

Diplomarbeit

**Realisierung komplexer
nichtlinearer Bewegungsvorgänge
durch kombinierte Mechanismen**

eingereicht 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, 2003

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen

Aufgabenstellung für die Diplomarbeit

im Studiengang: Maschinenbau

in der Studienrichtung: Kraftfahrzeug- und Schienenfahrzeugtechnik

Name des Studenten: Lutz Wirsig

Thema: Realisierung komplexer nichtlinearer Bewegungsvorgänge durch kombinierte Mechanismen

Zielsetzung: Durch die Kombination von Koppelgetrieben mit einer Kurvenscheibe als Antriebsglied lassen sich komplizierte nichtlineare Bewegungsvorgänge sehr günstig realisieren. Neben der Auslegung der kinematischen Struktur spielt die Zwanglaufsicherung an der Kurvenscheibe für die Drehzahlerhöhung eine entscheidende Rolle.

Für komplexe nichtlineare Bewegungsvorgänge in Montageautomaten sind die notwendigen Algorithmen häufig vorkommender Übertragungsfunktionen aufzustellen.

Die Zwanglaufsicherung soll durch Doppelkurven erfolgen.

Als nachnutzbares Ergebnis ist in Abstimmung mit dem Industriepartner eine Anwendersoftware zu entwickeln.

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

Ausgehändigt am: 04.08.2003

Einzureichen am: 01.12.2003

Die von der Fakultät erlassenen Richtlinien zur Anfertigung der Diplomarbeit sowie die Diplomprüfungsordnung sind zu beachten.

Prof. Dr.-Ing. habil. W. Fischer
Leiter der Studienrichtung

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

Inhaltsverzeichnis

Abbildungsverzeichnis	iv
Tabellenverzeichnis	vi
Formelzeichen	vii
1 Einleitung	1
2 Grundbegriffe und Bezeichnungen	3
2.1 Bewegungsgesetze	3
2.2 Bewegungsplan, Bewegungsdiagramm und Bewegungsaufgaben	5
2.3 Normierte Bewegungsgesetze für Rast in Rast	8
2.3.1 Symmetrische normierte Bewegungsgesetze für Rast in Rast	8
2.3.1.1 Beispiel: 3-4-5 Polynom	10
2.3.2 Unsymmetrische normierte Bewegungsgesetze für Rast in Rast . . .	11
2.3.2.1 Beispiel: 3-4-5 Polynom	12
2.4 Doppelkurvenscheiben	15
3 Ermittlung der Abmessungen	17
3.1 Ermittlung der A_0 -Bereiche	19
3.2 Bewegungsverlauf	21
3.3 Übertragungsfunktionen	21
3.4 Rollenmittelpunktskurve und Kurvenprofil	24
3.4.1 Kurve	24
3.4.2 Gegenkurve	28
3.5 Grundkreiswinkel und Grundkreisradius	29
3.6 Übertragungswinkel	30
3.7 Krümmungsradius	31
4 Dynamisches Verhalten von Kurvengetrieben	33
4.1 Modellbildung	34
4.1.1 Lösung der Differentialgleichung	34

4.1.2	Interpretation des Bewegungsgesetzes	36
4.2	Beispiele	39
4.2.1	Querhub	39
4.2.2	Schnittbewegung	42
4.2.3	Setzhub	45
4.2.4	Vorschub	48
5	Anwendungsbereiche der Modelle	51
6	Die Anwendung	52
6.1	Installation	52
6.1.1	Windows	52
6.1.2	Linux	52
6.1.3	Andere Betriebssysteme	53
6.2	Menüeinträge	53
6.2.1	Datei	53
6.2.2	Ausgabe	58
6.2.3	Hilfe	61
6.3	Fragen und Antworten	61
7	Zusammenfassung	62
	Literatur	63
	Eidesstattliche Erklärung	64
	Stichwortverzeichnis	64
	Anhang	67
A	Beispiele für Doppelkurvenscheiben	68
A.1	Querhub	68
A.2	Schnittkurve	70
A.3	Setzhub	72
A.4	Vorschub	74

B	Mathematica-Dateien	76
B.1	Querhub_dyn.nb	76
B.2	Schnitt_dyn.nb	80
B.3	Setzhub_dyn.nb	84
B.4	Vorschub_dyn.nb	88
C	C++ Quellcode	92
C.1	functionplot.h	92
C.2	opticurv273.cpp	127

Abbildungsverzeichnis

1	Schema eines Übertragungsgetriebes (hier:Kurvenkoppelgetriebe)	1
2	An- und Abtriebsbewegung a) Stößel b) Schwinge	3
3	Bewegungsplan	5
4	Bewegungsdiagramm	6
5	Zusammenhang zwischen realem und normiertem Bewegungsgesetz	8
6	Normiertes symmetrisches Bewegungsgesetz	9
7	Symmetrisches 3-4-5 Polynom	11
8	Normiertes unsymmetrisches Bewegungsgesetz	12
9	Unsymmetrisches 3-4-5 Polynom	14
10	Doppelkurvenscheibe mit a) einer Außen- und einer Innenkurvenflanke b) zwei Außenkurvenflanken	15
11	Vergleich von a) Nutkurvenscheibe und b) Doppelkurvenscheibe	16
12	Übertragungswinkel μ	17
13	Kurvengetriebe mit Schwinghebel a) F-Kurvengetr. b) P-Kurvengetr.	18
14	Kurvengetriebe mit Schieber a) F-Kurvengetr. b) P-Kurvengetr.	18
15	Ermittlung der A_0 -Bereiche nach <i>Flocke</i> für Kurvengetriebe mit a) Schwinghebel b) Schieber	20
16	Kurvenkoppelgetriebe mit Doppelschwinghebel	21
17	Grundfigur zur Berechnung der Rollenmittelpunktskurve bei einem Kurvengetriebe mit Schwinghebel	25
18	Grundfigur zur Berechnung der inneren und äußeren Arbeitskurve	26
19	Grundfigur zur Berechnung der Rollenmittelpunktskurve der Gegenkurve	28
20	Übertragungswinkel μ am Kurvengetriebe	30
21	Spitzenbildung und Unterschnitt bei zu kleinem Krümmungsradius r_K	32
22	Bewegungsdiagramm der allgemeinen Rast-in-Rast-Bewegung	33
23	Einmassenmodell eines Kurvengetriebes	34
24	Relative Genauigkeiten für Rast-in-Rast-Bewegung	38
25	Bewegungsverlauf $s(\varphi)$ beim Querhub	40
26	Vergleich der Querhubverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungsgesetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom	41
27	schwingungsarme Querhubkurvenscheibe	41

28	Bewegungsverlauf $s(\varphi)$ der Schnittbewegung	43
29	Vergleich der Schnittbewegungsverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungsgesetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom	44
30	schwingungsarme Schnittkurvenscheibe	44
31	Bewegungsverlauf $s(\varphi)$ der Setzhubbewegung	46
32	Vergleich der Setzhubbewegungsverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungsgesetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom	47
33	schwingungsarme Setzhubkurvenscheibe	47
34	Bewegungsverlauf $s(\varphi)$ der Vorschubbewegung	49
35	Vergleich der Vorschubbewegungsverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungsgesetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom	50
36	schwingungsarme Vorschubkurvenscheibe	50
37	Dialogfenster	53
38	Definition des Bewegungsverlaufs	54
39	Bewegungsplan	54
40	Kurvenkoppelgetriebe mit Doppelschwinghebel	55
41	Vorzeichenbeispiele a) beide negativ b) beide positiv c) eines positiv und eines negativ	56
42	Funktionsplotter	58
43	Verläufe ψ , ψ' und ψ'' beim Querhub	69
44	Doppelkurvenscheibe für die Querhubbewegung	69
45	Verläufe ψ , ψ' und ψ'' bei der Schnittkurvenscheibe	71
46	Doppelkurvenscheibe für die Schnittbewegung	71
47	Verläufe ψ , ψ' und ψ'' beim Setzhub	73
48	Doppelkurvenscheibe für die Setzhubbewegung	73
49	Verläufe ψ , ψ' und ψ'' beim Vorschub	75
50	Doppelkurvenscheibe für die Vorschubbewegung	75

