TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen, Institut für Energietechnik
Bitzer-Professur für Kälte-, Kryo- und Kompressorentechnik

Thema: *Dialogorientierte Bereitstellung von Parametern zur Approximation des Realgasverhaltens*

von: *LI, Chaobai*

vorgelegter Großer Beleg

Tag der Einreichung: 17.07.2017

(Platz für die Aufgabenstellung)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungs-, Symbol- und Indexverzeichnis

Abkürzungen

RGPKVA	Real Gas Properties - KVA
GKVA	Grafische KVA
KVA	Kolbenverdichter-Anlage
OOP	Objektorientierte Programmierung

Symbole

p	Druck
ρ	Dichte
R	Gaskonstante
T	Temperatur
h	Enthalpie
s	Entropie
κ	Isentropenexponent
K	Verhältnis von isobarer Wärmekapazität zu Gaskonstante
π	Stufendruckverhältnis

Indizies

ein	Eingang der Anlage
aus	Ausgang der Anlage
es	Eingang der Stufe
as	Ausgang der Stufe
U	Umgebung

1 Einleitung

In diesem Großen Beleg wird das Programm RGPKVA zur dialogorientierten Bereitstellung von Parametern des Realgasverhaltens beschrieben.

RGPKVA ist eine Erweiterung zu dem an der Bitzer-Stiftung-Professur der Technischen Universität Dresden entwickelten Computerprogramm GKVA. Das Programm GKVA als eine Benutzeroberfläche zur Eingabe der umfangreichen Eingangsparameter in das Programm KVA für die thermodynamische Nachrechnung von Kolbenverdichter-Anlagen, verfügt über eine Teilfunktion, die die Zustandsparameter(Temperatur, Druck, usw.) des Arbeitsmediums am Ein- und Ausgang der gesamten Anlage bzw. der Einzeln unter Idealgasbetrachtung berechnet[1].

Ziel von RGPKVA ist es, diese Teilfunktion so zu erweitern, dass neben der originalen Idealgasbetrachtung auch eine Ermittlung der Zustandsparameter unter Realgasbetrachtung mit Hilfe externer Programmbibliotheken möglich wird. Diese Ergebnisse sollen grafisch denen für den Idealgasfall gegenüber gestellt werden. Darüber hinaus werden die Parameter für die in KVA eingesetzten Algorithmen zur Approximation des Realgasverhalten ermittelt und deren Genauigkeit zwecks Überprüfung in Diagrammen dargestellt.

In diesem Beitrag werden zunächst die mathematischen Grundlagen für die in RGPKVA verwendeten Funktionen beschrieben. Danach wird die programminterne Logik von RGPKVA erläutert. Zum Schluss wird beschrieben, wie sich die Approximationsfehler bei unterschiedlichen Anfangsbedingungen verhalten.

2 Mathematische Grundlagen

Die verschiedene Funktionalitäten des Computerprogramms RGPKVA lassen sich mathematisch beschreiben. In diesem Kapitel werden hierzu folgende Algorithmen behandelt:

Die Festlegung und Darstellung der Ideal- und Realgasprozesse. Anhand einer vorgegebenen - im allgemeinen Fall mehrstufigen Verdichtung - werden für Idealgas und Realgas die Zustandsparameter am Ein- und Austritt der Stufen ermittelt. Darüber hinaus müssen so viele Zustandspunkte (p, T) berechnet werden, dass alle dazwischen liegenden Zustandsänderungen grafisch dargestellt werden können.

Ermittlung der Parameter zur Approximation von R und K des Realgases. Für Realgas wurden ein Potenzansatz in KVA zur vereinfachten Berechnung von (1) der Gaskonstante $R = \frac{p}{\rho T}$ und (2) dem Verhältnis K von isobarer Wärmekapazität zu Gaskonstante $K = \frac{h}{RT} = \kappa/(\kappa - 1)$ eingesetzt. Teils der Aufgabe von RGPKVA ist die Ermittlung der Parameter zu diesem Potenzansatz. Um die Genauigkeit dieser Approximationsmethode zu überprüfen, muss deren Ergebnisse über den Zustandsbereich ebenfalls mit den Realgaswerten verglichen werden. Die relativen Approximationsfehler werden ebenfalls berechnet und grafisch veranschaulicht.

2.1 Verdichtungsprozess

Die erste Aufgabe in RGPKVA ist die grafische Gegenüberstellung vom Verlauf der Zustandsparameter des Arbeitsmediums unter sowohl der Idealgas- als auch der Realgasbetrachtung.

Für beide Teilaufgaben ist es zuerst zweckmäßig, die relevante Zustandspunkte herauszufinden. Nachher wird es unterschieden, auf welcher Weise die Zustandsverläufe als diskrete Punkte $(p_\mathrm{i}, T_\mathrm{i})$ in ein $T(p)$-Diagramm eingetragen werden. Beim Idealgasprozess reicht die Verwendung von einfachen mathematischen Modellen, während für den Realgasprozess externe Stoffdatenbibliotheken eingesetzt werden müssen.

2.1.1 Festlegung des Druckniveaus

Der gesamte Prozess, der im RGPKVA berechnet wird, besteht aus isentropen Verdichtungen in der jeweiligen Stufe und anschließender vollständiger isobarer Rückkühlung zwischen dieser und der nächsten Stufen.

Die Drücke vor den isobaren Rückkühlungen werden aus dem Ein- und Ausgangsdruck der gesamten Anlage und der Stufenzahl als Elemente einer geometrischen Reihe i_st berechnet. Die Eingangsdrücke $p_{\mathrm{es,i}}$ der Stufen sind

$$p_{\mathrm{es},1} = p_\mathrm{ein}\pi^0 = p_\mathrm{ein}$$

$$p_{\mathrm{es},2} = p_\mathrm{ein}\pi^1$$

$$\vdots$$

$$p_{\mathrm{es},i_\mathrm{st}} = p_\mathrm{ein}\pi^{i_\mathrm{st}-1}$$

wobei

$$\pi = \left(\frac{p_\mathrm{aus}}{p_\mathrm{ein}}\right)^{\frac{1}{i_\mathrm{st}}}$$

das Stufendruckverhältnis wird für alle Stufen gleich angenommen. Der Ausgangsdruck einer Stufe ist der Eingangsdruck der nachfolgenden Stufe:

$$p_{\mathrm{as},1} = p_{\mathrm{es},2} = p_{\mathrm{ein}}\pi^1$$

$$p_{\mathrm{as},2} = p_{\mathrm{es},3} = p_{\mathrm{ein}}\pi^2$$

$$\vdots$$

$$p_{\mathrm{as},i_{\mathrm{st}}} = p_{\mathrm{ein}}\pi^{i_{\mathrm{st}}} = p_{\mathrm{aus}}$$

Weil der GKVA-Benutzer, die Stufendrücke nach erster Nachrechnung auch manuell außerhalb dieser Zahlreihe annehmen darf, wird diese beschriebene Festlegung in RGPKVA nicht bzw. nicht erneut durchgeführt. Stattdessen liest RGPKVA ausschließlich die Ergebnisse von GKVA mit.

2.1.2 Darstellung von Idealgasprozessen im $T(p)$-Diagramm

Zusätzlich zu den Drücken $p_{\mathrm{es,i}}$ und $p_{\mathrm{as,i}}$ in 2.1.1, werden zur Beschreibung der Idealgasprozesse auch die Temperaturen an den Ein- und Ausgängen aller Stufen als $T_{\mathrm{es,i}}$ bzw. $T_{\mathrm{as,i}}$ von GKVA nach RGPKVA übermittelt.

Es wird betrachtet, dass für Idealgas bereits die Kenntnisse über (p, T) zu Beginn und Ende des Prozesses eine grafische Darstellung in $T(p)$-Diagrammen ausreichen können. Bezeichnet man (p_1, T_1) als Eingangszustand und (p_2, T_2) als Endzustand, so gilt

1. die Prozesslinie als eine horizontale Grade, wenn $T_1 = T_2$ jedoch $p_1 \neq p_2$ ist. In diesem Falle wird ein isotherme Zustandsänderung dargestellt.

2. die Prozesslinie als eine vertikale Grade, wenn $p_1 = p_2$ jedoch $T_1 \neq T_2$ ist. Dies ist eine isobare Zustandsänderung.

3. die Prozesslinie als eine exponentielle Kurve, wenn sowohl $p_1 \neq p_2$ als auch $T_1 \neq T_2$ ist.

Im letzten Fall betrachtet man die Zustandsänderung als polytropisch. Generell gilt:

$$p_1\nu_1^n = p_2\nu_2^n = \mathrm{const.} \qquad (\text{Polytrope Zustandsänderung})$$

$$\frac{p_1\nu_1}{T_1} = \frac{p_2\nu_2}{T_2} = \mathrm{const.} \qquad (\text{Zustandsgleichung Idealgas})$$

Daraus folgt:

$$\frac{p_2}{p_1} = (\frac{T_2}{T_1})^{\frac{n}{n-1}} = (\frac{T_2}{T_2})^{\frac{1}{N}}$$

wobei $N = \frac{n-1}{n}$ eine Konstante ist. Die isentrope Zustandsänderung sieht man hier als einen Sonderfall mit $n = \kappa$ und $N = K$ an.

N lässt sich jetzt berechnen, durch

$$N = \frac{\ln(T_2/T_1)}{\ln(p_2/p_1)}$$

und für alle sonstigen Zustandspunkte auf dieser Kurve mit Zwischendrücke p_i, ergeben sich deren entsprechende Temperaturen

$$T_i = T_1(\frac{p_i}{p_1})^N$$

Somit erhält man hinreichende Zustandspunkte (p_i, T_i) zwischen (p_1, T_1) und (p_2, T_2), mit denen ein Idealgasprozess dargestellt werden kann.

2.1.3 Ermittlung und Darstellung von Realgasprozessen

Analog zu den Berechnungen mit Idealgas, wird RGPKVA auch die Ermittlung über Realgaszustände nach bekannten Drücken am Ein- und Ausgang aller Stufen veranlassen. Ziel ist die Bestimmung der Realgastemperatur zwischen Stufen, mit der weitere Parameter wie die R und K sich leicht herausfinden lassen.

Diese Ermittlung unterscheidet sich von den Idealgasberechnungen stark, indem anstatt mathematischen Modell die Stoffdaten direkt aus einer Computerbibliothek namens CoolProp unter Eingabe zweier bekannten Zustandsparametern als Randbedingungen abgefragt werden. Es ist allerdings zu erwähnen, dass technisch bedingt nicht jede Kombination von Zustandsparametern in CoolProp zum Erfolg führen kann. Während in Einphasengebiet für ein definiertes Gasgemisch eine Abfrage mit (p, T) meistens ohne Probleme absolviert werden kann, versagt diese mit (p, s)-Eingaben. Dieses Problem lässt sich durch eine Suche mit dem Newton-Raphson-Verfahren beheben.

Das Newton-Raphson-Verfahren zur Bestimmung der isentropen Zustandsänderung. Die Frage ist die Bestimmung der Endtemperatur in der isentropen Realgasverdichtung: für eine bestimmte Stufe i wird der Eingangszustand als $(p_{\mathrm{es,i}}, T_{\mathrm{es,i}})$ gegeben. Der Ausgangsdruck $p_{\mathrm{as,i}}$ wird in 2.1.1 festgelegt. Gesucht ist die Ausgangstemperatur $T_{\mathrm{as,i}}$ nach einer isentropen Verdichtung.

Mit $(p_{\mathrm{es,i}}, T_{\mathrm{es,i}})$ am Stufeneingang, erhält man per CoolProp-Abfrage typischerweise folgende Parameter für das Gasgemisch:

1. die Entropie s_i
2. die Enthalpie $h_{\mathrm{es,i}}$
3. die Dichte $\rho_{\mathrm{es,i}}$

woraus sich die Gaskonstante R und das Verhältnis K durch

$$R_{\mathrm{es,i}} = \frac{p_{\mathrm{es,i}}}{\rho_{\mathrm{es,i}} T_{\mathrm{es,i}}} \tag{2.1.1}$$

$$K_{\mathrm{es,i}} = \frac{h_{\mathrm{es,i}}}{R_{\mathrm{es,i}} T_{\mathrm{es,i}}} \tag{2.1.2}$$

berechnen lassen.

Am Stufenausgang kann CoolProp den Realgaszustand nicht durch $(p_{\mathrm{as,i}}, s_{\mathrm{i}})$-Anfrage beantworten. Die Lösung erfordert eine Umformulierung der Frage: bekannt ist ein Druck $p_{\mathrm{as,i}}$, wie findet man eine Temperatur T_{x}, damit nach einer Abfrage $s(T_{\mathrm{x}}) = \mathrm{CoolProp}(p_{\mathrm{as,i}}, T_{\mathrm{x}})$ die Bedingung

$$\frac{|s(T_{\mathrm{x}}) - s_i|}{s_{\mathrm{i}}} < \varepsilon \tag{2.1.3}$$

erfüllt, wobei ε die Toleranz an relativen Approximationsfehler heißt.

Definiert man eine Fehlerfunktion

$$e(T_{\mathrm{x}}) = s(T_{\mathrm{x}}) - s_i \tag{2.1.4}$$

so wandelt sich das o.g. Problem in die Suche nach der Wurzel von (2.1.4). Die ersten Abschätzung zu T_{x} kann von den Zustandsparametern am Stufenanfang mittels Gleichung für isentrope Verdichtung von Idealgas abgeleitet werden:

$$T_{\mathrm{x},0} = T_{\mathrm{es,i}}(\frac{p_{\mathrm{es,i}}}{p_{\mathrm{as,i}}})^{K_{\mathrm{es,i}}}$$

wobei $K_{\mathrm{es,i}} = \frac{\kappa_{\mathrm{es,i}}-1}{\kappa_{\mathrm{es,i}}}$ ist. Setzt man $T_{\mathrm{x},0}$ in Funktion (2.1.4) ein, ergibt sich der Approximationsfehler 1. Versuch:

$$e_0 = e(T_{\mathrm{x},0}) = s(T_{\mathrm{x},0}) - s_i$$

der nach einer Korrektur $\Delta T_{\mathrm{x},0}$ an $T_{\mathrm{x},0}$ beseitigt werden soll, d. h. es soll

$$e(T_{\mathrm{x},0} + \Delta T_{\mathrm{x},0}) \approx 0$$

werden.

Hierzu wird einen schrittweisen Versuch δT an Gleichung (2.1.4) gegeben, wonach man die Ableitung an der Stelle $T_{\mathrm{x},0}$

$$k_0 = \frac{\delta e(T_{x,0})}{\delta T} = \frac{e(T_{\mathrm{x},0} + \delta T) - e(T_{\mathrm{x},0})}{\delta T}$$

nährungsweise bekommt. Die Korrektur errechnet man unter der Annahme, dass k_0 bis zur nächsten Versuchsstelle mit $T_{\mathrm{x},1} = T_{\mathrm{x},0} + \Delta T_{x,0}$ geltend bleibt. Daher gilt die Beziehung:

$$\frac{e(T_{\mathrm{x},0} + \delta T) - e(T_{\mathrm{x},0})}{\delta T} = k_0 \approx \frac{e(T_{\mathrm{x},0} + \Delta T_{x,0}) - e(T_{\mathrm{x},0})}{\Delta T_{x,0}} = \frac{e(T_{\mathrm{x},1}) - e_0}{\Delta T_{\mathrm{x},0}} \approx \frac{-e_0}{\Delta T_{\mathrm{x},0}}$$

$$\Delta T_{\mathrm{x},0} = -\frac{e_0}{k_0}$$

Die neue $T_{\mathrm{x},1}$,$T_{\mathrm{x},2}$, ..., $T_{\mathrm{x,n}}$ werden somit wiederholend gebildet und in die Fehlerfunktion (2.1.4) eingesetzt, bis die Bedingung (2.1.3) zur Erfüllung kommt. Die letzte Temperatur $T_{\mathrm{x,n}}$ in dieser Reihe wird danach als die gesuchte Endtemperatur $T_{\mathrm{as,i}}$ isentroper Verdichtung angesehen. Zum Schluss werden die Zustandsparameter am Stufenausgang durch CoolProp-Anfrage $(p_{\mathrm{as,i}}, T_{\mathrm{as,i}})$ bestimmt, mit denen die Werte R und K analog zu den Gleichungen (2.1.1) und (2.1.2) (mit „as“ statt „es“ in Indizes) berechnet werden.

