnet
Else
    'ist 'Coefficients' als "Range" gegeben, fängt der untere Index
    'mit 1 an, weshalb der Exponent um 1 reduziert wird
    If bCoeffRange = True Then
        For i = 1 To UBound(Coefficients)
            sum = sum + Coefficients(i, 1) * x ^ (i - 1)
        Next
    'ist 'Coefficients' dagegen als Vektor gegeben, kann der
    'Startindex beliebig sein, weshalb der Exponent um den unteren
    'Index von 'Coefficients' korrigiert wird
    Else
        For i = LBound(Coefficients) To UBound(Coefficients)
            sum = sum + Coefficients(i) * x ^ (i - LBound(Coefficients))
        Next
    End If
End If

Polynom = sum

End Function

'=====
' Berechnen der Polynomkoeffizienten a0,...,an für eine polynomiale Ausgleichsfunktion
' 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: 17.08.1995, 22.09.1996
' Sprache: VBA for EXCEL7
' -----
' Stefan Pinnow      11.05.2015
' - "Parameterkontrolle" überarbeitet.
' - ...
'
Function PolynomReg(x As Range, y As Range, Polynomgrad As Double) As Variant

    'Anzahl x- und y-Werte
    Dim AnzX As Integer, AnzY As Integer
    'Anzahl auszugleichender Datenpunkte

```

```

Dim m As Integer
'Dynamisches Array für die Summe der Potenzen von xk
115 Dim Sxk() As Double
'Dynamisches Array für die Summe der Potenzen von xk * yk
Dim Sxkyk() As Double
'Dynamisches Array für die Koeffizientenmatrix G
Dim G() As Double, g1 As Variant
120 'Dynamisches Array für den Konstantenvektor c
Dim c() As Double
'Dynamisches Array für die Polynomkoeffizienten a0,...,an
Dim a() As Double
'Laufvariablen
125 Dim i As Integer, j As Integer, k As Integer

'---
'Parameterkontrolle
'---
130 'Polynomgrad' muss eine ganze Zahl >= 1 sein
If Not IsNumeric(Polynomgrad) Then
    PolynomReg = CVErr(xlErrValue)
    Exit Function
ElseIf Cint(Polynomgrad) <> Polynomgrad Then
135     PolynomReg = CVErr(xlErrValue)
    Exit Function
ElseIf Polynomgrad < 1 Then
    PolynomReg = CVErr(xlErrValue)
    Exit Function
140 End If

AnzX = x.Count 'Anzahl Datenpunkte in den Arrays bestimmen
AnzY = y.Count

145 'die Anzahl der X- und Y-Punkte muss übereinstimmen
If (AnzX <> AnzY) Then
    PolynomReg = CVErr(xlErrValue)
    Exit Function
End If

150 'der Polynomgrad muss kleiner sein als die Anzahl der gegebenen Punkte
If AnzX <= Polynomgrad Then
    PolynomReg = CVErr(xlErrValue)
    Exit Function
End If
155 '---

m = AnzX

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

```

```

170 Next i
    For i = 0 To Polynomgrad
        Sxkyk(i) = 0
        For k = 1 To m
            Sxkyk(i) = Sxkyk(i) + x(k) ^ i * y(k)
175 Next k
Next i

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

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

195 'Gleichungssystem G * a = c lösen 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
200 a(i - 1) = a(i - 1) + g1(i, j) * c(j)
        Next j
    Next i
    PolynomReg = a 'Koeffizientenvektor a0,...,an retournieren