Tabellenverzeichnis

1	Bewegungsaufgaben	6
2	Mögliche Kombinationen von Bewegungsaufgaben	7
3	Vorgaben für die Querhubbewegung	39
4	Vorgaben für die Schnittbewegung	42
5	Vorgaben für die Setzhubbewegung	45
6	Vorgaben für die Vorschubbewegung	48

Formelzeichen

Zeichen	Bezeichnung	Einheit
$a = \ddot{s}$	Beschleunigung des Abtriebsgliedes (gerade geführt)	[mm/s ²]
A_0	Lagerstelle der Kurvenscheibe	[–]
b	Dämpfungskonstante	[Ns/m]
B_0	Lagerstelle des Rollenhebels	[–]
B_{ik}	Rollenmittelpunkt der Kurve im Bewegungsabschnitt ik	[–]
B_{ik}^*	Rollenmittelpunkt der Gegenkurve im Bewegungsabschnitt ik	[–]
B_a	Rollenmittelpunkt der Kurve zu Beginn der Bewegung	[–]
B_a^*	Rollenmittelpunkt der Gegenkurve zu Beginn der Bewegung	[–]
c	Steifigkeit	[N/m]
C_a	Beschleunigungskennwert	[–]
C_j	Ruckkennwert	[–]
C_{Mstat}	statischer Momentenkennwert	[–]
C_{Mdyn}	dynamischer Momentenkennwert	[–]
C_v	Geschwindigkeitskennwert	[–]
S_{ik}	Gelenkpunkt zwischen Koppel und Schieber im Abschnitt ik	[–]
e_s	Exzentrizität	[mm]
y_{s0}	Exzentrizität der Koppel	[mm]
f_{ik}	normierter Weg	[–]
ik	Nummerierung der Bewegungsabschnitte	[–]
k	Ordnungszahl der Harmonischen	[–]
k_{B32}	Rollenmittelpunktskurve der Kurve	[–]
k_{B32}^*	Rollenmittelpunktskurve der Gegenkurve	[–]
k_G	Grundkreis der Kurve	[–]

k_G^*	Grundkreis der Gegenkurve	[—]
l_1	Gestelllänge (Glieder 1)	[mm]
l_3	Rollenhebellänge (Glieder 3) Kurve	[mm]
l_3^*	Rollenhebellänge (Glieder 3) Gegenkurve	[mm]
l_4	Koppellänge (Glieder 4)	[mm]
l_{sk}	Schubkurbellänge (Glieder 3)	[mm]
m	Masse	[kg]
n	Schrittweite des Drehwinkels	[°]
n_H	Anzahl der Hauptharmonischen	[—]
n_r	Normale an die Relativbahn	[—]
r_G	Grundkreisradius der Kurve	[mm]
r_G^*	Grundkreisradius der Gegenkurve	[mm]
r_K	Krümmungsradius	[mm]
r_R	Laufrollenradius	[mm]
s	Abtriebsweg	[mm]
x_{S0}	Hub in der Nullstellung	[mm]
s_G	Grundhub	[mm]
s_{Hik}	Gesamtweg des gerade geführten Abtriebsgliedes im Abschnitt ik	[mm]
t	Zeit	[s]
t_a	Tangente an die Absolutbahn des Abtriebsgliedes	[—]
t_r	Tangente an die Relativbahn des Übertragungsgliedes gegenüber dem Antriebsglied	[—]
$v = \dot{s}$	Geschwindigkeit des Abtriebsgliedes (gerade geführt)	[mm/s]
z_{ik}	normierter Drehwinkel	[—]
α	Lagewinkel der Gestelllänge	[°]
α_d	Rastbreite der längeren (unteren) Rast	[°]

α_R	Winkel zwischen den Rollenhebeln (l_3 und l_3^*)	[°]
α_{sk}	Anfangsauslenkung der Koppel	[°]
β	Winkel zwischen Rollenhebel und Schubkurbel	[°]
β_d	Rastbreite der kürzeren (oberen) Rast	[°]
γ_d	Abstand der Rastmitte der oberen Rast von der Rastmitte der unteren Rast	[°]
δ_d	Rastbreite des Übergangs von längerer zu kürzerer Rast	[°]
η	Abstimmungsverhältnis	[—]
ϑ	Lagewinkel von Glied 4 beim Kurvenkoppelgetriebe	[°]
ϑ_L	<i>Lehrsches</i> Dämpfungsmaß	[—]
λ	Wendepunktparameter	[—]
μ	Übertragungswinkel	[°]
ν	Verhältnis der Rastbreiten ($\nu = \beta_d/\alpha_d$)	[—]
φ	Antriebswinkel	[°]
φ_{Hik}	Gesamtdrehwinkel der Kurvenscheibe im Abschnitt ik	[°]
χ_d	Winkel zu Beginn der kürzeren Rast	[°]
ψ	Abtriebswinkel	[°]
ψ_0	Abtriebswinkel in Nullstellung	[°]
ψ_G	Grundwinkel der Kurve	[°]
ψ_G^*	Grundwinkel der Gegenkurve	[°]
ψ_{Hik}	Gesamtdrehwinkel des schwingend geführten Abtriebsgliedes im Abschnitt ik	[°]
ω	Kreisfrequenz	[1/s]
ω_0	Eigenkreisfrequenz	[1/s]
Ω	Erregerkreisfrequenz	[1/s]

1 Einleitung

Der Steuerung von Arbeitsmaschinen durch rein mechanische Mittel kommt auch heute noch große Bedeutung zu. Dies gilt insbesondere für Verarbeitungsmaschinen wie beispielsweise Textil-, Papier-, Druckerei- und Verpackungsmaschinen. Notwendige Arbeitsbewegungen werden oft durch Übertragungsgetriebe wie Kurvengetriebe mit nachgeschalteten Koppelgetriebe realisiert (Abb. 1)

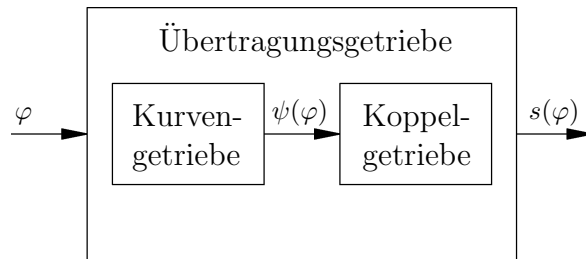


Abbildung 1: Schema eines Übertragungsgetriebes (hier: Kurvenkoppelgetriebe)

Durch den Einsatz solcher Getriebe können Bewegungsaufgaben verwirklicht werden, die mit Kurvengetrieben oder Koppelgetrieben sonst nicht lösbar sind. Auf diese Weise läßt sich

- die Kniehebelwirkung an der Schubkurbel nutzen,
- der Abtriebswinkel ψ vergrößern,
- die Antriebswinkelgeschwindigkeit der Kurvenscheibe beeinflussen,
- die Kraft- und Bewegungsübertragung zwischen An- und Abtriebsglied verbessern.

