Technische Universität Graz
**Institut für Baumechanik**

Bachelorprojekt im Studiengang Bauingenieurwissenschaften
am Institut für Baumechanik

Technische Universität Graz

# Programm zur Berechnung von Querschnittskennwerten

## Python-Programm mit symbolischer oder numerischer Ergebnisausgabe

Philipp Egg, Michael Kaiser

Graz, 1. Februar 2017

Betreuer: Dipl.-Ing. BSc. Michael Gfrerer

## Danksagung

Im Rahmen dieser Arbeit haben wir uns sehr intensiv mit der EDV-unterstützten Ermittlung von Querschnittskennwerten beschäftigt. Hierzu haben wir unser Wissen insbesondere in den Bereichen Mechanik und Informatik vertieft.
Besonderer Dank für die ausgezeichnete und zeitintensive Betreuung gilt Herrn Dipl.-Ing. BSc. Michael Gfrerer. Durch seine große fachliche Kompetenz konnte er uns in allen möglichen Belangen stets zielführende Ratschläge geben.


## Abstract


The aim of this project was to write a program that computes cross-section values, such as the first moment of area, the moment of inertia and the centroid. Further, the program is capable of rotating the two axes through a certain angle to determine the maximum of the second moment of area. This particular angle is also included in the output of the calculation.
The cross section can be entered either numerically or with the help of predefined symbols which offer a simplified way of defining the inputs to subsequently run the calculation.
Aside from the technical aspects, the project also aimed to produce a .pdf-file that includes a comprehensive list of results and a sketch to visualize the computed cross-section. The programming language is based on python3 in combination with LaTeXand the project draws on multiple libraries to realize the aims outlined above.

# INHALTSVERZEICHNIS

# 1 EINLEITUNG: ZIEL, WARUM?

Vorliegender Bericht stellt unser Bachelorprojekt vor. Der Inhalt unseres Projektes ist ein Programm zur Berechnung von Querschnittskennwerten, die in der Mechanik Verwendung finden. Es gibt zahlreiche EDV-Programme (zum Beispiel Statikprogramme), die zur Berechnung von Querschnittskennwerten geeignet sind, allerdings geben die meisten nur Zahlenwerte aus, während in der Mechanik symbolisch gerechnet wird. Ziel des Projektes war es, ein Programm zu schreiben, dass diese Querschnittskennwerte symbolisch berechnet. Als zusätzliche Funktion kann im Eingabefeld eine numerische Berechnung gewählt werden, womit auch Zahlenwerte berechnet werden können.
Es werden folgende Querschnittskennwerte berechnet:

- Fläche – A
- Statisches Moment – S
  - $S_y$
  - $S_z$
- Schwerpunkt – s
  - $y_s$
  - $z_s$
- Flächenträgheitsmoment – I
  - $I_y$
  - $I_z$
  - $I_{yz}$
- Hauptträgheitsmomente und der Winkel zur Hauptachse

Das Programm wurde in der Programmiersprache Python geschrieben. Für die symbolische Berechnung haben wir sympy als zusätzliches Package verwendet.

- tkinter – Systemeingabe
- pillow – Grafiken anzeigen
- sympy – symbolisches Rechnen

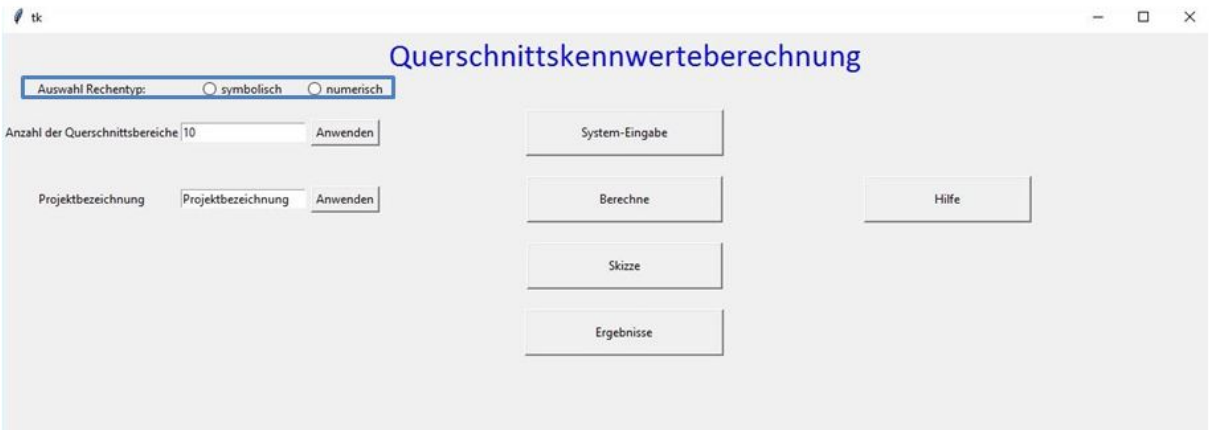
Je nachdem ob diese Packages auf dem Computer bereits vorhanden sind oder nicht, müssen diese vor dem ersten Ausführen des Programms installiert werden.
Sollte Python noch nicht am Computer installiert sein und muss daher neu installiert werden, kann zum Beispiel PythonAnaconda verwendet werden. Hier sind die notwendigen Packages bereits enthalten und müssen somit nicht mehr extra installiert werden.
Um die pdf-Dateien der Ausgabe erstellen zu können, ist zudem LaTeX notwendig.

In den folgenden beiden Kapiteln wird genauer auf die Bedienung, die Funktionsweise und den Aufbau des Programms eingegangen.

# 2 HOW-TO - PROGRAMMANLEITUNG

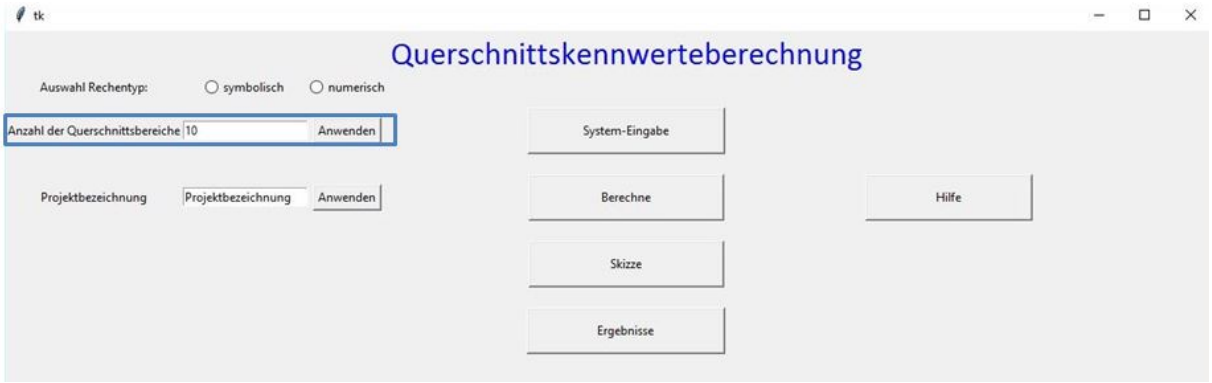
## 2.1 Mainwindow

Dieses Fenster öffnet sich beim Aufrufen, des Programms. Auf der linken Seite können allgemeine Angaben gemacht werden:



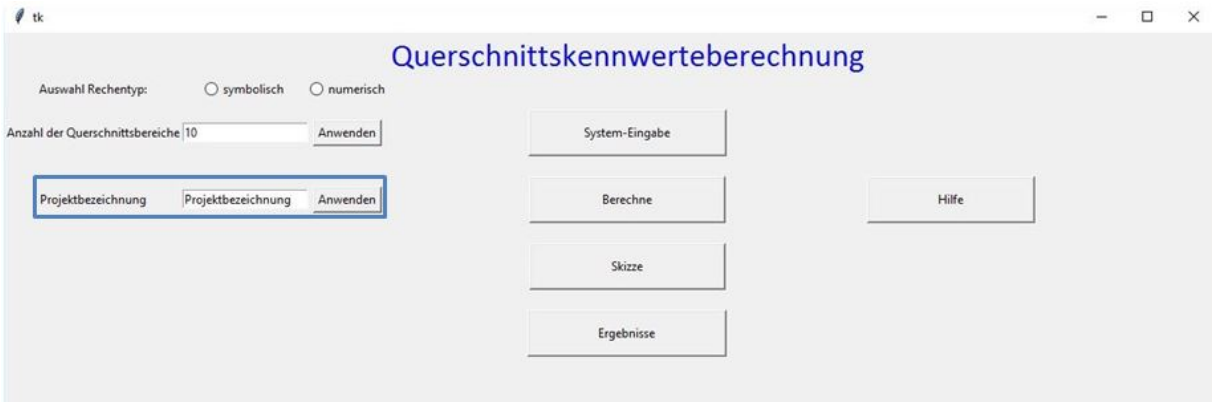

Auswahl des Rechentyps

Hier kann der Rechentyp ausgewählt werden, das heißt, dass je nach Auswahl die Berechnung in symbolischer Form durchgeführt wird, beziehungsweise bei der Auswahl von „numerisch“ die Ergebnisse in Form von Zahlenwerten ausgegeben werden. Bei der Auswahl einer numerischen Berechnung ist in Folge keine Vereinfachung wählbar, mehr dazu später unter dem Punkt „Systemeingabe“.




Auswahl der Querschnittsbereiche

Hier kann optional ausgewählt werden, wie viele Querschnittsbereiche es geben sollte. Ein Querschnittsbereich geht immer von einem Teilpunkt zu einem anderen Teilpunkt. Der Defaultwert von zehn Bereichen kann umgestellt werden, wenn ein Querschnitt mit mehr als zehn Einzelbereichen berechnet werden sollte. Hat der zu berechnende Querschnitt weniger Bereiche, kann ein kleinerer Wert eingegeben werden, dies ist jedoch nicht unbedingt notwendig. Es muss aber jedenfalls der Button „Anwenden“ gedrückt werden.




Eingabe Projektbezeichnung

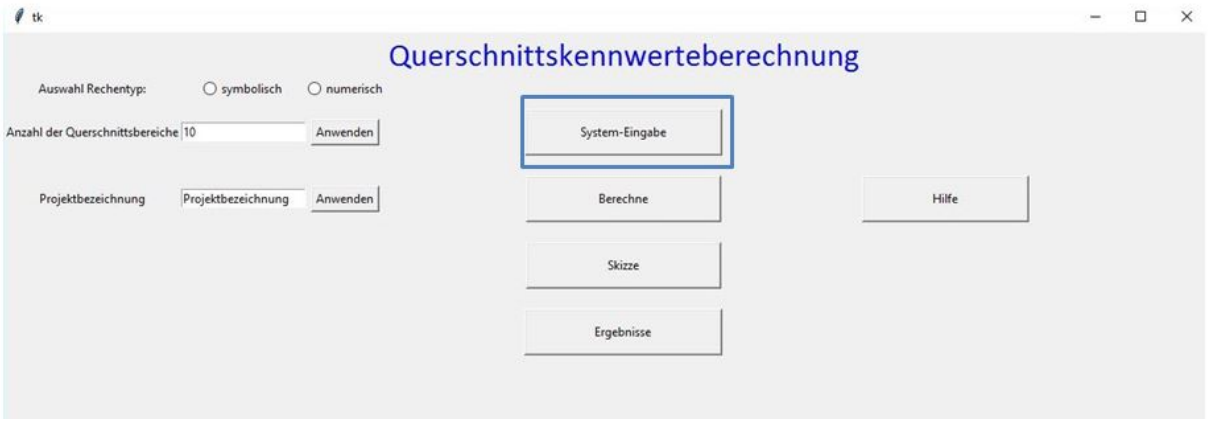
Hier kann eine Projektbezeichnung eingegeben werden, die dann am Ausgabedokument aufgedruckt wird. Sollte vom Nutzer keine spezifische Angabe gemacht werden, wird automatisch folgender Text ausgegeben: „Es wurde keine spezifische Projektbezeichnung angegeben!“. Da für die Ausgabe von Python automatisch ein LaTeX – Dokument generiert wird, können keine Umlaute, Satz- oder Sonderzeichen eingegeben werden. Ä, ö, ü (Groß– und Kleinschreibung), ß, !, “, $, %, /, _ etc. sind also nicht möglich beziehungsweise nicht zu empfehlen, da es zu einem Fehler beim Kompilieren der pdf–Datei kommen könnte.

Es muss aber wieder jedenfalls der Button „Anwenden“ gedrückt werden.

Rechts ist der Butten „Hilfe“. Wird er gedrückt öffnet sich eine Programmanleitung.

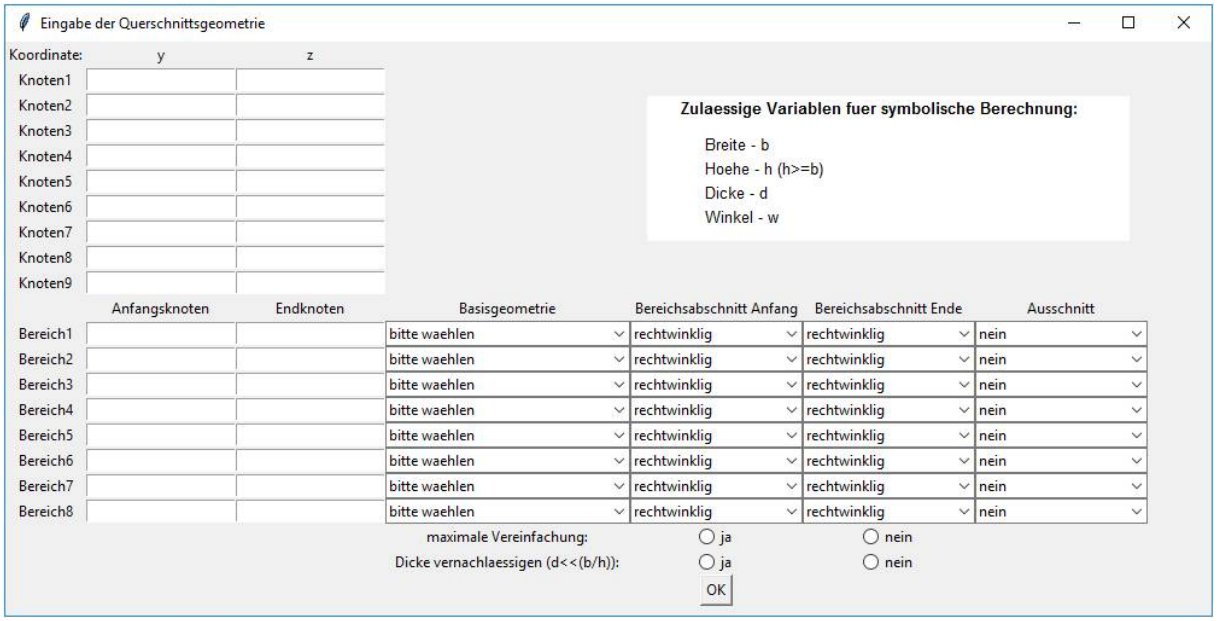
### 2.1.1 Systemeingabe

Drückt man im Mainwindow auf „System-Eingabe“, dann öffnet sich das Fenster zum Eingeben der Querschnitte.




Auswahlbutton Systemeingabe

#### 2.1.1.1 Eingabe von Knoten und Bereichen



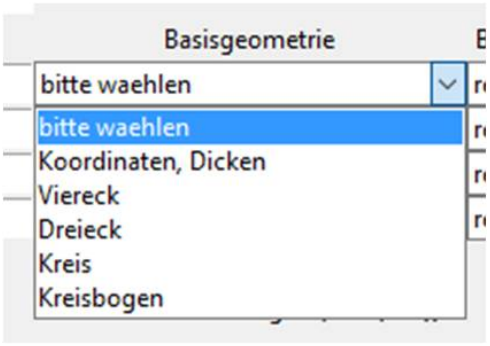

Hauptfenster der Systemeingabe

Im oberen Teil werden die Knoten eingegeben. Diese Knoten haben jeweils eine y - und eine z - Koordinate. Die positive Richtung der y - Koordinate ist nach links, die der z - Koordinate nach unten. Die Strecke zwischen den Knoten ist jeweils ein Bereich, das heißt bei n Bereichen gibt es n+1 Knoten. Diese Strecke ist nicht unbedingt die Achse des Bereiches (Näheres siehe unter „Koordinaten, Dicken“). Es müssen mindestens zwei Knoten eingegeben werden, auch wenn der gesamte Querschnitt über Koordinaten spezieller geometrischer Figuren eingegeben werden sollte. Diese Knoten werden in der Ausgabe kotiert.

Der zweite Block definiert die Bereiche. Dieser Block hat so viele Zeilen wie im Mainwindow unter „Anzahl der Querschnittsbereiche“ angegeben wird. Der obere Block zur Koordinateneingabe besteht demnach aus einer Zeile mehr.

Jede Zeile steht für einen Bereich. Handelt es sich um den Bereich n, kann dieser vom Knoten n-1 bis zum Knoten n+1 gehen. Es ist aber auch möglich, dass dieser zwischen zwei anderen Punkten liegt zum Beispiel bei I- oder T-Profilen. Für jeden Bereich wird nun eine Basisgeometrie bestimmt, dafür steht pro Bereich ein Dropdown-Menü zur Verfügung.

#### 2.1.1.2 Basisgeometrie

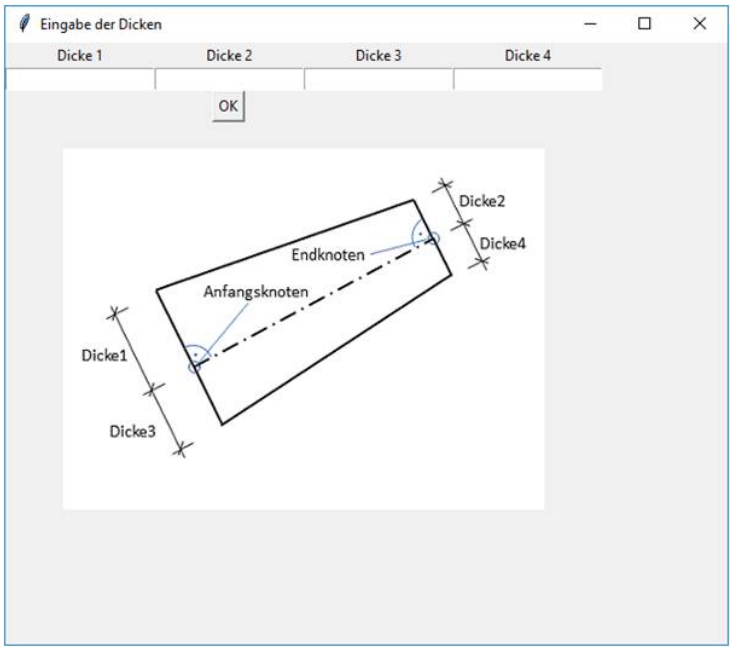



Auswahlmöglichkeiten für die Basisgeometrie

Die Basisgeometrie beschreibt die Geometrie eines Bereichs des zu berechnenden Querschnittes. Je nachdem ob die symbolische oder die numerische Berechnung gewählt wurde, werden Zahlen oder Symbole eingegeben. Als Symbole können „b“, „h“, „d“ und „w“ verwendet werden.

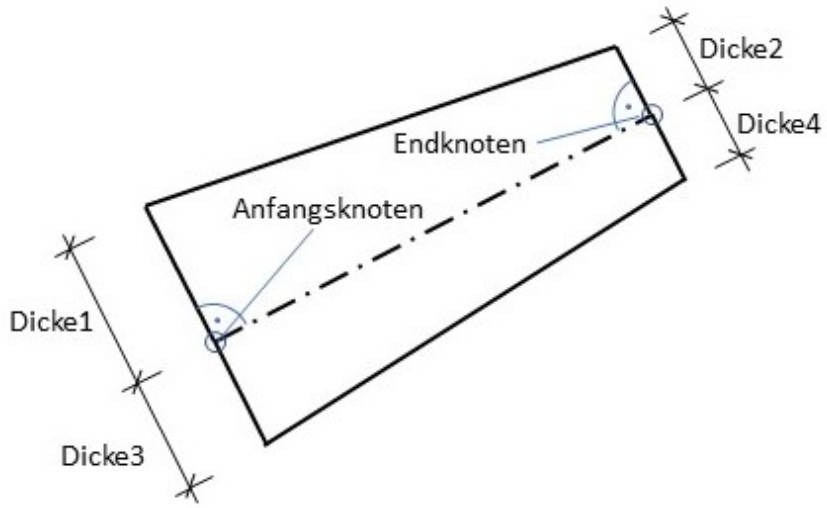
##### 2.1.1.2.1 Koordinaten und Dicken

Bei der ersten Möglichkeit „Koordinaten, Dicken“ bildet die Strecke zwischen den zuvor eingegebenen Koordinaten die Basis. Von dieser weg können vier Dicken eingegeben werden, die normal auf die Verbindungsstrecke der beiden Koordinaten stehen.




Eingabe von Koordinaten und Dicken

Die folgende Abbildung zeigt das Prinzip wie diese Eingabemethode funktioniert.



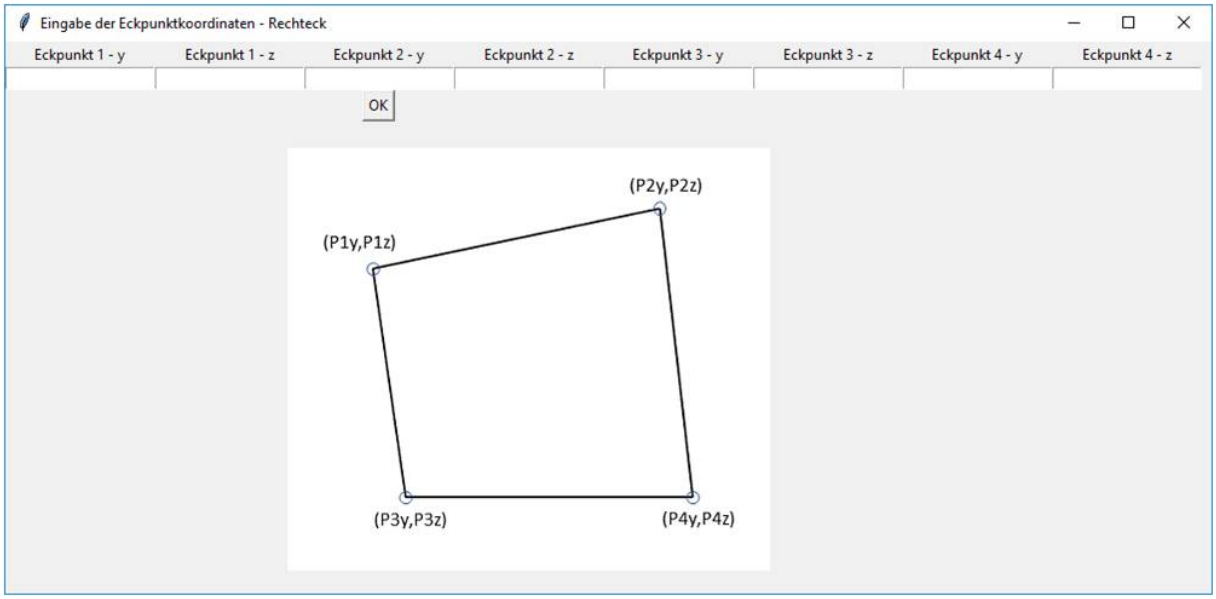

Prinzipskizze zur Eingabe von Koordinaten und Dicken

Wenn „Dicke 1“ und „Dicke 3“ beziehungsweise „Dicke 2“ und „Dicke 4“ jeweils denselben Wert haben, ist die Strecke zwischen der „Anfangsknoten“ und „Endknoten“ die Achse des Querschnittbereiches.

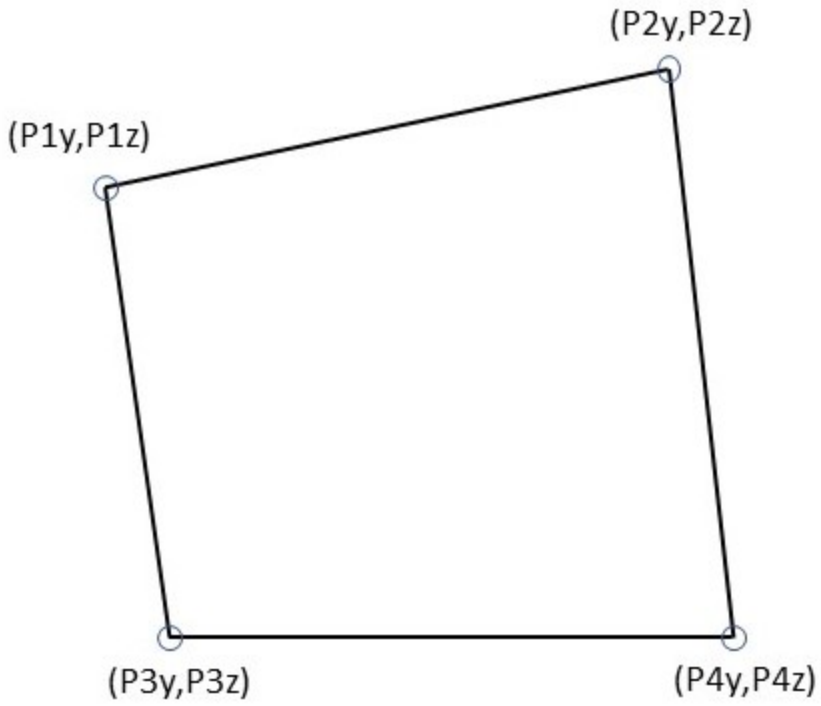
#### 2.1.1.2.2 weitere Eingabeoptionen

Weiters ist es möglich einen Querschnittsbereich mithilfe einer definierten geometrischen Figur anzugeben. Bei Viereck und Dreieck sind jeweils die Eckpunktkoordinaten anzugeben. Beim Kreis werden der Mittelpunkt mit y - und z - Koordinaten und der Radius eingegeben. Zudem kann auch ein Kreisbogen eingegeben werden.

#### 2.1.1.2.3 Viereck

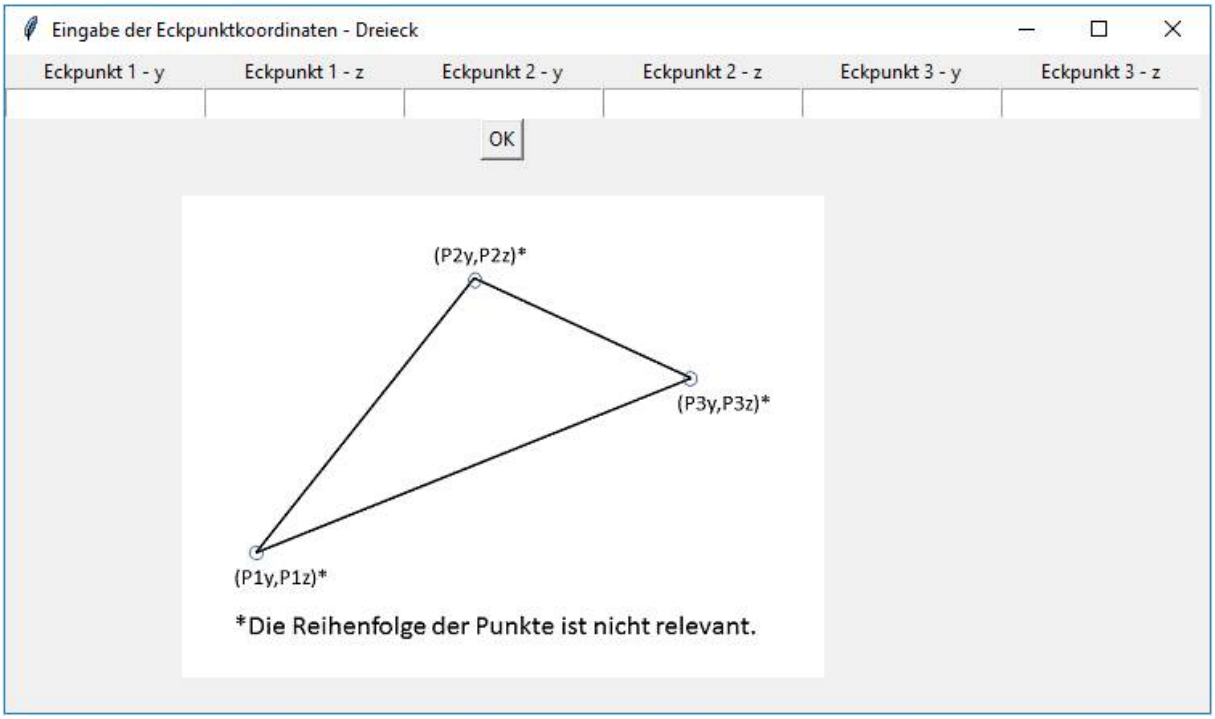



Eingabe eines Vierecks über die Eckpunktkoordinaten

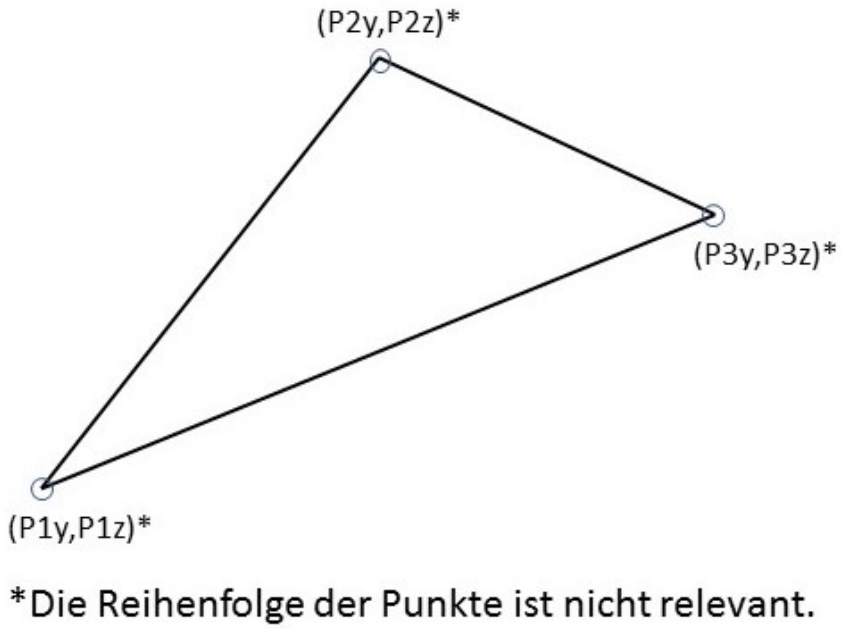



Prinzipskizze zur Eingabe eines Vierecks über die Eckpunktkoordinaten

#### 2.1.1.2.4 Dreieck

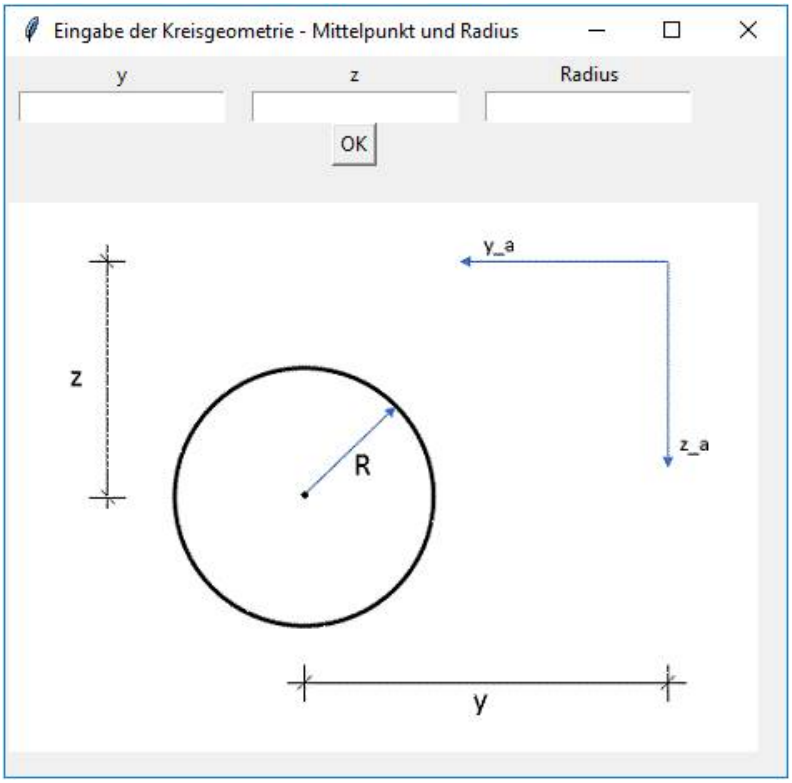



Eingabe eines Dreiecks über die Eckpunktkoordinaten

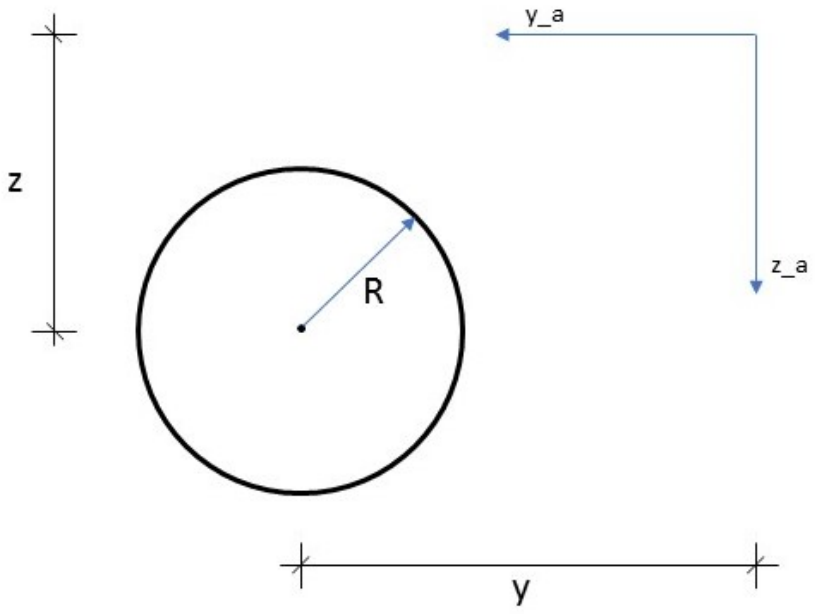



Prinzipskizze zur Eingabe eines Dreiecks über die Eckpunktkoordinaten

#### 2.1.1.2.5 Kreis



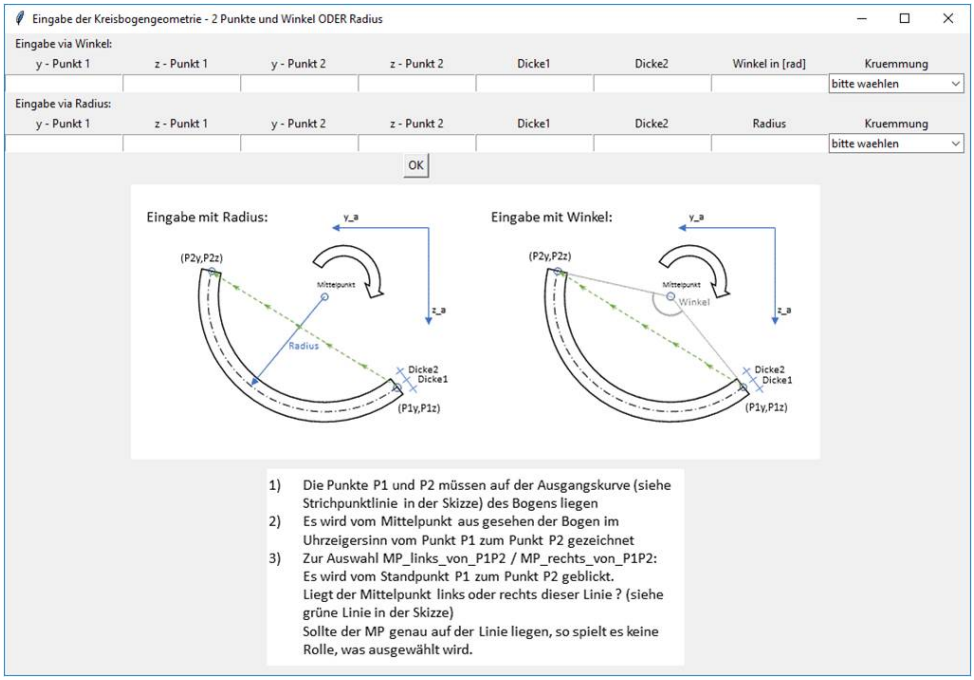

Eingabe eines Kreises über die Mittelpunktkoordinaten und den Radius



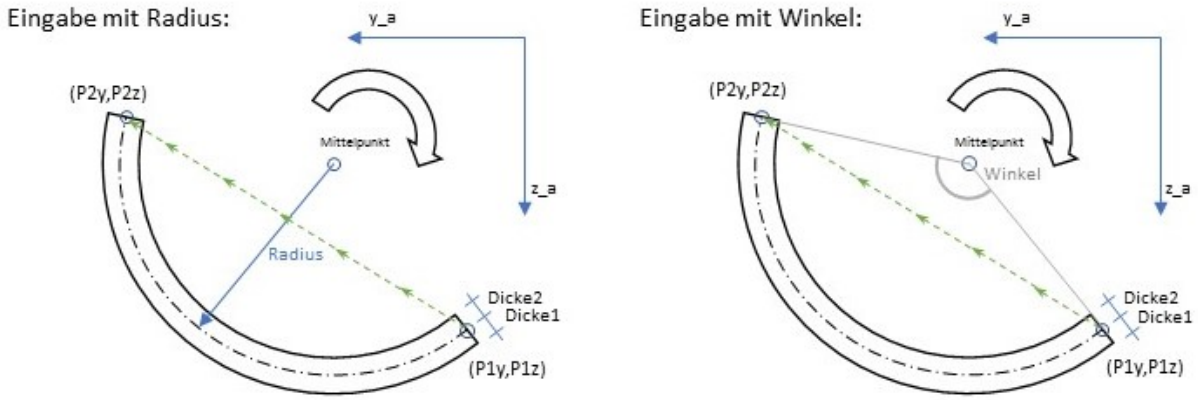

Prinzipskizze zur Eingabe eines Kreises über die Mittelpunktkoordinaten und den Radius

#### 2.1.1.2.6 Kreisbogen

Beim Kreisbogen werden die Anfangs- und Endpunktskoordinaten angegeben. Die beiden Dicken sind so zu verstehen, dass der Bogen, welcher sich über die Eckpunkte und den Radius bildet einmal um die Dicke1 erweitert wird und einmal um die Dicke2 verringert wird. Für den Bogen kann entweder der Öffnungswinkel oder der Radius als Eingabeparameter verwendet werden. Je nachdem, welcher der beiden Parameter gewünscht wird, sollte die obere oder die unter Zeile ausgefüllt werden. Als weiterer Eingabeparameter ist für den Kreisbogen die Krümmung relevant. Hier kann zwischen „MP_links_von_P1P2“ und „MP_rechts_von_P1P2“ gewählt werden. Mit dieser Eingabemethode ist es auch möglich, einen Kreissektor einzugeben.



