

**Großer Beleg**

**Anwendungsorientiertes Programm zur  
Auslegung von Kurvenkoppelgetrieben**

von

**Lutz Wirsig**

betreut von

Prof. Dr. rer. nat. habil. K.-H. Modler

Doz. Dr.-Ing. E.-C. Lovasz

Technische Universität Dresden

Fakultät Maschinenwesen


Institut für Festkörpermechanik

Professur für Getriebelehre

**Dresden, 2002**

## Aufgabenstellung für den Großen Beleg

im Studiengang Maschinenbau

Für Herrn **Lutz Wirsig**, Matr.-Nr.: 2549652 

Thema: „Anwendungsorientiertes Programm zur Auslegung von Kurvenkoppelgetrieben“


Es soll ein anwendungsorientiertes Programm erstellt werden, das für die Auslegung von Kurvengetrieben mit nachgeschalteten Koppelgetrieben geeignet ist.

Die Arbeit soll wie folgt gegliedert werden:

1. Auswahl der Programmiersprache
2. Festlegung des Funktionsumfanges
3. Programmentwurf
4. Auswertung
5. Dokumentation

Für die Bearbeitung der Aufgaben ist ein Zeitumfang von 500 Std. innerhalb von 6 Monaten vorgesehen.

Die Arbeit ist in 2facher Ausfertigung abzugeben.

Ausgabe der Aufgabenstellung:      ??.??.2002 

Abgabe des Belegs:                      ??.??.2002 

Prof. Dr. rer. nat. habil. K.-H. Modler  
(betreuender Hochschullehrer)

Doz. Dr.-Ing. E.-C. Lowasz  
(wissenschaftlicher Betreuer)

# Inhaltsverzeichnis

<b>Formelzeichen</b>	<b>iii</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Getriebetechnische Vorbetrachtungen</b>	<b>2</b>
2.1 Bewegungsgesetze . . . . .	2
2.2 Bewegungsplan, Bewegungsdiagramm und Bewegungsaufgaben . . . .	4
2.3 Normierte Bewegungsgesetze für Rast in Rast . . . . .	7
2.3.1 Symmetrische normierte Bewegungsgesetze für Rast in Rast .	7
2.3.2 Unsymmetrische normierte Bewegungsgesetze für Rast in Rast	10
2.4 Kinematische Abmessungen bei Kurvengetrieben . . . . .	14
2.4.1 Ermittlung des Grundkreisradius . . . . .	16
2.4.2 Ermittlung der Mittelpunktskurve . . . . .	18
<b>3 Programmentwurf</b>	<b>22</b>
3.1 Auswahl der Programmiersprache . . . . .	22
3.2 Wirkungsweise und Funktionsumfang des Programms . . . . .	23
3.3 Kinematische Abmessungen im Programm . . . . .	26
3.3.1 Übertragungsfunktionen . . . . .	26
3.3.2 Bewegungsgesetz . . . . .	28
3.3.3 Grundkreiswinkel und Grundkreisradius . . . . .	30
3.3.4 Übertragungswinkel . . . . .	30
3.3.5 Krümmungsradius . . . . .	32
<b>4 Zusammenfassung</b>	<b>33</b>
<b>Anhang</b>	<b>34</b>
<b>A Quellcode</b>	<b>35</b>
<b>B Beispiele</b>	<b>84</b>
B.1 Querhubkurvenscheibe (opticurv version 1.0.3) . . . . .	84
B.2 Schnittkurvenscheibe (opticurv version 1.0.3) . . . . .	85
B.3 Setzhubkurvenscheibe (opticurv version 1.0.3) . . . . .	86
B.4 Vorschubkurvenscheibe (opticurv version 1.0.3) . . . . .	87
<b>Literatur</b>	<b>88</b>
<b>Stichwortverzeichnis</b>	<b>89</b>

# Abbildungsverzeichnis

1 An- und Abtriebsbewegung a) Stößel b) Schwinge . . . . .	2
--	---

2	Bewegungsplan . . . . .	4
3	Bewegungsdiagramm . . . . .	5
4	Zusammenhang zwischen realem und normiertem Bewegungsgesetz .	7
5	Normiertes symmetrisches Bewegungsgesetz . . . . .	8
6	Symmetrisches 3-4-5 Polynom . . . . .	10
7	Normiertes unsymmetrisches Bewegungsgesetz . . . . .	11
8	Unsymmetrisches 3-4-5 Polynom . . . . .	13
9	Übertragungswinkel $\mu$ . . . . .	14
10	Kurvengetriebe mit Schwinghebel a) F-Kurvengetr. b) P-Kurvengetr.	15
11	Kurvengetriebe mit Schieber a) F-Kurvengetr. b) P-Kurvengetr. . . .	15
12	Ermittlung der $A_0$ -Bereiche nach <i>Flocke</i> für Kurvengetriebe mit a) Schwinghebel b) Schieber . . . . .	17
13	Grundfigur zur Berechnung der Rollenmittelpunktskurve bei einem Kurvengetriebe mit Schwinghebel . . . . .	18
14	Grundfigur zur Berechnung der Rollenmittelpunktskurve bei einem Kurvengetriebe mit Schieber . . . . .	20
15	Symbole für Programmstrukturen a) Wiederholung b) Alternative c) Mehrfachverzweigung d) Reihung . . . . .	24
16	Struktogramm des Hauptprogramms . . . . .	25
17	Kurvenkoppelgetriebe mit Schwinghebel . . . . .	26
18	Bewegungsplan des Kurvenkoppelgetriebes . . . . .	29
19	Übertragungswinkel $\mu$ am Kurvengetriebe . . . . .	30
20	Spitzenbildung und Unterschnitt bei zu kleinem Krümmungsradius $r_K$	32

## Tabellenverzeichnis

1	Bewegungsaufgaben . . . . .	5
2	Mögliche Kombinationen von Bewegungsaufgaben . . . . .	6
3	Vor- und Nachteile von Programmiersprachen . . . . .	23
4	Eingabeparameter für die Querhubkurvenscheibe . . . . .	84
5	Eingabeparameter für die Schnittkurvenscheibe . . . . .	85
6	Eingabeparameter für die Setzhubkurvenscheibe . . . . .	86
7	Eingabeparameter für die Vorschubkurvenscheibe . . . . .	87

# Formelzeichen

Zeichen	Bezeichnung	Einheit
$a = \ddot{s}$	Beschleunigung des Abtriebsgliedes (gerade geführt)	[mm/s <sup>2</sup> ]
$A_0$	Lagerstelle der Kurvenscheibe	[—]
$B_0$	Lagerstelle des Rollenhebels	[—]
$B_{ik}$	Rollenmittelpunkt im Bewegungsabschnitt $ik$	[—]
$B_a$	Rollenmittelpunkt zu Beginn der Bewegung	[—]
$C_a$	Beschleunigungskennwert	[—]
$C_j$	Ruckkennwert	[—]
$C_{Mstat}$	statischer Momentenkennwert	[—]
$C_{Mdyn}$	dynamischer Momentenkennwert	[—]
$C_v$	Geschwindigkeitskennwert	[—]
$e$	Exzentrizität	[mm]
$e_{sk}$	Exzentrizität der Koppel	[mm]
$f_{ik}$	normierter Weg	[—]
$ik$	Nummerierung der Bewegungsabschnitte	[—]
$k_{B32}$	Rollenmittelpunktskurve	[—]
$k_G$	Grundkreis	[—]
$l_1$	Gestelllänge (Glied 1)	[mm]
$l_3$	Rollenhebellänge (Glied 3)	[mm]
$l_4$	Koppellänge (Glied 4)	[mm]
$l_{sk}$	Schubkurbellänge (Glied 3)	[mm]
$r_G$	Grundkreisradius	[mm]
$r_K$	Krümmungsradius	[mm]
$r_R$	Laufrollenradius	[mm]

$s$	Abtriebsweg	[mm]
$s_0$	Hub in der Nullstellung	[mm]
$s_G$	Grundhub	[mm]
$s_{Hik}$	Gesamtweg des gerade geführten Abtriebsgliedes im Abschnitt $ik$	[mm]
$t$	Zeit	[s]
$v = \dot{s}$	Geschwindigkeit des Abtriebsgliedes (gerade geführt)	[mm/s]
$z_{ik}$	normierter Drehwinkel	[—]
$\alpha_{sk}$	Anfangsauslenkung der Koppel	[°]
$\vartheta$	Lagewinkel von Glied 4 beim Kurvenkoppelgetriebe	[°]
$\lambda$	Wendepunktparameter	[—]
$\mu$	Übertragungswinkel	[°]
$\varphi$	Antriebswinkel	[°]
$\varphi_{Hik}$	Gesamtdrehwinkel der Kurvenscheibe im Abschnitt $ik$	[°]
$\psi$	Abtriebswinkel	[°]
$\psi_0$	Abtriebswinkel in Nullstellung	[°]
$\psi_G$	Grundwinkel	[°]
$\psi_{Hik}$	Gesamtdrehwinkel des schwingend geführten Abtriebsgliedes im Abschnitt $ik$	[°]

# 1 nleitung

## 2 Getriebetechnische Vorbetrachtungen

### 2.1 Bewegungsgesetze

Die Bewegungsgesetze beschreiben die Relativbewegung zwischen zwei Getriebegliedern. Dies ist in der Regel die Bewegung des Abtriebsgliedes in Abhängigkeit von der Bewegung des Antriebsgliedes.

Der Antriebswinkel  $\varphi(t)$ , der Abtriebsweg  $s[\varphi(t)]$  und der Abtriebswinkel  $\psi[\varphi(t)]$  kennzeichnen die jeweilige Bewegung der Getriebeglieder (Abb. 1).

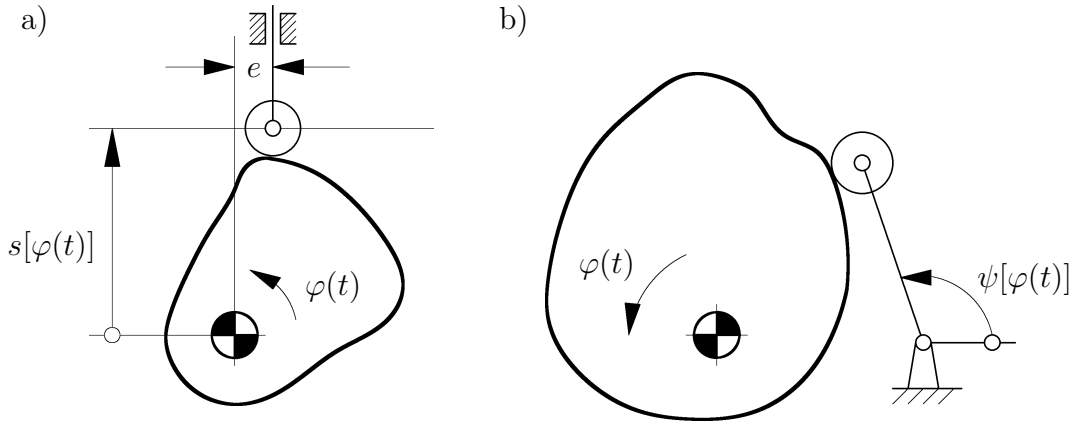


Abbildung 1: An- und Abtriebsbewegung a) Stößel b) Schwinde [7]

Die Geschwindigkeiten und Beschleunigungen der Abtriebsglieder ergeben sich aus den Ableitungen ihrer Bewegungen. Ableitungen nach dem Drehwinkel  $\varphi$  werden im Folgenden durch einen Strich und Ableitungen nach der Zeit werden durch einen Punkt gekennzeichnet. Dies gilt jedoch nicht für Substitutionsvariablen (z.B.  $u$  und  $v$ ). Diese werden generell durch einen Strich angegeben, um die Anwendung der Differentiationsregeln zu veranschaulichen.

Die 1. Ableitungen lassen sich nach der Kettenregel <sup>1</sup> bilden. Bei der Ableitung des Weges  $s$  nach der Zeit sind:

$$\begin{aligned} u[v(x)] &= s[\varphi(t)] \\ u'[v(x)] &= s'[\varphi(t)] \\ v'(x) &= \dot{\varphi}(t) \end{aligned}$$

und schließlich

$$\dot{s} = s'[\varphi(t)] \cdot \dot{\varphi}(t) = s' \cdot \dot{\varphi} \quad (2.1)$$

---

<sup>1</sup> $f(x) = u[v(x)] \rightsquigarrow f'(x) = u'[v(x)] \cdot v'(x)$



Bei der Ableitung des Abtriebswinkels  $\psi$  nach der Zeit sind:

$$\begin{aligned} u[v(x)] &= \psi[\varphi(t)] \\ u'[v(x)] &= \psi'[\varphi(t)] \\ v'(x) &= \dot{\varphi}(t) \end{aligned}$$

und schließlich

$$\dot{\psi} = \psi'[\varphi(t)] \cdot \dot{\varphi}(t) = \psi' \cdot \dot{\varphi} \quad (2.2)$$

Die zweiten Ableitungen bildet man nach der Produktregel<sup>2</sup>. Es gilt für die Beschleunigung  $\ddot{s}$ :

$$\begin{aligned} u &= s'[\varphi(t)] & \rightsquigarrow u' &= s''[\varphi(t)] \cdot \dot{\varphi}(t) \\ v &= \dot{\varphi}(t) & \rightsquigarrow v' &= \ddot{\varphi}(t) \end{aligned}$$

und schließlich

$$\ddot{s} = s''[\varphi(t)] \cdot \dot{\varphi}(t) \cdot \dot{\varphi}(t) + \ddot{\varphi}(t) \cdot s'[\varphi(t)] = s'' \cdot \dot{\varphi}^2 + \ddot{\varphi} \cdot s' \quad (2.3)$$

Für die Winkelbeschleunigung  $\ddot{\psi}$  ergibt sich:

$$\begin{aligned} u &= \psi'[\varphi(t)] & \rightsquigarrow u' &= \psi''[\varphi(t)] \cdot \dot{\varphi}(t) \\ v &= \dot{\varphi}(t) & \rightsquigarrow v' &= \ddot{\varphi}(t) \end{aligned}$$

und schließlich

$$\ddot{\psi} = \psi''[\varphi(t)] \cdot \dot{\varphi}(t) \cdot \dot{\varphi}(t) + \ddot{\varphi}(t) \cdot \psi'[\varphi(t)] = \psi'' \cdot \dot{\varphi}^2 + \ddot{\varphi} \cdot \psi' \quad (2.4)$$

Die Gleichungen 2.1 bis 2.4 zeigen, daß für die Abtriebsglieder die gleichen Bewegungsgesetze gelten. Im weiteren Text werden daher nur die Beziehungen für den Weg  $s$  der geradlinig geführten Abtriebsglieder behandelt. Bei Getrieben mit Schwinghebel als Abtriebsglied ist in den Gleichungen der Weg  $s$  durch den Abtriebswinkel  $\psi$  zu ersetzen.

---

<sup>2</sup> $y = u \cdot v \rightsquigarrow y' = u'v + v'u$

## 2.2 Bewegungsplan, Bewegungsdiagramm und Bewegungsaufgaben

Im Bewegungsplan wird die geforderte Abtriebsbewegung dargestellt und es wird jeder Bewegung (z.B. Rast, Übergang) ein Abschnitt zugeordnet (Abb. 2).

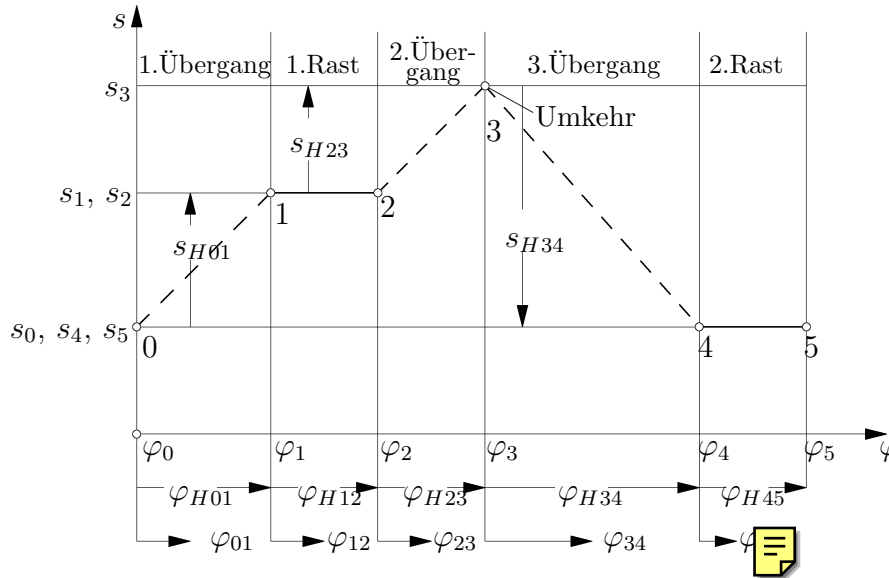


Abbildung 2: Bewegungsplan [7]

Der Gesamtdrehwinkel  $\varphi_H$  eines Abschnittes bekommt die Indizes  $ik$ :

$$\varphi_{Hik} = \varphi_k - \varphi_i \quad (2.5)$$

mit

$\varphi_i$  Drehwinkel des Antriebsgliedes zu Beginn des Bewegungsabschnittes

$\varphi_k$  Drehwinkel des Antriebsgliedes am Ende des Bewegungsabschnittes

Analog gilt für den Gesamtweg  $s_H$  eines Bewegungsabschnittes:

$$s_{Hik} = s_k - s_i \quad (2.6)$$

mit

$s_i$  Weg des Abtriebsgliedes zu Beginn des Bewegungsabschnittes

$s_k$  Weg des Abtriebsgliedes am Ende des Bewegungsabschnittes

Die laufenden Winkel- und Wegkoordinaten eines Abschnittes sind:

$$\varphi_{ik} = \varphi - \varphi_i \quad (2.7)$$

$$s_{ik} = s - s_i \quad (2.8)$$

Nach Auswahl der Bewegungsgesetze für die einzelnen Abschnitte ergibt sich aus dem Bewegungsplan ein Bewegungsdiagramm (Abb. 3).

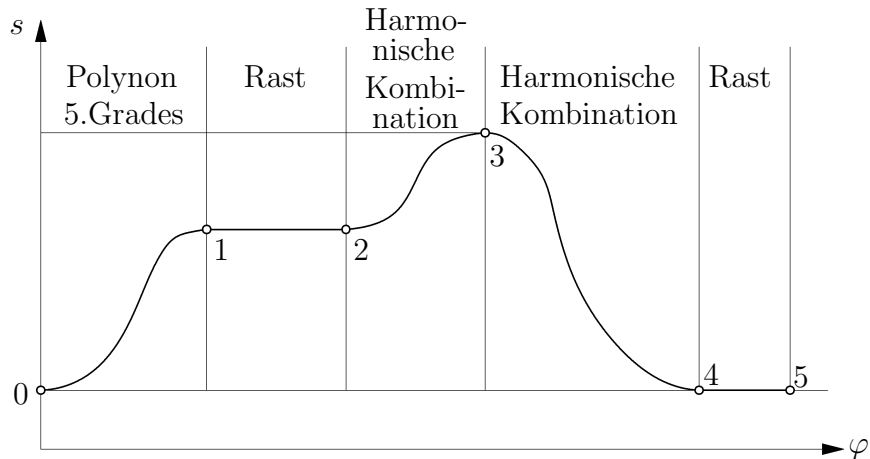


Abbildung 3: Bewegungsdiagramm [7]

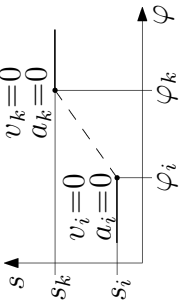
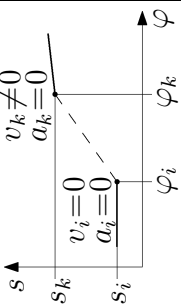
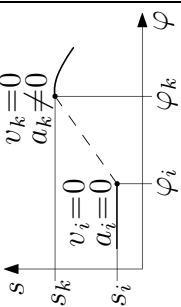
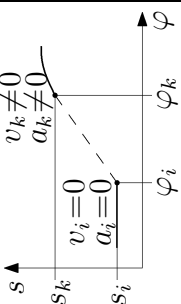
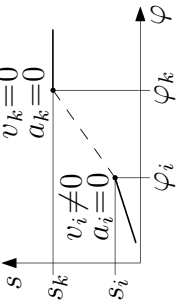
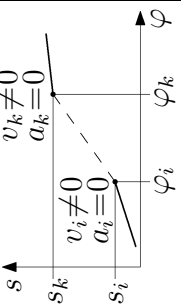
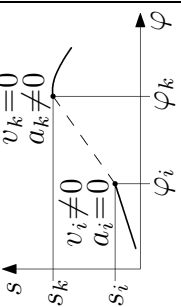
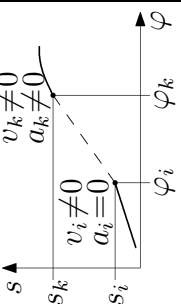
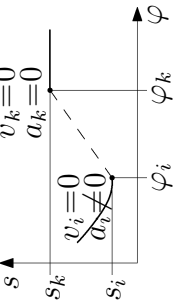
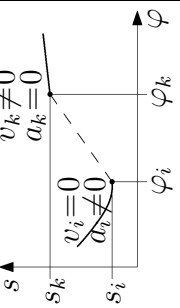
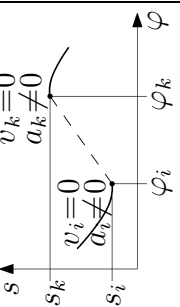
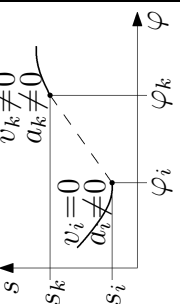
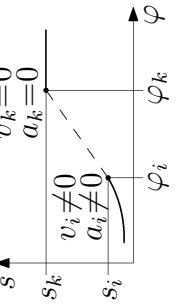
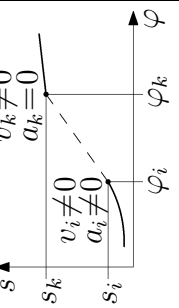
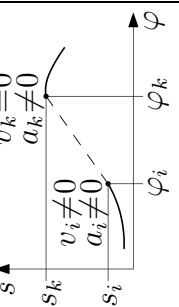
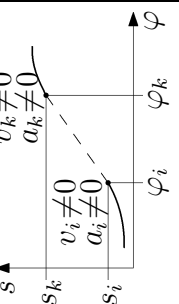
Die geforderte Abtriebsbewegung setzt sich aus verschiedenen Bewegungsaufgaben zusammen. Abhängig von Geschwindigkeit und Beschleunigung ergeben sich vier Typen von Bewegungsaufgaben (Tab. 1).

Tabelle 1: Bewegungsaufgaben [7]

Geschwindigkeit $v$ und Beschleunigung $a$ am Randpunkt eines Bewegungsabschnittes	Bewegungsaufgabe	Abkürzung
$v = 0; a = 0$	Rast	R
$v \neq 0; a = 0$	konstante Geschwindigkeit	G
$v = 0; a \neq 0$	Umkehr	U
$v \neq 0; a \neq 0$	Bewegung	B

Einen Überblick über die Kombination von Bewegungsaufgaben gibt Tabelle 2.

Tabelle 2: Mögliche Kombinationen von Bewegungsaufgaben [7]

in Übergang von	Rast	konstante Geschwindigkeit	Umkehr	Bewegung
Rast	 <p><b>R-R</b></p>	 <p><b>R-G</b></p>	 <p><b>R-U</b></p>	 <p><b>R-B</b></p>
konstante Geschwin- digkeit	 <p><b>G-R</b></p>	 <p><b>G-G</b></p>	 <p><b>G-U</b></p>	 <p><b>G-B</b></p>
Umkehr	 <p><b>U-R</b></p>	 <p><b>U-G</b></p>	 <p><b>U-U</b></p>	 <p><b>U-B</b></p>
Bewegung	 <p><b>B-R</b></p>	 <p><b>B-G</b></p>	 <p><b>B-U</b></p>	 <p><b>B-B</b></p>



Besitzt das Bewegungsgesetz  $f_{ik}$  bei  $s_{Hik}/2$ , d. h. bei  $f_{ik} = 0.5$ , einen Wendepunkt so spricht man von einem symmetrischen Bewegungsgesetz (Abb. 5). Es gilt die Symmetriebeziehung:

$$f_{ik}(z_{ik}) = 1 - f_{ik}(1 - z_{ik}) \quad (2.13)$$

Liegt der Wendepunkt nicht bei  $s_{Hik}/2$  ist es ein unsymmetrisches Bewegungsgesetz (siehe Abschn. 2.3.2).

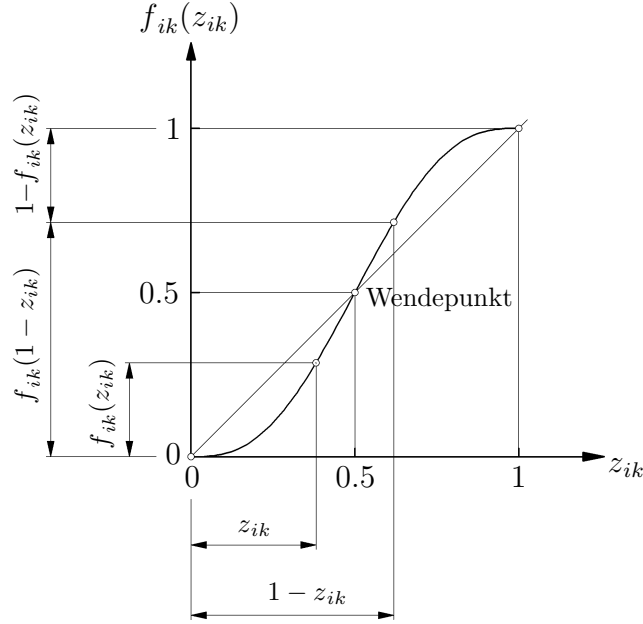


Abbildung 5: Normiertes symmetrisches Bewegungsgesetz [4]

Die normierten Bewegungsgesetze erster, zweiter und dritter Ordnung sind die entsprechenden Ableitungen des normierten Bewegungsgesetzes.

$$\frac{df_{ik}}{dz_{ik}} = f'_{ik} \quad (2.14)$$

$$\frac{d^2 f_{ik}}{dz_{ik}^2} = f''_{ik} \quad (2.15)$$

$$\frac{d^3 f_{ik}}{dz_{ik}^3} = f'''_{ik} \quad (2.16)$$

Kinematische Kennwerte wie z.B. Maximalgeschwindigkeit und Maximalbeschleunigung dienen als Kenngrößen zur Beurteilung der Bewegungsgesetze. Für geradlinig geführte Abtriebsglieder gilt:

$$s'_{ik} = C_v \frac{s_{Hik}}{\varphi_{Hik}} \quad (2.17)$$

$$s''_{ik} = C_a \frac{s_{Hik}}{2} \frac{1}{\varphi_{Hik}} \quad (2.18)$$

$$s'''_{ik} = C_j \frac{s_{Hik}}{3} \frac{1}{\varphi_{Hik}} \quad (2.19)$$

Hierin sind die Kennwerte  $C_v$ ,  $C_a$  und  $C_j$  die Maximalwerte der Ableitungen  $f'_{ik}$ ,  $f''_{ik}$  und  $f'''_{ik}$ .