205 End Function

```

Für die Lösung der Aufgabe wird ein Berechnungsblatt wie in Abb. 2.13 erstellt. Die Werte in den beiden linken Spalten dieser Tabelle (mit den auf zwei Nachkommastellen gerundeten Werten für die Temperaturen) wurden Abb. 3.16 entnommen:

1. Der Grad des Polynoms wird in einer Zelle eingetragen (z. B. der Wert 2).
2. Die benutzerdefinierte Funktion berechnet die Parameter der Funktion entsprechend Gleichung (2.32). Für die Ermittlung der Parameter wird im Arbeitsblatt ein Zellbereich oder Array mit einer ausreichenden Anzahl von Zellen ausgewählt, z. B. für ein Polynom achten Grades neun Zellen. Für die Bestimmung der Parameter erfolgt der Aufruf der Funktion **PolynomReg**. Die Eingabe wird mit der Tastenkombination **Strg**+**↑**+**Enter** statt einfach nur mit **Enter** abgeschlossen, da **PolynomReg** eine Arrayfunktion ist.
3. Die Berechnung der Funktionswerte erfolgt in der dritten Spalte der Tabelle unter Anwendung der benutzerdefinierten Funktion **Polynom**. Dabei wird dem optionalen Argument der Wert **WAHR** zugewiesen, womit auch die **#NV**-Werte des ausgegebenen Polynomparameterblocks verarbeitet werden können.

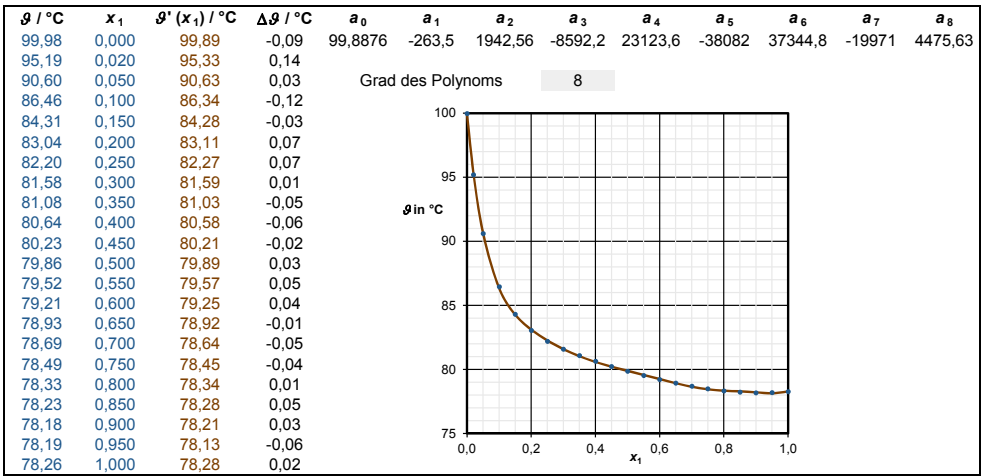


Abbildung 2.13: Excel-Berechnungsblatt für die Berechnung einer Polynomfunktion für die Siedelinie des realen binären Systems Ethanol-Wasser mit einer benutzerdefinierten Funktion

- 4. Der Grad des Polynoms wird erhöht, bis der berechnete Linienzug im Diagramm die gegebenen Punkte mit ausreichender Genauigkeit wiedergibt.
- 5. Im Diagramm sind die vorgegebenen Datenpunkte der Siedelinie und die mit der Polynomfunktion berechneten Daten als Linienzug dargestellt. Die vierte Spalte enthält die Abweichungen zwischen diesen Daten.

Die in Abschnitt 2.10.2 verwendete Trendlinienfunktion liefert (bis zum Polynom sechsten Grades) identische Ergebnisse; jedoch stehen die ermittelten Koeffizienten des Polynoms nicht in Zellen für weitere Berechnungen zur Verfügung.

2.8 Nullstellensuche mit dem ZDQ-Verfahren

In Abschnitt 2.4 wurde eine Nullstellensuche unter Verwendung des Solvers durchgeführt. Nachfolgend wird ein VBA-Modul zur Nullstellenberechnung auf Basis des auf dem NEWTON-Verfahren beruhenden ZDQ-Verfahrens erstellt. Zum physikalischen Hintergrund der Aufgabenstellung siehe Abschnitt 2.4.

Beispiel 2.13

Die Berechnung der Nullstelle der Funktion $f(\varphi) = 2\pi\varepsilon + \sin(\varphi) - \varphi$ für $\varepsilon = 0,001$ soll auf Basis des ZDQ-Verfahrens mit einer benutzerdefinierten Funktion erfolgen. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.14.)

Die Nullstelle einer differenzierbaren Funktion $f(x) = 0$ kann mit dem NEWTON-Verfahren bestimmt werden. Dafür ist die Ableitung der Funktion erforderlich.

Beim Sekantenverfahren, der Bisektion oder dem nachfolgend vorgestellten modifizierten NEWTON-Verfahren wird die Ermittlung der Ableitung umgangen. Es beruht auf der Berechnung des *Differenzenquotienten*

$$\frac{\Delta y}{\Delta x} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad (2.33)$$

zur näherungsweisen Bestimmung der Ableitung. Der Differenzenquotient entspricht der Steigung der Sekante des Graphen durch die Punkte $(x_0, f(x_0))$ und $(x_1, f(x_1))$.

Beim zentralen Differenzenquotienten ZDQ liegen die zur Differenzbildung verwendeten Stellen symmetrisch um den x -Wert, für den die Ableitung angenähert werden soll. Das ZDQ-Verfahren lässt sich aus dem NEWTON-Verfahren herleiten, indem die Ableitung durch den zentralen Differenzenquotienten gemäß Gleichung (2.30) ersetzt wird. Damit ergibt sich die Iterationsvorschrift