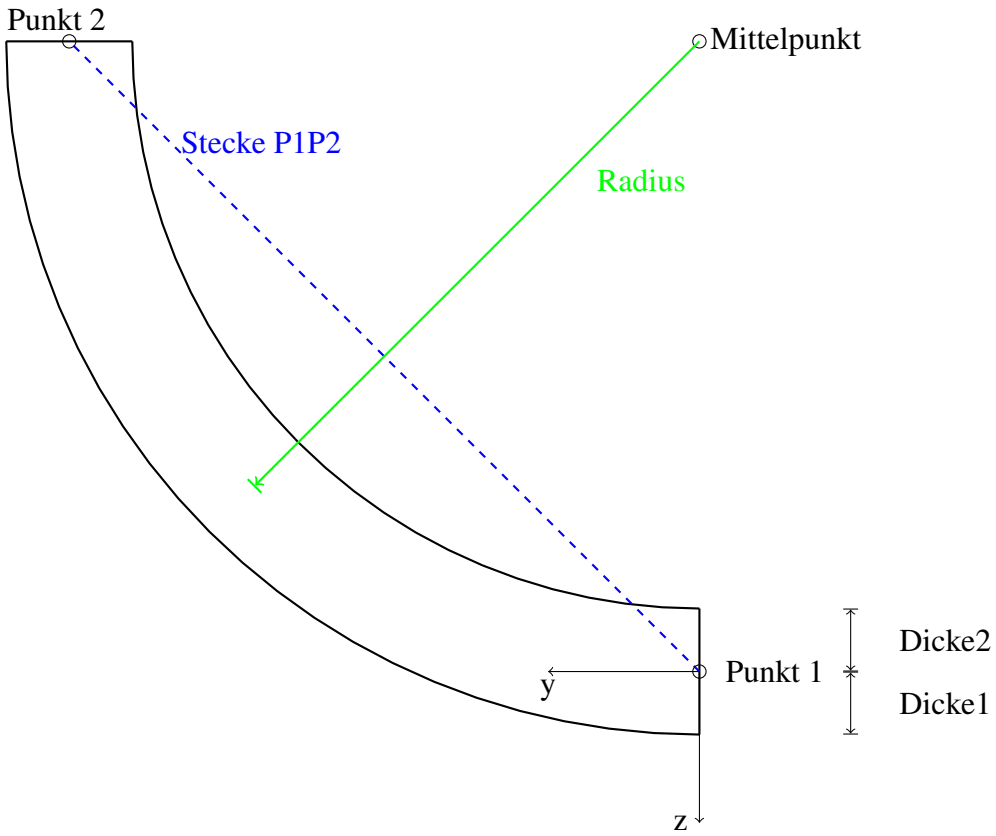

Eingabefenster für den Kreisbogen




Prinzipskizze zur Eingabe des Kreisbogens

Folgend wird ein Eingabebeispiel eines Kreisbogens gezeigt. Die Skizze ist aus der Ausgabe übernommen und mit Erklärungen ergänzt worden.

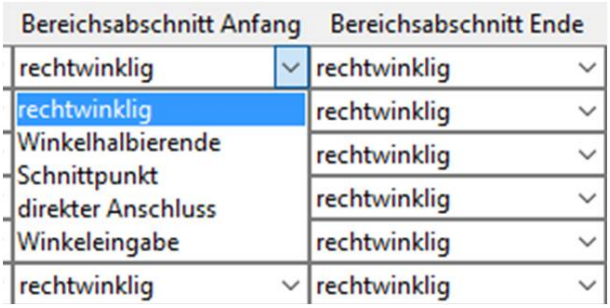

Eingabe des Beispiels



#### 2.1.1.3 Bereichsabschnitt

Diese Optionen dienen dazu, den Randbereich am Knotenanfang und am Knotenende zu modifizieren.

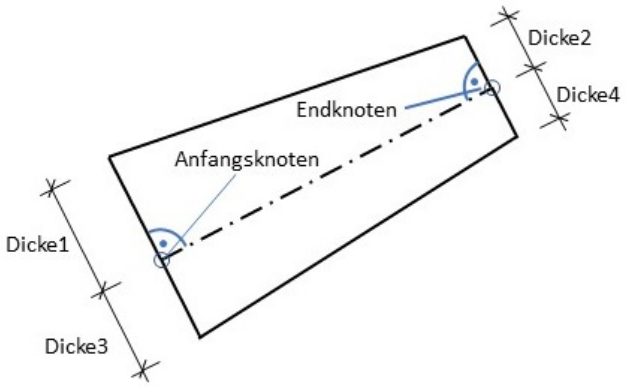
Erstellt wurden die Funktionen für die Anwendung auf die Basisgeometrie “Koordinaten, Dicken“, jedoch ist grundsätzlich auch eine Kombination mit der Basisgeometrie “Viereck“ möglich, wobei dafür die Reihenfolge der Koordinateneingabe eine Rolle spielt. Für alle übrigen Auswahlmöglichkeiten werden diese Optionen ignoriert.



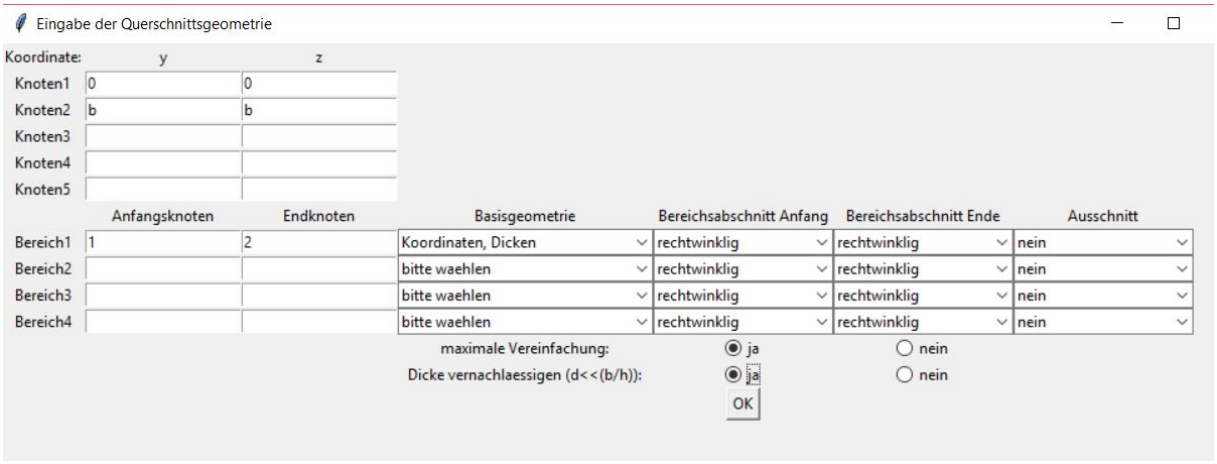

Auswahlmöglichkeit für Abschnitte

#### 2.1.1.3.1 rechtwinklig

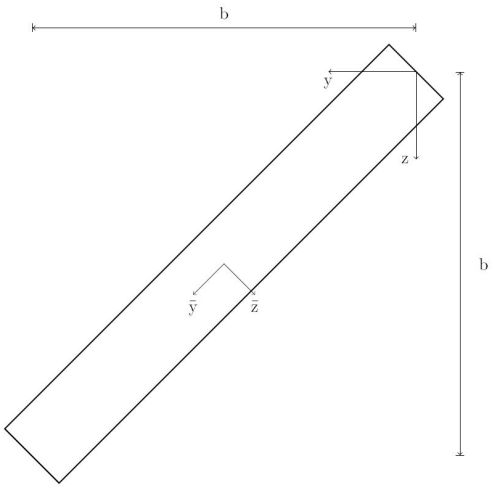
Bei dieser Option wird der Abschnitt des Trägers durch eine Gerade gebildet, welche rechtwinklig auf die Achse steht und durch den Anfangsknoten bzw. Endknoten verläuft.




Prinzipskizze für die Auswahlmöglichkeit “rechtwinklig“



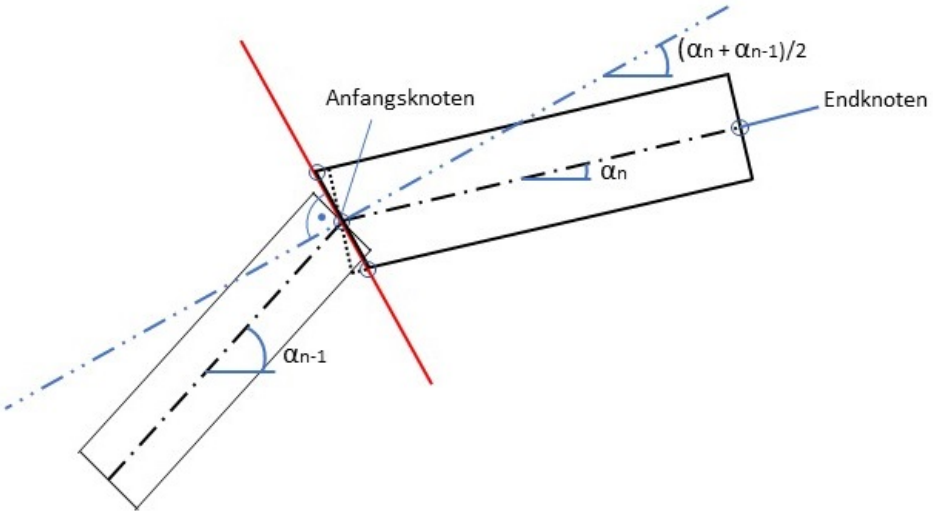

Beispieleingabe



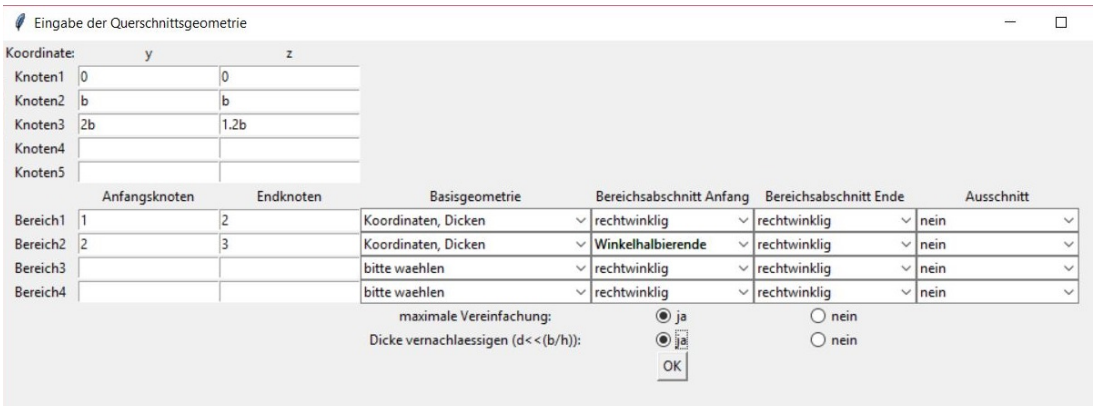

Programmausgabe für die Beispieleingabe

#### 2.1.1.3.2 Winkelhalbierende

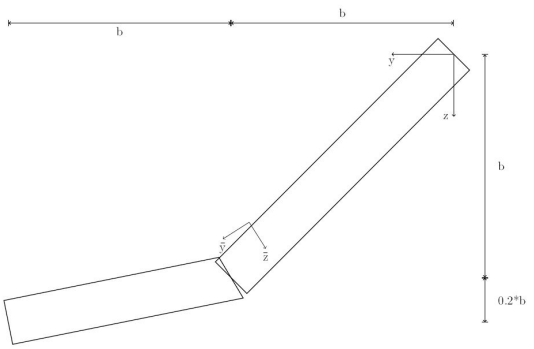
Wird für den “Bereichsabschnitt Anfang“ die Option “Winkelhalbierende“ gewählt, so ist die Schnittlinie die Winkelhalbierende des Winkels zwischen den Achsen. Wird diese Option für den “Bereichsabschnitt Ende“ gewählt, so verläuft die Gerade durch den Knotenendpunkt und der Winkel wird mit dem nachfolgenden Bereich gemittelt.




Prinzipskizze für die Auswahlmöglichkeit “Winkelhalbierende“



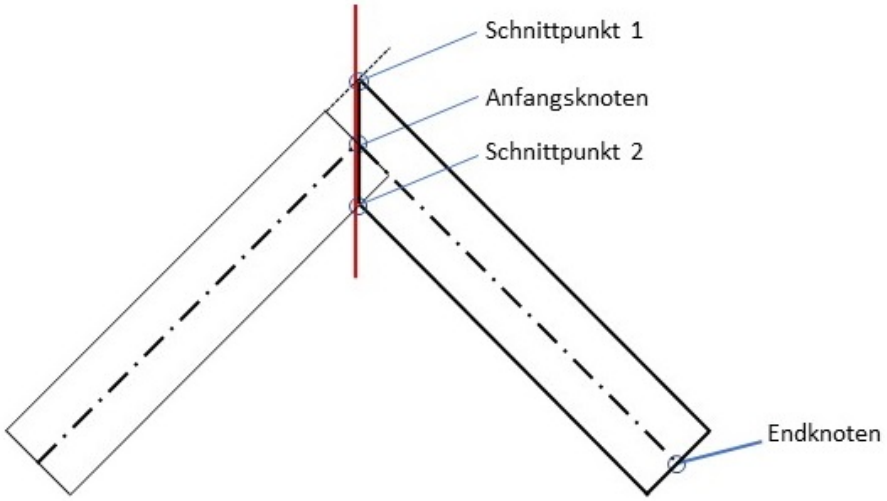

Beispieleingabe



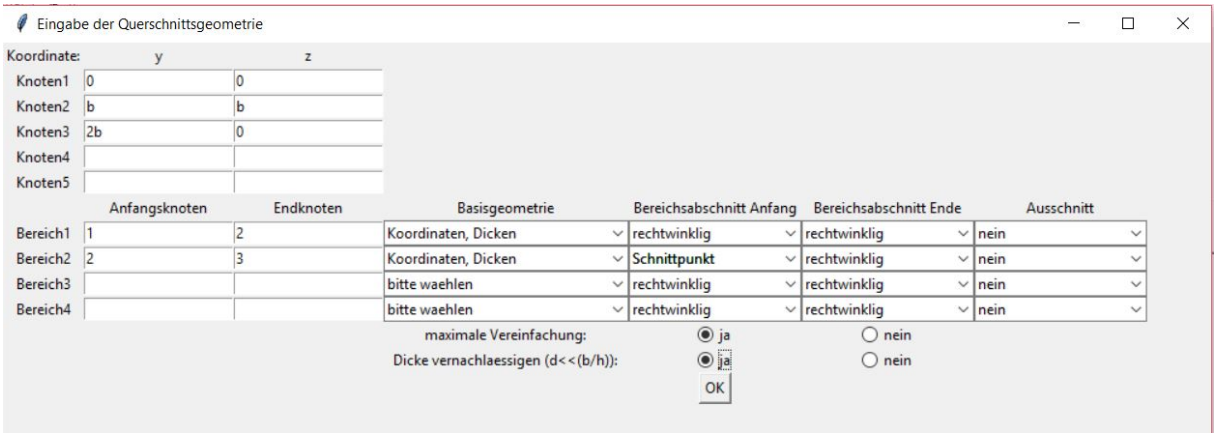

Programmausgabe für die Beispieleingabe

#### 2.1.1.3.3 Schnittpunkt

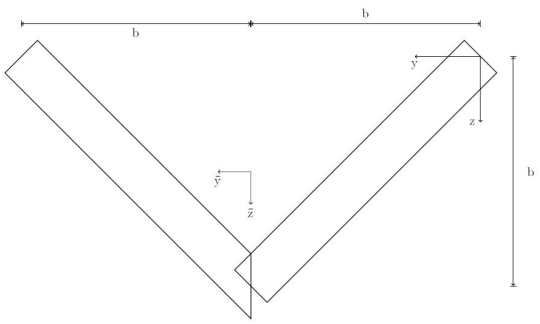
Diese Option berechnet die Schnittpunkte der oberen und unteren Bereichskante mit dem Anschlussbereich. Die Abschnittlinie wird durch eine Gerade gebildet, welche durch beide Schnittpunkte verläuft.




Prinzipskizze für die Auswahlmöglichkeit “Schnittpunkte“



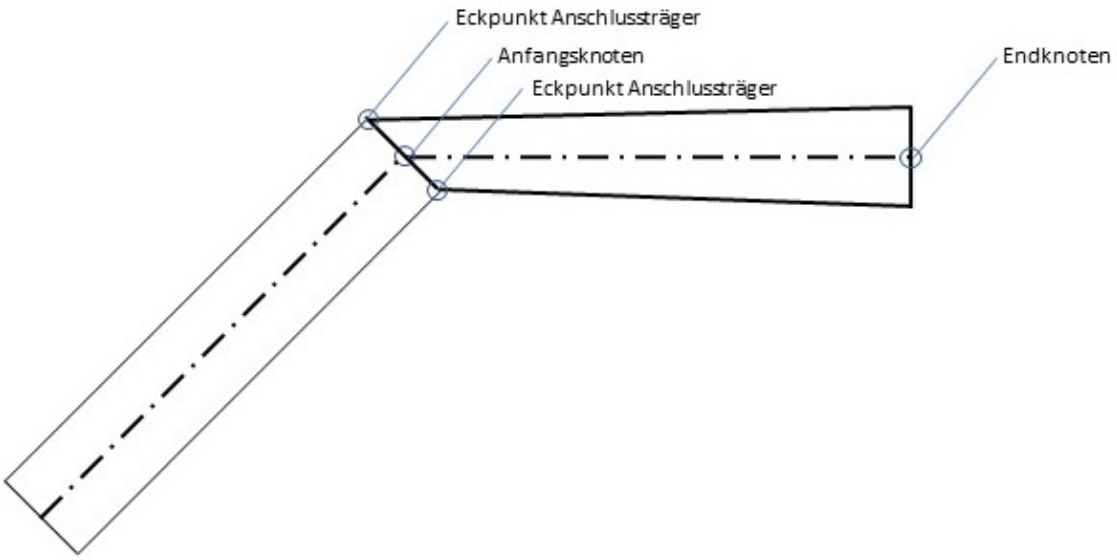

Beispieleingabe für das Ergebnis der Prinzipskizze



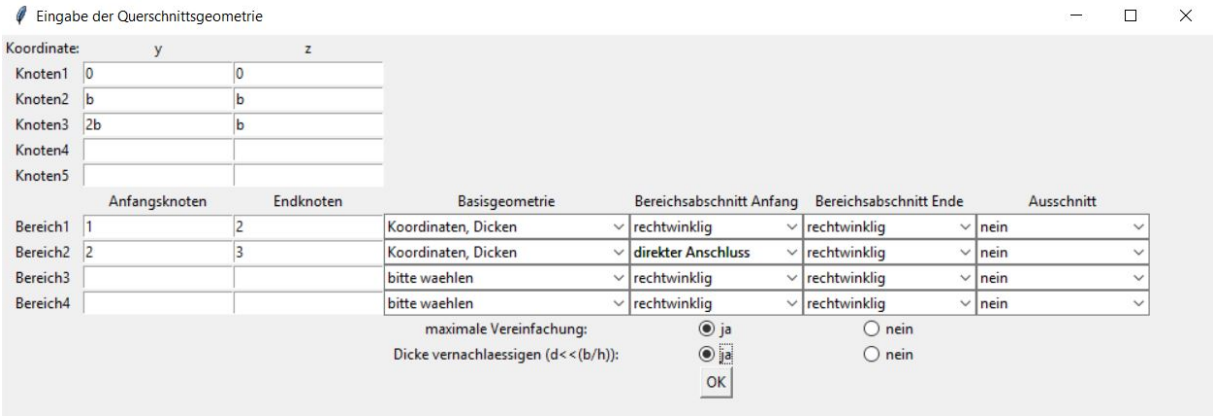

Programmausgabe für die Beispieleingabe

#### 2.1.1.3.4 direkter Anschluss

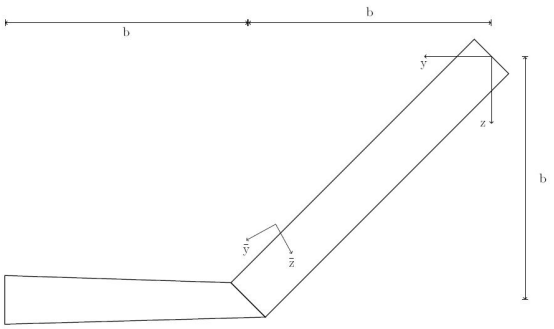
Diese Auswahlmöglichkeit dient dazu, die Eckpunkte des Anschlussträgers direkt zu übernehmen.




Prinzipskizze für die Auswahlmöglichkeit “direkter Anschluss“



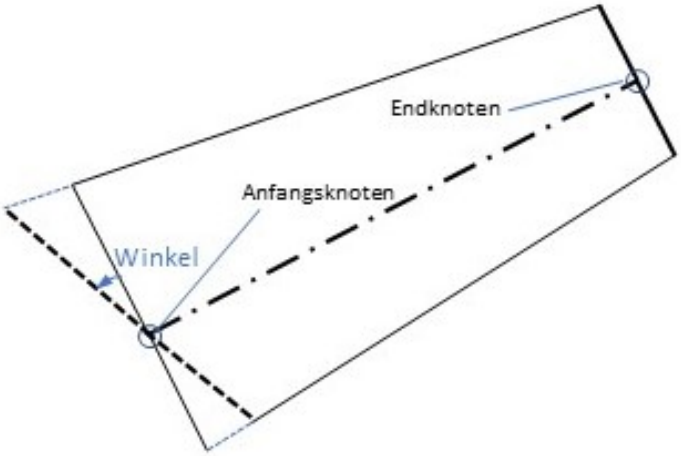

Beispieleingabe



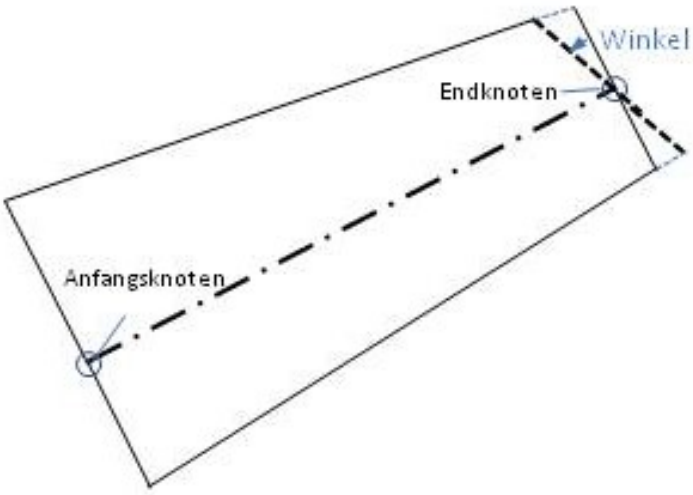

Programmausgabe für die Beispieleingabe

#### 2.1.1.3.5 Winkeleingabe

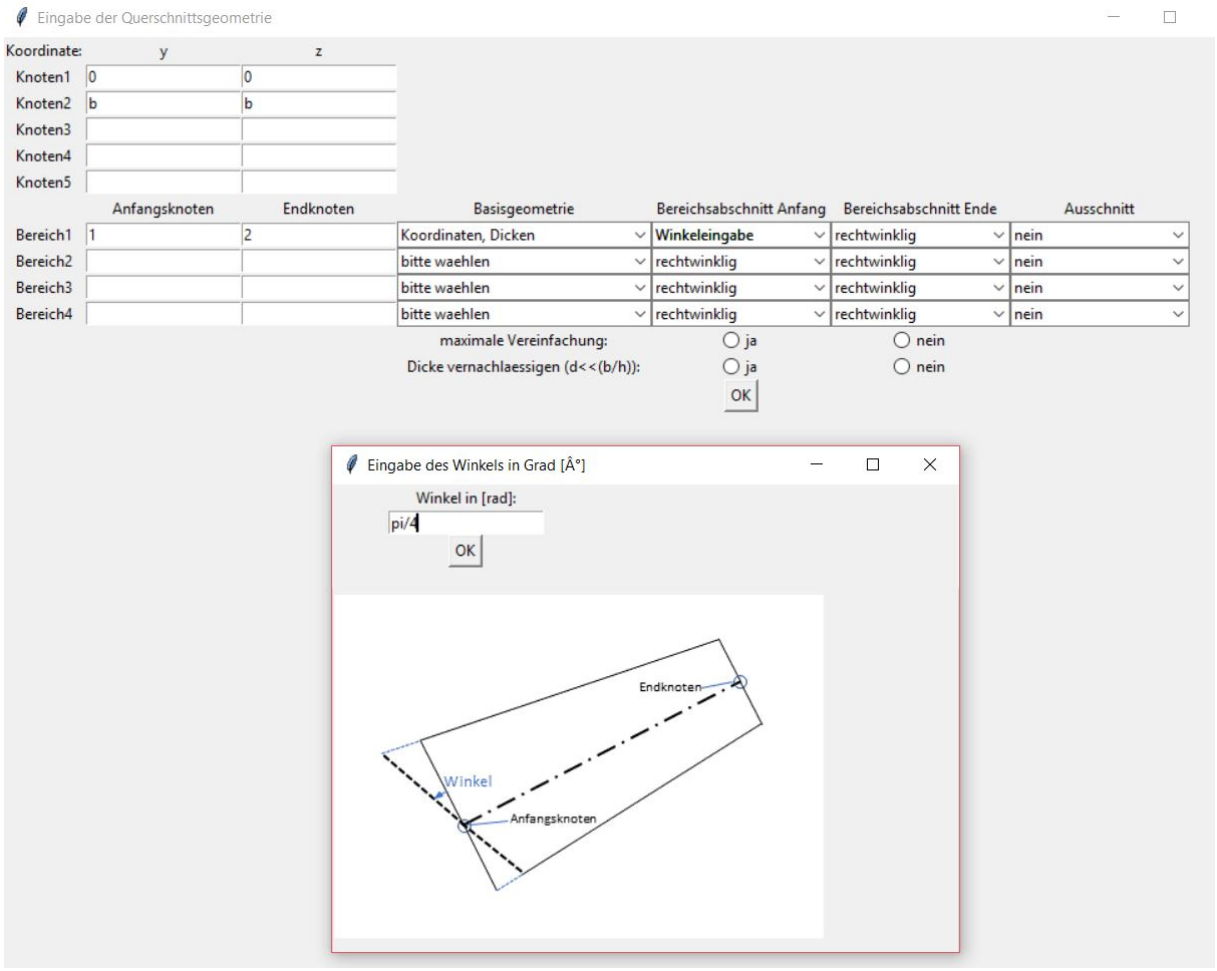
Mittels dieser Auswahl können die Abschnitte um einen bestimmten Winkel gedreht werden. Dabei wird als Basis ein rechtwinkliger Abschnitt verwendet, welcher um den Anfangsknoten bzw. Endknoten gedreht wird. Die Drehrichtung ist dabei mathematisch positiv (gegen den Uhrzeigersinn). Zu beachten ist, dass bei der numerischen Eingabe der Winkel in Grad eingegeben werden muss. Bei der symbolischen Eingabe hingegen erfolgt die Eingabe in Radiant - dabei ist die Variable “w“ oder ein Zahlenwert zu verwenden.



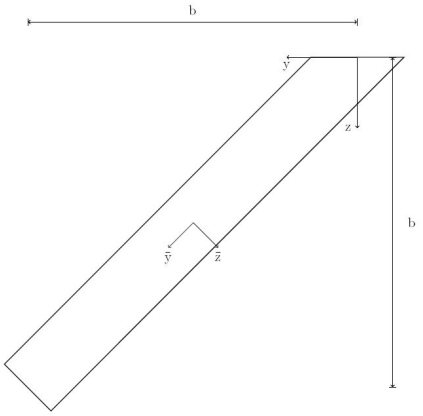

Prinzipskizze für die Auswahlmöglichkeit “Winkeleingabe“ am Knotenanfang




Prinzipskizze für die Auswahlmöglichkeit “Winkeleingabe“ am Knotenende



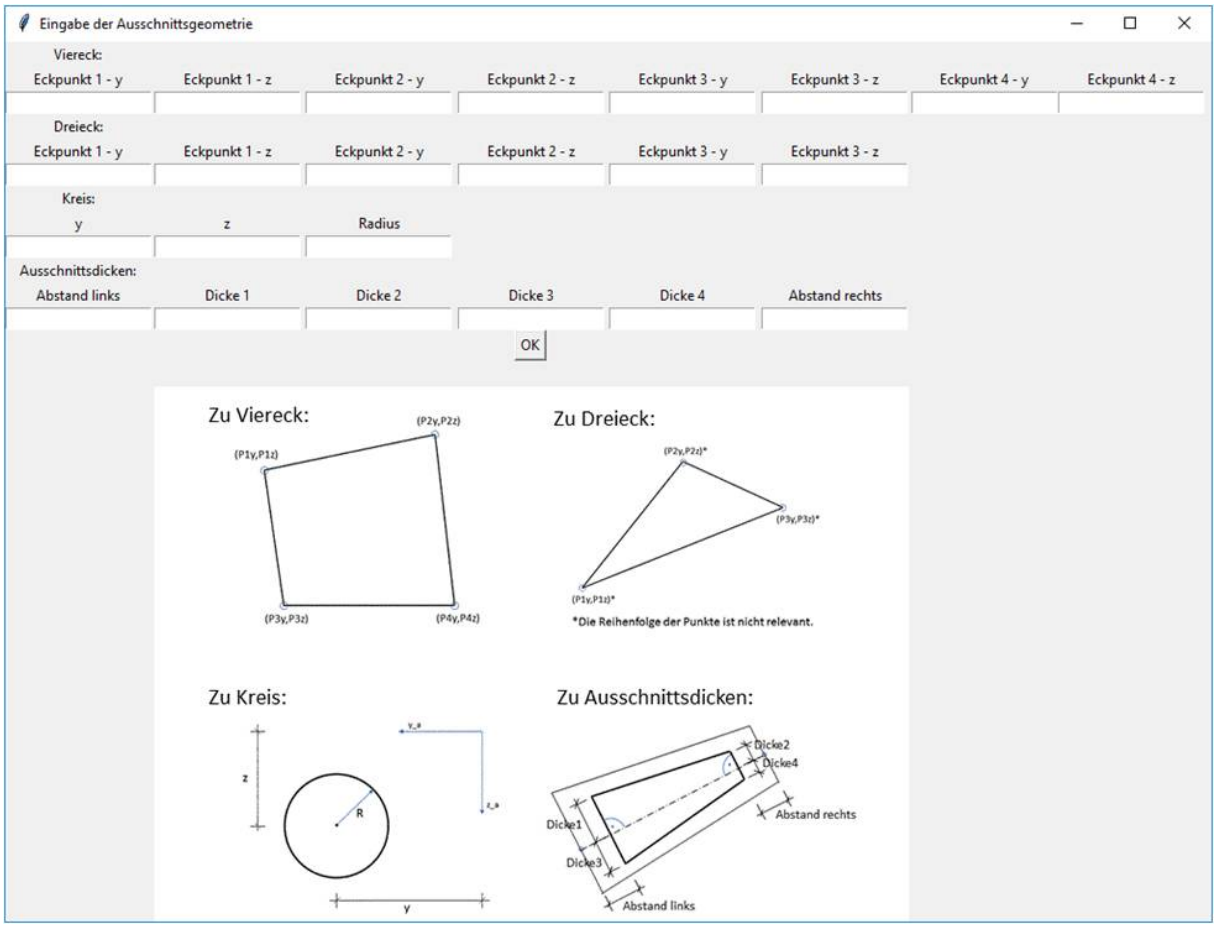

Beispieleingabe



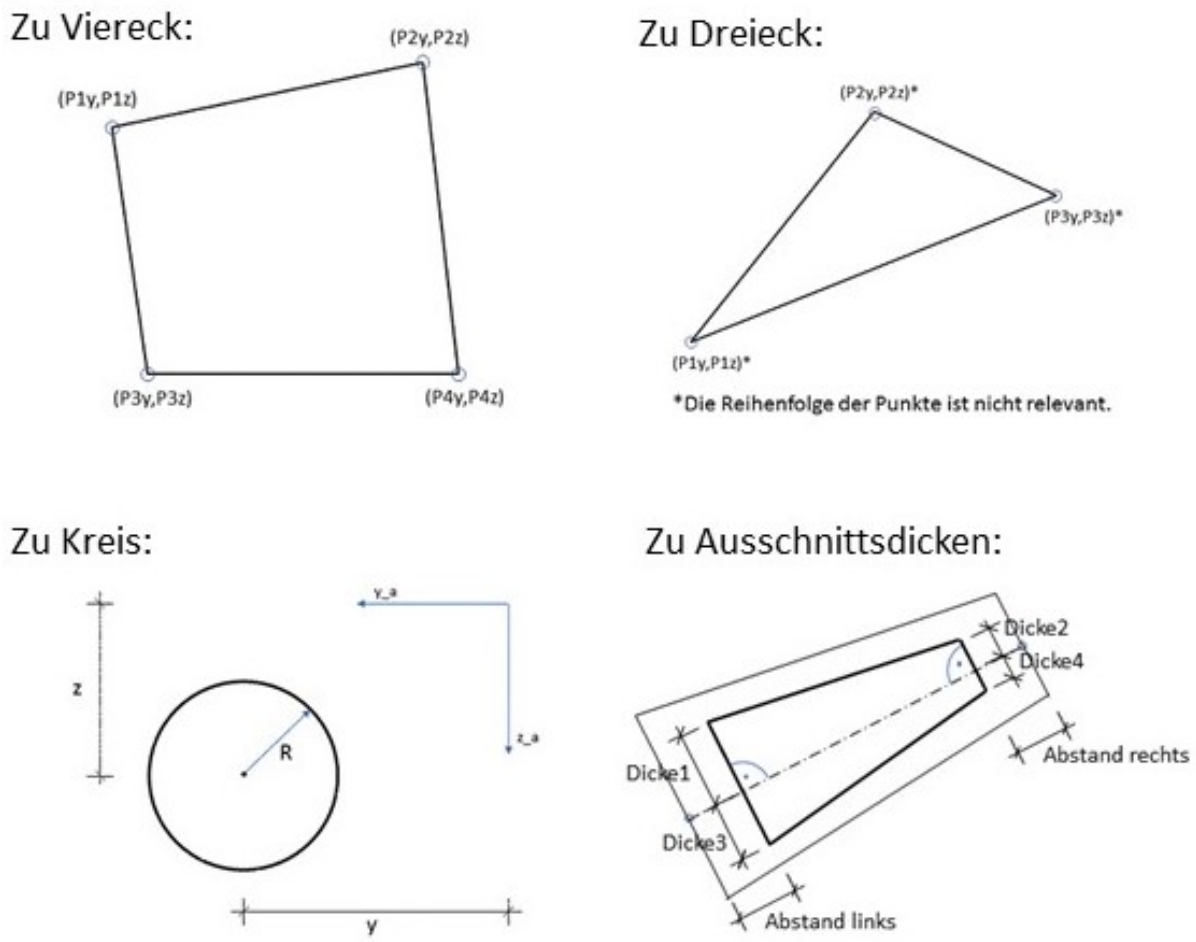

Programmausgabe für die Beispieleingabe

#### 2.1.1.4 Ausschnitte

Falls der Querschnitt Ausschnitte hat, sind auch diese bereichsweise anzugeben. Hierzu wird bei der Abfrage zu den Ausschnitten „ja“ ausgewählt. Folglich öffnet sich ein Fenster und es kann der Ausschnitt in Form von Rechteck, Dreieck und Kreis angegeben werden oder es werden vier Dicken eingegeben und der Abstand von den Rändern.




Eingabefenster für Ausschnitte



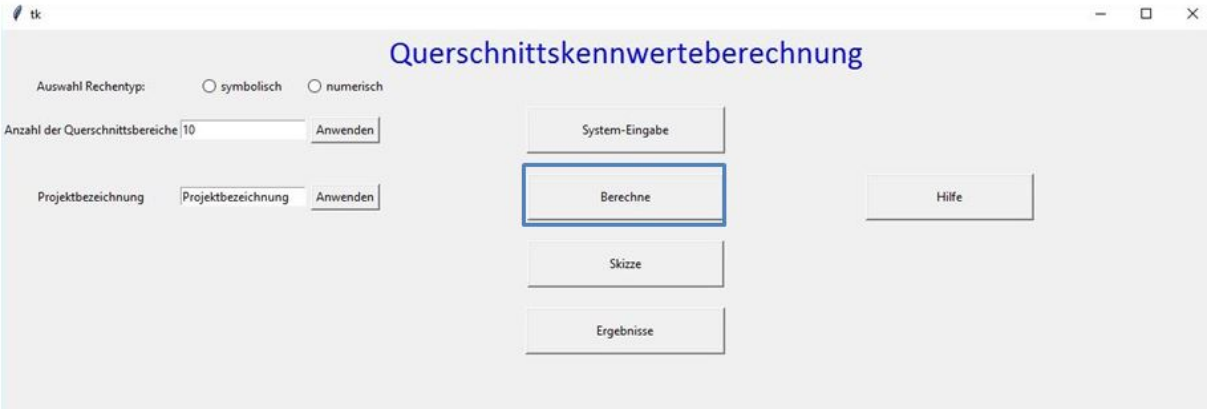

Prinzipskizzen zur Eingabe von Ausschnitten

#### 2.1.1.5 Abschluss der Eingaben

Alle Fenster sind nach dem Ausfüllen mit Klick auf „OK“ zu bestätigen. Bei der symbolischen Berechnung kann noch optional ausgewählt werden, dass das Ergebnis maximal vereinfacht werden sollte, beziehungsweise dass die Dicke vernachlässigt werden soll. Ersteres bedeutet, dass die Wurzeln, trigonometrischen Funktionen etc. zu Dezimalfaktoren vor den symbolischen Ausdrücken zusammengefasst werden. Die Dicke kann vernachlässigt werden, wenn gilt d<<(b/h). Dies trifft häufig auf typische Walzprofile zu.

### 2.1.2 Berechnung

Wenn alle Daten des Querschnitts in der System-Eingabe erfasst wurden und man diese mit „OK “bestätigte, dann ist der nächste Schritt im Mainwindow den Button „Berechnung“ zu drücken.