Der Entwurf dieser Mechanismen erfordert umfangreiche Kenntnisse der Getriebelehre und ist ohne geeignete Hilfsmittel sehr aufwendig. Ziel dieser Arbeit ist deshalb, ein praxisorientiertes Programm in Abstimmung mit den Industriepartnern *XENON Automatisierungstechnik GmbH* und *IbH Hagedorn* zu entwickeln, um den Zeit- und Arbeitsaufwand beim Getriebeentwurf reduzieren zu können.

Die Arbeit beschreibt zunächst die wichtigsten Grundlagen und Zusammenhänge. Es werden die Begriffe Bewegungsgesetz, Bewegungsplan, Bewegungsdiagramm, Bewegungsaufgabe erklärt und die Ermittlung von normierten Grundgesetzen vorgestellt. Es folgt ein kurzer Überblick über Ausführungen und Eigenschaften von Doppelkurvenscheiben.

An die Grundlagen schließt sich das Kapitel „Ermittlung der Abmessungen“ an. Nach der Definition der Größen wird auf die Ermittlung von A_0 -Bereichen eingegangen. Anschließend werden die Algorithmen zur Berechnung von Bewegungsverlauf, Übertragungsfunktionen, Rollenmittelpunktskurve und Kurvenprofil hergeleitet. Ebenfalls berechnet werden der

Grundkreiswinkel, der Grundkreisradius, der Übertragungswinkel und der Krümmungsradius. Letztere können mit zur Beurteilung des Getriebes herangezogen werden.

Das Kapitel „Dynamisches Verhalten von Kurvengetrieben“ beschreibt die Auslegung von Kurvengetrieben mit guten dynamischen Eigenschaften. Die Verwendung des Modells wird anhand von Beispielen demonstriert.

Gesichtspunkte zum Vergleich von vollkommen starren Modellen und Modellen, die die Elastizitäten der Gelenke berücksichtigen, enthält das Kapitel „Anwendungsbereiche der Modelle“.

Im darauf folgenden Kapitel ist die Anwendungsbeschreibung des im Rahmen dieser Arbeit entwickelten Programms „*opticurv*“ zu finden. Darin werden im Wesentlichen die Installation, die Parametereingabe und die Ergebnisausgabe behandelt.

In der Zusammenfassung werden die Forschungsergebnisse kurz dargestellt und Vorschläge zum weiteren Vorgehen aufgezeigt.

Der Anhang beinhaltet Beispiele für Kurvenscheibenberechnungen und den Quellcode von *opticurv*.

2 Grundbegriffe und Bezeichnungen

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 $s(\varphi)$ und der Abtriebswinkel $\psi(\varphi)$ kennzeichnen die jeweilige Bewegung der Getriebeglieder (Abb. 2).

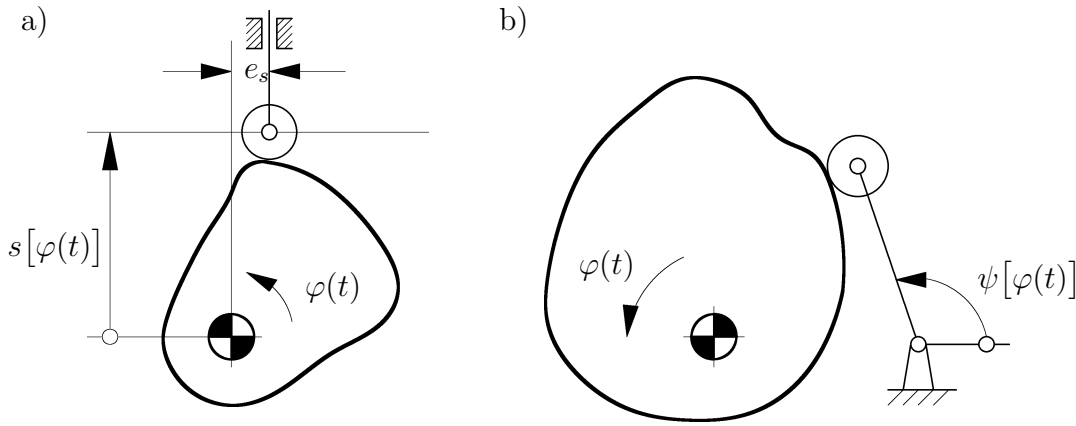


Abbildung 2: An- und Abtriebsbewegung a) Stößel b) Schwinge [14]

Die Geschwindigkeiten und Beschleunigungen der Abtriebsglieder ergeben sich aus den Ableitungen ihrer Bewegungen. Ableitungen nach dem Drehwinkel φ 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 ¹ 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

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

$$^1 f(x) = u[v(x)] \rightsquigarrow f'(x) = u'[v(x)] \cdot v'(x)$$

Bei der Ableitung des Abtriebswinkels ψ 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 Produktregel². 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 ψ zu ersetzen.

² $y = u \cdot v \rightsquigarrow y' = u'v + v'u$

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. 39).

Der Gesamtdrehwinkel φ_H eines Abschnittes bekommt die Indizes ik :

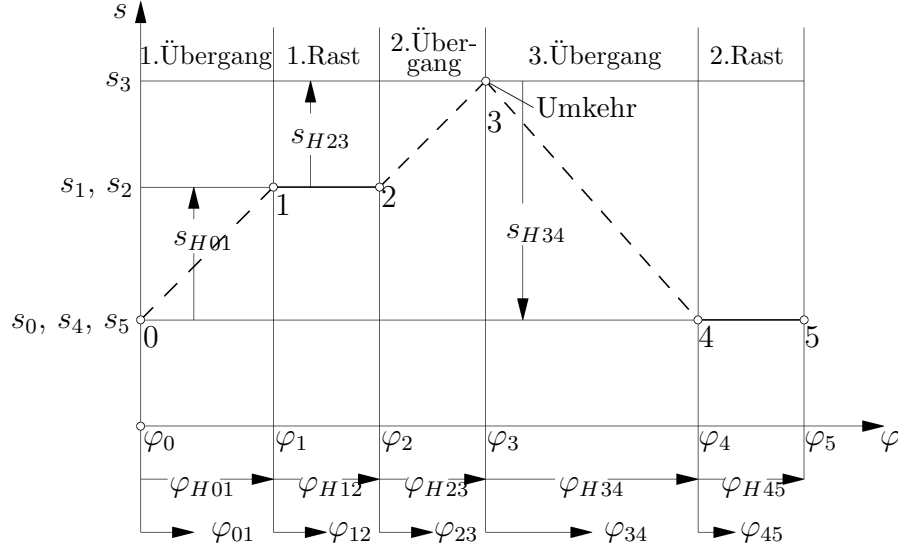


Abbildung 3: Bewegungsplan [14]

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

mit

φ_i Drehwinkel des Antriebsgliedes zu Beginn des Bewegungsabschnittes

φ_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. 4).