$$x_1 = x_0 - \frac{\Delta x}{\Delta y} f(x_0) . \quad (2.34)$$

Zu beachten ist das Konvergenzverhalten, worin sich die beiden genannten Verfahren unterscheiden, und die Wahl des geeigneten Startwerts, auch hinsichtlich der Konvergenz, siehe einschlägige Literatur.

Die in diesem Abschnitt beschriebene Nullstellensuche kann auch für andere Aufgabenstellungen benutzt werden (siehe Abschnitte 4.4, 9.1 und 9.3). Deshalb ist die Entwicklung einer allgemeinen Funktion `ZDQ_Solver` sinnvoll, die allein das ZDQ-Verfahren abbildet und von der eigentlichen Funktion aufgerufen wird. Vorgehensweise bei der Umsetzung in VBA, siehe dazu den nachfolgenden Code für die benutzerdefinierte Funktion in Listing 2.12:

1. VBA-Editor aufrufen, neues Modul einfügen: Siehe Abschnitt 2.5.1.
2. Neues Modul umbenennen: Das Modul sollte einen aussagekräftigen Namen bekommen (z. B. „modZDQ“), siehe Abschnitt 2.5.1.
3. Vor dem Erstellen der benutzerdefinierten Funktion werden die globalen Konstanten definiert: Der Abstand zwischen zwei x -Werten `gcdDelta` = $\Delta x/2$, die maximale Anzahl der Iterationen `gciMaxIteration` und das Abbruchkriterium `gcdAbbruchkrit`.
4. Die Funktion `ZDQ_Solver` hat die Parameter `Startwert` und `y` vom Typ `Double`, `FunktionsName` vom Typ `String` und `arr` vom Typ `Variant`. Es erfolgt die Deklaration der Variablen `dx0`, `dZielfkt` und `dDiffQuot` vom Typ `Double`, der Laufvariablen `i` vom Typ `Integer` und der Variablen `fx`, `fxp` und `fxm` vom Typ `Double`.
5. Wie bereits in Abschnitt 2.2 beschrieben, ist mit dem Befehl `Debug.Print` die Ausgabe von Informationen im Direktbereich möglich. Dazu wird mit dem ersten `Debug.Print` ein neuer Bereich abgegrenzt und mit dem zweiten `Debug.Print` der Aufruf der ZDQ-Funktion ohne den optionalen Parameter `arr` ausgegeben, siehe den Direktbereich in Abb. 2.15.

6. Der weitere Ablauf: Initialisierung von `dX0` mit dem `Startwert` und Ausführen der nachfolgenden Anweisung einschließlich Berechnung der Werte der Zielfunktion `dZielfkt` und des Differenzenquotienten `dDiffQuot` mit der maximalen Iterationszahl `gciMaxIteration`, solange der Absolutwert der Zielfunktion `Abs(dZielfkt)` größer als der Wert des Abbruchkriteriums `gcdAbbruchkrit` ist. Dabei Berechnung des neuen $f(x)$ -Wertes der Nullstelle mit `dX0` gemäß der Iterationsvorschrift in Gleichung (2.34). Die Fallunterscheidung zur Berechnung der $f(x)$ -Werte `fx`, `fxp` und `fxm` ist erforderlich, um auch Funktionen aufrufen zu können, die mehr als ein Argument besitzen. Alle diese zusätzlichen Argumente können mittels des Parameter-Arrays `arr` an `FunktionsName` weitergegeben werden (siehe dazu z. B. Listing 4.2).
7. Mit dem dritten `Debug.Print` werden bei jedem Schleifendurchlauf die Werte des Schleifenzählers `i`, des x -Wertes `dX0` und der Zielfunktion `dZielfkt` angezeigt, siehe Abb. 2.15. Wenn `Abs(dZielfkt(dX0)) < gcdAbbruchkrit` wird `dX0` an die Funktion `ZDQ_Solver` übergeben. Wenn nicht, wird die `CVErr`-Funktion verwendet, um einen Fehler auszulösen und in diesem Fall der benutzerdefinierten Funktion `ZDQ_Solver` der Fehlerwert `#NV` zugewiesen.

Hinweis: Das erstellte Modul kann in ein Add-In „ausgelagert“ und aus diesem aufgerufen werden, siehe Abschnitt 2.6. Dadurch vereinfachen sich die Aufrufe des Moduls in den Beispielen der Abschnitte 4.4, 9.1 und 9.3.

Listing 2.12: VBA-Code für die Nullstellensuche nach dem ZDQ-Verfahren