Berechnungsbutton im Mainwindow

Nun werden die Querschnittskennwerte automatisch berechnet.

### 2.1.3 Skizze

Klickt man nach der Berechnung auf den Button „ Skizze“ öffnet sich eine pdf-Datei in der eine Skizze des Querschnitts zu sehen ist. Diese Skizze wird auch im Ausgabedokument mit den Teil- und Endergebnissen ausgegeben.

### 2.1.4 Ergebnisse

Dieses Dokument wird aufgerufen, wenn im Mainwindow auf den Button „Ergebnisse“ gedrückt wird. In diesem Dokument werden die Projektbezeichnung, die Querschnittskizze (diese wird im Rahmen der Berechnung automatisch mit tikz, also mit LaTeX erstellt) und die Teil- und Endergebnisse der Querschnittskennwerte angegeben. Dieses Dokument wird mit LaTeX erstellt und automatisch geöffnet (pdf-Reader ist erforderlich). Es sei darauf hingewiesen, dass diese Datei bei einer neuen Eingabe überschrieben wird, sofern zuvor nicht der Dateiname oder der Speicherort individuell geändert wurde.
Falls der Speicherort der .tex-Dateien geändert wird, sollten alle .tex-Dateien in den neuen Zielordner gebracht werden, da ansonsten die Skizze nicht mehr dargestellt werden kann. Außerdem den Dateinamen „QSSK_BAC_Projekt_16_Ausgabe_01.tex“ aus diesem Grund bitte nicht ändern.

# 3 FUNKTIONEN

Folgend wird immer wieder ein Teil des Programmcodes dargestellt. Aufgrund der LaTeX- Formatierung muss hierfür der Programmcode umgebrochen werden. Hierzu wurde im Code folgendes Zeichen eingeführt: „\n“. Die Umbrüche, welche in den Beispielsequenzen in diesem Kapitel auf „\n“ folgen, sind im tatsächlich ausführbaren Programmcode nicht enthalten.

## 3.1 Eingabe

### 3.1.1 Klassen

Die Eingabe der Parameter zur Querschnittskennwertberechnung erfolgt über eine graphische Benutzeroberfläche, die mittels TKinter (GUI) programmiert wurde.
Die Programmstruktur hierzu ist folgender Maßen aufgebaut:
Jedes Fenster, das sich in der Eingabe öffnen lässt, ist eine eigene Klasse. Die Klasse „MainWindow“ ist beispielsweise die Klasse, die für das Hauptfenster zuständig ist, das beim Öffnen des Programmes als Erstes erscheint. Aus dieser Klasse heraus, werden auch die anderen Klassen angesprochen, die wiederum für ein separates Fenster zuständig sind, wie beispielsweise die Systemeingabe oder auch die Fenster innerhalb dieser, wie „Eingabe Koordinaten, Dicken“. Die Abschnitte der Bereiche links und rechts (Dropdown-Menü) sind Teil der Klasse „PopupWindow“ (Klasse für das Fenster der Systemeingabe).
Jeder Button greift auf eine eigene Klasse zu, das heißt, dass zum Beispiel beim Anklicken des Buttons „Hilfe“ die Klasse „class Hilfe ( ) :“ ausgeführt wird. Diese Klasse ist dafür zuständig, dass sich die .pdf-Datei „HowTo “ öffnet. Im Programmcode sieht dies folgendermaßen aus:

```
class Hilfe():

    def __init__(self,master):
        os.startfile("HowTo.pdf")
```

Folgend ist der Programmcode des ersten Teils der Klasse „MainWindow“ dargestellt. Anschließend werden die wichtigsten Zeilen kurz erklärt:

```
class MainWindow(PopupWindow):
    def __init__(self,master):
        self.master = master
        self.master.geometry("1200x1200")
        self.intro = Label(master, text= \n
        "Querschnittskennwerteberechnung",\n
        fg="blue", font="Calibri␣24")
        self.intro.grid(row=1, column=6)
        self.eingabe = Button(master, text= \n
        "System-Eingabe", command=self.popup)
        self.eingabe.grid(row=8, column=6, ipadx=50, \n
        ipady=10, pady=10)
        self.berechne = Button(master,\n
        text="Berechne", command=self.berechne)
        self.berechne.grid(row=11, column=6, \n
        ipadx=67, ipady=10, pady=10)
```

```
self.skizze = Button(master, text="Skizze", \n
command=self.zeichne)
self.skizze.grid(row=14, column=6, ipadx=76, \n
ipady=10, pady=10)
self.ergebnisse = Button(master,\n
text="Ergebnisse", command=self.ergebnisse)
self.ergebnisse.grid(row=15, column=6,\n
ipadx=65, ipady=10, pady=10)
self.Hilfe = Button(master, text="Hilfe", \n
command=self.Hilfestellung)
self.Hilfe.grid(row=11, column=8, ipadx=65, \n
ipady=10, pady=10)

#Eingabe der Anzahl der Querschnittsbereiche \n
#zur weiteren Verwendung zur Geomentrieengabe

Label(master, text= \n
"Anzahl der Querschnittsbereiche")\n.
grid(row = 8, column = 1)
self.querschnittsbereiche_anz = Entry(master)
self.querschnittsbereiche_anz.insert(0,10)
self.querschnittsbereiche_anz.grid(row = 8, \n
column = 2)

self.button2=Button(text='Anwenden', \n
command=self.anwenden)
self.button2.grid(row=8, column=3)
```

„self.master.geometry("1200x1200")“: Hier wird die Größe des Fensters festgelegt.
„self.eingabe = Button(master, text=“System-Eingabe“, command=self.popup)“: Festlegung des Buttons zur System-Eingabe. In der Zeile darunter werden die Parameter des Buttons (Position und Größe) definiert. Analog in den weiteren Zeilen für Berechnung, Skizze, Ergebnisse und Hilfe.
Der darauf folgende Block ist für die Eingabe der Anzahl der Querschnittsbereiche verantwortlich. Die ersten vier Zeilen sind für das Eingabefeld an sich zuständig und die Zeilen mit „self.button2...“ für den Knopf „Anwenden“.
Zu beachten ist, dass die Variablen, unter denen die verschieden Eingabeparameter gespeichert werden, innerhalb des Programms „global“ sind. Nur so, kann in nachfolgenden Klassen (zum Beispiel in der Berechnung) darauf zugegriffen werden. Dieses „Globalsetzen“ geschieht hauptsächlich in der Klasse „PopupWindow“, also jener, die für das Fenster der Systemeingabe zuständig ist. In einigen Ausnahmefällen (z.B. beim Kreisbogen) aber auch direkt in der jeweiligen Klasse. Folgend ist das Definieren der Variablen für die Dicken und das zugehörige „Globalsetzen“ dargestellt.

```
global dicke1
dicke1 = {}
global dicke2
dicke2 = {}
global dicke3
dicke3 = {}
global dicke4
dicke4 = {}
```

#### 3.1.1.1 Funktionen

In den einzelnen Klassen gibt es Funktionen. In den Klassen, die ein Eingabefenster darstellen ist zumindest eine Funktion zum Exportieren der Eingaben vorhanden. Für den Bereichsabschnitt rechts via Winkeleingabe sieht diese Funktion wie folgt aus:

```
def exportvalue(self):
    winkel_re[winkel_re_index] = \n
    self.winkel_re.get()
    self.top.destroy()
```

Analog ist dies auch in den anderen Klassen aufgebaut. Mit „self.top.destroy()“ wird das Fenster bei Klick auf „OK“ automatisch geschlossen.
In der Klasse „PopupWindow“ sind noch weitere Funktionen vorhanden, die ausgewählte Eingaben für folgende Klassen (Berechnung) speichern. Zum Beispiel ist die folgende Funktion für die Auswahl der Basisgeometrie zuständig.

```
def basisgeom_selection(self,event):
    cb=event.widget
    global cb_index
    cb_index = list(self.basisgeometrie.values())\n
    .index(cb) + 1
    if cb.get()=="Koordinaten, Dicken":
        self.dw=DickenWindow(self)
    if cb.get() == "Viereck":
        self.vw=ViereckWindow(self)
    if cb.get() == "Dreieck":
        self.dw=DreieckWindow(self)
    if cb.get() == "Kreis":
        self.cw=CircleWindow(self)
    if cb.get() == "Kreisbogen":
        self.aw=ArcWindow(self)
```

Ebenso sind in der Klasse „MainWindow“ ähnliche Funktionen vorhanden. Exemplarisch sollte jene gezeigt werden, die für die Übernahme der Projektbezeichnung zuständig ist:

```
def projektbezeichnung_druck(self):
    global projektbezeichnung_name
    projektbezeichnung_name=self. \n
    projektbezeichnung.get()
    print(projektbezeichnung_name)
    if projektbezeichnung_name == \n
```

```
        "Projektbezeichnung":
            projektbezeichnung_name = \n
            "Es wurde keine spezifische" \n
            "Projektbezeichnung angegeben!"
```

## 3.2 Berechnung

### 3.2.1 Klasse: BerechneSymoblic(MainWindow)

#### 3.2.1.1 Ablauf

Als erster Schritt werden die Eingabewerte übernommen, indem das Dictionary “Bereicheingabe“ in zwei neue Dictionarys gespeichert wird. Dafür werden positive Geometrien in “value“ und Ausschnitte in “ausschnitte“ gespeichert. Anhand des Eintrags “Basisgeometrie“, welcher bei der Eingabe ausgewählt wurde, werden die Werte der entsprechenden Geometrie zugeordnet. Anschließend wird für den Fall, dass ein Abschnitt modifiziert wurde, dieser mittels der Funktion “abschnittrechts“ bzw. “abschnittlinks“ angepasst. Diese Funktion ändert direkt im Dictionary die entsprechenden Eckpunkte.

```
dictionary = Bereicheingabe
listekoordinateny_edit=listekoordinateny
listekoordinatenz_edit=listekoordinatenz
listeverbindungen_edit=KnotenBereiche
values = {}
ausschnitte={}

anzahlstaebe=len(listeverbindungen_edit)
counter = 1

#Gehe die Staebe der Reihe nach durch
for num in range (1,anzahlstaebe+1):
    #Erstelle ein dictionary fuer die positiven Eingaben
    values[num] = {}
    #Erstelle ein dictionary fuer die Ausschnitte
    ausschnitte[num]={}

    if num in dictionary:
        counter=num
        #stabwerte ... dictionary mit den jeweiligen werten
        stabwerte=dictionary[num]
        knotenanfang =listeverbindungen_edit[counter-1][0]
        knotenende =listeverbindungen_edit[counter-1][1]
        koordinatenyanfang=listekoordinateny_edit[knotenanfang-1]
        koordinatenzanfang=listekoordinatenz_edit[knotenanfang-1]
        koordinatenyende=listekoordinateny_edit[knotenende-1]
        koordinatenzende=listekoordinatenz_edit[knotenende-1]
        geometrie = stabwerte["basisgeometrie"]

        values[num]['abschnitt_links'] = stabwerte['abschnitt_links']
        values[num]['abschnitt_rechts'] = stabwerte['abschnitt_rechts']
        values[num]["basisgeometrie"]= geometrie
        values[num]["koordinatenyanfang"]=koordinatenyanfang
```

```
values[num]["koordinatenzanfang"] = koordinatenzanfang
values[num]["koordinatenyende"] = koordinatenyende
values[num]["koordinatenzende"] = koordinatenzende
values[num]["winkel"]= stabwerte["winkel"]

if geometrie=="Koordinaten, Dicken":
    values[num]["dicke"] = stabwerte["dicke"]
    values[num]= eckpunkte(values[num])
elif geometrie=="Viereck":
    values[num]['koordinatenviereck'] \n
    = stabwerte['koordinatenviereck']
    values[num] = eckpunktekoordinaten(values[num])
elif geometrie=="Dreieck":
    values[num]['koordinatendreieck'] \n
    = stabwerte['koordinatendreieck']
    values[num] = eckpunktekoordinaten(values[num])
elif geometrie=="Kreis":
    values[num]['koordinatenkreis'] \n
    = stabwerte['koordinatenkreis']
elif geometrie =="Kreisbogen":
    values[num]['wertekreisbogen'] \n
    = stabwerte['wertekreisbogen']
    values[num]["Eingabe"] =  stabwerte['Eingabe']

else:
    print ("Error")

#Ausschnitte:
if stabwerte["Ausschnittjn"] == "ja":
    if stabwerte["Ausschnitt"]== "Koordinaten, Dicken":
        ausschnitte[num]["koordinatenyanfang"] \n
        = koordinatenyanfang
        ausschnitte[num]["koordinatenzanfang"] \n
        = koordinatenzanfang
        ausschnitte[num]["koordinatenyende"] \n
        = koordinatenyende
        ausschnitte[num]["koordinatenzende"] \n
        = koordinatenzende
        ausschnitte[num]['Ausschnittdicken'] \n
        = stabwerte['Ausschnittdicken']
        ausschnitte[num]["basisgeometrie"] \n
        = stabwerte['Ausschnitt']
        ausschnitte[num] \n
        = eckpunkteausschnittdicken(ausschnitte[num])
    elif stabwerte["Ausschnitt"]=="Rechteck":
        ausschnitte[num]['koordinatenviereck'] \n
        = stabwerte['Ausschnittrechteck']
        ausschnitte[num]["basisgeometrie"] \n
        = stabwerte['Ausschnitt']
        ausschnitte[num] = eckpunktekoordinaten(ausschnitte[num])
    elif stabwerte["Ausschnitt"] == "Dreieck":
        ausschnitte[num]['koordinatendreieck'] \n
        = stabwerte['Ausschnittdreieck']
        ausschnitte[num]["basisgeometrie"] \n
        = stabwerte['Ausschnitt']
        ausschnitte[num] = eckpunktekoordinaten(ausschnitte[num])
    elif stabwerte["Ausschnitt"] == "Kreis":
```

```
                ausschnitte[num]['koordinatenkreis'] \n
                = stabwerte['Ausschnittkreis']
                ausschnitte[num]["basisgeometrie"] \n
                = stabwerte['Ausschnitt']
            else:
                pass
            ausschnitte[num]= StatMoment(ausschnitte[num])

        else:
            pass

for num in range(1, anzahlstaebe + 1):
    stabwerte = values[num]
    abschnittlinks_str = stabwerte['abschnitt_links']
    abschnittrechts_str = stabwerte['abschnitt_rechts']
    if  abschnittlinks_str != "rechtwinklig" and abschnittlinks_str \n
    != "Winkeleingabe" and abschnittlinks_str != "direkter␣Anschluss":
        stabwerte = abschnittlinks(stabwerte,values[num-1])
    else:
        pass

    if  abschnittrechts_str != "rechtwinklig" and abschnittrechts_str \n
    != "Winkeleingabe" and abschnittrechts_str != "direkter␣Anschluss":
        stabwerte = abschnittrechts(stabwerte,values[num+1])
    else:
        pass

for num in range(1, anzahlstaebe + 1):
    stabwerte = values[num]
    abschnittlinks_str = stabwerte['abschnitt_links']
    abschnittrechts_str = stabwerte['abschnitt_rechts']
    if abschnittlinks_str == "direkter␣Anschluss":
        stabwerte = abschnittlinks(stabwerte,values[num-1])
    else:
        pass

    if abschnittrechts_str == "direkter␣Anschluss":
        stabwerte = abschnittrechts(stabwerte,values[num+1])
    else:
        pass
    stabwerte = StatMoment(stabwerte)
```

Als Nächstes wird die Funktion “StatMoment“ auf jeden Eintrag in beiden Dictionarys angewendet. Hierbei wird wiederum direkt im Dictionary der Eintrag “Sy“ und “Sz“ und “Flaeche“ hinzugefügt. Nun werden die Einträge für das Statische Moment aufsummiert. Sollte bei der Eingabe eine Leerzeile vorhanden sein, so wird die Schleife aufgrund der Verwendung von try & except trotzdem stabil bleiben.

```
sy = 0
sz = 0
flaeche = 0
for counter in range(1,anzahlstaebe+1):
    try:
        sy+=values[counter]["Sy"]
        sz+=values[counter]["Sz"]
```

```
            flaeche+=values[counter]["Flaeche"]
        except:
            pass
        try:
            sy-=ausschnitte[counter]["Sy"]
            sz-=ausschnitte[counter]["Sz"]
            flaeche-=ausschnitte[counter]["Flaeche"]
        except:
            pass
flaeche_ausgabe = ausgabe(flaeche)
sy_ausgabe= ausgabe(sy)
sz_ausgabe = ausgabe(sz)

print ("Gesamtflaeche:\n",flaeche_ausgabe)
```

Auf die errechneten Summen wird nun die Funktionen “schwerpunkt“ angewandt. Als weiterer Schritt muss nun der Koordinatenursprung in den Schwerpunkt verschoben werden. Dafür ist die Funktion “transform“ zuständig, welche im Dictionary einen Eintrag mit den transformierten Koordinaten abspeichert. Mit diesen transformierten Punkten wird nun die Funktion “flachentraegheitsmoment“ aufgerufen. Wiederum werden alle Teilflächenträgheitsmomente aufsummiert, und das gesuchte Ergebnis zu erhalten. Jetzt können die Funktionen “haupttraegheitsmoment“, “drehunghauptachse“ und “findehauptachse“ ausgeführt werden, und die restlichen Ergebnisse zu erhalten.

```
werte = schwerpunkte(sy,sz,flaeche)
zs = werte[0]
ys = werte[1]
zs_ausgabe = ausgabe(zs)
ys_ausgabe = ausgabe(ys)
print("Schwerpunkt_ys", ys_ausgabe,"\n")
print("Schwerpunkt_zs:",zs_ausgabe,"\n")

for counter in range(1, anzahlstaebe + 1):
    stabwerte= values[counter]
    stabwerte = transform(stabwerte,zs,ys)
    stabwerte = flaechentraegheitsmoment(stabwerte)
    values[counter]=stabwerte
    try:
        stabwerteausschnitt = ausschnitte[counter]
        stabwerteausschnitt = transform(stabwerteausschnitt, zs, ys)
        stabwerteausschnitt \n
        = flaechentraegheitsmoment(stabwerteausschnitt)
        ausschnitte[counter] = stabwerteausschnitt
    except KeyError:
        pass

iy=0
iz=0
iyz=0
for counter in range(1,anzahlstaebe+1):
    try:
        iy+=values[counter]["Iy"]
        iz+=values[counter]["Iz"]
        iyz+=values[counter]["Iyz"]
    except:
```

```
            pass
        try:
            iy-=ausschnitte[counter]["Iy"]
            iz-=ausschnitte[counter]["Iz"]
            iyz-=ausschnitte[counter]["Iyz"]
        except:
            pass

iy_calc = calc_fast(iy)
iz_calc = calc_fast(iz)
iyz_calc = calc_fast(iyz)
iy_ausgabe = ausgabe(iy_calc)
iz_ausgabe = ausgabe(iz_calc)
iyz_ausgabe = ausgabe(iyz_calc)

print("\n", "Gesamtes Flaechentraegheitsmoment Iy:" ,(iy_ausgabe))
print("\n", "Gesamtes Flaechentraegheitsmoment Iz:" ,(iz_ausgabe))
print("\n", "Gesamtes Flaechentraegheitsmoment Iyz:" ,(iyz_ausgabe))

ih = haupttraegheitsmoment(iy,iz,iyz)
ih1_ausgabe = ih[2]
ih2_ausgabe = ih[3]

print("\n", "Haupttraegheitsmoment1:", (ih1_ausgabe))
print("Haupttraegheitsmoment2:", (ih2_ausgabe))


#Ausgabe des Drehwinkels
phi = drehunghauptachse(ih,iy,iz,iyz)
phiingrad = ((phi/(pi)*180))
phiingrad_ausgabe = ausgabe(phiingrad)

print("\n", "Drehwinkel:", (phiingrad_ausgabe))


#Ausgabe der Bezugsachse
global AchseDrehung
findehauptachse = bezugsachse(ih,iy,iz,iyz,phi)
if findehauptachse == "iyphi":
    print ("\n","Hauptachse 1 entspricht Iy um {}Âř gedreht" \n
    .format((phiingrad_ausgabe)))
    AchseDrehung = str("Iy")
elif findehauptachse == "izphi":
    print("Hauptachse 1 entspricht Iz um {}Âř gedreht" \n
    .format((phiingrad)))
    AchseDrehung = str("Iz")
else:
    print ("Error")

print ("Values \n",values)

print ("Ausschnitte \n",ausschnitte)
```

#### 3.2.1.2 Funktion: eckpunkte(dictionary)

Diese Funktion berechnet die Eckpunkte für die Eingabe “Koordinaten, Dicken“. Dabei wird im übergebenen Dictionary ein neuer Eintrag hinzugefügt.

Die Eingaben werden aus dem Dictionary übernommen und in lokalen Variablen abgespeichert. Wird für die Bereichsabschnitte eine andere Option als “Winkeleingabe“ gewählt, so wird der Winkel mit null abgespeichert - dies entspricht einem rechtwinkligen Abschnitt.

```
def eckpunkte(dictionary):
    print("Eckpunkte werden berechnet ...")
    geometrie = dictionary["basisgeometrie"]
    if geometrie == "Koordinaten, Dicken":
        #Uebergabe Knotenpunkt (anfang&ende)
        pstart = [dictionary["koordinatenyanfang"], dictionary \n
        ["koordinatenzanfang"]]
        pende = [dictionary["koordinatenyende"], dictionary \n
        ["koordinatenzende"]]

        #Uebergabe der Dicken fuer P1,P2,P3,P4
        dicke1 = dictionary["dicke"][0]
        dicke2 = dictionary["dicke"][1]
        dicke3 = dictionary["dicke"][2]
        dicke4 = dictionary["dicke"][3]

        #Winkel aus "Winkeleingabe"
        alpha = dictionary["winkel"][0]
        beta = dictionary["winkel"][1]

        deltaz = pende[1] - pstart[1]
        deltay = pende[0] - pstart[0]
```

Anschließend wird ein Begrenzungsvektor erstellt, welcher bereits um den entsprechenden Winkel gedreht wurde. Mathematisch entspricht dies einer Drehmatrix, welche mit dem Vektor multipliziert wird. Um eine Nulldivision zu vermeiden, muss der Fall “$deltay == 0$“ abgefangen werden.

```
        #Vektor fuer Begrenzungslinie aufstellen
        if deltay == 0:
            if vergleich(deltaz) >= 0:
                #Begrenzungsvektor rechtwinklig auf Achse
                vektoryrw = -1
                vektorzrw = 0
                #rechtwinkligen Begrenzungsvektor drehen
                vektory1alpha = (-1) * cos(alpha) - 0 * sin(alpha)
                vektorz1alpha = (-1) * sin(alpha) + 0 * cos(alpha)

                vektory2beta = (-1) * cos(beta) - 0 * sin(beta)
                vektorz2beta = (-1) * sin(beta) + 0 * cos(beta)

                normierung = sqrt(vektoryrw ** 2 + vektorzrw ** 2)
            else:
                vektoryrw = 1
                vektorzrw = 0
```

```
            vektory1alpha = (1) * cos(alpha) - 0 * sin(alpha)
            vektorz1alpha = (1) * sin(alpha) + 0 * cos(alpha)

            vektory2beta = (1) * cos(beta) - 0 * sin(beta)
            vektorz2beta = (1) * sin(beta) + 0 * cos(beta)

            normierung = sqrt(vektoryrw ** 2 + vektorzrw ** 2)

    else:
        steigungachse = (deltaz / deltay)
        vektoryrw = (-steigungachse)
        vektorzrw = (1)

        vektory1alpha=(-steigungachse)*cos(alpha)-1*sin(alpha)
        vektorz1alpha =(-steigungachse)*sin(alpha)+1*cos(alpha)

        vektory2beta =(-steigungachse)*cos(beta)-1*sin(beta)
        vektorz2beta =(-steigungachse)*sin(beta)+1*cos(beta)

        normierung = sqrt(vektoryrw**2 + vektorzrw ** 2)
```

Zwecks einer Vergleichsoperation werden zusätzlich die Eckpunkte für $\alpha = 0$ und $\beta = 0$ berechnet. Dies wird dafür benötigt, wenn eine Trägerkante eine unendliche Steigung aufweisen würde.

```
#Brechnung der Eckpunkte, wenn alpha und beta = 0
#...notwendig fuer Vergleiche
p_rw_links_oben_y =(pstart[0] + (vektoryrw/normierung) \n
* dicke1)
p_rw_links_oben_z =(pstart[1] + (vektorzrw/normierung) \n
* dicke1)

p_rw_rechts_oben_y = (pende[0] + (vektoryrw/normierung) \n
* dicke2)
p_rw_rechts_oben_z = (pende[1] + (vektorzrw/normierung) \n
* dicke2)

p_rw_links_unten_y =(pstart[0] - (vektoryrw/normierung) \n
* dicke3)
p_rw_links_unten_z =(pstart[1] - (vektorzrw/normierung) \n
* dicke3)

p_rw_rechts_unten_y = (pende[0] - (vektoryrw/normierung) \n
* dicke4)
p_rw_rechts_unten_z = (pende[1] - (vektorzrw/normierung) \n
* dicke4)
```

Nun kann die horizontale Länge des bereits gedrehten Begrenzungsvektors berechnet werden. Dies wird realisiert, indem der Schnittpunkt von zwei Geradengleichungen berechnet wird.

```
#Brechnung der horizontalen Laenge der bereits gedrehten
#Begrenzungslinie
if (p_rw_rechts_oben_y - p_rw_links_oben_y)==0:
```

```
        varp1_1 = solve(pstart[0] + vektory1alpha / normierung \n
        * x - p_rw_links_oben_y, x)
        dd_p1 = varp1_1[0]

        varp2_1 = solve(pende[0] + vektory2beta / normierung \n
        * x - p_rw_rechts_oben_y, x)
        dd_p2 = varp2_1[0]
    else:
        steigungf1y = 1
        steigungf1z = (p_rw_rechts_oben_z - p_rw_links_oben_z)/ \n
        (p_rw_rechts_oben_y - p_rw_links_oben_y)
        normierungf1 = sqrt(steigungf1y**2 + steigungf1z ** 2)

        varp1_1 = solve(pstart[0] + vektory1alpha / normierung \n
        * x - p_rw_links_oben_y - steigungf1y / normierungf1 * y,y)
        varp1_2 = solve(pstart[1] + vektorz1alpha / normierung \n
        * x - p_rw_links_oben_z - steigungf1z / normierungf1 \n
        * varp1_1[0], x)
        dd_p1 = varp1_2[0]

        varp2_1 = solve(pende[0] + vektory2beta / normierung \n
        * x - p_rw_rechts_oben_y - steigungf1y / normierungf1 * y,y)
        varp2_2 = solve(pende[1] + vektorz2beta / normierung \n
        * x - p_rw_rechts_oben_z - steigungf1z / normierungf1 \n
        * varp2_1[0], x)
        dd_p2 = varp2_2[0]



    if (p_rw_rechts_unten_y - p_rw_links_unten_y)==0:
        varp3_1 = solve(pstart[0] + vektory1alpha / normierung \n
        * x - p_rw_links_unten_y, x)
        dd_p3 = varp3_1[0]

        varp4_1 = solve(pende[0] + vektory2beta / normierung \n
        * x - p_rw_rechts_unten_y, x)
        dd_p4 = varp4_1[0]
    else:
        steigungf2y = 1
        steigungf2z = (p_rw_rechts_unten_z - p_rw_links_unten_z) \n
        / (p_rw_rechts_unten_y - p_rw_links_unten_y)
        normierungf2 = sqrt(steigungf2y ** 2 + steigungf2z ** 2)

        varp3_1 = solve(pstart[0] + vektory1alpha / normierung \n
        * x - p_rw_links_unten_y - steigungf2y / normierungf2 \n
        * y,y)
        varp3_2 = solve(pstart[1] + vektorz1alpha / normierung \n
        * x - p_rw_links_unten_z - steigungf2z / normierungf2 \n
        *varp3_1[0], x)
        dd_p3 = varp3_2[0]

        varp4_1 = solve(pende[0] + vektory2beta / normierung \n
        * x - p_rw_rechts_unten_y - steigungf2y / normierungf2 \n
        * y,y)
        varp4_2 = solve(pende[1] + vektorz2beta / normierung \n
        * x - p_rw_rechts_unten_z - steigungf2z / normierungf2 \n
        * varp4_1[0], x)
```

```
            dd_p4 = varp4_2[0]
```

Abschließend werden die Eckpunkte berechnet und in das Dictionary eingefügt.

```
        #Ermittlung der tatsaechlichen Eckpunkte
        p1y = (pstart[0] + vektory1alpha/normierung * dd_p1)
        p1z = (pstart[1] + vektorz1alpha/normierung * dd_p1)

        p2y = (pende[0] + vektory2beta / normierung * dd_p2)
        p2z = (pende[1] + vektorz2beta / normierung * dd_p2)

        p3y = (pstart[0] + vektory1alpha / normierung * dd_p3)
        p3z = (pstart[1] + vektorz1alpha / normierung * dd_p3)

        p4y = (pende[0] + vektory2beta / normierung * dd_p4)
        p4z = (pende[1] + vektorz2beta / normierung * dd_p4)

        randpunkte=[[p1y,p1z],[p2y,p2z],[p3y,p3z],[p4y,p4z]]

    else:
        print ("Error␣Geometrie")

    dictionary["eckpunkte"] = randpunkte
    return dictionary
```

#### 3.2.1.3 Funktion: eckpunktekoordinaten(dictionary)

Wurden die Eckpunkte koordinativ eingegeben, so wird diese Funktion angewendet. Dabei werden lediglich die Koordinaten der Eingabe in das Dictionary gespeichert.

```
def eckpunktekoordinaten(dictionary):
    #Uebernahme der Eingabe
    print("Eckpunkte werden berechnet ...")
    if dictionary["basisgeometrie"]== "Viereck" \n
    or dictionary["basisgeometrie"]== "Rechteck":
        koordinaten = dictionary["koordinatenviereck"]
        [p1y,p1z,p2y,p2z,p3y,p3z,p4y,p4z] = [koordinaten[0], \n
        koordinaten[1],koordinaten[2],koordinaten[3], \n
        koordinaten[4],koordinaten[5],koordinaten[6], \n
        koordinaten[7]]

    elif dictionary["basisgeometrie"] == "Dreieck":
        koordinaten = dictionary["koordinatendreieck"]
        [p1y, p1z, p2y, p2z, p3y, p3z, p4y, p4z] = [koordinaten[0], \n
        koordinaten[1], koordinaten[2],koordinaten[3], koordinaten[4],\n
        koordinaten[5],koordinaten[4], koordinaten[5]]

    else:
        pass

    dictionary["eckpunkte"] = [[p1y, p1z], [p2y, p2z], [p3y, p3z], \n
    [p4y, p4z]]
    return dictionary
```

#### 3.2.1.4 Funktion: eckpunkteausschnittdicken(dictionary)

Diese Funktion dient zur Auswertung der Eckpunkte, falls ein Ausschnitt mittels “Koordinaten, Dicken“ eingegeben wurde. Dabei wird eine simple Vektoraddition unter Berücksichtigung des linken bzw. rechten Abstands durchgeführt. Die Eckpunkte werden im Dictionary abgespeichert.

```
def eckpunkteausschnittdicken(dictionary):
    [al,ar,d1,d2,d3,d4] = dictionary["Ausschnittdicken"]
    koordinatenyanfang = dictionary["koordinatenyanfang"]
    koordinatenzanfang = dictionary["koordinatenzanfang"]
    koordinatenyende = dictionary["koordinatenyende"]
    koordinatenzende = dictionary["koordinatenzende"]

    deltay = koordinatenyende - koordinatenyanfang
    deltaz = koordinatenzende - koordinatenzanfang
    normierungdelta = sqrt(deltay**2 + deltaz**2)

    ky = -deltaz/normierungdelta
    kz = deltay/normierungdelta

    p1y = koordinatenyanfang + al * deltay/normierungdelta + \n
    d1 * ky
    p1z = koordinatenzanfang + al * deltaz/normierungdelta + \n
    d1 * kz

    p2y = koordinatenyende - ar * deltay/normierungdelta + \n
    d2 * ky
    p2z = koordinatenzende - ar * deltaz/normierungdelta + \n
    d2 * kz

    p3y = koordinatenyanfang + al * deltay/normierungdelta \n
    - d3 * ky
    p3z = koordinatenzanfang + al * deltaz/normierungdelta \n
    - d3 * kz

    p4y = koordinatenyende - ar * deltay / normierungdelta \n
    - d4 * ky
    p4z = koordinatenzende - ar * deltaz / normierungdelta \n
    - d4 * kz

    dictionary["eckpunkte"] = [[p1y, p1z], [p2y, p2z], \n
    [p3y, p3z], [p4y, p4z]]
    return dictionary
```

#### 3.2.1.5 Funktion: abschnittlinks(dictionary1, dictionary0)

Diese Funktion modifiziert den linken Abschnitt. Analog dazu gibt es eine Funktion für den rechten Abschnitt. Im Rahmen dieser Beschreibung wird diese nicht erneut beschrieben, jedoch kann die Methode im Anhang eingesehen werden.

Zuerst werden die entsprechenden Daten der Eingabe ausgewertet.

```
def abschnittlinks(dictionary1,dictionary0):
    randpunkte1 = dictionary1["eckpunkte"]
    randpunkte0 = dictionary0["eckpunkte"]
    abschnittlinks_str = dictionary1["abschnitt_links"]
    [[p1y1, p1z1], [p2y1, p2z1], [p3y1, p3z1], [p4y1, p4z1]] \n
    = randpunkte1
    [[p1y0, p1z0], [p2y0, p2z0], [p3y0, p3z0], [p4y0, p4z0]] \n
    = randpunkte0
```

Wurde für den Bereichsabschnitt “Schnittpunkt“ ausgewählt, so werden zwei Geradengleichungen erstellt. Dabei bezieht sich die erste Gleichung auf den vorherigen Abschnitt und die zweite beschreibt die Beschränkung des aktuellen Bereichs. Diese werden gleichgesetzt und der horizontale Abstand, bei welchem sich die beiden Geraden treffen, wird ermittelt. Dies wird jeweils für die obere und die untere Begrenzung durchgeführt.

```
    if abschnittlinks_str == "Schnittpunkt":
        k1yo = p2y1-p1y1
        k1zo = p2z1-p1z1
        k0yo = p2y0-p1y0
        k0zo = p2z0-p1z0

        k1yu = p4y1-p3y1
        k1zu = p4z1-p3z1
        k0yu = p4y0-p3y0
        k0zu = p4z0-p3z0

        if k0yo == 0:
            x1 = solve(p1y1 + k1yo * x - p2y0 - k0yo * y, x)
            x1 = x1[0]
        else:
            y1 = solve(p1y1+k1yo*x-p2y0-k0yo*y,y)
            y1 = y1[0]
            x1 = solve(p1z1+k1zo*x-p2z0-k0zo*y1,x)
            x1 = x1[0]

        if k0yu == 0:
            x2 = solve(p3y1 + k1yu * x - p4y0 - k0yu * y, x)
            x2 = x2[0]
        else:
            y2 = solve(p3y1 + k1yu * x - p4y0 - k0yu * y, y)
            y2 = y2[0]
            x2 = solve(p3z1 + k1zu * x - p4z0 - k0zu * y2, x)
            x2 = x2[0]


        p1y1 = (p1y1+k1yo*x1)
```

```
        p1z1 = (p1z1+k1zo*x1)
        p3y1 = (p3y1 + k1yu * x2)
        p3z1 = (p3z1 + k1zu * x2)

        randpunkte1 = [[p1y1, p1z1], [p2y1, p2z1], [p3y1, p3z1], \n
        [p4y1, p4z1]]
```

Für den Fall, dass die Auswahl “Winkelhalbierende“ getroffen wurde, werden die Steigungen des vorherigen und des aktuellen Abschnitts berechnet. Diese werden halbiert, aufsummiert und um neunzig Grad gedreht. Anschließend werden wieder beide Schnittpunkte des neuen Abschnitts mit den Trägerrandbereichen berechnet.

```
    elif abschnittlinks_str == "Winkelhalbierende":
        pstart1 = [dictionary1["koordinatenyanfang"], \n
        dictionary1["koordinatenzanfang"]]
        pende1 = [dictionary1["koordinatenyende"], \n
        dictionary1["koordinatenzende"]]
        pstart0 = [dictionary0["koordinatenyanfang"], \n
        dictionary0["koordinatenzanfang"]]
        pende0 = [dictionary0["koordinatenyende"], \n
        dictionary0["koordinatenzende"]]
        pstart1y = pstart1[0]
        pstart1z = pstart1[1]
        pende1y = pende1[0]
        pende1z = pende1[1]
        pstart0y = pstart0[0]
        pstart0z = pstart0[1]
        pende0y = pende0[0]
        pende0z = pende0[1]

        k1ay = pende1y-pstart1y
        k1az = pende1z-pstart1z
        k0ay = pende0y-pstart0y
        k0az = pende0z-pstart0z

        kay = (k1az+k0az)/2
        kaz = -(k1ay+k0ay)/2

        k1yo = p2y1 - p1y1
        k1zo = p2z1 - p1z1
        k1yu = p4y1 - p3y1
        k1zu = p4z1 - p3z1

        if kay ==0:
            x1 = solve(p1y1 + k1yo * x - pstart1y - kay * y, x)
            x1 = x1[0]
        else:
            y1 = solve(p1y1 + k1yo * x - pstart1y - kay *y, y)
            y1 = y1[0]
            x1 = solve(p1z1 + k1zo * x - pstart1z - kaz * y1, x)
            x1 = x1[0]

        if kay == 0:
            x2 = solve(p3y1 + k1yu * x - pstart1y - kay * y, x)
            x2 = x2[0]
        else:
```

```
            y2 = solve(p3y1 + k1yu * x - pstart1y - kay * y, y)
            y2 = y2[0]
            x2 = solve(p3z1 + k1zu * x - pstart1z - kaz * y2, x)
            x2 = x2[0]


        p1y1 = (p1y1+k1yo*x1)
        p1z1 = (p1z1+k1zo*x1)
        p3y1 = (p3y1 + k1yu * x2)
        p3z1 = (p3z1 + k1zu * x2)

        randpunkte1 = [[p1y1, p1z1], [p2y1, p2z1], [p3y1, p3z1], \n
        [p4y1, p4z1]]
```

Die Auswahl “direkter Anschluss“ bewirkt, dass die entsprechenden Eckpunkte des vorherigen Bereiches übernommen werden.

```
    elif abschnittlinks_str == "direkter Anschluss":
        randpunkte1 = [[p2y0, p2z0], [p2y1, p2z1], [p4y0, p4z0], \n
        [p4y1, p4z1]]

    else:
        print ("error in geometrie_left")

    dictionary1["eckpunkte"] = randpunkte1
    return dictionary1
```

#### 3.2.1.6 Funktion: vergleich(var)

Diese Funktion berechnet einen Referenzwert für eine symbolischen Variable, welcher für Vergleichsoperationen benötigt wird.

```
def vergleich(var):
    var = str(var)
    var1 = var.replace("b", "1")
    var1 = var1.replace("A1s", "Abs")
    var1 = var1.replace("pi", "3.14159265359")
    var2 = var1.replace("h", "1.1")
    var3 = var2.replace("d", "0.1")
    var4 = var3.replace("w", "0.1")
    var5 = eval(var4)
    return var5
```

#### 3.2.1.7 Funktion: StatMoment(dictionary)

Diese Funktion berechnet das Statische Moment und die Fläche eines Bereichs.