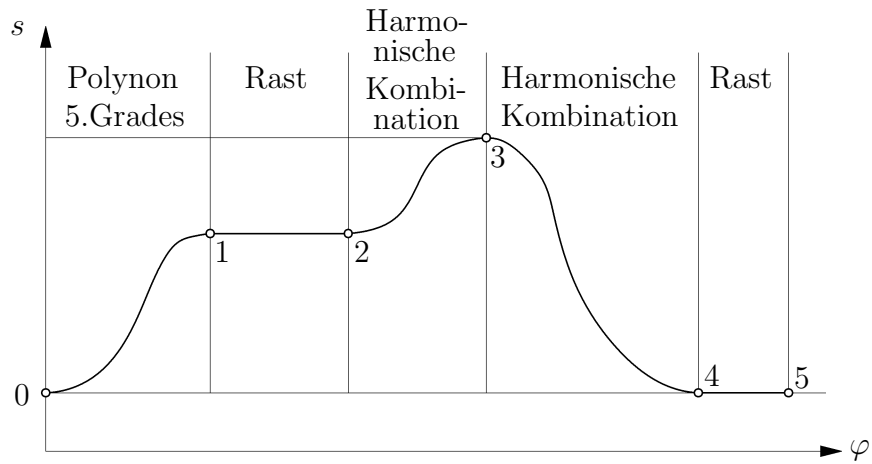


Abbildung 4: Bewegungsdiagramm [14]

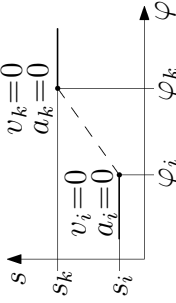
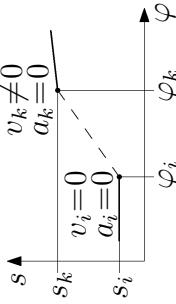
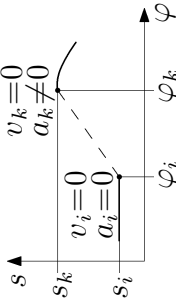
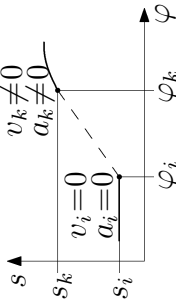
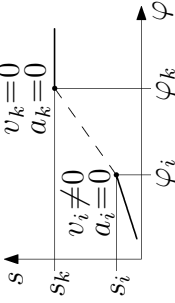
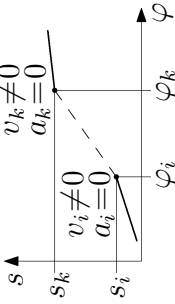
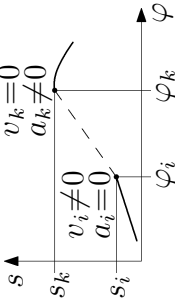
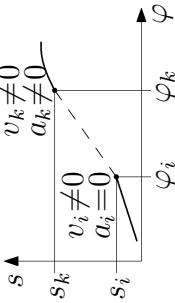
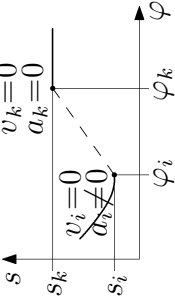
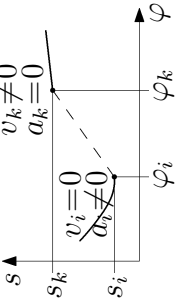
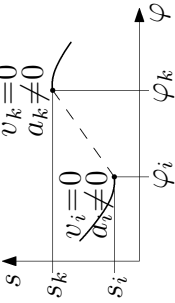
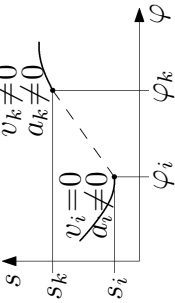
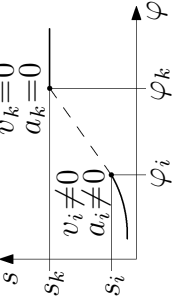
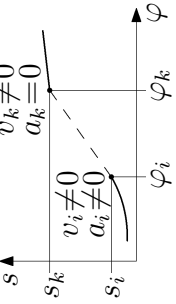
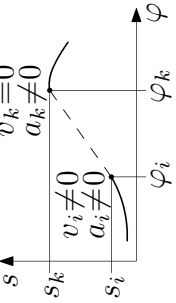
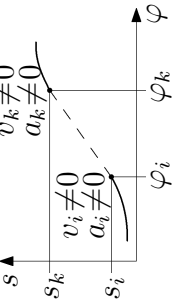
Die geforderte Abtriebsbewegung setzt sich aus verschiedenen Bewegungsaufgaben zusammen. Abhängig von Geschwindigkeit und Beschleunigung ergeben sich vier Typen von Bewegungsaufgaben (Tab. 1).

Tabelle 1: Bewegungsaufgaben [14]

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 [14]

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

2.3 Normierte Bewegungsgesetze für Rast in Rast

2.3.1 Symmetrische normierte Bewegungsgesetze für Rast in Rast

Normierte Größen sind meist auf ihren Nennwert bezogen. Sie nehmen Werte zwischen 0 und 1 an. Damit sind sie nicht nur besser vergleichbar, sondern auch besser weiterverwendbar, da sie dimensionslos sind.

Der normierte Drehwinkel z_{ik} ergibt sich, wenn man den Drehwinkel φ_{ik} auf den Gesamtdrehwinkel φ_{Hik} bezieht.

$$z_{ik} = \frac{\varphi_{ik}}{\varphi_{Hik}} \quad (2.9)$$

Beim normierten Weg f_{ik} ist der Weg s_{ik} auf den Gesamtweg s_{Hik} bezogen.

$$f_{ik} = \frac{s_{ik}}{s_{Hik}} \quad (2.10)$$

In Bild 5 ist der Zusammenhang zwischen realem und normiertem Bewegungsgesetz dargestellt.

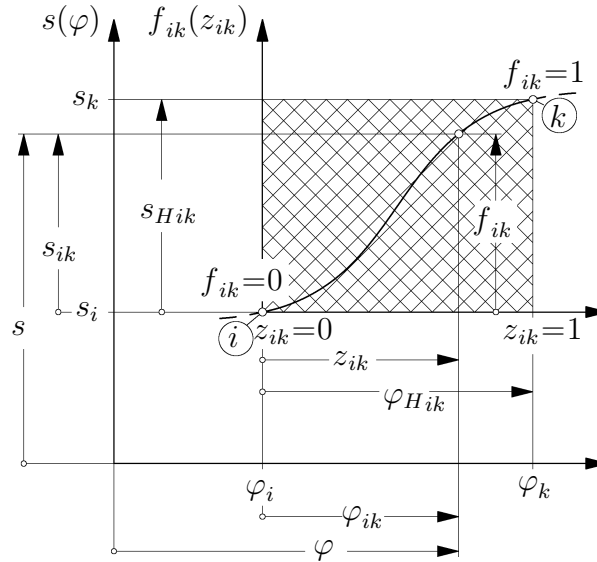


Abbildung 5: Zusammenhang zwischen realem und normiertem Bewegungsgesetz [14]

Das normierte Bewegungsgesetz ist der normierte Weg f_{ik} als Funktion des normierten Drehwinkels z_{ik} .

$$f_{ik} = f_{ik}(z_{ik}) \quad (2.11)$$

Es muß folgenden Randbedingungen genügen:

$$\left. \begin{array}{l} \varphi_{ik} = 0 \quad : \quad z_{ik} = 0, \quad f_{ik}(0) = 0 \\ \varphi_{ik} = \varphi_{Hik} \quad : \quad z_{ik} = 1, \quad f_{ik}(1) = 1 \end{array} \right\} \quad (2.12)$$

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. 6). 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).

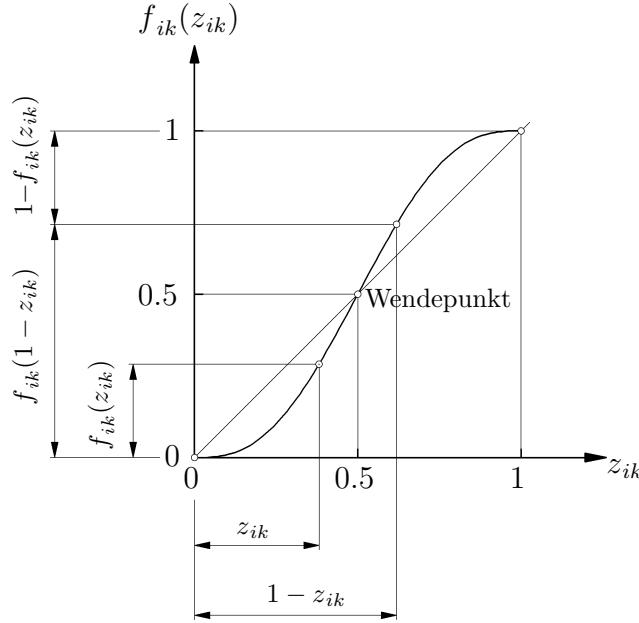


Abbildung 6: Normiertes symmetrisches Bewegungsgesetz [9]

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:

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

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

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

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 ($= C_v \cdot C_a$)

2.3.1.1 Beispiel: 3-4-5 Polynom

Das 3-4-5 Polynom ist ein Bewegungsgesetz, das gewährleistet, dass kein Ruck³ 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 7 dargestellt.