2.2 Approximation des Realgasverhaltens

Die zweite Aufgabe in RGPKVA ist die Durchführung der Approximationen für die Gaskonstante R und das Verhältnis K über bestehenden isentropen Zustandsänderungen des Realgases anhand der Ein- und Ausgangszuständen aller Stufen.

Es wird angenommen, dass sich die Werte R oder K exponentiell aus dem Druck- und Temperaturverhältnis zwischen Eingangs- und Endzustand sich ableiten lassen, d. h. es gilt die folgende Beziehung[2]:

$$\frac{R_1}{R_0} = (\frac{p_1}{p_0})^{a_R}(\frac{T_1}{T_0})^{b_R}$$

bzw.

$$\frac{K_1}{K_0} = (\frac{p_1}{p_0})^{a_K} (\frac{T_1}{T_0})^{b_K}$$

Die Koeffizienten a_R, a_K, b_R und b_K können mit bekannten Zustandsparametern an gewählten Zustandspunkten bestimmt werden. Über die gesamten Zustandsbereiche werden demnächst den Ablauf von R und K anhand dieser Beziehungen interpoliert.

Es wird zunächst betrachtet, dass für R und K die obere Beziehungen in folgendem Format verallgemeinert werden können:

$$\frac{f(p,T)}{f_0} = (\frac{p}{p_0})^a (\frac{T}{T_0})^b \tag{2.2.1}$$

wobei f für die beiden Werte R und K repräsentieren darf. Diese allgemeine Schreibweise wird in folgenden Text verwendet.

2.2.1 Bestimmung von a und b anhand 2 bekannten Zustandspunkten

Die Grundlagen der Approximation beziehen sich zuerst auf 2 Zustandspunkten: (p_L, T_L) und (p_H, T_H), wobei $p_L < p_H$ gilt. Da dieses Algorithmus hauptsächlich isentropen Verdichtungen annähert, wird zunächst angenommen, dass die Eingangstemperatur niedriger als die Endtemperatur liegt: $T_L < T_H$.

Zur Bestimmung von a und b wird zunächst die Gleichung (2.2.1) umgeformt:

$$\ln f - \ln f_0 = a(\ln p - \ln p_0) + b(\ln T - \ln T_0)$$

Diese Form stellt eine lineare Beziehung dar. Wird diese mit $z \propto \ln f$, $x \propto \ln p$ und $y \propto \ln T$ in einem 3D-Koordinatensystem aufgezeichnet, so erscheint die Gleichung eine Schrägebene, siehe Abb. 1.

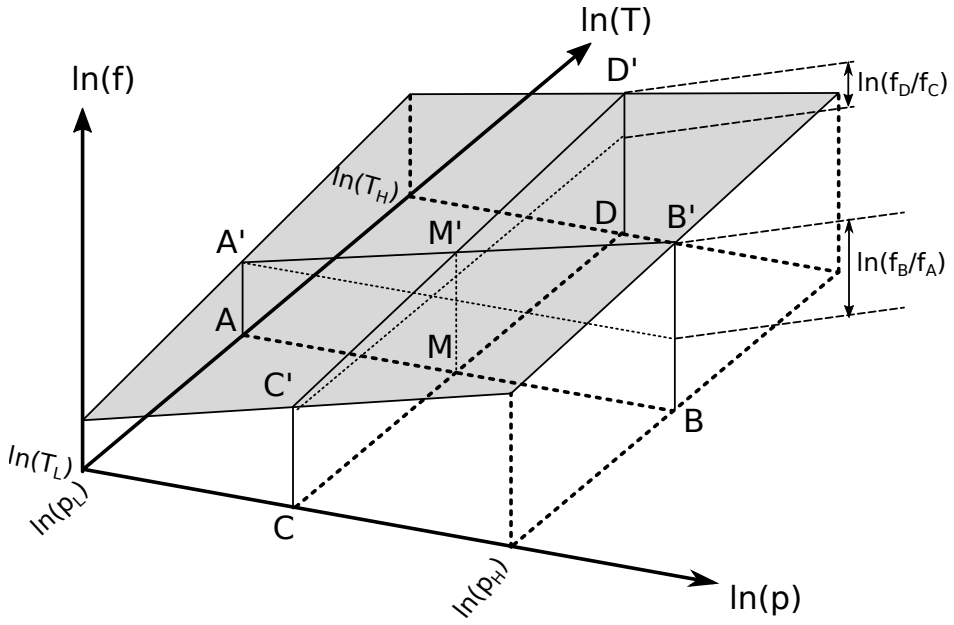


Abbildung 1: Darstellung der Gleichung (2.2.1) in 3D-Koordinatensystem

Auf Ebene $p - T$ bildet sich zwischen Punkten (p_L, T_L) und (p_H, T_H) ein Rechteck. Aus deren jeweiligen Seiten werden die geometrischen Mittelpunkten A, B, C, D festgelegt, woraus die Gerade AB und CD an der Stelle M schneidet. D. h. es gilt:

$$T_A = T_B = T_M = \sqrt{T_L T_H}$$

$$p_C = p_D = T_M = \sqrt{p_L p_H}$$

Die entsprechende Realgaswerte f_i werden nachher für $i = A, B, C, D, M$ durch externe Computerbibliothek gesucht, die die Punkte A', B', C', D' und M' feststellen.

An dem Punkt M ergibt sich das Koeffizient a als die Steigung der Grade A'B' in der Ebene AA'B'B. Da die Achse T auf diese Ebene senkrecht durchdringt, gilt $\ln T_B = \ln T_A$:

$$a = \frac{\ln f_B - \ln f_A - b(\ln T_B - \ln T_A)}{\ln p_B - \ln p_A} = \frac{\ln f_B - \ln f_A}{\ln p_H - \ln p_L}$$

Analog zu dieser Vorgehensweise lässt sich b aus Ebene CC'D'D ableiten:

$$b = \frac{\ln f_D - \ln f_C}{\ln T_D - \ln T_C}$$

Es wird wiederum hingewiesen, dass f die Abstraktion für Werte R und K steht. Deswegen werden in der Praxis für jede solche Approximation insgesamt 4 Koeffizienten abgeleitet: rp, rt, kp, kt, die sich jeweils für die Ableitung von R oder K nach p oder T handelt.

2.2.2 Approximation in 3 Zustandsbereichen mittels parabolischer Interpolation

Um bessere Genauigkeit zu erzielen, kann die in 2.2.1 beschriebene Methode so erweitert werden, dass über den Geltungsbereich der Wert f nicht durch eine ganze Ebene, sondern eine gekrümmte Fläche approximiert wird.

Zu dieser Betrachtung wird der Zustandsbereich, auf dem die Approximation durchgeführt wird, in 3 Regionen eingeteilt, siehe Abb. 2.

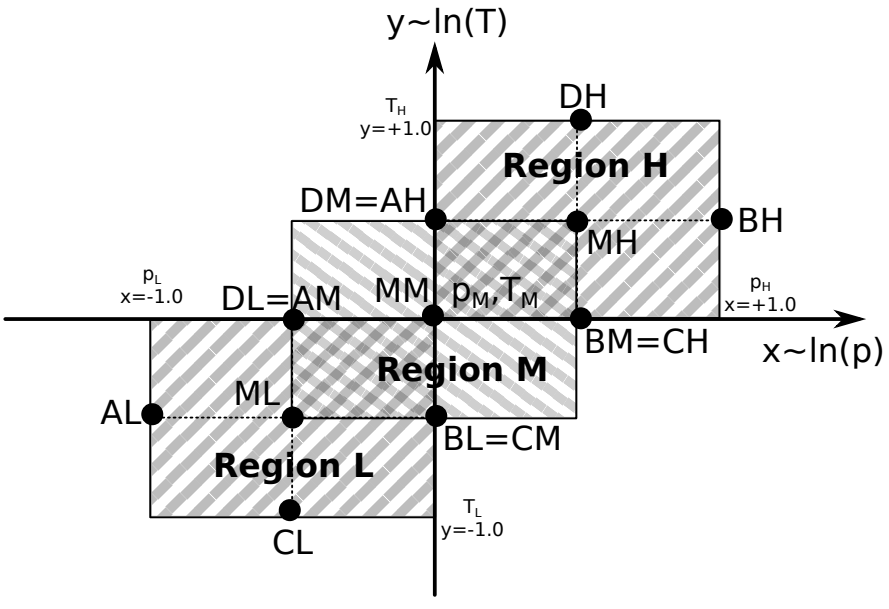


Abbildung 2: Dimensionslose Koordinaten und 3 Regionen zur Approximation

Dimensionslose Koordinaten In der Abbildung werden die dimensionslosen Koordinaten vorgestellt, die zunächst die Beschreibung von Drücken und Temperaturen der jeweiligen charakteristischen Punkten erleichtern. Aus dem maximalen und minimalen Druck p_H und

p_L berechnet man den Mitteldruck $p_M = \sqrt{p_L p_H}$. Der Abstand zwischen p_L und p_M gleicht den zwischen p_M und p_H auf einer logarithmischen Achse:

$$\ln(p_H) - \ln(p_M) = \ln \frac{p_H}{p_M} = \frac{1}{2}(\ln p_H - \ln p_L)$$

Der Punkt M wird zu Stelle $x = 0$ zugeordnet. Wenn x nach

$$x = f_{s,x} \ln \frac{p}{p_M}$$

definiert und $f_{s,x}$ mit

$$f_{s,x} = \frac{1.0}{\ln p_H - \ln p_M}$$

festgestellt wird, dann erhält man die Skala von -1.0 bis $+1.0$, auf der die Drücke wie folgt

$$p(x = -1.0) = p_L$$

$$p(x = 0) = p_M$$

$$p(x = +1.0) = p_H$$

abgelesen werden. Der gleiche Vorgang gilt auch für die T-Achse, wobei

$$f_{s,y} = \frac{1.0}{\ln T_H - \ln T_M}$$

und

$$y = f_{s,y} \ln \frac{T}{T_M}$$

gebildet werden.

Die 3 Regionen für Approximationen In der Abb. 2 auf der vorherigen Seite sind 3 rechteckige Regionen definiert:

1. Region L: von $x = -1.0$ bis $x = 0$, mit $y = -1.0$ bis $y = 0.0$
2. Region M: von $x = -0.5$ bis $x = 0.5$, mit $y = -0.5$ bis $y = 0.5$
3. Region H: von $x = 0$ bis $x = +1.0$, mit $y = 0$ bis $y = +1.0$

Für alle Regionen werden Approximationen nach 2.2.1 durchgeführt, sodass anstatt einen Wert nun für a, b und auch f jeweils 3 Punkten verfügbar sind: $\{f_i, a_i, b_i\} \quad i = \mathrm{L}, \mathrm{M}, \mathrm{H}$, die entsprechend als die Werte an den zentralen Punkten LM, MM oder HM angesehen werden sollen.

Approximation nach parabolischer Interpolation Für jeden Wert $g \in \{f, a, b\}$ wird aus deren 3 Punkten an der Stelle $g(-0.5)$, $g(0)$, $g(0.5)$ eine Parabel gebildet, mit der den Ablauf dieses Werts über dem gesamten Bereich $x \in [-1.0, +1.0]$ interpoliert wird:

$$g(x) = a_1 x^2 + a_2 x + a_3$$

Die Koeffizienten hierzu sind aus

$$a_1 = 2[g(-0.5) + g(0.5) - 2g(0)]$$

$$a_2 = g(0.5) - g(-0.5)$$
$$a_3 = g(0)$$

zu berechnen.

Zur Nachrechnung des Realgasverhaltens mittels Approximation, wird schließend zwei Geraden aus dem Zustandsbereich ausgewählt, die beiden in der Form von

$$y = x + d \qquad x \in [-1.0, +1.0]$$

angegeben sind.

Für die 1. Gerade bleibt stetig $d = 0$, für die 2. Gerade ist d frei zu wählen. Es ist zu bemerken, dass bei $d = 0$ die erste Gerade durch $(-1.0, -1.0)$ und $(+1.0, +1.0)$ läuft. Erinnernd die Definition der dimensionslosen Koordination $x \propto \ln p$ und $y \propto \ln T$, die zu

$$k_T \ln T = k_p \ln p \qquad k_T, k_p = \text{const.}$$

oder

$$k_T \ln \frac{T_L}{T_L} = k_p \ln \frac{p_H}{p_H}$$

$$\frac{T_H}{T_H} = (\frac{p_H}{p_L})^q \qquad q = k_p/k_T = \text{const.}$$

führt, dann wird aus der letzten Form erkennbar, dass diese Gerade tatsächlich eine polytropische (oder als Sonderfall: isentrope) Zustandsänderung von (p_L, T_L) nach (p_H, T_H) unter Idealgasbetrachtung darstellt, die als eine Referenz für den Vergleich von Ideal- und Realgasverhalten gut geeignet ist. Die erste Gerade dient deswegen als eine Bezugslinie.

Mit x gegeben können $p(x)$, $a(x)$, $b(x)$ und $f(x)$ berechnet werden. Bei der Interpolation von a, b und f wurden Stützwerte auf Punkten LM, MM, HM ausgewählt, die sich auf der Bezugslinie befinden. Daher sollen die Berechnungen hier ebenfalls als auf der Bezugslinie angesehen.

Die 2 Geraden geben zusätzlich die Werte T als $T_1(y_1 = x)$ (Bezugslinie) und $T_2(y_2 = x + d)$. Dadurch sind alle Angaben zur Approximation von f an (x, y_2) vorhanden:

$$f(x, y_2) = f(x, y_1)(\frac{T_2}{T_1})^{b(x)}$$

Zur Überprüfung von den Approximationsfehlern werden zusätzlich die Realgaswerte aus Computerbibliothek abgefragt. Der relative Approximationsfehler ist mit

$$e_f(x, y_2) = |1 - \frac{f_{\text{real}}(p(x), T_2(y_2))}{f(x, y_2)}| \times 100\%$$

ebenfalls eine Funktion von x (oder p), der sich grafisch darstellen lassen.

3 Technische Umsetzung

3.1 Programmierung

3.1.1 Python als Programmiersprache

Das Programm RGPKVA wurde in der gleichen Programmiersprache wie GKVA in Python 2 geschrieben. Python ist eine Programmiersprache mit vielen Vorteilen[3]:

Modern. Die Python-Sprache unterstützt relativ moderne Techniken für die Programmierung. Im Gegensatz zu FORTRAN oder C, die in wissenschaftlichen Bereichen ebenfalls häufig zum Einsatz kommen, werden in Python Technik wie die objektorientierte Programmierung, dynamische Typisierung, Garbage Collection (automatische Arbeitsspeicherbereinigung) usw. angeboten.
Auf Kosten einer reduzierten Effizienz hinsichtlich der Laufzeit, was für die Aufgaben von RGPKVA mit geringem Rechnungsaufwand teilweise zulässig ist, erhält der Programmer besser strukturierten Quellcode, der sowohl während der Programmierung für eine schnelle Fehlerbeseitigung als auch in der Zukunft für eine einfache Weiterentwicklung sehr geeignet ist.

Plattformunabhängig. Die Python-Sprache ist eine Interpretersprache. Die Quellcode in Python wird, im Vergleich zu den kompilierenden Sprachen wie FORTRAN und C, nicht in Maschinencoden, die je nach Hardwaren sich stark variieren können, übersetzt, sondern während der Laufzeit durch den Python-Interpreter eingelesen und ausgeführt. Der Interpreter abstrahiert somit den großen Unterschied zwischen Betriebssystemen und Hardwaren, und der gleiche Quellcode kann unverändert von der Plattform unabhängig laufen.

Lesbarkeit. Die Syntax von Python ist bewusst auf gute Lesbarkeit ausgelegt. Dabei werden in Python vorübergehend Schlüsselwörter aus englischer Sprache gegenüber anderen Programmiersprachen wie C, C++ usw. weniger Sonderzeichen(z. B.: `not` anstatt `!`) verwendet, mit denen das Programm ähnlich wie die Natursprache schreiben lässt. Anstatt Klammern `{}` wie C oder Java werden Codeblöcke in Python mit Einrücken unterschiedlicher Tiefe getrennt, die als eine zwingend vorgeschriebene Regel chaotische Quellcode vermeidet.