- $C_v$  Geschwindigkeitskennwert ( $= f'_{ikmax}$ )
- $C_a$  Beschleunigungskennwert ( $= f''_{ikmax}$ )
- $C_j$  Ruckkennwert ( $= f'''_{ikmax}$ )

Die statischen Belastungen des Abtriebsgliedes sind geschwindigkeitsabhängig. Der Geschwindigkeitskennwert ist demnach auch als Kennwert für das Moment verwendbar.

$C_{Mstat}$  statischer Momentenkennwert ( $= C_v$ )

Dynamische Belastungen werden durch Trägheitskräfte  $m \cdot a$  hervorgerufen. Der Kennwert für den dynamischen Momentenverlauf ist das Produkt von Geschwindigkeitskennwert  $C_v$  und Beschleunigungskennwert  $C_a$ .

$C_{Mdyn}$  dynamischer Momentenkennwert ( $= C_v \cdot C_a$ )

### Beispiel: 3-4-5 Polynom

Das 3-4-5 Polynom ist ein Bewegungsgesetz, das gewährleistet, daß kein Ruck<sup>3</sup> am Abtriebsglied auftritt.

Die Weggleichung ist ein Polynom 5. Grades mit den entsprechenden Ableitungen.

$$f = 10z^3 - 15z^4 + 6z^5 \quad (2.20)$$

$$f' = 30z^2 - 60z^3 + 30z^4 \quad (2.21)$$

$$f'' = 60z - 180z^2 + 120z^3 \quad (2.22)$$

Die Verläufe sind für  $0 \leq z \leq 1$  in Bild 6 dargestellt.

---

<sup>3</sup>Beschleunigungssprung

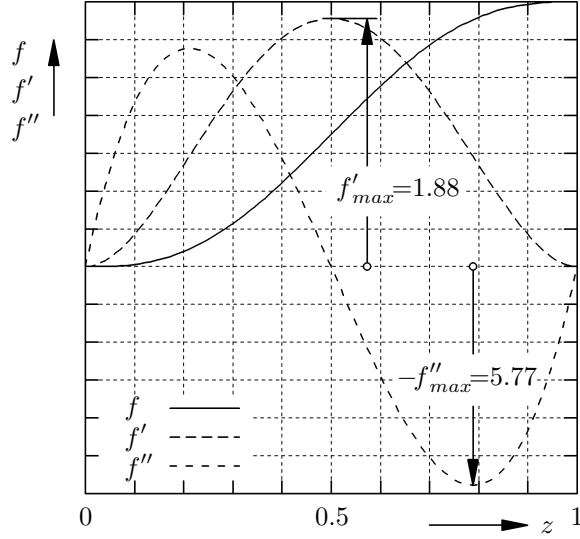


Abbildung 6: Symmetrisches 3-4-5 Polynom [5]

### 2.3.2 Unsymmetrische normierte Bewegungsgesetze für Rast in Rast

Bei unsymmetrischen Bewegungsgesetzen liegt eine Wendepunktverschiebung vor. Die Funktion setzt sich zur Hälfte aus einer verkleinerten und zur Hälfte aus einer vergrößerten symmetrischen Funktion zusammen (Abb. 7).

Im Bereich  $0 \leq z \leq \lambda$  ist das unsymmetrische Bewegungsgesetz:

$$f(z) = 2\lambda f(\bar{z}) \quad (2.23)$$

Die Bewegungsgesetze erster, zweiter und dritter Ordnung sind:

$$f'(z) = f'(\bar{z}) \quad (2.24)$$

$$f''(z) = f''(\bar{z}) \frac{1}{2\lambda} \quad (2.25)$$

$$f'''(z) = f'''(\bar{z}) \left[ \frac{1}{2\lambda} \right]^2 \quad (2.26)$$

Darin ist

$$\bar{z} = \frac{z}{2\lambda} \quad (2.27)$$



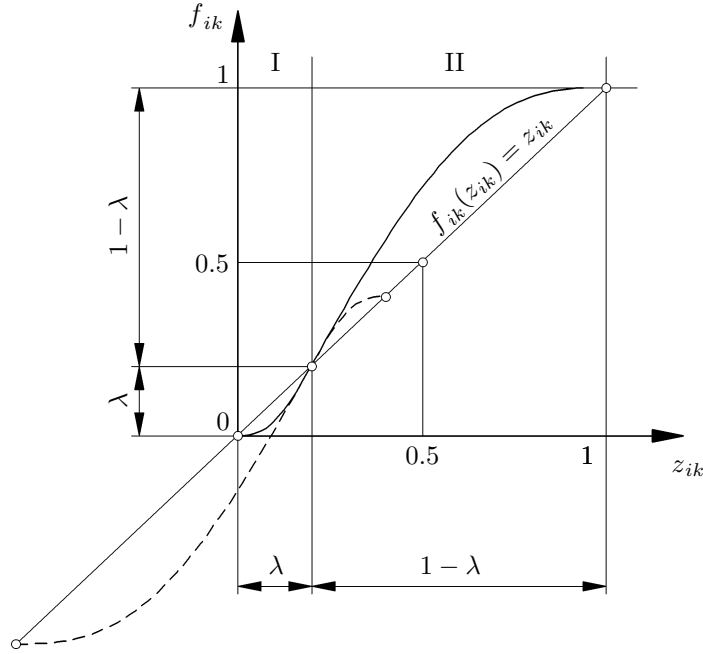


Abbildung 7: Normiertes unsymmetrisches Bewegungsgesetz [4]

Im Bereich  $\lambda \leq z \leq 1$  ist das unsymmetrische Bewegungsgesetz:

$$f(z) = \lambda + 2(1 - \lambda) [f(\bar{z} - 0.5)] \quad (2.28)$$

Die Bewegungsgesetze erster, zweiter und dritter Ordnung sind:

$$f'(z) = f'(\bar{z}) \quad (2.29)$$

$$f''(z) = f''(\bar{z}) \frac{1}{2(1 - \lambda)} \quad (2.30)$$

$$f'''(z) = f'''(\bar{z}) \left[ \frac{1}{2(1 - \lambda)} \right]^2 \quad (2.31)$$

mit

$$\bar{z} = 0.5 + \frac{z - \lambda}{2(1 - \lambda)} \quad (2.32)$$

### Beispiel: 3-4-5 Polynom

Für den Bereich  $0 \leq z \leq \lambda$  erhält man als Bewegungsgesetz:

$$\begin{aligned} f(z) &= 2\lambda f(\bar{z}) \\ &= 2\lambda(10\bar{z}^3 - 15\bar{z}^4 + 6\bar{z}^5) \\ &= \frac{5}{2\lambda^2} z^3 - \frac{15}{8\lambda^3} z^4 + \frac{3}{8\lambda^4} z^5 \end{aligned} \quad \left/ \bar{z} = \frac{z}{2\lambda} \right. \quad (2.33)$$

Mit gewählten  $\lambda$ , hier  $\lambda = 0.2$ , folgen das Bewegungsgesetz

$$f(z) = \frac{125}{2}z^3 - \frac{1875}{8}z^4 + \frac{1875}{16}z^5 \quad (2.34)$$

und die Ableitungen

$$f'(z) = \frac{375}{2}z^2 - \frac{1875}{2}z^3 + \frac{9375}{16}z^4 \quad (2.35)$$

$$f''(z) = 375z - \frac{5625}{2}z^2 + \frac{9375}{4}z^3 \quad (2.36)$$

Im Bereich  $\lambda \leq z \leq 1$  ergibt sich folgende Beziehung:

$$\begin{aligned} f(z) &= \lambda + 2(1 - \lambda) [f(\bar{z}) - 0.5] \\ &= \lambda + 2(1 - \lambda) [(10\bar{z}^3 - 15\bar{z}^4 + 6\bar{z}^5) - 0.5] \quad \Big/ \bar{z} = 0.5 + \frac{z - \lambda}{2(1 - \lambda)} \\ &= \frac{1}{8(\lambda - 1)^5} (7\lambda - 35\lambda^2 + 60\lambda^3 - 40\lambda^4 + 8\lambda^5 - 15z + 75\lambda z - 120\lambda^2 z \\ &\quad + 60\lambda^3 z - 30\lambda z^2 + 90\lambda^2 z^2 - 60\lambda^3 z^2 + 10z^3 - 30\lambda z^3 \\ &\quad + 20\lambda^3 z^3 + 15\lambda z^4 - 15\lambda^2 z^4 - 3z^5 + 3\lambda z^5) \end{aligned} \quad (2.37)$$

Mit gewählten  $\lambda$ , hier  $\lambda = 0.2$ , folgen das Bewegungsgesetz

$$f(z) = \frac{1}{2048} (-327 + 3375z + 2250z^2 - 3250z^3 - 1875z^4 + 1875z^5) \quad (2.38)$$

und die Ableitungen

$$f'(z) = \frac{1}{2048} (3375 + 4500z - 9750z^2 - 7500z^3 + 9375z^4) \quad (2.39)$$

$$f''(z) = \frac{1}{2048} (4500 - 19500z - 22500z^2 + 37500z^3) \quad (2.40)$$

Die Verläufe des zusammengesetzten Bewegungsgesetzes und seine Ableitungen sind für eine Wendepunktverschiebung von  $\lambda = 0.2$  im Bereich  $0 \leq z \leq 1$  in Bild 8 dargestellt.

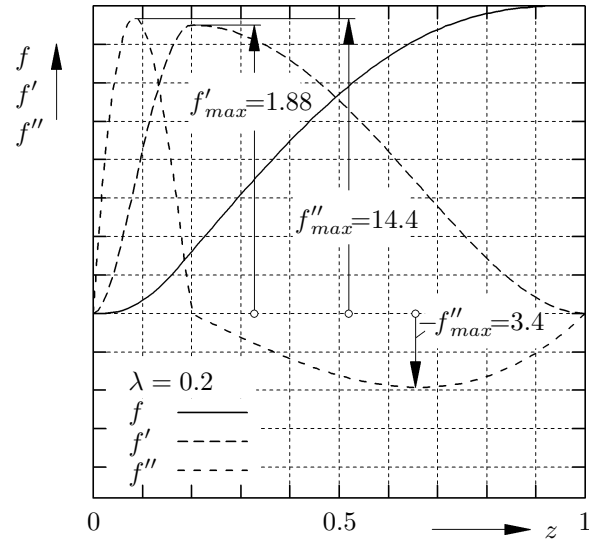


Abbildung 8: Unsymmetrisches 3-4-5 Polynom

## 2.4 Kinematische Abmessungen bei Kurvengetrieben

Bei der Wahl der Abmessungen sollte nach Möglichkeit ein günstiger Übertragungswinkel  $\mu$  erreicht werden. Er ist ein Maß für die Güte der Kraftübertragung. Es ist der spitze Winkel zwischen der Tangente  $t_a$  an die Absolutbahn des Abtriebsgliedes und der Tangente  $t_r$  an die Relativbahn des Übertragungsgliedes gegenüber dem Antriebsglied (Abb. 9).

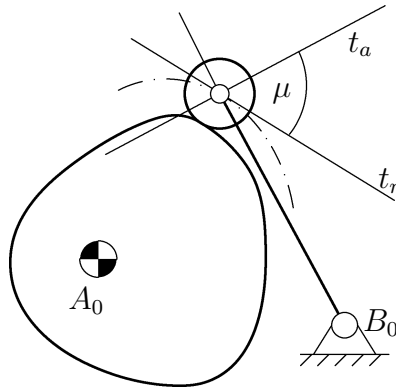


Abbildung 9: Übertragungswinkel  $\mu$  [4]

Günstige Übertragungswinkel liegen vor, wenn er bei langsamlaufenden Getrieben ( $n < 30\text{min}^{-1}$ ) *nicht kleiner als*  $45^\circ$  und bei schnelllaufenden Getrieben ( $n > 30\text{min}^{-1}$ ) *nicht kleiner als*  $60^\circ$  wird. Ausnahme bilden nur Kurvengetriebe mit Rollenstößel. Bei ihnen sollte der Übertragungswinkel  $\mu$  auch bei Drehzahlen von  $n < 30\text{min}^{-1}$  nicht kleiner als  $60^\circ$  werden.

Bei geschmierten Kontaktflächen dürfen diese Werte ggfs. unterschritten werden. Neben dem Übertragungswinkel  $\mu$  sind noch folgende Abmessungen festzulegen.

- Kurvengetriebe mit Schwinghebel (Abb. 10)

Gestelllänge  $l_1 (= \overline{A_0 B_0})$

Rollenhebellänge, Schwingenlänge  $l_3 (= \overline{B B_0})$

Grundkreisradius  $r_G$

Grundkreis  $k_G$

Grundwinkel  $\psi_G$

- Kurvengetriebe mit Schieber (Abb. 11)

Exzentrizität  $e$

Grundhub  $s_G$

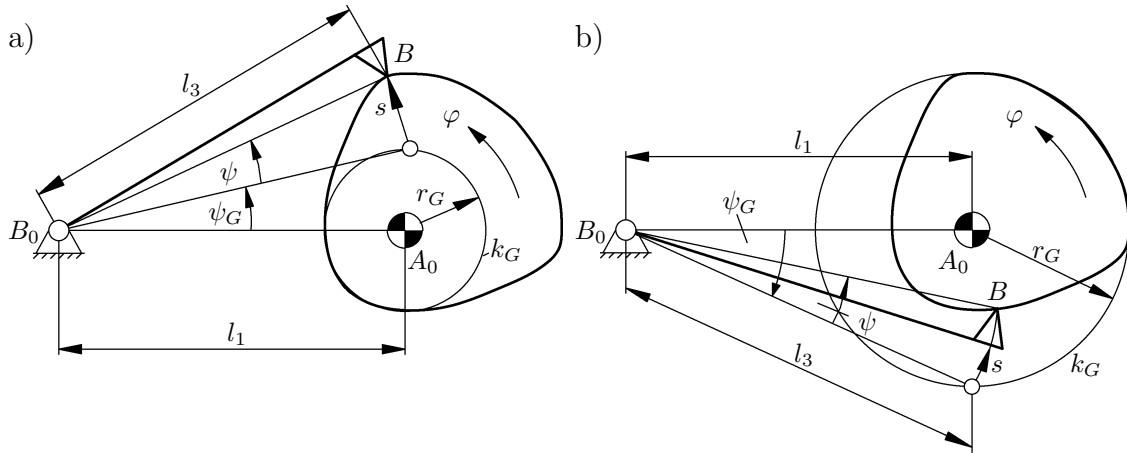


Abbildung 10: Kurvengetriebe mit Schwinghebel a) F-Kurvengetriebe b) P-Kurvengetriebe [4]

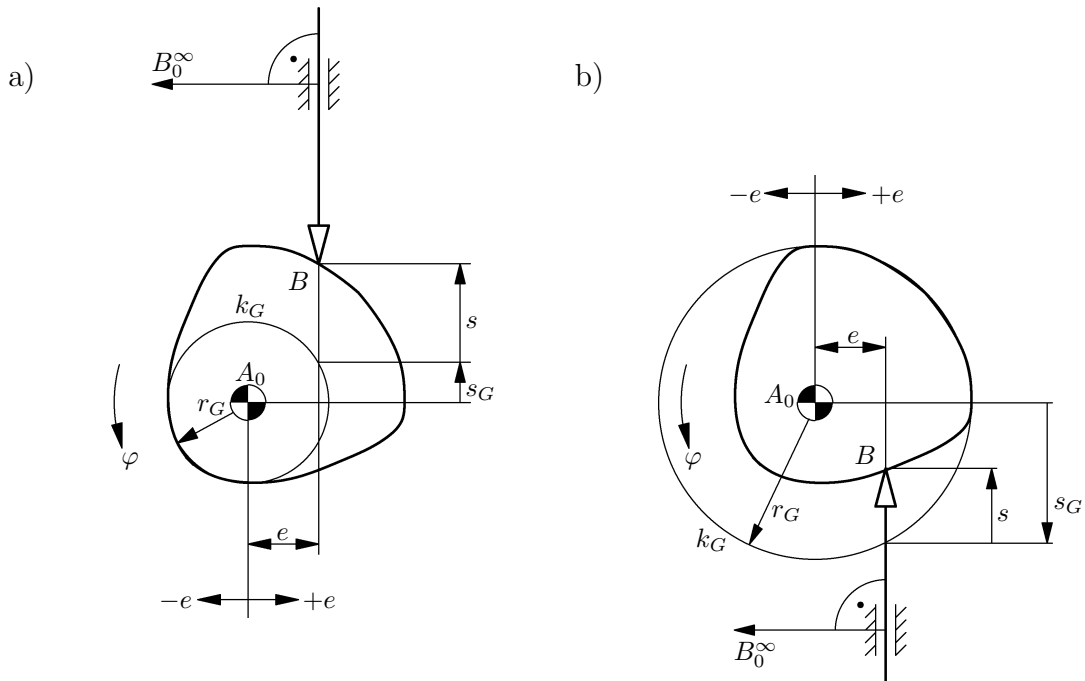


Abbildung 11: Kurvengetriebe mit Schieber a) F-Kurvengetriebe b) P-Kurvengetriebe [4]

Die Unterscheidung von F-Kurvengetriebe und P-Kurvengetriebe bezieht sich auf die Bewegung des Punktes  $B$  gegenüber dem Gestellpunkt  $A_0$  aus der Grundstellung<sup>4</sup> heraus.

Bewegt sich  $B$  von  $A_0$  weg, ist es ein F-Kurvengetriebe<sup>5</sup>. Bewegt er sich zu  $A_0$  hin,

<sup>4</sup>Ausgangsposition von  $B$  ( $\varphi = 0$ ,  $s = 0$  im Bewegungsdiagramm)

<sup>5</sup>Zentrifugalbewegung

ist es ein P-Kurvengetriebe <sup>6</sup>.

Damit sind für die Kurvengetriebe die für die Auslegung notwendigen Größen definiert.

### 2.4.1 Ermittlung des Grundkreisradius

Der Grundkreisradius  $r_G$  kann näherungsweise mit dem Verfahren nach *Flocke* bestimmt werden.

Dazu sind folgende Schritte notwendig (Abb. 12):

- Abtriebsglied (z.B. Schwinghebel, Schieber) in den Endlagen aufzeichnen
- maximale Längen  $\langle v_{B \max} \rangle_P$  bzw.  $\langle v_{B \max} \rangle_N$  berechnen

a) Kurvengetriebe mit Schwinghebel:

$$\langle v_{B \max} \rangle_{P/N} = \pm \frac{\langle \psi_H \rangle}{\varphi_{HP/N}} \cdot C_{vP/N} \quad (2.41)$$

a) Kurvengetriebe mit Schieber:

$$\langle v_{B \max} \rangle_{P/N} = \pm \frac{\langle s_H \rangle}{\varphi_{HP/N}} \cdot C_{vP/N} \quad (2.42)$$

Darin sind

$\langle v_{B \max} \rangle_{P/N}$	darstellende Größe <sup>7</sup> der maximalen Geschwindigkeit des Punktes B für Gleich- bzw. Gegenlauf
$\psi_H$	Gesamthubwinkel
$s_H$	Gesamthubweg
$\varphi_{P/N}$	Drehwinkel für Gleich- bzw. Gegenlauf
$C_{vP/N}$	Geschwindigkeitskennwert ( $= f'_{max}(z)$ ) für Gleich- bzw. Gegenlauf abhängig vom jeweils gewählten Bewegungsgesetz $f_{ik}(z_{ik})$

- Längen  $\langle v_{B \max} \rangle_P$  und  $\langle v_{B \max} \rangle_N$  im Rollenmittelpunkt um 90° gedreht<sup>8</sup> antragen
- in die Spitzen der gedrehten Geschwindigkeiten  $\langle \overline{v}_{B \max} \rangle_P$  und  $\langle \overline{v}_{B \max} \rangle_N$  den geforderten minimalen Übertragungswinkel  $\mu_{min}$  an beide Seiten auftragen und den Drehpunkt  $A_0$  in die dabei entstehenden  $A_0$ -Bereiche legen
- der Grundkreisradius  $r_G$  ist der Abstand zwischen  $A_0$  und  $B_a$

---

<sup>6</sup>Zentripetalbewegung

<sup>7</sup>darstellende Größe = Maßstab · wirkliche Größe

<sup>8</sup>Drehsinn des Antriebswinkels  $\varphi$

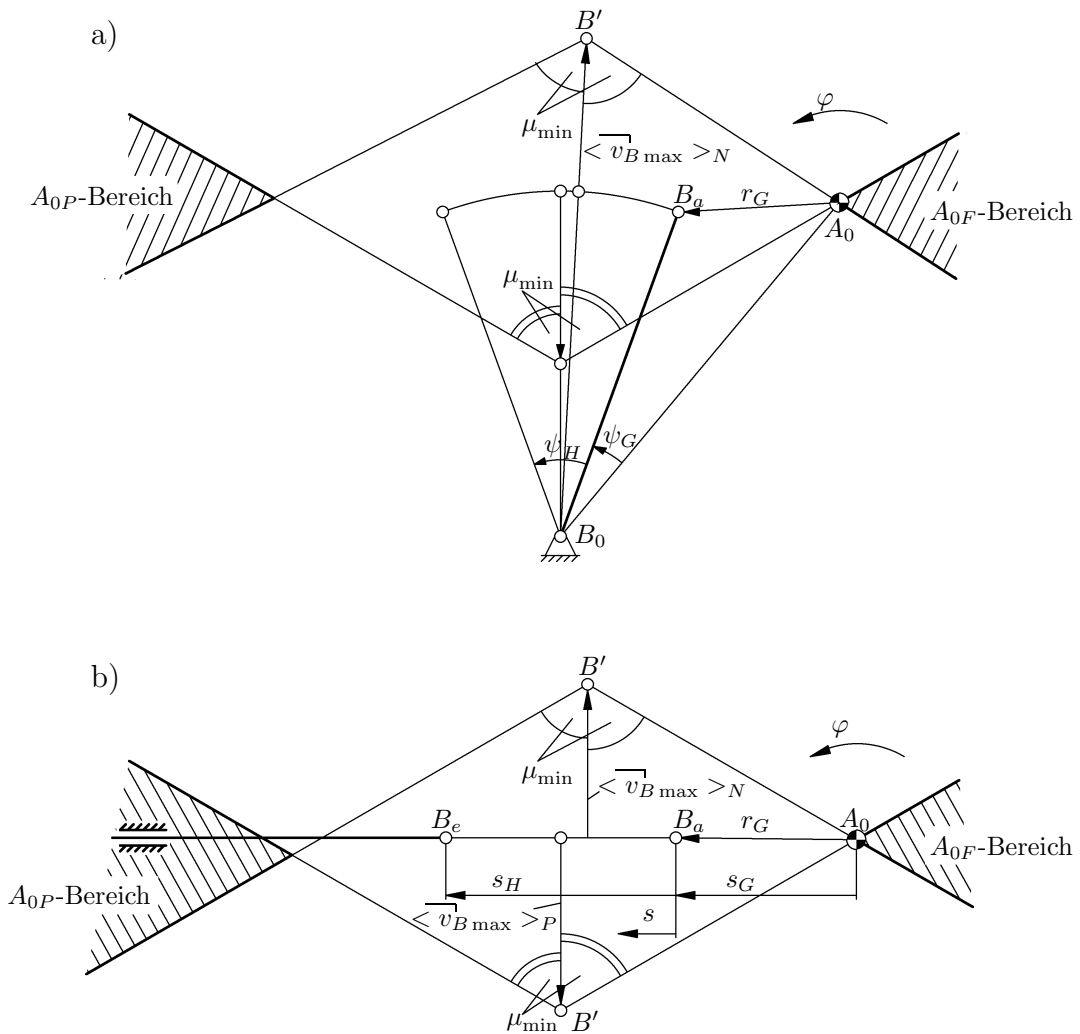


Abbildung 12: Ermittlung der  $A_0$ -Bereiche nach Flocke für Kurvengetriebe mit a) Schwinghebel b) Schieber [4]

Der Berechnung der Rollenmittelpunktskurve wird die Abbildung 13 zugrunde gelegt. Dabei ist nach dem Prinzip der kinematischen Umkehrung die Kurvenscheibe feststehend und das Gestell dreht sich im Sinne von  $-\varphi$  um  $A_0$ .