```
Option Explicit

'=====
Const gcdDelta As Double = 10 ^ (-6)           'Delta x
5 Const gciMaxIteration As Integer = 50         'max. Anzahl Iterationen
Const gcdAbbruchkrit As Double = 0.5 * 10 ^ (-12) 'Abbruchkriterium
'=====

'Allgemeine Funktion zur Verwendung des ZDQ-Solvers.
10 ' 0 = y - f(x)
'Parameter:
' Startwert = Startwert für x für die Iteration
' y = y-Wert, für den der x-Wert gesucht werden soll
' FunktionsName = Funktion, für die die Iteration durchgeführt werden soll
15 ' arr = ParameterArray, welches alle weiteren benötigten Parameter
' enthält, die für 'FunktionsName' erforderlich sind
Function ZDQ_Solver( _
    Startwert As Double, _
    y As Double, _
20 FunktionsName As String, _
    Optional arr As Variant _
    ) As Variant

    Dim dX0 As Double
    Dim dZielfkt As Double
    Dim dDiffQuot As Double
    Dim i As Integer
```

```

30  Dim fx As Double      'zum Speichern von f(x)
    Dim fxp As Double    'zum Speichern von f(x+Dx)
    Dim fxm As Double    'zum Speichern von f(x-Dx)

    Debug.Print "-----"
35  Debug.Print "Funktionsaufruf: 'ZDQ_Solver(" & Startwert & ";" & _
        y & "; Funktion: " & FunktionsName & ")'"