Einfach. Viele Syntaxerweiterungen (sog. „Syntaktischer Zucker“) werden in Python eingeführt. Diese Erweiterungen können den Arbeitsaufwand bei der Programmierung enorm erleichtern, zum Beispiel[4]:

lambda-Expression. Eine einfache Formel zur Berechnung einer Funktion $y(x) = x^2 + 2x + 1$ lässt sich wie folgt in Python programmieren:

```
y = lambda x: x ** 2 + 2 * x + 1
```

Gleiches Programm in C erfordert die Definition einer neuen Funktion plus vorhergehende Einbindung der mathematischen Bibliothek:

```
#include <math.h>
float y(float x){
        return a * pow(x, 2) + b * x + c;
}
```

Liste. Die Liste als eine Datenstruktur in Python erfüllt analog zu einem Array anderer Programmiersprache die Aufgabe, mehrere Objekte in eine Reihe zu speichern. Der Umgang mit Listen ist jedoch mit Syntaxerweiterungen sehr gut vereinfacht. Um die Liste als eine Zahlenfolge von 1...5 zu initialisieren, wird nur eine Codezeile gebraucht:

```
xn = range(1, 6)    # danach: xn == [1, 2, 3, 4, 5]
```

Wenn ferner aus `xn` und der oben beschriebener lambda-Expression `y` eine weiter Liste `yn` zu erzeugen ist, schreibt man nur eine weitere Zeile:

```
yn = [y(xi) for xi in xn]  # danach: yn == [4, 9, 16,
    25, 36]
```

3.1.2 Externe Computerbibliotheken

Zusammen mit Python wird nicht nur der Interpreter, sondern auch umfangreiche Programmbibliotheken geliefert. Neben den Standardbibliotheken, die bereits für Aufgaben wie den Umgang mit Betriebssystem, Dateien, Netzwerk oder sogar eine eigene grafische Benutzeroberfläche konzipiert wurden, sind auch zahlreiche Bibliotheken von dritten Anbietern erhältlich.

Für diese Arbeit sieht man hierzu folgende Bibliotheken vor, die die verschiedene Anforderungen von RGPKVA ermöglicht:

1. `wxPython` ist eine Bibliothek für die Benutzeroberfläche. Sie umfasst sehr umfängliche Funktionen, die fast alle herkömmliche Benutzeroberflächen im modernen Betriebssystemen realisieren können, und erzeugt gleichzeitig visuell vertraute Ergebnisse.

2. `numpy` und `scipy` sind Bibliotheken, in denen die für wissenschaftliche Berechnungen wichtige Funktionalitäten und Algorithmen integriert wurden.

3. `matplotlib` ist eine nützliche Bibliothek für grafische Darstellung und arbeitet auch mit `numpy` reibungslos zusammen. Außerdem bietet sie eine Einbindung mit `wxPython`, sodass die Diagramme in die Benutzeroberfläche nahtlos eingebaut werden können.

4. `CoolProp` ist eine Bibliothek zur Berechnung von Stoffdaten. Sie wurden unter der MIT-Lizenz freigegeben, die eine Nutzung unter Angaben originaler Lizenzbestimmung gebührenfrei und uneingeschränkt genehmigt. `CoolProp` beinhaltet Stoffdaten von über 100 Fluide[1], und erlaubt Abfragen von Eigenschafts- oder Zustandsparameter sowohl für Reinstoffe als auch für vordefinierte Stoffgemische.

3.1.3 Objektorientierte Programmierung

Das Programm RGPKVA wurde in der Weise sog. objektorientierten Programmierung (OOP) geschrieben. Hiermit bezeichnet man ein Typ von Programmierparadigma, wie die Daten und Logik im gesamten Programm organisiert werden.

[1] Zum Stand der Arbeit: 122

In einer traditionellen Programmierung wird typischerweise der Ablauf des Programms festgelegt. Ein gewöhnliches Programm könnte so aussiehen: zuerst wird der Nutzer abgefragt, oder werden Eingaben aus verschiedenen Art und Weisen gesammelt. Danach werden anhand der Daten Berechnungen nacheinander veranlasst, zwischendurch wird der Verlauf wegen Erfüllung oder Nichterfüllung vorgeschriebener Bedingungen zurück-/fortgesetzt oder abgebrochen. Solches Programmierung mit festgeschriebener Prozedur nennt man Prozedurale Programmierung.

Ein Problem bei prozeduraler Programmierung ist, dass in solchen Programmen die Daten (gespeichert in Variablen) erst durch eine Prozedur geändert werden müssen. Meistens werden solche Variablen auch als globale Variablen definiert, deren Verwaltung bei großen Programmen außerst schwierig ist. Bei zunehmender Komplexität, wo das Programm gleichzeitig mehrere Aufgaben (Bedienen durch Nutzer an der Benutzeroberfläche, die Durchführung der Algorithmen im Hintergrund, usw.) berücksichtigen muss, wandelt sich dieses Programmierparadigma schnell in einen Albtraum.

Die OOP-Paradigma. In OOP-Programmierung organisiert man die funktional benachbarten Daten und Logik in Objekten. Das gesamte Programm wird in OOP in klare Aspekten eingeteilt. Jedem Objekt einschließlich seinen internen Variablen wird der Interpreter individuell Arbeitsspeicher zuweisen, und jedes Objekt besitzt eine Menge vordefinierter „Methoden“, die auf die objekt-eignen Daten zurückgreifen und durch außere Quellcode aufrufen lassen. So ist z. B. ein Fenster der Benutzeroberfläche ein Objekt, und das Algorithmus der Berechnung ein anderes Objekt. Wenn der Nutzer die Eingabe im Fenster aktualisiert, erfolgt die Bearbeitung beispielsweise die Wiedergabe in der Benutzeroberfläche und die Überprüfung zuerst ebenfalls innerhalb des gleichen Objekts. Das Berechnungsobjekt wird erst aufgerufen, wenn diese Eingable als akzeptabel angesehen.

Prototyp und Vererbung. Objekte sind in der Praxis nicht durch Quellcode direkt definiert. Die Quellcode im OOP schreibt hierzu ein Prototyp („`class`“ oder „Klasse“) vor, der als ein Muster für Objekte die Variablen und Methoden festlegt. Objekte als „lebende“ Datenstruktur werden nach Versorgung von relevanten Parametern an einen Prototyp durch Interpreter erzeugt.

Ein Prototyp kann von Grund auf programmiert werden, oder von einem anderen Prototyp vorerst durch „Vererben“ die gemeinsam genutzte Logik kopieren, worauf man anschließend nur noch den Unterschied hinzufügen muss. Mit der letzteren Weise baut man eine Hierarchie von Prototypen, die auch den abstrahierten Programmaufbau reflektiert. Diese Technik wird in RGPKVA verbreitet verwendet, die im folgenden Kapiteln näher beschrieben wird, siehe 3.2 und 3.3.2.

3.1.4 Modulare Organisation der Quellcode

Das Quellcode von RGPKVA wird nach den Aufgaben in mehreren Dateien und Ordnern eingeteilt, die in Python als Modulen angesehen werden. Für das Programm GKVA dient RGPKVA selbst ebenfalls als ein Modul.

Die Dateienstruktur von RGPKVA sieht wie folgt aus:

1. `rgpkva/` ist der Wurzelpfad vom RGPKVA. Dieser Ordner wird mit neben der originalen Quellcode von GKVA am gleichen Pfad gelegt, sodass innerhalb GKVA einen direkten Aufruf und Datenübermittlung an RGPKVA möglich ist.

2. `rgpkva/gui` ist ein Modul von RGPKVA, das mit den Aufgaben von grafischen Benutzeroberflächen (englisch „GUI-Graphic User Interface“) sich beschäftigt.

3. `rgpkva/calc` umfasst alle mit der Berechnungen relevante Quellcoden. Die werden im Lauf des Programms von Seite der Benutzeroberfläche gerufen, und veranlassen selbständig die weiteren Schritte zur Berechnungen. Die Ergebnisse werden auch in diesem Modul gespeichert, die durch eine Schnittstelle an GKVA sortiert zurückgegeben werden.

3.1.5 Events

Mit „Events“ wird eine programmiertechnisch Technik bezeichnet, die im Programm RGPKVA eingesetzt wurde. Das Quellcode, das sich in der Datei `rgpkva/helper.py` befindet, hat auf den ersten Blick nur 4 Zeilen von Bedeutung, nichtsdestotrotz wird es allerorts in RGPKVA gebraucht und dient als ein „Kleber“, mit der die verschiedenen Teile des Programms zusammenarbeiten:

```
class Event(list):
    def __call__(self, *args, **kwargs):
        for f in self:
            f(*args, **kwargs)
```

Die Klasse `Event` wird als eine Unterklasse von `list` geerbt, die letzte in Python eine native Klasse ist. Darauf wird Python hingewiesen, dass ein Objekt von `Event`-Klasse ohne Weiteres gleiche Funktionalität wie ein `list` besitzen muss, nämlich das Beherbergen beliebig vieler sonstiger Objekte.

Der einzige Unterschied zwischen der Unterklasse `Event` und ihrer Basisklasse `list`, wird durch die Definition einer speziellen Methode `__call__` verdeutlicht. Nach [5] wird eine Klasse wegen des Hinzufügens von dieser Methode für sonstigen Codes wie gewöhnliche Funktionen aufrufbar gemacht.

Nachdem ein Aufruf ankommt, iteriert die `for`-Schleife von Zeile 3 über alle Funktionen, die in `Event` vorher eingetragen worden sind, und ruft sie nacheinander mit gleichen Argumenten weiter auf (Zeile 4), wie der Code hier selbst beim Aufruf erhielten.

In anderen Worten funktioniert die Klasse `Event` wie eine Funkstation: ein Teilprogramm bereitet sich für die Interaktion mit sonstigen Codes einen oder mehreren `Event` vor, und ruft diese nach seiner eigenen Logik. Die andere Teilprogramme „abonnieren“ diesen `Event` , und erhalten die Anrufe gleichzeitig mit Daten gleicher Formaten. In dieser Weise kann ein Teil des Programms Signale an andere Teile senden, ohne wissen zu müssen, welche Teile hierzu zum Empfangen gebunden sind.

Diese Technik hat den Vorteil, dass zwischen den Teilprogrammen keine enge Verbindung, als man traditionell für jede solche explizit programmieren muss, nötig ist, die zur komplizierten Struktur führt und der Fehlerbeseitigung problematisch ist. Sollte das gesamte Programm mit neuen Aufgaben erweitert werden, braucht man auch nicht in die vorhandenen Quellcodes einzuarbeiten, sondern nur der neuen Code auf vorhandenen `Event` anschließen.

3.2 Benutzeroberfläche

Es wird in diesem Kapitel die Umsetzung der Benutzeroberfläche in RGPKVA vorgestellt. Die Benutzeroberfläche ist mit der Computerbibliothek `wxPython` in Python geschrieben.

wxPython sieht vor, dass die Benutzeroberfläche aus verschiedenen Steuerelementen aufgebaut wird, wobei manche Steuerelemente gleichzeitig als Behälter weiterer Elementen dienen und auch ihre visuellen Größen bestimmen. In diesem Sinn lässt sich die Struktur der Benutzeroberfläche mit Abb. 3 auf der nächsten Seite erklären.

3.2.1 `Panel RGPKVA`: RGPKVA als ein Panelelement und eine Schnittstelle

In der Abbildung 3 sieht man, dass die Benutzeroberfläche RGPKVA grundsätzlich als ein Panelelement definiert wird. Ein Panel ist gegenüber einem Fenster zu unterscheiden: ein Fenster ist ein selbständiges Element, das mit Titel, Grenzen und Tasten zur Maximierung, Minimierung oder Ausschaltung versehen ist und sich individuell im Bildschirm darstellen lässt, während ein Panel nur einen Behälter von weiteren Kind-Elementen innerhalb eines Fensters ist.

Ein Panel kann nicht direkt auf den Bildschirm dargestellt werden. Der kann und muss als Inhalt zuerst in einen Fenster oder sonstige Behälterelemente, in unserem Fall vorzugsweise einen Reiter in GKVA, eingebunden werden.

Dieser Aufbau bestimmt das Programm RGPKVA eine Erweiterung(englisch „Plug-in“) zu GKVA: von GKVA wird nur wenige Quellcode benötigt, um RGPKVA als ein zusätzliches Element in seine Benutzeroberfläche hinzuzufügen.

Programmiertechnisch angesehen, ist RGPKVA eine Unterklasse geerbt aus der Klasse `Panel` von der Bibliothek `wxPython`. Zusätzliche Quellcode für die Versorgung und Rückmeldung der Parameter, die durch beide Programme zu berechnen sind, wird dazu programmiert.

Dieser Teil befindet sich unter `rgpkva/gui/__init__.py`, für GKVA bedeutet dieser Pfad den Ausgangspunkt, RGPKVA zu initialisieren.

3.2.2 „Splitter“: Einteilung der Benutzeroberfläche in 4 Quadranten

Die Fläche auf dem Panel RGPKVA wird zunächst in 4 Quadranten eingeteilt. Es wird für das Panel ein vertikaler Splitter („Trenner“), der die Benutzeroberfläche zuerst nach links und rechts in 2 Seiten schneidet, programmiert. Für die jeweilige Seite programmiert man ein weiteren horizontalen Splitter, wodurch das Panel am Ende in insgesamt 4 Quadranten eingeteilt wird, die für folgende Inhalten bestimmt sind:

1. links oben: ein Platz für die Tabelle mit allen Zustandsparametern usw., der durch eine Klasse `HTMLDisplay` realisiert wird.

2. links unten: die Eingabefelder für alle Einstellungen zu RGPKVA, die von einem modifizierten Gitterelement `SettingsGrid` beherbergt werden.

3. rechts oben: ein Menü zur Auswahl von bis zu maximal 2 Diagrammen, die später darzustellen sind. Es wird hierzu zuerst ein Steuerelement `ListBox` aus wxPython modifiziert, das ursprünglich eine Liste bereitstellt und dem Nutzer zum Wählen ermöglicht. Die Modifikation wird als Unterklasse `RestrictedListBox` benannt und beschränkt die Anzahl der wahlbaren Optionen bis auf maximal zwei Einträge. Danach wird dieses Element in ein Panel namens `PlotDisplayController` eingebunden.

4. rechts unten: ein Panel zur grafischen Darstellung basierend auf `matplotlib`. Diese Bibliothek stellt 2 wxPython-kompatiblen Steuerelemente zur Verfügung, die in einem Panel `PlotDisplay` integriert werden.

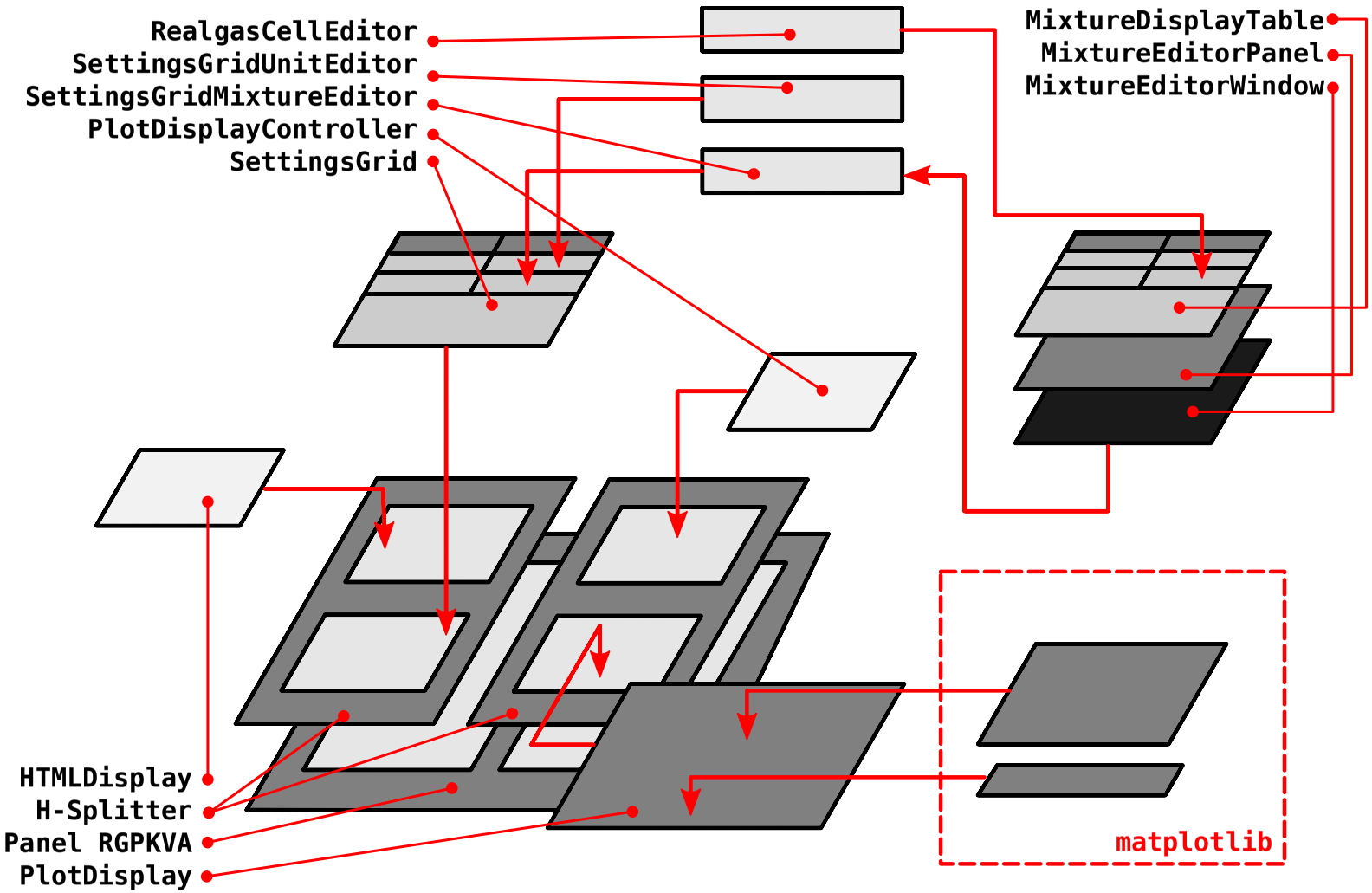


Abbildung 3: Abstrahierter Aufbau der Benutzeroberfläche RGPKVA

In wxPython haben die Splitter zusätzlich als Behälter eine wichtige Funktion, nämlich die Bestimmung der visuellen Größen aller internen Steuerelementen. Die Größe des von Splitter erzeugten Plätzen sind mit der Maus verstellbar, sodass bei Bedarf ein Quadrant, beispielsweise die Diagrammdarstellung vergrößert bzw. verkleinert werden kann.