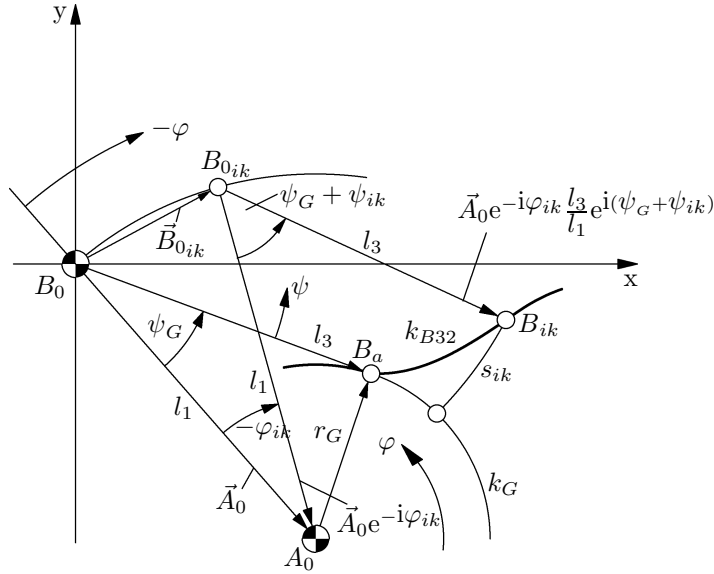


Abbildung 13: Grundfigur zur Berechnung der Rollenmittelpunktskurve bei einem Kurvengetriebe mit Schwinghebel [4]

Ausgehend von den Koordinaten der Punkte  $A_0(x_{A_0}, y_{A_0})$ ,  $B_a(x_{B_a}, y_{B_a})$  und  $B_{ik}(x_{B_{ik}}, y_{B_{ik}})$  können folgende Hauptabmessungen berechnet werden:

$$\left. \begin{aligned} l_1 &= \sqrt{x_{A0}^2 + y_{A0}^2} \\ r_G &= \sqrt{(x_{Ba} - x_{A0})^2 + (y_{Ba} - y_{A0})^2} \\ \psi_G &= \arccos [(l_3^2 + l_1^2 - r_G^2) / (2l_1l_3)] \end{aligned} \right\} \quad (2.43)$$

Damit kann die Kurve  $k_{B32}$  punktweise unter Vorgabe des Kurvenscheibendrehwinkels  $-\varphi$  berechnet werden. Für den Vektor des Punktes  $B_{0ik}$  gilt:

$$\vec{B}_{0_{ik}} = \vec{A}_0 - \vec{A}_0 e^{-i\varphi_{ik}} \quad (2.44)$$

Für den Vektor  $\vec{B}_{ik}$  auf der Mittelpunktskurve gilt:

$$\vec{B}_{ik} = \vec{B}_{0ik} + \vec{A}_0 e^{-i\varphi_{ik}} \frac{l_3}{l_1} e^{i(\psi_G + \psi_{ik})} \quad (2.45)$$



Gl. 2.44 in Gl. 2.45 Einsetzen ergibt:

$$\begin{aligned}\vec{B}_{ik} &= \vec{A}_0 - \vec{A}_0 e^{-i\varphi_{ik}} + \vec{A}_0 e^{-i\varphi_{ik}} \frac{l_3}{l_1} e^{i(\psi_G + \psi_{ik})} \\ &= \vec{A}_0 \left[ 1 - e^{-i\varphi_{ik}} + \frac{l_3}{l_1} e^{i(\psi_G + \psi_{ik} - \varphi_{ik})} \right]\end{aligned}\quad (2.46)$$

Unter Anwendung der kartesischen Darstellung  $\vec{Z} = x_Z + iy_Z$  und der Eulerschen Formeln  $e^{i\varphi} = \cos \varphi + i \sin \varphi$ ,  $e^{-i\varphi} = \cos \varphi - i \sin \varphi$  auf Gleichung 2.46 folgt daraus:

$$\begin{aligned}x_{Bik} + iy_{Bik} &= (x_{A0} + iy_{A0}) \left\{ 1 - (\cos \varphi_{ik} - i \sin \varphi_{ik}) + \frac{l_3}{l_1} [\cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right. \\ &\quad \left. + i \sin(\psi_G + \psi_{ik} - \varphi_{ik})] \right\}\end{aligned}\quad (2.47)$$

Hieraus sind die Koordinaten des Vektors  $\vec{B}_{ik}$  ablesbar:

$$\begin{aligned}x_{Bik} &= x_{A0} \left[ 1 - \cos \varphi_{ik} + \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ &\quad - y_{A0} \left[ \sin \varphi_{ik} + \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right]\end{aligned}\quad (2.48)$$

$$\begin{aligned}y_{Bik} &= x_{A0} \left[ \sin \varphi_{ik} + \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ &\quad + y_{A0} \left[ 1 - \cos \varphi_{ik} + \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right]\end{aligned}\quad (2.49)$$

Zur Berechnung des Kurvenprofils (innere und äußere Arbeitskurve) wird zunächst der Tangentenvektor  $\vec{B}'_{ik}$  durch Differentiation von Gl. 2.46 nach  $\varphi_{ik}$  gebildet:

$$\vec{B}'_{ik} = \frac{d\vec{B}_{ik}}{d\varphi_{ik}} = i\vec{A}_0 \left[ e^{-i\varphi_{ik}} + \frac{l_3}{l_1} (\psi'_{ik} - 1) e^{i(\psi_G + \psi_{ik} - \varphi_{ik})} \right]\quad (2.50)$$

Die Ableitungen der Koordinaten  $x_{Bik}$  und  $y_{Bik}$  lauten:

$$\begin{aligned}x'_{Bik} &= x_{A0} \left[ \sin \varphi_{ik} - (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ &\quad - y_{A0} \left[ \cos \varphi_{ik} + (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right]\end{aligned}\quad (2.51)$$

$$\begin{aligned}y'_{Bik} &= x_{A0} \left[ \cos \varphi_{ik} + (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ &\quad + y_{A0} \left[ \sin \varphi_{ik} - (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right]\end{aligned}\quad (2.52)$$

Die Vektoren  $\vec{B}_{ik}^i$  und  $\vec{B}_{ik}^a$  von innerer und äußerer Arbeitskurve sind:

$$\vec{B}_{ik}^i = \vec{B}_{ik} + ir_R \frac{\vec{B}'_{ik}}{|\vec{B}'_{ik}|} \quad (2.53)$$

$$\vec{B}_{ik}^a = \vec{B}_{ik} - ir_R \frac{\vec{B}'_{ik}}{|\vec{B}'_{ik}|} \quad (2.54)$$

Hierin ist  $r_R$  der Laufrollenradius.

In der Ebene gilt für den Betrag  $|\vec{B}'_{ik}|$  folgende Gleichung:

$$|\vec{B}'_{ik}| = \sqrt{x_{Bik}^{\prime 2} + y_{Bik}^{\prime 2}} \quad (2.55)$$

Die Abbildung 14 zeigt die zur Berechnung der Rollenmittelpunktsbahn verwendete Grundfigur für Kurvengetriebe mit Schieber.

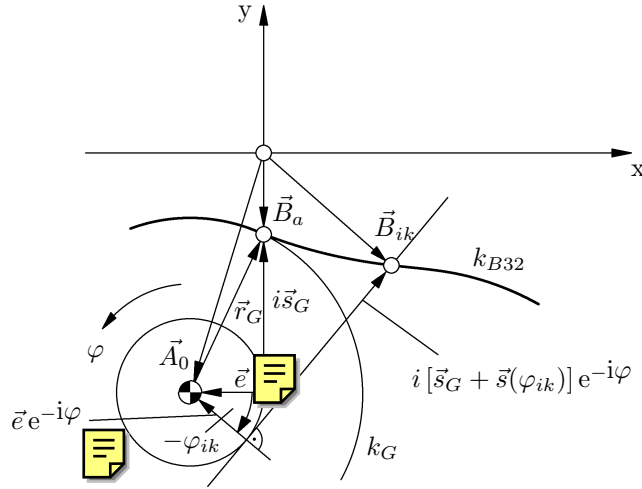


Abbildung 14: Grundfigur zur Berechnung der Rollenmittelpunktskurve bei einem Kurvengetriebe mit Schieber [4]

Zunächst berechnen wir den Grundkreisradius  $r_G$ , die Exzentrizität  $e$  und den Grundhub  $s_G$ .

$$\left. \begin{aligned} r_G &= \sqrt{(x_{Ba} - x_{A0})^2 + (y_{Ba} - y_{A0})^2} \\ e &= -x_{A0} \\ s_G &= \sqrt{r_G^2 - e^2} \end{aligned} \right\} \quad (2.56)$$

Die Kurvenscheibe betrachten wir wieder als feststehend und drehen das Gestell um  $-\varphi$ . Die Rollenmittelpunktskurve  $k_{B32}$  berechnen wir punktweise für entsprechende Winkelvorgaben  $\varphi_{ik}$ . Für die Vektoren  $\vec{B}_a$  und  $\vec{B}_{ik}$  gilt:

$$\vec{B}_a = \vec{A}_0 - \vec{e} + i\vec{s}_G \quad (2.57)$$

$$\vec{B}_{ik} = \vec{A}_0 - \vec{e} e^{-i\varphi_{ik}} + i [\vec{s}_G + \vec{s}(\varphi_{ik})] e^{-i\varphi} \quad (2.58)$$

Mit der umgestellten Gleichung 2.10 von Seite 7 ( $s_{ik} = s_{Hik} \cdot f(z_{ik})$ ) folgt daraus:

$$\vec{B}_{ik} = \vec{A}_0 - \vec{e} e^{-i\varphi_{ik}} + i [\vec{s}_G + \vec{s}_H \cdot f(z_{ik})] e^{-i\varphi} \quad (2.59)$$

Nach Einführung der kartesischen Darstellung  $\vec{Z} = x_Z + iy_Z$  und der Anwendung der Eulerschen Formeln  $e^{i\varphi} = \cos \varphi + i \sin \varphi$ ,  $e^{-i\varphi} = \cos \varphi - i \sin \varphi$  erhält:

$$\begin{aligned} x_{Bik} + iy_{Bik} &= x_{A0} + iy_{A0} + \left\{ -e + i[s_G + s_H f(z_{ik})] \right\} (\cos \varphi - i \sin \varphi) \\ &= x_{A0} + iy_{A0} + \left\{ -x_{A0} \cos \varphi + i \cos \varphi [s_G + s_H f(z_{ik})] \right. \\ &\quad \left. + ix_{A0} \sin \varphi + \sin \varphi [s_G + s_H f(z_{ik})] \right\} \end{aligned} \quad (2.60)$$

Die daraus abgelesenen Koordinaten des Vektors  $\vec{B}_{ik}$  lauten:

$$x_{Bik} = x_{A0}(1 - \cos \varphi) + \sin \varphi [s_G + s_H f(z_{ik})] \quad (2.61)$$

$$y_{Bik} = y_{A0} + x_{A0} \sin \varphi + \cos \varphi [s_G + s_H f(z_{ik})] \quad (2.62)$$

Die Ableitung von Gleichung 2.59 ist der Tangentenvektor  $\vec{B}'_{ik}$  der Rollenmittelpunktskurve.

$$\vec{B}'_{ik} = e^{-i\varphi} [s_G + s(\varphi)] - ie^{-i\varphi} (e - s') \quad (2.63)$$

Die Arbeitskurvenberechnung erfolgt wieder mit den Beziehungen 2.53 und 2.54.

## 3 Programmentwurf

### 3.1 Auswahl der Programmiersprache

Nach [10] sind die Programmiersprachen durch folgende Qualitätsmaßstäbe zu bewerten:

- Korrektheit (exakte Erfüllung der Aufgabe)
- Adaptierbarkeit (Anpassung an ähnliche Probleme)
- Portabilität (Anpassung an andere Betriebssysteme)
- Kompatibilität (Kombinierbarkeit von Teilsystemen)
- Zuverlässigkeit (Wahrscheinlichkeit für befriedigende Ausführung)
- Robustheit (Verkräften von Fehlbedienung)
- Verfügbarkeit (Ausfall- bzw. Standzeiten)
- Benutzerfreundlichkeit (bewertet Schnittstelle Benutzer  $\Leftrightarrow$  Programm)
- Wartbarkeit (Fehlerbeseitigung und funktionale Erweiterung/ Anpassung)
- Effizienz und Leistung (bewertet Nutzung aller Betriebsmittel (wie Speicher und Rechenzeit))

Diese Forderungen sind i. allg. nicht alle gleichzeitig von einer Programmiersprache erfüllbar. Die Auswahl der Programmiersprache hängt also davon ab, auf welche Anforderungen für das Projekt besonderer Wert gelegt werden muß. In Tabelle 3 sind wesentliche Vor- und Nachteile von bekannten Programmiersprachen aufgelistet.

Da es eine Vielzahl von Betriebssystemen gibt, ist es zeitgemäß, portierbare Software zu schreiben. Weiterhin ist es sinnvoll eine Programmiersprache zu verwenden, die weit verbreitet ist. Es ist dadurch eher wahrscheinlich, daß schon andere Software in dieser Sprache existiert. Die Kombination dieser Module ist dann problemlos möglich.

Diese Gründe haben die Wahl auf C/C++ fallen lassen. Wie die Tabelle 3 zeigt, sind mit C/C++ fast alle Problemstellungen lösbar. Dadurch gibt es keine „Sackgassen“, die auf die Programmiersprache zurückzuführen sind.

Die Tabelle zeigt ebenfalls den Nachteil, daß Oberflächen nur beschränkt plattformunabhängig sind. Die Lösung für dieses Problem ist die von der norwegischen Firma *Trolltech* entwickelte C++ Klassenbibliothek Qt.

Tabelle 3: Vor- und Nachteile von Programmiersprachen

	C/C++	Delphi	Java	Visual Basic
Vorteile	in Industrie am weitesten verbreitet	geringer Lernaufwand (Tage)	plattformunabhängig	einfach für die Oberflächenprogrammierung
	sehr schnell		robust	
	damit kann fast alles programmiert werden			
	Grundsprache plattformunabhängig			
	Mehrfachvererbung			
Nachteile	GUI-sprache nur beschränkt plattformunabhängig	beschränkt plattformunabhängig	langsam	nicht plattformunabhängig
	hoher Lernaufwand (Monate)	nur Einfachvererbung	hoher Lernaufwand (Monate)	
			nur Einfachvererbung	

### 3.2 Wirkungsweise und Funktionsumfang des Programms

Im Entwurfsstadium bietet es sich an, die Wirkungsweise und den Funktionsumfang anhand von Struktogrammen<sup>9</sup> darzustellen. Auf diese Weise erhält man einen besseren Überblick über die Handlungen, die mit den Eingabedaten durchgeführt werden sollen, um zu den gewünschten Ergebnissen zu kommen.

Abbildung 15 gibt eine Übersicht über die Grundsymbole der Programmstrukturen.

Das mit dieser Methode erstellte Struktogramm ist in Abbildung 16 dargestellt.

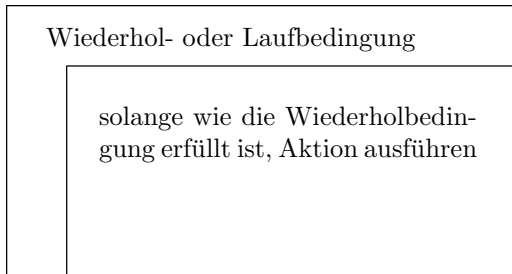
Nach dem Programmstart soll der Anwender des Programms wählen können, ob er ein neues Projekt erstellen oder ob er ein vorhandenes Projekt öffnen möchte. Weiterhin muß auch die Möglichkeit bestehen, das Programm zu beenden.

Wählt der Anwender aus dieser Mehrfachverzweigung einen Fall aus, dann werden die Schritte dieses Zweiges ausgeführt. Dabei unterscheiden sich die beiden Berechnungszweige nur durch die Zuführung der Eingabeparameter.

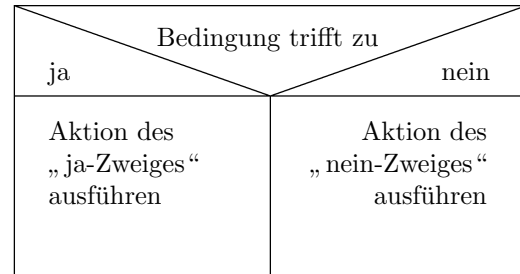
Während der gesamten Laufzeit des Programms soll es möglich sein, das Pro-

<sup>9</sup>nach Nassi Schneidermann (DIN 66261)

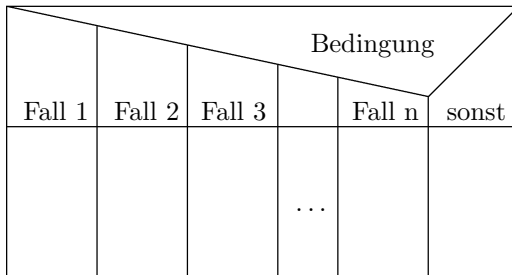
a) Wiederholung



b) Alternative



c) Mehrfachverzweigung



d) Reihung

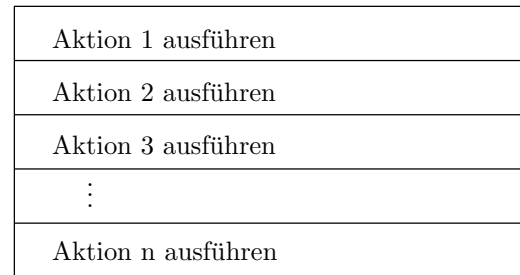


Abbildung 15: Symbole für Programmstrukturen a) Wiederholung b) Alternative c) Mehrfachverzweigung d) Reihung [2]

programm zu beenden. Dies kommt durch die Wiederholung zum Ausdruck.

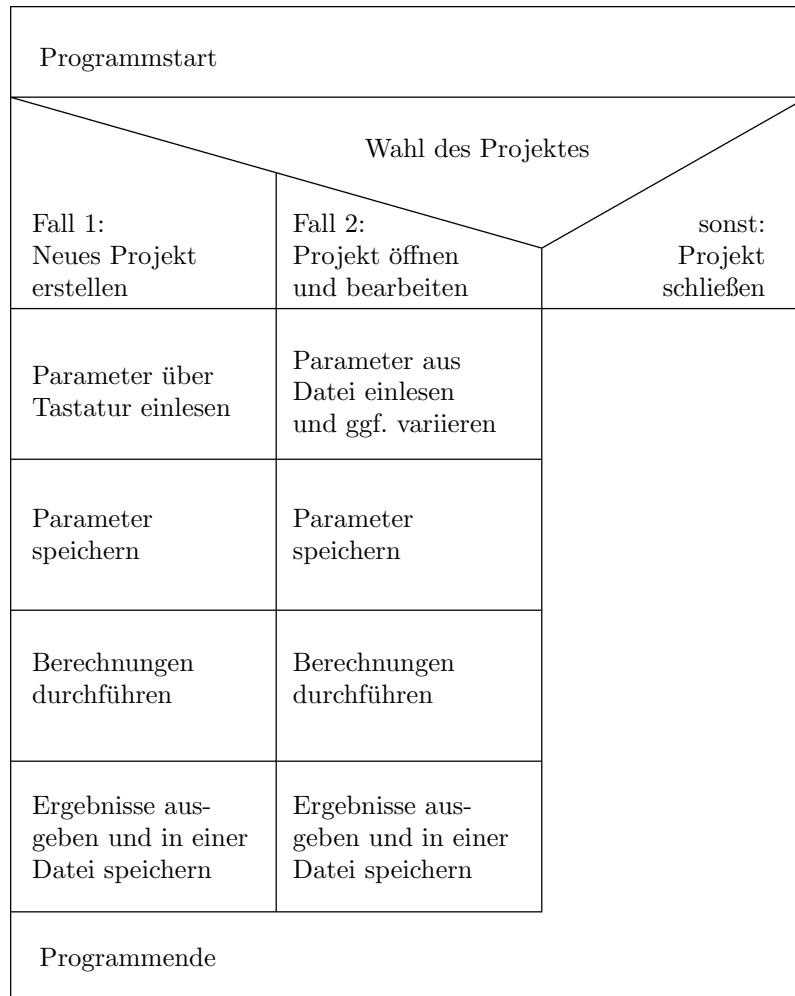


Abbildung 16: Struktogramm des Hauptprogramms

Dieser Abschnitt beschreibt die Ermittlung der im Programm verwendeten Formeln nach Bild 17.



Für den Punkt  $D_{ik}$  ergibt sich folgende Vektorgleichung:

Umstellung und Bildung der konjugiert komplexen Terme ergibt:

$$l_4 e^{-i\vartheta} = s_0 \pm s(\varphi) - i l_{sk} \cos \alpha_{sk} - l_{sk} e^{-i(\beta+\psi)} \quad (3.3)$$

$$l_4^2 = [s_0 \pm s(\varphi)]^2 + l_{sk}^2 \cos^2 \alpha_{sk} + l_{sk}^2 - 2l_{sk} \cdot \cos(\psi + \beta) \cdot [s_0 \pm s(\varphi)] + 2l_{sk} \cdot \sin(\psi + \beta) \cdot l_{sk} \cos \alpha_{sk} \quad (3.4)$$

26



Nach dem Umstellen von  $l_4^2$  und dem Einführen der Substitutionen

$$\left. \begin{aligned} A &= -2l_{sk} \cdot [s_0 \pm s(\varphi)] \\ B &= +2l_{sk} \cdot l_{sk} \cos \alpha_{sk} \\ C &= [s_0 \pm s(\varphi)]^2 + l_{sk}^2 \cos^2 \alpha_{sk} + l_{sk}^2 - l_4^2 \end{aligned} \right\} \quad (3.5)$$

folgt daraus die vereinfachte Form der Übertragungsleichung 0. Ordnung:

$$0 = A \cos(\psi + \beta) + B \sin(\psi + \beta) + C \quad (3.6)$$

Durch Anwendung der Theoreme

$$\cos(\psi + \beta) = \frac{1 - \tan^2 \frac{\psi + \beta}{2}}{1 + \tan^2 \frac{\psi + \beta}{2}} \quad \sin(\psi + \beta) = \frac{2 \tan \frac{\psi + \beta}{2}}{1 + \tan^2 \frac{\psi + \beta}{2}} \quad (3.7)$$

und die Multiplikation der Gleichung mit dem Hauptnenner  $1 + \tan^2[(\psi + \beta)/2]$  erhält man

$$\begin{aligned} 0 &= A \left( 1 - \tan^2 \frac{\psi + \beta}{2} \right) + 2B \tan \frac{\psi + \beta}{2} + C \left( 1 + \tan^2 \frac{\psi + \beta}{2} \right) \\ &= (C - A) \tan^2 \frac{\psi + \beta}{2} + 2B \tan \frac{\psi + \beta}{2} + A + C \quad / \div (C - A) \\ &= \tan^2 \frac{\psi + \beta}{2} - \frac{2B}{A - C} \tan \frac{\psi + \beta}{2} - \frac{A + C}{A - C} \end{aligned} \quad (3.8)$$

Gl. (3.8) ist eine quadratische Gleichung. Es gilt

$$\left( \tan \frac{\psi + \beta}{2} \right)_{1/2} = \frac{B}{A - C} \pm \sqrt{\frac{B^2}{(A - C)^2} - \frac{A + C}{A - C}} \quad (3.9)$$

Bildet man für den Ausdruck unter der Klammer den Hauptnenner, so läßt sich dieser vor die Wurzel ziehen

$$\begin{aligned} \left( \tan \frac{\psi + \beta}{2} \right)_{1/2} &= \frac{B}{A - C} \pm \sqrt{\frac{B^2}{(A - C)^2} - \frac{(A + C)(A - C)}{(A - C)^2}} \\ &= \frac{B}{A - C} \pm \frac{1}{A - C} \sqrt{A^2 + B^2 - C^2} \\ &= \frac{B \pm \sqrt{A^2 + B^2 - C^2}}{A - C} \end{aligned} \quad (3.10)$$

Hiervon ist der arctan zu bilden

$$\begin{aligned} \frac{\psi + \beta}{2} &= \arctan \frac{B \pm \sqrt{A^2 + B^2 - C^2}}{A - C} \quad / \cdot 2; -\beta \\ \psi &= 2 \arctan \frac{B \pm \sqrt{A^2 + B^2 - C^2}}{A - C} - \beta \end{aligned} \quad (3.11)$$

Dies ist die *Übertragungsfunktion 0. Ordnung*  $\psi = \psi(\varphi)$ . Darin sind  $l_{sk}$ ,  $s_0$ ,  $\alpha_{sk}$  und  $l_4$  geometrische Größen, die für das zu berechnende Getriebe bekannt sein müssen. Als Bewegungsgesetz für den Hubverlauf  $s(\varphi)$  wird das *3-4-5 Polynom* verwendet. In Abschnitt 3.3.2 auf Seite 28 wird auf alle notwendigen Gleichungen für die Bewegungsabschnitte eingegangen.

Eine Differentiation von Gl. (3.4) nach dem Drehwinkel  $\varphi$  ergibt:

$$0 = 2l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) \psi' + 2l_{sk} [s_0 \pm s(\varphi)] \sin(\psi + \beta) \psi' - 2l_{sk} \cos(\psi + \beta) \cdot [\pm s'(\varphi)] + 2[s_0 \pm s(\varphi)] \cdot [\pm s'(\varphi)] \quad (3.12)$$

Auflösen nach  $\psi'$  ergibt die *Übertragungsfunktion 1. Ordnung*  $\psi' = \psi'(\varphi)$ .

$$\psi' = \frac{2l_{sk} \cos(\psi + \beta) \cdot [\pm s'(\varphi)] - 2[s_0 \pm s(\varphi)] \cdot [\pm s'(\varphi)]}{l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) + 2l_{sk} [s_0 \pm s(\varphi)] \sin(\psi + \beta)} \quad (3.13)$$

Diese Gleichung enthält neben den geometrischen Größen  $l_{sk}$ ,  $s_0$ ,  $\alpha_{sk}$ ,  $l_4$  und dem Hubverlauf  $s(\varphi)$  auch noch die Funktion  $\psi(\varphi)$ , die aber bereits bekannt ist, siehe Gl. (3.11). Für die *Übertragungsfunktion 2. Ordnung* wird die Gl. (3.12) nach dem Drehwinkel  $\varphi$  differenziert<sup>11</sup>

$$0 = -2l_{sk}^2 \cos \alpha_{sk} \sin(\psi + \beta) \cdot \psi'^2 + 2l_{sk} [s_0 \pm s(\varphi)] \cos(\psi + \beta) \cdot \psi'^2 + 4l_{sk} \sin(\psi + \beta) \cdot [\pm s'(\varphi)] \cdot \psi' + 2[\pm s'(\varphi)]^2 + 2l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) \cdot \psi'' + 2l_{sk} [s_0 \pm s(\varphi)] \sin(\psi + \beta) \cdot \psi'' - 2l_{sk} \cos(\psi + \beta) \cdot [\pm s''(\varphi)] + 2[s_0 \pm s(\varphi)] \cdot [\pm s''(\varphi)] \quad (3.14)$$

und diese Ableitung nach  $\psi''$  aufgelöst

$$\psi'' = \frac{2l_{sk}^2 \cos \alpha_{sk} \sin(\psi + \beta) \cdot \psi'^2 - [s_0 \pm s(\varphi)] \cos(\psi + \beta) \cdot \psi'^2}{2l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) + [s_0 \pm s(\varphi)] \sin(\psi + \beta)} + \frac{-\sin(\psi + \beta) \cdot [\pm s'(\varphi)] \cdot \psi' - [s'(\varphi)]^2}{2l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) + 2[s_0 \pm s(\varphi)] \sin(\psi + \beta)} + \frac{[\pm s''(\varphi)] - [s_0 \pm s(\varphi)] \cdot [\pm s''(\varphi)]}{2l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) + 2[s_0 \pm s(\varphi)] \sin(\psi + \beta)} \quad (3.15)$$

### 3.3.2 Bewegungsgesetz

In Bild 18 sind die vereinbarten Bewegungsabschnitte dargestellt.