Ein Kreis kann einfach in die entsprechenden Formeln eingesetzt werden.

```
def StatMoment(dictionary):
    print("Flaeche und Statisches Moment wird berechnet ...")
    if dictionary["basisgeometrie"] == "Kreis":
        [ym,zm,radius]=dictionary["koordinatenkreis"]
        A = radius**2 * pi
        Sy = zm * A
        Sz = ym * A
```

Für den Bogen müssen zuerst die geometrischen Werte berechnet werden. Dabei muss unterschieden werden, ob die Eingabe mittels Radius oder Winkel übergeben wurde. Bei der Eingabe eines Radius wird ein virtuelles Dreieck in den Bogen gesetzt, um den Winkel auszurechnen. Mittels dem Winkel und dem Radius kann auf den Mittelpunkt gerechnet werden. Der Winkel muss nun noch auf das Koordinatensystem bezogen werden, d.h. der Quadrant muss berücksichtigt werden.

```
        if dictionary["Eingabe"] == "Radius":
            [p1y,p1z,p2y,p2z,dicke1,dicke2,radius,k] = \n
            dictionary["wertekreisbogen"]
            deltaz = p2z - p1z
            deltay = p2y - p1y
            diagonal = sqrt(deltaz ** 2 + deltay ** 2)

            winkel = asin(diagonal/(2*radius))*2
            m = diagonal / (2 * tan(winkel/2))

            if str(k) == "MP_rechts_von_P1P2":
                m = -m
            elif str(k) == "MP_links_von_P1P2":
                pass
            else:
                print ("Error")

            mpy = p1y + (deltay/diagonal) * (diagonal/2) \n
            + (-deltaz/diagonal) * m
            mpz = p1z + (deltaz/diagonal) * (diagonal/2) \n
            + (deltay/diagonal) * m

            deltamp_p1_z = vergleich(p1z - mpz)
            deltamp_p1_y = vergleich(p1y - mpy)
            deltamp_p2_z = vergleich(p2z - mpz)
            deltamp_p2_y = vergleich(p2y - mpy)

            if deltamp_p1_z >=0 and deltamp_p1_y >= 0:
                quadrant1 = 0
            elif deltamp_p1_z >=0 and deltamp_p1_y <= 0:
                quadrant1 = pi
            elif deltamp_p1_z <=0 and deltamp_p1_y <= 0:
                quadrant1 = pi
            elif deltamp_p1_z <=0 and deltamp_p1_y >= 0:
                quadrant1 = 2*pi
```

```
else:
    print ("Error in Quadrant")

if deltamp_p2_z >=0 and deltamp_p2_y >= 0:
    quadrant2 = 0
elif deltamp_p2_z >=0 and deltamp_p2_y <= 0:
    quadrant2 = pi
elif deltamp_p2_z <=0 and deltamp_p2_y <= 0:
    quadrant2 = pi
elif deltamp_p2_z <=0 and deltamp_p2_y >= 0:
    quadrant2 = 2*pi
else:
    print ("Error in Quadrant")


if abs(vergleich(p1y - mpy)) <= 0.0001 \n
or abs(vergleich(p1z - mpz)) <= 0.0001:
    if abs(vergleich(p1y-mpy))<= 0.0001:
        if vergleich(p1z) >= vergleich(mpz):
            winkel1 = pi/2
        else:
            winkel1 = 3*pi/2
    elif abs(vergleich(p1z-mpz)) <= 0.0001:
        if vergleich(p1y) >= vergleich(mpy):
            winkel1 = 0
        else:
            winkel1 = pi
    else:
        winkel1 = atan((p1z - mpz) / (p1y - mpy)) \n
        + quadrant1


else:
    winkel1 = atan((p1z - mpz) / (p1y - mpy))+ quadrant1


if abs(vergleich(p2y - mpy)) <= 0.0001 \n
or abs(vergleich(p2z - mpz)) <= 0.0001:
    if abs(vergleich(p2y-mpy))<= 0.0001:
        if vergleich(p2z) >= vergleich(mpz):
            winkel2 = pi/2
        else:
            winkel2 = 3*pi/2
    elif abs(vergleich(p2z-mpz)) <= 0.0001:
        if vergleich(p2y) >= vergleich(mpy):
            winkel2 = 0
        else:
            winkel2 = pi
    else:
        winkel2 = atan((p2z - mpz) / (p2y - mpy)) \n
        + quadrant2

else:
    winkel2 = atan((p2z - mpz) / (p2y - mpy))+ quadrant2

winkel_temp = [winkel1,winkel2]
```

```
winkel1 = winkel_temp[0]
winkel2 = winkel_temp[1]

if vergleich(winkel2)>vergleich(winkel1):
    winkel1 += 2*pi
```

Wird der Bogen über einen Winkel eingegeben, so wird wieder ein virtuelles Dreieck in den Bogen gesetzt und mithilfe des halben Winkels der Radius berechnet. Nun kann der Mittelpunkt und bezogene Winkel gleich wie bei der Eingabe mit dem Radius berechnet werden.

```
elif dictionary["Eingabe"] == "Winkel":
    [p1y, p1z, p2y, p2z,dicke1,dicke2,winkel,k] \n
    = dictionary["wertekreisbogen"]

    #berechne Mittelpunkt
    deltaz = p2z-p1z
    deltay = p2y-p1y
    diagonal = sqrt(deltaz**2+deltay**2)
    radius = diagonal/(2*sin(winkel/2))

    m = diagonal/(2*tan(winkel/2))

    if str(k) == "MP_rechts_von_P1P2":
        m = -m
    elif str(k) == "MP_links_von_P1P2":
        pass
    else:
        print("Error")

    mpy = p1y + (deltay/diagonal) * (diagonal/2) \n
    + (-deltaz/diagonal) * m
    mpz = p1z + (deltaz/diagonal) * (diagonal/2) \n
    + (deltay/diagonal) * m

    deltamp_p1_z = vergleich(p1z - mpz)
    deltamp_p1_y = vergleich(p1y - mpy)
    deltamp_p2_z = vergleich(p2z - mpz)
    deltamp_p2_y = vergleich(p2y - mpy)

    if deltamp_p1_z >= 0 and deltamp_p1_y >= 0:
        quadrant1 = 0
    elif deltamp_p1_z >= 0 and deltamp_p1_y <= 0:
        quadrant1 = pi
    elif deltamp_p1_z <= 0 and deltamp_p1_y <= 0:
        quadrant1 = pi
    elif deltamp_p1_z <= 0 and deltamp_p1_y >= 0:
        quadrant1 = 2 * pi
    else:
        print("Error in Quadrant")

    if deltamp_p2_z >= 0 and deltamp_p2_y >= 0:
        quadrant2 = 0
    elif deltamp_p2_z >= 0 and deltamp_p2_y <= 0:
        quadrant2 = pi
    elif deltamp_p2_z <= 0 and deltamp_p2_y <= 0:
        quadrant2 = pi
    elif deltamp_p2_z <= 0 and deltamp_p2_y >= 0:
```

```
    quadrant2 = 2 * pi
else:
    print("Error in Quadrant")

if abs(vergleich(p1y - mpy)) <= 0.0001 \n
or abs(vergleich(p1z - mpz)) <= 0.0001:
    if abs(vergleich(p1y-mpy))<= 0.0001:
        if vergleich(p1z) >= vergleich(mpz):
            winkel1 = pi/2
        else:
            winkel1 = 3*pi/2
    elif abs(vergleich(p1z-mpz)) <= 0.0001:
        if vergleich(p1y) >= vergleich(mpy):
            winkel1 = 0
        else:
            winkel1 = pi
    else:
        winkel1 = atan((p1z - mpz) / (p1y - mpy)) \n
        + quadrant1

else:
    winkel1 = atan((p1z - mpz) / (p1y - mpy))+ quadrant1


if abs(vergleich(p2y - mpy)) <= 0.0001 \n
or abs(vergleich(p2z - mpz)) <= 0.0001:
    if abs(vergleich(p2y-mpy))<= 0.0001:
        if vergleich(p2z) >= vergleich(mpz):
            winkel2 = pi/2
        else:
            winkel2 = 3*pi/2
    elif abs(vergleich(p2z-mpz)) <= 0.0001:
        if vergleich(p2y) >= vergleich(mpy):
            winkel2 = 0
        else:
            winkel2 = pi
    else:
        winkel2 = atan((p2z - mpz) / (p2y - mpy)) \n
        + quadrant2

else:
    winkel2 = atan((p2z - mpz) / (p2y - mpy))+ quadrant2

winkel_temp = [winkel1, winkel2]
winkel1 = winkel_temp[0]
winkel2 = winkel_temp[1]

if vergleich(winkel2)>vergleich(winkel1):
    winkel1 += 2*pi
```

Beim Bogen wurde das Integral

$$A = \int_{\phi=winkel1}^{winkel2} \int_{r=radius1}^{radius2} r\,dr d\phi,$$

$$S_y = \int_{\phi=winkel1}^{winkel2} \int_{r=radius1}^{radius2} r(r\sin(\phi)+mpz)\,dr d\phi,$$

$$S_z = \int_{\phi=winkel1}^{winkel2} \int_{r=radius1}^{radius2} r(r\cos(\phi)+mpy)\,dr d\phi$$

berechnet. Nun müssen die Werte nur noch eingesetzt werden.

```
            radius1 = radius + dicke1
            radius2 = radius - dicke2
            dicke = dicke1 + dicke2

            #integrate((r*cos(w)+mp)*r,(r,r1,r2),(w,w1,w2))

            A = winkel1*(radius1**2/2 - radius2**2/2) \n
            - winkel2*(radius1**2/2 - radius2**2/2)

            Sy=mpz*radius1**2*winkel1/2 - mpz*radius1**2*winkel2/2 \n
            - mpz*radius2**2*winkel1/2 + mpz*radius2**2*winkel2/2 \n
            - radius1**3*cos(winkel1)/3 + radius1**3*cos(winkel2)/3 \n
            + radius2**3*cos(winkel1)/3 - radius2**3*cos(winkel2)/3

            Sz=mpy*radius1**2*winkel1/2 - mpy*radius1**2*winkel2/2 \n
            - mpy*radius2**2*winkel1/2 + mpy*radius2**2*winkel2/2 \n
            + radius1**3*sin(winkel1)/3 - radius1**3*sin(winkel2)/3 \n
            - radius2**3*sin(winkel1)/3 + radius2**3*sin(winkel2)/3

        else:
            print ("Error")

        dictionary["wertekreisbogen2"]=[mpy,mpz,radius1,radius2, \n
        winkel1,winkel2]
        dictionary["auswertungkreisbogen"] = [mpy, mpz, p1y, p1z, \n
        p2y, p2z, winkel1, winkel2, radius1, radius2]
```

Wurde kein Bogen oder Kreis gewählt, so stehen vier Koordinaten zur Verfügung. Diese Koordinaten werden wie eine Affinkombination aufgefasst. Dafür wird eine Jakobi-Determinante berechnet und über die Koeffizienten integriert. Dabei muss beachtet werden, dass die Reihenfolge der Koordinaten nicht beliebig ist. Folglich wird nur eine positive Determinante zugelassen.

```
    else:
        p1 = dictionary["eckpunkte"][0]
        p2 = dictionary["eckpunkte"][1]
        p3 = dictionary["eckpunkte"][2]
        p4 = dictionary["eckpunkte"][3]

        y = p1[0] * (1 - n) * (1 - l) + p3[0] * (1 - l) * n + p2[0] \n
        * (1 - n) * l + p4[0] * n * l
        z = p1[1] * (1 - n) * (1 - l) + p3[1] * (1 - l) * n + p2[1] \n
        * (1 - n) * l + p4[1] * n * l
        ye = y.diff(l, 1)
        yn = y.diff(n, 1)
        ze = z.diff(l, 1)
```

```
        zn = z.diff(n, 1)
        y = simplify(y)
        z = simplify(z)
        det = simplify(ye * zn - yn * ze)
        A = integrate((det), (l, 0, 1), (n, 0, 1))
        if vergleich(A) <= 0:
            det = - det
            A = integrate((det), (l, 0, 1), (n, 0, 1))
        else:
            pass
        Sy = integrate(z * (det), (l, 0, 1), (n, 0, 1))
        Sz = integrate(y * (det), (l, 0, 1), (n, 0, 1))


    Sy = calc_fast(Sy)
    Sy_ausgabe = ausgabe(Sy)
    Sz = calc_fast(Sz)
    Sz_ausgabe = ausgabe(Sz)
    A = calc_fast(A)
    A_ausgabe = ausgabe(A)

    dictionary["Flaeche"]=A
    dictionary["Sy"] = Sy
    dictionary["Sz"] = Sz
    dictionary["Sy_ausgabe"]= Sy_ausgabe
    dictionary["Sz_ausgabe"] = Sz_ausgabe
    dictionary["A_ausgabe"] = A_ausgabe
    print("Teilflaeche:", A)
    print("Statisches␣Moment_Teilergebnis:", "\n", "Sy:", Sy, \n
    "\n", "Sz:", Sz)
    return dictionary
```

#### 3.2.1.8 Funktion: schwerpunkte(sy, sz, A)

Der Schwerpunkt wird mittels den Standardformeln berechnet.

```
def schwerpunkte(sy,sz,A):
    print("Schwerpunkte␣werden␣berechnet␣...")
    zs = calc_fast(sy/A)
    zs_ausgabe = ausgabe(zs)
    ys = calc_fast(sz/A)
    ys_ausgabe = ausgabe(ys)
    return [zs,ys,zs_ausgabe,ys_ausgabe]
```

#### 3.2.1.9 Funktion: transform(dictionary, zs, ys)

Diese Funktion setzt den Koordinatenursprung in den Schwerpunkt. Dabei werden einfach die Punkte um den Schwerpunkt verschoben.

```
def transform(dictionary, zs, ys):
    print("Koordinatentransformation␣wird␣durchgefuehrt␣...")
    if dictionary["basisgeometrie"]=="Kreis":
        [ym, zm, radius] = dictionary["koordinatenkreis"]
        pt = [ym-ys,zm-zs,radius]
    elif dictionary["basisgeometrie"] == "Kreisbogen":
        [mpy,mpz,radius1,radius2,winkel1,winkel2] = \n
        dictionary["wertekreisbogen2"]
        pt = [mpy-ys,mpz-zs,radius1,radius2,winkel1,winkel2]
    else:
        p = dictionary["eckpunkte"]
        pt = []
        for c in p:
            fkt = lambda x, y: [(x - ys), (y - zs)]
            transf = fkt((c[0]), (c[1]))
            pt = pt + [transf]
    dictionary["transformiertepunkte"]=pt
    return dictionary
```

#### 3.2.1.10 Funktion: flaechentraegheitsmoment(dictionary)

Bei dieser Funktion bedarf es einer Fallunterscheidung in Anhängigkeit der Geometrie.

Im Falle eines Kreises werden die Werte in die vorberechneten Formeln eingesetzt, welche sich zu

$$I_y = \int_{\phi=0}^{2\pi} \int_{r=0}^{radius} r(r\sin(\phi)+zt)^2\, drd\phi,$$

$$I_z = \int_{\phi=0}^{2\pi} \int_{r=0}^{radius} r(r\cos(\phi)+yt)^2\, drd\phi,$$

$$I_{yz} = \int_{\phi=0}^{2\pi} \int_{r=0}^{radius} r(r\cos(\phi)+yt)(r\sin(\phi)+zt)\, drd\phi$$

ergeben.

```
def flaechentraegheitsmoment(dictionary):
    print("Flaechentraegheitsmoment␣wird␣brechnet␣...")
    if dictionary["basisgeometrie"]=="Kreis":
        [yt,zt,radius] = dictionary["transformiertepunkte"]
        iy =radius**4*pi/4+radius**2 * (zt)**2 * pi
        iz =radius**4*pi/4+radius**2 * (yt)**2 * pi
        iyz = radius**2 * yt * zt * pi
    elif dictionary["basisgeometrie"] == "Kreisbogen":
```

Für den Kreisbogen gilt:

$$I_y = \int_{\phi=winkel1}^{winkel2} \int_{r=radius1}^{radius2} r(\sin(\phi)+zt)^2\, drd\phi,$$

$$I_z = \int_{\phi=winkel1}^{winkel2} \int_{r=radius1}^{radius2} r(\cos(\phi)+yt)^2\, drd\phi,$$

$$I_{yz} = \int_{\phi=winkel1}^{winkel2} \int_{r=radius1}^{radius2} r(\cos(\phi)+yt)(\sin(\phi)+zt)\, drd\phi$$

```
    elif dictionary["basisgeometrie"] == "Kreisbogen":
        [mty, mtz, radius1,radius2,winkel1,winkel2] = \n
        dictionary["transformiertepunkte"]
        iy = mtz**2*radius1**2*winkel1/2 \n
        - mtz**2*radius1**2*winkel2/2 - mtz**2*radius2**2*winkel1/2 \n
        + mtz**2*radius2**2*winkel2/2 - 2 * mtz*radius1** 3 \n
        *cos(winkel1)/3 + 2*mtz*radius1**3*cos(winkel2)/3 + \n
        2*mtz*radius2**3*cos(winkel1)/3 - 2*mtz*radius2**3 \n
        *cos(winkel2)/3 + radius1**4*(winkel1/2 - sin(winkel1) \n
        *cos(winkel1)/2)/4 - radius1**4*(winkel2/2 - sin(winkel2) \n
        *cos(winkel2)/2)/4 - radius2**4*(winkel1/2 - sin(winkel1) \n
        *cos(winkel1)/2)/4 + radius2**4*(winkel2/2 - sin(winkel2) \n
        *cos(winkel2)/2)/4

        iz = mty**2*radius1**2*winkel1/2 - mty**2*radius1**2*winkel2/2 \n
        - mty**2*radius2**2*winkel1/2 + mty**2*radius2**2*winkel2/2 \n
        + 2*mty*radius1**3*sin(winkel1)/3 - 2*mty*radius1**3 \n
        *sin(winkel2)/3 - 2*mty*radius2**3*sin(winkel1)/3 + 2*mty \n
        *radius2**3*sin(winkel2)/3 + radius1**4*(winkel1/2 \n
        + sin(winkel1)*cos(winkel1)/2)/4 - radius1**4*(winkel2/2 \n
```

```
    + sin(winkel2)*cos(winkel2)/2)/4 - radius2**4*(winkel1/2 \n
    + sin(winkel1)*cos(winkel1)/2)/4 + radius2**4*(winkel2/2 \n
    + sin(winkel2)*cos(winkel2)/2)/4

    iyz = mty*mtz*radius1**2*winkel1/2 - mty*mtz*radius1**2 \n
    *winkel2/2 - mty*mtz*radius2**2*winkel1/2 + mty*mtz \n
    *radius2**2*winkel2/2 + radius1**4*sin(winkel1)**2/8 \n
    - radius1**4*sin(winkel2)**2/8 + radius1**3*(-mty \n
    *cos(winkel1)/3 + mtz*sin(winkel1)/3) - radius1**3 \n
    *(-mty*cos(winkel2)/3 + mtz*sin(winkel2)/3) - radius2**4 \n
    *sin(winkel1)**2/8 + radius2**4*sin(winkel2)**2/8 \n
    - radius2**3*(-mty*cos(winkel1)/3 + mtz*sin(winkel1)/3) \n
    + radius2**3*(-mty*cos(winkel2)/3 + mtz*sin(winkel2)/3)
```

Andernfalls (analoge Vorgehensweise wie bei der Berechnung des statischen Moments):

```
else:
    p = dictionary["transformiertepunkte"]
    p1 = p[0]
    p2 = p[1]
    p3 = p[2]
    p4 = p[3]

    y = p1[0] * (1 - n) * (1 - l) + p3[0] * (1 - l) * n+ p2[0] \n
    * (1 - n) * l + p4[0] * n * l
    z = p1[1] * (1 - n) * (1 - l) + p3[1] * (1 - l) * n+ p2[1] \n
    * (1 - n) * l + p4[1] * n * l
    ye = y.diff(l, 1)
    yn = y.diff(n, 1)
    ze = z.diff(l, 1)
    zn = z.diff(n, 1)
    y = simplify(y)
    z = simplify(z)
    det = simplify(ye * zn - yn * ze)
    A = integrate((det), (l, 0, 1), (n, 0, 1))
    if vergleich(A) <= 0:
        det = - det
    else:
        pass
    iy = integrate(z ** 2 * det, (l, 0, 1),(n, 0, 1))
    iz = integrate(y ** 2 * det, (l, 0, 1), (n, 0, 1))
    iyz = integrate(y * z * det, (l, 0, 1), (n, 0, 1))

iy_calc = calc_fast(iy)
iz_calc = calc_fast(iz)
iyz_calc = calc_fast(iyz)

iy_ausgabe = ausgabe(iy_calc)
iz_ausgabe = ausgabe(iz_calc)
iyz_ausgabe = ausgabe(iyz_calc)

dictionary["Iy"]=iy_calc
dictionary["Iz"] = iz_calc
dictionary["Iyz"] = iyz_calc
dictionary["Iy_ausgabe"] = iy_ausgabe
dictionary["Iz_ausgabe"] = iz_ausgabe
dictionary["Iyz_ausgabe"] = iyz_ausgabe
```

```
print("Flaechentraegheitsmoment_Teilergebnis:", "\n", "Iy:", \n
iy_ausgabe, "\n", "Iz:", iz_ausgabe, "\n", "Iyz:", iyz_ausgabe)
```

#### 3.2.1.11 Funktion: haupttraegheitsmoment(iy, iz, iyz)

Die Hauptträgheitsmomente werden entsprechend der Standardformel berechnet.

```
def haupttraegheitsmoment(iy, iz, iyz):
    print("Haupttraegheitsmoment␣wird␣brechnet␣...")
    ih1 = ((iy + iz) / 2 + sqrt(((iy - iz) / 2) ** 2 + iyz ** 2))
    ih2 = ((iy + iz) / 2 - sqrt(((iy - iz) / 2) ** 2 + iyz ** 2))
    ih1_ausgabe = ausgabe(ih1)
    ih2_ausgabe = ausgabe(ih2)
    ih = [ih1, ih2, ih1_ausgabe, ih2_ausgabe]
    return ih
```

#### 3.2.1.12 Funktion: drehunghauptachse (ih, iy, iz, iyz)

Der Drehwinkel wird entsprechend der Standardformel berechnet. Zusätzlich werden die Spezialfälle abgefangen.

```
def drehunghauptachse(ih, iy, iz, iyz):
    print("Drehwinkel␣wird␣brechnet␣...")
    if abs(vergleich(iy - iz)) <= 0.001:
        if abs(vergleich(iyz)) <= 0.001:
            phi = 0
        else:
            phi = pi / 4
    else:
        phi = (atan(-2 * iyz / (iy - iz))) / 2
    return (phi)
```

#### 3.2.1.13 Funktion: bezugsachse(ih, iy, iz, iyz, phi)

Diese Funktion dreht die $I_y$ und $I_z$ um den bereits ermittelten Drehwinkel und vergleicht diese Werte mit dem Hauptträgheitsmoment. So wird die Achse bestimmt, welche gedreht werden muss, um das maximale Flächenträgheitsmoment zu erhalten.

```
def bezugsachse(ih, iy, iz, iyz, phi):
    print("Hauptachse␣wird␣ermittelt␣...")
    # Drehe Iy und Iy um phi
    iyphi = (1 / 2) * (iy + iz) + (1 / 2) * (iy - iz) * cos(2 * phi) \n
    - iyz * sin(2 * phi)
    izphi = (1 / 2) * (iy + iz) - (1 / 2) * (iy - iz) * cos(2 * phi) \n
    + iyz * sin(2 * phi)

    deltaiyphi = float(vergleich(iyphi - ih[0]))
    deltaizphi = float(vergleich(izphi - ih[0]))

    # Ordne die Bezugsachse zu
    if abs(deltaiyphi) <= 0.001:
        hauptachse = "iyphi"
    elif abs(deltaizphi) <= 0.001:
        hauptachse = "izphi"
    else:
        hauptachse = "Fehler␣in␣Funkt.␣Bezugsachse"

    return hauptachse
```

#### 3.2.1.14 Funktion: ausgabe(value)

Diese Funktion bereitet einen Wert für die Ausgabe auf. Dabei wird, wenn “ohne Dicken“ ausgewählt wurde, der Ausdruck expandiert und alle Ausdrücke, in welchen die Dicke in einer höheren Ordnung vorkommt, durch Null ersetzt. Sollte in dem Ausdruck eine Wurzel vorkommen, so muss der Ausdruck zuerst numerisch ausgewertet werden, da man sonst wahrscheinlich relevante Terme vernachlässigen würde.

```
def ausgabe (value):
    if ohne_dicke == "ja":
        value1 = expand(value)
        value2 = str(value1)
        if "sqrt" not in value2:
            value3 = value2.replace("d**","0*")
            value = eval(value3)
        else:
            value2 = eval(value2)
            value3 = simplify(value2).evalf()
            value3 = str(value3)
            value3 = value3.replace("d**","0*")
            value = eval(value3)
    else:
        pass
```

Hinsichtlich der maximal möglichen Vereinfachung wird der Ausdruck numerisch ausgewertet.

```
    if vereinfachung == "ja":
        value = str(value)
        value = replace_e(value)
        value = eval(value)
        value = simplify(value).evalf(3)
        value = simplify(value)
    elif vereinfachung == "nein":
        value = simplify(value)
    else:
        pass
```

Für den Spezialfall, dass es notwendig wird, Wurzeln numerisch zu Vereinfachen (unter der Annahme eines positiven Wertes), wandelt Python die Potenz “x“ zu “x.0“ um. Um den Wert einheitlich zusammenfassen zu können, muss dieser Zusatz wieder entfernt werden.

```
    value = str(value)
    for c in range (1,10):
        old = "**" + str(c) + ".0"
        new = "**" + str(c)
        value = value.replace(old,new)
    value = eval(value)

    return value
```

#### 3.2.1.15 Funktion: calc_fast(value)

Zur Beschleunigung des Ablaufs, werden mittels dieser Funktion extrem kleine Restwerte (Fehlerterme) entfernt. Zudem wird der Ausdruck vereinfacht.

```
def calc_fast(value):
    value = str(value)
    value = replace_e(value)
    value = eval(value)
    if vereinfachung == "ja":
        value = eval(str(value))
        value2 = simplify(value).evalf()
    elif vereinfachung == "nein":
        value2 = simplify(eval(str(value)))
    else:
        print ("Error in calc fast")

    return value2
```

#### 3.2.1.16 Funktion: unique(old_list)

Diese Funktion entfernt doppelte Werte aus einer Liste.

```
def unique(old_list):
    new_list = []
    [new_list.append(i) for i in old_list if not new_list.count(i)]
    return new_list
```

#### 3.2.1.17 Funktion: replace_e (value)

Diese Funktion entfernt extrem kleine Terme. Jedoch wird der Term nicht entfernt, wenn er sich im Nenner befindet. Dies dient als zusätzlicher Sicherheitsmechanismus.

```
def replace_e (value):
    mylist = []
    for c in value:
        mylist.append(c)
    counter = 0
    klammer = 0
    length = len(value)
    for c in range (0,length):
        if  mylist[c] == "/":
            counter += 2
        if counter >= 1:
            if  mylist[c]=="(":
                klammer += 1
            elif mylist[c]==")":
                klammer -= 1
            if klammer == 0:
                counter -= 1
        if klammer==0 and counter == 0:
            if mylist[c] == "e" and mylist[c+1] == "-":
                mylist[c-1]="*"
                mylist[c]="0"
                mylist[c+1]="*"
    value = "".join(mylist)
    return value
```

### 3.2.2 Klasse: Berechne(MainWindow)

#### 3.2.2.1 Ablauf

Wird eine Eingabe im numerischen Modus getätigt, so wird diese Klasse ausgeführt. Alle Funktionen und die Berechnungsabläufe dieser Klasse sind ident mit jenen der Klasse “ Berechne-Symbolic “, mit Ausnahme der Methoden zur Vereinfachung. Hierbei wird einfach aus dem Eingangswert ein float() erzeugt. Eine Trennung der symbolischen und numerischen Berechnung wurde zwecks der Reduzierung der Fehleranfälligkeit gewählt.

## 3.3 Ausgabe

Die Ausgabe setzt sich aus zwei Ausgabemöglichkeiten zusammen. Einerseits wird nur eine Skizze via tikz erstellt und andererseits werden die Ergebnisse in Kombination mit der Skizze ausgegeben.

Hierzu gibt es drei Klassen:

- class BerechneSymbolic(MainWindow)/class Berechne(MainWindow)
- class Skizze (MainWindow):
- class Ausgabe (MainWindow):

### 3.3.1 Klasse: BerechneSymbolic(MainWindow)/Berechne(MainWindow)

In dieser Klasse werden die Eckpunkte mittels eines Skalierungsfaktor auf eine entsprechende Größe angepasst. Zudem werden die erforderlichen Daten für die Bemaßung berechnet.

#### 3.3.1.1 Funktion: auswertungrandpunktskizze(values, ausschnitte)

Diese Funktion berechnet einen Skalierungsfaktor und wendet diesen auf die ursprüngliche Geometrie an.
Zuerst werden die Werte in Abhängigkeit der Basisgeometrie abgespeichert.

```
def auswertungrandpunkteskizze(values,ausschnitte):
    kreis = []
    kreisbogen = []
    viereck = []
    dreieck = []

    kreisausschnitt = []
    kreisbogenausschnitt = []
    viereckausschnitt = []
    dreieckausschnitt = []



    for c in values:
        stabwerte = values[c]
        geometrie = stabwerte["basisgeometrie"]
        if geometrie=="Kreis":
            [ym, zm, radius] = stabwerte["koordinatenkreis"]
            [ym, zm, radius] = [float(vergleich(ym)), \n
            float(vergleich(zm)),float(vergleich(radius))]
            kreis = kreis + [[-ym, -zm, radius]]
        elif geometrie=="Kreisbogen":
            [mpy, mpz, p1y, p1z, p2y, p2z, winkel1, winkel2, \n
            radius1,radius2] = stabwerte["auswertungkreisbogen"]
            [mpy, mpz, p1y, p1z, p2y, p2z, winkel1, winkel2, \n
            radius1, radius2]= [float(vergleich(mpy)), \n
            float(vergleich(mpz)), float(vergleich(p1y)), \n
            float(vergleich(p1z)), float(vergleich(p2y)), \n
```

```
        float(vergleich(p2z)), float(vergleich(winkel1)), \n
        float(vergleich(winkel2)), float(vergleich(radius1)), \n
        float(vergleich(radius2))]
        kreisbogen = kreisbogen + [[[[-mpy, -mpz],[-mpy \n
        - cos(winkel1) * radius1 , -mpz - sin(winkel1) * radius1] \n
        ,[-mpy - cos(winkel2) * radius1 , -mpz - sin(winkel2) \n
        * radius1],[-float(180-(winkel1 / pi * 180)), \n
        -float(180-(winkel2 / pi * 180))],[radius1]], \n
        [[-mpy, -mpz],[-mpy - cos(winkel1) * radius2 , -mpz \n
        - sin(winkel1) * radius2],[-mpy - cos(winkel2) * radius2 , \n
        -mpz - sin(winkel2) * radius2],[-float(180-(winkel1 / pi \n
        * 180)),-float(180-(winkel2 / pi * 180))],[radius2]]]]

    elif geometrie == "Koordinaten␣Dicken" or geometrie \n
    == "Viereck":
        [[p1y, p1z], [p2y, p2z], [p3y, p3z], [p4y, p4z]] \n
        = stabwerte["eckpunkte"]
        [[p1y, p1z], [p2y, p2z], [p3y, p3z], [p4y, p4z]] \n
        = [[float(vergleich(p1y)), float(vergleich(p1z))], \n
        [float(vergleich(p2y)), float(vergleich(p2z))], \n
        [float(vergleich(p3y)), float(vergleich(p3z))], \n
        [float(vergleich(p4y)), float(vergleich(p4z))]]

        viereck = viereck + [[[-p1y, -p1z], [-p2y, -p2z], \n
        [-p4y, -p4z],[-p3y, -p3z]]]
    elif geometrie == "Dreieck":
        [[p1y, p1z], [p2y, p2z], [p3y, p3z], [p4y, p4z]] \n
        = stabwerte["eckpunkte"]
        [[p1y, p1z], [p2y, p2z], [p3y, p3z]] \n
        = [[float(vergleich(p1y)), float(vergleich(p1z))], \n
        [float(vergleich(p2y)), float(vergleich(p2z))], \n
        [float(vergleich(p3y)), float(vergleich(p3z))]]
        dreieck = dreieck + [[[-p1y, -p1z], [-p2y, -p2z], \n
        [-p3y, -p3z]]]
    else:
        pass

    for c in ausschnitte:
        stabwerteaus = ausschnitte[c]
        if stabwerteaus =={}:
            pass
        else:
            geometrieaus = stabwerteaus["basisgeometrie"]
            if geometrieaus == "Kreis":
                [ym, zm, radius] \n
                = stabwerteaus["koordinatenkreis"]
                [ym, zm, radius] = [float(vergleich(ym)), \n
                float(vergleich(zm)), float(vergleich(radius))]
                kreisausschnitt = kreisausschnitt + [[-ym, -zm, \n
                radius]]
            elif geometrieaus == "Koordinaten␣Dicken" \n
            or geometrieaus == "Viereck" \n
            or geometrieaus == "Rechteck":
                [[p1y, p1z], [p2y, p2z], [p3y, p3z], [p4y, p4z]] \n
                = stabwerteaus["eckpunkte"]
                [[p1y, p1z], [p2y, p2z], [p3y, p3z], [p4y, p4z]] \n
                = [[float(vergleich(p1y)), float(vergleich(p1z))], \n
```

```
                [float(vergleich(p2y)), float(vergleich(p2z))], \n
                [float(vergleich(p3y)), float(vergleich(p3z))], \n
                [float(vergleich(p4y)), float(vergleich(p4z))]]
                viereckausschnitt = viereckausschnitt + \n
                [[[-p1y, -p1z], [-p2y, -p2z], [-p4y, -p4z], \n
                [-p3y, -p3z]]]
            elif geometrieaus == "Dreieck":
                [[p1y, p1z], [p2y, p2z], [p3y, p3z], [p4y, p4z]] \n
                = stabwerteaus["eckpunkte"]
                [[p1y, p1z], [p2y, p2z], [p3y, p3z]] \n
                = [[float(vergleich(p1y)), float(vergleich(p1z))], \n
                [float(vergleich(p2y)), float(vergleich(p2z))], \n
                [float(vergleich(p3y)), float(vergleich(p3z))]]
                dreieckausschnitt = dreieckausschnitt \n
                + [[[-p1y, -p1z], [-p2y, -p2z], [-p3y, -p3z]]]
            else:
                print ("error")
```

Im Anschluss werden die maßgebenden Größen berechnet. Bei einem Kreis entspricht dies dem Mittelpunkt $\pm$ zweimal dem Radius. Für den Bogen werden zum Mittelpunkt zweimal der Innenradius bzw. Außenradius addiert. Bei einem Dreieck oder Viereck wird einfach die Koordinatensumme gebildet.
Nachdem die Kennzahlen für jeden Abschnitt erstellt worden sind, wird davon für weitere Betrachtungen das Maximum und Minimum verwendet.

```
# calculate alpha
kreisy = []
kreisz = []
kreisbogeny = []
kreisbogenz = []
vierecky = []
viereckz = []
dreicky = []
dreickz = []

try:
    for c in kreis:
        kreisy = kreisy + [c[0]-2*c[2]]
        kreisz = kreisz + [c[1]-2*c[2]]
    maxkreisy = max(kreisy)
    minkreisy = min(kreisy)
    maxkreisz = max(kreisz)
    minkreisz = min(kreisz)
except:
    [maxkreisy,minkreisy,maxkreisz,minkreisz] = [0,0,0,0]

try:
    for c in kreisbogen:
        kreisbogeny = kreisbogeny + [c[0][0][0] - 2*c[1][4][0]]
        kreisbogenz = kreisbogenz + [c[0][0][1] - 2*c[1][4][0]]
        kreisbogeny = kreisbogeny + [c[0][0][0] - 2*c[0][4][0]]
        kreisbogenz = kreisbogenz + [c[0][0][1] - 2*c[0][4][0]]
    maxkreisbogeny = max(kreisbogeny)
    minkreisbogeny = min(kreisbogeny)
    maxkreisbogenz = max(kreisbogenz)
```

```
        minkreisbogenz = min(kreisbogenz)
    except:
        [maxkreisbogeny,minkreisbogeny,maxkreisbogenz,minkreisbogenz] \n
        = [0,0,0,0]

    try:
        for c in viereck:
            vierecky = vierecky + [c[0][0]] + [c[1][0]]+ [c[2][0]] \n
            + [c[3][0]]
            viereckz = viereckz + [c[0][1]] + [c[1][1]]+ [c[2][1]] \n
            + [c[3][1]]
        maxvierecky = max(vierecky)
        minvierecky = min(vierecky)
        maxviereckz = max(viereckz)
        minviereckz = min(viereckz)
    except:
        [maxvierecky,minvierecky,maxviereckz,minviereckz] = [0,0,0,0]

    try:
        for c in dreieck:
            dreicky = dreicky + [c[0][0]] + [c[1][0]] + [c[2][0]]
            dreickz = dreicky + [c[0][1]] + [c[1][1]] + [c[2][1]]
        maxdreiecky = max(dreicky)
        mindreiecky = min(dreicky)
        maxdreieckz = max(dreickz)
        mindreieckz = min(dreickz)
    except:
        [maxdreiecky,mindreiecky,maxdreieckz,mindreieckz]=[0,0,0,0]

    maxy = max([maxkreisy,maxkreisbogeny,maxvierecky,maxdreiecky])
    miny = min([minkreisy, minkreisbogeny, minvierecky, mindreiecky])
    maxz = max([maxkreisz, maxkreisbogenz, maxviereckz, maxdreieckz])
    minz = min([minkreisz, minkreisbogenz, minviereckz, mindreieckz])
```