3.2.3 `MixtureEditorWindow`: Dialogfenster zur Definition des Gasgemisches

Rechts oben in der Abbildung 3 wird ein Fenster `MixtureEditorWindow` aufgezeichnet. Dieser ist neben dem Panel RGPKVA das einzige Element, das in einem eigenen Fenster präsentiert wird, siehe Abbildung 4.

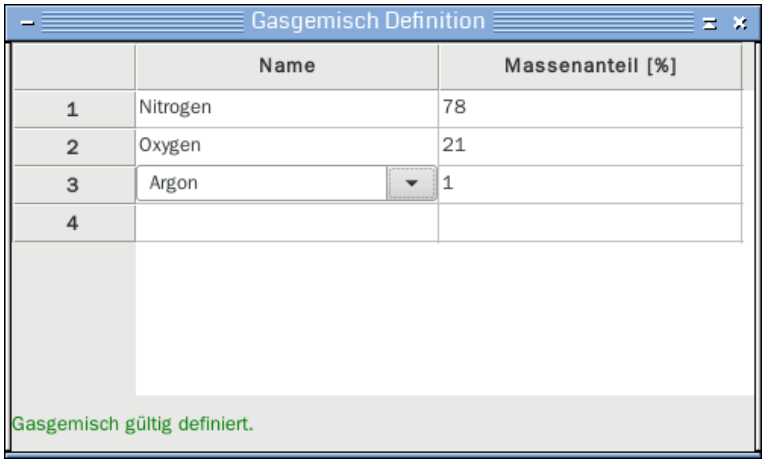


Abbildung 4: `MixtureEditorWindow`

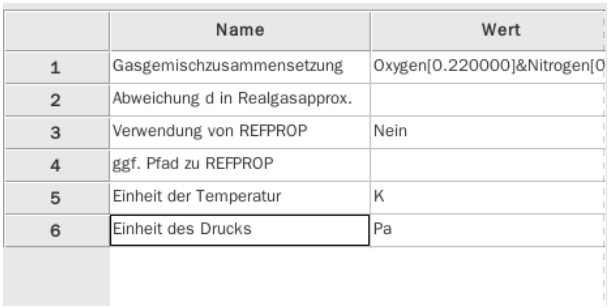
Der `MixtureEditorWindow` mit einem Panel `MixtureEditorPanel` ist ein vollständiges Fenster, worin sich ein Gitterelement `MixtureDisplayTable` Platz findet. Alle Felder in diesem Gitterelement können editiert werden. Um das Programm schnell und aussschließlich fehlerfreie Daten zu bekommen, werden den Feldern Eingabehilfen (siehe 3.2.4) programmiert, die die Gültigkeitsregeln durchsetzen.

3.2.4 „Editor“: Maßgeschnittene Eingabehilfen

Für die Bearbeitung von komplizierten Einstellungen, werden in RGPKVA 2 Gridelemente (aus Klasse `Grid` von `wxPython`) eingesetzt: das Element `SettingsGrid` erfasst wichtige Einstellungen für die Darstellung und Berechnung(Abb. 5 auf der nächsten Seite), und `MixtureEditorWindow`(Abb. 4) die Komponenten und deren Massenanteilen des zu definierenden Gasgemisches.

Viele Felder in diesen Gridelementen unterliegen für ihre Gültigkeit eigene Randbedingungen:

- In `SettingsGrid`:
  - die „Gasgemischzusammensetzung“ muss gemäß der Bezeichnungsregel von CoolProp als eine Zeichenfolge gegeben werden.
  - die Optionen für Einheitseinstellungen, die dem Nutzer angeboten werden und die Diagrammdarstellungen in RGPKVA steuern, müssen mit dem RGPKVA-internen Code (das Einheitssystem, siehe 3.3.2) übereinstimmen.



	Name	Wert
1	Gasgemischzusammensetzung	Oxygen[0.220000]&Nitrogen[0
2	Abweichung d in Realgasapprox.	
3	Verwendung von REFPROP	Nein
4	ggf. Pfad zu REFPROP	
5	Einheit der Temperatur	K
6	Einheit des Drucks	Pa

Abbildung 5: `SettingGrid` auf der Benutzeroberfläche

- In `MixtureEditorWindow`: Die Bezeichnung von Einzelkomponenten müssen der Liste von CoolProp entsprechen.

Es ist generell benutzerfreundlich, wenn man diese Randbedingungen nicht erst nach der Eingabe prüfen und mit Fehlermeldungen den Benutzern zur Änderung zu zwingen, sondern durch spezielle Maßnahmen schon während dem Eingabenvorgang auf nur gültigen Ergebnisse zu gewährleisten. Solche Maßnahmen werden durch die Anwendung von `GridCellEditor` realisiert, die als Eingabehilfen den Nutzer in geeigneter Weise abfragen und nur korrekten Eingaben erzeugen.

Zu diesem Zweck wird in der Praxis eine Reihe von Unterklassen aus `GridCellEditor` abgeleitet und dem Gridelement versorgt:

1. `RealgasCellEditor` als ein Combobox-Steuerelement, für das nur eine Option gewählt werden kann. Die verfügbare Optionen werden nach Abfrage an CoolProp festgelegt, damit die Bezeichnungen immer korrekt sind.

2. `SettingsGridMixtureEditor` schaltet nach Starten einer Editierung sofort das Fenster `MixtureEditorWindow` (3.2.3) ein und wartet auf seine Rückmeldung über eine gültige Gasgemischbezeichnung. Die neue Bezeichung als eine Zeichenfolge wird danach automatisch eingetragen. Eine manuelle Eingabe ohne Fenster wird ausgeschlossen.

3. `SettingsGridUnitEditor` verknüpft sich mit dem internen Einheitssystem (3.3.2) von RGPKVA. Je nach der physikalischen Größe werden nur die passende Einheiten zur Auswahl aufgelistet.

3.2.5 `HTMLDisplay`: Darstellung der Zustandsparameter in Tabellen

Zur Gegenüberstellung der Zustandsparameter für Ideal- und Realgas, wurde alle Zahlen in eine Tabelle während der Berechnung eingetragen, siehe Abb. 6.

Aufgrund von der Komplexität der Tabelle, die statt einem einfachen Gitter mit Zellen über mehreren Zeilen(in der Abbildung z. B. die Schrifte „Eingang“, „K“, usw.) versehen ist, wird diese Darstellung mit HTML realisiert. HTML(englisch „Hypertext Markup Language“) ist die primäre Auszeichungssprache, die ein struktuiertes Dokument beschreibt. HTML wird überwiegend im Internet zur Übermittlung von Webseiten eingesetzt, findet jedoch ihre Nutzung auch ohne Internet.

HTML lässt sich in `wxPython` durch 2 Möglichkeiten wiedergeben: `WebView` oder `HTMLWindow`. `WebView` ist ein voll funktionstüchtiger Internetbrowser, der neben Wiedergabe von HTML-Dokumente auch mit sonstigen dynamischen Funktionalitäten, beispielsweise die Ausführung von JavaScript-Code, ausgestattet wird. Zum Vergleich verfügt

Stufe			1	2	3
Eingang	p [Pa]		103000.00	242848.86	572578.35
	T [K]	ideal	333.00	333.00	333.00
		real	333.00	333.00	333.00
	R [J/kg/K]	ideal	285.00	285.00	285.00
		real	287.03	287.00	286.94
	K	ideal	3.50	3.50	3.50
		real	3.50	3.49	3.46
Ausgang	p [Pa]		242848.86	572578.35	1350000.00
	T [K]	ideal	425.47	425.47	425.47
		real	424.90	424.99	425.21
	R [J/kg/K]	ideal	285.00	285.00	285.00
		real	287.22	287.45	288.02
	K	ideal	3.50	3.50	3.50
		real	3.53	3.51	3.48

Abbildung 6: in `HTMLDisplay` dargestellte Tabelle

ein `HTMLWindow` über nur eine Teilmenge aus dem HTML-Standard, der sich jedoch unabhängig von der Installation eines Internetbrowsers durch `wxPython` aufrufen lassen, und benötigt daher auch deutlich wenigere Arbeitsspeicher.

Die Darstellung der Zustandsparameter nutzt hier ein Teil der HTML-Sprache für die Auszeichnung einer Tabelle, und benötigt keine JavaScript-Unterstützung. Aus diesem Grund wird in RGPKVA ein Steuerelement von `HTMLWindow` eingesetzt, das danach mit einem `Panel` eingebunden und anschließend im Platz rechts oben in der Benutzeroberfläche präsentiert wird.

3.3 Dienst zu Berechnungen

Im Programm RGPKVA werden die Berechnungen getrennt von der Benutzeroberfläche programmiert. Die Programme hierzu bilden ein weiteres Modul unter dem Pfad `rgpkva /calc/`. Für die Benutzeroberfläche wird beim Starten eine Klasse `Calculator` initialisiert, die die tatsächliche Eingabe vom Nutzer und GKVA einliest, berechnet, und die Ergebnisse zurück an der Benutzeroberfläche und dem GKVA bereitstellt. Aus dieser Betrachtung wird die Klasse `Calculator` sämtlich ihrer eigenen Erweiterungen als einen Dienst im Kapiteltitel genannt.

3.3.1 Aufgaben des Berechnungsdiensts

Bevor die technische Umsetzung erläutert werden kann, ist es zweckmäßig, dem Leser alle Aufforderungen für den Berechnungsdienst vorzustellen, sodass die Notwendigkeit der eingesetzten Techniken ersichtlich zur Verständigung kommt.

Die Berechnungsaufgaben durchführen. Für RGPKVA sind in der Praxis folgende Aufgaben nach Möglichkeit zu erzielen:

1. Detaillierte Berechnung über Idealgasprozesse. Obwohl werden für alle Stufenein- und ausgänge Zustandsparameter des Idealgasprozesses von GKVA angegeben, ist es für die grafische Darstellung unvermeidbar, die Zustandspunkte innerhalb einer bestimmten Zustandsänderung zusätzlich herauszufinden, damit auf Diagramme kontinuierlichen Kurven aufgezeichnet werden können.

2. Bestimmung des Realgasprozesses. Die zwischenstufige Zustandsparameter im Falle des Realgasprozesses müssen von RGPKVA komplett bestimmt werden. Nachher sind für die Diagramme ebenfalls Zustandspunkte für jede individuelle Zustandsänderung festzustellen, wobei die genaue Vorgehensweise sich jedoch sehr enorm von Aufgabe 1 unterscheidet.

3. Die Approximation anhand den Befunden aus Aufgabe 2 einleiten, und die Ergebnisse gleichwohl als diskrete Punkte für Darstellungen bereitzustellen.

Die Berechnung nach Erfüllung und Änderung der Abhängigkeiten veranlassen. Es sind zu betrachten, dass die oben genannte Aufgaben inhaltlich sich voneinander trennen lassen, jedoch gleichzeitig von den Ergebnissen voneinander abhangen. So ist zum Beispiel die Approximation, die zwar anderes Algorithmus verwendet, kann aber nur erst starten, wenn die Realgaszustandsparameter schon bestimmt sind.

Zusätzlich sind diese Aufgaben erneut zu starten, wenn der Nutzer im RGPKVA oder GKVA die Eingaben ändert. Sollte die stufigen Druckniveaus somit betroffen, dann müssen alle Aufgaben—Idealgas- und Realgasprozesse plus Approximation—neue berechnet werden. Wird jedoch nur die Idealgaskostante R geändert, dann muss auch nur die Idealgasprozesse aktualisiert werden. Eine Änderung an die Definition der Realgaszusammensetzung berührt zwar keine Idealgasprozesse, muss allerdings zu neuen Berechnung des Realgasprozesses und der Realgasapproximation führen.

Dem Programm RGPKVA sind in diesem Sinn so zu organisieren, dass alle Berechnungsaufgaben erst und immer durchzuführen, wenn deren Abhängigkeiten an anderen Daten erfüllt oder geändert werden.

Eingaben aus GKVA filtern. Das Programm GKVA verwendet eine periodische Methode für seine interne Datenverarbeitung und -übermittlung. Mit den Daten über stufigen Drücke, Idealgasparameter und sonstige Einstellungen wird RGPKVA ungefähr alle 3 Sekunden aufgerufen. Diese Strategie kann zum Problem führen, wenn RGPKVA ebenfalls nach jeweiliger Aktualisierung neue Berechnungen veranlassen: denn auf einem normalen Computer kosten die komplette Realgasberechnung und -approximation einen Zeitaufwand von einer bis mehreren Sekunden, was, wenn periodisch wiederholt wird, die Nutzerinteraktion blockieren kann.

Der Benutzeroberfläche über Aktualisierungen informieren. Der Berechnungsdienst verwaltet neben den Ergebnissen aller Algorithmen auch die Einstellungen für die Benutzeroberfläche, vorallem die Einheiten zur Darstellung. Zu jedem Zeitpunkt wird die Benutzeroberfläche durch allen diesen Daten bestimmt. Für sie werden Signale vom Berechnungsdienst gebraucht, die die Aktualisierung von Darstellungen hinweisen, egal ob dies von einer neuen Dateineingabe oder einer geänderten Einstellung verursacht wird.

3.3.2 Einheitensysteme in RGPKVA

In RGPKVA wird zum Austausch von physikalischen Größen zwischen alle Teilprogramme ein Einheitssystem aufgebaut.

Ein Merkmal von physikalischen Größen ist die wichtige Rolle von ihrer Einheiten. In Python können obwohl den Datentyp `float`(Fließkommazahl) verwendet werden, der ist jedoch für ein klares Programm unausreichend, indem eine gleiche Größe von Benutzereingaben bis zur Übermittlung an externe Computerbibliotheken oder Wiedergabe auf der Benutzeroberfläche in unterschiedlichen Einheiten gegeben werden muss, für die eine Konvertierung ständig zu beachten ist.

Aus der Sicht des OOP-Paradigm (siehe 3.1.3) bildet dieses Problem ein ideales Beispiel für die Anwendung von Klassen und Unterklassen: es wird zuerst die Klasse `PhysicalQuantity` in der Datei `rgpkva/calc/units.py` definiert, die eine physikalische Größe abstrahiert. Die Methoden zum Einlesen von einem physikalischen Wert inkl. seiner Einheit, und zur Wiedergabe der Werte in verschiedenen anderen Einheiten, werden zu dieser Basisklasse programmiert. Danach werden verschiedene Unterklassen mit tatsächlichen Information über Einheitskonvertierung von dieser Basisklasse geerbt, die dann überall in RGPKVA benutzt werden.