³Beschleunigungssprung

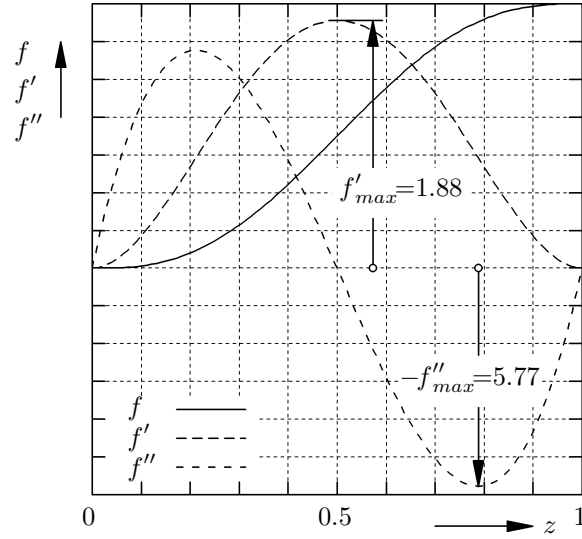


Abbildung 7: Symmetrisches 3-4-5 Polynom [11]

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. 8).

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

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

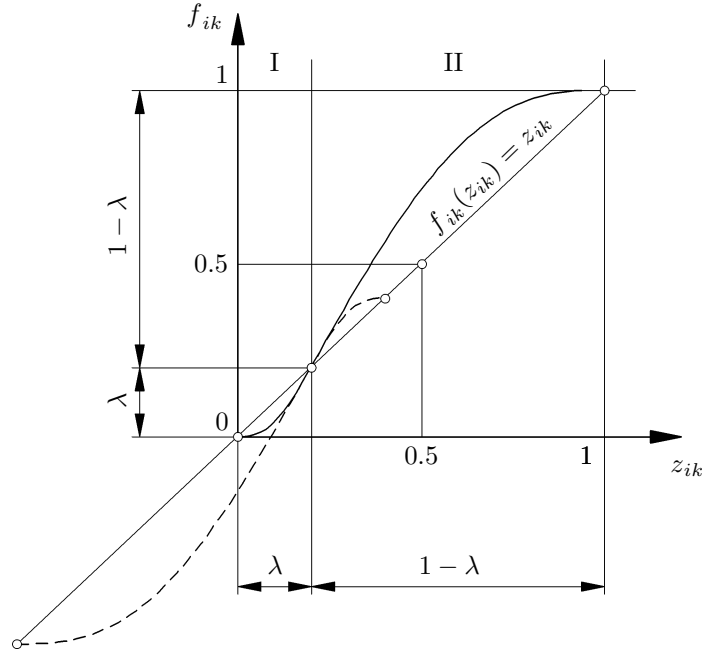


Abbildung 8: Normiertes unsymmetrisches Bewegungsgesetz [9]

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

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

2.3.2.1 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 λ , 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 λ , 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 9 dargestellt.

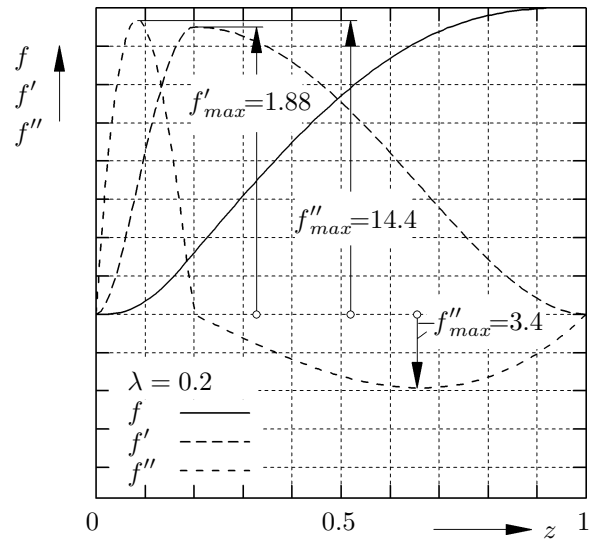


Abbildung 9: Unsymmetrisches 3-4-5 Polynom

2.4 Doppelkurvenscheiben

Doppelkurven sichern den Zwangslauf des Abtriebsgliedes durch den Formschluss von zwei fest miteinander verbundenen Kurvenscheiben (Kurve und Gegenkurve). Gebräuchlich sind Kurvenscheibenpaarungen mit Außen- und Innenkontur oder mit zwei Außenkonturen (Abb. 10).

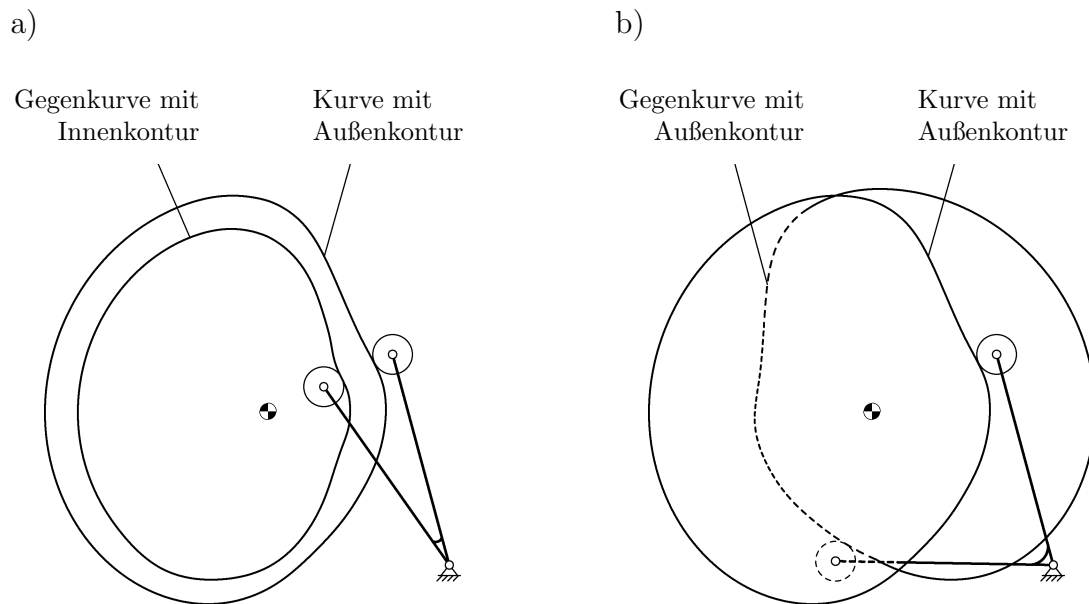


Abbildung 10: Doppelkurvenscheibe mit a) einer Außen- und einer Innenkurvenflanke b) zwei Außenkurvenflanken

Doppelkurven haben gegenüber Nutkurvenscheiben folgende Vorteile:

- geringer Platzbedarf
- genaues Abrollen auf der Kurvenkontur
- weniger Verschleiß

Sie benötigen weniger Platz, weil die Wanddicke zwischen Nutaußenkante und Scheibenaußenkante entfällt. Zum Vergleich sind in Bild 11 eine Nutkurvenscheibe und eine Doppelkurvenscheibe dargestellt, welche den gleichen Bewegungsverlauf erzeugen.