Nun erfolgen geometrische Betrachtungen, um den Skalierungsfaktor “alpha“ zu erhalten. Dafür werden hauptsächlich die Seitenbreiten durch die maßgebenden Kennwerte geteilt. Das kleinste Ergebnis dieser Divisionen entspricht dann dem Skalierungsfaktor.

```
    #hier kann der faktor alpha angepasst werden
    alphalist = []
    if (maxy-miny) == 0:
        pass
    else:
        alpha1 = 15/(maxy-miny)
        alphalist.append(alpha1)

    if maxy == 0:
        pass
    else:
        alpha2 = 15/ maxy
        alphalist.append(alpha2)

    if miny == 0:
        pass
    else:
        alpha3 = 15 / abs(miny)
```

```
        alphalist.append(alpha3)

    if (maxz - minz)==0:
        pass
    else:
        alpha4 = 10 / (maxz - minz)
        alphalist.append(alpha4)

    if maxz == 0:
        pass
    else:
        alpha5 = 10 / maxz
        alphalist.append(alpha5)

    if minz==0:
        pass
    else:
        alpha6 = 10 / abs(minz)
        alphalist.append(alpha6)

    alpha = min(alphalist)
    print ("skalierungsfaktor",alpha)

    numkreis = len(kreis)
    for c in range(0,numkreis):
        kreis[c]= [*map(lambda x: x * alpha,kreis[c])]
```

Abschließend werden noch die Werte in der vorher gespeicherten Liste mit dem Skalierungsfaktor multipliziert.

```
    numbogen = len(kreisbogen)
    for c in range(0,numbogen):
        kreisbogen[c][0][0] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][0][0])]
        kreisbogen[c][0][1] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][0][1])]
        kreisbogen[c][0][2] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][0][2])]
        kreisbogen[c][0][4] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][0][4])]
        kreisbogen[c][1][0] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][1][0])]
        kreisbogen[c][1][1] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][1][1])]
        kreisbogen[c][1][2] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][1][2])]
        kreisbogen[c][1][4] = [*map(lambda x: x * alpha, \n
        kreisbogen[c][1][4])]


    numviereck = len(viereck)
    for c in range(0, numviereck):
        viereck[c][0] = [*map(lambda x: x * alpha, viereck[c][0])]
        viereck[c][1] = [*map(lambda x: x * alpha, viereck[c][1])]
        viereck[c][2] = [*map(lambda x: x * alpha, viereck[c][2])]
        viereck[c][3] = [*map(lambda x: x * alpha, viereck[c][3])]
```

```
numdreieck = len(dreieck)
for c in range(0, numdreieck):
    dreieck[c][0] = [*map(lambda x: x * alpha, dreieck[c][0])]
    dreieck[c][1] = [*map(lambda x: x * alpha, dreieck[c][1])]
    dreieck[c][2] = [*map(lambda x: x * alpha, dreieck[c][2])]

#Ausschnitte
numkreisausschnitt = len(kreisausschnitt)
for c in range(0,numkreisausschnitt):
    kreisausschnitt[c]= [*map(lambda x: x * alpha, \n
    kreisausschnitt[c])]

numviereckausschnitt = len(viereckausschnitt)
for c in range(0, numviereckausschnitt):
    viereckausschnitt[c][0] = [*map(lambda x: x * alpha, \n
    viereckausschnitt[c][0])]
    viereckausschnitt[c][1] = [*map(lambda x: x * alpha, \n
    viereckausschnitt[c][1])]
    viereckausschnitt[c][2] = [*map(lambda x: x * alpha, \n
    viereckausschnitt[c][2])]
    viereckausschnitt[c][3] = [*map(lambda x: x * alpha, \n
    viereckausschnitt[c][3])]

numdreieckausschnitt = len(dreieckausschnitt)
for c in range(0, numdreieckausschnitt):
    dreieckausschnitt[c][0] = [*map(lambda x: x * alpha, \n
    dreieckausschnitt[c][0])]
    dreieckausschnitt[c][1] = [*map(lambda x: x * alpha, \n
    dreieckausschnitt[c][1])]
    dreieckausschnitt[c][2] = [*map(lambda x: x * alpha, \n
    dreieckausschnitt[c][2])]

winkel = float(vergleich(phi))

ys_skaliert = float(vergleich(ys)*alpha)
zs_skaliert = float(vergleich(zs)*alpha)

p1y_hauptachse = float(ys_skaliert+cos(winkel))
p1z_hauptachse = float(zs_skaliert+sin(winkel))
p2y_hauptachse = float(ys_skaliert+cos(winkel+pi/2))
p2z_hauptachse = float(zs_skaliert+sin(winkel+pi/2))
hauptachse_schwerpunkt = [[-p1y_hauptachse,-p1z_hauptachse], \n
[-ys_skaliert,-zs_skaliert],[-p2y_hauptachse,-p2z_hauptachse]]

dictionaryhautachse_schwerpunkt \n
= {"hauptachse":hauptachse_schwerpunkt}
dictionarybasis = {"kreis":kreis, "kreisbogen":kreisbogen, \n
"viereck":viereck, "dreieck":dreieck}
dictionaryausschnitt = {"kreis": kreisausschnitt, \n
"viereck": viereckausschnitt, "dreieck": dreieckausschnitt}
dictionarys =[dictionarybasis,dictionaryausschnitt, \n
dictionaryhautachse_schwerpunkt,alpha]
return dictionarys
```

#### 3.3.1.2 Funktion: beschriftung(dictionary, dictionaryausschnitte, alpha)

Zuerst werden bei dieser Funktion alle zu bemaßenden Punkte abgespeichert.

```
def beschriftung(dicionary,dictionaryausschnitte,alpha):
    #Hier die Anpassung der Abstaende vornehmen::
    ymin_bemassung = 1
    zmin_bemassung = 1
    versatzknoteny = 1.3
    versatzknotenz = 1.3
    deltaversatzknotenz=0.3

    allpoints = []
    pointsy = []
    pointsz = []
    differenzeny = []
    differenzenz = []

    knoteny = []
    knotenz = []

    for c in dicionary:
        if dictionary[c]["basisgeometrie"] == "Kreisbogen":
            [mpy, mpz, p1y, p1z, p2y, p2z, winkel1, winkel2, radius1, \n
            radius2] = dicionary[c]["auswertungkreisbogen"]
            allpoints += [p1y,p1z,p2y,p2z]
        elif dictionary[c]["basisgeometrie"] == "Kreis":
            [y,z,r]= dicionary[c]["koordinatenkreis"]
            allpoints += [y-r,z-r,y,z,y+r,z+r]
        else:
            [p1y,p1z,p2y,p2z] = [dicionary[c]["koordinatenyanfang"], \n
            dicionary[c]["koordinatenzanfang"], \n
            dicionary[c]["koordinatenyende"], \n
            dicionary[c]["koordinatenzende"]]
            allpoints += [p1y,p1z,p2y,p2z]

    for c in dictionaryausschnitte:
        try:
            if dictionaryausschnitte[c]["basisgeometrie"] == "Kreis":
                [ya, za, ra] \n
                = dictionaryausschnitte[c]["koordinatenkreis"]
                allpoints += [ya-ra,za-ra,ya,za,ya+ra,za+ra]
            else:
                [p1y,p1z,p2y,p2z] \n
                = [dictionaryausschnitte[c]["koordinatenyanfang"], \n
                dictionaryausschnitte[c]["koordinatenzanfang"], \n
                dictionaryausschnitte[c]["koordinatenyende"], \n
                dictionaryausschnitte[c]["koordinatenyende"]]
                allpoints += [p1y,p1z,p2y,p2z]
        except:
            pass
```

Dabei werden die Punkte der Größe nach sortiert und doppelte Einträge entfernt.

```
    pointsy = allpoints[::2]
    pointsz = allpoints[1::2]
```

```
pointsy = sorted(pointsy, key=lambda x: vergleich(x))
pointsz = sorted(pointsz, key=lambda x: vergleich(x))

pointsy = unique(pointsy)
pointsz = unique(pointsz)

#Abgrenzungen
pointsy_koo = [*map(lambda x: vergleich(x) * alpha, pointsy)]
pointsz_koo = [*map(lambda x: vergleich(x) * alpha, pointsz)]
```

Nun werden die Differenzen, Knotenpunkte der Beschriftung und Maßkette berechnet.

```
#Werte Beschriftung
count = 0
for c in range (0,len(pointsy)):
    if count == 0:
        pass
    else:
        abstandy = [simplify(pointsy[c]-pointsy[c-1]).evalf(3)]
        differenzeny += abstandy
    count =+ 1

count = 0
for c in range(0,len(pointsz)):
    if count == 0:
        pass
    else:
        abstandz = [simplify(pointsz[c]-pointsz[c-1]).evalf(3)]
        differenzenz += abstandz
    count =+ 1

#Knoten Beschriftung
count = 0
for c in range(0,len(pointsy_koo)):
    if count == 0:
        pass
    else:
        kooy = [(pointsy_koo[c - 1]+ (pointsy_koo[c] \n
        - pointsy_koo[c - 1])/2)]
        knoteny += kooy
    count = + 1

count = 0
for c in range(0,len(pointsz_koo)):
    if count == 0:
        pass
    else:
        kooz = [(pointsz_koo[c - 1] + (pointsz_koo[c] \n
        - pointsz_koo[c - 1])/2)]
        knotenz += kooz
    count = + 1

#Auswertung
werteketteY = []
werteketteZ = []
```

```
posknoteny = []
posknotenz = []

knoten_beschriftungy = []
knoten_beschriftungz = []


#Masskette
zkoo = float(pointsz_koo[0]-zmin_bemassung)
for c in range(0,len(pointsy_koo)):
    wertekettey += [[float(-pointsy_koo[c]),-zkoo]]

ykoo = float(pointsy_koo[0]-ymin_bemassung)
for c in range(0,len(pointsz_koo)):
    wertekettez += [[-ykoo,float(-pointsz_koo[c])]]

#Position Knoten
counter = 0
for c in range(0,len(knoteny)):
    counter += 1
    zkn = float(pointsz_koo[0] - versatzknotenz)
    zkn += deltaversatzknotenz*((-1)**counter)
    posknoteny += [[float(-knoteny[c]),-zkn]]

ykn = float(pointsy_koo[0]-versatzknoteny)
for c in range(0,len(knotenz)):
    posknotenz += [[-ykn,float(-knotenz[c])]]

for c in range (0,len(posknoteny)):
    knoten_beschriftungy += [[posknoteny[c][0],posknoteny[c][1], \n
    str(differenzeny[c])]]
for c in range (0,len(posknotenz)):
    knoten_beschriftungz += [[posknotenz[c][0], posknotenz[c][1], \n
    str(differenzenz[c])]]


return [wertekettey,wertekettez,differenzeny,differenzenz, \n
posknoteny,posknotenz,knoten_beschriftungy,knoten_beschriftungz]
```

### 3.3.2 Klassen: Skizze (MainWindow)/ Ausgabe (MainWindow)

#### 3.3.2.1 Funktion

Die auszugebenden Formeln bzw. die Querschnittskizze werden zuvor am Ende der Klasse Berechnung in einen LaTeX-fähigen Code gebracht. Dazu später mehr im nächsten Unterpunkt. Mit dem Befehl „texfile.write(„\\...“ wird ein LaTeX– Dokument erstellt. Für die Querschnittskizze wird die zuvor als „standalone“ erzeugte Skizze eingefügt. Für die Ergebnisausgabe werden die bereits übersetzten Formeln in die tex – Datei aufgenommen. Mit den „os.“-Befehlen wird das PDF erzeugt und geöffnet.
Als Beispiel wird die Klasse der Skizze hier gezeigt:

```
class Skizze(MainWindow):
    def __init__(self):
        #LaTex-Skrip-Erstellung
        with open("QSSK_BAC_Projekt_16_Ausgabe" \n
        "_02.tex", "w") as texfile:
            texfile.write("\\documentclass[a4paper," \n
            "11pt]{scrartcl}\n")
            texfile.write("\\usepackage{standalone}\n")
            texfile.write("\\usepackage{tikz}\n")

            texfile.write("\\begin{document}\n")
            texfile.write("\\begin{center}\n")
            texfile.write("\input{QSSK_BAC_Projekt_16" \n
            "_Ausgabe_01}\n")
            texfile.write("\\end{center}\n")

            texfile.write("\\end{document}\n");


            texfile.close()

            os.system("pdflatex␣QSSK_BAC_Projekt_16" \n
            "_Ausgabe_02.tex")
            os.startfile("QSSK_BAC_Projekt_16_Ausgabe" \n
            "_02.pdf")
```

Analog funktioniert, wie oben bereits beschrieben, auch die Klasse zur Ausgabe der Formeln. Hier gibt es allerdings noch einige if-Schleifen, da bei nur einem Ergebnis oder zu langen Ausdrücken für die Breite eines A4-Blattes keine Zwischenergebnisse angezeigt werden und bei der Anzeige von Zwischenergebnissen nach jedem ein Pluszeichen steht bzw. nach dem letzten das Gleichheitszeichen.
Als Beispiel wird hier die Ausgabe der Fläche dargestellt:

```
            #Ausgabe der Flaeche
            texfile.write("Fl\"ache:\n")
            texfile.write("\\begin{equation*}\n")
            texfile.write("A=\int_A␣\;\\text{d}A␣=")
            texfile.write("\\end{equation*}\n")
            texfile.write("\\begin{equation*}\n")
            texfile.write("\\begin{split}\n")
            if (len(A_tex_EE)) == 1:
```

```
            texfile.write(str(A_tex_ges)+"\\bigg" \n
            "[␣m^2␣\\bigg]"+"\n")
        else:
            if Rechentyp == 'symb' and  max(Anz_A)>= 110:
                texfile.write(str(A_tex_ges)+"\\bigg" \n
                "[m^2\\bigg]"+"\n")
            else:
                for position in range ( 0, len \n
                (A_tex_EE)+1, 1):
                    if position < (len(A_tex_EE))-1:
                        print("wert␣" + str(position))
                        print("len" + str(len(A_tex_EE)-1))
                        print()
                        texfile.write(str(A_tex_EE \n
                        [position]))
                        texfile.write("+")
                        texfile.write("\\")
                        texfile.write("\\")
                    elif position == (len(A_tex_EE))-1:
                        texfile.write(str(A_tex_EE \n
                        [position]))
                        texfile.write("=")
                        texfile.write("\\")
                        texfile.write("\\")
                        texfile.write(str(A_tex_ges)+ \n
                        "\\bigg[m^2\\bigg]"+"\n")
    texfile.write("\\end{split}\n")
    texfile.write("\\end{equation*}\n")
```

Für die anderen Querschnittskennwerte funktioniert der Ablauf analog wie hier.
Anmerkung: „\n“ innerhalb von Anführungsstrichen (und in pinker Schrift) sind kein Umbruch für das vorliegende Dokumentenformat, sondern für LaTeX bei der Ergebnisausgabe erforderlich (Ende der Zeile bei automatischer Übersetzung endet mit diesem Zeichen).
Zur Erstellung der LaTeX-Übersetzung im Bereich der Berechnungsklassen (sowohl symbolisch, als auch numerisch funktioniert dies ident) ist zwar keine Funktion im engeren Sinn notwendig, jedoch sollte die Vorgangsweise hier dennoch kurz erläutert werden. Die oberhalb eingegebenen bzw. berechneten Parameter der Querschnittsgeometrie (Eckpunkte, Mittelpunkt, Radius, Winkeln etc.) werden in bestimmten Reihenfolgen in Listen gespeichert. Zur Übersetzung wird über diese Listen iteriert und es werden in der, für das LaTeX- Zeichenprogramm tikz notwendigen Reihenfolge, die Befehlszeichen und Wörter mit den Parameterwerten kombiniert. Unten wird dies anhand des Kreises gezeigt. Für die anderen Basisgeometrien und Ausschnitte wird analog vorgegangen.

```
#BG-Kreis
    if len(list_bg_ski_kreis) >= 1:
        for AnzEl_K in range(0, len(list_bg_ski \n
            _kreis),1):
            texfile.write("\\draw[thick]␣")
            Skizzenkoordinaten_Kreis = \n
            list_bg_ski_kreis[AnzEl_K]
            print(Skizzenkoordinaten_Kreis)
            global SP_tex_K
            SP_tex_K = ''
            SP_tex_K += '('
```

```
SP_tex_K += \n
str(Skizzenkoordinaten_Kreis[0])
SP_tex_K += ','
SP_tex_K += \n
str(Skizzenkoordinaten_Kreis[1])
SP_tex_K += ') circle ('
SP_tex_K += \n
str(Skizzenkoordinaten_Kreis[2])
SP_tex_K += ');'
print(SP_tex_K)
texfile.write(str(SP_tex_K) + "\n")
```

Die Bemaßung funktioniert ebenfalls so, dass über Listen iteriert wird, in denen die Punkte der Pfeile auf den Achsen sind. Dies wird für beide Achsen durchgeführt. Anschließend wird noch über eine weiter Liste iteriert, und somit die Beschriftung eingefügt. Als Beispiel wird folgend die Maßkette der y-Achse dargestellt:

```
#Bemassung
    texfile.write("%Bemassung\n")

#Linien
    #horizontal (y-Achse)
    for pos_horiz in range (0,len \n
    (list_bm_lin_horiz)-1,1):
        if pos_horiz <= len(list_bm_lin_horiz)-2:
            list_1 = list_bm_lin_horiz[pos_horiz]
            pos2 = pos_horiz + 1
            print(pos2)
            print(list_bm_lin_horiz)
            list_2 = list_bm_lin_horiz[pos2]
            global punkt_bem
            punkt_horiz_bem = ''
            texfile.write('\draw[|<->|](')
            punkt_horiz_bem += str(list_1[0])
            punkt_horiz_bem += ','
            punkt_horiz_bem += str(list_1[1])
            punkt_horiz_bem += ')--('
            punkt_horiz_bem += str(list_2[0])
            punkt_horiz_bem += ','
            punkt_horiz_bem += str(list_2[1])
            punkt_horiz_bem += ');\n'
        texfile.write(str(punkt_horiz_bem))
        print('horizLine'+str(punkt_horiz_bem))
```

Anschließend werden die Rechenergebnisse in LaTeX- fähigen Code übersetzt (Befehl „latex(..)“). Wichtig ist hier, dass die Variablen wieder global gesetzt werden, da sie erst in der Klasse „Ausgabe“ verwendet werden. Hierzu ein Beispiel:

```
Anz_Sy=[]
global Anz_Sz
Anz_Sz=[]
global Anz_Iy
Anz_Iy=[]
global Anz_Iz
```

```
Anz_Iz=[]
global Anz_Iyz
Anz_Iyz=[]

for AT in liste_A_Ausgabe:
    AT_tex = latex(AT)
    A_tex_EE.append(AT_tex)
    Anz_A_TE=len(AT_tex)
    Anz_A.append(Anz_A_TE)
print("LaTeX-Skript A" + str(A_tex_EE))
print("Anz_A " + str(Anz_A))
print("maxAnzA " +str(max(Anz_A)))


A_tex_ges=(latex(flaeche_ausgabe))
```

Analog funktioniert dies für die anderen Querschnittskennwerte sowie im Bereich der numerischen Berechnung. Einziger Unterschied ist, dass die Zahlen auf vier Nachkommastellen gerundet werden. Als Beispiel hier angegeben für $y_s$ und $z_s$ :

```
if Rechentyp == 'num':
    ys=round(ys,4)
ys_tex=latex(ys_ausgabe)
print (ys_tex)

if Rechentyp == 'num':
    zs=round(zs,4)
zs_tex=latex(zs_ausgabe)
```

# 4 PROJEKTREFLEXION

## 4.1 Problemstellung

Für die Implementierung des Programmes waren gute Kenntnisse hinsichtlich der Berechnung von Querschnittskennwerten erforderlich. Zudem musste ein Berechnungsalgorithmus entwickelt werden, welcher auf nahezu beliebige Geometrien angewendet werden kann und dabei stets korrekte Ergebnisse liefert. Ein zusätzliches Erfordernis war, dass Querschnitte nicht nur numerisch, sondern auch symbolisch eingegeben werden können. Des Weiteren war in der Aufgabenstellung enthalten, dass die Eingabe mittels GUI-Programmierung verwirklicht werden sollte und das Ergebnis der Ausgabe eine .pdf-Datei zu sein hat.

## 4.2 Problemlösung

Die Vorgehensweise zur Findung eines passenden Algorithmus war, vorerst den Fokus nur auf die Funktion zu legen. Dazu wurde mit Excel die grobe Struktur vernetzt. Im Anschluss wurde die Struktur in Python implementiert. Im späteren Verlauf stellte sich jedoch heraus, dass der anfängliche Ansatz für symbolische Berechnungen aufgrund enorm ansteigender Komplexität der Vergleichsoperationen nicht geeignet war. Deshalb wurde nach einer anderen Methode gesucht. Dank einem Hinweis von unserem Betreuer kamen wir auf eine stabile und effiziente Lösung des Problems. Dies beinhaltete die interdisziplinäre Verknüpfung der darstellenden Geometrie, Mathematik und Mechanik. Um die Implementierung bewerkstelligen zu können war es erforderlich, sich intensiv mit Python3 und LaTeX zu beschäftigen.

## 4.3 Lernerfolge

Rückblickend war es für uns ein sehr interessantes Projekt, welches jedoch durchaus mit hohen Anforderungen verbunden war. Insbesondere vertieften wir unsere Programmierkenntnisse. Zudem sind wir bei der Implementierung auf viele Probleme gestoßen und durch die Findung einer geeigneten Lösung konnten wir ein tieferes Verständnis für die mathematischen und mechanischen Hintergründe erlangen. Auch der organisatorische Aspekt erforderte ein strukturiertes Vorgehen und eine koordinierte Zusammenarbeit mit dem Teampartner.

# 5 BEISPIELE

Folgend werden einige Beispiele gezeigt. Hierzu werden die Ergebnisse, wie sie vom Programm ausgegeben werden, nachgerechnet und verglichen. Der Kopf der Programmausgabe wird nicht jedes Mal dargestellt, da er immer gleich bleibt. Daher wird der Kopf hier einmal dargestellt.

BAC Projekt

P. Egg, M. Kaiser

3. Januar 2017

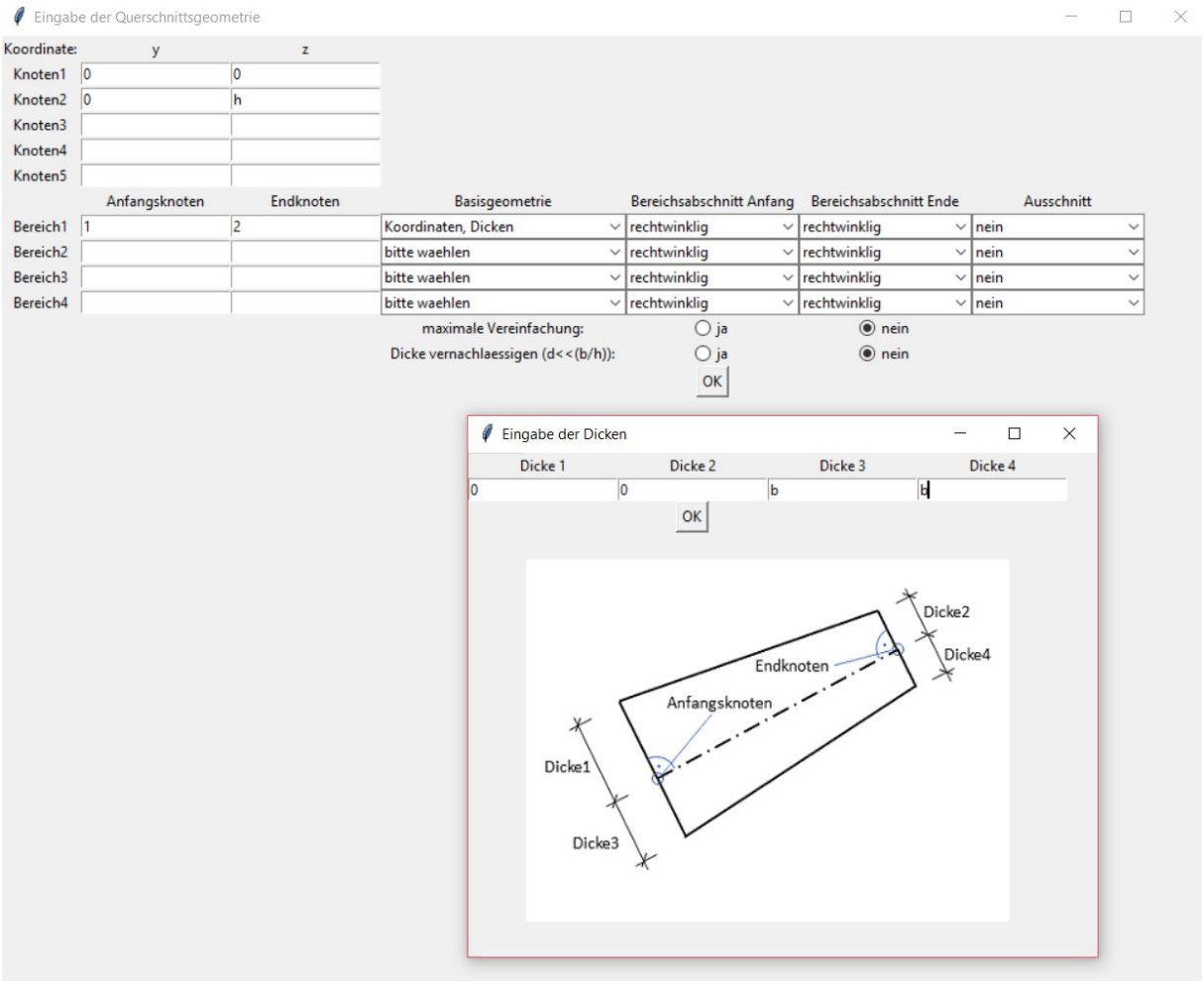
## 1 Projektbezeichnung

Es wurde keine spezifische Projektbezeichnung angegeben!

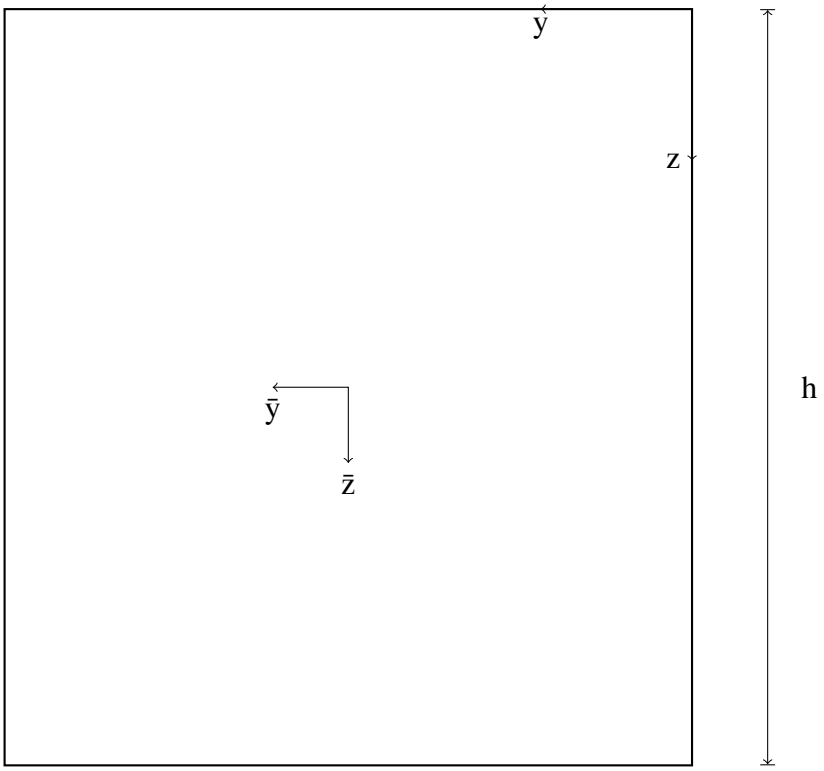
Wird im Eingabefenster eine individuelle Projektbezeichnung eingegeben, dann wird diese unter *1 Projektbezeichnung* angegeben, ansonsten wird ein Hinweis gedruckt, dass keine Bezeichnung eingegeben wurde.

## 5.1 Beispiel 1

Als erstes Beispiel wird ein rechteckiger Querschnitt behandelt, welcher die Höhe h und der Breite b aufweist.
Die Eingabe erfolgt hierbei über die Option “Koordinaten, Dicken“. Alternativ wäre auch eine koordinative Eingabe möglich.



### 5.1.1 Programmausgabe Beispiel 1




Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$

$$bh\left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z\,\mathrm{d}A =$$

$$\frac{bh^2}{2}\left[m^3\right]$$

$$S_z = \int_A y\,\mathrm{d}A =$$

$$\frac{b^2h}{2}\left[m^3\right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = \frac{b}{2}\left[m\right]$$

$$z_s = \frac{S_y}{A} = \frac{h}{2}\left[m\right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A =$$

$$\frac{bh^3}{12} \left[ m^4 \right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$\frac{b^3 h}{12} \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$0 \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$\frac{bh}{24} \left( b^2 + h^2 + \left| b^2 - h^2 \right| \right) \left[ m^4 \right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$\frac{bh}{24} \left( b^2 + h^2 - \left| b^2 - h^2 \right| \right) \left[ m^4 \right]$$

Die Hauptachse 1 entspricht Iy um 0 ° gedreht.

### 5.1.2 Handrechnung zu Beispiel 1

Fläche:

$$A = bh$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A$$

$$S_y = \int_{y=0}^{b} \int_{z=0}^{h} z \, dz dy$$

$$S_y = \frac{h^2 b}{2}$$

$$S_z = \int_{y=0}^{b} \int_{z=0}^{h} y \, dz dy$$

$$S_z = \frac{b^2 h}{2}$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = \frac{\frac{b^2 h}{2}}{bh} = \frac{b}{2}$$

$$z_s = \frac{S_y}{A} = \frac{\frac{h^2 b}{2}}{bh} = \frac{h}{2}$$

$$I_y = \int_A z^2 \, \mathrm{d}A$$

$$I_z = \int_A y^2 \, \mathrm{d}A$$

$$I_{yz} = \int_A yz \, \mathrm{d}A$$

Flächenträgheitsmomente:

$$I_y = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=\frac{-h}{2}}^{\frac{h}{2}} z^2 \, dzdy$$

$$I_y = \frac{h^3 b}{12}$$

$$I_z = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=\frac{-h}{2}}^{\frac{h}{2}} y^2 \, dzdy$$

$$I_z = \frac{b^3 h}{12}$$

$$I_{yz} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=\frac{-h}{2}}^{\frac{h}{2}} yz \, dzdy$$

$$I_{yz} = 0$$

Für die Hauptträgheitsmomente gilt:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$

$$I^{(1)} = I_y = \frac{bh^3}{12}$$

$$I^{(2)} = I_z = \frac{hb^3}{12}$$

Für den Drehwinkel gilt:

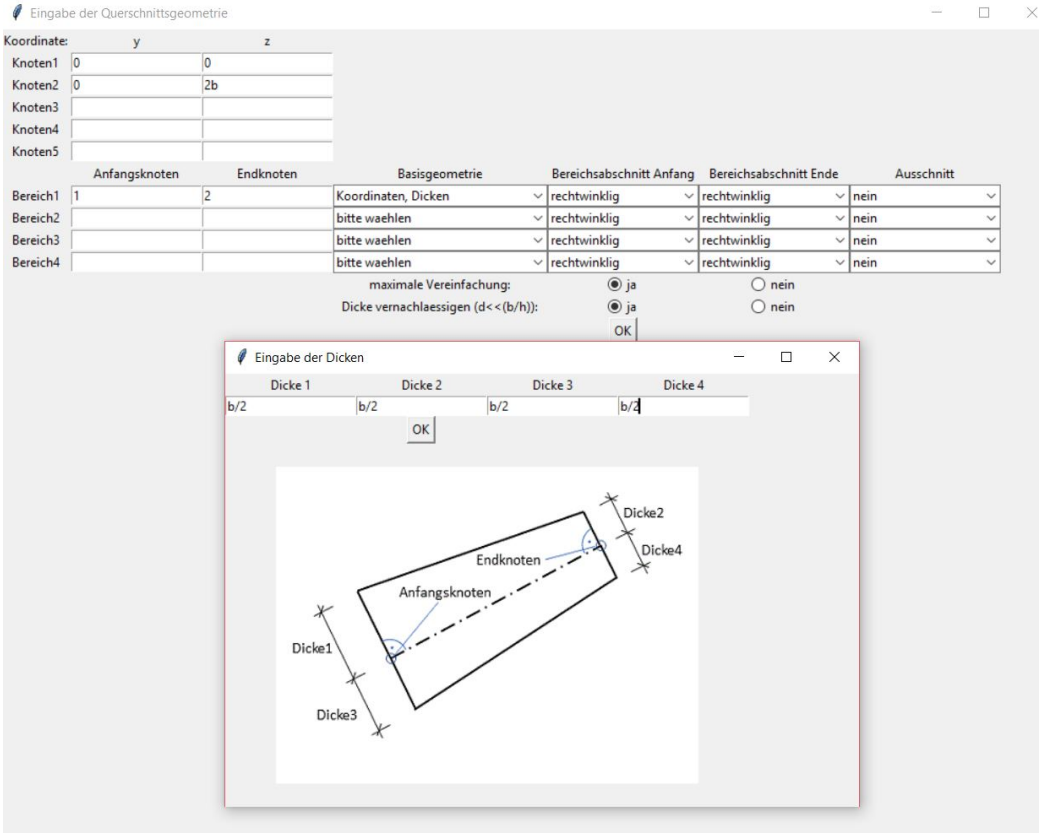
$$tan(2\phi) = -\frac{2I_{yz}}{I_y - Iz}$$

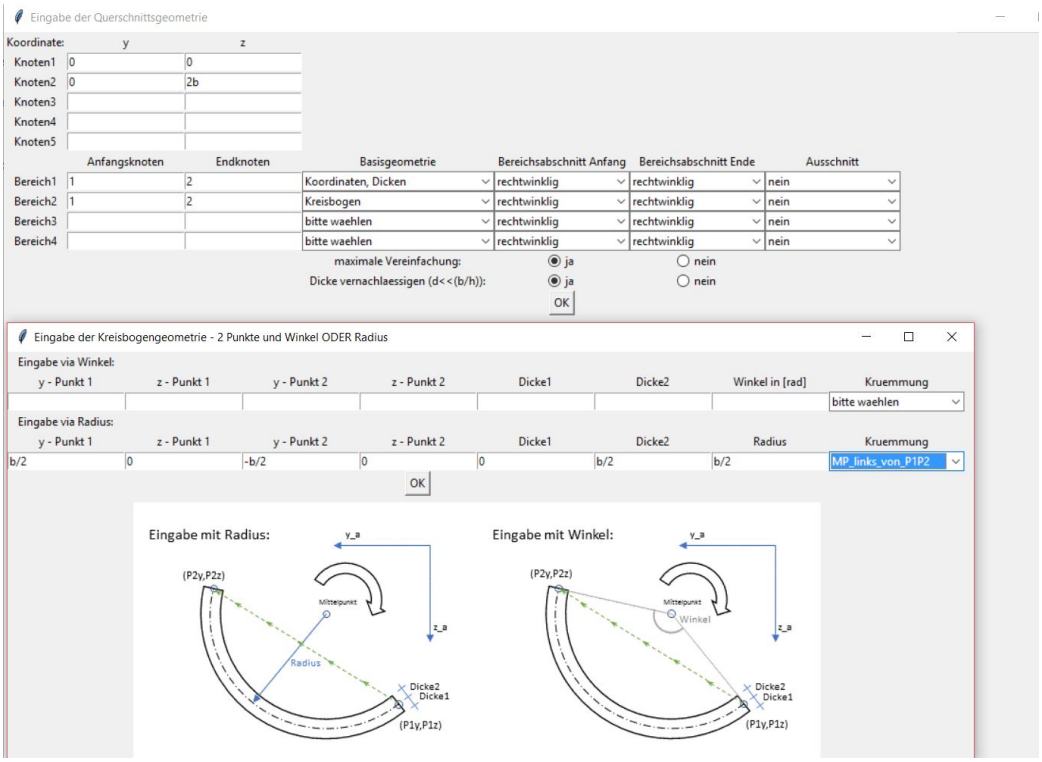
Da $I_{yz} = 0$ gilt, ergibt sich der Drehwinkel $\phi$ zu $\frac{arctan(0)}{2} = 0$.

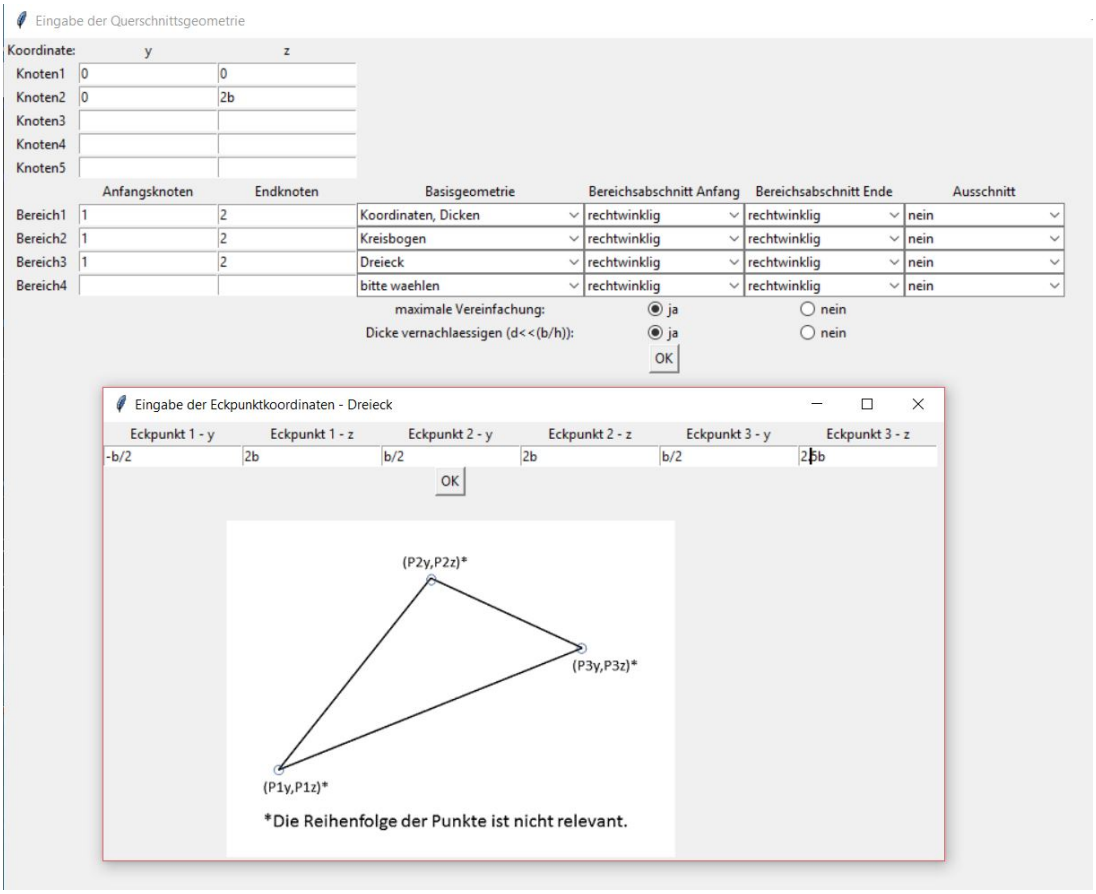
Somit stimmen die Ergebnisse der Handrechnung von Beispiel 1 mit jenen des Programmes überein.