Diese Vorgehensweise kann durch folgendes Beispiel an der Definition von `Pressure`-Klasse vereinfacht erläutert werden:

```
class PhysicalQuantity:
    # (die Coden zu dieser abstrakten Klasse)

    def __init__(self, v):
       ... # bei der Initialisierung wird der Wert eingelesen
          .
    def units(self, unitName):
       ... # die "units" Methode legt der Klasse fest, in
         welcher Einheit
           # der Wert eingegeben worden ist.
    def to(self, unitName):
       ... # die "to" Methode konvertiert der Wert dieser
         Größe in eine andere Einheit

class Pressure(PhysicalQuantity):
    # eine Unterklasse, geerbt von "PhysicalQuantity"

    defaultUnit = "Pa"
    inputTable = {
        "Pa":  lambda i: i,
        "MPa": lambda i: i * 1.0e6,
        "kPa": lambda i: i * 1.0e3,
        "bar": lambda i: i * 1.0e5,
        "atm": lambda i: i * 101325.0,
    }
    outputTable = {
        "Pa": lambda v: v,
        "MPa": lambda v: v / 1.0e6,
```

```
        "kPa": lambda v: v / 1.0e3,
        "bar": lambda v: v / 1.0e5,
        "atm": lambda v: v / 101325.0
    }
```

Die Code in Basisklasse `PhysicalQuantity` kennt keine konkreten Regeln über die zu konventierenden Einheiten. Diese werden erst versorgt, wenn eine Unterklasse(hier im Beispiel die `Pressure`) diese vorschreibt.

In anderen Orten von RGPKVA kann beispielsweise

```
p_0 = Pressure(101325).units("Pa")
```

geschrieben werden, wodurch `p_0` nun das Objekt von `Pressure` hält. Wird eine dritte Computerbibliothek eine Druckeingabe in MPa verlangt, so führt die Expression

```
p_0.to("MPa")
```

zu einer `float` Ausgabe (hier `0.101325`).

Der Basisklasse `PhysicalQuantity` werden zusätzlich mit Sondermethoden ausgestattet, sodass ähnlich wie die native `float`-Klasse, lässt sich ein `PhysicalQuantity` unter Umstände auch mit anderen Objekt gleicher Klasse algebraisch berechnen, woraus einen neuen `PhysicalQuantity` oder `float` sich ergibt. Diese Ausstattung sind für z. B. die Ableitung von Druckverhältnis sehr hilfsreich.

3.3.3 CoolProp und Schnittstelle zu REFPROP

In RGPKVA werden überwiegend Realgaszustandsparameter ermittelt. Die Ermittlung wird wegen ihrer Komplexität nicht als Bestandteil in RGPKVA enthalten, sondern von einer Computerbibliothek dritter Seite übernommen. In der Praxis können zu diesem Zweck 2 Bibliotheken ausgewählt werden: CoolProp oder REFPROP.

Auswahl einer Bibliothek CoolProp[7] ist eine Computerbibliothek zur Berechnung von thermodynamischen Eigenschaften von sowohl Reinstoffen als auch Stoffgemischen. Sie ist unter der MIT-Lizenz im Internet veröffentlicht, die ihre Benutzung in einem Projekt kostenlos und unbeschränkt, bedingt mit einer Quellenangabe, freigibt. CoolProp läuft unter verschiedenen Betriebssystemen(Linux, MacOS und Windows), und bietet programmische Schnittstelle in zahlreichen Programmiersprachen(C, Python, MATLAB, GNU Octave, JAVA, FORTRAN usw.) und Anwendungen(Microsoft Excel, Libreoffice, EES) an.

REFPROP(NIST Reference Fluid Thermodynamic and Transport Properties Database)[8] wird von der NIST(National Institute of Standards and Technology) in Amerika angeboten, das als ein Windows-Programm die Fluideigenschaften standardisiert als Bezugsdaten bereitstellt. REFPROP hat dem CoolProp vergleichsweise umfangreichere Funktionalitäten, ist jedoch nicht gebührenfrei und unbeschränkt: der Preis für eine neue Bestellung liegt derzeit bei $250,00, für ein Upgrade von bestehender Installation auch noch $125,00[9]. Zudem unterliegt die Bestellung den Ausfuhrbestimmungen von USA.

Mit REFPROP wird eine DLL-Datei als programmische Schnittstelle mitgeliefert, deren Nutzung für Python jedoch teilweise kompliziert ist. Allerdings besitzt CoolProp eine Möglichkeit, nach Aufforderung die Anfragen anstatt programmintern zu lösen an REFPROP weiterzuleiten. Für RGPKVA wird aus diesem Grund CoolProp ausgewählt, sodass neben den o.g. Vorteilen die Berechnungsaufgaben immer und von den Computereinstellungen des Nutzers unabhängig durchgeführt werden können, und bei Bedarf sich auch durch REFPROP prüfen lassen.

Nutzung von CoolProp In Python bzw. im Programm RGPKVA wird ausschließlich die High-Level Funktion `PropsSI` von CoolProp genutzt. Die Nutzung von dieser lautet beispielsweise wie folgt:

```
from CoolProp.CoolProp import PropsSI
s = PropsSI("S", "P", 101325.0, "T", 273.15, "Nitrogen")
```

Das Beispiel zeigt die 6 Eingaben beim Aufruf der Funktion `PropsSI`:

1. die 1. Eingabe `S` ist der zu berechnenden Parameter, hier die massenspezifische Entropie.
2. die 2. und 3. Eingabe gibt die erste Randbedingung für die Anfrage an: `P` bedeutet den Druck und 101325.0 den Wert in Pascal.
3. die 4. und 5. Eingaben sind die zweite Randbedingung, hier `T` für Temperatur mit `273.15` in Kelvin.
4. die 6. Eingabe besagt, auf welchem Stoff diese Anfrage bezogen ist. Diese ist entweder aus einer Liste der Computerbibliothek [6] zu entnehmen, wodurch ein Reinstoff hingewiesen wird, oder aus den Reinstoffnamen für ein Gasgemisch als eine Zeichenfolge zu bilden, wie z. B. für Luft mit 78% Stickstoff, 21% Sauerstoff und 1% Argon:

   ```
   Nitrogen[0.78]&Oxygen[0.21]&Argon[0.01]
   ```

Sollte die Anfrage weiter an REFPROP zu leiten, kann ein Präfix `REFPROP::` an die Stoffbezeichnung vorgeschaltet werden:

```
REFPROP::Nitrogen[0.78]&Oxygen[0.21]&Argon[0.01]
```

Nun wird CoolProp versuchen, Kommunikation zu der DLL-Datei von REFPROP herzustellen.

3.3.4 Organisation der Berechnung

Alle Berechnungsaufgaben in RGPKVA werden modular organisiert, siehe Abb. 7 auf der nächsten Seite. Ein Modul ist für einen Teil der Aufgaben zuständig. Alle Module werden an die Klasse `Calculator` angeschlossen. Diese ist eine Unterklasse geerbt aus `Event`(siehe 3.1.5), und dient als ein zentrales Signalzentrum für alle. Ruft ein Modul `Calculator` mit Daten auf, dann werden diese an allen weiteren Modulen übertragen.

Jedes Modul identifiziert sich mit einem eigenen Namen. Beim Aufruf an `Calculator` wird dieser immer angegeben. Empfängt ein Modul Datenübertragung von den anderen, führt es diese zuerst durch einen Filter, sodass nur die Daten, die für die Berechnung dem Modul interessiert sind, verbleiben und weiter den Be- oder Verarbeitungsvorgang tatsächlich durchlaufen.

Modul `Stage` Das Modul `Stage` dient als die Bindungsstelle zu den Daten aus GKVA. Da RGPKVA periodisch von GKVA aufgerufen wird, steht es hierzu die Hauptaufgabe die Feststellung, ob diese Daten sich von der letzten Aktualisierung abweichen. Bleibt die GKVA-Daten unverändert, werden weitere Bearbeitung übersprungen. Sollte doch neue Daten erhalten, werden sie nach kurzer Sortierung in einer vereinfachten Datenstruktur weiter an allen Modulen übertragen.

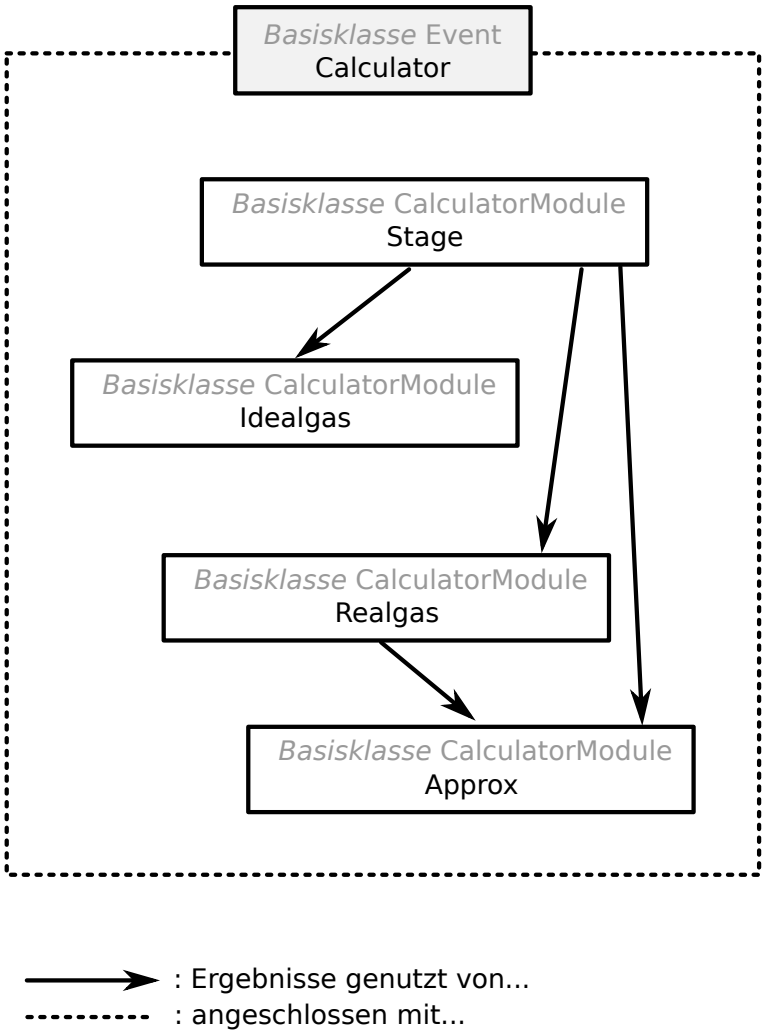


Abbildung 7: Organistation aller Berechnungsmodule

Modul `Idealgas` 2 Aufgaben werden dem Modul `Idealgas` zugeordnet. Die erste ist die Idealgasdaten, die schon durch GKVA berechnet worden sind, als eine für tabellarische Darstellung freundliche Format weiter umformen. Die zweite Aufgabe ist die Erzeugung einer Reihe von Zustandspunkten (p_i, T_i) nach der Vorgehensweise in 2.1.2, die für grafische Darstellung auf $T(p)$-Diagramm erforderlich sind.

Modul `Realgas` Dieses Modul führt die Realgasberechnungen nach 2.1.3 durch, und trägt die stufige Ergebnisse analog zu Modul `Idealgas` in eine übersichtliche Format ein, die durch die Benutzeroberfläche gegenüber den Werten des Idealgases veranschaulicht werden. Diskrete Zustandspunkten innerhalb einer Prozesslinie werden daher ebenfalls ermittelt.

Modul `Approx` Nachdem die Realgasberechnung zu Ergebnissen kommt, wird dieses Modul veranlassen. Die Realgasapproximationen werden anhand den stufigen Ein- und ausgangsdaten für jede isentrope Zustandsänderung durchgeführt, wozu das Algorithmus von 2.2 sich Anwendung findet. Die approximierte Parameter und deren relativen Fehler zu Realgaswerten werden für Diagramme vorbereitet.

4 Anwendung von RGPKVA

4.1 Bedienung

Abb. 8 auf der nächsten Seite zeigt die Benutzeroberfläche von RGPKVA in Laufzeit.

Die Benutzeroberfläche wird in 4 Quadranten eingeteilt: die Eingabe von relevanten Parametern erfolgt in der Tabelle unten links; unten rechts befindet sich die grafische Darstellung. Oben rechts können aus einer Liste bis zu maximal 2 verfügbare Darstellungen ausgewählt werden[2], die gegebenfalls im Diagramm die beide Y-Achsen besetzen. Die wichtige Zahlen für die Verdichtung in Anlagen unter Ideal- und Realgasbetrachtungen, werden in der Tabelle oben links gezeigt.

Die Eingaben in die unteren Tabelle können per Mausklick auf entsprechenden Zellen von Säule „Werte“ geändert werden. Bei Änderungen an „Gasgemischzusammensetzung“ und „Pfad zu REFPROP“ werden zweckmäßige Fenster aufgerufen, siehe Abb. 9 auf Seite 32.

Die Änderung wird erst wirksam, wenn der Fenster zur Eingabehilfe geschlossen ist, und die zu ändernde Zelle ihren Fokus (gezeigt als einen schwarzen Kasten um die Zelle) verliert, z. B. durch Mausklick auf eine andere Zelle. Danach werden die Berechnungen automatisch erneut.

4.2 Bedienung am Beispiel mit trockner Luft höherer Drücke

Die Funktionsfähigkeit des Programms RGPKVA bei der Realgasberechnung lässt sich an folgendem Beispiel testen.

Das Idealgas wird mit $R = 285.0\,\mathrm{J/(kg \cdot K)}$ und $K = 3.50$ ($\kappa = 1.4$) definiert. Das Realgasgemisch wird als trocknene Luft (`Nitrogen[0.78]&Oxygen[0.22]`) eingegeben.

Als Beispiel wird Luft mit Umgebungstemperatur $T_U = 273.15\,\mathrm{K}$ von $100\,\mathrm{bar}$ bis $1000\,\mathrm{bar}$ über 3 Stufen gleicher Stufendruckverhältnisse isentropisch verdichtet und zwischen jeden 2 benachbarten Stufen vollständig wieder zu T_U gekühlt. Die Parameter in dieser Anlage werden als in der Tabelle 1 eingetragen berechnet.

Stufe			1	2	3
Eingang	p [Pa]		10 000 000.0	21 544 346.9	46 415 888.34
	T [K]	ideal	273.15	273.15	273.15
		real	273.15	273.15	273.15
	R [$\frac{\mathrm{J}}{\mathrm{kg \cdot K}}$]	ideal	285.0	285.0	285.0
		real	239.60	235.70	285.67
	K	ideal	3.50	3.50	3.50
		real	3.35	2.99	2.19
Ausgang	p [Pa]		21 544 346.9	46 415 888.34	100 000 000.0
	T [K]	ideal	340.13	340.13	340.13
		real	342.30	336.37	327.38
	R [$\frac{\mathrm{J}}{\mathrm{kg \cdot K}}$]	ideal	285.0	285.0	285.0
		real	259.46	294.13	416.46
	K	ideal	3.50	3.50	3.50
		real	3.13	2.57	1.84

Tabelle 1: Parameter einer Luftverdichteranlage am Beispiel vom Kapitel 4.2

[2] Für eine komplette Liste aller erzeugten Diagrammen siehe Anhang A auf Seite 38.