In [4] ist die normierte Übertragungsfunktion für das 3-4-5 Polynom wie folgt angegeben:

$$f(z) = 10z^3 - 15z^4 + 6z^5 \quad (3.16)$$

<sup>11</sup>m. H. der Produktregel:  $y = u \cdot v \cdot u \rightsquigarrow y' = u'vw + uv'w + uvw'$

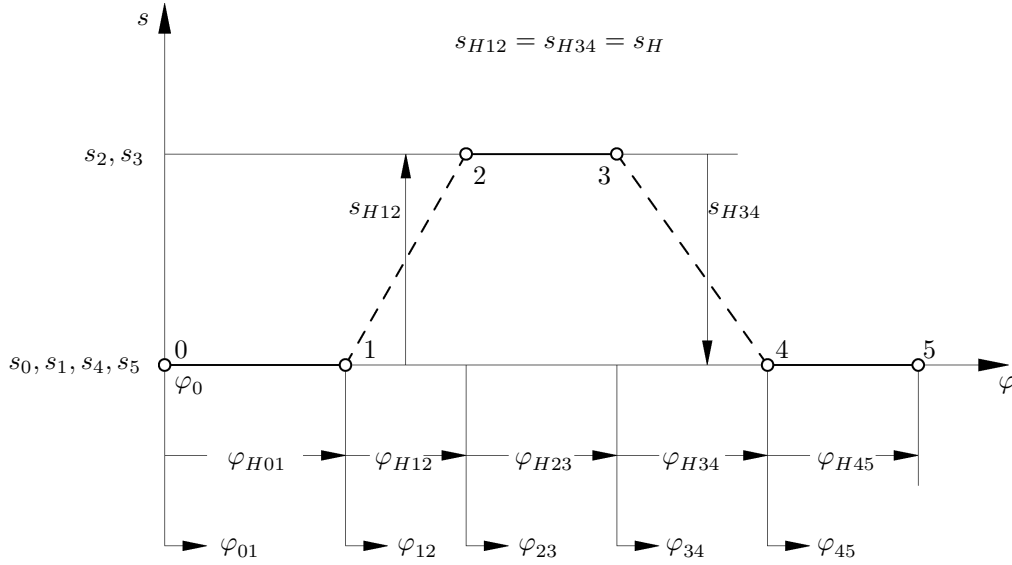


Abbildung 18: Bewegungsplan des Kurvenkoppelgetriebes

Mit den Gleichungen (2.9) und (2.10) von Seite 7 wird daraus

$$\frac{s_{ik}}{s_{Hik}} = 10 \frac{\varphi_{ik}^3}{\varphi_{Hik}^3} - 15 \frac{\varphi_{ik}^4}{\varphi_{Hik}^4} + 6 \frac{\varphi_{ik}^5}{\varphi_{Hik}^5} \quad \bigg/ \cdot s_{Hik}$$

$$s_{ik} = \frac{s_{Hik}}{\varphi_{Hik}^3} \left( 10 \varphi_{ik}^3 - 15 \frac{\varphi_{ik}^4}{\varphi_{Hik}} + 6 \frac{\varphi_{ik}^5}{\varphi_{Hik}^2} \right) \quad (3.17)$$

Für den Abschnitt 01 gilt:

$$\left. \begin{aligned} \varphi_0 &= 0 \\ \varphi_{01} &= \varphi - \varphi_0 = \varphi \\ s_{01} &= 0 \end{aligned} \right\} \quad (3.18)$$

Im Abschnitt 12 sind

$$\left. \begin{aligned} \varphi_1 &= \varphi_{H01} \\ \varphi_{12} &= \varphi - \varphi_1 \\ s_{12} &= \frac{s_H}{\varphi_{H12}^3} \left( 10 \varphi_{12}^3 - 15 \frac{\varphi_{12}^4}{\varphi_{H12}} + 6 \frac{\varphi_{12}^5}{\varphi_{H12}^2} \right) \end{aligned} \right\} \quad (3.19)$$

Weiterhin gilt

$$\left. \begin{aligned} \varphi_2 &= \varphi_{H01} + \varphi_{H12} \\ \varphi_{23} &= \varphi - \varphi_2 \\ s_{23} &= s_{H23} = s_H \end{aligned} \right\} \quad (3.20)$$

für den Abschnitt 23 und

$$\left. \begin{aligned} \varphi_3 &= \varphi_{H01} + \varphi_{H12} + \varphi_{H23} \\ \varphi_{34} &= \varphi - \varphi_3 \\ s_{34} &= s_H - \frac{s_H}{\varphi_{H34}^3} \left( 10\varphi_{34}^3 - 15\frac{\varphi_{34}^4}{\varphi_{H34}} + 6\frac{\varphi_{34}^5}{\varphi_{H34}^2} \right) \end{aligned} \right\} \quad (3.21)$$

für den Abschnitt 34. Im letzten Abschnitt sind

$$\left. \begin{aligned} \varphi_4 &= \varphi_{H01} + \varphi_{H12} + \varphi_{H23} + \varphi_{H34} \\ \varphi_{45} &= \varphi - \varphi_4 \\ s_{45} &= 0 \end{aligned} \right\} \quad (3.22)$$

### 3.3.3 Grundkreiswinkel und Grundkreisradius

Der Arcus-Tangens der Koordinaten des Punktes  $A_0$  entspricht dem Winkel  $\alpha$  (Abb. 17).

$$\alpha = \arctan \left( \frac{y_{A0}}{x_{A0}} \right) \quad (3.23)$$

Der Grundwinkel  $\psi_G$  ist die Differenz aus dem Winkel  $\psi_0$  und dem Winkel  $\alpha$

$$\psi_G = |\psi_0| - |\alpha| \quad (3.24)$$

Zum Grundkreisradius gelangt man über den *Kosinussatz*:

$$r_G = \sqrt{l_1^2 + l_3^2 - 2l_1l_3 \cos \psi_G} \quad (3.25)$$

### 3.3.4 Übertragungswinkel

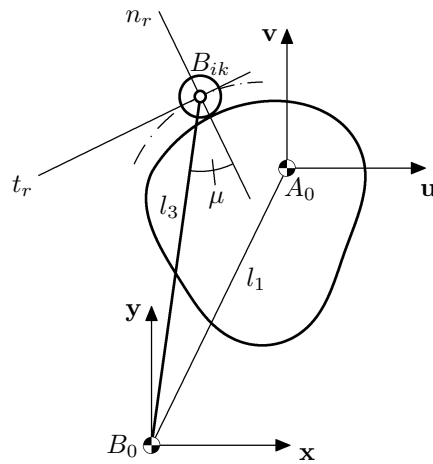


Abbildung 19: Übertragungswinkel  $\mu$  am Kurvengetriebe

Die Koordinaten des Punktes  $B_{ik}$  bezogen auf das (u,v)-Koordinatensystem in komplexer Schreibweise lauten:

$$B_{ik} = u + iv \quad (3.26)$$

Durch Ableiten von Gl. (3.26) erhält man die Geschwindigkeit des Punktes  $B_{ik}$  in Richtung der Tangente  $t_r$ .

$$B_{ik}' = u' + iv' \quad (3.27)$$

Der Geschwindigkeitsanteil in Richtung der Normalen  $n_r$  ist  $i \cdot B_{ik}'$ .

$$i \cdot B_{ik}' = i \underbrace{(u' + iv')}_{B_{ik}'} = iu' - v' \quad (3.28)$$

Dieser Vektor und der Vektor des Schwinghebels  $l_1$

$$l_1 = u + iv \quad (3.29)$$

schließen den Übertragungswinkel  $\mu$  ein. Er läßt sich über die Definitionsgleichung des skalaren Produktes berechnen.

$$\begin{aligned} \cos \mu &= \cos(l_1, iB_{ik}') \\ &= \frac{l_1 \cdot iB_{ik}'}{|l_1| \cdot |iB_{ik}'|} \quad / \arccos \\ \mu &= \arccos \frac{l_1 \cdot iB_{ik}'}{|l_1| \cdot |iB_{ik}'|} \end{aligned} \quad (3.30)$$

Für das Skalarprodukt<sup>12</sup>  $l_1 \cdot iB_{ik}'$  erhält man:

$$\begin{aligned} l_1 \cdot iB_{ik}' &= \frac{1}{2} \left( l_1 \cdot iB_{ik}' + \overline{l_1 \cdot iB_{ik}'} \right) \\ &= \frac{1}{2} \left[ \underbrace{(u + iv)}_{l_1} \underbrace{(-iu' - v')}_{\overline{iB_{ik}'}} + \underbrace{(u - iv)}_{\overline{l_1}} \underbrace{(iu' - v')}_{B_{ik}} \right] \\ &= \frac{1}{2} \left( -iuu' - uv' + vu' - ivv' + iuu' - uv' + vu' + ivv' \right) \\ &= \frac{1}{2} \left( -2uv' + 2vu' \right) \\ &= -uv' + vu' \end{aligned} \quad (3.31)$$

Die Beträge  $|l_1|$  und  $|iB_{ik}'|$  lauten:

$$|l_1| = \sqrt{u^2 + v^2} \quad (3.32)$$

<sup>12</sup>Die Rechenregel  $A \cdot B = \frac{1}{2} (A\overline{B} + \overline{A}B)$  liefert eine reelle Zahl.

$$|iB_{ik}'| = \sqrt{u'^2 + v'^2} \quad (3.33)$$

Einsetzen von Gl. (3.31) bis Gl. (3.33) in Gl. (3.30) ergibt:

$$\mu = \arccos \frac{-uv' + vv'}{\sqrt{u^2 + v^2} \sqrt{u'^2 + v'^2}} \quad (3.34)$$

### 3.3.5 Krümmungsradius

Der Krümmungsradius  $r_K$  dient als Kriterium zur Einschätzung der Ausführbarkeit und Benutzbarkeit eines Kurvengetriebes. Es gilt:

$$r_{Kik} = \frac{\sqrt{(x_{Bik}'^2 + y_{Bik}'^2)^3}}{x_{Bik}' y_{Bik}'' - x_{Bik}'' y_{Bik}'} \quad (3.35)$$

Um ausführbar und benutzbar zu sein, sollte das Getriebe folgende Bedingung erfüllen:

$$r_{K\min} \geq \frac{r_R}{0.7} \quad (3.36)$$

Ist der Krümmungsradius  $r_K$  gleich dem Rollenradius  $r_R$ , entsteht an der Kurvenkontur eine Spitze (Abb. 20). Wenn der Krümmungsradius  $r_K$  kleiner als der Rollenradius  $r_R$  ist, entsteht Hubverlust infolge von Unterschnitt.

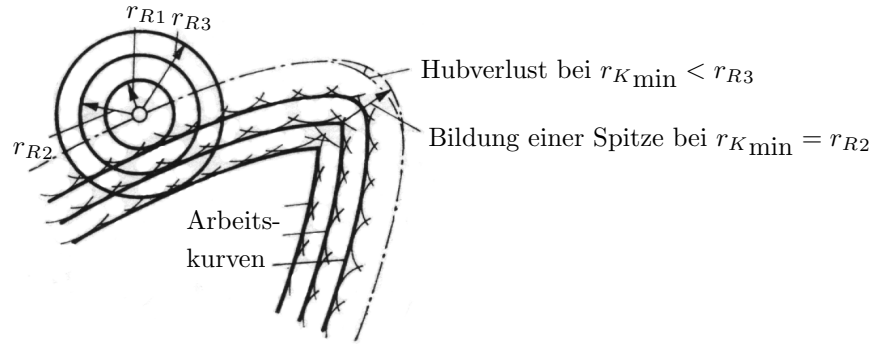


Abbildung 20: Spitzenbildung und Unterschnitt bei zu kleinem Krümmungsradius  $r_K$  [9]

## 4 Zusammenfassung

## Anhang



## A Quellcode

```
1  #include "include.h"
2  //----- class opticurv -----
3  class opticurv : public QWidget    //friend of TabDialog
4  {
5      Q_OBJECT    // notwendig, da MalFenster Elementfunktionen enthaelt
6
7      public:
8
9      double dphi_R1, dphi_An, dphi_R2, dphi_Ab, dphi_R3, dl, ds_H,
10         dalpha_sk, dl_4, dl_3, dx_H, dy_H, dr_R, dx_A0, dy_A0,
11         dbeta, dn_1, dn_2, dphi_R1fr, dphi_R2fr, dergebnis,
12         dr_G, dphi, ds, dpsi, dpsi_rad, dpsi_strich_Grad,
13         dpsi_zweistrich_Grad, dx_Bik, dy_Bik, dx_i_versetzt,
14         dy_i_versetzt, dx_B_versetzt, dy_B_versetzt, dx_a_versetzt,
15         dy_a_versetzt, dr_Bik, dmue, dsmue, dr_K, dphi_1, dphi_2,
16         dphi_3, dphi_4, dphi_5, dalpha, dgamma, ds_0sk, dl_1,
17         de_sk, dpsi_sk0, dpsi_0, dpsi_G, dri_schieber, dhub_gleich,
18         dquadrant, dpsi_positiv, ds_0_positiv;
19     long double phi_R1, phi_An, phi_R2, phi_Ab, phi_R3, l, s_H,
20         alpha_sk, l_4, l_3, x_H, y_H, r_R, x_A0, y_A0,
21         beta, n_1, n_2, phi_R1fr, phi_R2fr, ergebnis,
22         r_G, phi, alpha, gamma, s_0sk, e_sk, s_1, s_2, s_3,
23         s_4, s_5, A, B, C, D, E, F, G, H, I, J, K, L, M, N,
24         O, P, Q, R, S, T, psi_sk, psi_sk0, psi, psi_rad, A_0,
25         C_0, l_1, phi_1, phi_2, phi_3, phi_4, phi_5, s, r_Bik,
26         mue, psi_0, s_1_strich, s_2_strich, smue, r_Bik_strich,
27         s_3_strich, s_4_strich, s_5_strich, psi_strich,
28         s_1_zweistrich, s_2_zweistrich, s_3_zweistrich,
29         s_4_zweistrich, s_5_zweistrich, Zaehler_psi_zweistrich,
30         Nenner_psi_zweistrich, psi_zweistrich, psi_strich_Grad,
31         psi_zweistrich_Grad, psi_G, alphan, x_Bik, y_Bik,
32         sinpsibeta, cospsibeta, x_Bik_strich, y_Bik_strich,
33         Betrag_B_strich, x_i, y_i, x_a, y_a, x_i_versetzt,
34         y_i_versetzt, x_B_versetzt, y_B_versetzt, x_a_versetzt,
35         y_a_versetzt, zaehler_mue, nenner_mue, x_Bik_zweistrich,
36         y_Bik_zweistrich, wurzelausdruck, potenz_wurzelausdruck,
37         r_K, zaehler_r_K, nenner_r_K;
38     double *z_phi;
39     long double *z_s, *z_psi, *z_psi_rad, *z_psi_strich_Grad,
40         *z_psi_zweistrich_Grad, *z_x_Bik, *z_y_Bik, *z_x_i_versetzt,
41         *z_y_i_versetzt, *z_x_B_versetzt, *z_y_B_versetzt, *z_x_a_versetzt,
42         *z_y_a_versetzt;
```

```

43
44     char alpha_richtung, s_richtung, Ba_Qudrant, positiv, positiv2;
45     QString filename, filename_old, filename_asc, filename_asc2,
46         text;
47     QButtonGroup *buttonbox1, *buttonbox2, *buttonbox3, *buttonbox4,
48         *buttonbox5;
49     QRadioButton *ja1, *nein1, *ja2, *nein2, *ja3, *nein3, *ja4, *nein4,
50         *ja5, *nein5;
51     QLabel *abfrage1, *abfrage2, *abfrage3, *abfrage4, *abfrage5;
52
53     opticurv()
54     {
55         dateimenu = new QPopupMenu; //... erzeugt ein Datei-Menue
56         dateimenu->insertItem( "&Neu", this, SLOT( NeuSlot() ) );
57         dateimenu->insertItem( "Ö&ffnen", this, SLOT( OeffnenSlot() ) );
58         dateimenu->insertItem( "S&chlieSSen", qApp, SLOT( quit() ) );
59
60         hilfemenu = new QPopupMenu; //... erzeugt ein Hilfe-Menue
61         hilfemenu->insertItem( "Ü&ber opticurv", this, SLOT( AboutSlot() ) );
62
63         menubalken = new QMenuBar( this ); //... erzeugt einen Menue-Balken
64         menubalken->insertItem( "&Datei", dateimenu );
65         menubalken->insertSeparator();
66         menubalken->insertItem( "&Hilfe", hilfemenu );
67
68         anzeige = new QTextView( this );
69         anzeige->setGeometry( 0, 20, 400, 380 );
70     }
71
72     ~opticurv() { } // noch kein Code fuer den Destruktor
73
74     signals:
75
76     public slots:
77
78     protected:
79     QTextView* anzeige;
80
81     private slots:
82     void AboutSlot()
83     {
84         QMessageBox::information( this, "opticurv", "Programm zur Auslegung"
85                                     " von\nKurvenkoppelgetrieben");
86     }

```

```

87
88 void NeuSlot()
89 {
90     dphi_R1 = dphi_An = dphi_R2 = dphi_Ab = dphi_R3 = 1;
91     dl = ds_H = dalpha_sk = dl_4 = dl_3 = 1;
92     dx_H = dy_H = dr_R = dx_A0 = dy_A0 = 1;
93     dbeta = dr_G = dn_1 = 1;
94
95     void setFilter(const QString& ocv);
96
97     filename = QFileDialog::getSaveFileName( QString::null, "*.ocv", this );
98     if( !filename.isEmpty() )
99     {
100         filename_old = filename;
101         filename = filename + ".ocv"; // Endung .ocv an filename anhängen
102         filename_asc = filename_old + "1.asc"; // Endung .asc an filename_asc
103
104         filename_asc2 = filename_old + "2.asc";
105
106         QFile ofl(filename);
107         ofl.open(IO_WriteOnly | IO_Truncate); // Datei für die Ausgabe öffnen
108         ofl.close();
109         QFile ofl_asc(filename_asc);
110         ofl_asc.open(IO_WriteOnly | IO_Truncate);
111         ofl_asc.close();
112         QFile ofl_asc2(filename_asc2);
113         ofl_asc2.open(IO_WriteOnly | IO_Truncate);
114         ofl_asc2.close();
115
116         tabdialog = new QTabDialog(0,0,true);
117         tabdialog->setGeometry( 120, 120, 280, 330 );
118         tabdialog->setCaption( "Eingabeparameter" );
119         setupTab1();
120         setupTab2();
121         setupTab3();
122         setupTab4();
123         setupTab5();
124         setupTab6();
125         setupTab7();
126         setup_ergAnz();
127         tabdialog->show();
128     }
129 }
130

```

```

131 void OeffnenSlot()
132 {
133     void setFilter(const QString& ocv);
134     filename = QFileDialog::getOpenFileName( QString::null, "*.ocv", this );
135     if( !filename.isEmpty() )
136     {
137         double rdaten[24];
138         QFile f(filename);
139         f.open(IO_ReadOnly);
140         f.readBlock((char*)rdaten,sizeof(rdaten));
141         f.close();
142         dphi_R1=rdaten[0];
143         dphi_An=rdaten[1];
144         dphi_R2=rdaten[2];
145         dphi_Ab=rdaten[3];
146         dphi_R3=rdaten[4];
147         dl=rdaten[5];
148         ds_H=rdaten[6];
149         dalpha_sk=rdaten[7];
150         dl_4=rdaten[8];
151         dl_3=rdaten[9];
152         dx_H=rdaten[10];
153         dy_H=rdaten[11];
154         dr_R=rdaten[12];
155         dx_A0=rdaten[13];
156         dy_A0=rdaten[14];
157         dbeta=rdaten[15];
158         dr_G=rdaten[16];
159         dn_1=rdaten[17];
160         dri_schieber=rdaten[18];
161         dhub_gleich=rdaten[19];
162         dquadrant=rdaten[20];
163         dpsi_positiv=rdaten[21];
164         ds_0_positiv=rdaten[22];
165
166         filename_old = filename;
167         filename_old.remove( filename_old.length()-4,4 );
168         filename_asc = filename_old + "1.asc"; // Endung .asc an
169                                                  // filename_asc anhängen
170         filename_asc2 = filename_old + "2.asc";
171
172         QFile ofle(filename);
173         ofle.open(IO_WriteOnly | IO_Truncate); // Datei für die Ausgabe öffnen
174         ofle.close();

```

```

175     QFile ofle_asc(filename_asc);
176     ofle_asc.open(IO_WriteOnly | IO_Truncate);
177     ofle_asc.close();
178     QFile ofle_asc2(filename_asc2);
179     ofle_asc2.open(IO_WriteOnly | IO_Truncate);
180     ofle_asc2.close();
181
182     tabdialog = new QTabDialog(0,0,true);
183     tabdialog->setGeometry( 120, 120, 280, 330 );
184     tabdialog->setCaption( "Eingabeparameter" );
185     setupTab1();
186     setupTab2();
187     setupTab3();
188     setupTab4();
189     setupTab5();
190     setupTab6();
191     setupTab7();
192     setup_ergAnz();
193     tabdialog->show();
194 }
195 }
196
197 void speichere()
198 {
199     sichereAbfrage();
200     QFile f(filename);
201     f.open(IO_WriteOnly);
202     double wdaten[24] = {dphi_R1,dphi_An,dphi_R2,dphi_Ab,dphi_R3,d1,ds_H,
203                          dalpha_sk,d1_4,d1_3,dx_H,dy_H,dr_R,dx_A0,dy_A0,
204                          dbeta,dr_G,dn_1,dri_schieber,dhub_gleich,
205                          dquadrant,dpsi_positiv,ds_0_positiv};
206     f.writeBlock((char*)wdaten,sizeof(wdaten));
207     f.close();
208     calculate();
209 }
210
211 void neu_phi_R1( const QString& neuWert )
212 {
213     double phiR1 = neuWert.toDouble(&ok);
214     if(ok == true)
215         dphi_R1 = phiR1;
216     berechne(phi_R1Eingabe);
217 }
218

```

```

219 void neu_phi_An( const QString& neuWert )
220 {
221     double phiAn = neuWert.toDouble(&ok);
222     if(ok == true)
223         dphi_An = phiAn;
224     berechne(phi_AnEingabe);
225 }
226
227 void neu_phi_R2( const QString& neuWert )
228 {
229     double phiR2 = neuWert.toDouble(&ok);
230     if(ok == true)
231         dphi_R2 = phiR2;
232     berechne(phi_R2Eingabe);
233 }
234
235 void neu_phi_Ab( const QString& neuWert )
236 {
237     double phiAb = neuWert.toDouble(&ok);
238     if(ok == true)
239         dphi_Ab = phiAb;
240     berechne(phi_AbEingabe);
241 }
242
243 void neu_phi_R3( const QString& neuWert )
244 {
245     double phiR3 = neuWert.toDouble(&ok);
246     if(ok == true)
247         dphi_R3 = phiR3;
248     berechne(phi_R3Eingabe);
249 }
250
251 void neu_l( const QString& neuWert )
252 {
253     double l1 = neuWert.toDouble(&ok);
254     if(ok == true)
255         dl = l1;
256     berechne(lEingabe);
257 }
258
259 void neu_s_H( const QString& neuWert )
260 {
261     double sH = neuWert.toDouble(&ok);
262     if(ok == true)

```

```

263         ds_H = sH;
264         berechne(s_HEingabe);
265     }
266
267     void neu_alpha_sk( const QString& neuWert )
268     {
269         double alphask = neuWert.toDouble(&ok);
270         if(ok == true)
271             dalpha_sk = alphask;
272         berechne(alpha_skEingabe);
273     }
274
275     void neu_l_4( const QString& neuWert )
276     {
277         double l4 = neuWert.toDouble(&ok);
278         if(ok == true)
279             dl_4 = l4;
280         berechne(l_4Eingabe);
281     }
282
283     void neu_l_3( const QString& neuWert )
284     {
285         double l3 = neuWert.toDouble(&ok);
286         if(ok == true)
287             dl_3 = l3;
288         berechne(l_3Eingabe);
289     }
290     void neu_x_H( const QString& neuWert )
291     {
292         double xH = neuWert.toDouble(&ok);
293         if(ok == true)
294             dx_H = xH;
295         berechne(x_HEingabe);
296     }
297
298     void neu_y_H( const QString& neuWert )
299     {
300         double yH = neuWert.toDouble(&ok);
301         if(ok == true)
302             dy_H = yH;
303         berechne(y_HEingabe);
304     }
305
306     void neu_r_R( const QString& neuWert )