## 5.2 Beispiel 2

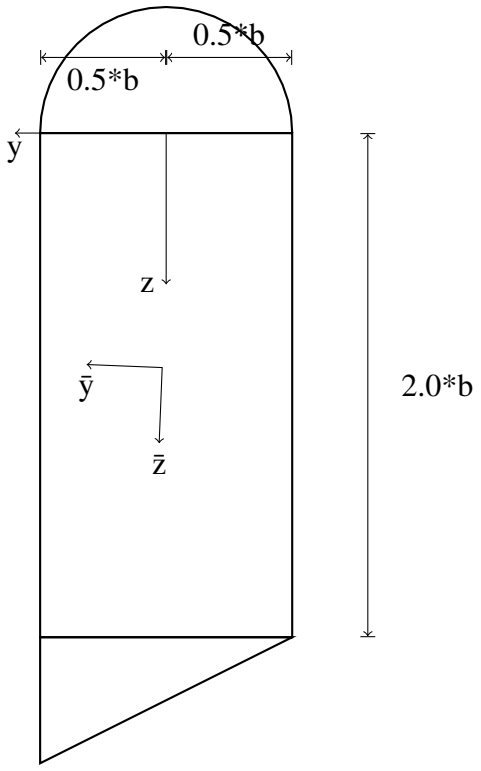
Das Beispiel 2 ist ein Rechteck, auf welchem ein Dreieck aufgesetzt wird. Zusätzlich wird eine halbkreisförmige Scheibe angehängt.






### 5.2.1 Programmausgabe Beispiel 2



Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$
$$2.0b^2 +$$
$$0.393b^2 +$$
$$0.25b^2 =$$
$$2.64b^2 \left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z\,\mathrm{d}A =$$
$$2.0b^3 +$$
$$-0.0833b^3 +$$
$$0.542b^3 =$$
$$2.46b^3 \left[m^3\right]$$

$$S_z = \int_A y\,\mathrm{d}A =$$
$$0 +$$
$$0 +$$
$$0.0417b^3 =$$
$$0.0417b^3 \left[m^3\right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0.0158b \left[m\right]$$
$$z_s = \frac{S_y}{A} = 0.93b \left[m\right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2\,\mathrm{d}A =$$
$$0.676b^4 +$$
$$0.519b^4 +$$
$$0.386b^4 =$$
$$1.58b^4 \left[m^4\right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$0.167b^4 +$$

$$0.0245b^4 +$$

$$0.0195b^4 =$$

$$0.211b^4 \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$\left( -0.0022b^4 \right) +$$

$$\left( 0.00707b^4 \right) +$$

$$\left( 0.0501b^4 \right) =$$

$$0.055b^4 \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$1.58b^4 \left[ m^4 \right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$0.209b^4 \left[ m^4 \right]$$

Die Hauptachse 1 entspricht Iy um $-2.29\,^\circ$ gedreht.

### 5.2.2 Handrechnung zu Beispiel 2

Fläche:

Teilfläche Rechteck:

$$A_1 = b2b = 2b^2$$

Teilfläche Dreieck:

$$A_2 = \frac{b^2}{4} = 0.25b^2$$

$$A_3 = \frac{r^2\pi}{2} = \frac{\pi b^2}{8} = 0.393b^2$$

$$A = A_1 + A_2 + A_3 = 2.643b^2$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A$$

Statisches Moment Rechteck:

$$S_{y1} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=0}^{2b} z \, dzdy$$

$$S_{y1} = 2b^3$$

$$S_{z1} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=0}^{2b} y \, dzdy$$

$$S_{z1} = 0$$

Statisches Moment Dreieck:

$$S_{y2} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=2b}^{\frac{9b}{4}+\frac{y}{2}} z \, dzdy$$

$$S_{y2} = 0.54167b^3$$

$$S_{z2} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=2b}^{\frac{9b}{4}+\frac{y}{2}} y \, dzdy$$

$$S_{z2} = 0.0416b^3$$

Statisches Moment Kreis:

$$S_{y3} = \frac{r^2\pi}{2}\left(-\frac{4r}{3\pi}\right)$$

$$S_{y3} = -0.0833b^3$$

$$S_{z3} = \frac{r^2\pi}{2} * 0$$

$$S_{z3} = 0$$

Somit folgt $S_y$ als Summe von $S_{y1}$ + $S_{y2}$ + $S_{y3}$:

$$S_y = 2b^3 + 0.54167b^3 + (-0.0833b^3) = 2.458b^3$$

Analoges Vorgehen für $S_z$:

$$S_z = 0 + 0.0416b^3 + 0 = 0.0416b^3$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = \frac{0.0416b^3}{2.643b^2} = 0.01574b$$

$$z_s = \frac{S_y}{A} = \frac{2.458b^3}{2.643b^2} = 0.932b$$

Flächenträgheitsmomente:
Die Flächenträgheitsmomente setzten sich aus den Flächenträgheitsmomenten der Einzelbereiche und den zugehörigen Steineranteilen zusammen.

$$I_y = \int_A z^2 \, \mathrm{d}A$$

$$I_z = \int_A y^2 \, \mathrm{d}A$$

$$I_{yz} = \int_A yz \, \mathrm{d}A$$

Satz von Steiner:

$$I'_y = \int_A z^2 \, \mathrm{d}A + A(z'_s)^2$$

$$I'_z = \int_A y^2 \, \mathrm{d}A + A(y'_s)^2$$

$$I'_{yz} = \int_A yz \, \mathrm{d}A + A(y'_s z'_s)$$

Flächenträgheitsmoment Rechteck

$$I_{y1} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=-b}^{b} z^2 \, dzdy + 2b^2(z_s - b)^2$$

$$I_{y1} = \frac{(2b)^3 b}{12} + 2b^2(-0.068b)^2 = 0.676b^4$$

$$I_{z1} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=-b}^{b} y^2 \, dzdy + 2b^2(y_s)^2$$

$$I_{z1} = \frac{(2b)^3 b}{12} + 2b^2(0.0158b)^2 = 0.167b^4$$

$$I_{yz1} = \int_{y=\frac{-b}{2}}^{\frac{b}{2}} \int_{z=-b}^{b} yz \, dzdy + 2b^2(y_s)(z_s - b)$$

$$I_{yz1} = 0 - 0.022b^4 = -0.022b^4$$

Flächenträgheitsmoment Dreieck

$$I_{y2} = \int_{y=\frac{-2b}{3}}^{\frac{b}{3}} \int_{z=\frac{-b}{6}}^{\frac{b}{6}+\frac{y}{2}} z^2 \, dzdy + \frac{b^2}{4}\left(2b + \frac{b}{6} - z_s\right)^2$$

$$I_{y2} = 0.0034b^4 + 0.382b^4 = 0.386b^4$$

$$I_{z2} = \int_{y=\frac{-2b}{3}}^{\frac{b}{3}} \int_{z=\frac{-b}{6}}^{\frac{b}{6}+\frac{y}{2}} y^2 \, dzdy + \frac{b^2}{4}\left(\frac{b}{6} - y_s\right)^2$$

$$I_{z2} = 0.0139b^4 + 0.0555b^4 = 0.195b^4$$

$$I_{yz2} = \int_{y=\frac{-2b}{3}}^{\frac{b}{3}} \int_{z=\frac{-b}{6}}^{\frac{b}{6}+\frac{y}{2}} yz \, dzdy + \frac{b^2}{4}\left(\frac{b}{6} - y_s\right)\left(2b + \frac{b}{6} - z_s\right)$$

$$I_{yz2} = 0.0138b^4 + 0.0466b^4 = 0.05011b^4$$

Flächenträgheitsmoment Kreis

$$I_{y3} = \frac{r^4}{72\pi}(9\pi^2 - 64) + \frac{r^2\pi}{2}\left(-z_s - \frac{4r}{3\pi}\right)^2$$
$$I_{y3} = 0.00686b^4 + 0.5123b^4 = 0.519b^4$$

$$I_{z3} = \frac{r^4\pi}{8} + \frac{r^2\pi}{2}(-y_s)^2$$
$$I_{z3} = 0.0245b^4 + 0.0001b^4 = 0.0245^4$$

$$I_{yz3} = 0 + \frac{r^2\pi}{2}(-y_s)\left(-z_s - \frac{4r}{3\pi}\right)$$
$$I_{yz3} = 0 + 0.00707b^4 = 0.00707b^4$$

Somit folgt $I_y$ als Summe von $I_{y1} + I_{y2} + I_{y3}$:

$$I_y = 0.676b^4 + 0.386b^4 + 0.519b^4 = 1.58b^4$$

Somit folgt $I_z$ als Summe von $I_{z1} + I_{z2} + I_{z3}$:

$$I_z = 0.167b^4 + 0.0195b^4 + 0.0254b^4 = 0.211b^4$$

Somit folgt $I_{yz}$ als Summe von $I_{yz1} + I_{yz2} + I_{yz3}$:

$$I_{yz} = -0.0022^4 + 0.0501b^4 + 0.00707b^4 = 0.055b^4$$

Hautträgheitsmomente:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$
$$I^{(1)} = \frac{1.58b^4 + 0.211b^4}{2} \pm \sqrt{\left(\frac{1.58b^4 - 0.211b^4}{2}\right)^2 + (0.055b^4)^2}$$
$$I^{(1)} = 1.5822b^4$$

$$I^{(2)} = 0.209b^4$$

Für den Drehwinkel gilt:

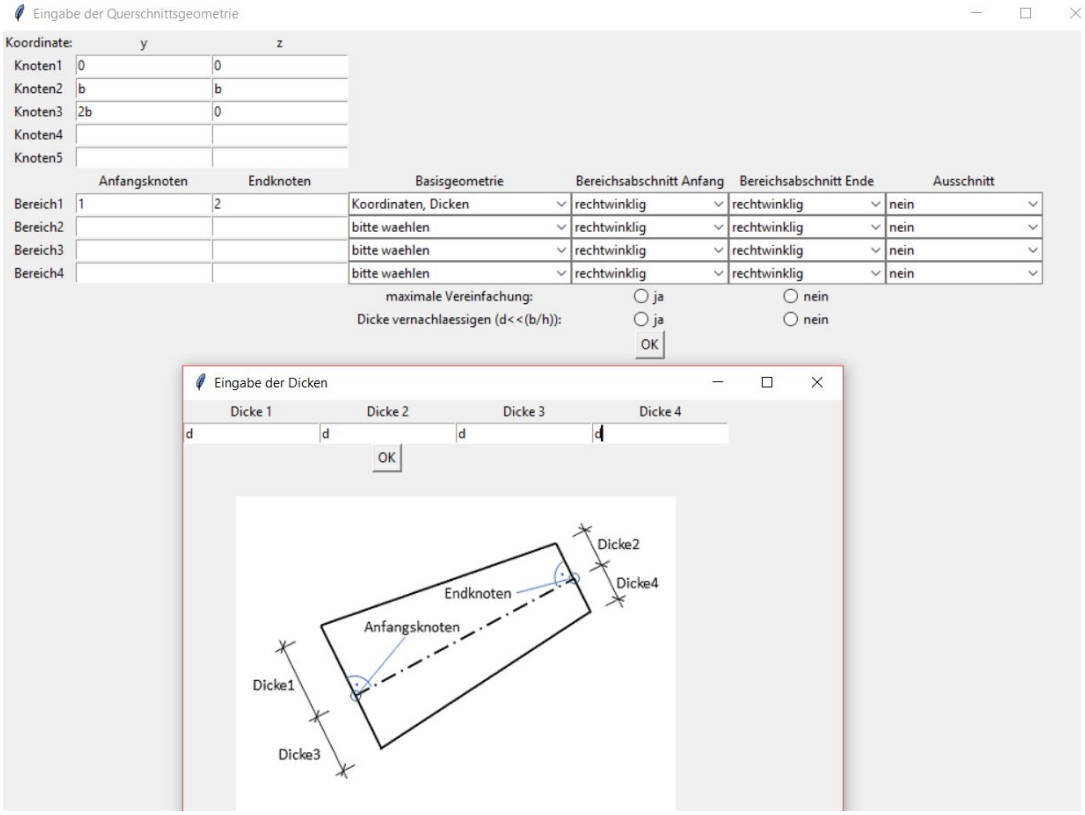
$$tan(2\phi) = -\frac{2(0.055b^4)}{1.58b^4 - 0.211b^4}$$

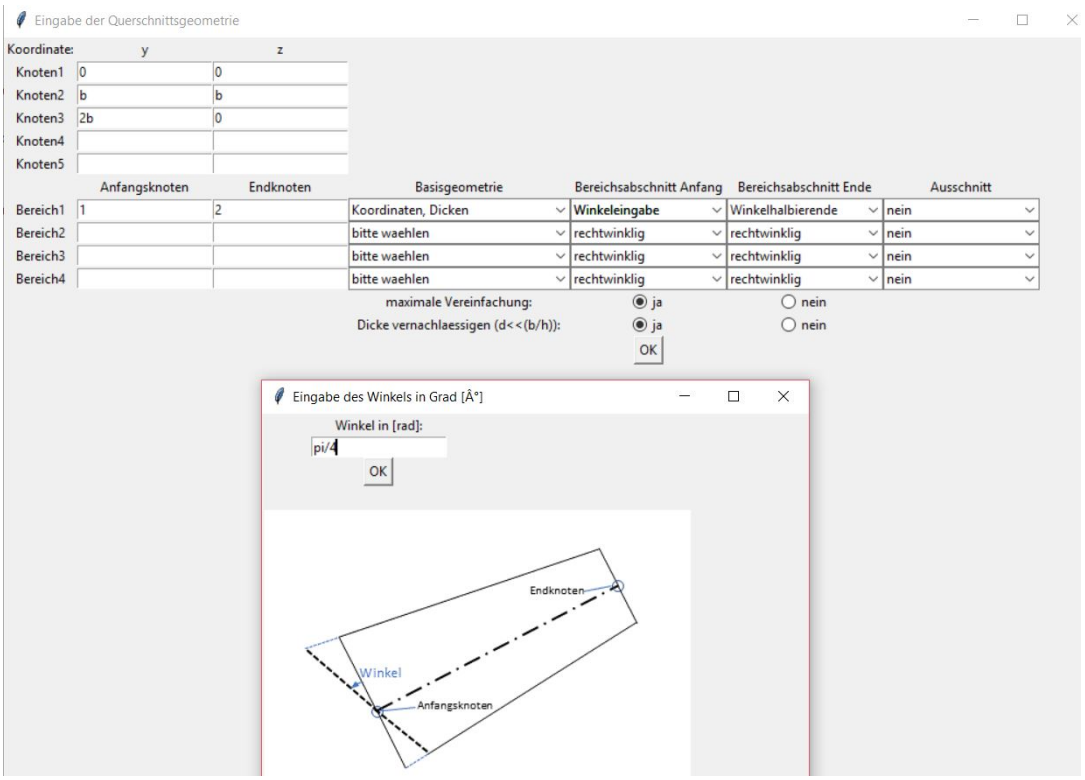
$$\phi = -\frac{1}{2}\arctan(0.08035) = -2.297^\circ$$

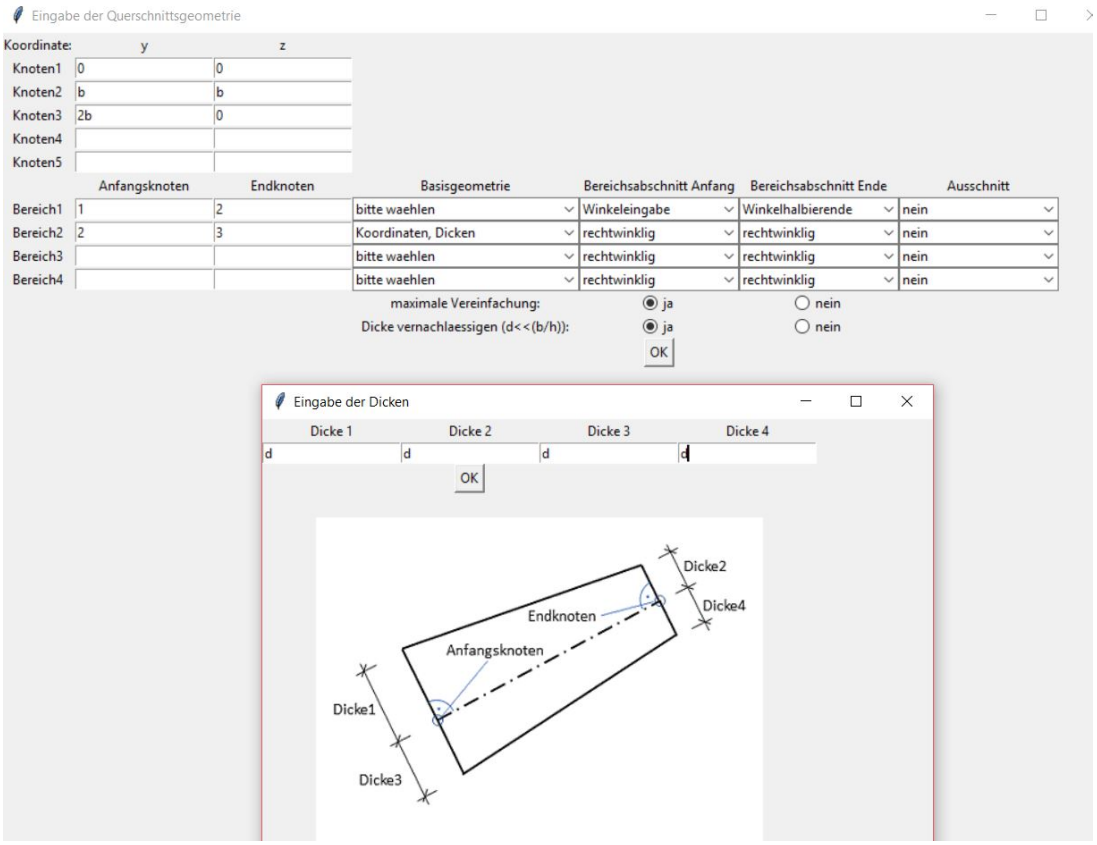
Somit stimmen die Ergebnisse der Handrechnung von Beispiel 2 mit jenen des Programmes überein.

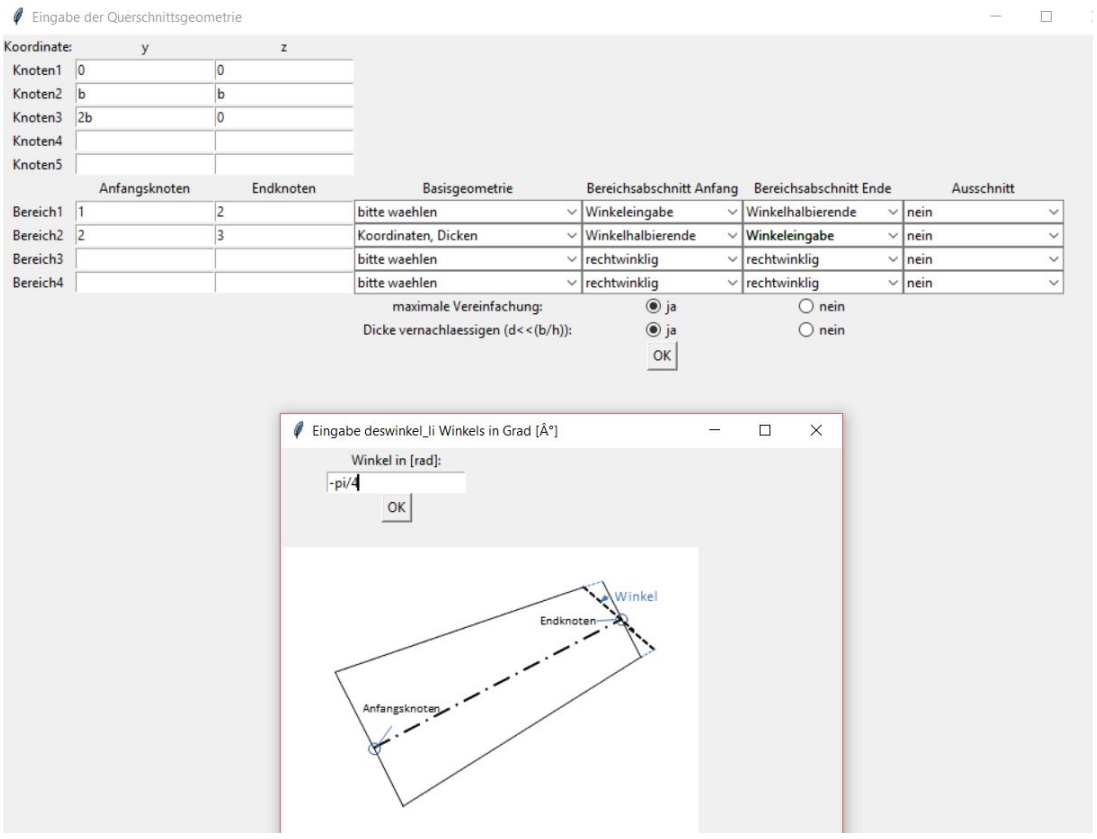
## 5.3 Beispiel 3

Dieses Beispiel soll die Funktion von “Koordinaten, Dicken“ in Kombination mit dem Abschnitt “Winkelhalbierende“ verdeutlichen. Dazu wird ein V-förmiges Profil vorgerechnet.

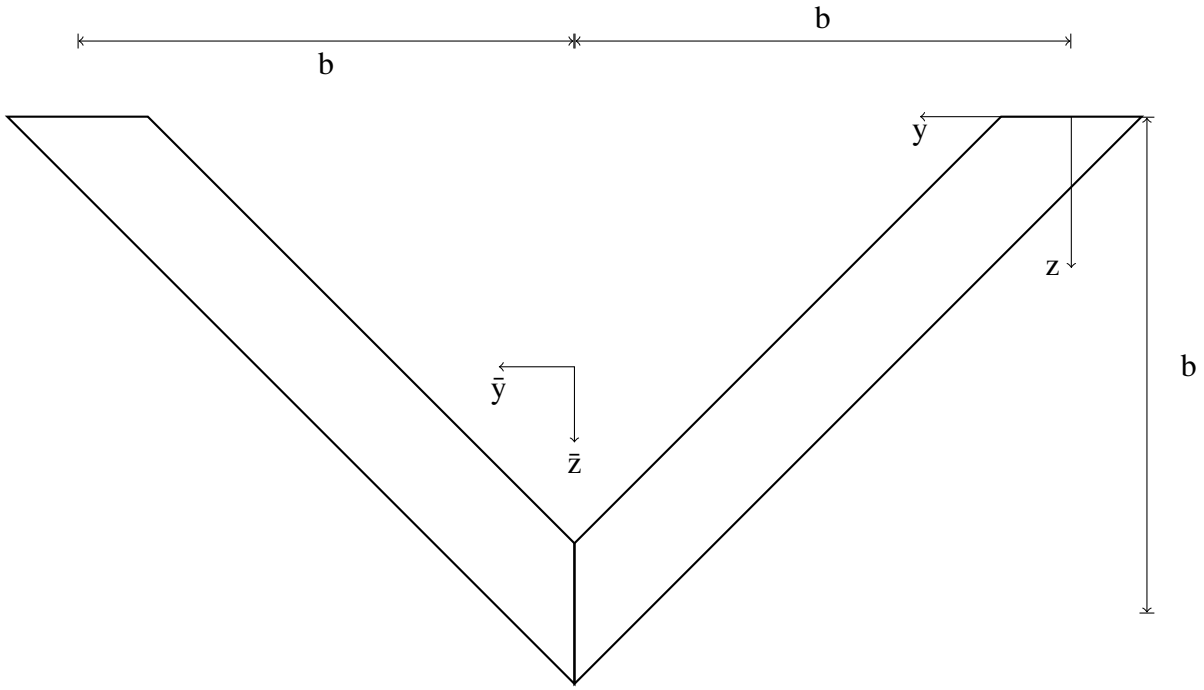



Eingabe der Querschnittsgeometrie
Koordinate: y z
Knoten1 0 0
Knoten2 b b
Knoten3 2b 0
Knoten4
Knoten5
Anfangsknoten Endknoten Basisgeometrie Bereichsabschnitt Anfang Bereichsabschnitt Ende Ausschnitt
Bereich1 1 2 bitte waehlen Winkeleingabe Winkelhalbierende nein
Bereich2 2 3 Koordinaten, Dicken rechtwinklig rechtwinklig nein
Bereich3 bitte waehlen rechtwinklig rechtwinklig nein
Bereich4 bitte waehlen rechtwinklig rechtwinklig nein
maximale Vereinfachung: ja nein
Dicke vernachlaessigen (d<<(b/h)): ja nein
OK
Eingabe der Dicken
Dicke 1 Dicke 2 Dicke 3 Dicke 4
d d d d
OK
Dicke2
Dicke4
Endknoten
Anfangsknoten
Dicke1
Dicke3


Eingabe der Querschnittsgeometrie
Koordinate: y z
Knoten1 0 0
Knoten2 b b
Knoten3 2b 0
Knoten4
Knoten5
Anfangsknoten Endknoten Basisgeometrie Bereichsabschnitt Anfang Bereichsabschnitt Ende Ausschnitt
Bereich1 1 2 bitte waehlen Winkeleingabe Winkelhalbierende nein
Bereich2 2 3 Koordinaten, Dicken Winkelhalbierende Winkeleingabe nein
Bereich3 bitte waehlen rechtwinklig rechtwinklig nein
Bereich4 bitte waehlen rechtwinklig rechtwinklig nein
maximale Vereinfachung: ja nein
Dicke vernachlaessigen (d<<(b/h)): ja nein
OK
Eingabe deswinkel_li Winkels in Grad [Â°]
Winkel in [rad]:
-pi/4
OK
Winkel
Endknoten
Anfangsknoten

### 5.3.1 Programmausgabe Beispiel 3




Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$
$$2.83bd+$$
$$2.83bd =$$
$$5.66bd\left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z\,\mathrm{d}A =$$
$$1.41b^2d+$$
$$1.41b^2d =$$
$$2.83b^2d\left[m^3\right]$$
$$S_z = \int_A y\,\mathrm{d}A =$$
$$1.41b^2d+$$
$$4.24b^2d =$$
$$5.66b^2d\left[m^3\right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 1.0b\left[m\right]$$

$$z_s = \frac{S_y}{A} = 0.5b\left[m\right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \,\mathrm{d}A =$$

$$0.236b^3d +$$

$$0.236b^3d =$$

$$0.471b^3d\left[m^4\right]$$

$$I_z = \int_A y^2 \,\mathrm{d}A =$$

$$0.943b^3d +$$

$$0.943b^3d =$$

$$1.89b^3d\left[m^4\right]$$

$$I_{yz} = \int_A yz \,\mathrm{d}A =$$

$$\left(0.236b^3d\right) +$$

$$\left(-0.236b^3d\right) =$$

$$0\left[m^4\right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$1.89b^3d\left[m^4\right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$0.471b^3d\left[m^4\right]$$

Die Hauptachse 1 entspricht Iz um 0 ° gedreht.

### 5.3.2 Handrechnung zu Beispiel 3

Berechnung unter Berücksichtigung von «d, d.h. Terme mit d, welche eine höhere Ordung als eins aufweisen, werden vernachlässigt.

Fläche:

$$A_1 = A_2 = 2\sqrt{2}bd$$

$$A = A_1 + A_2 = 4\sqrt{2}bd$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A$$

$$S_z = \int_A y \, \mathrm{d}A$$

$$S_{y1} = \int_{y=-d\sqrt{2}+z}^{d\sqrt{2}+z} \int_{z=0}^{b-\sqrt{2}d} z \, dzdy$$

$$S_{y1} = \sqrt{2}bd^2$$

$$S_{z1} = \int_{y=-d\sqrt{2}+z}^{d\sqrt{2}+z} \int_{z=0}^{b-\sqrt{2}d} y \, dzdy$$

$$S_{z1} = \sqrt{2}b^2d$$

$$S_{y2} = S_{y1} = \sqrt{2}bd^2$$

$$S_{z2} = \int_{y=b+z}^{b+2\sqrt{2}d+zb-\sqrt{2}d} \int_{z=0} y \, dzdy$$

$$S_{z2} = 3\sqrt{2}b^2d$$

Somit folgt $S_y$ als Summe von $S_{y1}$ + $S_{y2}$:

$$S_y = \sqrt{2}db^2 + \sqrt{2}db^2 = 2\sqrt{2}db^2$$

Analoges Vorgehen für $S_z$:

$$S_z = \sqrt{2}b^2d + 3\sqrt{2}b^2d = 4\sqrt{2}b^2d$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = \frac{4\sqrt{2}b^2 d}{4\sqrt{2}bd} = b$$

$$z_s = \frac{S_y}{A} = \frac{2\sqrt{2}db^2}{4\sqrt{2}bd} = \frac{b}{2}$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A$$

$$I_z = \int_A y^2 \, \mathrm{d}A$$

$$I_{yz} = \int_A yz \, \mathrm{d}A$$

Satz von Steiner:

$$I'_y = \int_A z^2 \, \mathrm{d}A + A(z'_s)^2$$

$$I'_z = \int_A y^2 \, \mathrm{d}A + A(y'_s)^2$$

$$I'_{yz} = \int_A yz \, \mathrm{d}A + A(y'_s z'_s)$$

Flächenträgheitsmoment

$$I_{y1} = \int_{y=-\sqrt{2}d+z-\frac{b}{2}}^{\sqrt{2}d+z-\frac{b}{2}} \int_{z=\frac{-b}{2}}^{\frac{b}{2}} z^2 \, dzdy$$

$$I_{y1} = I_{y2} = \frac{b^3 d\sqrt{2}}{6}$$

$$I_{z1} = \int_{y=-\sqrt{2}d+z-\frac{b}{2}}^{\sqrt{2}d+z-\frac{b}{2}} \int_{z=\frac{-b}{2}}^{\frac{b}{2}} y^2 \, dzdy$$

$$I_{z1} = I_{z2} = \frac{2b^3 d\sqrt{2}}{3}$$

$$I_{yz} = 0$$

Somit folgt $I_y$ als Summe von $I_{y1} + I_{y2}$:

$$I_y = \frac{b^3 d\sqrt{2}}{6} + \frac{b^3 d\sqrt{2}}{6} = \frac{b^3 d\sqrt{2}}{3}$$

Somit folgt $I_z$ als Summe von $I_{z1} + I_{z2}$:

$$I_z = \frac{2b^3 d\sqrt{2}}{3} + \frac{2b^3 d\sqrt{2}}{3} = \frac{4b^3 d\sqrt{2}}{3}$$

$$I_{yz} = 0$$

Hauptträgheitsmomente:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$

$$I^{(1,2)} = \frac{\frac{b^3 d\sqrt{2}}{3} + \frac{4b^3 d\sqrt{2}}{3}}{2} \pm \sqrt{\left(\frac{\frac{b^3 d\sqrt{2}}{3} - \frac{4b^3 d\sqrt{2}}{3}}{2}\right)^2 + 0}$$

$$I^{(1)} = 1.89 b^3 d$$

$$I^{(2)} = 0,471 b^3 d$$

Für den Drehwinkel gilt:

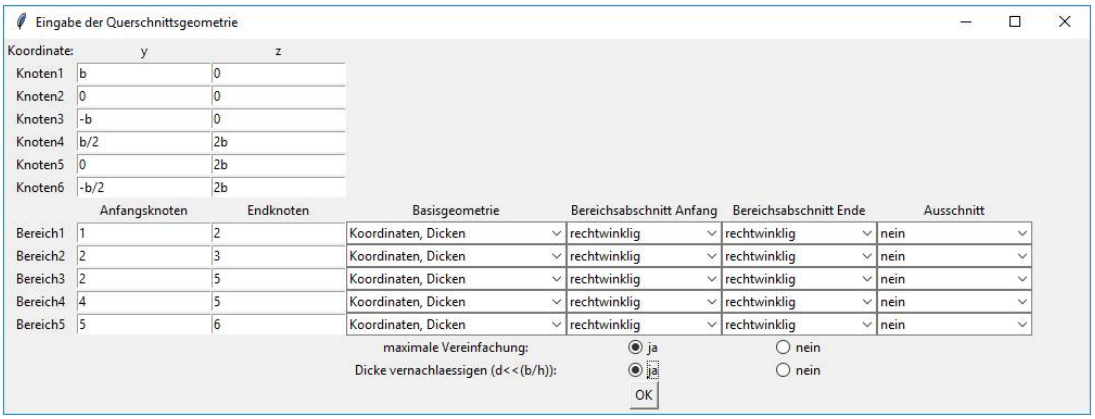
$$tan(2\phi) = -\frac{0}{1.89 b^3 d - 0,471 b^3 d}$$

$$\phi = -\frac{1}{2}\arctan(0) = 0^\circ$$

Somit stimmen die Ergebnisse der Handrechnung von Beispiel 3 mit jenen des Programmes überein.

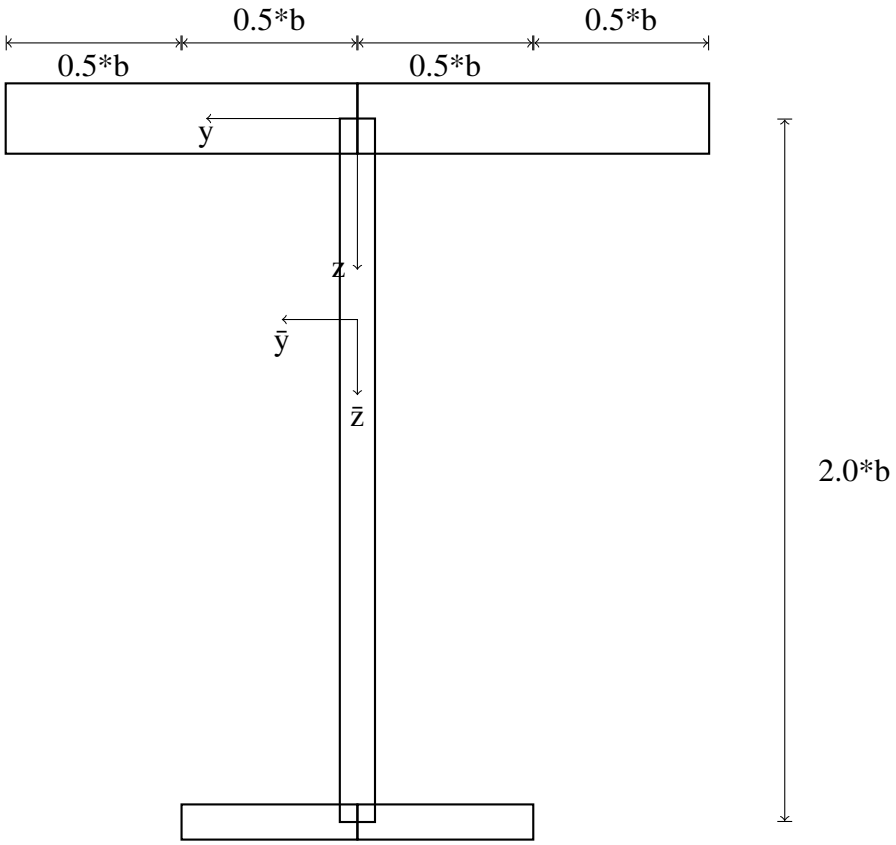
## 5.4 Beispiel 4

Bei Beispiel 4 handelt es sich um einen I-Träger mit unterschiedlich breiten und dicken Flanschen, wobei der Untergurt im Vergleich zum Obergurt schmäler und dünner ist. Die Dicke d kann allerdings für den gesamten Querschnitt mit der Eigenschaft d « b angesehen werden. Daher werden Terme die d mit Ordnung 2 oder höher enthalten nicht berücksichtigt.
Eine Besonderheit bei der Eingabe dieses Beispiels ist, dass die Flansche jeweils als zwei Bereiche eingegeben werden, obwohl sie mit konstanter Dicke durchlaufen. Grund dafür ist, dass somit der Steg leichter einzugeben ist. Auf der folgenden Abbildung ist die Eingabe dargestellt:




Der Steg verläuft also von Konten 2 zu Knoten 5. Die Dicke ist für die Bereiche 1 und 2 d und für die Bereiche 3 bis 5 $\frac{d}{2}$.

### 5.4.1 Programmausgabe Beispiel 4




Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$
$$2.0bd+$$
$$2.0bd+$$
$$2.0bd+$$
$$0.5bd+$$
$$0.5bd =$$
$$7.0bd\left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A =$$

$$\begin{aligned} 0+ \\ 0+ \\ 2.0b^2d+ \\ b^2d+ \\ b^2d = \end{aligned}$$

$$4.0b^2d \left[ m^3 \right]$$

$$S_z = \int_A y \, \mathrm{d}A =$$

$$\begin{aligned} b^2d+ \\ -b^2d+ \\ 0+ \\ 0.125b^2d+ \\ -0.125b^2d = \end{aligned}$$

$$0 \left[ m^3 \right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0 \left[ m \right]$$

$$z_s = \frac{S_y}{A} = 0.571b \left[ m \right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A =$$

$$\begin{aligned} 0.653b^3d+ \\ 0.653b^3d+ \\ 1.03b^3d+ \\ 1.02b^3d+ \\ 1.02b^3d = \end{aligned}$$

$$4.38b^3d \left[ m^4 \right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$\begin{aligned}
0.667b^3d+ \\
0.667b^3d+ \\
0+ \\
0.0417b^3d+ \\
0.0417b^3d = \\
1.42b^3d\left[m^4\right]
\end{aligned}$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$\begin{aligned}
\left(-0.571b^3d\right)+ \\
\left(0.571b^3d\right)+ \\
\left(0\right)+ \\
\left(0.179b^3d\right)+ \\
\left(-0.179b^3d\right) = \\
0\left[m^4\right]
\end{aligned}$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$4.38b^3d\left[m^4\right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$1.42b^3d\left[m^4\right]$$

Die Hauptachse 1 entspricht Iy um 0 ° gedreht.