Stufe			1	2	3
Eingang	p [Pa]		100000.00	215443.47	464158.88
	T [K]	ideal	300.00	300.00	300.00
		real	300.00	300.00	300.00
	R [J/kg/K]	ideal	285.00	285.00	285.00
		real	287.71	287.62	287.43
	K	ideal	3.50	3.50	3.50
		real	3.50	3.49	3.49
Ausgang	p [Pa]		215443.47	464158.88	1000000.00
	T [K]	ideal	373.56	373.56	373.56
		real	373.20	373.28	373.45
	R [J/kg/K]	ideal	285.00	285.00	285.00
		real	287.88	287.97	288.20
	K	ideal	3.50	3.50	3.50
		real	3.50	3.49	3.48

	Name	Wert
1	Gasgemischzusammensetzung	Oxygen[0.220000]&Nitrogen[0
2	Abweichung d in Realgasapprox.	
3	Verwendung von REFPROP	Nein
4	ggf. Pfad zu REFPROP	
5	Einheit der Temperatur	K
6	Einheit des Drucks	Pa

T(p)-Diagramm des Ideal- und Realgases
T(p)-Diagramm in Realgasapproximation
Ablauf der Gaskonstante R von Ideal- und Realgas
Ablauf der Werte K von Ideal- und Realgas
Ablauf der Gaskonstante R in Realgasapproximation
Ablauf der Werte K in Realgasapproximation
Approximationsfehler von R und K

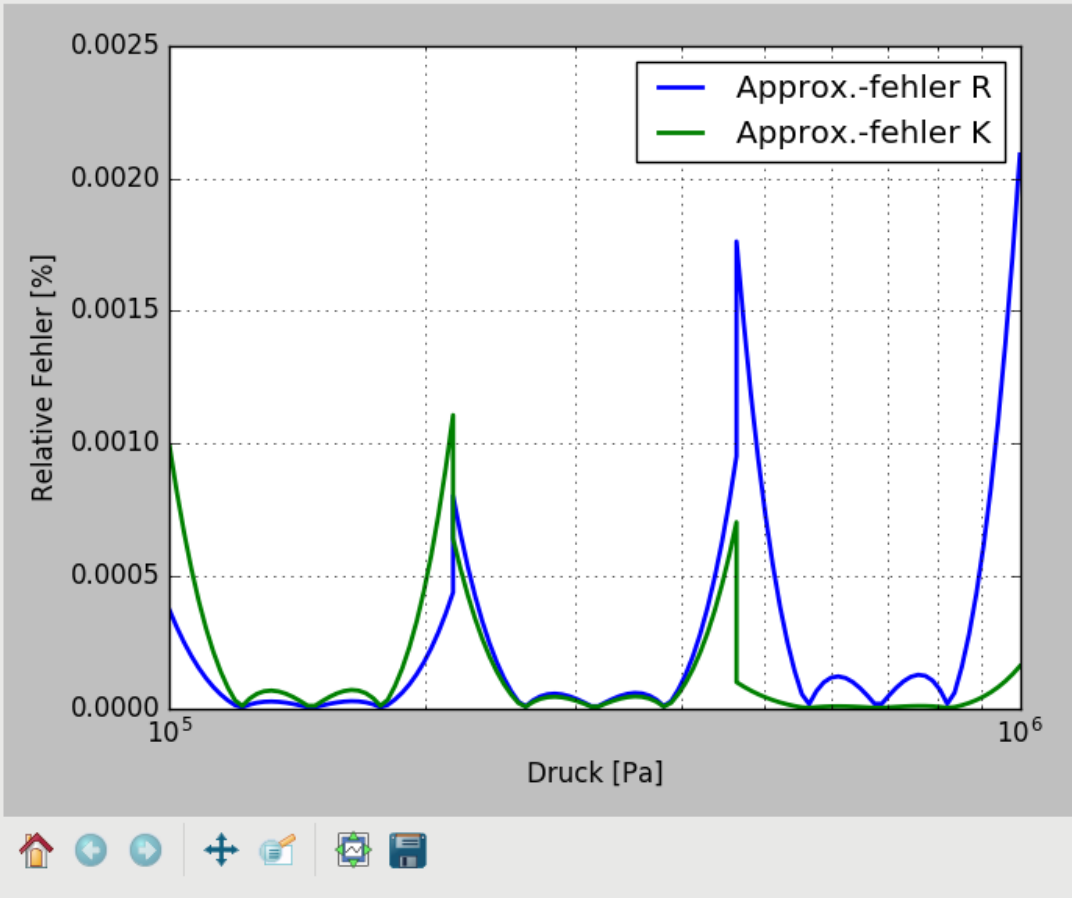
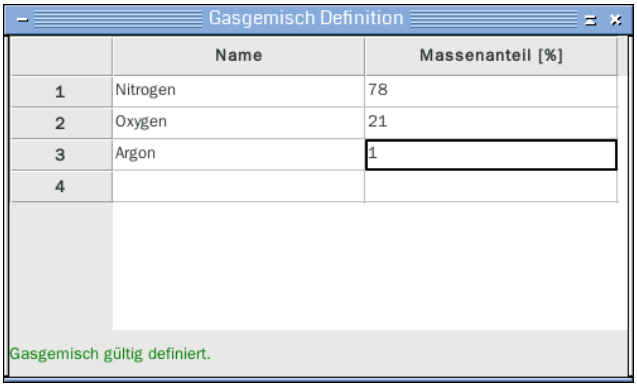
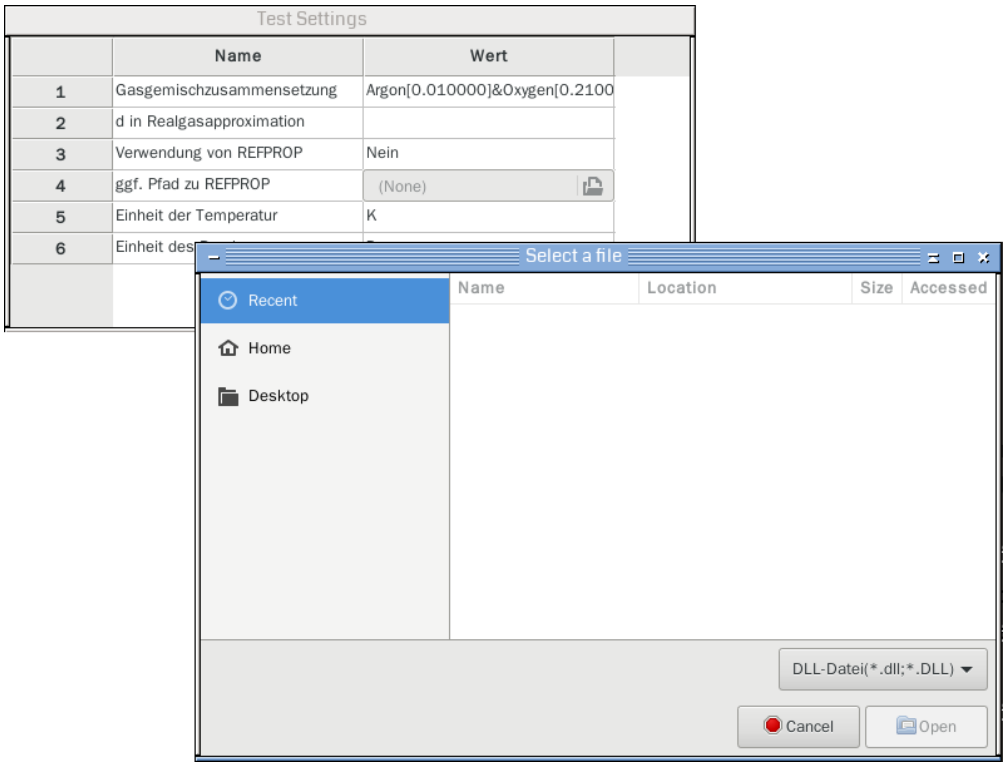


Abbildung 8: Die Benutzeroberfläche von RGPKVA



(a) zur Definition des Gasgemisches



(b) Auswahl der DLL-Bibliothek von REFPROP

Abbildung 9: Fenster als Eingabehilfen

Der Ablauf von Zustandsparameter T nach p von Ideal- und Realgas wird in Abb. 10a auf der nächsten Seite gegenüber einander dargestellt. Die Abweichung von Temperaturen aus beiden Fallen ist besonders bei höhem Druck in der letzten Stufe erkennbar.

Zusätzlich kann aus der Abbildung beobachtet werden, dass die Realgastemperatur in Stufe 1 generell oberhalb der Idealgastemperatur liegt, wonach diese Beziehung in Stufen 2 und 3 sich umkehrt. Dieses Phänomen stimmt mit dem Realgasfaktor z (Abb. 10b auf der nächsten Seite) von Luft überein, wobei z zwischen $100\,\mathrm{bar}$ und etwa $250\,\mathrm{bar}$ kleiner als 1.0 liegt und danach diese übersteigt. Aus Zustandsgleichung realer Gase $p = z \cdot \rho R_S T$ führt $z < 1$ unter gleicher Bedingung vom Druck p erhöhte Temperatur T, und bei $z > 1$ im Gegensatz, die diese Berechnung bestätigt.

4.3 Bedienung am Beispiel mit Erdgas

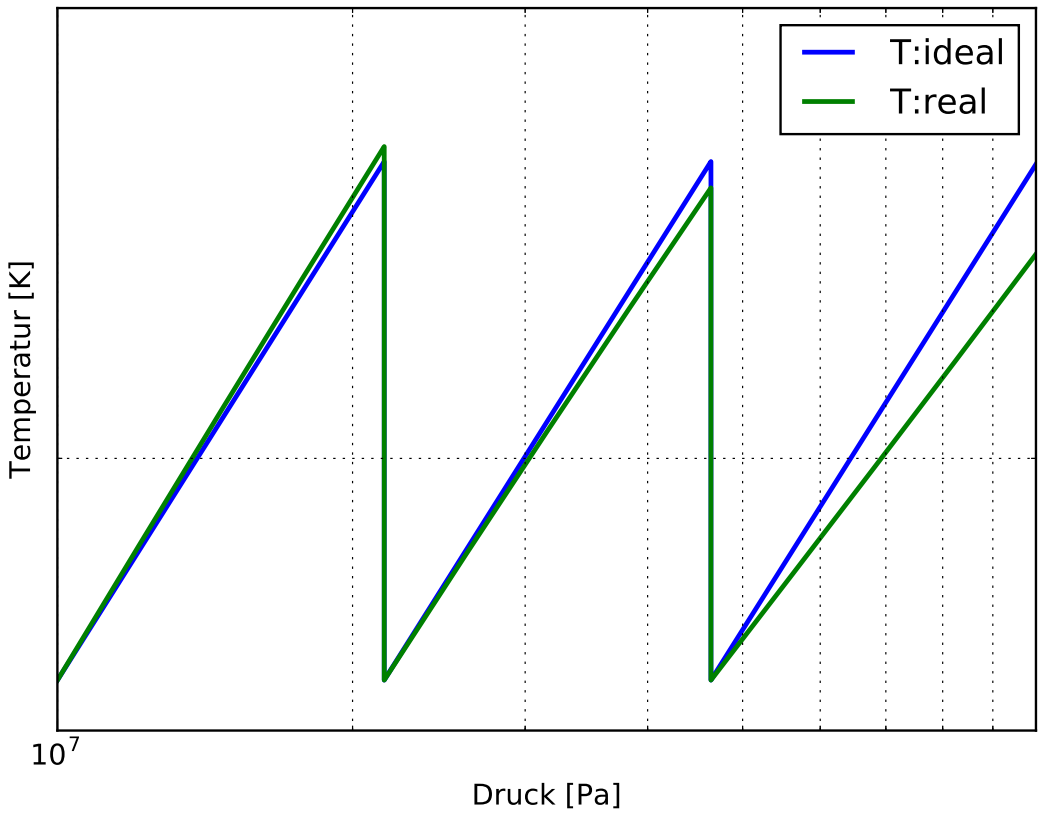
Ein weiteres Beispiel von RGPKVA wird anhand einer Verdichteranlage mit Erdgas demonstriert.

Die Erdgaszusammensetzung wird hiermit zu 90% Methan, 5% Ethan und 5% Stickstoff (in Massenanteilen) an RGPKVA eingegeben: `Methane[0.9]&Ethane[0.05]&Nitrogen[0.05]`, zu dessen Vergleich das Idealgas mit $R = 477.75\,\frac{\mathrm{J}}{\mathrm{kg \cdot K}}$ und $K = 5.88 (\kappa = 1.20)$ definiert wird. Die Verdichtung erfolgt von $p_{\mathrm{ein}} = 1\,\mathrm{bar}$ zu $p_{\mathrm{aus}} = 10\,\mathrm{bar}$ über 3 Stufen gleicher Stufendruckverhältnissen. Die Umgebungstemperatur liegt bei $T_U = 300\,\mathrm{K}$. Die Tabelle 2 enthält Parameter für diese Anlage.

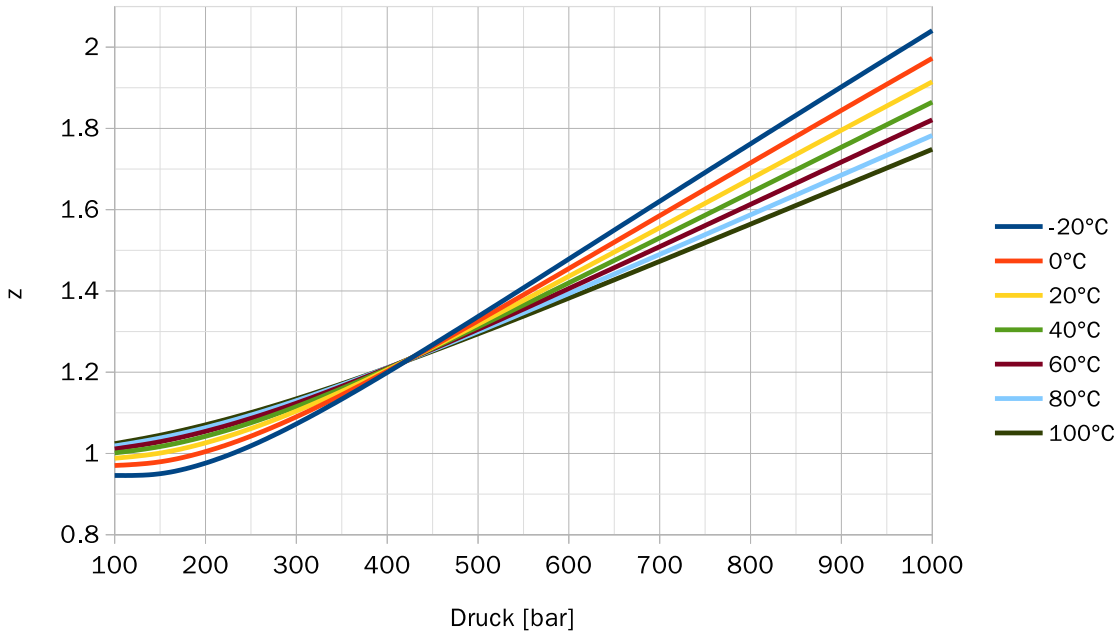
Stufe			1	2	3
Eingang	p [Pa]		100 000.0	215 443.47	464 158.88
	T [K]	ideal	300.0	300.0	300.0
		real	300.0	300.0	300.0
	R $[\frac{\mathrm{J}}{\mathrm{kg \cdot K}}]$	ideal	477.75	477.75	477.75
		real	478.57	477.58	475.46
	K	ideal	5.88	5.88	5.88
		real	5.88	5.89	5.90
Ausgang	p [Pa]		215 443.47	464 158.88	1 000 000.0
	T [K]	ideal	373.56	373.56	373.56
		real	355.83	356.00	356.38
	R $[\frac{\mathrm{J}}{\mathrm{kg \cdot K}}]$	ideal	477.75	477.75	477.75
		real	478.50	477.44	475.21
	K	ideal	5.88	5.88	5.88
		real	5.67	5.67	5.67

Tabelle 2: Parameter einer Erdgasverdichteranlage am Beispiel vom Kapitel 4.3

Aus der Tabelle sind die Temperaturunterschiede zwischen Ideal- und Realgasbetrachtung deutlich. Zudem sind der schwankende Ablauf vom Verhältnis K und die abnehmende Gaskonstante R für Realgas leicht zu erkennen, die auch in Abb. 11 auf Seite 35 grafisch dargestellt werden.