    'Initialisiere 'dX0'
    dX0 = Startwert

40  'Iteriere
    For i = 1 To gciMaxIteration
        'Falls kein ParameterArray gegeben ist,
        'rufe 'FunktionsName' ohne dieses auf, sonst mit
45  If IsMissing(arr) Then
            fx = Application.Run(FunktionsName, dX0)
        Else
            fx = Application.Run(FunktionsName, dX0, arr)
        End If

50  dZielFkt = y - fx
        If Abs(dZielFkt) < gcdAbbruchkrit Then
            Exit For
        End If

55  'Falls kein ParameterArray gegeben ist,
        'rufe 'FunktionsName' ohne dieses auf, sonst mit
        If IsMissing(arr) Then
            fxp = Application.Run(FunktionsName, dX0 + gcdDelta)
            fxm = Application.Run(FunktionsName, dX0 - gcdDelta)
60  Else
            fxp = Application.Run(FunktionsName, dX0 + gcdDelta, arr)
            fxm = Application.Run(FunktionsName, dX0 - gcdDelta, arr)
        End If

65  dDiffQuot = ((y - fxp) - (y - fxm)) / (dX0 + gcdDelta - (dX0 - gcdDelta))

        dX0 = dX0 - dZielFkt / dDiffQuot
        Debug.Print "i = " & i, "x = " & dX0, "ZielFkt = " & dZielFkt
70  Next i

    'Falls 'dZielFkt' kleiner ist als das Abbruchkriterium, wird das
    'Ergebnis zurückgegeben, andernfalls ein Fehlerwert
    If Abs(dZielFkt) < gcdAbbruchkrit Then
75  ZDQ_Solver = dX0
    Else
        ZDQ_Solver = CVErr(xlErrNA)
    End If

80  End Function

```

Mit dieser Vorarbeit ist nun die eigentliche Aufgabe – das Finden der Nullstelle – relativ einfach. Es müssen lediglich zwei weitere benutzerdefinierte Funktionen geschrieben werden: eine Funktion zur Berechnung von $f(x)$ bzw. in diesem Fall $f(\varphi)$,

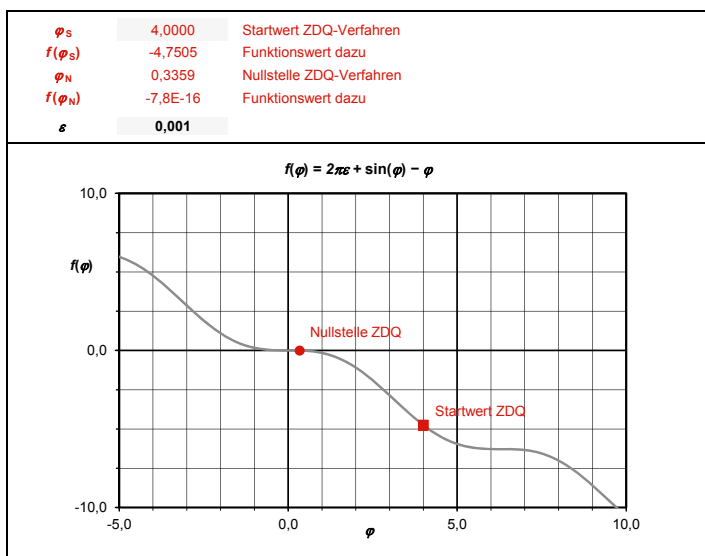


Abbildung 2.14: Nullstellensuche unter Anwendung des ZDQ-Verfahrens

und eine Funktion, die den ZDQ-Solver aufruft. Dabei entspricht $2\pi\varepsilon + \sin(\varphi) - \varphi = 0$ der Bedingung $2\pi\varepsilon = \varphi - \sin(\varphi)$, wobei der konstante Term (hier $2\pi\varepsilon$) beim Aufruf des Solvers berechnet wird. Vorgehensweise bei der Umsetzung in VBA, siehe dazu den nachfolgenden Code für die beiden benutzerdefinierten Funktionen in Listing 2.13:

1. VBA-Editor aufrufen, neues Modul einfügen: Siehe Abschnitt 2.5.1.
2. Neues Modul umbenennen: Das Modul sollte einen aussagekräftigen Namen bekommen (z. B. „modNullstelleSinusfkt“), siehe Abschnitt 2.5.1.
3. Erstellung der Funktion `SolverSinusfkt` mit den Parametern `Startwert_phi` und `epsilon` jeweils vom Typ `Double`, die direkt die eben erstellte Funktion `ZDQ_Solver` aufruft. Dabei ist $2\pi\varepsilon$ der zu übergebende y -Wert.
4. Erstellung der Funktion `WinkelPhi` ($f(x)$ -Funktion) als `Private` (da sonst nicht benötigt). In diesem Fall berechnet die Funktion die Differenz $\varphi - \sin(\varphi)$.

Die Berechnung startet mit Aufruf der Funktion `SolverSinusfkt` im Arbeitsblatt in der Zelle φ_N (Nullstelle ZDQ-Verfahren) in Abhängigkeit der Parameter φ_S (Startwert ZDQ-Verfahren) und ε entsprechend Abb. 2.14.

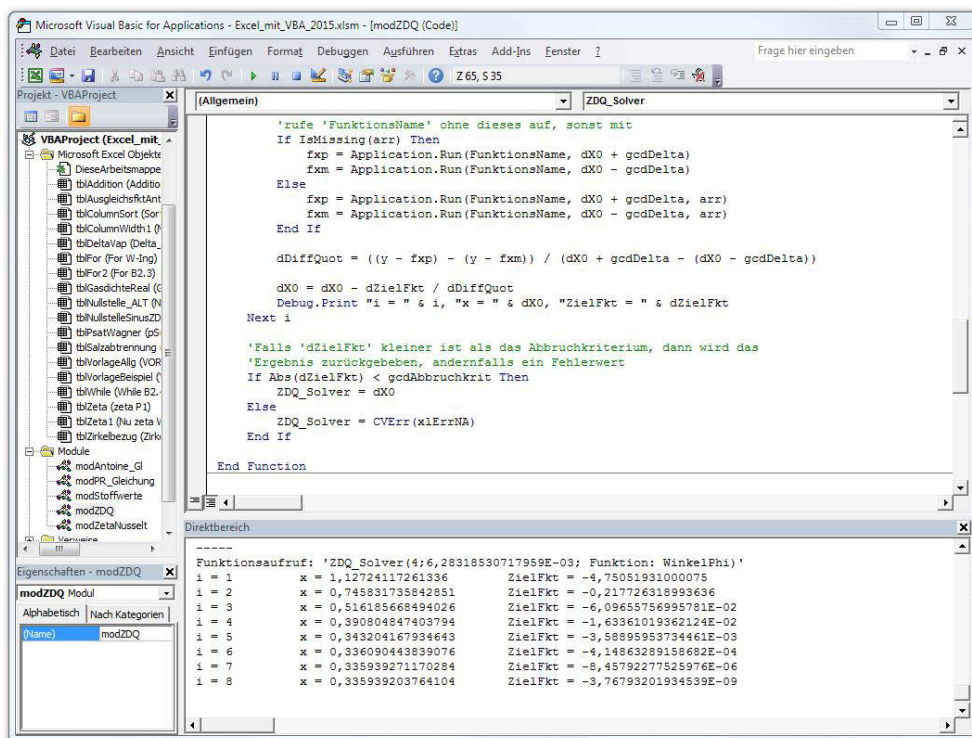


Abbildung 2.15: Ausgabe von Informationen im Direktbereich mit dem `Debug.Print`-Befehl

Listing 2.13: VBA-Code für die Nullstellensuche der Funktion gemäß Beispiel 2.13 unter Verwendung des ZDQ-Verfahrens