Das bessere Abrollen resultiert daraus, dass die Rollen jeweils nur an einer Flanke anliegen. Dies ist bei Nutkurvenscheiben nicht der Fall. Dort wechselt die Rolle aufgrund des Richtungswechsels der Massenkräfte die Flanke. Dadurch weicht der Rollenmittelpunkt von der Rollenmittelpunktsbahn ab. Wie groß der Fehler ist, hängt vom vorhandenen Spiel ab. Es ist jedoch ein Mindestmaß an Spiel vorzusehen, da es bei zu geringen Spiel zum Gleiten an der Gegenkurve und dem damit verbundenen Gleitverschleiß kommt.

Kommt es bei Doppelkurvenscheiben dennoch durch

- ungenaues Ausrichten der Scheiben zueinander
- Fertigungstoleranzen der Kurvenkontur
- Fehler der Längen l_1 , l_3 und l_3^* (vgl. Abb. 40)

zu Spiel im Kurvengelenk, sind die Kurvenscheiben zueinander einzuarbeiten (z.B. Einschleifen).

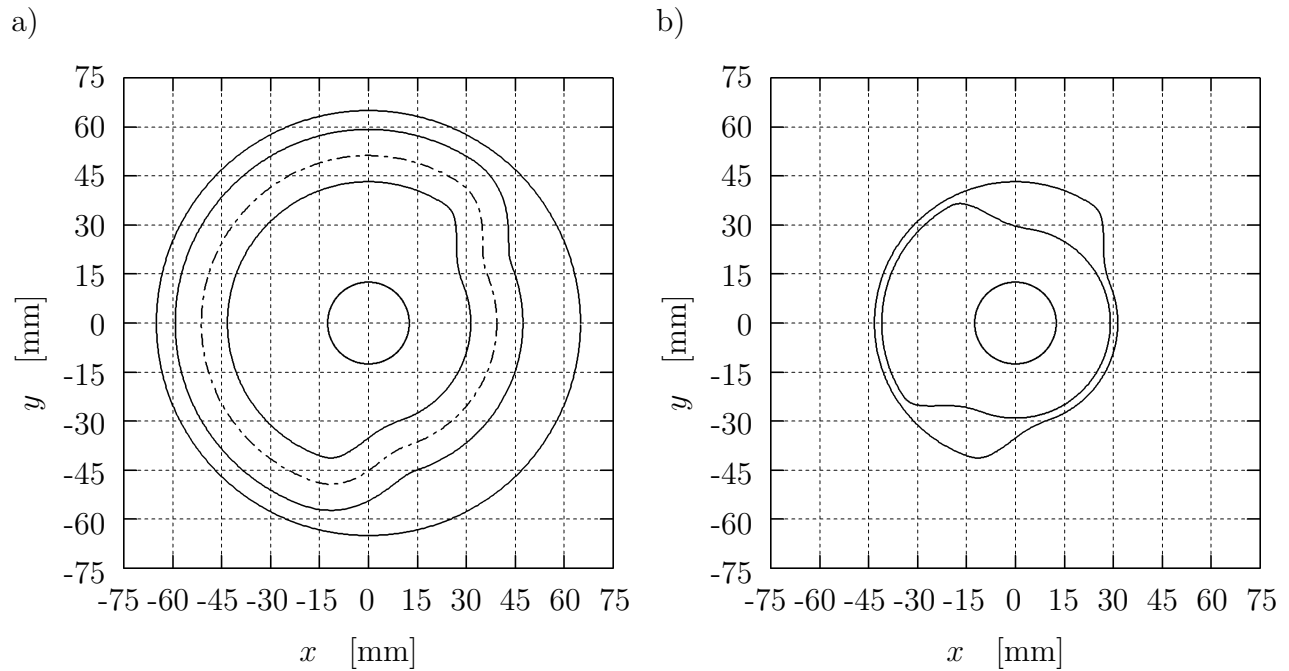


Abbildung 11: Vergleich von a) Nutkurvenscheibe und b) Doppelkurvenscheibe

3 Ermittlung der Abmessungen

Bei der Wahl der Abmessungen sollte nach Möglichkeit ein günstiger Übertragungswinkel μ 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. 12).

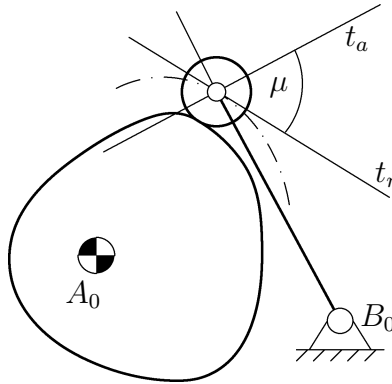


Abbildung 12: Übertragungswinkel μ [9]

Ein günstiger Übertragungswinkel liegt vor, wenn er bei langsamlaufenden Getrieben ($n < 30\text{min}^{-1}$) *nicht kleiner als* 45° und bei schnelllaufenden Getrieben ($n > 30\text{min}^{-1}$) *nicht kleiner als* 60° wird. Ausnahme bilden nur Kurvengetriebe mit Rollenstößel. Bei ihnen sollte der Übertragungswinkel μ auch bei Drehzahlen von $n < 30\text{min}^{-1}$ nicht kleiner als 60° werden. Bei geschmierten Kontaktflächen dürfen diese Werte ggfs. unterschritten werden. Neben dem Übertragungswinkel μ sind noch folgende Abmessungen festzulegen.

- Kurvengetriebe mit Schwinghebel (Abb. 13)

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

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

Grundkreisradius r_G

Grundkreis k_G

Grundwinkel ψ_G

- Kurvengetriebe mit Schieber (Abb. 14)

Exzentrizität e_s

Grundhub s_G

3.1 Ermittlung der A_0 -Bereiche

Die A_0 -Bereiche können näherungsweise mit dem Verfahren nach *Flocke* bestimmt werden. Dazu sind folgende Schritte notwendig (Abb. 15):

- 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:

$$\langle v_{B\max} \rangle_{P/N} = \pm \frac{\langle \psi_H \rangle}{\varphi_{HP/N}} \cdot C_{vP/N} \quad (3.1)$$

a) Kurvengetriebe mit Schieber:

$$\langle v_{B\max} \rangle_{P/N} = \pm \frac{\langle s_H \rangle}{\varphi_{HP/N}} \cdot C_{vP/N} \quad (3.2)$$

Darin sind

$\langle v_{B\max} \rangle_{P/N}$	darstellende Größe ⁷ der maximalen Geschwindigkeit des Punktes B für Gleich- bzw. Gegenlauf
ψ_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 $f_{ik}(z_{ik})$

Gleichlauf bedeutet, dass sich An- und Abtriebsglied im gleichen Drehsinn bewegen. Bei Gegenlauf bewegen sich An- und Abtriebsglied im entgegengesetzten Drehsinn.