```

```

307     {
308         double rR = neuWert.toDouble(&ok);
309         if(ok == true)
310             dr_R = rR;
311         berechne(r_REingabe);
312     }
313
314     void neu_x_A0( const QString& neuWert )
315     {
316         double xA0 = neuWert.toDouble(&ok);
317         if(ok == true)
318             dx_A0 = xA0;
319         berechne(x_A0Eingabe);
320     }
321
322     void neu_y_A0( const QString& neuWert )
323     {
324         double yA0 = neuWert.toDouble(&ok);
325         if(ok == true)
326             dy_A0 = yA0;
327         berechne(y_A0Eingabe);
328     }
329
330     void neu_beta( const QString& neuWert )
331     {
332         double bta = neuWert.toDouble(&ok);
333         if(ok == true)
334             dbeta = bta;
335         berechne(betaEingabe);
336     }
337
338     void neu_r_G( const QString& neuWert )
339     {
340         double rG = neuWert.toDouble(&ok);
341         if(ok == true)
342             dr_G = rG;
343         berechne(r_GEingabe);
344     }
345
346     void neu_n_1( const QString& neuWert )
347     {
348         double n1 = neuWert.toDouble(&ok);
349         if(ok == true)
350             dn_1 = n1;

```



```

351     berechne(n_1Eingabe);
352 }
353
354 private:
355
356 QMenuBar*    menubalken;
357 QPopupMenu*  dateimenu;
358 QPopupMenu*  hilfemenu;
359 QTabDialog*  tabdialog;
360 QScrollView* scrollview;
361 QCheckBox*   _checkbox;
362 QLineEdit*  *phi_R1Eingabe, *phi_AnEingabe, *phi_R2Eingabe,
363             *phi_AbEingabe, *phi_R3Eingabe, *lEingabe, *s_HEingabe,
364             *alpha_skEingabe, *l_4Eingabe, *l_3Eingabe, *x_HEingabe,
365             *y_HEingabe, *r_REingabe, *x_A0Eingabe, *y_A0Eingabe,
366             *x_BaEingabe, *y_BaEingabe,
367             *betaEingabe, *r_GEingabe,
368             *n_1Eingabe;
369 QLabel*     ergebnisAnzeige;
370 bool ok;
371
372 void berechne( QLineEdit *eingabeWidget )
373 {
374     if(ok == true)
375     {
376         QColor farbe( 0, 255, 255);
377         ergebnis = 1;
378         ergebnisAnzeige->setNum(dergebnis);
379         eingabeWidget->setPalette( QPalette( farbe, farbe ));
380         ergebnisAnzeige->setPalette( QPalette( farbe, farbe ));
381     }
382     else
383     {
384         eingabeWidget->setPalette( QPalette( Qt::red, Qt::red ));
385         ergebnisAnzeige->setPalette( QPalette( Qt::red, Qt::red ));
386         ergebnisAnzeige->setText( "undefiniert" );
387     }
388     repaint();
389 }
390
391 void setupTab1();
392 void setupTab2();
393 void setupTab3();
394 void setupTab4();

```

```

395     void setupTab5();
396     void setupTab6();
397     void setupTab7();
398     void sichereAbfrage();
399     void setup_ergAnz();
400     void calculate();
401 };
402 void opticurv::setupTab1()
403 {
404     QWidget* firstpage = new QWidget( this );
405     firstpage->setGeometry(10,10,260,310);
406     tabdialog->addTab( firstpage, "1.-5.");
407     QColor farbe( 0, 255, 255 );
408     ergebnis = 1;
409
410     phi_R1Eingabe = new QLineEdit( firstpage );
411     phi_R1Eingabe->setGeometry( 140,40,100, 30);
412     QString phi_R1ausgabe=phi_R1Eingabe->text();
413     phi_R1ausgabe.setNum( dphi_R1 );
414     phi_R1Eingabe->setText( phi_R1ausgabe );
415     phi_R1Eingabe->setPalette( QPalette( farbe, farbe ) );
416     QLabel* labelR1 = new QLabel( "1.) phi_H01 [Grad]",firstpage );
417     labelR1->setGeometry( 20, 40, 120, 30);
418
419     phi_AnEingabe = new QLineEdit( firstpage );
420     phi_AnEingabe->setGeometry( 140, 70, 100,30);
421     QString phi_Anausgabe=phi_AnEingabe->text();
422     phi_Anausgabe.setNum( dphi_An );
423     phi_AnEingabe->setText( phi_Anausgabe );
424     phi_AnEingabe->setPalette( QPalette( farbe, farbe ) );
425     QLabel* labelAn = new QLabel( "2.) phi_H12 [Grad]",firstpage );
426     labelAn->setGeometry( 20, 70, 120, 30);
427
428     phi_R2Eingabe = new QLineEdit( firstpage );
429     phi_R2Eingabe->setGeometry( 140,100,100, 30);
430     QString phi_R2ausgabe=phi_R2Eingabe->text();
431     phi_R2ausgabe.setNum( dphi_R2 );
432     phi_R2Eingabe->setText( phi_R2ausgabe );
433     phi_R2Eingabe->setPalette( QPalette( farbe, farbe ) );
434     QLabel* labelR2 = new QLabel( "3.) phi_H23 [Grad]",firstpage );
435     labelR2->setGeometry( 20, 100, 120, 30);
436
437     phi_AbEingabe = new QLineEdit( firstpage );
438     phi_AbEingabe->setGeometry( 140, 130, 100,30);

```

```

439     QString phi_Abausgabe=phi_AbEingabe->text();
440     phi_Abausgabe.setNum( dphi_Ab );
441     phi_AbEingabe->setText( phi_Abausgabe );
442     phi_AbEingabe->setPalette( QPalette( farbe, farbe ) );
443     QLabel* labelAb = new QLabel( "4.)  phi_H34      [Grad]",firstpage );
444     labelAb->setGeometry( 20, 130, 120, 30);
445
446     phi_R3Eingabe = new QLineEdit( firstpage );
447     phi_R3Eingabe->setGeometry( 140,160,100, 30);
448     QString phi_R3ausgabe=phi_R3Eingabe->text();
449     phi_R3ausgabe.setNum( dphi_R3 );
450     phi_R3Eingabe->setText( phi_R3ausgabe );
451     phi_R3Eingabe->setPalette( QPalette( farbe, farbe ) );
452     QLabel* labelR3 = new QLabel( "5.)  phi_H45      [Grad]",firstpage );
453     labelR3->setGeometry( 20, 160, 120, 30);
454
455     QObject::connect(phi_R1Eingabe,
456                     SIGNAL( textChanged( const QString & ) ),
457                     this, SLOT( neu_phi_R1( const QString & ) ) );
458
459     QObject::connect(phi_AnEingabe,
460                     SIGNAL( textChanged( const QString & ) ),
461                     this, SLOT( neu_phi_An( const QString & ) ) );
462
463     QObject::connect(phi_R2Eingabe,
464                     SIGNAL( textChanged( const QString & ) ),
465                     this, SLOT( neu_phi_R2( const QString & ) ) );
466
467     QObject::connect(phi_AbEingabe,
468                     SIGNAL( textChanged( const QString & ) ),
469                     this, SLOT( neu_phi_Ab( const QString & ) ) );
470
471     QObject::connect(phi_R3Eingabe,
472                     SIGNAL( textChanged( const QString & ) ),
473                     this, SLOT( neu_phi_R3( const QString & ) ) );
474
475 }
476
477 void opticurv::setupTab2()
478 {
479     QWidget* secondpage = new QWidget( this );
480     secondpage->setGeometry(10,10,260,310);
481     tabdialog->addTab( secondpage, "6.-10.");
482     QColor farbe( 0, 255, 255 );

```

```

483     ergebnis = 1;
484
485     lEingabe = new QLineEdit( secondpage );
486     lEingabe->setGeometry( 140,40,100, 30);
487     QString lausgabe=lEingabe->text();
488     lausgabe.setNum( dl );
489     lEingabe->setText( lausgabe );
490     lEingabe->setPalette( QPalette( farbe, farbe ) );
491     QLabel* labell = new QLabel( "6.)  l_3                [mm]",secondpage );
492     labell->setGeometry( 20, 40, 120, 30);
493
494     l_3Eingabe = new QLineEdit( secondpage );
495     l_3Eingabe->setGeometry( 140, 70, 100,30);
496     QString l_3ausgabe=l_3Eingabe->text();
497     l_3ausgabe.setNum( dl_3 );
498     l_3Eingabe->setText( l_3ausgabe );
499     l_3Eingabe->setPalette( QPalette( farbe, farbe ) );
500     QLabel* labell_3 = new QLabel( "7.)  l_4                [mm]",secondpage );
501     labell_3->setGeometry( 20, 70, 120, 30);
502
503     l_4Eingabe = new QLineEdit( secondpage );
504     l_4Eingabe->setGeometry( 140,100,100, 30);
505     QString l_4ausgabe=l_4Eingabe->text();
506     l_4ausgabe.setNum( dl_4 );
507     l_4Eingabe->setText( l_4ausgabe );
508     l_4Eingabe->setPalette( QPalette( farbe, farbe ) );
509     QLabel* labell_4 = new QLabel( "8.)  l_sk                [mm]",secondpage );
510     labell_4->setGeometry( 20, 100, 120, 30);
511
512     s_HEingabe = new QLineEdit( secondpage );
513     s_HEingabe->setGeometry( 140, 130, 100,30);
514     QString s_Hausgabe=s_HEingabe->text();
515     s_Hausgabe.setNum( ds_H );
516     s_HEingabe->setText( s_Hausgabe );
517     s_HEingabe->setPalette( QPalette( farbe, farbe ) );
518     QLabel* labels_H = new QLabel( "9.)  s_H                [mm]",secondpage );
519     labels_H->setGeometry( 20, 130, 120, 30);
520
521     alpha_skEingabe = new QLineEdit( secondpage );
522     alpha_skEingabe->setGeometry( 140,160,100, 30);
523     QString alpha_skausgabe=alpha_skEingabe->text();
524     alpha_skausgabe.setNum( dalpha_sk );
525     alpha_skEingabe->setText( alpha_skausgabe );
526     alpha_skEingabe->setPalette( QPalette( farbe, farbe ) );

```

```

527     QLabel* labelalpha_sk = new QLabel( "10.)  alpha_sk [Grad]",secondpage );
528     labelalpha_sk->setGeometry( 20, 160, 120, 30);
529
530
531     QObject::connect(lEingabe,
532                     SIGNAL( textChanged( const QString & ) ),
533                     this, SLOT( neu_l( const QString & ) ) );
534
535     QObject::connect(s_HEingabe,
536                     SIGNAL( textChanged( const QString & ) ),
537                     this, SLOT( neu_s_H( const QString & ) ) );
538
539     QObject::connect(alpha_skEingabe,
540                     SIGNAL( textChanged( const QString & ) ),
541                     this, SLOT( neu_alpha_sk( const QString & ) ) );
542
543     QObject::connect(l_4Eingabe,
544                     SIGNAL( textChanged( const QString & ) ),
545                     this, SLOT( neu_l_4( const QString & ) ) );
546
547     QObject::connect(l_3Eingabe,
548                     SIGNAL( textChanged( const QString & ) ),
549                     this, SLOT( neu_l_3( const QString & ) ) );
550 }
551
552 void opticurv::setupTab3()
553 {
554     QWidget* thirdpage = new QWidget( this );
555     thirdpage->setGeometry(10,10,260,310);
556     tabdialog->addTab( thirdpage, "11.-15.");
557     QColor farbe( 0, 255, 255 );
558     ergebnis = 1;
559
560     x_HEingabe = new QLineEdit( thirdpage );
561     x_HEingabe->setGeometry( 140,40,100, 30);
562     QString x_Hausgabe=x_HEingabe->text();
563     x_Hausgabe.setNum( dx_H );
564     x_HEingabe->setText( x_Hausgabe );
565     x_HEingabe->setPalette( QPalette( farbe, farbe ) );
566     QLabel* labelx_H = new QLabel( "11.)  x_H          [mm]",thirdpage );
567     labelx_H->setGeometry( 20, 40, 120, 30);
568
569     y_HEingabe = new QLineEdit( thirdpage );
570     y_HEingabe->setGeometry( 140, 70, 100,30);

```

```

571     QString y_Hausgabe=y_HEingabe->text();
572     y_Hausgabe.setNum( dy_H );
573     y_HEingabe->setText( y_Hausgabe );
574     y_HEingabe->setPalette( QPalette( farbe, farbe ) );
575     QLabel* labely_H = new QLabel( "12.) y_H [mm]",thirdpage );
576     labely_H->setGeometry( 20, 70, 120, 30);
577
578     r_REingabe = new QLineEdit( thirdpage );
579     r_REingabe->setGeometry( 140,100,100, 30);
580     QString r_Rausgabe=r_REingabe->text();
581     r_Rausgabe.setNum( dr_R );
582     r_REingabe->setText( r_Rausgabe );
583     r_REingabe->setPalette( QPalette( farbe, farbe ) );
584     QLabel* labelr_R = new QLabel( "13.) r_R [mm]",thirdpage );
585     labelr_R->setGeometry( 20, 100, 120, 30);
586
587     x_A0Eingabe = new QLineEdit( thirdpage );
588     x_A0Eingabe->setGeometry( 140, 130, 100,30);
589     QString x_A0ausgabe=x_A0Eingabe->text();
590     x_A0ausgabe.setNum( dx_A0 );
591     x_A0Eingabe->setText( x_A0ausgabe );
592     x_A0Eingabe->setPalette( QPalette( farbe, farbe ) );
593     QLabel* labelx_A0 = new QLabel( "14.) x_A0 [mm]",thirdpage );
594     labelx_A0->setGeometry( 20, 130, 120, 30);
595
596     y_A0Eingabe = new QLineEdit( thirdpage );
597     y_A0Eingabe->setGeometry( 140,160,100, 30);
598     QString y_A0ausgabe=y_A0Eingabe->text();
599     y_A0ausgabe.setNum( dy_A0 );
600     y_A0Eingabe->setText( y_A0ausgabe );
601     y_A0Eingabe->setPalette( QPalette( farbe, farbe ) );
602     QLabel* labely_A0 =new QLabel( "15.) y_A0 [mm]",thirdpage );
603     labely_A0->setGeometry( 20, 160, 120, 30);
604
605     QObject::connect(x_HEingabe,
606                     SIGNAL( textChanged( const QString & ) ),
607                     this, SLOT( neu_x_H( const QString & ) ) );
608
609     QObject::connect(y_HEingabe,
610                     SIGNAL( textChanged( const QString & ) ),
611                     this, SLOT( neu_y_H( const QString & ) ) );
612
613     QObject::connect(r_REingabe,
614                     SIGNAL( textChanged( const QString & ) ),

```

```

615         this, SLOT( neu_r_R( const QString & ) ) );
616
617     QObject::connect(x_A0Eingabe,
618         SIGNAL( textChanged( const QString & ) ),
619         this, SLOT( neu_x_A0( const QString & ) ) );
620
621     QObject::connect(y_A0Eingabe,
622         SIGNAL( textChanged( const QString & ) ),
623         this, SLOT( neu_y_A0( const QString & ) ) );
624 }
625
626 void opticurv::setupTab4()
627 {
628     QWidget* fourthpage = new QWidget( this );
629     fourthpage->setGeometry(10,10,260,310);
630     tabdialog->addTab( fourthpage, "16.-20.");
631     QColor farbe( 0, 255, 255 );
632     ergebnis = 1;
633
634
635     betaEingabe = new QLineEdit( fourthpage );
636     betaEingabe->setGeometry( 140,40,100, 30);
637     QString betaausgabe=betaEingabe->text();
638     betaausgabe.setNum( dbeta );
639     betaEingabe->setText( betaausgabe );
640     betaEingabe->setPalette( QPalette( farbe, farbe ) );
641     QLabel* labelbeta = new QLabel( "16.)  beta          [Grad]",fourthpage );
642     labelbeta->setGeometry( 20, 40, 120, 30);
643
644     r_GEingabe = new QLineEdit( fourthpage );
645     r_GEingabe->setGeometry( 140, 70, 100,30);
646     QString r_Gausgabe=r_GEingabe->text();
647     r_Gausgabe.setNum( dr_G );
648     r_GEingabe->setText( r_Gausgabe );
649     r_GEingabe->setPalette( QPalette( farbe, farbe ) );
650     QLabel* labelr_G = new QLabel( "17.)  r_G          [mm]",fourthpage );
651     labelr_G->setGeometry( 20, 70, 120, 30);
652
653     n_1Eingabe = new QLineEdit( fourthpage );
654     n_1Eingabe->setGeometry( 140,100,100, 30);
655     QString n_1ausgabe=n_1Eingabe->text();
656     n_1ausgabe.setNum( dn_1 );
657     n_1Eingabe->setText( n_1ausgabe );
658     n_1Eingabe->setPalette( QPalette( farbe, farbe ) );

```

```

659     QLabel* labeln_1 = new QLabel( "18.) n                                [-]",fourthpage );
660     labeln_1->setGeometry( 20, 100, 120, 30);
661
662     x_BaEingabe = new QLineEdit( fourthpage );
663     x_BaEingabe->setGeometry( 140, 130, 100,30);
664     QString x_Baausgabe=x_BaEingabe->text();
665     x_BaEingabe->setPalette( QPalette( farbe, farbe ) );
666     QLabel* labelx_Ba = new QLabel( "19.)",fourthpage );
667     labelx_Ba->setGeometry( 20, 130, 120, 30);
668
669     y_BaEingabe = new QLineEdit( fourthpage );
670     y_BaEingabe->setGeometry( 140,160,100, 30);
671     QString y_Baausgabe=y_BaEingabe->text();
672     y_BaEingabe->setPalette( QPalette( farbe, farbe ) );
673     QLabel* labely_Ba = new QLabel( "20.)",fourthpage );
674     labely_Ba->setGeometry( 20, 160, 120, 30);
675
676     QObject::connect(betaEingabe,
677                     SIGNAL( textChanged( const QString & ) ),
678                     this, SLOT( neu_beta( const QString & ) ) );
679
680     QObject::connect(r_GEingabe,
681                     SIGNAL( textChanged( const QString & ) ),
682                     this, SLOT( neu_r_G( const QString & ) ) );
683
684     QObject::connect(n_1Eingabe,
685                     SIGNAL( textChanged( const QString & ) ),
686                     this, SLOT( neu_n_1( const QString & ) ) );
687 }
688 void opticurv::setupTab5()
689 {
690     QWidget* fifthpage = new QWidget( this );
691     fifthpage->setGeometry(10,10,260,310);
692     tabdialog->addTab( fifthpage, "21.-22.");
693     QColor farbe( 0, 255, 255 );
694     ergebnis = 1;
695
696     buttonbox1 = new QButtonGroup( 1, Horizontal, "21.) Anfangsauslenkung der",
697                                   fifthpage);
698     buttonbox1->setGeometry( 20, 10, 220, 100 );
699     buttonbox1->setFrameStyle( QFrame::Panel | QFrame::Sunken );
700     abfrage1 = new QLabel( buttonbox1 );
701     abfrage1->setText( "Schubkurbel zum Schieber hin\ngerichtet?" );
702     abfrage1->setGeometry( 10, 120, 200, 100 );

```



```

703     ja1 = new QRadioButton( "Ja", buttonbox1 );
704     nein1 = new QRadioButton( "Nein", buttonbox1 );
705
706     if(dri_schieber == 1)
707     {
708         ja1->setChecked( true );
709     }
710     else if(dri_schieber == 2)
711     {
712         nein1->setChecked( true );
713     }
714     else
715     {
716         ja1->setChecked( true );
717     }
718
719     buttonbox2 = new QButtonGroup( 1, Horizontal, "22.) Hubrichtungen von",
720                                     fifthpage);
721     buttonbox2->setGeometry( 20, 125, 220, 100 );
722     buttonbox2->setFrameStyle( QFrame::Panel | QFrame::Sunken );
723     abfrage2 = new QLabel( buttonbox2 );
724     abfrage2->setText( "Hub s_0 und Hub s(phi) gleich\ngerichtet?" );
725     abfrage2->setGeometry( 10, 120, 200, 100 );
726     ja2 = new QRadioButton( "Ja", buttonbox2 );
727     nein2 = new QRadioButton( "Nein", buttonbox2 );
728     if(dhub_gleich == 1)
729     {
730         ja2->setChecked( true );
731     }
732     else if(dhub_gleich == 2)
733     {
734         nein2->setChecked( true );
735     }
736     else
737     {
738         ja2->setChecked( true );
739     }
740
741 }
742 void opticurv::setupTab6()
743 {
744     QWidget* sixthpage = new QWidget( this );
745     sixthpage->setGeometry(10,10,260,310);
746     tabdialog->addTab( sixthpage, "23.-24.");

```

```

747     QColor farbe( 0, 255, 255 );
748     ergebnis = 1;
749
750     buttonbox3 = new QButtonGroup( 1, Horizontal, "23.) In welchem Quadrant",
751                                     sixthpage);
752     buttonbox3->setGeometry( 20, 10, 220, 100 );
753     buttonbox3->setFrameStyle( QFrame::Panel | QFrame::Sunken );
754     abfrage3 = new QLabel( buttonbox3 );
755     abfrage3->setText( "befindet sich der Punkt B_a?" );
756     abfrage3->setGeometry( 10, 120, 200, 100 );
757     ja3 = new QRadioButton( "1. bzw. 4. Quadrant", buttonbox3 );
758     nein3 = new QRadioButton( "2. bzw. 3. Quadrant", buttonbox3 );
759     if(dquadrant == 1)
760     {
761         ja3->setChecked( true );
762     }
763     else if(dquadrant == 2)
764     {
765         nein3->setChecked( true );
766     }
767     else
768     {
769         ja3->setChecked( true );
770     }
771
772     buttonbox4 = new QButtonGroup( 1, Horizontal, "24.) Ist die Bewegung des",
773                                     sixthpage);
774     buttonbox4->setGeometry( 20, 125, 220, 100 );
775     buttonbox4->setFrameStyle( QFrame::Panel | QFrame::Sunken );
776     abfrage4 = new QLabel( buttonbox4 );
777     abfrage4->setText( "Schwinghebels positiv oder negativ?" );
778     abfrage4->setGeometry( 10, 120, 200, 100 );
779     ja4 = new QRadioButton( "positiv", buttonbox4 );
780     nein4 = new QRadioButton( "negativ", buttonbox4 );
781     if(dpsi_positiv == 1)
782     {
783         ja4->setChecked( true );
784     }
785     else if(dpsi_positiv == 2)
786     {
787         nein4->setChecked( true );
788     }
789     else
790     {

```

```

791     ja4->setChecked( true );
792 }
793
794 }
795
796 void opticurv::setupTab7()
797 {
798     QWidget* seventhpage = new QWidget( this );
799     seventhpage->setGeometry(10,10,260,310);
800     tabdialog->addTab( seventhpage, "25.");
801     QColor farbe( 0, 255, 255 );
802     ergebnis = 1;
803
804     buttonbox5 = new QButtonGroup( 1, Horizontal, "25.) Ist der Hub s_0",
805                                     seventhpage);
806     buttonbox5->setGeometry( 20, 10, 220, 100 );
807     buttonbox5->setFrameStyle( QFrame::Panel | QFrame::Sunken );
808     abfrage5 = new QLabel( buttonbox5 );
809     abfrage5->setText( "positiv oder negativ?" );
810     abfrage5->setGeometry( 10, 120, 200, 100 );
811     ja5 = new QRadioButton( "positiv", buttonbox5 );
812     nein5 = new QRadioButton( "negativ", buttonbox5 );
813     if(ds_0_positiv == 1)
814     {
815         ja5->setChecked( true );
816     }
817     else if(ds_0_positiv == 2)
818     {
819         nein5->setChecked( true );
820     }
821     else
822     {
823         ja5->setChecked( true );
824     }
825
826 }
827
828
829 void opticurv::sichereAbfrage()
830 {
831     if ( ja1->isChecked() )alpha_richtung = 'y';
832     if ( nein1->isChecked() )alpha_richtung = 'n';
833     if ( ja2->isChecked() )s_richtung = 'y';
834     if ( nein2->isChecked() )s_richtung = 'n';

```

```

835     if ( ja3->isChecked() )Ba_Qudrant = 'y';
836     if ( nein3->isChecked() )Ba_Qudrant = 'n';
837     if ( ja4->isChecked() )positiv = 'y';
838     if ( nein4->isChecked() )positiv = 'n';
839     if ( ja5->isChecked() )positiv2 = 'y';
840     if ( nein5->isChecked() )positiv2 = 'n';
841
842     if(alpha_richtung == 'y')
843     {
844         dri_schieber = 1;
845     }
846     else if(alpha_richtung == 'n')
847     {
848         dri_schieber = 2;
849     }
850     if(s_richtung == 'y')
851     {
852         dhub_gleich = 1;
853     }
854     else if(s_richtung == 'n')
855     {
856         dhub_gleich = 2;
857     }
858     if(Ba_Qudrant == 'y')
859     {
860         dquadrant = 1;
861     }
862     else if(Ba_Qudrant == 'n')
863     {
864         dquadrant = 2;
865     }
866     if(positiv == 'y')
867     {
868         dpsi_positiv = 1;
869     }
870     else if(positiv == 'n')
871     {
872         dpsi_positiv = 2;
873     }
874     if(positiv2 == 'y')
875     {
876         ds_0_positiv = 1;
877     }
878     else if(positiv2 == 'n')