### 5.4.2 Handrechnung zu Beispiel 4

Fläche:

$$A_1 = 2d2b = 4bd$$

$$A_2 = 2db = 2bd$$

$$A_3 = bd = bd$$

$$A = A_1 + A_2 + A_3 = 7bd$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A$$

$$S_{y1} = \int_{y=-b}^{b} \int_{z=0}^{2d} z \, dzdy$$

Lösen des Integrals ergibt mit Berücksichtigen von d « b:

$$S_{y1} = 4bd^2$$

Mit $d^2$ fällt $S_{y1}$ also weg.

$$S_{y2} = \int_{y=-\frac{d}{2}}^{\frac{d}{2}} \int_{z=0}^{2b} z \, dzdy$$

Lösen des Integrals ergibt:

$$S_{y2} = 2b^2d$$

$$S_{y3} = \int_{y=-\frac{b}{2}}^{\frac{b}{2}} \int_{z=2b}^{2b+d} z \, dzdy$$

Lösen des Integrals ergibt mit Berücksichtigen von d « b:

$$S_{y3} = 2b^2d$$

Somit folgt $S_y$ als Summe von $S_{y1}$ + $S_{y2}$ + $S_{y3}$:

$$S_y = 4b^2d$$

Aufgrund der Symmetrie ist die y-Schwerpunktskoordinate 0 und somit gilt: $S_z$=0.

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0$$

$$z_s = \frac{S_y}{A} = \frac{4b^2d}{7bd} = \frac{4}{7}b = 0,5714b$$

Flächenträgheitsmomente:
Die Flächenträgheitsmomente setzten sich aus den Flächenträgheitsmomenten der Einzelbereiche und den zugehörigen Steineranteilen zusammen.

$$I_y = \int_A z^2 \, \mathrm{d}A$$

$$I_z = \int_A y^2 \, \mathrm{d}A$$

$$I_{yz} = \int_A yz \, \mathrm{d}A$$

Satz von Steiner:

$$I'_y = \int_A z^2 \, \mathrm{d}A + A(z'_s)^2$$

$$I'_z = \int_A y^2 \, \mathrm{d}A + A(y'_s)^2$$

$$I'_{yz} = \int_A yz \, \mathrm{d}A + A(y'_s z'_s)$$

Flächenträgheitsmoment eines Rechtecks:

$$I = \frac{bh^3}{12}$$

Mit diesen Formeln folgt für $I_y$, $I_z$ und $I_{yz}$ unter Berücksichtigen von d « b:

$$I_{y1} = 0 + 4bd(d + 0,5714b)^2 = 1,30599b^3d$$

$$I_{y2} = \frac{2}{3}b^3d + 0,367b^3d = 1,0337b^3d$$

$$I_{y3} = 0 + bd * \left(1,4286b - \frac{d}{2}\right)^2 = 2,04089b^3d$$

Somit folgt $I_y$ als Summe von $I_{y1}$ + $I_{y2}$ + $I_{y3}$:

$$I_y = 4,3806b^3d$$

Für $I_z$ ist der Steinteranteil 0, da $y_s$=0 ist.

$$I_{z1} = \frac{(2b)^3 2d}{12} = 1,333333b^3d$$

$I_{z2}$ ist wegen $\frac{d^3 2b}{12}$ gleich 0.

$$I_{z3} = \frac{b^3d}{12} = 0,0833333b^3d$$

Somit folgt $I_z$ als Summe von $I_{z1}$ + $I_{z2}$ + $I_{z3}$:

$$I_z = 1,4167b^3d$$

Die Haupträgheitsmomente der Rechtecke sind 0, weil die Teilschwerachsen Hautachsen sind. Weiters ist aufgrund von $y_s$=0 der Steineranteil ebenfalls 0 womit für $I_{yz}$=0 folgt.
Hautträgheitsmomente:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$

$$I^{(1)} = 4,3806 b^3 d$$

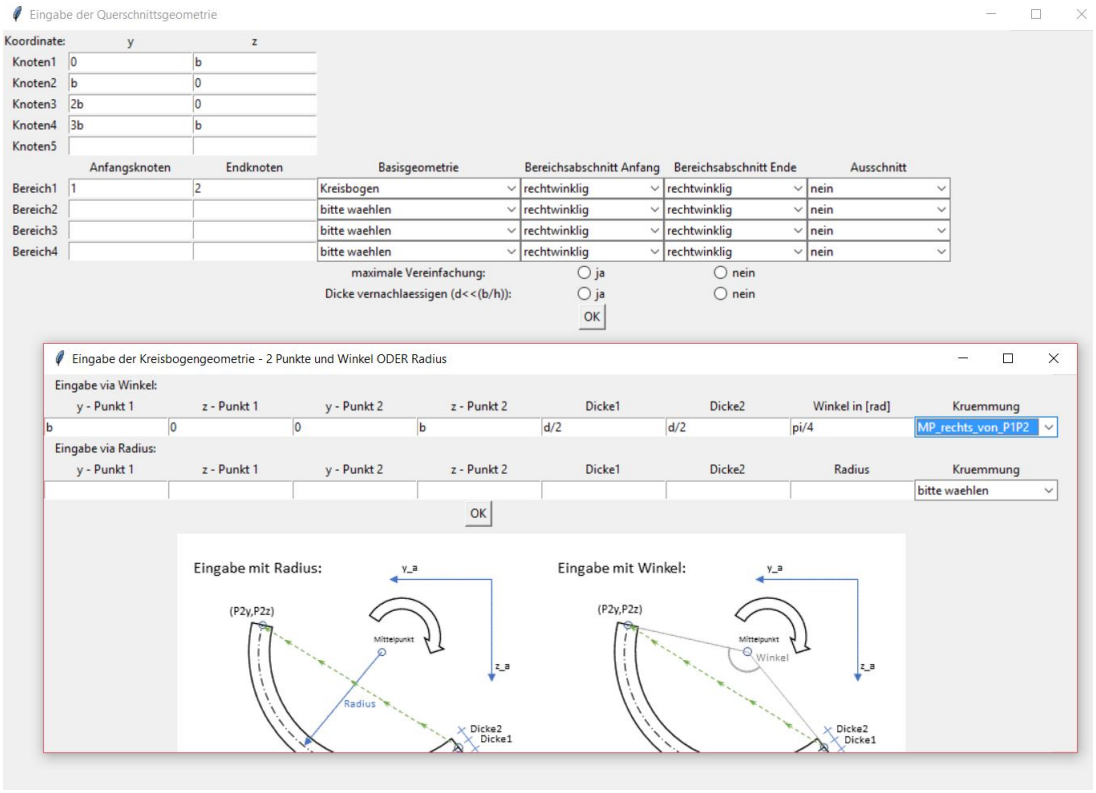
$$I^{(2)} = 1,4167 b^3 d$$

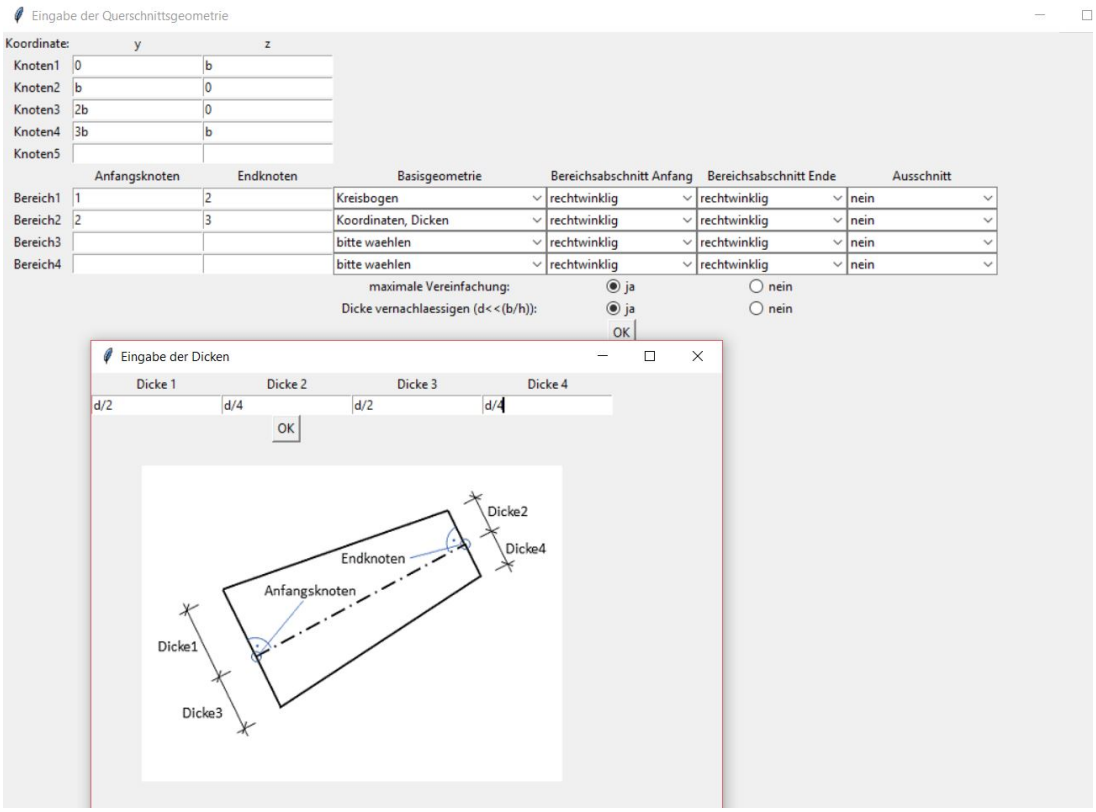
Mit arctan(0)=0 folgt für den Drehwinkel der Hautachsen: $\phi = 0°$

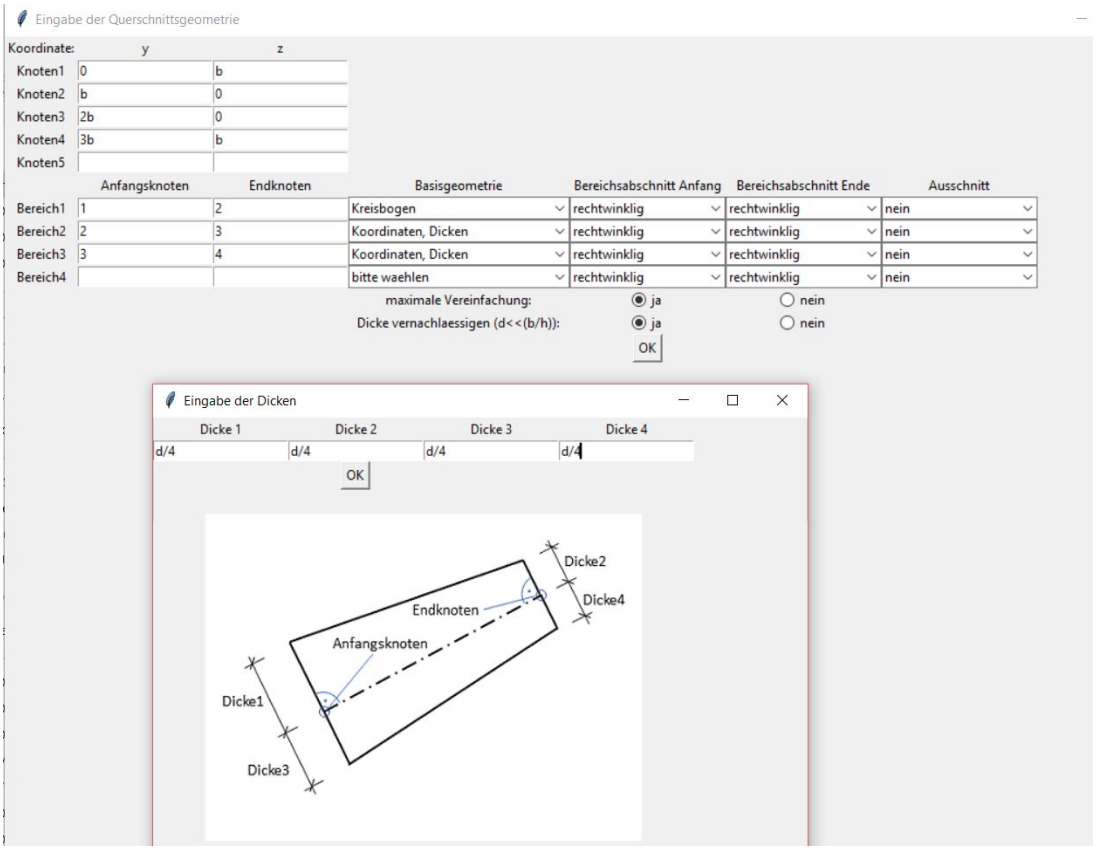
Somit stimmen die Ergebnisse der Handrechnung von Beispiel 4 mit jenen des Programmes überein.

## 5.5 Beispiel 5

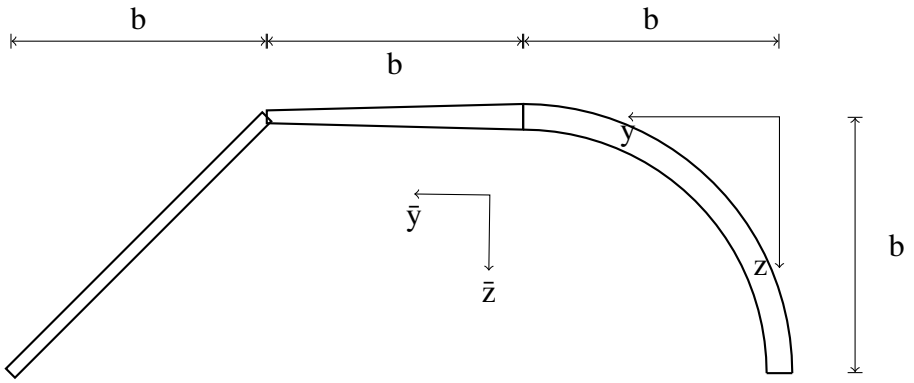
Bei diesem Beispiel handelt es sich um einen Querschnitt, welcher sich aus einem Bogen, einem Trapez und einem Rechteck zusammensetzt.






### 5.5.1 Programmausgabe Beispiel 5




Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$

$$1.57bd+$$
$$0.75bd+$$
$$0.707bd =$$
$$3.03bd\left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A =$$
$$0.571 b^2 d +$$
$$0 +$$
$$0.354 b^2 d =$$
$$0.924 b^2 d \left[ m^3 \right]$$

$$S_z = \int_A y \, \mathrm{d}A =$$
$$0.571 b^2 d +$$
$$1.08 b^2 d +$$
$$1.77 b^2 d =$$
$$3.42 b^2 d \left[ m^3 \right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 1.13 b \left[ m \right]$$
$$z_s = \frac{S_y}{A} = 0.305 b \left[ m \right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A =$$
$$0.154 b^3 d +$$
$$0.0699 b^3 d +$$
$$0.0857 b^3 d =$$
$$0.31 b^3 d \left[ m^4 \right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$
$$1.07 b^3 d +$$
$$0.134 b^3 d +$$
$$1.39 b^3 d =$$
$$2.59 b^3 d \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$\left( -0.207 b^3 d \right) +$$

$$\left( -0.072 b^3 d \right) +$$

$$\left( 0.248 b^3 d \right) =$$

$$-0.031 b^3 d \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$2.59 b^3 d \left[ m^4 \right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$0.31 b^3 d \left[ m^4 \right]$$

Die Hauptachse 1 entspricht Iz um $-0.778\ ^\circ$ gedreht.

### 5.5.2 Handrechnung zu Beispiel 5

Berechnung unter Berücksichtigung von «d, d.h. Terme mit d, welche eine höhere Ordung als eins aufweisen, werden vernachlässigt.

Fläche Bogen:

$$A_1 = \int_{\phi=0}^{\frac{\pi}{2}} \int_{r_2=r-\frac{d}{2}}^{r+\frac{d}{2}} r \, dr d\phi$$

$$A_1 = \frac{bd\pi}{2}$$

Statische Momente Bogen:

$$S_{y1} = \int_{\phi=0}^{\frac{\pi}{2}} \int_{r=b-\frac{d}{2}}^{b+\frac{d}{2}} r^2 \sin(\phi) \, dr d\phi$$

$$S_{z1} = \int_{\phi=0}^{\frac{\pi}{2}} \int_{r=b-\frac{d}{2}}^{b+\frac{d}{2}} r^2 \cos(\phi) \, dr d\phi$$

$$S_{y1} = S_{z1} = \frac{3b^2 d}{3}$$

Schwerpunkt Bogen:

$$y_{s1} = \frac{S_z}{A} = \frac{b^2 d}{\frac{bd\pi}{2}} = \frac{2b}{\pi}$$

$$z_{s1} = \frac{S_y}{A} = \frac{b^2 d}{\frac{bd\pi}{2}} = \frac{2b}{\pi}$$

Fläche Trapez:

$$A_2 = \frac{1}{2}\left(d + \left(\frac{d}{2}\right)\right)b = \frac{3}{4}bd$$

Statische Momente Trapez:

$$S_{y2} = 0$$

$$S_{z2} = \int_{y=0}^{b} \int_{z=-\frac{d}{2}+\frac{d}{4b}y}^{\frac{d}{2}-\frac{d}{4b}y} y \, dydz$$

$$S_{z2} = \frac{b^2 d}{3}$$

Schwerpunkt Trapez:

$$y_{s2} = \frac{S_z}{A} = \frac{\frac{db^2}{3}}{\frac{3}{4}bd} = \frac{4b}{9}$$

$$z_{s2} = \frac{S_y}{A} = \frac{0}{\frac{3}{4}bd} = 0$$

Fläche Rechteck:

$$A_3 = \sqrt{2}b\frac{d}{2}$$

Statische Momente Rechteck:

$$S_{y3} = 0$$

$$S_{z3} = \sqrt{2}b^2\frac{d}{4}$$

Schwerpunkt Rechteck:

$$y_{s3} = \frac{S_z}{A} = \frac{\sqrt{2}b^2\frac{d}{4}}{\sqrt{2}b\frac{d}{2}} = \frac{b}{2}$$

$$z_{s3} = \frac{S_z}{A} = \frac{0}{\sqrt{2}b\frac{d}{2}} = 0$$

Gesamtfläche:

$$A = A_1 + A_2 + A_3 = \frac{bd\pi}{2} + \frac{3}{4}bd + \sqrt{2}b\frac{d}{2} = 3.03bd$$

Gesamtschwerpunkt:

$$y_s = \frac{y_{s1}A_1 + y_{s2}A_2 + y_{s3}A_3}{A}$$

$$y_s = \frac{\frac{2b}{\pi}\frac{bd\pi}{2} - \frac{3bd}{4}\frac{3}{4}bd - \frac{3b}{2}\sqrt{2}b\frac{d}{2}}{3.03bd} = -0.13b$$

Im Koordinatensystem der Berechnung des Programms würde dies dem Wert 1.13b entsprechen.

$$z_s = \frac{\frac{2b}{\pi}\frac{bd\pi}{2} + \frac{3bd}{4}b + \frac{bd}{\sqrt{2}}\frac{b}{2}}{3.03bd} = 0.695b$$

Im Koordinatensystem der Berechnung des Programms würde dies dem Wert 0.305b entsprechen.

Flächenträgheitsmomente Bogen:

$$I_{ys1} = \int_{\phi=0}^{\frac{\pi}{2}} \int_{r=b-\frac{d}{2}}^{b+\frac{d}{2}} r^3 \sin(\phi)^2 \, dr d\phi - \left(\frac{2a}{\pi}\right)^2 \frac{bd\pi}{2}$$

$$I_{zs1} = \int_{\phi=0}^{\frac{\pi}{2}} \int_{r=b-\frac{d}{2}}^{b+\frac{d}{2}} r^3 cos(\phi)^2 \, dr d\phi - \left(\frac{2a}{\pi}\right)^2 \frac{bd\pi}{2}$$

$$I_{ys1} = I_{zs1} = \frac{\pi b^3 d}{4} - \frac{2b^3 d}{\pi}$$

$$I_{yzs1} = \int_{\phi=0}^{\frac{\pi}{2}} \int_{r_2=r-\frac{d}{2}}^{r+\frac{d}{2}} r^3 \sin(\phi)\sin(\phi) \, dr d\phi - \left(\frac{2b}{\pi}\right)^2 \frac{bd\pi}{2}$$

$$I_{yzs1} = b^3 d\left(\frac{1}{2} - \frac{2}{\pi}\right)$$

Flächenträgheitsmomente Trapez:

$$I_{ys2} = \int_{y=\frac{-5b}{9}}^{\frac{4b}{9}} \int_{z=-\frac{7d}{18}-\frac{d}{4b}y}^{\frac{7d}{18}+\frac{d}{4b}y} z^2 \, dydz$$

$$I_{ys2} = 0$$

$$I_{zs2} = \int_{y=\frac{-5b}{9}}^{\frac{4b}{9}} \int_{z=-\frac{7d}{18}-\frac{d}{4b}y}^{\frac{7d}{18}+\frac{d}{4b}y} y^2 \, dydz$$

$$I_{zs2} = \frac{13b^3d}{216}$$

$$I_{yzs2} = \int_{y=\frac{-5b}{9}}^{\frac{4b}{9}} \int_{z=-\frac{7d}{18}-\frac{d}{4b}y}^{\frac{7d}{18}+\frac{d}{4b}y} yz \, dydz$$

$$I_{yzs2} = 0$$

Flächenträgheitsmomente Rechteck:

$$I_{ys3} = \int_{y=-\frac{b}{2}}^{\frac{b}{2}} \int_{z=-\frac{d}{4}\sqrt{2}+y}^{\frac{d}{4}\sqrt{2}+y} z^2 \, dydz$$

$$I_{ys2} = \frac{b^3d}{12\sqrt{2}}$$

$$I_{zs3} = \int_{y=-\frac{b}{2}}^{\frac{b}{2}} \int_{z=-\frac{d}{4}\sqrt{2}+y}^{\frac{d}{4}\sqrt{2}+y} y^2 \, dydz$$

$$I_{ys2} = \frac{b^3d}{12\sqrt{2}}$$

$$I_{yzs3} = \int_{y=-\frac{b}{2}}^{\frac{b}{2}} \int_{z=-\frac{d}{4}\sqrt{2}+y}^{\frac{d}{4}\sqrt{2}+y} yz \, dydz$$

$$I_{ys2} = \frac{b^3d}{12\sqrt{2}}$$

Teilflächtenträgheitsmomente bezogen auf den Gesamtschwerpunkt:
Bogen:

$$I_{y1} = b^3 d\left(\frac{\pi}{4} - \frac{2}{\pi}\right) + \left(0.694b - \frac{2b}{\pi}\right)^2 \frac{b * d\pi}{2} = 0.154 b^3 d$$

$$I_{z1} = b^3 d\left(\frac{\pi}{4} - \frac{2}{\pi}\right) + \left(-\frac{2b}{\pi} - 0.130b\right)^2 \frac{b * d\pi}{2} = 1.072 b^3 d$$

$$I_{yz1} = b^3 d\left(\frac{\pi}{4} - \frac{2}{\pi}\right) + \left(-\frac{2b}{\pi} - 0.130b\right)\left(0.694b - \frac{2b}{\pi}\right)\frac{bd\pi}{2} = 1.072 b^3 d$$

Trapez:

$$I_{y2} = 0 + (-b + 0.694b)^2 \frac{3}{4} bd = 0.0699 b^3 d$$

$$I_{z2} = \frac{13 b^3 d}{216} + \left(\frac{4}{9}b - 0.130b\right)^2 \frac{3}{4} bd = 0.133 b^3 d$$

$$I_{yz2} = 0 + \left(\frac{4}{9}b - 0.130b\right)\left(\frac{4}{9}b - 0.130b\right)\frac{3}{4} bd = -0.0719 b^3 d$$

Rechteck:

$$I_{y3} = \frac{b^3 d}{12\sqrt{2}} + \left(0.694b - \frac{a}{2}\right)^2 \frac{bd}{\sqrt{2}} = 0.0857 b^3 d$$

$$I_{z3} = \frac{b^3 d}{12\sqrt{2}} + \left(\frac{3a}{2} - 0.130b\right)^2 \frac{bd}{\sqrt{2}} = 1.386 b^3 d$$

$$I_{yz3} = \frac{b^3 d}{12\sqrt{2}} + \left(0.694b - \frac{a}{2}\right)\left(\frac{3a}{2} - 0.130b\right)\frac{bd}{\sqrt{2}} = 0.248 b^3 d$$

Gesamtes Teilflächtenträgheitsmoment:

$$I_y = I_{y1} + I_{y2} + I_{y3} = b^3 d(0.154 + 0.0699 + 0.0857) = 0.3097 b^3 d$$

$$I_z = I_{z1} + I_{z2} + I_{z3} = b^3 d(1.0722 + 0.1343 + 1.3859) = 2.592 b^3 d$$

$$I_{yz} = I_{yz1} + I_{yz2} + I_{yz3} = b^3 d(-0.2066 - 0.07197 + 0.2475) = (-)0.0312 b^3 d$$

Hautträgheitsmomente:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$

$$I^{(1,2)} = \frac{0.3097b^3b + 2.592b^3d}{2} \pm \sqrt{\left(\frac{0.3097b^3b - 2.592b^3d}{2}\right)^2 + (0.0312b^3d)^2}$$

$$I^{(1)} = 2.593b^3d$$

$$I^{(2)} = 0.309b^3d$$

Für den Drehwinkel gilt:

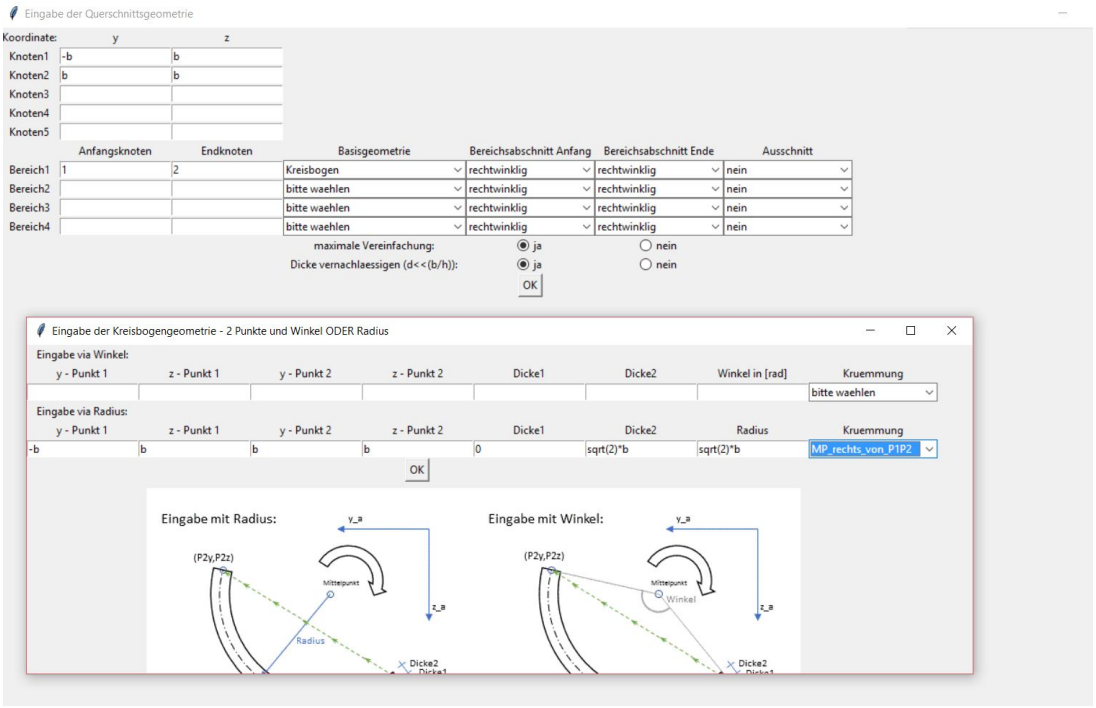
$$tan(2\phi) = -\frac{2(0.0312b^3d)}{0.3097b^3d - 2.592b^3d}$$

$$\phi = -\frac{1}{2}\arctan(0.0273) = -0.778^\circ$$

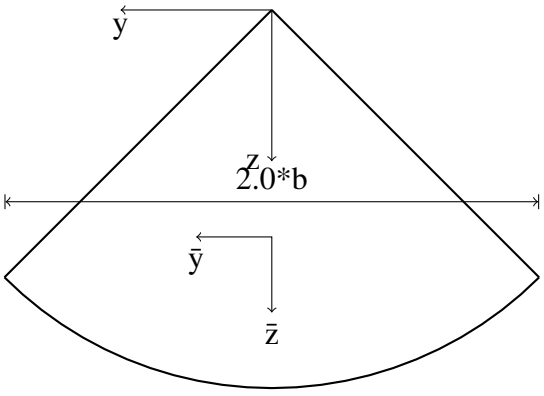
Somit stimmen die Ergebnisse der Handrechnung von Beispiel 5 mit jenen des Programmes überein.

## 5.6 Beispiel 6

Anhand von Beispiel 6 soll die Eingabemögleichkeit von Kreissegmenten vorgeführt werden. Der folgende Querschnitt besteht deshalb aus einem Viertelkreis.




### 5.6.1 Programmausgabe Beispiel 6




Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$

$$1.57b^2 \left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A =$$

$$1.33 b^3 \left[ m^3 \right]$$

$$S_z = \int_A y \, \mathrm{d}A =$$

$$0 \left[ m^3 \right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0 \left[ m \right]$$

$$z_s = \frac{S_y}{A} = 0.849 b \left[ m \right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A =$$

$$0.154 b^4 \left[ m^4 \right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$0.285 b^4 \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$0 \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$0.285 b^4 \left[ m^4 \right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$0.154 b^4 \left[ m^4 \right]$$

Die Hauptachse 1 entspricht Iz um 0 ° gedreht.

### 5.6.2 Handrechnung Beispiel 6

Fläche:

$$A = \frac{(\sqrt{2}b)^2\pi}{4} = \frac{b^2\pi}{2} = 1.57b^2$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A$$

$$S_z = \int_A y \, \mathrm{d}A$$

$$S_y = \int_{\phi=-\pi/4}^{\pi/4} \int_{r=0}^{\sqrt{2}b} r^2 \cos(\phi) \, dr d\phi = \frac{4b^3}{3}$$

$$S_z = \int_{\phi=-\pi/4}^{\pi/4} \int_{r=0}^{\sqrt{2}b} r^2 \sin(\phi) \, dr d\phi = 0$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = \frac{0}{\frac{b^2 * \pi}{2}} = 0$$

$$z_s = \frac{S_y}{A} = \frac{\frac{4b^3}{3}}{\frac{b^2 * \pi}{2}} = \frac{8}{3\pi} = 0.849b$$

Flächenträgheitsmoment:

$$I_y = \int_A z^2 \, \mathrm{d}A$$

$$I_z = \int_A y^2 \, \mathrm{d}A$$

$$I_{yz} = \int_A yz \, \mathrm{d}A$$

Satz von Steiner:

$$I'_y = \int_A z^2 \, \mathrm{d}A + A(z'_s)^2$$

$$I'_z = \int_A y^2 \, \mathrm{d}A + A(y'_s)^2$$

$$I'_{yz} = \int_A yz \, \mathrm{d}A + A(y'_s z'_s)$$

Einsetzen in die Formeln ergibt:

$$I_y = \int_{\phi=-\pi/4}^{\pi/4} \int_{r=0}^{\sqrt{2}b} r^3 \cos(\phi)^2 \, dr d\phi - (0.849b)^2 \frac{b^2\pi}{2} = 0.154b^4$$

$$I_z = \int_{\phi=-\pi/4}^{\pi/4} \int_{r=0}^{\sqrt{2}b} r^3 \sin(\phi)^2 \, dr d\phi = 0.285b^4$$

$$I_{yz} = \int_{\phi=-\pi/4}^{\pi/4} \int_{r=0}^{\sqrt{2}b} r^3 \sin(\phi)\cos\phi \, dr d\phi = 0$$

Für die Hauptträgheitsmomente gilt:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$

$$I^{(1,2)} = \frac{0.154b^4 + 0.285b^4}{2} \pm \sqrt{\left(\frac{0.154b^4 - 0.285b^4}{2}\right)^2 + 0^2}$$

$$I^{(1)} = I_z = 0.285b^4$$
$$I^{(2)} = I_y = 0.154b^4$$

Für den Drehwinkel gilt:

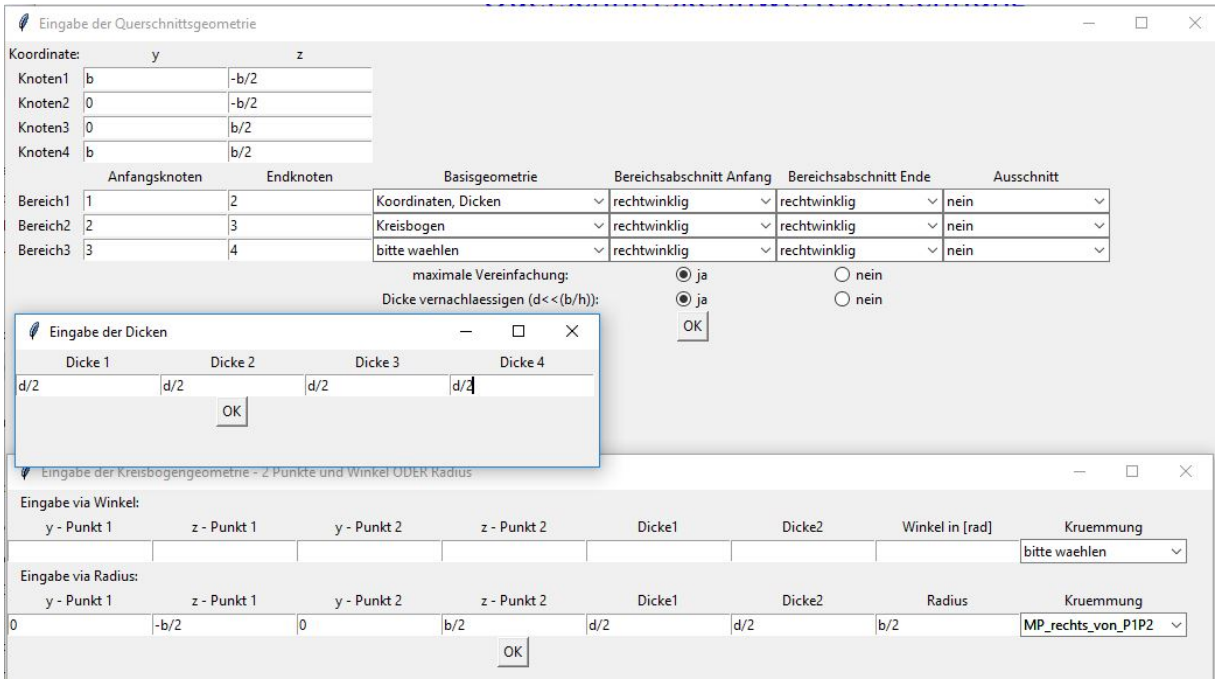
$$tan(2\phi) = -\frac{2I_{yz}}{I_y - Iz}$$

Da $I_{yz} = 0$ gilt, ergibt sich der Drehwinkel $\phi$ zu $\frac{arctan(0)}{2} = 0$.

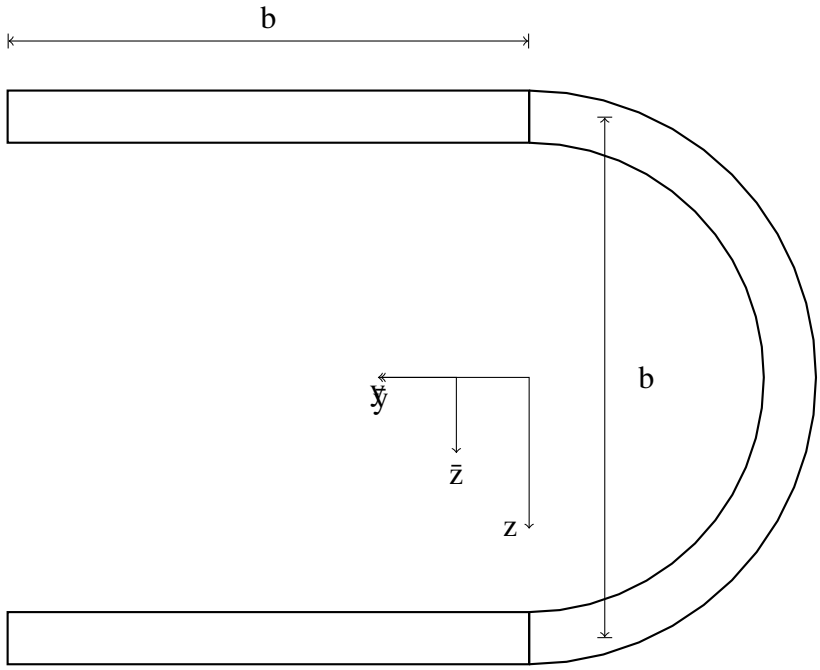
Somit stimmen die Ergebnisse der Handrechnung von Beispiel 6 mit jenen des Programmes überein.

## 5.7 Beispiel 7

Bei Beispiel 7 handelt es sich um einen Querschnitt, der aus zwei Rechtecken und einem Halbkreisbogen besteht. Die Dicke ist über den Querschnitt konstant d. Aufgrund von d « b werden Terme die d mit Ordnung 2 oder höher enthalten nicht berücksichtigt. Auf der folgenden Abbildung ist die Eingabe dargestellt:




### 5.7.1 Programmausgabe Beispiel 7



Querschnittskennwerte

Fläche:

$$A = \int_A \mathrm{d}A =$$
$$bd +$$
$$1.57bd +$$
$$bd =$$
$$3.57bd \left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z\,\mathrm{d}A =$$
$$-0.5b^2d +$$
$$0 +$$
$$0.5b^2d =$$
$$0 \left[m^3\right]$$

$$S_z = \int_A y\,\mathrm{d}A =$$
$$0.5b^2d +$$
$$-0.5b^2d +$$
$$0.5b^2d =$$
$$0.5b^2d \left[m^3\right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0.14b \left[m\right]$$
$$z_s = \frac{S_y}{A} = 0 \left[m\right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2\,\mathrm{d}A =$$
$$0.25b^3d +$$
$$0.196b^3d +$$
$$0.25b^3d =$$
$$0.696b^3d \left[m^4\right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$0.213 b^3 d +$$

$$0.367 b^3 d +$$

$$0.213 b^3 d =$$

$$0.793 b^3 d \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$\left( -0.18 b^3 d \right) +$$

$$\left( 0 \right) +$$

$$\left( 0.18 b^3 d \right) =$$

$$0 \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$0.793 b^3 d \left[ m^4 \right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$0.696 b^3 d \left[ m^4 \right]$$

Die Hauptachse 1 entspricht Iz um 0 ° gedreht.