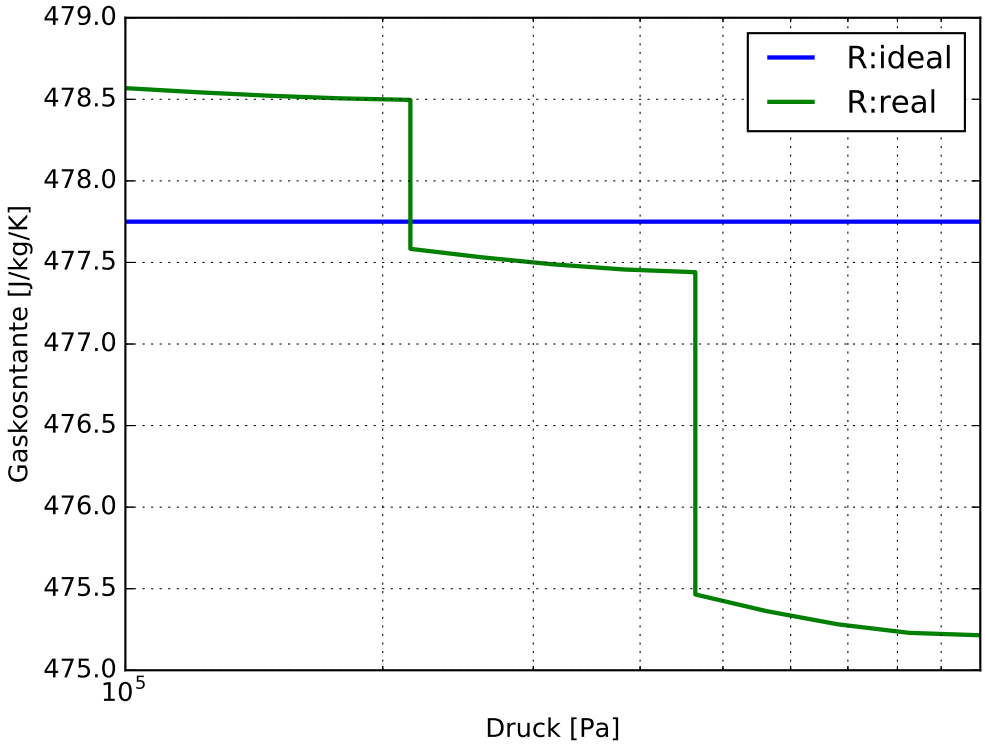


(a) $T(p)$-Diagramm von Ideal- und Realgas

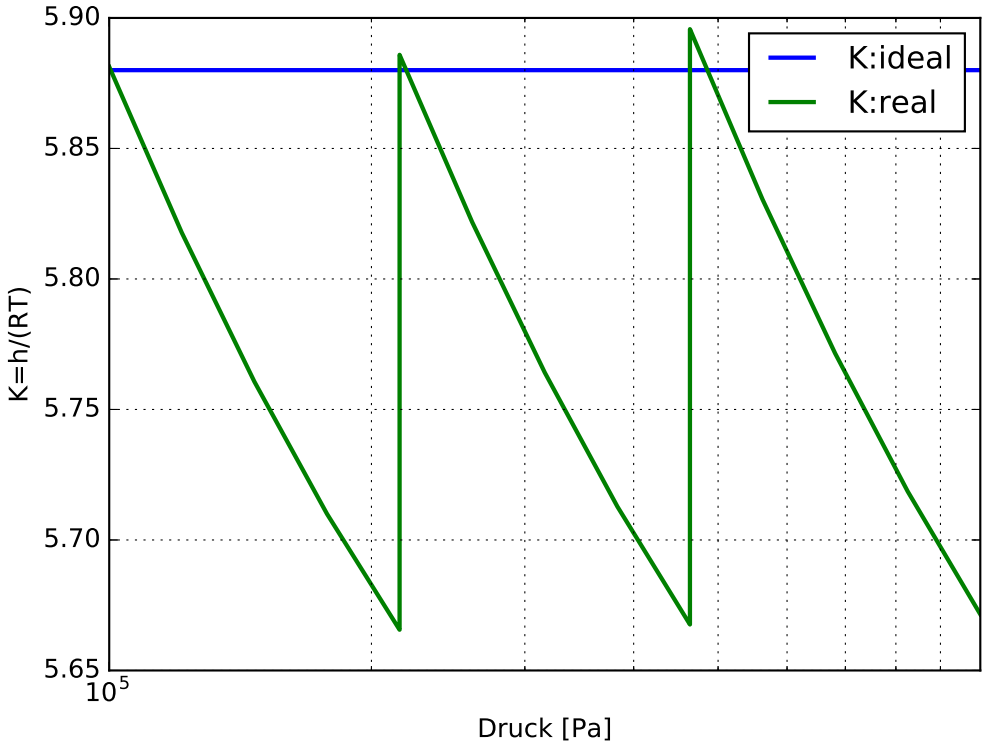


(b) Realgasfaktor z nach Angaben von CoolProp

Abbildung 10: $T(p)$-Diagramm der Luftverdichteranlage vom Kapitel 4.2 und Realgasfaktor



(a) Gaskonstante R



(b) Verhältnis K

Abbildung 11: Ablauf von R und K zum Beispiel einer Erdgasverdichteranlage vom Kapitel 4.3

4.4 Approximationsfehler bei verschiedenen Eingangstemperaturen

Anschließend zu dem Beispiel in Kapitel 4.3 wird untersucht, wie sich die Approximationsfehler bei unterschiedlichen Eingangstemperatur verhält. Die Berechnungen werden mit Umgebungs- bzw. Eingangstemperaturen T_E zwischen $273.15\,\mathrm{K}$ und $373.15\,\mathrm{K}$ durchgeführt. Die Maxima von relativen Approximationsfehlern (Diagramme im Anhang B auf Seite 43) bei Parametern R und K aus jeweiliger Stufe, werden als e_R und e_K danach tabellarisch zusammengefügt, siehe Tabelle 3.

T_E [K]	T_A [K]	Stufe 1		Stufe 2		Stufe 3	
		e_R [%]	e_K [%]	e_R [%]	e_K [%]	e_R [%]	e_K [%]
273.15	340.13	0.0004	0.0041	0.0010	0.0038	0.0024	0.0030
323.15	402.39	0.0004	0.0032	0.0010	0.0028	0.0023	0.0019
373.15	441.45	0.0004	0.0017	0.0009	0.0013	0.0022	0.0007

Tabelle 3: Relative Approximationsfehler von R und K

Die Genauigkeit bei der Approximation von R ist generell besser als K. Der Approximationsfehler e_R weist wenigere Abhängigkeiten von den Temperaturen T_E und T_A auf, wohingegen sich e_K mit erhöhten Temperaturen sichtlich abnimmt. Da die Approximation auf isentroper Zustandsänderung von Idealgas sich basiert, soll diese Tendenz erwartet werden, indem bei höheren Temperaturen das Realgasverhalten sich an Idealgasverhalten annähert.

Literaturverzeichnis

[1] Hesse, U. und Will, G.: *Thermodynamic Calculation of Reciprocating Compressor Plants*. Vienna: 9th Conference of the EFRC, 2014.

[2] Will, G.: *Realgas-Approximation - Auszug aus Programmdokumentation zu KVA*. unveröffentl. Bericht, TU Dresden, Kälte-,Kryo- und Kompressorentechnik, 2016/7.

[3] Lutz, Mark: *Learning Python*. 5. Sebastopol: O'Reilly Media, 2013. - ISBN 978-1-4493-5573-9

[4] van Rossum, Guido; Drake, Fred L.: *The Python Language Reference*: Network Theory Ltd., 2011. - ISBN 978-1-9069-6614-0

[5] *Data model*. Online verfügbar: `https://docs.python.org/2/reference/datamodel.html`. Zugriff am 14.07.2017.

[6] *List of Fluids*. Online verfügbar: `http://www.coolprop.org/fluid_properties/PurePseudoPure.html#list-of-fluids`. Zugriff am 14.07.2017.

[7] Bell, I. H.; Wronski, J.; Quoilin, S. und Lemort, V.: *Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp*. Industrial & Engineering Chemistry Research 53 (2014), Nr. 6, S. 2498-2508.

[8] Lemmon, E. W.; Huber, M. L. und McLinden, M. O.: *NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties - REFPROP. 9.0*. Gaithersburg: National Institute of Standards and Technology, Standard Reference Data Program, 2010.

[9] *REFPROP*. Online verfügbar: `https://www.nist.gov/srd/refprop`. Zugriff am 14.07.2017.