```

```

879     {
880         ds_0_positiv = 2;
881     }
882 }
883
884 void opticurv::setup_ergAnz()
885 {
886     QColor farbe( 0, 255, 255 );
887     ergebnisAnzeige = new QLabel( tabdialog );
888     ergebnisAnzeige->setGeometry( 20, 295, 70, 30);
889     ergebnisAnzeige->setNum( 1 );
890     ergebnisAnzeige->setAlignment( QLabel::AlignCenter );
891     ergebnisAnzeige->setPalette( QPalette( farbe, farbe ) );
892     ergebnisAnzeige->show();
893     tabdialog->setOkButton( "&Ausfuehren" );
894     tabdialog->setCancelButton( "A&bbrechen" );
895
896     connect( tabdialog, SIGNAL( applyButtonPressed() ), this,
897             SLOT( speichere() ) );
898 }
899 void opticurv::calculate()
900 {
901
902     long double rad = (pi/180);
903
904     /* implizite Typkonvertierungen */
905
906     phi_R1 = dphi_R1;
907     phi_An = dphi_An;
908     phi_R2 = dphi_R2;
909     phi_Ab = dphi_Ab;
910     phi_R3 = dphi_R3;
911     l = dl;
912     s_H = ds_H;
913     alpha_sk = dalpha_sk;
914     l_4 = dl_4;
915     l_3 = dl_3;
916     x_H = dx_H;
917     y_H = dy_H;
918     r_R = dr_R;
919     x_A0 = dx_A0;
920     y_A0 = dy_A0;
921     beta = dbeta;
922     n_1 = dn_1;

```

```

923     n_2 = dn_2;
924     phi_R1fr = dphi_R1fr;
925     phi_R2fr = dphi_R2fr;
926     ergebnis = dergebnis;
927     r_G = dr_G;
928     phi = dphi;
929
930     alpha = ((atan(y_A0/x_A0))/rad);
931
932     static const char* text_Grad[] = {" Grad",0};
933     static const char* text_mm[] = {" mm",0};
934
935     QString text;
936
937     gamma = ((asin(y_H/l))/rad);
938
939
940     /* Berechnung der Gestelllaenge l_1 */
941
942     l_1 = (sqrt((pow(x_A0,2))+(pow(y_A0,2))));
943
944     /* Bezeichnungen */
945
946     phi_1 = phi_R1;
947     phi_2 = phi_R1 + phi_An;
948     phi_3 = phi_R1 + phi_An + phi_R2;
949     phi_4 = phi_R1 + phi_An + phi_R2 + phi_Ab;
950     phi_5 = phi_R1 + phi_An + phi_R2 + phi_Ab + phi_R3;
951
952     /* Berechnung von s_0sk */
953
954     if(alpha_richtung == 'y')
955     {
956         if(positiv2 == 'y')
957         {
958             s_0sk = l_3-l_4*(sin(alpha_sk*rad));
959         }
960         else if(positiv2 == 'n')
961         {
962             s_0sk = (-1)*(l_3-l_4*(sin(alpha_sk*rad)));
963         }
964     }
965     else if(alpha_richtung == 'n')
966     {

```

```

967     s_0sk = l_3+l_4*(sin(alpha_sk*rad));
968 }
969
970     e_sk = l_4*(cos(alpha_sk*rad));
971
972     /* implizite Typkonvertierungen */
973
974     dphi_1 = phi_1;
975     dphi_2 = phi_2;
976     dphi_3 = phi_3;
977     dphi_4 = phi_4;
978     dphi_5 = phi_5;
979     dalpha = alpha;
980     dgamma = gamma;
981     ds_0sk = s_0sk;
982     dl_1 = l_1;
983     de_sk = e_sk;
984
985     /* Debugging */
986
987     QString version = "opticurv version 1.0.3 - TU Dresden";
988     QDate datum = QDate::currentDate();
989     QTime zeit = QTime::currentTime();
990     QString dt = filename + "\n" + version + " - " + datum.toString() +
991         " - " + zeit.toString();
992     QString datetime = "<blockquote><br>" + version + "<br>" +
993         datum.toString() + " - " + zeit.toString() +
994         "<br>" + filename + "<br><br></blockquote>";
995     qDebug( dt );
996     qDebug( "phi_1 = %Lg; phi_2 = %Lg; phi_3 = %Lg; phi_4 = %Lg; phi_5 = %Lg",
997         phi_1, phi_2, phi_3, phi_4, phi_5);
998     qDebug( "alpha = %Lg; gamma = %Lg; s_0sk = %Lg; l_1 = %Lg; e_sk = %Lg",
999         alpha, gamma, s_0sk, l_1, e_sk);
1000
1001     QString erg_ausgabe_phi_1=anzeige->text();
1002     erg_ausgabe_phi_1 = "<br><br><blockquote> phi_1 = " +
1003         erg_ausgabe_phi_1.setNum( dphi_1 ) + " Grad";
1004
1005     QString erg_ausgabe_phi_2=anzeige->text();
1006     erg_ausgabe_phi_2 = "<br> phi_2 = " + erg_ausgabe_phi_2.setNum( dphi_2 ) +
1007         " Grad";
1008
1009     QString erg_ausgabe_phi_3=anzeige->text();
1010     erg_ausgabe_phi_3 = "<br> phi_3 = " + erg_ausgabe_phi_3.setNum( dphi_3 ) +

```

```

1011         " Grad";
1012
1013     QString erg_ausgabe_phi_4=anzeige->text();
1014     erg_ausgabe_phi_4 = "<br> phi_4 = " + erg_ausgabe_phi_4.setNum( dphi_4 ) +
1015         " Grad";
1016
1017     QString erg_ausgabe_phi_5=anzeige->text();
1018     erg_ausgabe_phi_5 = "<br> phi_5 = " + erg_ausgabe_phi_5.setNum( dphi_5 ) +
1019         " Grad";
1020
1021     QString erg_ausgabe_alpha=anzeige->text();
1022     erg_ausgabe_alpha = "<br> alpha = " + erg_ausgabe_alpha.setNum( dalpha ) +
1023         " Grad";
1024
1025     QString erg_ausgabe_gamma=anzeige->text();
1026     erg_ausgabe_gamma = "<br> gamma = " + erg_ausgabe_gamma.setNum( dgamma ) +
1027         " Grad";
1028
1029     QString erg_ausgabe_s_0sk=anzeige->text();
1030     erg_ausgabe_s_0sk = "<br> s_0sk = " + erg_ausgabe_s_0sk.setNum( ds_0sk ) +
1031         " mm";
1032
1033     QString erg_ausgabe_l_1=anzeige->text();
1034     erg_ausgabe_l_1 = "<br> l_1 = " + erg_ausgabe_l_1.setNum( dl_1 ) +
1035         " mm";
1036
1037     QString erg_ausgabe_e_sk=anzeige->text();
1038     erg_ausgabe_e_sk = "<br> e_sk = " + erg_ausgabe_e_sk.setNum( de_sk ) +
1039         " mm";
1040
1041
1042     static const char* tabellenkopf[]={ " <html><head></head>"
1043                                         " <body><table>"
1044                                         " <thead><tr>"
1045                                         " <th>phi</th>"
1046                                         " <th>s(phi)</th>"
1047                                         " <th>psi(phi)</th>"
1048                                         " <th>psi(phi)</th>"
1049                                         " <th>psi'(phi)</th>"
1050                                         " <th>psi''(phi)</th>"
1051                                         " <th>x_Bik</th>"
1052                                         " <th>y_Bik</th>"
1053                                         " <th>x_i_vers</th>"
1054                                         " <th>y_i_vers</th>"

```



```

1055         "<th>x_B_vers</th>"
1056         "<th>y_B_vers</th>"
1057         "<th>x_a_vers</th>"
1058         "<th>y_a_vers</th>"
1059         "<th>mue</th>"
1060         "<th>r_K</th>"
1061         "</tr></thead>"
1062         "<tbody>"
1063         "<tr>"
1064         "<td>[Grad]</td>"
1065         "<td>[mm]</td>"
1066         "<td>[Grad]</td>"
1067         "<td>[rad]</td>"
1068         "<td>[rad]</td>"
1069         "<td>[rad]</td>"
1070         "<td>[mm]</td>"
1071         "<td>[mm]</td>"
1072         "<td>[mm]</td>"
1073         "<td>[mm]</td>"
1074         "<td>[mm]</td>"
1075         "<td>[mm]</td>"
1076         "<td>[mm]</td>"
1077         "<td>[mm]</td>"
1078         "<td>[Grad]</td>"
1079         "<td>[mm]</td>"
1080         "</tr>",0};
1081
1082     text+=datetime + tabellenkopf[0];
1083
1084     if(s_richtung == 'y')
1085     {
1086         for(phi=0; phi<360; phi+=n_1)
1087         {
1088             /* Hub in den Bewegungsabschnitten von phi */
1089
1090             s_1 = 0;
1091             s_2 = ((s_H/(pow(phi_An, 3)))*(10*(pow(phi-phi_1, 3))-(15/phi_An)*
1092                 (pow(phi-phi_1, 4))+(6/(pow(phi_An, 2)))*(pow(phi-phi_1, 5))));
1093             s_3 = s_H;
1094             s_4 = (s_H-(s_H/(pow(phi_Ab, 3)))*(10*(pow(phi-phi_3, 3))-(15/phi_Ab)*
1095                 (pow(phi-phi_3, 4))+(6/(pow(phi_Ab, 2)))*(pow(phi-phi_3, 5))));
1096             s_5 = 0;
1097
1098             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */

```

```

1099
1100     s_1_strich=0;
1101     s_2_strich=((s_H/(pow(phi_An,3)))*(30*(pow((phi-phi_1),2))-
1102               (60/phi_An)*(pow((phi-phi_1),3))+(30/(pow(phi_An,2)))*
1103               (pow((phi-phi_1),4))));
1104     s_3_strich=0;
1105     s_4_strich=(((-s_H)/(pow(phi_Ab,3)))*(30*(pow((phi-phi_3),2))-
1106               (60/phi_Ab)*(pow((phi-phi_3),3))+(30/(pow(phi_Ab,2)))*
1107               (pow((phi-phi_3),4))));
1108     s_5_strich=0;
1109
1110     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
1111
1112     s_1_zweistrich=0;
1113     s_2_zweistrich=((s_H/(pow(phi_An,3)))*(60*(phi-phi_1)-((180/phi_An)*
1114               (pow((phi-phi_1),2)))+((120/(pow(phi_An,2)))*
1115               (pow((phi-phi_1),3))));
1116     s_3_zweistrich=0;
1117     s_4_zweistrich=((-s_H/(pow(phi_Ab,3)))*(60*(phi-phi_3)-((180/phi_Ab)*
1118               (pow((phi-phi_3),2)))+((120/(pow(phi_Ab,2)))*
1119               (pow((phi-phi_3),3))));
1120     s_5_zweistrich=0;
1121
1122     /* Schleife für Bewegungsabschnitte von phi */
1123
1124     if(phi>=0 && phi<phi_1)
1125     {
1126         /* Hubverlaufs Berechnung */
1127
1128         s = 0;
1129
1130         /* Substitutionen für psi */
1131
1132         A = (2*l_4*(s_0sk+s_1));
1133         C = (pow((s_0sk+s_1),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1134
1135         /* Substitutionen für psi_strich */
1136
1137         D = (2*l_4*s_1_strich);
1138         E = (2*(s_0sk+s_1)*s_1_strich);
1139         F = (2*l_4*(s_0sk+s_1));
1140
1141         /* Substitutionen für psi_zweistrich */
1142

```

```

1143     H = (2*l_4*(s_0sk+s_1));
1144     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1145     J = (4*l_4*s_1_strich);
1146     K = (2*l_4*s_1_zweistrich);
1147     L = (2*(pow(s_1_strich,2))+2*(s_0sk+s_1)*s_1_zweistrich);
1148 }
1149 else if(phi>= phi_1 && phi<phi_2)
1150 {
1151     /* Hubverlaufsrechnung */
1152
1153     s = ((s_H/(pow(phi_An, 3)))*(10*(pow(phi-phi_1, 3))-(15/phi_An)*
1154         (pow(phi-phi_1, 4))+(6/(pow(phi_An, 2)))*(pow(phi-phi_1, 5))));
1155
1156     /* Substitutionen für psi */
1157
1158     A = (2*l_4*(s_0sk+s_2));
1159     C = (pow((s_0sk+s_2),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1160
1161     /* Substitutionen für psi_strich */
1162
1163     D = (2*l_4*s_2_strich);
1164     E = (2*(s_0sk+s_2)*s_2_strich);
1165     F = (2*l_4*(s_0sk+s_2));
1166
1167     /* Substitutionen für psi_zweistrich */
1168
1169     H = (2*l_4*(s_0sk+s_2));
1170     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1171     J = (4*l_4*s_2_strich);
1172     K = (2*l_4*s_2_zweistrich);
1173     L = (2*(pow(s_2_strich,2))+2*(s_0sk+s_2)*s_2_zweistrich);
1174 }
1175 else if(phi>=phi_2 && phi<phi_3)
1176 {
1177     /* Hubverlaufsrechnung */
1178
1179     s = s_H;
1180
1181     /* Substitutionen für psi */
1182
1183     A = (2*l_4*(s_0sk+s_3));
1184     C = (pow((s_0sk+s_3),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1185
1186     /* Substitutionen für psi_strich */

```

```

1187
1188     D = (2*l_4*s_3_strich);
1189     E = (2*(s_0sk+s_3)*s_3_strich);
1190     F = (2*l_4*(s_0sk+s_3));
1191
1192     /* Substitutionen für psi_zweistrich */
1193
1194     H = (2*l_4*(s_0sk+s_3));
1195     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1196     J = (4*l_4*s_3_strich);
1197     K = (2*l_4*s_3_zweistrich);
1198     L = (2*(pow(s_3_strich,2))+2*(s_0sk+s_3)*s_3_zweistrich);
1199 }
1200 else if(phi>=phi_3 && phi<phi_4)
1201 {
1202     /* Hubverlaufsberechnung */
1203
1204     s = (s_H-(s_H/(pow(phi_Ab, 3)))*(10*(pow(phi-phi_3, 3))-(15/phi_Ab)*
1205         (pow(phi-phi_3, 4))+(6/(pow(phi_Ab, 2)))*(pow(phi-phi_3, 5)))));
1206
1207     /* Substitutionen für psi */
1208
1209     A = (2*l_4*(s_0sk+s_4));
1210     C = (pow((s_0sk+s_4),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1211
1212     /* Substitutionen für psi_strich */
1213
1214     D = (2*l_4*s_4_strich);
1215     E = (2*(s_0sk+s_4)*s_4_strich);
1216     F = (2*l_4*(s_0sk+s_4));
1217
1218     /* Substitutionen für psi_zweistrich */
1219
1220     H = (2*l_4*(s_0sk+s_4));
1221     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1222     J = (4*l_4*s_4_strich);
1223     K = (2*l_4*s_4_zweistrich);
1224     L = (2*(pow(s_4_strich,2))+2*(s_0sk+s_4)*s_4_zweistrich);
1225 }
1226 else
1227 {
1228     /* Hubverlaufsberechnung */
1229
1230     s = 0;

```

```

1231
1232     /* Substitutionen für psi */
1233
1234     A = (2*l_4*(s_0sk+s_5));
1235     C = (pow((s_0sk+s_5),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1236
1237     /* Substitutionen für psi_strich */
1238
1239     D = (2*l_4*s_5_strich);
1240     E = (2*(s_0sk+s_5)*s_5_strich);
1241     F = (2*l_4*(s_0sk+s_5));
1242
1243     /* Substitutionen für psi_zweistrich */
1244
1245     H = (2*l_4*(s_0sk+s_5));
1246     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1247     J = (4*l_4*s_5_strich);
1248     K = (2*l_4*s_5_zweistrich);
1249     L = (2*(pow(s_5_strich,2))+2*(s_0sk+s_5)*s_5_zweistrich);
1250 }
1251 B = (-2*l_4*e_sk);
1252 G = (2*l_4*e_sk);
1253 A_0 = (2*l_4*(s_0sk));
1254 C_0 = (pow((s_0sk),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1255
1256 if(Ba_Qudrant == 'y')
1257 {
1258     psi_sk = (2*(atan((B+(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/
1259         (A-C)))-(beta*rad));
1260 }
1261 else if(Ba_Qudrant == 'n')
1262 {
1263     psi_sk = (2*(atan((B-(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/
1264         (A-C)))-(beta*rad));
1265 }
1266 if(Ba_Qudrant == 'y')
1267 {
1268     psi_sk0 = (2*(atan((B+(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/
1269         (A_0-C_0)))-(beta*rad));
1270 }
1271 else if(Ba_Qudrant == 'n')
1272 {
1273     psi_sk0 = (2*(atan((B-(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/
1274         (A_0-C_0)))-(beta*rad));

```

```

1275     }
1276
1277     if(positiv == 'y')
1278     {
1279         psi = (psi_sk - psi_sk0)/rad;
1280     }
1281     else if(positiv == 'n')
1282     {
1283         psi = (-psi_sk + psi_sk0)/rad;
1284     }
1285
1286     psi_rad = psi*rad;
1287     psi_strich = (((D*(cos((psi+beta)*rad))+E)/
1288                   (F*(sin((psi+beta)*rad))+G*(cos((psi+beta)*rad)))));
1289     psi_strich_Grad = (((D*(cos((psi+beta)*rad))+E)/
1290                       (F*(sin((psi+beta)*rad))+G*
1291                       (cos((psi+beta)*rad)))))/rad;
1292     Zaehler_psi_zweistrich = (-H*(pow(psi_strich,2))*
1293                               cos((psi+beta)*rad)+I*(pow(psi_strich,2))*
1294                               sin((psi+beta)*rad)-J*psi_strich*sin
1295                               ((psi+beta)*rad)+K*cos((psi+beta)*rad)+L);
1296     Nenner_psi_zweistrich = (I*cos((psi+beta)*rad)+M*sin((psi+beta)*rad));
1297     psi_zweistrich = (Zaehler_psi_zweistrich/Nenner_psi_zweistrich);
1298     psi_zweistrich_Grad = ((Zaehler_psi_zweistrich/Nenner_psi_zweistrich)/
1299                           (pow(rad,2)));
1300
1301     /* Berechnung von psi_0 */
1302
1303     psi_0 = (psi_sk0/rad);
1304
1305     /* Berechnung des Grundkreiswinkels psi_G */
1306
1307     alphan = (atan((y_A0/x_A0)));
1308     psi_G = ((acos(((pow(l,2))+(pow(l_1,2))-(pow(r_G,2)))/(2*l_1*l))))/rad);
1309     // psi_G = ((fabs(psi_0))-(fabs(alphan)));
1310
1311     /* Berechnung des Grundkreisradius r_G */
1312
1313     //r_G = 50.199;           // Behelfsansatz, muSS noch geändert werden
1314     //r_G = (sqrt((pow(l,2))+(pow(l_1,2))-(2*l_1*l*(cos(psi_G)))));
1315
1316     /* Substitutionen für Koordinaten des Rollenmittelpunktes */
1317
1318     M = (1-cos((phi)*rad)+(l/l_1)*cos((psi_G+psi+phi)*rad));

```

```

1319     N = (-sin((phi)*rad)+(1/l_1)*sin((psi_G+psi+phi)*rad));
1320
1321     /* Koordinaten des Rollenmittelpunktes */
1322
1323     x_Bik = (x_A0*M-y_A0*N);
1324     y_Bik = (x_A0*N+y_A0*M);
1325
1326     /* Ableitungen x'_Bik und y'_Bik */
1327
1328     x_Bik_strich = ((x_A0*sin((phi)*rad)+y_A0*cos((phi)*rad)-(1/l_1)*
1329                     (psi_strich+1)*(x_A0*sin((psi_G+psi+phi)*rad)+
1330                     y_A0*cos((psi_G+psi+phi)*rad))));
1331     y_Bik_strich = ((x_A0*cos((phi)*rad)-y_A0*sin((phi)*rad)-(1/l_1)*
1332                     (psi_strich+1)*(x_A0*cos((psi_G+psi+phi)*rad)-
1333                     y_A0*sin((psi_G+psi+phi)*rad))));
1334
1335     /* Betrag des Tangentenvektors B' */
1336
1337     Betrag_B_strich = (sqrt((pow((fabs(x_Bik_strich)),2))+
1338                             (pow((fabs(y_Bik_strich)),2))));
1339
1340     /* innere Koordinaten */
1341
1342     x_i = (x_Bik+r_R*(y_Bik_strich/Betrag_B_strich));
1343     y_i = (y_Bik+r_R*(x_Bik_strich/Betrag_B_strich));
1344
1345     /* äUSSere Koordinaten */
1346
1347     x_a = (x_Bik-r_R*(y_Bik_strich/Betrag_B_strich));
1348     y_a = (y_Bik-r_R*(x_Bik_strich/Betrag_B_strich));
1349
1350     /* Koordinatensystem versetzen */
1351
1352     x_i_versetzt = (x_i-x_A0);
1353     y_i_versetzt = (y_i-y_A0);
1354     x_B_versetzt = (x_Bik-x_A0);
1355     y_B_versetzt = (y_Bik-y_A0);
1356     x_a_versetzt = (x_a-x_A0);
1357     y_a_versetzt = (y_a-y_A0);
1358
1359     /* Berechnung von psi_0 */
1360
1361     psi_0 = (psi_sk0/rad);
1362

```

```

1363      /* Berechnung von mue */
1364
1365      r_Bik = (sqrt((pow((fabs(x_B_versetzt)),2))+
1366                  (pow((fabs(y_B_versetzt)),2))));
1367
1368      r_Bik_strich = (sqrt((pow((fabs(x_Bik_strich)),2))+
1369                          (pow((fabs(y_Bik_strich)),2))));
1370
1371      zaehler_mue = (x_Bik_strich*y_B_versetzt+x_B_versetzt*y_Bik_strich);
1372
1373      nenner_mue = (r_Bik*r_Bik_strich);
1374
1375      smue = (fabs(zaehler_mue/nenner_mue));
1376
1377      mue = (fabs((((-pi)/2)+acos(smue))/rad));
1378
1379      /* Berechnung von r_K */
1380
1381      Q = (cos(phi*rad)-(1/l_1)*(cos((psi_G+psi+phi)*rad))*
1382          (pow((fabs(psi_strich+1)),2))-(1/l_1)*
1383          (sin((psi_G+psi+phi)*rad))*psi_zweistrich);
1384
1385      R = (sin(phi*rad)-(1/l_1)*(sin((psi_G+psi+phi)*rad))*
1386          (pow((fabs(psi_strich+1)),2))-(1/l_1)*
1387          (cos((psi_G+psi+phi)*rad))*psi_zweistrich);
1388
1389      x_Bik_zweistrich = (x_A0*Q-y_A0*R);
1390
1391      S = (sin(phi*rad)-(1/l_1)*(sin((psi_G+psi+phi)*rad))*
1392          (pow((fabs(psi_strich+1)),2))+(1/l_1)*
1393          (cos((psi_G+psi+phi)*rad))*psi_zweistrich);
1394
1395      T = (cos(phi*rad)-(1/l_1)*(cos((psi_G+psi+phi)*rad))*
1396          (pow((fabs(psi_strich+1)),2))+(1/l_1)*
1397          (sin((psi_G+psi+phi)*rad))*psi_zweistrich);
1398
1399      y_Bik_zweistrich = (x_A0*S+y_A0*T);
1400
1401      wurzelausdruck = ((pow(x_Bik_strich, 2))+
1402                      (pow(y_Bik_strich, 2)));
1403
1404      potenz_wurzelausdruck = (pow(wurzelausdruck, 3));
1405
1406      zaehler_r_K = (sqrt(fabs(potenz_wurzelausdruck)));

```



```

1407
1408     nenner_r_K = (x_Bik_strich*y_Bik_zweistrich-
1409                  x_Bik_zweistrich*y_Bik_strich);
1410
1411     r_K = (zaehler_r_K/nenner_r_K);
1412
1413
1414     ofstream ofl;
1415
1416     ofl.open(filename_asc, ios::app);
1417     ofl << phi << ", "
1418         << s << ", "
1419         << psi << ", "
1420         << psi_rad << ", "
1421         << psi_strich_Grad << ", "
1422         << psi_zweistrich_Grad << ", "
1423         << x_Bik << ", "
1424         << y_Bik << ", "
1425         << x_i_versetzt << ", "
1426         << y_i_versetzt << ", "
1427         << x_B_versetzt << ", "
1428         << y_B_versetzt << ", "
1429         << x_a_versetzt << ", "
1430         << y_a_versetzt << ", "
1431         << mue << ", "
1432         << r_K << endl;    // endl : Manipulator, erzeugt Zeilenvorschub
1433     ofl.close();
1434
1435     ofl.open(filename_asc2, ios::app);
1436     ofl << x_i_versetzt << ", "
1437         << y_i_versetzt << ", "
1438         << x_B_versetzt << ", "
1439         << y_B_versetzt << ", "
1440         << x_a_versetzt << ", "
1441         << y_a_versetzt << endl;    // endl : Manipulator, erzeugt
1442                                     // Zeilenvorschub
1443     ofl.close();
1444
1445     /* implizite Typkonvertierungen für Ausgabe notw. */
1446
1447     dphi = phi;
1448     ds = s;
1449     dpsi = psi;
1450     dpsi_rad = psi_rad;

```

```

1451     dpsi_strich_Grad = psi_strich_Grad;
1452     dpsi_zweistrich_Grad = psi_zweistrich_Grad;
1453     dx_Bik = x_Bik;
1454     dy_Bik = y_Bik;
1455     dx_i_versetzt = x_i_versetzt;
1456     dy_i_versetzt = y_i_versetzt;
1457     dx_B_versetzt = x_B_versetzt;
1458     dy_B_versetzt = y_B_versetzt;
1459     dx_a_versetzt = x_a_versetzt;
1460     dy_a_versetzt = y_a_versetzt;
1461     dr_Bik = r_Bik;
1462     dmue = mue;
1463     dr_K = r_K;
1464
1465     QString erg_ausgabe_phi=anzeige->text();
1466     erg_ausgabe_phi.setNum( dphi );
1467     static const char* text_phi_a[] = {"<tr>"
1468                                         "<td>" ,0};
1469     static const char* text_phi_b[] = {"</td>" ,0};
1470     text+=text_phi_a[0]+erg_ausgabe_phi+text_phi_b[0];
1471
1472     QString erg_ausgabe_s=anzeige->text();
1473     erg_ausgabe_s.setNum( ds );
1474     static const char* text_s_a[] = {"<td>" ,0};
1475     static const char* text_s_b[] = {"</td>" ,0};
1476     text+=text_s_a[0]+erg_ausgabe_s+text_s_b[0];
1477
1478     QString erg_ausgabe_psi=anzeige->text();
1479     erg_ausgabe_psi.setNum( dpsi );
1480     static const char* text_psi_a[] = {"<td>" ,0};
1481     static const char* text_psi_b[] = {"</td>" ,0};
1482     text+=text_psi_a[0]+erg_ausgabe_psi+text_psi_b[0];
1483
1484     QString erg_ausgabe_psi_rad=anzeige->text();
1485     erg_ausgabe_psi_rad.setNum( dpsi_rad );
1486     static const char* text_psi_rad_a[] = {"<td>" ,0};
1487     static const char* text_psi_rad_b[] = {"</td>" ,0};
1488     text+=text_psi_rad_a[0]+erg_ausgabe_psi_rad+text_psi_rad_b[0];
1489
1490     QString erg_ausgabe_psi_strich_Grad=anzeige->text();
1491     erg_ausgabe_psi_strich_Grad.setNum( dpsi_strich_Grad );
1492     static const char* text_psi_strich_Grad_a[] = {"<td>" ,0};
1493     static const char* text_psi_strich_Grad_b[] = {"</td>" ,0};
1494     text+=text_psi_strich_Grad_a[0]+erg_ausgabe_psi_strich_Grad+