### 5.7.2 Handrechnung Beispiel 7

Zur Berechnung der Integrale beim Flächenträgheitsmoment wurden Integraltabellen verwendet, die aus dem Taschenbuch Mathematischer Formeln für Ingeniuere und Naturwissenschaftler von H.J.Bartsch (23. Auflage) [3] stammen.

Fläche:

$$A_1 = A_3 = bd$$

$$A_2 = \frac{r^2\pi}{2} = \left[\left(\frac{b}{2}+\frac{d}{2}\right)^2 - \left(\frac{b}{2}-\frac{d}{2}\right)^2\right]\frac{\pi}{2} = \frac{bd\pi}{2}$$

$$A = A_1 + A_2 + A_3 = 3,5708bd$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A$$

$$S_{y1} = \int_{y=0}^{b} \int_{z=-(\frac{b}{2}+\frac{d}{2})}^{-(\frac{b}{2}-\frac{d}{2})} z \, dzdy$$

$$S_{y1} = -\frac{b^2 d}{2}.$$

$$S_{y2} = \int_{R=\frac{b}{2}-\frac{d}{2}}^{\frac{b}{2}+\frac{d}{2}} \int_{\phi=0}^{\pi} Rcos(\phi)\, Rd\phi dR$$

$$S_{y2} = 0$$

$$S_{y3} = \int_{y=0}^{b} \int_{z=\frac{b}{2}-\frac{d}{2}}^{\frac{b}{2}+\frac{d}{2}} z \, dzdy$$

$$S_{y3} = \frac{b^2 d}{2}.$$

Somit folgt $S_y$ als Summe von $S_{y1}$ + $S_{y2}$ + $S_{y3}$:

$$S_y = 0$$

$$S_z = \int_A y \, \mathrm{d}A$$

$$S_{z1} = \int_{z=-(\frac{b}{2}+\frac{d}{2})}^{-(\frac{b}{2}-\frac{d}{2})} \int_{y=0}^{b} y \, dydz$$

$$S_{z1} = \frac{b^2 d}{2}.$$

$$S_{z2} = \int_{R=\frac{b}{2}-\frac{d}{2}}^{\frac{b}{2}+\frac{d}{2}} \int_{\phi=0}^{\pi} Rcos(\phi) \, Rd\phi dR$$

$$S_{z2} = -\frac{b^2 d}{2}.$$

$$S_{z3} = \int_{y=0}^{b} \int_{z=\frac{b}{2}-\frac{d}{2}}^{\frac{b}{2}+\frac{d}{2}} z \, dzdy$$

$$S_{z3} = \frac{b^2 d}{2}.$$

Somit folgt $S_z$ als Summe von $S_{z1}$ + $S_{z2}$ + $S_{z3}$:

$$S_z = \frac{b^2 d}{2}.$$

Somit folgt für die Schwerpunktskoordinaten:

$$y_s = \frac{S_z}{A} = 0.1400b$$

$$z_s = \frac{S_y}{A} = 0$$

Flächenträgheitsmomente:
Die Flächenträgheitsmomente setzten sich aus den Flächenträgheitsmomenten der Einzelbereiche und den zugehörigen Steineranteilen zusammen. Für die trigonometrischen Terme, wurden, wie bereits erwähnt, Integralsätze herangezogen:

$$\int cos^2(ax)dx = \frac{x}{2} + \frac{1}{4a} sin(2ax)$$

$$\int sin^2(ax)dx = \frac{x}{2} - \frac{1}{4a} sin(2ax)$$

$$\int sin(ax)cos(ax)dx = \frac{1}{2a} sin^2(ax)$$

Wobei in diesem Fall für a=1 und für x=$\phi$ gilt.

$$I_y = \int_A z^2 \, \mathrm{d}A$$

$$I_z = \int_A y^2 \, \mathrm{d}A$$

$$I_{yz} = \int_A yz \, \mathrm{d}A$$

Satz von Steiner:

$$I'_y = \int_A z^2 \, \mathrm{d}A + A(z'_s)^2$$

$$I'_z = \int_A y^2 \, \mathrm{d}A + A(y'_s)^2$$

$$I'_{yz} = \int_A yz \, \mathrm{d}A + A(y'_s z'_s)$$

Flächenträgheitsmoment eines Rechtecks:

$$I = \frac{bh^3}{12}$$

Mit diesen Formeln folgt für $I_y$, $I_z$ und $I_{yz}$:

$I_y$ der Rechtecke ist bei Vernachlässigung der Dicke 0. Daher gehen hier nur die Steineranteile ein:

$$I_{y1} = 0 + bd\left(\frac{b}{2}\right)^2 = 0,25b^3d$$

$$I_{y2} = \int_{R=\frac{b}{2}-\frac{d}{2}}^{\frac{b}{2}+\frac{d}{2}} \int_{\phi=0}^{\pi} R^2 cos^2(\phi) \, Rd\phi dR = ... = \frac{b^3 d\pi}{16} = 0,1963b^3d$$

$$I_{y3} = 0 + bd\left(-\frac{b}{2}\right)^2 = 0,25b^3d$$

Somit folgt $I_y$ als Summe von $I_{y1}$ + $I_{y2}$ + $I_{y3}$:

$$I_y = 0,696b^3d$$

$$I_{z1} = I_{z3} = \frac{b^3 d}{12} + bd * \left[ \left( \frac{1}{2} - 0,14 \right) b \right]^2 = 0,2129 b^3 d$$

Teilflächenträgheitsmoment des Halbkreisbogens:

$$I_{z2} = \int_{R=\frac{b}{2}-\frac{d}{2}}^{\frac{b}{2}+\frac{d}{2}} \int_{\phi=0}^{\pi} R^2 sin^2(\phi)\, R d\phi dR = ... = \frac{b^3 d\pi}{16} = 0,1963 b^3 d$$

Für den Steineranteil wird als Erstes dieses Teilflächenträgheitsmoment auf den Eigenschwerpunkt (eSP) bezogen:

$$I_{z,KB,eSP} = 0,1963 b^3 d - \frac{bd\pi}{2} * \left( \frac{b}{\pi} \right)^2 = 0,03714 b^3 d$$

Anschließend kann dieses neue Teilflächenträgheitsmoment des Kreisbogens im Teilschwerpunkt unter Nutzung des Satzes von Steiner auf den Gesamtschwerpunkt (gSP) bezogen werden:

$$I_{z,KB,gSP} = 0,03714 b^3 d + \frac{bd\pi}{2} * \left[ b \left( \frac{1}{\pi} + 0,14 \right) \right]^2 = (0,0371 + 0,3299) b^3 d = 0,367 b^3 d$$

Somit folgt $I_z$ als Summe von $I_{z1}$ + $I_{z2}$ + $I_{z3}$:

$$I_z = 0,7928 b^3 d$$

$$I_{yz} = 0,0$$

Hauptträgheitsmomente:

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$= \frac{(0,696 + 0,793) b^3 d}{2} + \sqrt{\left( \frac{(0,696 - 0,793) b^3 d}{2} \right)^2 + 0} = 0,793 b^3 d$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

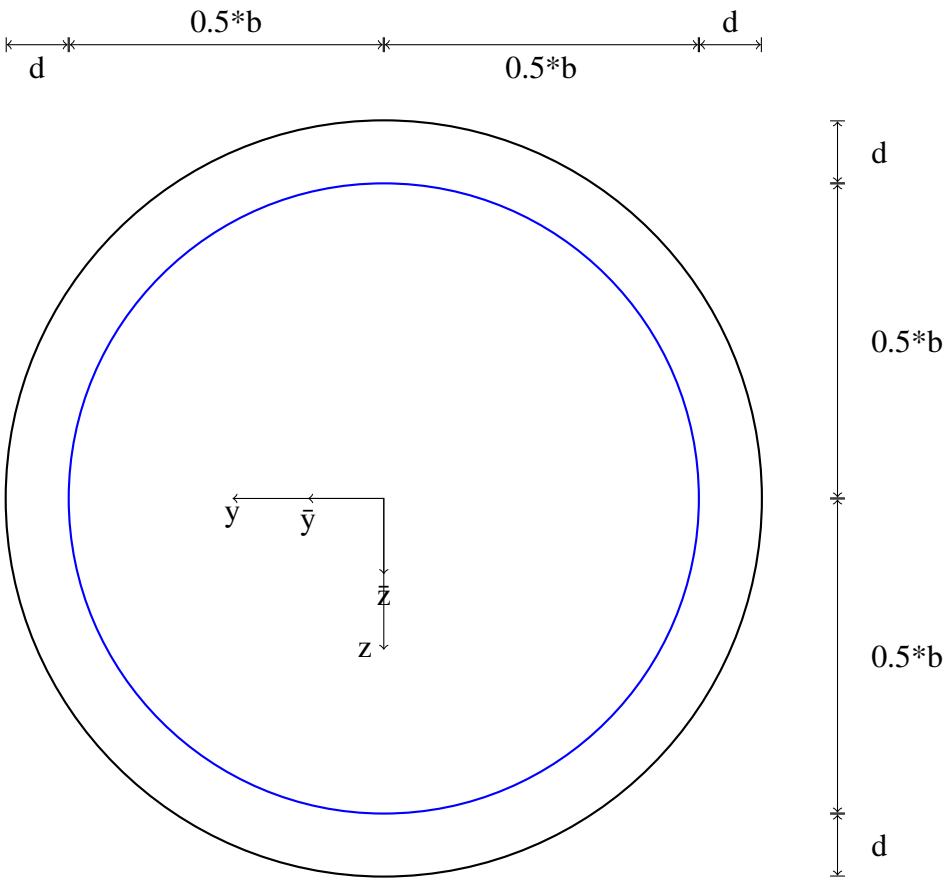
$$= \frac{(0,696 + 0,793) b^3 d}{2} - \sqrt{\left( \frac{(0,696 - 0,793) b^3 d}{2} \right)^2 + 0} = 0,696 b^3 d$$

Mit arctan(0)=0 folgt für den Drehwinkel der Hautachsen: $\phi = 0^\circ$
Somit stimmen die Ergebnisse der Handrechnung von Beispiel 7 mit jenen des Programmes überein.

## 5.8 Beispiel 8

Bei Beispiel 8 handelt es sich um einen Kreisringquerschnitt. Die Dicke des Ringes beträgt d, der innere Durchmesser b.

### 5.8.1 Programmausgabe Beispiel 8




Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$

$$\pi d\,(b+d)\left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z\,\mathrm{d}A =$$

$$0\left[m^3\right]$$

$$S_z = \int_A y \, \mathrm{d}A =$$

$$0 \left[ m^3 \right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0 \left[ m \right]$$

$$z_s = \frac{S_y}{A} = 0 \left[ m \right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A =$$

$$-\frac{\pi}{64} \left( b^4 - (b+2d)^4 \right) \left[ m^4 \right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$-\frac{\pi}{64} \left( b^4 - (b+2d)^4 \right) \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$0 \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$\frac{\pi}{64} \left( -b^4 + (b+2d)^4 \right) \left[ m^4 \right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$\frac{\pi}{64} \left( -b^4 + (b+2d)^4 \right) \left[ m^4 \right]$$

Die Hauptachse 1 entspricht Iy um 0° gedreht.

### 5.8.2 Handrechnung Beispiel 8

Fläche:

$$A = \frac{\pi}{4}\left(D^2 - d^2\right) = \frac{\pi}{4}\left(b^2 + 4db + 4d^2 - b^2\right) = d\pi(b+d)$$

Aus Symmetriegründen folgt, dass der Schwerpunkt im Kreisringmittelpunkt liegen muss, da dieser der Koordinatenurspung ist, liegt der Schwerpunkt bei (0/0). Damit dies erüllt wird, muss gelten: $S_y = S_z = 0$.
Flächtenträgheitsmomente:

$$I_y = I_z = \frac{\pi}{4}\left(r_a^4 - r_i^4\right) = \frac{\pi}{4}\left[\left(\frac{b}{2} + d\right)^4 - \left(\frac{b}{2}\right)^4\right]$$

wird dieser Term mit $\left(\frac{2}{2}\right)^4$ erweitert folgt:

$$I_y = I_z = \frac{\pi}{64}\left[\left(b + 2d\right)^4 - b^4\right]$$

Wiederrum aus Symmetriegründen folgt, dass $I_{yz} = 0$ gelten muss.

Hautträgheitsmomente:

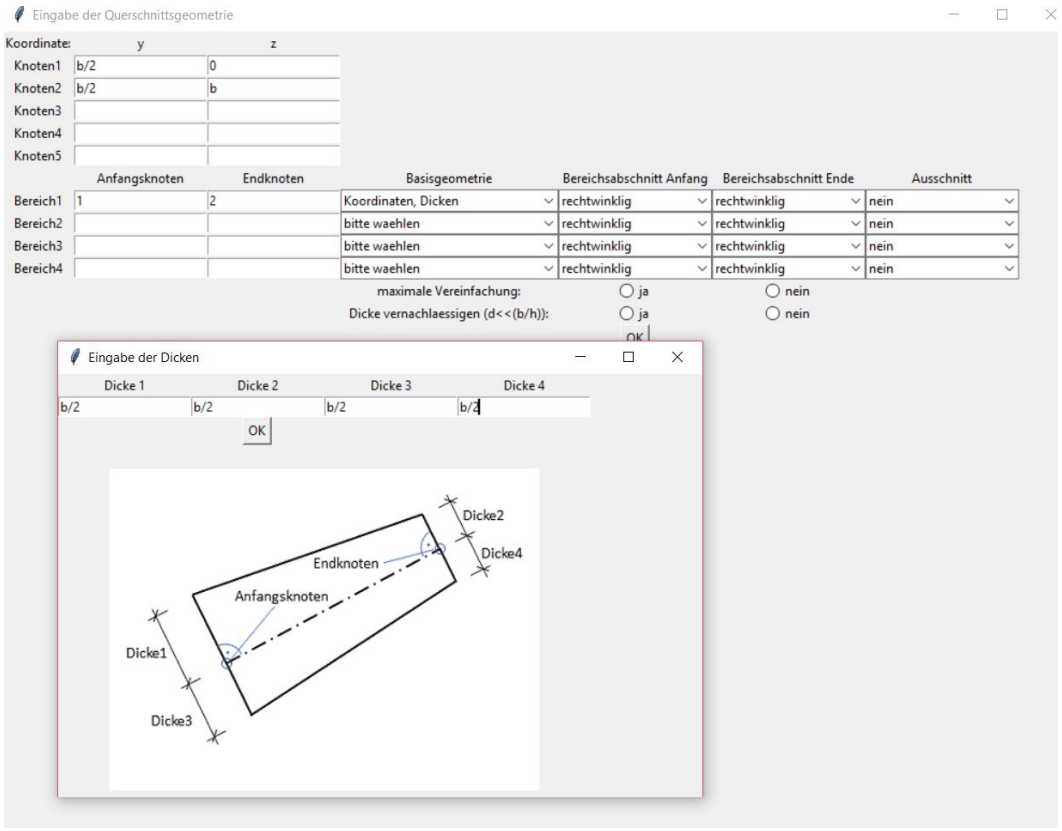
$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} = I_y = I_z = I^{(1)} = I^{(2)} = \frac{\pi}{64}\left[\left(b + 2d\right)^4 - b^4\right]$$

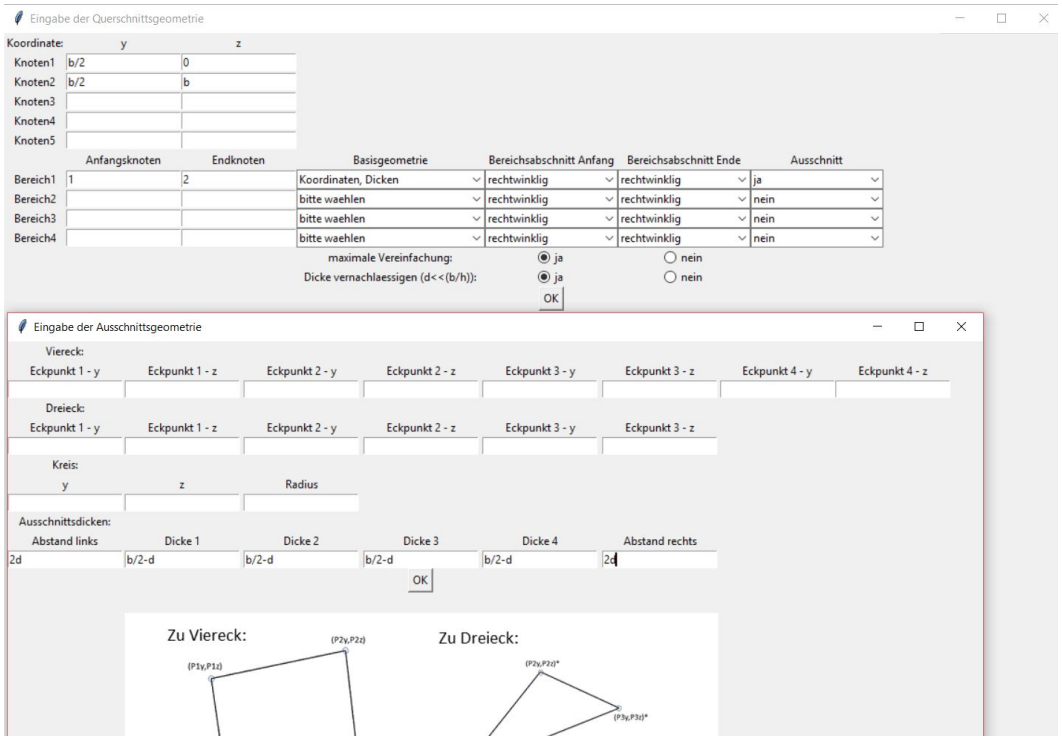
Mit arctan(0)=0 folgt für den Drehwinkel der Hautachsen: $\phi = 0^\circ$

Somit stimmen die Ergebnisse der Handrechnung von Beispiel 8 mit jenen des Programmes überein.

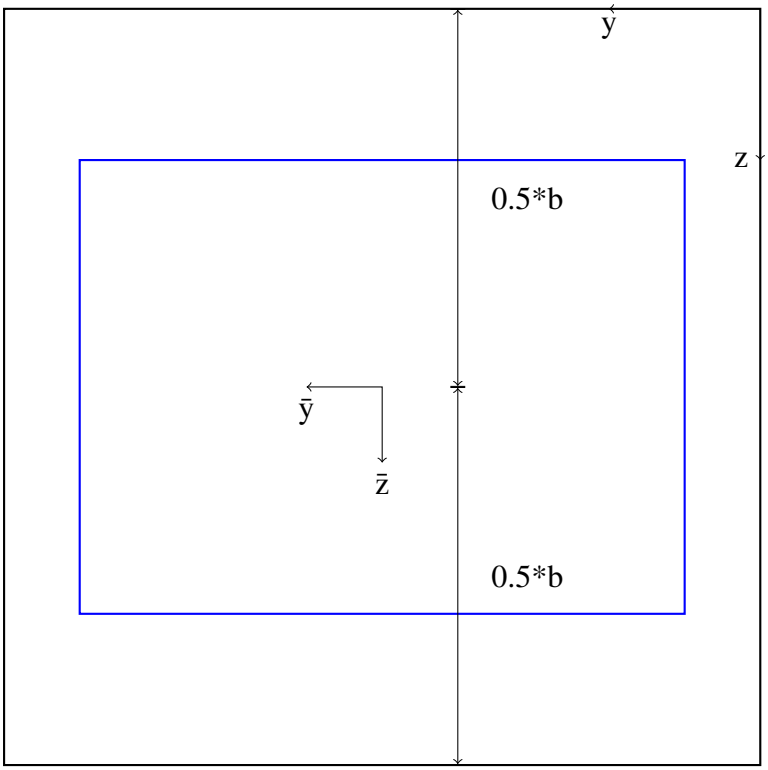
## 5.9 Beispiel 9

In diesem Beispiel wird ein Hohlprofil mit unterschiedlichen Stärken behandelt. Die Eingabe erfolgt mittels “Koordinaten, Dicken“. Als Alternative wäre die koordinative Eingabe anzuführen.



### 5.9.1 Programmausgabe Beispiel 9




Querschnittskennwerte
Fläche:

$$A = \int_A \mathrm{d}A =$$

$$6.0bd\left[m^2\right]$$

Statische Momente:

$$S_y = \int_A z\,\mathrm{d}A =$$

$$3.0b^2d\left[m^3\right]$$

$$S_z = \int_A y\,\mathrm{d}A =$$

$$3.0b^2d\left[m^3\right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0.5b\left[m\right]$$

$$z_s = \frac{S_y}{A} = 0.5b\left[m\right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A =$$

$$1.17 b^3 d \left[ m^4 \right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$0.833 b^3 d \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$0 \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$1.17 b^3 d \left[ m^4 \right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left( \frac{I_y - I_z}{2} \right)^2 + I_{yz}^2} =$$

$$0.833 b^3 d \left[ m^4 \right]$$

Die Hauptachse 1 entspricht Iy um 0 ° gedreht.

### 5.9.2 Handrechnung zu Beispiel 9

Fläche:

$$A = b^2 - (b-4d)(b-2d) = 6bd$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A$$

Lösen des Integrals ergibt:

$$S_y = \int_{y=0}^{b} \int_{z=0}^{b} z \, dzdy - \int_{y=d}^{b-db-4d} \int_{z=2d} z \, dzdy = 3b^2 d$$

$$S_z = \int_{y=0}^{b} \int_{z=0}^{b} y \, dzdy - \int_{y=d}^{b-db-4d} \int_{z=2d} y \, dzdy = 3b^2 d$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = \frac{3b^2 d}{6bd} = \frac{b}{2}$$

$$z_s = \frac{S_y}{A} = \frac{3b^2 d}{6bd} = \frac{b}{2}$$

$$I_y = \int_A z^2 \, \mathrm{d}A$$

$$I_z = \int_A y^2 \, \mathrm{d}A$$

$$I_{yz} = \int_A yz \, \mathrm{d}A$$

Satz von Steiner:

$$I'_y = \int_A z^2 \, \mathrm{d}A + A(z'_s)^2$$

$$I'_z = \int_A y^2 \, \mathrm{d}A + A(y'_s)^2$$

$$I'_{yz} = \int_A yz \, \mathrm{d}A + A(y'_s z'_s)$$

Einsetzen in die Formeln ergibt:

$$I_y = \int_{y=-\frac{b}{2}}^{\frac{b}{2}} \int_{z=-\frac{b}{2}}^{\frac{b}{2}} z^2 \, dzdy - \int_{y=-\frac{b}{2}+dz=-\frac{b}{2}+2d}^{\frac{b}{2}-d} \int^{\frac{b}{2}-2d} z^2 \, dzdy = \frac{7b^3d}{6}$$

$$I_z = \int_{y=-\frac{b}{2}}^{\frac{b}{2}} \int_{z=-\frac{b}{2}}^{\frac{b}{2}} y^2 \, dzdy - \int_{y=-\frac{b}{2}+dz=-\frac{b}{2}+2d}^{\frac{b}{2}-d} \int^{\frac{b}{2}-2d} y^2 \, dzdy = \frac{5b^3d}{6}$$

$$I_{yz} = \int_{y=-\frac{b}{2}}^{\frac{b}{2}} \int_{z=-\frac{b}{2}}^{\frac{b}{2}} yz \, dzdy - \int_{y=-\frac{b}{2}+dz=-\frac{b}{2}+2d}^{\frac{b}{2}-d} \int^{\frac{b}{2}-2d} yz \, dzdy = 0$$

Für die Hauptträgheitsmomente gilt:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$

$$I^{(1,2)} = \frac{\frac{7b^3d}{6} + \frac{5b^3d}{6}}{2} \pm \sqrt{\left(\frac{\frac{7b^3d}{6} - \frac{5b^3d}{6}}{2}\right)^2 + 0^2}$$

$$I^{(1)} = I_y = 1.17b^3d$$

$$I^{(2)} = I_z = 0.833b^3d$$

Für den Drehwinkel gilt:

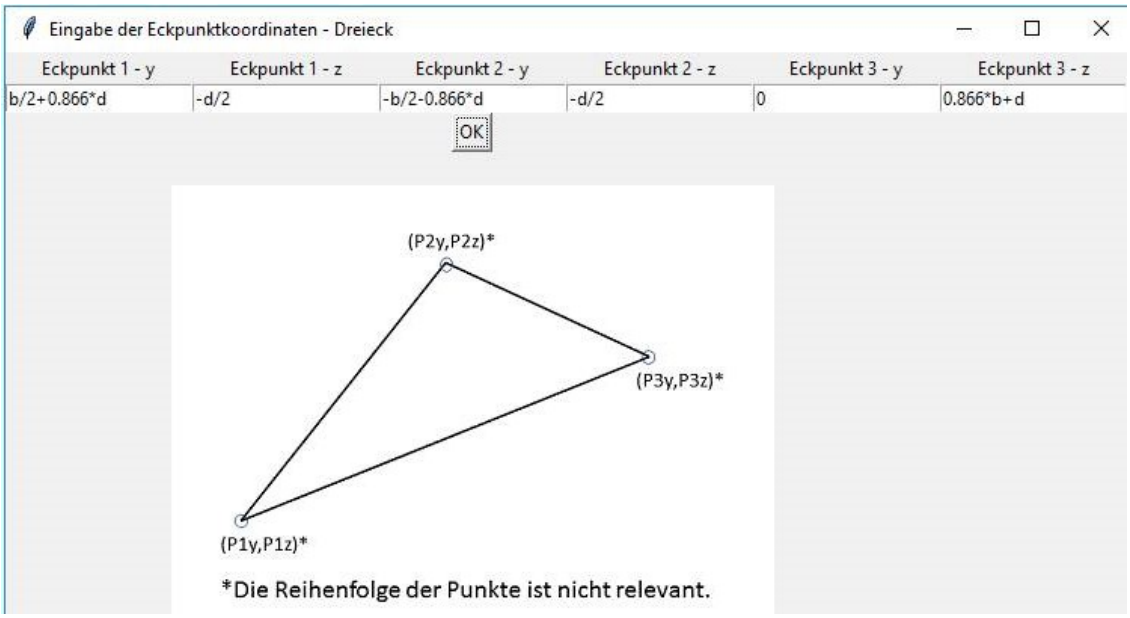
$$tan(2\phi) = -\frac{2I_{yz}}{I_y - Iz}$$

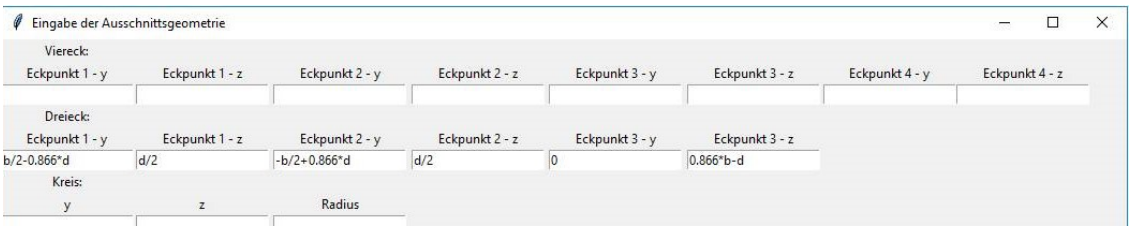
Da $I_{yz} = 0$ gilt, ergibt sich der Drehwinkel $\phi$ zu $\frac{arctan(0)}{2} = 0$.

Somit stimmen die Ergebnisse der Handrechnung von Beispiel 9 mit jenen des Programmes überein.

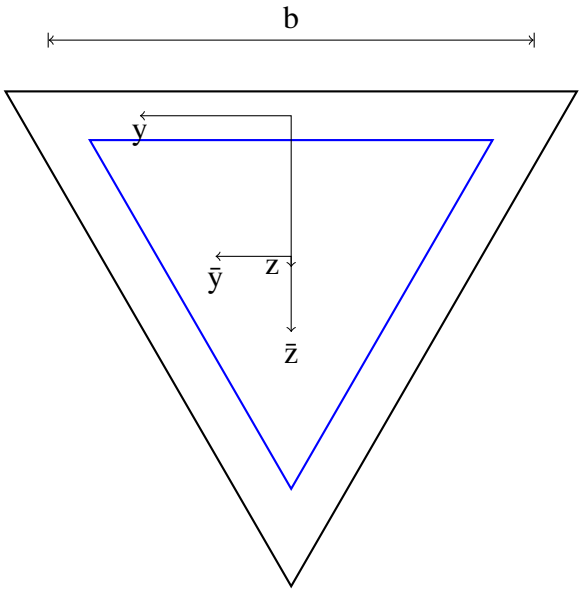
## 5.10 Beispiel 10

Bei Beispiel 10 handelt es sich um ein gleichseitiges Dreieck mit einem kleineren gleichseitigen Dreieck als Ausschnitt.




### 5.10.1 Programmausgabe Beispiel 10



Querschnittskennwerte Fläche:

$$A = \int_A \mathrm{d}A =$$

$$3.0bd \left[ m^2 \right]$$

Statische Momente:

$$S_y = \int_A z \, \mathrm{d}A =$$

$$0.866b^2 d \left[ m^3 \right]$$

$$S_z = \int_A y \, \mathrm{d}A =$$

$$0 \left[ m^3 \right]$$

Schwerpunkt:

$$y_s = \frac{S_z}{A} = 0 \left[ m \right]$$

$$z_s = \frac{S_y}{A} = 0.289b \left[ m \right]$$

Flächenträgheitsmomente:

$$I_y = \int_A z^2 \, \mathrm{d}A =$$

$$0.25b^3 d \left[ m^4 \right]$$

$$I_z = \int_A y^2 \, \mathrm{d}A =$$

$$0.25b^3 d \left[ m^4 \right]$$

$$I_{yz} = \int_A yz \, \mathrm{d}A =$$

$$0 \left[ m^4 \right]$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$0.25 b^3 d \left[m^4\right]$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$0.25 b^3 d \left[m^4\right]$$

Die Hauptachse 1 entspricht Iy um 0° gedreht.

### 5.10.2 Handrechnung Beispiel 10

Fläche:

$$A_1 = \frac{a h_a}{2} = \frac{(b + \sqrt{3}d)(\frac{\sqrt{3}}{2}b + \frac{3d}{2})}{2} = \frac{\frac{\sqrt{3}}{2}b^2 + \frac{3bd}{2} + \frac{3db}{2} + \frac{3\sqrt{3}d^2}{2}}{2}$$

$$A_2 = \frac{a h_a}{2} = \frac{(b - \sqrt{3}d)(\frac{\sqrt{3}}{2}b - \frac{3d}{2})}{2} = \frac{\frac{\sqrt{3}}{2}b^2 - \frac{3bd}{2} - \frac{3db}{2} + \frac{3\sqrt{3}d^2}{2}}{2}$$

Das Beispiel wurde unter Berücksichtigung der Vereinfachung, dass für d « b Terme mit $d^n$ mit Ordnung 2 oder höher weg fallen.
$A_1$ ist die Fläche des äußeren Dreiecks, $A_2$ die des inneren somit ergibt sich die Gesamtfläche aus der Subtraktion von $A_1$ - $A_2$.
Somit folgt für A:

$$A = \frac{2\frac{1}{2}(3bd + 3bd)}{2} = \frac{6bd}{2} = 3bd$$

Aus Symmetriegründen folgt, dass der y-Schwerpunkt 0 ist, somit gilt $S_z$ = 0.

$$z_s = \frac{1}{3}(z_1 + z_2 + z_3) = \frac{1}{3}\left(\left(-\frac{d}{2}\right) + \left(-\frac{d}{2}\right) + \frac{\sqrt{3}}{2}b + d\right) = \frac{1}{3}\frac{\sqrt{3}}{2}b = \frac{\sqrt{3}}{6}b = 0,2887b$$

Obige Formel mit $z_i$ als z- Koordinaten der Dreieckseckpunkte.
Mit $z_s$ und der Fläche folgt für $S_y$=$\frac{\sqrt{3}}{2}b^3$=$0,866b^3$.
Flächtenträgheitsmomente:

$$I_y = \frac{bh^3}{36}$$

$$I_{y1} = \frac{bh^3}{36} = \frac{(b + \sqrt{3}d)(\frac{\sqrt{3}}{2}b + \frac{3d}{2})^3}{36}$$

$$I_{y2} = \frac{bh^3}{36} = \frac{(b - \sqrt{3}d)(\frac{\sqrt{3}}{2}b - \frac{3d}{2})^3}{36}$$

Mit $I_{y1}$ für das Flächtenträgheitsmoment des äußeren Dreiecks und $I_{y2}$ für das Flächenträgheitsmoment des inneren Dreiecks ergibt die Subtraktion der ausmultiplizierten Formeln unter Berücksichtigung, dass für d « b Terme mit $d^n$ mit Ordnung 2 oder höher entfallen:

$$I_{y1} - I_{y2} = \frac{1}{36}\left[2\frac{9}{2}\left(\frac{\sqrt{3}}{2}\right)^2 b^3 d + 2\sqrt{3}\left(\frac{\sqrt{3}}{2}\right)^3 b^3 d\right] =$$

$$\frac{b^3 d}{36}\left[9\left(\frac{\sqrt{3}}{2}\right)^2 + 2\sqrt{3}\left(\frac{\sqrt{3}}{2}\right)^3\right] = \frac{9}{36}b^3 d = \frac{1}{4}b^3 d =$$

$$0,25b^3 d$$

$I_z$ ist ebenfalls $0,25b^3 d$.
Um dieses Ergebnis zu erhalten werden die drei Rechtecke, welchen den Querschnitt bilden, einzeln betrachtet und mit dem Satz von Steiner das Gesamtträgheitsmoment berechnet:

$$I_{z1} = \frac{b^3 d}{12}$$

Der Steineranteil ist hier null. Für die beiden geneigten Rechtecke sind Transformationsformeln notwendig. Hierzu wird das Koordinatensystem um 30° so gedreht, dass die z-Achse parallel zur Längsseite des Rechtecks ist. Die Transformationsformel lautet:

$$I_{z'} = \frac{1}{2}(I_y + I_z) - \frac{1}{2}(I_y - I_z)cos(2\phi) + I_{yz}sin(2\phi)$$

$I_y$ ist hier $I_y$ des Teilrechtecks im gedrehten Koordinatensystem und beträgt $I_y = \frac{b^3 d}{12}$, $I_z = \frac{d^3 b}{12}$, fällt also weg und $I_{yz} = 0$.
Damit folgt für das Gesamtflächenträgheitsmoment:

$$I_z = \frac{b^3 d}{12} + 2\left(bd\left(\frac{b}{4}\right)^2 + 0,020833b^3 d\right) = 0,25b^3 d$$

Aufgrund der Symmetrie ist $I_{yz}$ = 0.

Hautträgheitsmomente:

$$I^{(1,2)} = \frac{I_y + I_z}{2} \pm \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2}$$

$$I^{(1)} = \frac{I_y + I_z}{2} + \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$= \frac{(0,25+0,25)b^3 d}{2} + \sqrt{\left(\frac{(0,25-0,25)b^3 d}{2}\right)^2 + 0} = 0,25b^3 d$$

$$I^{(2)} = \frac{I_y + I_z}{2} - \sqrt{\left(\frac{I_y - I_z}{2}\right)^2 + I_{yz}^2} =$$

$$= \frac{(0,25+0,25)b^3 d}{2} - \sqrt{\left(\frac{(0,25-0,25)b^3 d}{2}\right)^2 + 0} = 0,25b^3 d$$

Mit arctan(0)=0 folgt für den Drehwinkel der Hautachsen: $\phi$ = 0°
Somit stimmen die Ergebnisse der Handrechnung von Beispiel 10 mit jenen des Programmes überein.

# 6 SCHLUSSFOLGERUNG UND AUSBLICK

## 6.1 Konklusion

Insgesamt sind wir mit dem Ergebnis des Projekts zufrieden. Es ist eine Vielzahl an Berechnungen möglich, jedoch kann speziell die symbolische Berechnung bei komplexen Geometrien laufzeitintensiv werden. Zudem kann es sein, dass die Ergebnisse sehr lang werden, wodurch sie, obwohl korrekt und entsprechend vereinfacht, nicht mehr vernünftig interpretiert werden können. Trotz umfassender Tests sind auch Programmierfehler nicht ausgeschlossen.

## 6.2 Erweiterungsmöglichkeiten

Das Programm könnte in mehreren Punkten erweitert werden. Einerseits wäre es möglich, weitere Querschnittskennwerte zu ermitteln. Hierbei würde infolge eine Betrachtung der Torsion - und Schubspannung erfolgen. Insbesondere der Verlauf der Schubspannungen sowie der Schubmittelpunkt wären interessant, jedoch müssen dafür Schnittkräfte bekannt sein. Dafür würde es sich anbieten, das Gesamtsystem zu betrachten. Durch Informationen der Systemgeometrie, Materialparameter und Belastung könnten Schnittkraftverläufe ermittelt werden. Infolge dessen wäre die Ermittlung der Spannungsverteilung möglich.

# LITERATURVERZEICHNIS


[1] diverse themen zu python. `http://stackoverflow.com/`.

[2] Python 3 documentation. `https://docs.python.org/3/`.

[3] Hans-Jochen Bartsch. *Taschenbuch mathematischer Formeln für Ingenieure und Naturwissenschaftler*. Carl Hanser Verlag GmbH Co KG, 2014.

[4] Klaus Braune, Joachim Lammarsch, and Marion Lammarsch. *LaTeX: Basissystem, Layout, Formelsatz*. Springer-Verlag, 2007.

[5] D Gross, W Hauger, J Schröder, and WA Wall. Technische mechanik band 1 – statik; 12. *Berlin/Heidelberg*, 2013.

[6] Hauger Gross, Werner Hauger, J Schröder, and WA Wall. *Technische Mechanik 2 Elastostatik*. Springer-Verlag, Berlin Heidelberg, 2011.

[7] T Krapfenbauer. Bautabellen. *Verlag Jugend & Volk GmbH, Wien, edition*, 18, 2011.

[8] Marco Öchsner and Andreas Öchsner. *Das Textverarbeitungssystem LaTeX: Eine praktische Einführung in die Erstellung wissenschaftlicher Dokumente*. Springer-Verlag, 2015.

[9] Mark Pilgrim. *Python 3-Intensivkurs: Projekte erfolgreich realisieren*. Springer-Verlag, 2010.

[10] Univ.-Prof. Dr.-Ing. Schanz Martin. Baumechanik 1 + 2. Vorlesungsskriptum TU Graz, 2015/16.