- Längen $\langle v_{B\max} \rangle_P$ und $\langle v_{B\max} \rangle_N$ im Rollenmittelpunkt um 90° gedreht⁸ 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 μ_{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

⁷darstellende Größe = Maßstab · wirkliche Größe

⁸Drehsinn des Antriebswinkels φ

3.2 Bewegungsverlauf

Als Bewegungsgesetz für den Hubverlauf $s(\varphi)$ werden das *3-4-5 Polynom* und die *Bestehornsinoide* verwendet. Das *3-4-5 Polynom* lautet:

$$f(z) = 10z^3 - 15z^4 + 6z^5 \quad (3.3)$$

Mit den Gleichungen (2.9) und (2.10) von Seite 8 wird daraus

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

$$s_{ik} = \frac{s_{Hik}}{\varphi_{Hik}^3} \left(10 \varphi_{ik}^3 - 15 \frac{\varphi_{ik}^4}{\varphi_{Hik}} + 6 \frac{\varphi_{ik}^5}{\varphi_{Hik}^2} \right) \quad (3.4)$$

Die *Bestehornsinoide* lautet:

$$f(z) = z - \frac{1}{2\pi} \sin(2\pi z) \quad (3.5)$$

Nach Einsetzen der Gleichungen (2.9) und (2.10) in (3.5) erhält man

$$s_{ik} = s_{Hik} \frac{\varphi_{ik}}{\varphi_{Hik}} - \frac{s_{Hik}}{2\pi} \sin\left(2\pi \frac{\varphi_{ik}}{\varphi_{Hik}}\right) \quad (3.6)$$

3.3 Übertragungsfunktionen

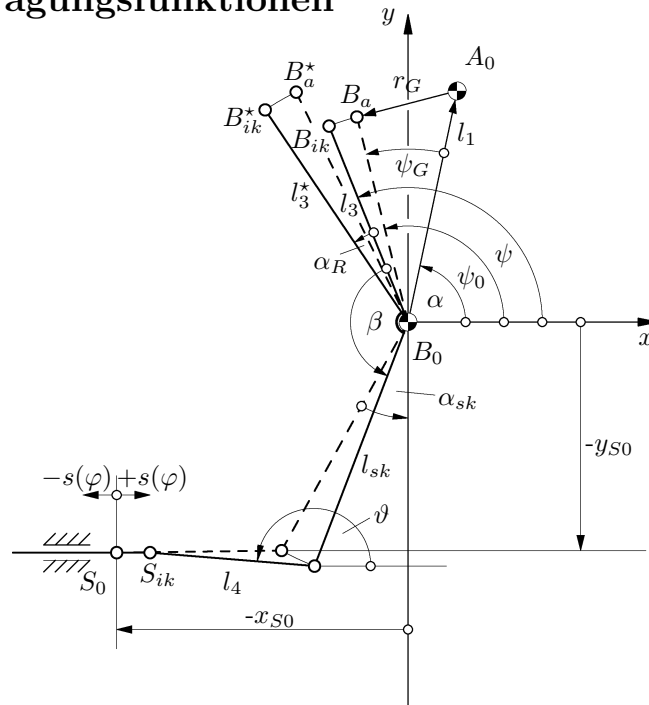


Abbildung 16: Kurvenkoppelgetriebe mit Doppelschwinghebel

Für den Punkt S_{ik} ergibt sich nach Bild 40 folgende Vektorgleichung:

$$l_{sk}e^{i(\beta+\psi)} + l_4e^{i\vartheta} = x_{S0} \pm s(\varphi) - \underbrace{i l_{sk} \cos \alpha_{sk}}_{y_{S0}} \quad (3.7)$$

Umstellung und Bildung der konjugiert komplexen Gleichungen ergibt:

$$l_4e^{i\vartheta} = x_{S0} \pm s(\varphi) - i l_{sk} \cos \alpha_{sk} - l_{sk}e^{i(\beta+\psi)} \quad (3.8)$$

$$l_4e^{-i\vartheta} = x_{S0} \pm s(\varphi) + i l_{sk} \cos \alpha_{sk} - l_{sk}e^{-i(\beta+\psi)} \quad (3.9)$$

Das Produkt dieser beiden Gleichungen ist ein reelles Polynom. Der Realteil der *Eulerschen* Formel⁹ steht zweimal da und der Imaginärteil fällt raus.

$$l_4^2 = [x_{S0} \pm s(\varphi)]^2 + l_{sk}^2 \cos^2 \alpha_{sk} + l_{sk}^2 - 2l_{sk} \cdot \cos(\psi + \beta) \cdot [x_{S0} \pm s(\varphi)] - 2l_{sk} \cdot \sin(\psi + \beta) \cdot l_{sk} \cos \alpha_{sk} \quad (3.10)$$

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

$$\left. \begin{aligned} A &= -2l_{sk} \cdot [x_{S0} \pm s(\varphi)] \\ B &= -2l_{sk} \cdot l_{sk} \cos \alpha_{sk} \\ C &= [x_{S0} \pm s(\varphi)]^2 + l_{sk}^2 \cos^2 \alpha_{sk} + l_{sk}^2 - l_4^2 \end{aligned} \right\} \quad (3.11)$$

folgt daraus die vereinfachte Form der *Übertragungsleichung 0. Ordnung*:

$$0 = A \cos(\psi + \beta) + B \sin(\psi + \beta) + C \quad (3.12)$$

Durch Anwendung der Theoreme

$$\cos(\psi + \beta) = \frac{1 - \tan^2 \frac{\psi + \beta}{2}}{1 + \tan^2 \frac{\psi + \beta}{2}} \quad \sin(\psi + \beta) = \frac{2 \tan \frac{\psi + \beta}{2}}{1 + \tan^2 \frac{\psi + \beta}{2}} \quad (3.13)$$

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

$$\begin{aligned} 0 &= A \left(1 - \tan^2 \frac{\psi + \beta}{2} \right) + 2B \tan \frac{\psi + \beta}{2} + C \left(1 + \tan^2 \frac{\psi + \beta}{2} \right) \\ &= (C - A) \tan^2 \frac{\psi + \beta}{2} + 2B \tan \frac{\psi + \beta}{2} + A + C \quad \bigg/ \div (C - A) \\ &= \tan^2 \frac{\psi + \beta}{2} - \frac{2B}{A - C} \tan \frac{\psi + \beta}{2} - \frac{A + C}{A - C} \end{aligned} \quad (3.14)$$

Gl. (3.14) ist eine quadratische Gleichung. Es gilt

$$\left(\tan \frac{\psi + \beta}{2} \right)_{1/2} = \frac{B}{A - C} \pm \sqrt{\frac{B^2}{(A - C)^2} - \frac{A + C}{A - C}} \quad (3.15)$$

⁹ $e^{i\varphi} = \cos \varphi + i \sin \varphi$

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

$$\begin{aligned}
\left(\tan \frac{\psi + \beta}{2}\right)_{1/2} &= \frac{B}{A - C} \pm \sqrt{\frac{B^2}{(A - C)^2} - \frac{(A + C)(A - C)}{(A - C)^2}} \\
&= \frac{B}{A - C} \pm \frac{1}{A - C} \sqrt{A^2 + B^2 - C^2} \\
&= \frac{B \pm \sqrt{A^2 + B^2 - C^2}}{A - C}
\end{aligned} \tag{3.16}$$

Hiervon ist der Arcus-Tangens zu bilden

$$\begin{aligned}
\frac{\psi + \beta}{2} &= \arctan \frac{B \pm \sqrt{A^2 + B^2 - C^2}}{A - C} \quad \Big/ \cdot 2; -\beta \\
\psi &= 2 \arctan \frac{B \pm \sqrt{A^2 + B^2 - C^2}}{A - C} - \beta
\end{aligned} \tag{3.17}$$

Dies ist die *Übertragungsfunktion 0. Ordnung* $\psi = \psi(\varphi)$. Darin sind l_{sk} , x_{S0} , α_{sk} und l_4 geometrische Größen, die für das zu berechnende Getriebe bekannt sein müssen.