```

```

1495         text_psi_strich_Grad_b[0];
1496
1497     QString erg_ausgabe_psi_zweistrich_Grad=anzeige->text();
1498     erg_ausgabe_psi_zweistrich_Grad.setNum( dpsi_zweistrich_Grad );
1499     static const char* text_psi_zweistrich_Grad_a[] = {"<td>" ,0};
1500     static const char* text_psi_zweistrich_Grad_b[] = {"</td>" ,0};
1501     text+=text_psi_zweistrich_Grad_a[0]+erg_ausgabe_psi_zweistrich_Grad+
1502         text_psi_zweistrich_Grad_b[0];
1503
1504     QString erg_ausgabe_x_Bik=anzeige->text();
1505     erg_ausgabe_x_Bik.setNum( dx_Bik );
1506     static const char* text_x_Bik_a[] = {"<td>" ,0};
1507     static const char* text_x_Bik_b[] = {"</td>" ,0};
1508     text+=text_x_Bik_a[0]+erg_ausgabe_x_Bik+text_x_Bik_b[0];
1509
1510     QString erg_ausgabe_y_Bik=anzeige->text();
1511     erg_ausgabe_y_Bik.setNum( dy_Bik );
1512     static const char* text_y_Bik_a[] = {"<td>" ,0};
1513     static const char* text_y_Bik_b[] = {"</td>" ,0};
1514     text+=text_y_Bik_a[0]+erg_ausgabe_y_Bik+text_y_Bik_b[0];
1515
1516     QString erg_ausgabe_x_i_versetzt=anzeige->text();
1517     erg_ausgabe_x_i_versetzt.setNum( dx_i_versetzt );
1518     static const char* text_x_i_versetzt_a[] = {"<td>" ,0};
1519     static const char* text_x_i_versetzt_b[] = {"</td>" ,0};
1520     text+=text_x_i_versetzt_a[0]+erg_ausgabe_x_i_versetzt+
1521         text_x_i_versetzt_b[0];
1522
1523     QString erg_ausgabe_y_i_versetzt=anzeige->text();
1524     erg_ausgabe_y_i_versetzt.setNum( dy_i_versetzt );
1525     static const char* text_y_i_versetzt_a[] = {"<td>" ,0};
1526     static const char* text_y_i_versetzt_b[] = {"</td>" ,0};
1527     text+=text_y_i_versetzt_a[0]+erg_ausgabe_y_i_versetzt+
1528         text_y_i_versetzt_b[0];
1529
1530     QString erg_ausgabe_x_B_versetzt=anzeige->text();
1531     erg_ausgabe_x_B_versetzt.setNum( dx_B_versetzt );
1532     static const char* text_x_B_versetzt_a[] = {"<td>" ,0};
1533     static const char* text_x_B_versetzt_b[] = {"</td>" ,0};
1534     text+=text_x_B_versetzt_a[0]+erg_ausgabe_x_B_versetzt+
1535         text_x_B_versetzt_b[0];
1536
1537     QString erg_ausgabe_y_B_versetzt=anzeige->text();
1538     erg_ausgabe_y_B_versetzt.setNum( dy_B_versetzt );

```

```

1539     static const char* text_y_B_versetzt_a[] = {"<td>" ,0};
1540     static const char* text_y_B_versetzt_b[] = {"</td>" ,0};
1541     text+=text_y_B_versetzt_a[0]+erg_ausgabe_y_B_versetzt+
1542         text_y_B_versetzt_b[0];
1543
1544     QString erg_ausgabe_x_a_versetzt=anzeige->text();
1545     erg_ausgabe_x_a_versetzt.setNum( dx_a_versetzt );
1546     static const char* text_x_a_versetzt_a[] = {"<td>" ,0};
1547     static const char* text_x_a_versetzt_b[] = {"</td>" ,0};
1548     text+=text_x_a_versetzt_a[0]+erg_ausgabe_x_a_versetzt+
1549         text_x_a_versetzt_b[0];
1550
1551     QString erg_ausgabe_y_a_versetzt=anzeige->text();
1552     erg_ausgabe_y_a_versetzt.setNum( dy_a_versetzt );
1553     static const char* text_y_a_versetzt_a[] = {"<td>" ,0};
1554     static const char* text_y_a_versetzt_b[] = {"</td>" ,0};
1555     text+=text_y_a_versetzt_a[0]+erg_ausgabe_y_a_versetzt+
1556         text_y_a_versetzt_b[0];
1557
1558     QString erg_ausgabe_mue=anzeige->text();
1559     erg_ausgabe_mue.setNum( dmue );
1560     static const char* text_mue_a_a[] = {"<td>" ,0};
1561     static const char* text_mue_a_b[] = {"</td>" ,0};
1562     text+=text_mue_a_a[0]+erg_ausgabe_mue+
1563         text_mue_a_b[0];
1564
1565     QString erg_ausgabe_r_K=anzeige->text();
1566     erg_ausgabe_r_K.setNum( dr_K );
1567     static const char* text_r_K_a[] = {"<td>" ,0};
1568     static const char* text_r_K_b[] = {"</td>"
1569                                         "</tr>" ,0};
1570     text+=text_r_K_a[0]+erg_ausgabe_r_K+
1571         text_r_K_b[0];
1572
1573 }
1574 }
1575 else if(s_richtung == 'n')
1576 {
1577     for(phi=0; phi<360; phi+=n_1)
1578     {
1579         /* Hub in den Bewegungsabschnitten von phi */
1580
1581         s_1 = 0;
1582         s_2 = ((s_H/(pow(phi_An, 3)))*(10*(pow(phi-phi_1, 3))-(15/phi_An)*

```

```

1583         (pow(phi-phi_1, 4))+(6/(pow(phi_An, 2)))*
1584         (pow(phi-phi_1, 5))));
1585 s_3 = s_H;
1586 s_4 = (s_H-(s_H/(pow(phi_Ab, 3)))*(10*(pow(phi-phi_3, 3))-(15/phi_Ab)*
1587         (pow(phi-phi_3, 4))+(6/(pow(phi_Ab, 2)))*
1588         (pow(phi-phi_3, 5))));
1589 s_5 = 0;
1590
1591 /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
1592
1593 s_1_strich=0;
1594 s_2_strich=((s_H/(pow(phi_An,3)))*
1595             (30*(pow((phi-phi_1),2))-(60/phi_An)*
1596             (pow((phi-phi_1),3))+(30/(pow(phi_An,2)))*
1597             (pow((phi-phi_1),4))));
1598 s_3_strich=0;
1599 s_4_strich=(((-s_H)/(pow(phi_Ab,3)))*
1600             ((30*(pow((phi-phi_3),2)))-((60/phi_Ab)*
1601             (pow((phi-phi_3),3)))+(30/(pow(phi_Ab,2)))*
1602             (pow((phi-phi_3),4))));
1603 s_5_strich=0;
1604
1605
1606 /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
1607
1608 s_1_zweistrich=0;
1609 s_2_zweistrich=((s_H/(pow(phi_An,3)))*(60*(phi-phi_1)-((180/phi_An)*
1610             (pow((phi-phi_1),2)))+(120/(pow(phi_An,2)))*
1611             (pow((phi-phi_1),3))));
1612 s_3_zweistrich=0;
1613 s_4_zweistrich=(((-s_H/(pow(phi_Ab,3)))*(60*(phi-phi_3)-((180/phi_Ab)*
1614             (pow((phi-phi_3),2)))+(120/(pow(phi_Ab,2)))*
1615             (pow((phi-phi_3),3))));
1616 s_5_zweistrich=0;
1617
1618
1619 if(phi>=0 && phi<phi_1)
1620 {
1621     /* Hubverlaufsrechnung */
1622
1623     s = 0;
1624
1625     /* Substitutionen für psi */
1626

```

```

1627     A = (2*l_4*(s_0sk-s_1));
1628     C = (pow((s_0sk-s_1),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1629
1630     /* Substitutionen für psi_strich */
1631
1632     D = (-2*l_4*s_1_strich);
1633     E = (-2*(s_0sk-s_1)*s_1_strich);
1634     F = (2*l_4*(s_0sk-s_1));
1635
1636     /* Substitutionen für psi_zweistrich */
1637
1638     H = (2*l_4*(s_0sk-s_1));
1639     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1640     J = (4*l_4*s_1_strich);
1641     K = (2*l_4*s_1_zweistrich);
1642     L = (2*(pow(s_1_strich,2))-2*(s_0sk-s_1)*s_1_zweistrich);
1643 }
1644 else if(phi>= phi_1 && phi<phi_2)
1645 {
1646     /* Hubverlaufs berechnung */
1647
1648     s = ((s_H/(pow(phi_An, 3)))*(10*(pow(phi-phi_1, 3))-(15/phi_An)*
1649         (pow(phi-phi_1, 4))+(6/(pow(phi_An, 2)))*(pow(phi-phi_1, 5))));
1650
1651
1652     /* Substitutionen für psi */
1653
1654     A = (2*l_4*(s_0sk-s_2));
1655     C = (pow((s_0sk-s_2),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1656
1657     /* Substitutionen für psi_strich */
1658
1659     D = (-2*l_4*s_2_strich);
1660     E = (-2*(s_0sk-s_2)*s_2_strich);
1661     F = (2*l_4*(s_0sk-s_2));
1662
1663     /* Substitutionen für psi_zweistrich */
1664
1665     H = (2*l_4*(s_0sk-s_2));
1666     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1667     J = (4*l_4*s_2_strich);
1668     K = (2*l_4*s_2_zweistrich);
1669     L = (2*(pow(s_2_strich,2))-2*(s_0sk-s_2)*s_2_zweistrich);
1670 }

```

```

1671     else if(phi>=phi_2 && phi<phi_3)
1672     {
1673         /* Hubverlaufsrechnung */
1674
1675         s = s_H;
1676
1677         /* Substitutionen für psi */
1678
1679         A = (2*l_4*(s_0sk-s_3));
1680         C = (pow((s_0sk-s_3),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1681
1682         /* Substitutionen für psi_strich */
1683
1684         D = (-2*l_4*s_3_strich);
1685         E = (-2*(s_0sk-s_3)*s_3_strich);
1686         F = (2*l_4*(s_0sk-s_3));
1687
1688         /* Substitutionen für psi_zweistrich */
1689
1690         H = (2*l_4*(s_0sk-s_3));
1691         I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1692         J = (4*l_4*s_3_strich);
1693         K = (2*l_4*s_3_zweistrich);
1694         L = (2*(pow(s_3_strich,2))-2*(s_0sk-s_3)*s_3_zweistrich);
1695     }
1696     else if(phi>=phi_3 && phi<phi_4)
1697     {
1698         /* Hubverlaufsrechnung */
1699
1700         s = (s_H-(s_H/(pow(phi_Ab, 3)))*(10*(pow(phi-phi_3, 3))-(15/phi_Ab)*
1701             (pow(phi-phi_3, 4))+(6/(pow(phi_Ab, 2)))*(pow(phi-phi_3, 5))));
1702
1703         /* Substitutionen für psi */
1704
1705         A = (2*l_4*(s_0sk-s_4));
1706         C = (pow((s_0sk-s_4),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1707
1708         /* Substitutionen für psi_strich */
1709
1710         D = (-2*l_4*s_4_strich);
1711         E = (-2*(s_0sk-s_4)*s_4_strich);
1712         F = (2*l_4*(s_0sk-s_4));
1713
1714         /* Substitutionen für psi_zweistrich */

```

```

1715
1716     H = (2*l_4*(s_0sk-s_4));
1717     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1718     J = (4*l_4*s_4_strich);
1719     K = (2*l_4*s_4_zweistrich);
1720     L = (2*(pow(s_4_strich,2))-2*(s_0sk-s_4)*s_4_zweistrich);
1721 }
1722 else
1723 {
1724     /* Hubverlaufsrechnung */
1725
1726     s = 0;
1727
1728     /* Substitutionen für psi */
1729
1730     A = (2*l_4*(s_0sk-s_5));
1731     C = (pow((s_0sk-s_5),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1732
1733     /* Substitutionen für psi_strich */
1734
1735     D = (-2*l_4*s_5_strich);
1736     E = (-2*(s_0sk-s_5)*s_5_strich);
1737     F = (2*l_4*(s_0sk-s_5));
1738
1739     /* Substitutionen für psi_zweistrich */
1740
1741     H = (2*l_4*(s_0sk-s_5));
1742     I = (2*(pow(l_4,2))*cos((alpha_sk)*rad));
1743     J = (4*l_4*s_5_strich);
1744     K = (2*l_4*s_5_zweistrich);
1745     L = (2*(pow(s_5_strich,2))-2*(s_0sk-s_5)*s_5_zweistrich);
1746 }
1747 B = (-2*l_4*e_sk);
1748 G = (2*l_4*e_sk);
1749 A_0 = (2*l_4*(s_0sk));
1750 C_0 = (pow((s_0sk),2))+(pow(e_sk,2))-(pow(l_3,2))+(pow(l_4,2));
1751
1752 /* if(Ba_Qudrant == 'y')
1753 {
1754     psi_sk = (2*(atan((B+(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/(
1755         (A-C)))));
1756 }
1757 else if(Ba_Qudrant == 'n')
1758 {

```



```

1759     psi_sk = (2*(atan((B-(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/(
1760         (A-C))));
1761 }
1762 if(Ba_Qudrant == 'y')
1763 {
1764     psi_sk0 = (2*(atan((B+(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/(
1765         (A_0-C_0))));
1766 }
1767 else if(Ba_Qudrant == 'n')
1768 {
1769     psi_sk0 = (2*(atan((B-(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/(
1770         (A_0-C_0))));
1771 } */
1772
1773 if(Ba_Qudrant == 'y')
1774 {
1775     psi_sk = (2*(atan((B+(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/(
1776         (A-C)))-(beta*rad));
1777 }
1778 else if(Ba_Qudrant == 'n')
1779 {
1780     psi_sk = (2*(atan((B-(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/(
1781         (A-C)))-(beta*rad));
1782 }
1783 if(Ba_Qudrant == 'y')
1784 {
1785     psi_sk0 = (2*(atan((B+(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/(
1786         (A_0-C_0)))-(beta*rad));
1787 }
1788 else if(Ba_Qudrant == 'n')
1789 {
1790     psi_sk0 = (2*(atan((B-(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/(
1791         (A_0-C_0)))-(beta*rad));
1792 }
1793
1794 if(positiv == 'y')
1795 {
1796     psi = (psi_sk - psi_sk0)/rad;
1797 }
1798 else if(positiv == 'n')
1799 {
1800     psi = (-psi_sk + psi_sk0)/rad;
1801 }
1802

```

```

1803     psi_rad = psi*rad;
1804     psi_strich = (((D*((cos((psi+beta)*rad)))+E)/
1805                 (F*(sin((psi+beta)*rad))+G*(cos((psi+beta)*rad)))));
1806     psi_strich_Grad = (((D*((cos((psi+beta)*rad)))+E)/
1807                      (F*(sin((psi+beta)*rad))+G*(cos((psi+beta)*rad)))))/rad);
1808     Zaehler_psi_zweistrich = (-H*(pow(psi_strich,2))*
1809                              cos((psi+beta)*rad)+I*(pow(psi_strich,2))*
1810                              sin((psi+beta)*rad)+J*psi_strich*
1811                              sin((psi+beta)*rad)-K*cos((psi+beta)*rad)+L);
1812     Nenner_psi_zweistrich = (I*cos((psi+beta)*rad)+M*sin((psi+beta)*rad));
1813     psi_zweistrich = ((Zaehler_psi_zweistrich/Nenner_psi_zweistrich));
1814     psi_zweistrich_Grad = ((Zaehler_psi_zweistrich/Nenner_psi_zweistrich)/
1815                          (pow(rad,2)));
1816
1817     /* Berechnung von psi_0 */
1818
1819     psi_0 = (psi_sk0/rad);
1820
1821     /* Berechnung des Grundkreiswinkels psi_G */
1822
1823     psi_G = ((acos(((pow(l,2))+(pow(l_1,2))-(pow(r_G,2)))/(2*l_1*l_1)))/rad);
1824
1825     /* Substitutionen für Koordinaten des Rollenmittelpunktes */
1826
1827     M = (1-cos((phi)*rad)+(l/l_1)*cos((psi_G+psi+phi)*rad));
1828     N = (-sin((phi)*rad)+(l/l_1)*sin((psi_G+psi+phi)*rad));
1829
1830     /* Koordinaten des Rollenmittelpunktes */
1831
1832     x_Bik = (x_A0*M-y_A0*N);
1833     y_Bik = (x_A0*N+y_A0*M);
1834
1835     /* Ableitungen x'_Bik und y'_Bik */
1836
1837     x_Bik_strich = ((x_A0*sin((phi)*rad)+y_A0*cos((phi)*rad)-(l/l_1)*
1838                   (psi_strich+1)*(x_A0*sin((psi_G+psi+phi)*rad)+
1839                   y_A0*cos((psi_G+psi+phi)*rad))));
1840     y_Bik_strich = ((x_A0*cos((phi)*rad)-y_A0*sin((phi)*rad)-(l/l_1)*
1841                   (psi_strich+1)*(x_A0*cos((psi_G+psi+phi)*rad)-
1842                   y_A0*sin((psi_G+psi+phi)*rad))));
1843
1844     /* Betrag des Tangentenvektors B' */
1845
1846     Betrag_B_strich = (sqrt((pow((fabs(x_Bik_strich)),2))+

```

```

1847         (pow((fabs(y_Bik_strich)),2))));
1848
1849     /* innere Koordinaten */
1850
1851     x_i = (x_Bik+r_R*(y_Bik_strich/Betrag_B_strich));
1852     y_i = (y_Bik+r_R*(x_Bik_strich/Betrag_B_strich));
1853
1854     /* äUSSere Koordinaten */
1855
1856     x_a = (x_Bik-r_R*(y_Bik_strich/Betrag_B_strich));
1857     y_a = (y_Bik-r_R*(x_Bik_strich/Betrag_B_strich));
1858
1859     /* Koordinatensystem versetzen */
1860
1861     x_i_versetzt = (x_i-x_A0);
1862     y_i_versetzt = (y_i-y_A0);
1863     x_B_versetzt = (x_Bik-x_A0);
1864     y_B_versetzt = (y_Bik-y_A0);
1865     x_a_versetzt = (x_a-x_A0);
1866     y_a_versetzt = (y_a-y_A0);
1867
1868     /* Berechnung von psi_0 */
1869
1870     psi_0 = (psi_sk0/rad);
1871
1872     /* Berechnung von mue */
1873
1874     r_Bik = (sqrt((pow((fabs(x_B_versetzt)),2))+
1875         (pow((fabs(y_B_versetzt)),2))));
1876
1877     r_Bik_strich = (sqrt((pow((fabs(x_Bik_strich)),2))+
1878         (pow((fabs(y_Bik_strich)),2))));
1879
1880     zaehler_mue = (x_Bik_strich*y_B_versetzt+x_B_versetzt*y_Bik_strich);
1881
1882     nenner_mue = (r_Bik*r_Bik_strich);
1883
1884     smue = (fabs(zaehler_mue/nenner_mue));
1885
1886     mue = (fabs((((-pi)/2)+acos(smue))/rad));
1887
1888
1889     /* Berechnung von r_K */
1890

```

```

1891 Q = (cos(phi*rad)-(1/l_1)*(cos((psi_G+psi+phi)*rad))*
1892      (pow((fabs(psi_strich+1)),2))-(1/l_1)*
1893      (sin((psi_G+psi+phi)*rad))*psi_zweistrich);
1894
1895 R = (sin(phi*rad)-(1/l_1)*(sin((psi_G+psi+phi)*rad))*
1896      (pow((fabs(psi_strich+1)),2))-(1/l_1)*
1897      (cos((psi_G+psi+phi)*rad))*psi_zweistrich);
1898
1899 x_Bik_zweistrich = (x_A0*Q-y_A0*R);
1900
1901 S = (sin(phi*rad)-(1/l_1)*(sin((psi_G+psi+phi)*rad))*
1902      (pow((fabs(psi_strich+1)),2))+(1/l_1)*
1903      (cos((psi_G+psi+phi)*rad))*psi_zweistrich);
1904
1905 T = (cos(phi*rad)-(1/l_1)*(cos((psi_G+psi+phi)*rad))*
1906      (pow((fabs(psi_strich+1)),2))+(1/l_1)*
1907      (sin((psi_G+psi+phi)*rad))*psi_zweistrich);
1908
1909 y_Bik_zweistrich = (x_A0*S+y_A0*T);
1910
1911 wurzelausdruck = ((pow(x_Bik_strich, 2))+
1912                  (pow(y_Bik_strich, 2)));
1913
1914 potenz_wurzelausdruck = (pow(wurzelausdruck, 3));
1915
1916 zaehler_r_K = (sqrt(fabs(potenz_wurzelausdruck)));
1917
1918 nenner_r_K = (x_Bik_strich*y_Bik_zweistrich-
1919              x_Bik_zweistrich*y_Bik_strich);
1920
1921 r_K = (zaehler_r_K/nenner_r_K);
1922
1923 /* Datei für die Ausgabe öffnen ios::app --> neue Daten werden
1924                                stets am Dateiende angefügt */
1925
1926 ofstream ofl;
1927
1928 ofl.open(filename_asc, ios::app);
1929 ofl << phi << ", "
1930      << s << ", "
1931      << psi << ", "
1932      << psi_rad << ", "
1933      << psi_strich_Grad << ", "
1934      << psi_zweistrich_Grad << ", "

```

```

1935         << x_Bik << ","
1936         << y_Bik << ","
1937         << x_i_versetzt << ","
1938         << y_i_versetzt << ","
1939         << x_B_versetzt << ","
1940         << y_B_versetzt << ","
1941         << x_a_versetzt << ","
1942         << y_a_versetzt << ","
1943         << mue << ","
1944         << r_K << endl;    // endl : Manipulator, erzeugt Zeilenvorschub
1945 ofl.close();
1946
1947 ofl.open(filename_asc2, ios::app);
1948 ofl << x_i_versetzt << ","
1949         << y_i_versetzt << ","
1950         << x_B_versetzt << ","
1951         << y_B_versetzt << ","
1952         << x_a_versetzt << ","
1953         << y_a_versetzt << endl;    // endl : Manipulator, erzeugt
1954                                     // Zeilenvorschub
1955 ofl.close();
1956
1957 /* implizite Typkonvertierungen für Ausgabe notw. */
1958
1959 dphi = phi;
1960 ds = s;
1961 dpsi = psi;
1962 dpsi_rad = psi_rad;
1963 dpsi_strich_Grad = psi_strich_Grad;
1964 dpsi_zweistrich_Grad = psi_zweistrich_Grad;
1965 dx_Bik = x_Bik;
1966 dy_Bik = y_Bik;
1967 dx_i_versetzt = x_i_versetzt;
1968 dy_i_versetzt = y_i_versetzt;
1969 dx_B_versetzt = x_B_versetzt;
1970 dy_B_versetzt = y_B_versetzt;
1971 dx_a_versetzt = x_a_versetzt;
1972 dy_a_versetzt = y_a_versetzt;
1973 dr_Bik = r_Bik;
1974 dmue = mue;
1975 dsmue = smue;
1976 dr_K = r_K;
1977
1978 QString erg_ausgabe_phi=anzeige->text();

```

```

1979     erg_ausgabe_phi.setNum( dphi );
1980     static const char* text_phi_a[] = {"<tr>"
1981                                     "<td>" ,0};
1982     static const char* text_phi_b[] = {"</td>" ,0};
1983     text+=text_phi_a[0]+erg_ausgabe_phi+text_phi_b[0];
1984
1985     QString erg_ausgabe_s=anzeige->text();
1986     erg_ausgabe_s.setNum( ds );
1987     static const char* text_s_a[] = {"<td>" ,0};
1988     static const char* text_s_b[] = {"</td>" ,0};
1989     text+=text_s_a[0]+erg_ausgabe_s+text_s_b[0];
1990
1991     QString erg_ausgabe_psi=anzeige->text();
1992     erg_ausgabe_psi.setNum( dpsi );
1993     static const char* text_psi_a[] = {"<td>" ,0};
1994     static const char* text_psi_b[] = {"</td>" ,0};
1995     text+=text_psi_a[0]+erg_ausgabe_psi+text_psi_b[0];
1996
1997     QString erg_ausgabe_psi_rad=anzeige->text();
1998     erg_ausgabe_psi_rad.setNum( dpsi_rad );
1999     static const char* text_psi_rad_a[] = {"<td>" ,0};
2000     static const char* text_psi_rad_b[] = {"</td>" ,0};
2001     text+=text_psi_rad_a[0]+erg_ausgabe_psi_rad+text_psi_rad_b[0];
2002
2003     QString erg_ausgabe_psi_strich_Grad=anzeige->text();
2004     erg_ausgabe_psi_strich_Grad.setNum( dpsi_strich_Grad );
2005     static const char* text_psi_strich_Grad_a[] = {"<td>" ,0};
2006     static const char* text_psi_strich_Grad_b[] = {"</td>" ,0};
2007     text+=text_psi_strich_Grad_a[0]+erg_ausgabe_psi_strich_Grad+
2008         text_psi_strich_Grad_b[0];
2009
2010     QString erg_ausgabe_psi_zweistrich_Grad=anzeige->text();
2011     erg_ausgabe_psi_zweistrich_Grad.setNum( dpsi_zweistrich_Grad );
2012     static const char* text_psi_zweistrich_Grad_a[] = {"<td>" ,0};
2013     static const char* text_psi_zweistrich_Grad_b[] = {"</td>" ,0};
2014     text+=text_psi_zweistrich_Grad_a[0]+erg_ausgabe_psi_zweistrich_Grad+
2015         text_psi_zweistrich_Grad_b[0];
2016
2017     QString erg_ausgabe_x_Bik=anzeige->text();
2018     erg_ausgabe_x_Bik.setNum( dx_Bik );
2019     static const char* text_x_Bik_a[] = {"<td>" ,0};
2020     static const char* text_x_Bik_b[] = {"</td>" ,0};
2021     text+=text_x_Bik_a[0]+erg_ausgabe_x_Bik+text_x_Bik_b[0];
2022