```

'Bestimmung des unbenetzten Winkels der durchströmten Rohre ('phi' in Bogenmaß)
'bei der Zweiphasenströmung durch horizontale oder wenig geneigte
'Verdampferrohre in Abhängigkeit vom Dampfvolumenanteil ('epsilon')
'nach VDI-WA11, Abschnitt H3.1, Gl. (27)
5 Function SolverSinusfkt( _
    Startwert_phi As Double, _
    epsilon As Double _
    ) As Variant
    SolverSinusfkt = _
10     ZDQ_Solver( _
        Startwert_phi, _
        2 * Application.WorksheetFunction.Pi * epsilon, _
        "WinkelPhi" _
    )
15 End Function

'Hilfsfunktion zur Berechnung von f(x) für das ZDQ-Verfahren
Private Function WinkelPhi(phi As Double) As Double
    WinkelPhi = phi - Sin(phi)
20 End Function

```

2.9 Anwendung von Zirkelbezügen

Iterative Berechnungen sind in Excel mit drei Standardmethoden möglich:

- Mit einer Zielwertsuche,
- unter Verwendung des Solvers (siehe Abschnitt 2.4) und
- mit einem Zirkelbezug.

Dazu kommen mit VBA selbst erstellte Lösungsmethoden wie z. B. die in Abschnitt 2.8 beschriebene Nullstellensuche. Die Zielwertsuche wird hier nicht weiter behandelt. Sie ist unter **Daten** » **Datentools** » **Was-wäre-wenn-Analyse** » **Zielwertsuche** zu finden.

Für einfache Berechnungen bietet sich die Verwendung von Zirkelbezügen an. Dafür wird nachfolgend beispielhaft die Rückführung von Teilströmen bei der Auslegung verfahrenstechnischer Prozesse betrachtet. Weitere Beispiele für die Verwendung von Zirkelbezügen: Beispiel 10.1 in Abb. 10.3 zur Berechnung des Druckverlustes für ein innendurchströmtes Rohr und Beispiel 11.1 in Abb. 11.2 zur Berechnung der stationären Sinkgeschwindigkeit für ein kugelförmiges Einzelpartikel.

Beispiel 2.14

Aus einem Feedstrom (Strom 1), der aus Wasser und darin gelöstem Salz besteht, soll in einem stationären Prozess Salz (Strom 6) gewonnen werden, siehe dazu das Grundfließschema in Abb. 2.16. Dabei wird ein Teilstrom (Strom 4) des salzarmen Produktstroms in den Prozess zurückgeführt. Für den Prozess sollen die Massenströme \dot{m}_i und ihre Zusammensetzungen w_i unter Verwendung

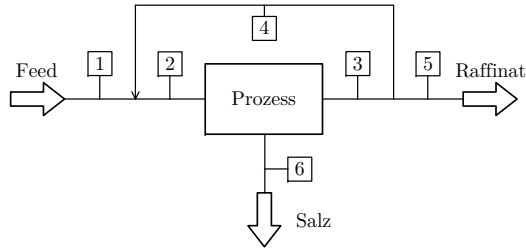


Abbildung 2.16: Prozess zur Abtrennung von Salz mit einer Rückführung

eines Zirkelbezugs in Excel berechnet werden. Gegeben sind der Massenstrom $\dot{m}_1 = 100 \text{ kg h}^{-1}$ mit einem Massenanteil an Salz von $w_{S1} = 0,30$. Außerdem wird vorgegeben, dass das Salz getrocknet abgezogen wird, also für den Massenanteil von Strom 6 gilt $w_{S6} = 1,00$. Für die Rückführung ist ein Verhältnis $v_R = \dot{m}_{S4}/\dot{m}_{S1} = 0,30$ und für die Ausschleusung des Salzes ein Verhältnis $v_A = \dot{m}_{S6}/\dot{m}_{S2} = 0,50$ einzuhalten. (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.17.)

Prozesstechnische Analyse der Aufgabenstellung: Es gelten die Bilanzgleichungen bei der Zusammenführung der Ströme 1 und 4

$$\dot{m}_2 = \dot{m}_1 + \dot{m}_4 \quad (2.35)$$

$$\dot{m}_{S2} = \dot{m}_{S1} + \dot{m}_{S4} \quad (2.36)$$

$$\dot{m}_{W2} = \dot{m}_{W1} + \dot{m}_{W4} , \quad (2.37)$$

bei der Aufteilung der Ströme 3 und 6

$$\dot{m}_2 = \dot{m}_3 + \dot{m}_6 \quad (2.38)$$

$$\dot{m}_{S2} = \dot{m}_{S3} + \dot{m}_{S6} \quad (2.39)$$

$$\dot{m}_{W2} = \dot{m}_{W3} + \dot{m}_{W6} \quad (2.40)$$

sowie bei der Aufteilung der Ströme 4 und 5

$$\dot{m}_3 = \dot{m}_4 + \dot{m}_5 \quad (2.41)$$

$$\dot{m}_{S3} = \dot{m}_{S4} + \dot{m}_{S5} \quad (2.42)$$

$$\dot{m}_{W3} = \dot{m}_{W4} + \dot{m}_{W5} . \quad (2.43)$$

Außerdem folgt die Summation der Massenanteile in den Strömen