Dresden, Juli 2017

A Alle Diagramme in der Benutzeroberfläche von RGPKVA

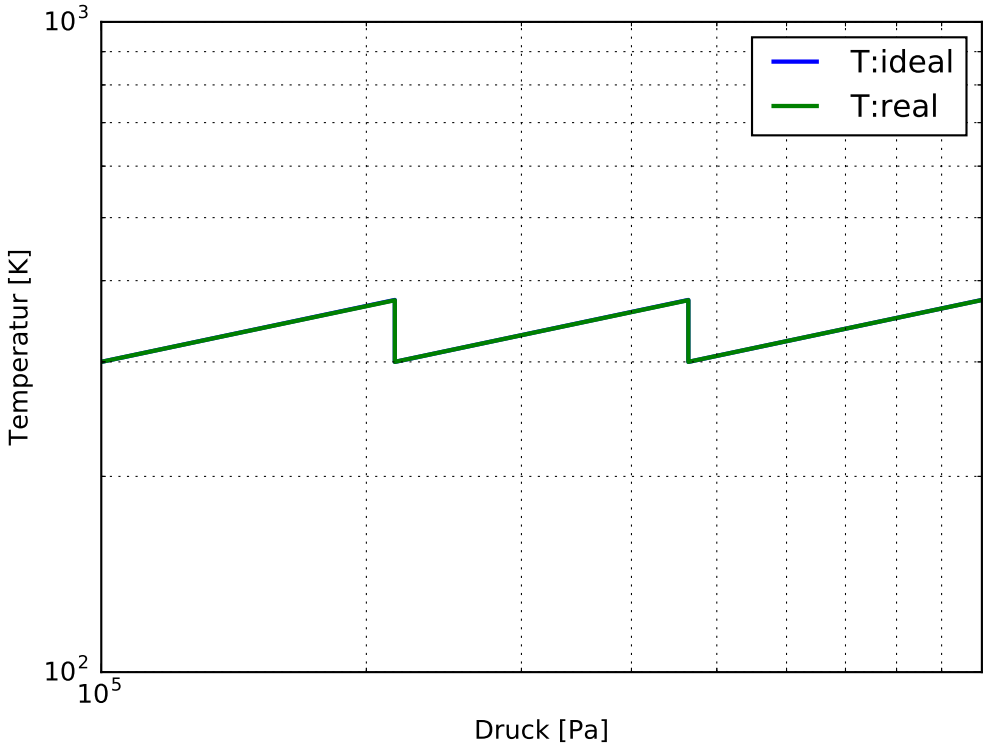


Abbildung 12: $T(p)$-Diagramm des Ideal- und Realgases

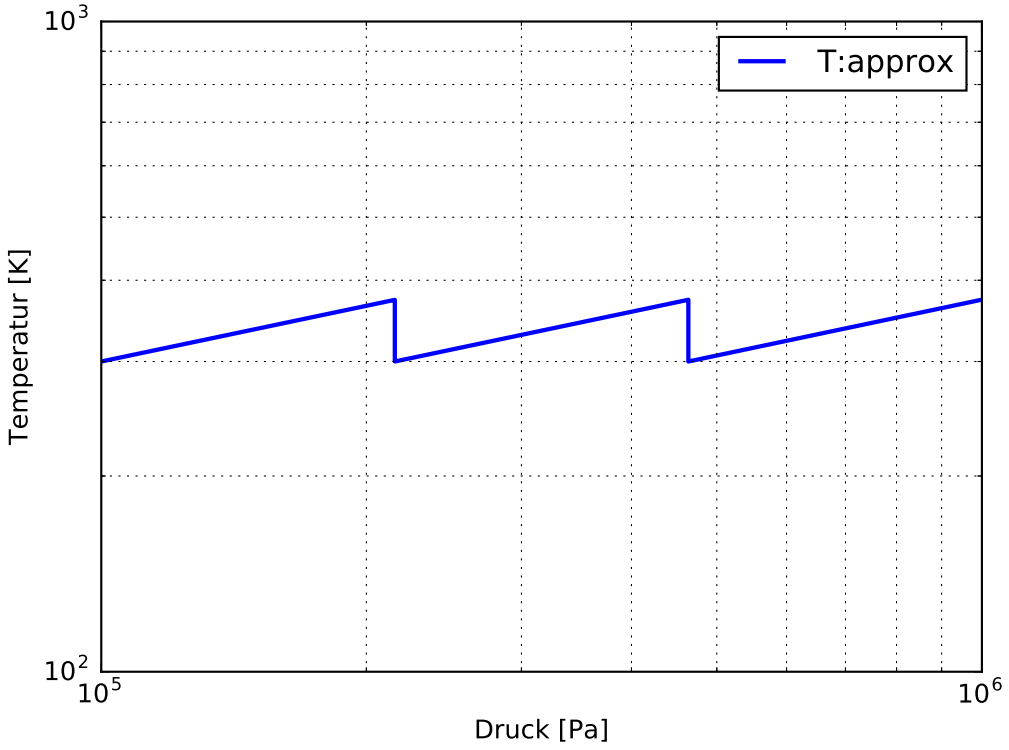


Abbildung 13: $T(p)$-Diagramm in Realgasapproximation

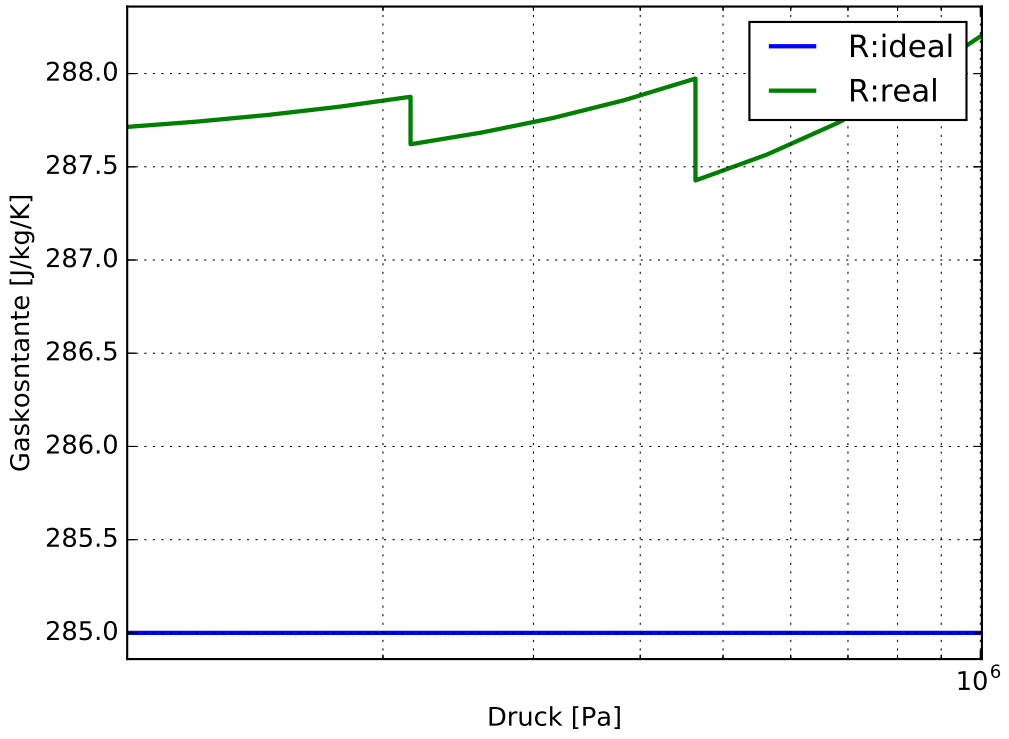


Abbildung 14: Ablauf der Gaskonstante R von Ideal- und Realgas

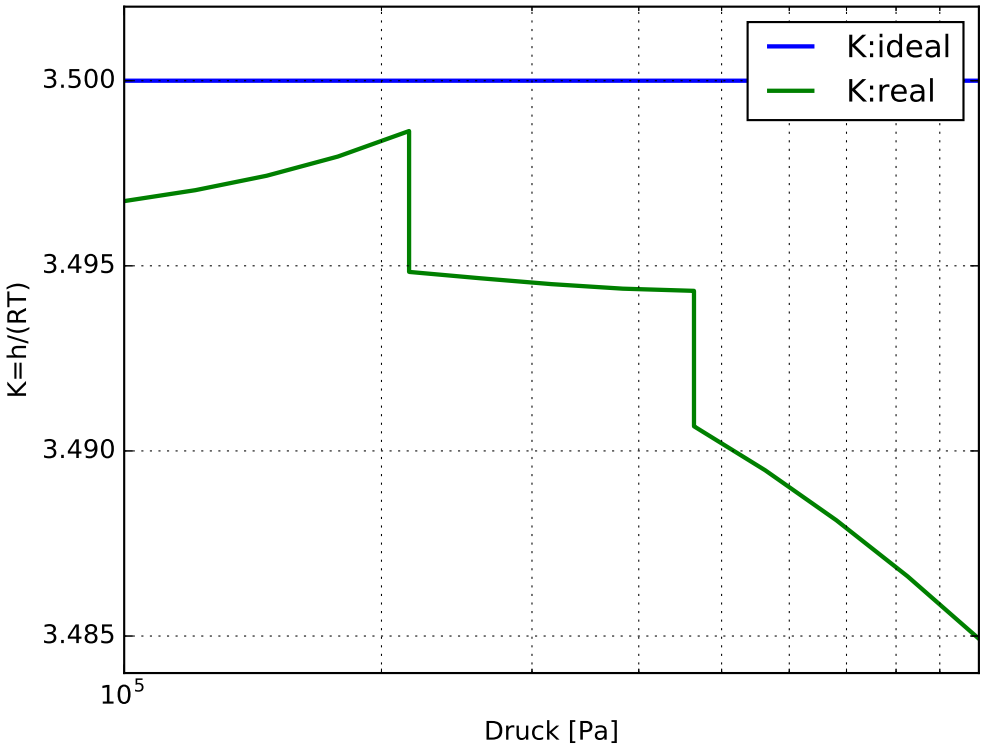


Abbildung 15: Ablauf der Werte K von Ideal- und Realgas

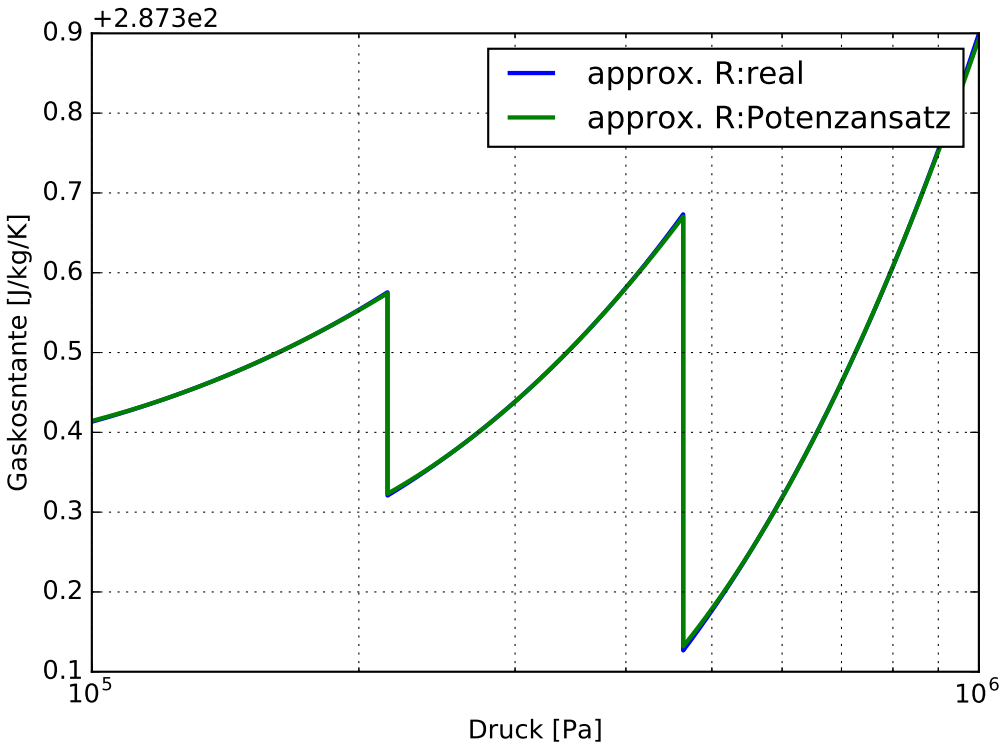


Abbildung 16: Ablauf der Gaskonstante R in Realgasapproximation

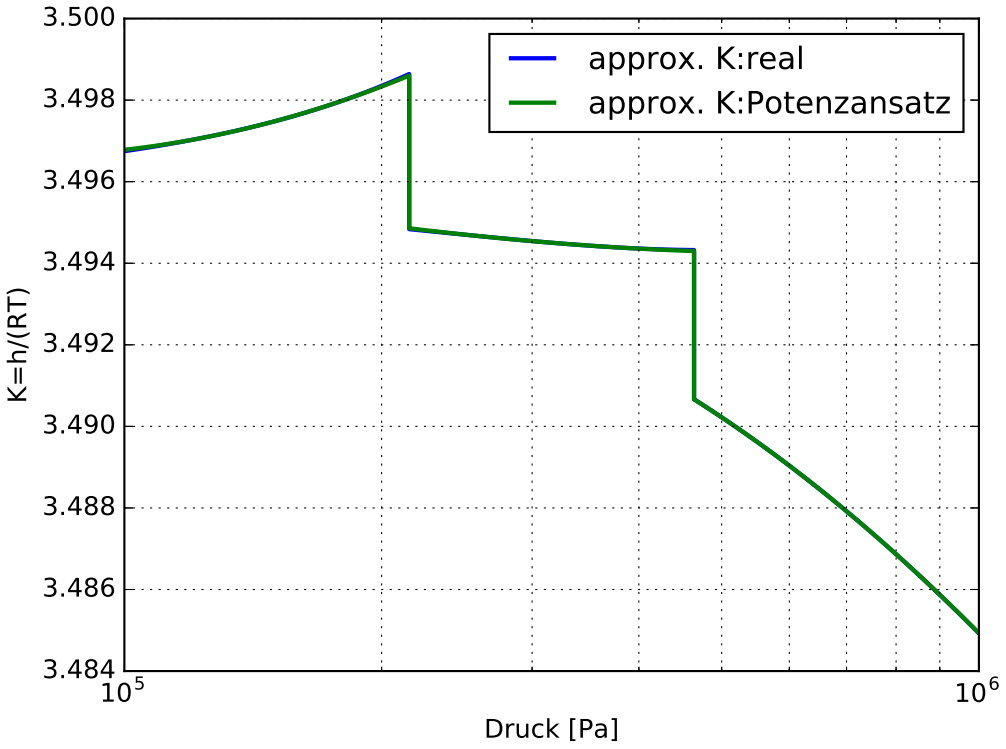


Abbildung 17: Ablauf der Werte K in Realgasapproximation

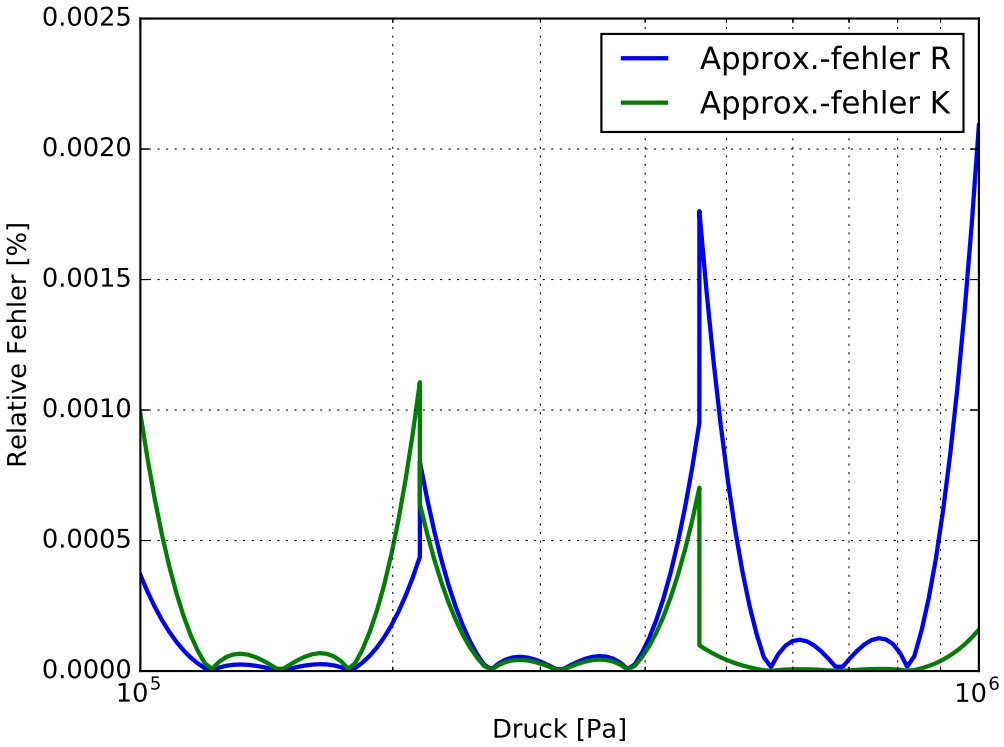


Abbildung 18: Approximationsfehler von R und K

B Approximationsfehler bei verschiedenen Eingangstemperaturen

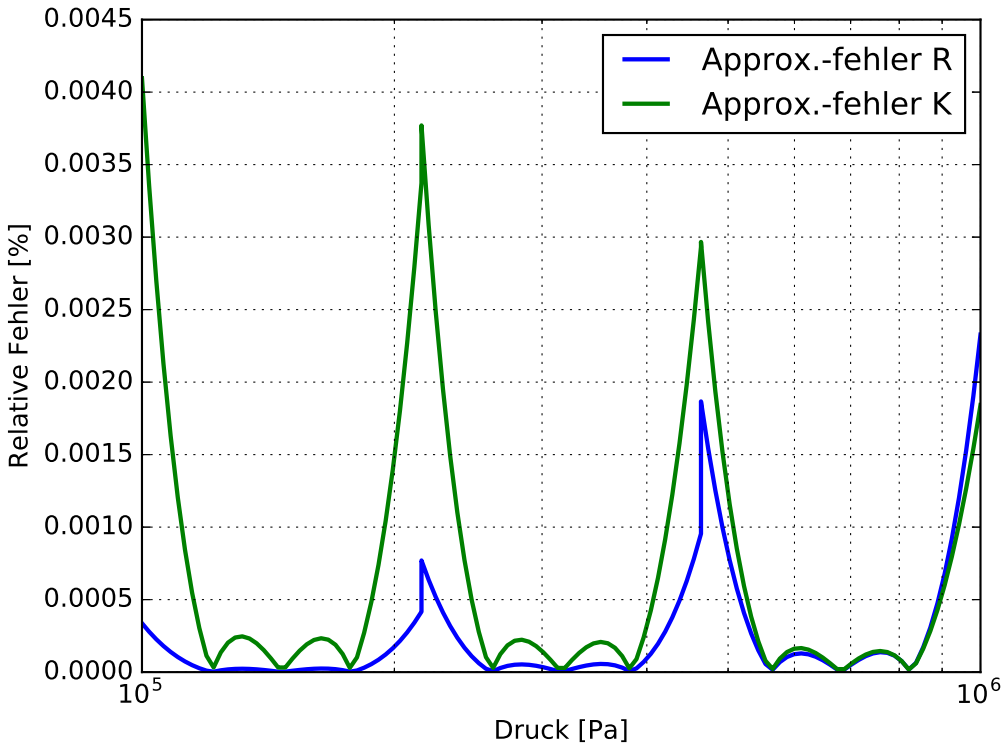


Abbildung 19: Erdgas, Verdichtung zum $T_E = 273.15\,\mathrm{K}$ angefangen

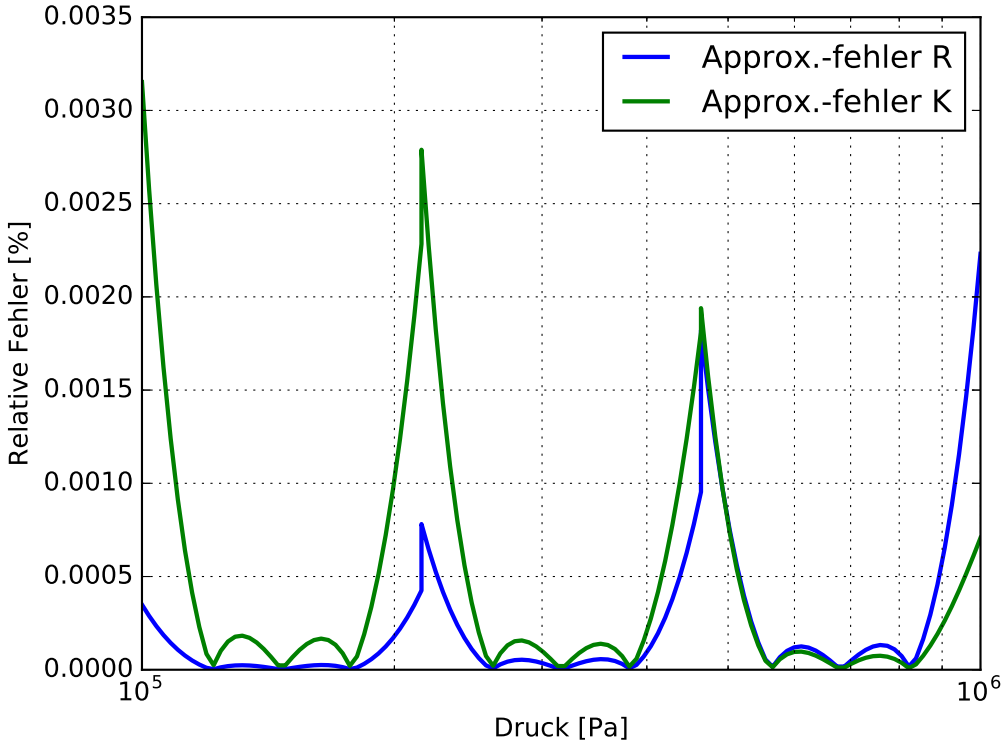


Abbildung 20: Erdgas, Verdichtung zum $T_E = 323.15\,\mathrm{K}$ angefangen

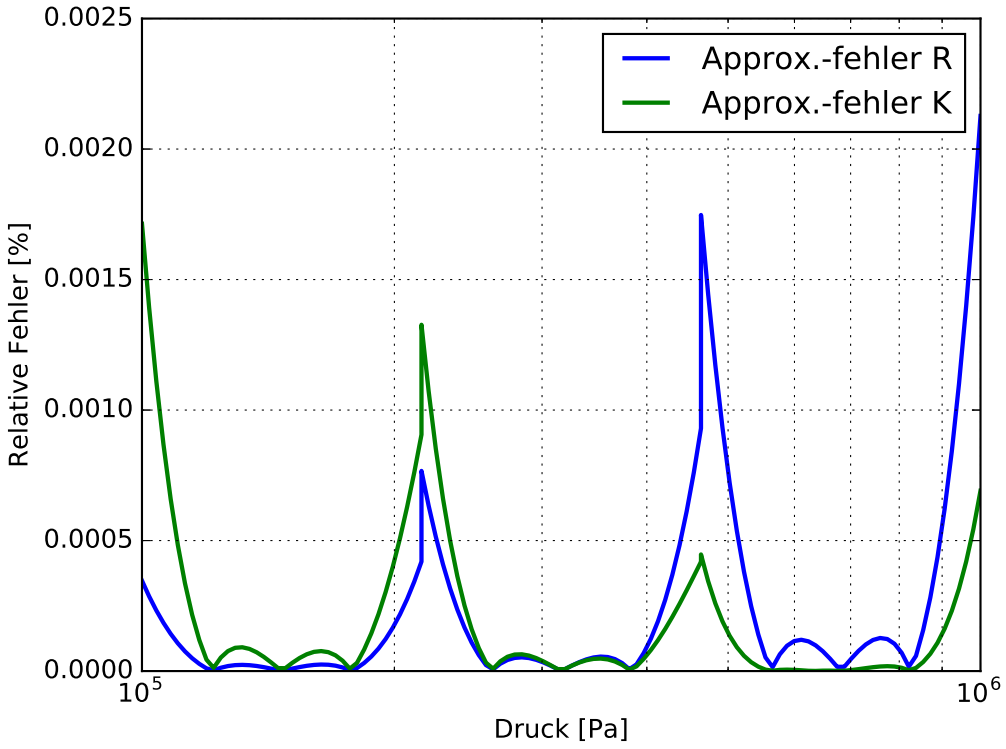


Abbildung 21: Erdgas, Verdichtung zum $T_E = 373.15\,\mathrm{K}$ angefangen

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Werken wörtlich oder sinngemäß übernommenen Gedanken sind unter Angabe der Quellen gekennzeichnet.

Ich versichere, dass ich bisher keine Prüfungsarbeit mit gleichem oder ähnlichen Thema bei einer Prüfungsbehörde oder anderen Hochschule vorgelegt habe.

.....

Ort, Datum Unterschrift