```

```

2023     QString erg_ausgabe_y_Bik=anzeige->text();
2024     erg_ausgabe_y_Bik.setNum( dy_Bik );
2025     static const char* text_y_Bik_a[] = {"<td>" ,0};
2026     static const char* text_y_Bik_b[] = {"</td>" ,0};
2027     text+=text_y_Bik_a[0]+erg_ausgabe_y_Bik+text_y_Bik_b[0];
2028
2029     QString erg_ausgabe_x_i_versetzt=anzeige->text();
2030     erg_ausgabe_x_i_versetzt.setNum( dx_i_versetzt );
2031     static const char* text_x_i_versetzt_a[] = {"<td>" ,0};
2032     static const char* text_x_i_versetzt_b[] = {"</td>" ,0};
2033     text+=text_x_i_versetzt_a[0]+erg_ausgabe_x_i_versetzt+
2034         text_x_i_versetzt_b[0];
2035
2036     QString erg_ausgabe_y_i_versetzt=anzeige->text();
2037     erg_ausgabe_y_i_versetzt.setNum( dy_i_versetzt );
2038     static const char* text_y_i_versetzt_a[] = {"<td>" ,0};
2039     static const char* text_y_i_versetzt_b[] = {"</td>" ,0};
2040     text+=text_y_i_versetzt_a[0]+erg_ausgabe_y_i_versetzt+
2041         text_y_i_versetzt_b[0];
2042
2043     QString erg_ausgabe_x_B_versetzt=anzeige->text();
2044     erg_ausgabe_x_B_versetzt.setNum( dx_B_versetzt );
2045     static const char* text_x_B_versetzt_a[] = {"<td>" ,0};
2046     static const char* text_x_B_versetzt_b[] = {"</td>" ,0};
2047     text+=text_x_B_versetzt_a[0]+erg_ausgabe_x_B_versetzt+
2048         text_x_B_versetzt_b[0];
2049
2050     QString erg_ausgabe_y_B_versetzt=anzeige->text();
2051     erg_ausgabe_y_B_versetzt.setNum( dy_B_versetzt );
2052     static const char* text_y_B_versetzt_a[] = {"<td>" ,0};
2053     static const char* text_y_B_versetzt_b[] = {"</td>" ,0};
2054     text+=text_y_B_versetzt_a[0]+erg_ausgabe_y_B_versetzt+
2055         text_y_B_versetzt_b[0];
2056
2057     QString erg_ausgabe_x_a_versetzt=anzeige->text();
2058     erg_ausgabe_x_a_versetzt.setNum( dx_a_versetzt );
2059     static const char* text_x_a_versetzt_a[] = {"<td>" ,0};
2060     static const char* text_x_a_versetzt_b[] = {"</td>" ,0};
2061     text+=text_x_a_versetzt_a[0]+erg_ausgabe_x_a_versetzt+
2062         text_x_a_versetzt_b[0];
2063
2064     QString erg_ausgabe_y_a_versetzt=anzeige->text();
2065     erg_ausgabe_y_a_versetzt.setNum( dy_a_versetzt );
2066     static const char* text_y_a_versetzt_a[] = {"<td>" ,0};

```

```

2067     static const char* text_y_a_versetzt_b[] = {"</td>" ,0};
2068     text+=text_y_a_versetzt_a[0]+erg_ausgabe_y_a_versetzt+
2069         text_y_a_versetzt_b[0];
2070
2071     QString erg_ausgabe_mue=anzeige->text();
2072     erg_ausgabe_mue.setNum( dmue );
2073     static const char* text_mue_a_a[] = {"<td>" ,0};
2074     static const char* text_mue_a_b[] = {"</td>" ,0};
2075     text+=text_mue_a_a[0]+erg_ausgabe_mue+
2076         text_mue_a_b[0];
2077
2078     QString erg_ausgabe_r_K=anzeige->text();
2079     erg_ausgabe_r_K.setNum( dr_K );
2080     static const char* text_r_K_a[] = {"<td>" ,0};
2081     static const char* text_r_K_b[] = {"</td>"
2082                                         "</tr>" ,0};
2083     text+=text_r_K_a[0]+erg_ausgabe_r_K+
2084         text_r_K_b[0];
2085
2086 }
2087 }
2088 static const char* tabellenrest[] = {"</tbody>"
2089                                     "</table>"
2090                                     "</body>"
2091                                     "</html>" ,0};
2092
2093 /* implizite Typkonvertierungen für Ausgabe notw. */
2094
2095 dpsi_sk0 = psi_sk0;
2096 dpsi_0 = psi_0;
2097 dpsi_G = psi_G;
2098
2099 /* Debugging */
2100
2101 qDebug( "psi_sk0 = %Lg; psi_0 = %Lg; psi_G = %Lg;",
2102         psi_sk0, psi_0, psi_G);
2103
2104 QString erg_ausgabe_psi_sk0=anzeige->text();
2105 erg_ausgabe_psi_sk0 = "<br> psi_sk0 = " + erg_ausgabe_psi_sk0.setNum(dpsi_sk0)
2106                     + " Grad";
2107
2108 QString erg_ausgabe_psi_0=anzeige->text();
2109 erg_ausgabe_psi_0 = "<br> psi_0 = " + erg_ausgabe_psi_0.setNum( dpsi_0 ) +
2110                     " Grad";

```



```

2111
2112     QString erg_ausgabe_psi_G=anzeige->text();
2113     erg_ausgabe_psi_G = "<br> psi_G = " + erg_ausgabe_psi_G.setNum( dpsi_G ) +
2114         " Grad<br><br></blockquote><br><br>";
2115
2116     text+=tabellenrest[0] + erg_ausgabe_phi_1 + erg_ausgabe_phi_2 +
2117         erg_ausgabe_phi_3 + erg_ausgabe_phi_4 + erg_ausgabe_phi_5 +
2118         erg_ausgabe_alpha + erg_ausgabe_gamma + erg_ausgabe_s_0sk +
2119         erg_ausgabe_l_1 + erg_ausgabe_e_sk + erg_ausgabe_psi_sk0 +
2120         erg_ausgabe_psi_0 + erg_ausgabe_psi_G;
2121
2122     anzeige->setText(text);
2123
2124 }
2125
2126 #include "opticurv.moc"
2127
2128 //----- main -----
2129 int main( int argc, char* argv[] )
2130 {
2131
2132     #ifdef QT_DLL
2133         QApplication myapp(argc,argv);
2134     #else
2135         QApplication myapp(argc,argv);
2136     #endif
2137
2138     opticurv* mywidget = new opticurv();
2139     mywidget->setGeometry( 50, 50, 400, 400 );
2140
2141     myapp.setMainWidget( mywidget );
2142     mywidget->setCaption("opticurv version 1.0.3 - TU Dresden");
2143     mywidget->show();
2144     return myapp.exec();
2145 }

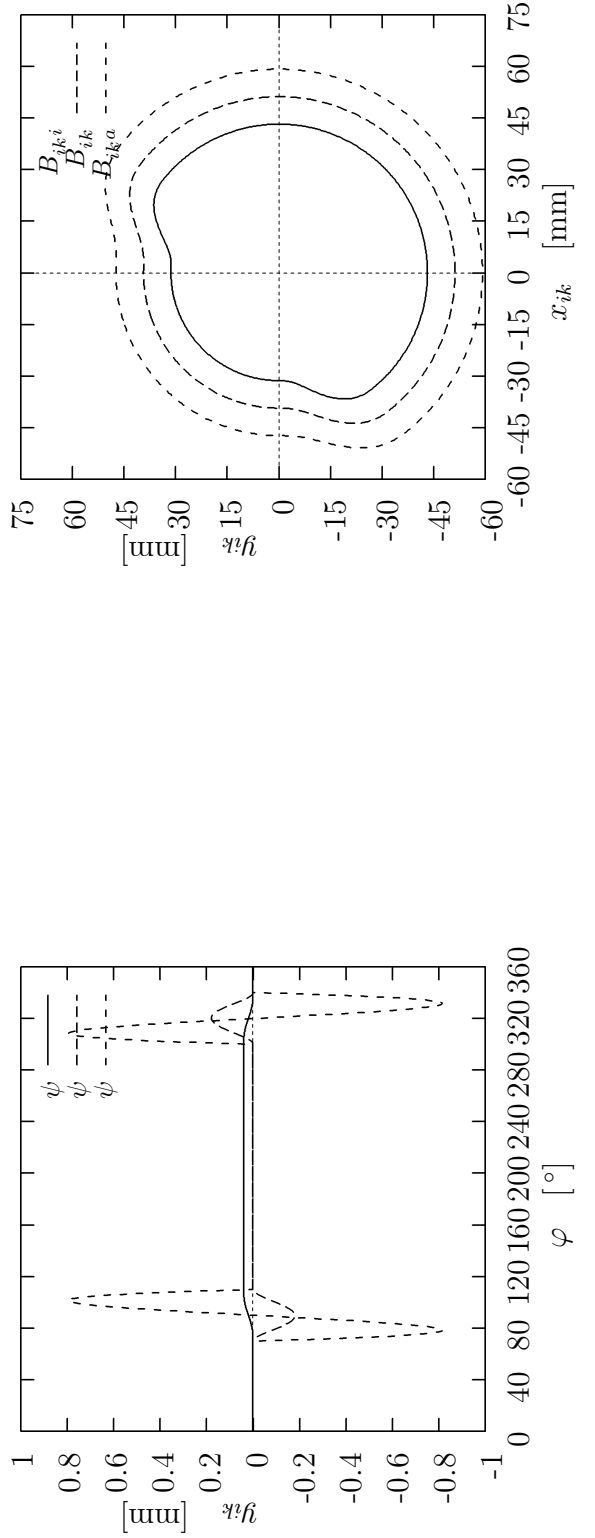
```

## B Beispiele

### B.1 Querhubkurvenscheibe (optcurv version 1.0.3)

Tabelle 4: Eingabeparameter für die Querhubkurvenscheibe

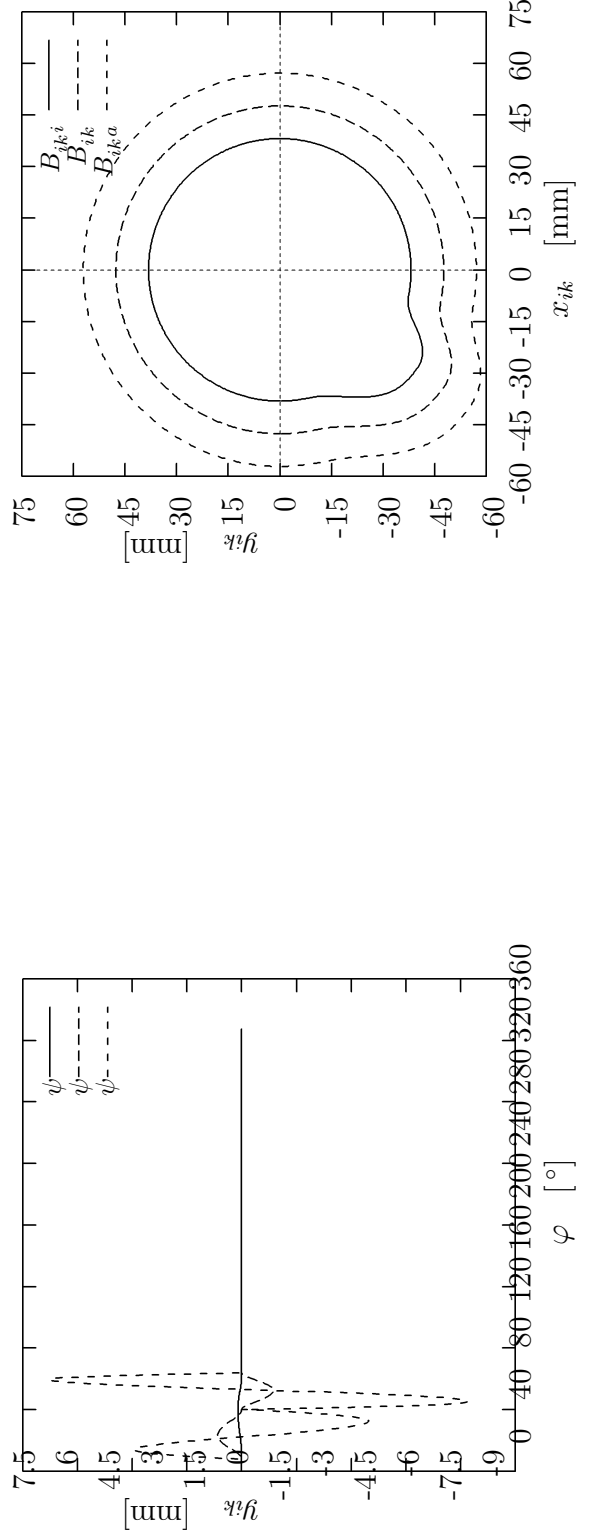
Winkel $\varphi_{H01}$ [°]	Winkel $\varphi_{H12}$ [°]	Winkel $\varphi_{H23}$ [°]	Winkel $\varphi_{H34}$ [°]
70	40	190	40
Winkel $\varphi_{H45}$ [°]	Schwinghebellänge $l_3$ [mm]	Koppellänge $l_4$ [mm]	Kurbellänge $l_{sk}$ [mm]
20	297.29	85	124.89
Gesamthub $s_H$ [mm]	Kurbelauslenkung $\alpha_{sk}$ [°]	Hebelversetzung $x_H$ [mm]	Hebelversetzung $y_H$ [mm]
5	1.97	0	0
Rollenradius $r_R$ [mm]	Lagerstelle $A_0 x_{A0}$ [mm]	Lagerstelle $A_0 y_{A0}$ [mm]	Winkel $\beta$ [°]
8	290	70	0
Grundkreisradius $r_G$ [mm]	Schrittweite $n$ [-]	Schubkurbel in Ri. Schieber	Hubrichtungen gleich
39.334	0.5	nein	nein
Quadrant von $B_a$	Bew. des Schwinghebels	Ri. von $s_0$	
erster	negativ	positiv	



## B.2 Schnittkurvenscheibe (opticurv version 1.0.3)

Tabelle 5: Eingabeparameter für die Schnittkurvenscheibe

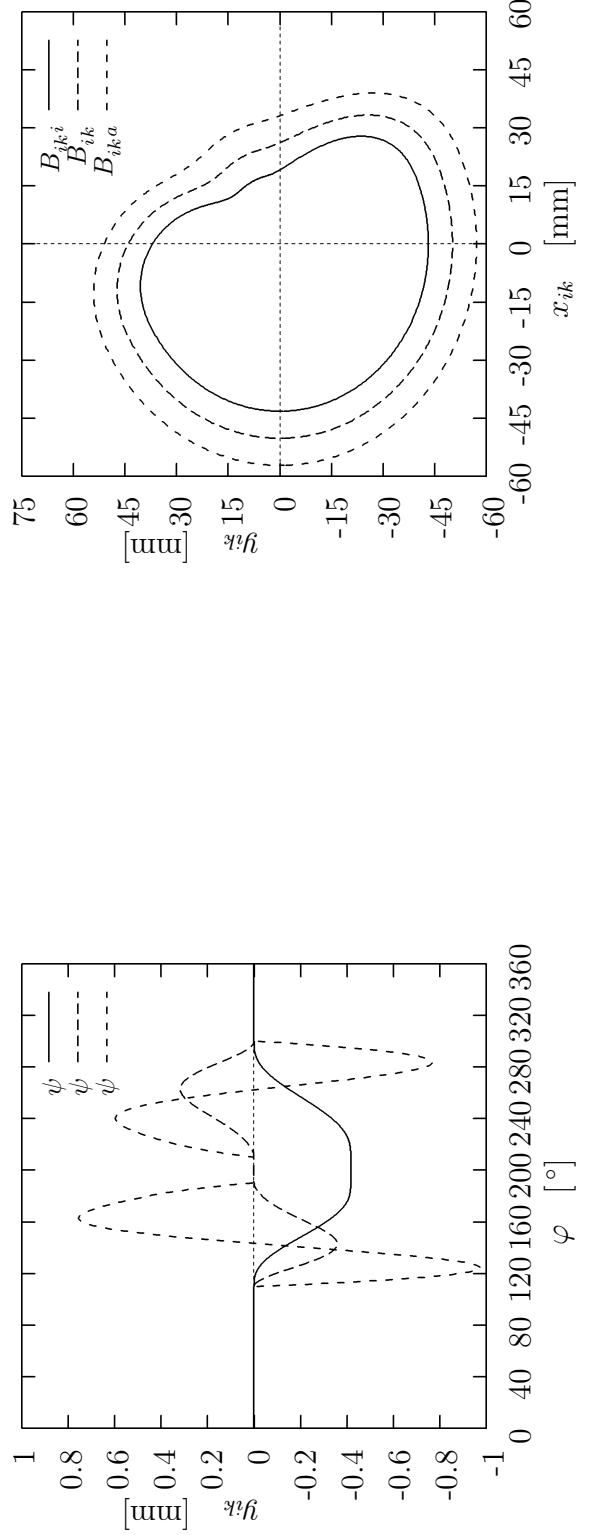
Winkel $\varphi_{H01}$ [°]	Winkel $\varphi_{H12}$ [°]	Winkel $\varphi_{H23}$ [°]	Winkel $\varphi_{H34}$ [°]
10	40	0	30
Winkel $\varphi_{H45}$ [°]	Schwinghebellänge $l_3$ [mm]	Koppellänge $l_4$ [mm]	Kurbellänge $l_{sk}$ [mm]
280	84.9	75.38	42
Gesamthub $s_H$ [mm]	Kurbelauslenkung $\alpha_{sk}$ [°]	Hebelversetzung $x_H$ [mm]	Hebelversetzung $y_H$ [mm]
5.5	6.3	0	0
Rollenradius $r_R$ [mm]	Lagerstelle $A_0 x_{A0}$ [mm]	Lagerstelle $A_0 y_{A0}$ [mm]	Winkel $\beta$ [°]
9.5	34.81	84.39	180
Grundkreisradius $r_G$ [mm]	Schrittweite $n$ [-]	Schubkurbel in Ri. Schieber	Hubrichtungen gleich
47.654	0.5	ja	ja
Quadrant von $B_a$	Bew. des Schwinghebels	Ri. von $s_0$	
erster	positiv	positiv	



### B.3 Setzhubkurvenscheibe (opticurv version 1.0.3)

Tabelle 6: Eingabeparameter für die Setzhubkurvenscheibe

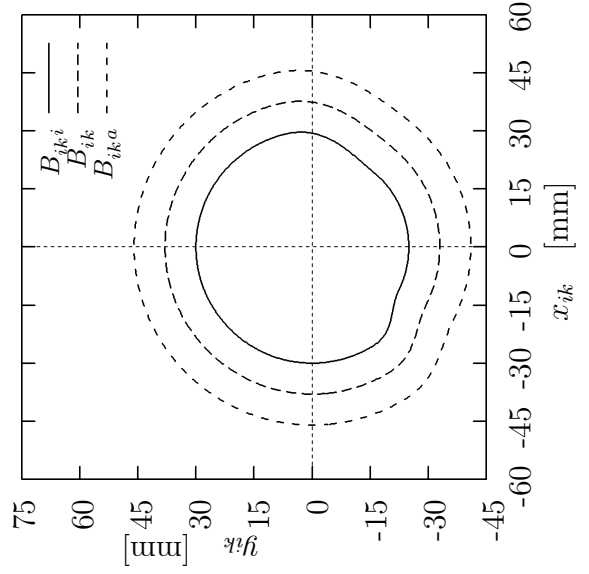
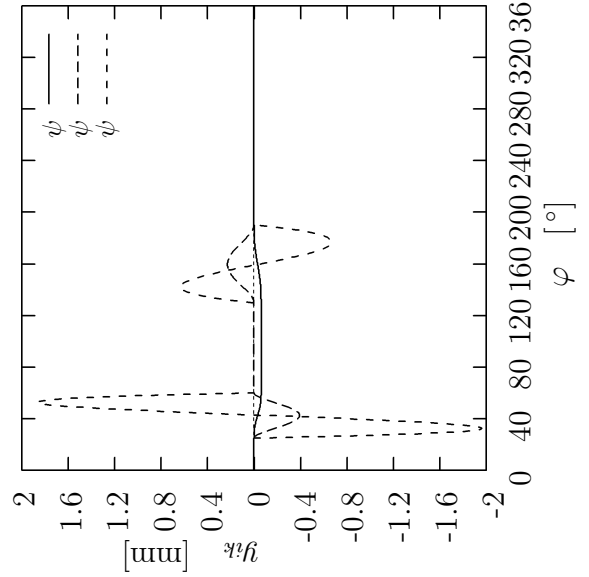
Winkel $\varphi_{H01}$ [°]	Winkel $\varphi_{H12}$ [°]	Winkel $\varphi_{H23}$ [°]	Winkel $\varphi_{H34}$ [°]
110	80	20	90
Winkel $\varphi_{H45}$ [°]	Schwinghebellänge $l_3$ [mm]	Koppellänge $l_4$ [mm]	Kurbellänge $l_{sk}$ [mm]
60	61.29	122.87	160.5
Gesamthub $s_H$ [mm]	Kurbelauslenkung $\alpha_{sk}$ [°]	Hebelversetzung $x_H$ [mm]	Hebelversetzung $y_H$ [mm]
58.5	34.68	0	0
Rollenradius $r_R$ [mm]	Lagerstelle $A_0 x_{A0}$ [mm]	Lagerstelle $A_0 y_{A0}$ [mm]	Winkel $\beta$ [°]
7	40.46	62	180
Grundkreisradius $r_G$ [mm]	Schrittweite $n$ [-]	Schubkurbel in Ri. Schieber	Hubrichtungen gleich
50.199	0.5	ja	nein
Quadrant von $B_a$	Bew. des Schwinghebels	Ri. von $s_0$	
zweiter	positiv	negativ	



## B.4 Vorschubkurvenscheibe (opticur version 1.0.3)

Tabelle 7: Eingabeparameter für die Vorschubkurvenscheibe

Winkel $\varphi_{H01}$ [°]	Winkel $\varphi_{H12}$ [°]	Winkel $\varphi_{H23}$ [°]	Winkel $\varphi_{H34}$ [°]
25	35	70	60
Winkel $\varphi_{H45}$ [°]	Schwinghebellänge $l_3$ [mm]	Koppellänge $l_4$ [mm]	Kurbellänge $l_{sk}$ [mm]
170	82.03	78	82.03
Gesamthub $s_H$ [mm]	Kurbelauslenkung $\alpha_{sk}$ [°]	Hebelversetzung $x_H$ [mm]	Hebelversetzung $y_H$ [mm]
5	5.64	0	0
Rollenradius $r_R$ [mm]	Lagerstelle $A_0 x_{A0}$ [mm]	Lagerstelle $A_0 y_{A0}$ [mm]	Winkel $\beta$ [°]
8	29.76	85.37	0
Grundkreisradius $r_G$ [mm]	Schrittweite $n$ [-]	Schubkurbel in Ri. Schieber	Hubrichtungen gleich
38.006	0.5	nein	nein
Quadrant von $B_a$	Bew. des Schwinghebels	Ri. von $s_0$	
erster	positiv	positiv	



# Literatur

- [1] Căărăbaș, Iosif; Mesaroș-Anghel, Voicu; Lovasz, Erwin-Christian: *Proiectarea mecanismelor*. 1.Aufl., Mirton, 2000
- [2] DIN 66261 *Sinnbilder für Struktogramme nach Nassi-Shneiderman* 24
- [3] Herold, Helmut: *Das Qt Buch; Portable GUI-Programmierung unter Linux/UNIX/Windows*. 1.Aufl., SuSE, 2001
- [4] Luck, Kurt; Modler, Karl-Heinz: *Getriebetechnik; Analyse, Synthese, Optimierung*. 2.Aufl., Springer, 1995 8, 11, 14, 15, 17, 18, 20, 28
- [5] Obst, Peter; Heydt, Wolfgang: *Konstruktion und Fertigung von Kurvenmechanismen*. 1.Aufl., Verlag Technik, 1964 10
- [6] VDI 2142 Blatt 1 *Auslegung ebener Kurvengetriebe; Grundlagen, Profilberechnung und Konstruktion*
- [7] VDI 2143 Blatt 1 *Bewegungsgesetze für Kurvengetriebe; Theoretische Grundlagen* 2, 4, 5, 6, 7
- [8] Volmer, Johannes: *Getriebetechnik; Grundlagen*. 2.Aufl., Verlag Technik, 1995
- [9] Volmer, Johannes: *Getriebetechnik; Kurvengetriebe*. 2.Aufl., Verlag Technik, 1989 32
- [10] Weber, Heinrich: *Software-Technik*. FHT Esslingen, WS 1994/95, Vorlesungsscript 22
- [11] Willms, Gerhard: *C++; Das Grundlagenbuch*. 2.Aufl., DATA BECKER, 2001

## Stichwortverzeichnis