$$w_{Wi} + w_{Si} = 1 . \quad (2.44)$$

Gegeben sind in der Aufgabenstellung die Verhältnisse

$$v_R = \frac{\dot{m}_{S4}}{\dot{m}_{S1}} \quad \text{und} \quad (2.45)$$

$$v_A = \frac{\dot{m}_{S6}}{\dot{m}_{S2}} . \quad (2.46)$$

Gesucht sind die Massenströme \dot{m}_i und ihre Zusammensetzungen w_{Si} . Mit den gegebenen Werten für den Massenstrom \dot{m}_1 , den Massenanteilen w_{S1} und w_{S6} sowie den Verhältnissen v_R und v_A ist die Aufgabenstellung eindeutig lösbar.

Für den Massenanteil des Salzes im Strom i gilt

$$w_{Si} = \frac{\dot{m}_{Si}}{\dot{m}_i} , \quad (2.47)$$

siehe dazu Tabelle A mit den Definitionen und Umrechnungen von Konzentrationsmaßen. Der Massenstrom i setzt sich aus den beiden Komponenten Wasser und Salz zusammen

$$\dot{m}_i = \dot{m}_{Wi} + \dot{m}_{Si} . \quad (2.48)$$

Für die Berechnung des Beispiels wird ein Excel-Berechnungsblatt wie in Abb. 2.17 angelegt:

1. Anlegen der Tabelle und Eintragen der in der Aufgabenstellung gegebenen Daten.
2. Stoffstrom 1: Für den Strom 1 werden der Massenstrom \dot{m}_1 und der Massenanteil w_{S1} im Berechnungsblatt gemäß Aufgabenstellung eingetragen. Mit Gleichung (2.47) lässt sich der Teilmassenstrom des Salzes \dot{m}_{S1} und mit Gleichung (2.48) der Teilmassenstrom des Wassers \dot{m}_{W1} berechnen.
3. Stoffstrom 2: Der Massenstrom \dot{m}_2 folgt aus Gleichung (2.35) mit der noch leeren Zelle für \dot{m}_4 als Vorbereitung des Zirkelbezugs. Analog dazu wird \dot{m}_{S2} mit Gleichung (2.36) berechnet. \dot{m}_{W2} folgt aus Gleichung (2.48) und w_{S2} aus Gleichung (2.47).
4. Stoffstrom 6: Der Teilmassenstrom \dot{m}_{S6} folgt aus Gleichung (2.46), \dot{m}_6 aus Gleichung (2.47) und \dot{m}_{W6} aus Gleichung (2.48).
5. Stoffstrom 3: Aus den Gleichungen (2.38) bis (2.40) folgen die Ströme \dot{m}_3 , \dot{m}_{S3} und \dot{m}_{W3} sowie w_{S3} aus Gleichung (2.47).
6. Stoffstrom 4: Für den Massenanteil von Stoffstrom 4 gilt $w_{S4} = w_{S3}$. Der Teilmassenstrom \dot{m}_{S4} folgt aus Gleichung (2.45) und der Massenstrom \dot{m}_4 aus Gleichung (2.47). Nach dem Eingeben dieser Gleichung in das Arbeitsblatt erscheint die Meldung „Zirkelbezugswarnung“. Um die iterative Berechnung durch einen Zirkelbezug zu ermöglichen, wird unter der Registerkarte **Datei** **Optionen** **Formeln** die Option „Iterative Berechnung aktivieren“ in den **Berechnungsoptionen** aktiviert. Der Massenstrom \dot{m}_{W4} folgt aus Gleichung (2.48).
7. Stoffstrom 5: Der Massenstrom \dot{m}_5 folgt aus Gleichung (2.41), der Teilmassenstrom \dot{m}_{S5} aus Gleichung (2.42), der Teilmassenstrom \dot{m}_{W5} aus Gleichung (2.43)

Massenströme und Massenanteile								
Nr.	<i>i</i>		1	2	3	4	5	6
Massenstrom	\dot{m}_i	kg h ⁻¹	100,0	169,0	149,5	69,0	80,5	19,5
Massenanteil Salz	w_{Si}	kg kg ⁻¹	0,30	0,23	0,13	0,13	0,13	1,00
Massenstrom Salz	\dot{m}_{Si}	kg h ⁻¹	30,0	39,0	19,5	9,0	10,5	19,5
Massenstrom Wasser	\dot{m}_{Wi}	kg h ⁻¹	70,0	130,0	130,0	60,0	70,0	0,0
Prozessparameter								
Verhältnis Rückführung	$v_R = \dot{m}_{S4} / \dot{m}_{S1}$	1	0,30					
Verhältnis Ausschleusung	$v_A = \dot{m}_{S6} / \dot{m}_{S2}$	1	0,50					

Abbildung 2.17: Excel-Berechnungsblatt für die Abtrennung von Salz mit einer Rückführung

und der Massenanteil w_{S5} aus Gleichung (2.47).

2.10 Ausgewählte Arbeitsblattfunktionen

Nachfolgend werden zwei ausgewählte Arbeitsblattfunktionen vorgestellt.

2.10.1 WENN-Funktion

Die WENN-Funktion ermöglicht eine Fallunterscheidung in einer Zelle im Arbeitsblatt:

```
=WENN(Prüfung; Dann_Wert; Sonst_Wert)
```

Prüfung ist ein Wert oder ein Ausdruck, der WAHR oder FALSCH sein kann. Wenn Prüfung WAHR ist, wird der Parameter Dann_Wert berechnet oder ausgegeben, andernfalls der Parameter Sonst_Wert. Es können Berechnungen ausgeführt oder Meldungen ausgegeben werden. Beispiel:

```
=WENN(p < p_tr; "Druck < Tripeldruck"; "Druck >= Tripeldruck")
```

Mit dieser Abfrage wird ermittelt, ob der Wert für den Druck in der Zelle p kleiner als der Tripeldruck in der Zelle p_tr ist.

WENN-Funktionen können auch verschachtelt werden:

```
=WENN(A1 > 100; 100; WENN(A1 < 0; 0; A1))
```

Oft ist die Verwendung der UND- oder der ODER-Funktion im Zusammenhang mit der WENN-Funktion sinnvoll. In Abschnitt 2.2 wurde beschrieben, wie durch Zuweisung von Namen an Zellen ein fester Zellbezug hergestellt wird. Eine Alternative zur WENN-Funktion kann die WAHL-Funktion sein, siehe Abschnitt 9.7.

2.10.2 RGP-Funktion

Wie in Abschnitt 2.7 beschrieben, ermöglicht Excel die Ermittlung von Ausgleichsfunktionen mit der Trendlinienfunktion, die den Nachteil hat, dass sie auf einer Diagrammdarstellung beruht und die Extraktion der Parameter der Ausgleichsfunktion

umständlich ist. Nachteil dieser und der nachfolgenden Methode ist die Einschränkung auf polynomische, logarithmische, Exponential- sowie Potenzfunktionen.

Beispiel 2.15

Für das reale binäre System Ethanol-Wasser soll für einen konstanten Druck $p = 1,013$ bar die Siedelinie durch ein Polynom als Ausgleichsfunktion auf Basis der RGP-Funktion beschrieben werden (vergleiche hierzu Beispiel 2.12 in Abschnitt 2.7.2). (Ergebnisse im Excel-Berechnungsblatt in Abb. 2.18.)

Die RGP-Funktion

`=RGP(Y_Werte; X_Werte; Konstante; Stats)`

basiert auf der GAUSSschen Methode der kleinsten Quadrate und gibt die Parameter eines linearen Trends in Form einer Matrix zurück. Excel erwartet die Gleichung

$$y = m_1x_1 + m_2x_2 + \dots + b. \quad (2.49)$$

y , x_i und m_i können Vektoren sein. Die Ergebnismatrix enthält in der obersten Zeile die Werte für m_n, m_{n-1}, \dots, m_1 und b . Zusätzlich können statistische Daten wie z. B. der Regressionskoeffizient ausgegeben werden.

Für Beispiel 2.15 soll ein Polynom sechsten Grades mit der Funktion

$$\vartheta = m_1x_1 + m_2x_1^2 + m_3x_1^3 + m_4x_1^4 + m_5x_1^5 + m_6x_1^6 + b \quad (2.50)$$

und darin x_1 als der Stoffmengenanteil der leichter siedenden Komponente Ethanol in der Flüssigphase verwendet werden. Die Gegenüberstellung mit Gleichung (2.49) ergibt $y = \vartheta$, $x_2 = x_1^2$, $x_3 = x_1^3$, $x_4 = x_1^4$, $x_5 = x_1^5$ und $x_6 = x_1^6$. Dementsprechend erfordert die Anwendung der RGP-Funktion eine Tabelle mit den Werten für ϑ und für die x_i^i für $i = 1$ bis zum Grad n des Polynoms, siehe Abb. 2.18. Die Werte in den beiden linken Spalten dieser Tabelle wurden Abb. 3.16 entnommen, die Werte in der äußersten rechten Spalte mit den – wie nachfolgend beschrieben – ermittelten Koeffizienten berechnet.

Vor dem Aufrufen der Funktion wird der Bereich für die Ausgabematrix im Arbeitsblatt markiert. Diese Matrix besteht aus fünf Zeilen und für Gleichung (2.50) aus sieben Spalten (siehe den grau umrandeten Bereich in Abb. 2.18). Für die y -Werte wird die Spalte mit den Werten für ϑ markiert und für die x -Werte der Bereich mit den x_1^i . Für **Konstante** wird **WAHR** eingegeben, womit b berechnet und nicht zu null gesetzt wird. Für **Stats** wird ebenfalls **WAHR** eingegeben, damit die Kenngrößen der Regression ausgegeben werden. Die RGP-Funktion ist eine sogenannte „Arrayformel“ und muss deshalb mit **Strg** + **⇧** + **Enter** statt einfach nur mit **Enter** bestätigt werden. Zu den Kenngrößen der Regression siehe die Hilfestellung in Excel.

Die maximale Abweichung zwischen den in der linken Spalte der Tabelle in Abb. 2.18 vorgegebenen Werten für die Temperaturen und den anhand der Polynomfunktion berechneten Werten in der rechten Spalte der Tabelle liegt unter 0,5 K. Die Abweichungen werden in dieser Tabelle aus Platzgründen nicht dargestellt, sind aber – wegen der

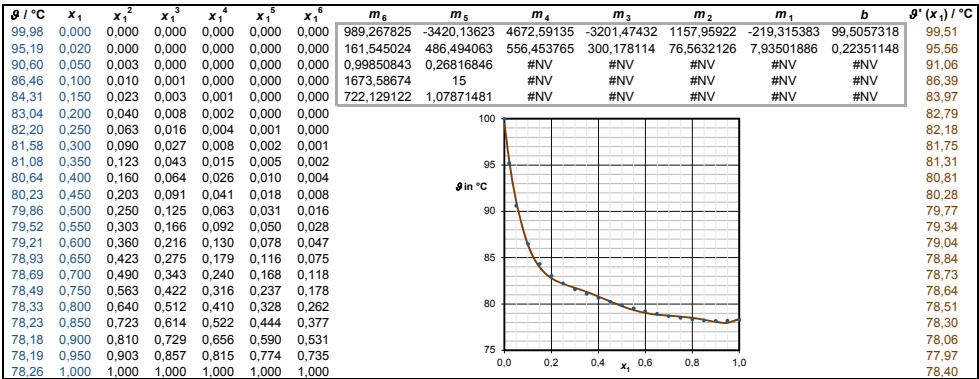


Abbildung 2.18: Excel-Berechnungsblatt für die Berechnung einer Polynomfunktion für die Siedelinie des realen binären Systems Ethanol-Wasser mit der RGP-Funktion

unterschiedlichen gewählten Grade der Polynome – in diesem Beispiel etwas größer als in Beispiel 2.12, vergleiche Abb. 2.13.

Verfahrenstechnik mit EXCEL

Verfahrenstechnische Berechnungen effektiv
durchführen und professionell dokumentieren

Feuerriegel, U.

2016, XVII, 381 S. 152 Abb., 94 Abb. in Farbe.,

Hardcover

ISBN: 978-3-658-02902-9