Eine Differentiation von Gl. (3.10) nach dem Drehwinkel φ ergibt:

$$\begin{aligned}
0 &= -2 l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) \psi' + 2 l_{sk} [x_{S0} \pm s(\varphi)] \sin(\psi + \beta) \psi' \\
&\quad - 2 l_{sk} \cos(\psi + \beta) \cdot [\pm s'(\varphi)] + 2 [x_{S0} \pm s(\varphi)] \cdot [\pm s'(\varphi)] \quad \Big/ \div 2 \\
&= -l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) \psi' + l_{sk} [x_{S0} \pm s(\varphi)] \sin(\psi + \beta) \psi' \\
&\quad - l_{sk} \cos(\psi + \beta) \cdot [\pm s'(\varphi)] + [x_{S0} \pm s(\varphi)] \cdot [\pm s'(\varphi)]
\end{aligned} \tag{3.18}$$

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

$$\psi' = \frac{-l_{sk} \cos(\psi + \beta) \cdot [\pm s'(\varphi)] - [x_{S0} \pm s(\varphi)] \cdot [\pm s'(\varphi)]}{l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) + l_{sk} [x_{S0} \pm s(\varphi)] \sin(\psi + \beta)} \tag{3.19}$$

Diese Gleichung enthält neben den geometrischen Größen l_{sk} , x_{S0} , α_{sk} , l_4 und dem Hubverlauf $s(\varphi)$ auch noch die Funktion $\psi(\varphi)$, die aber bereits bekannt ist, siehe Gl. (3.17). Für die *Übertragungsfunktion 2. Ordnung* wird die Gl. (3.18) nach dem Drehwinkel φ differenziert¹⁰

$$\begin{aligned}
0 &= l_{sk}^2 \cos \alpha_{sk} \sin(\psi + \beta) \cdot \psi'^2 + l_{sk} [x_{S0} \pm s(\varphi)] \cos(\psi + \beta) \cdot \psi'^2 \\
&\quad + 2 l_{sk} \sin(\psi + \beta) \cdot [\pm s'(\varphi)] \cdot \psi' + [\pm s'(\varphi)]^2 \\
&\quad - l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) \cdot \psi'' + l_{sk} [x_{S0} \pm s(\varphi)] \sin(\psi + \beta) \cdot \psi'' \\
&\quad - l_{sk} \cos(\psi + \beta) \cdot [\pm s''(\varphi)] + [x_{S0} \pm s(\varphi)] \cdot [\pm s''(\varphi)]
\end{aligned} \tag{3.20}$$

¹⁰m. H. der Produktregel: $y = u \cdot v \cdot u \rightsquigarrow y' = u'vw + uv'w + uvw'$

und diese Ableitung nach ψ'' aufgelöst

$$\begin{aligned}\psi'' = & \frac{l_{sk}^2 \cos \alpha_{sk} \sin(\psi + \beta) \cdot \psi'^2 + l_{sk} [x_{s0} \pm s(\varphi)] \cos(\psi + \beta) \cdot \psi'^2}{l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) - l_{sk} [x_{s0} \pm s(\varphi)] \sin(\psi + \beta)} \\ & + \frac{2 l_{sk} \sin(\psi + \beta) \cdot [\pm s'(\varphi)] \cdot \psi' + [\pm s'(\varphi)]^2}{l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) - l_{sk} [x_{s0} \pm s(\varphi)] \sin(\psi + \beta)} \\ & + \frac{- l_{sk} \cos(\psi + \beta) \cdot [\pm s''(\varphi)] + [x_{s0} \pm s(\varphi)] \cdot [\pm s''(\varphi)]}{l_{sk}^2 \cos \alpha_{sk} \cos(\psi + \beta) - l_{sk} [x_{s0} \pm s(\varphi)] \sin(\psi + \beta)}\end{aligned}\quad (3.21)$$

Ein anderer Weg die Übertragungsfunktionen höherer Ordnungen zu finden, ist die Approximation durch die zentralen Differenzenformeln. Sie lauten für die ersten drei Ableitungen:

$$\psi' = \frac{\psi(\varphi + \Delta\varphi) - \psi(\varphi - \Delta\varphi)}{2\Delta\varphi} \quad (3.22)$$

$$\psi'' = \frac{\psi(\varphi + \Delta\varphi) - 2\psi(\varphi) + \psi(\varphi - \Delta\varphi)}{(\Delta\varphi)^2} \quad (3.23)$$

$$\psi''' = \frac{\psi(\varphi + 2\Delta\varphi) - 2\psi(\varphi + \Delta\varphi) + 2\psi(\varphi - \Delta\varphi) - \psi(\varphi - 2\Delta\varphi)}{2(\Delta\varphi)^2} \quad (3.24)$$

Dabei ist $\Delta\varphi$ ein kleiner Winkel von $\Delta\varphi \stackrel{!}{=} 10^{-4}$ rad. Mit dieser Methode erhält man ebenfalls sehr genaue Ergebnisse. Sie ist programmiertechnisch einfacher zu realisieren.

3.4 Rollenmittelpunktskurve und Kurvenprofil

3.4.1 Kurve

Der Berechnung der Rollenmittelpunktskurve für die Kurve wird die Abbildung 17 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 .

Ausgehend von den Koordinaten der Punkte $A_0(x_{A0}, y_{A0})$, $B_a(x_{Ba}, y_{Ba})$ und $B_{ik}(x_{B_{ik}}, y_{B_{ik}})$ können folgende Hauptabmessungen berechnet werden:

$$\left. \begin{aligned} l_1 &= \sqrt{x_{A0}^2 + y_{A0}^2} \\ r_G &= \sqrt{(x_{Ba} - x_{A0})^2 + (y_{Ba} - y_{A0})^2} \\ \psi_G &= \arccos [(l_3^2 + l_1^2 - r_G^2) / (2l_1 l_3)] \end{aligned} \right\} \quad (3.25)$$

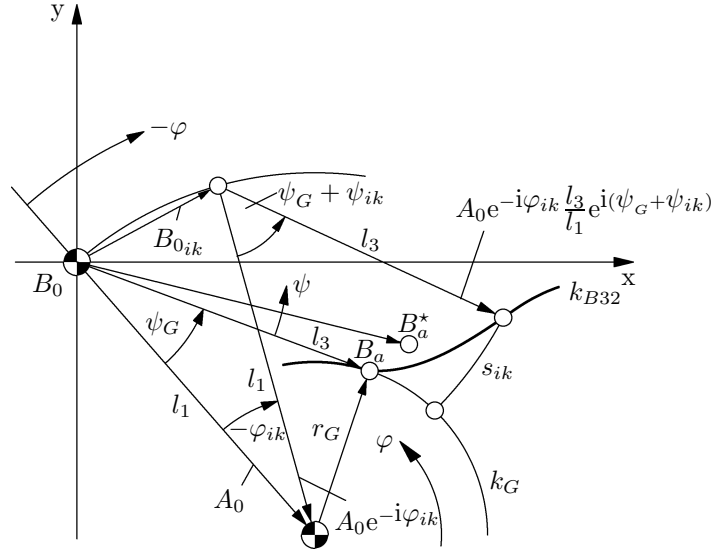


Abbildung 17: Grundfigur zur Berechnung der Rollenmittelpunktskurve bei einem Kurvengetriebe mit Schwinghebel [9]

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

$$B_{0ik} = A_0 - A_0 e^{-i\varphi_{ik}} \quad (3.26)$$

Für den Vektor B_{ik} auf der Mittelpunktskurve gilt:

$$B_{ik} = B_{0ik} + A_0 e^{-i\varphi_{ik}} \frac{l_3}{l_1} e^{i(\psi_G + \psi_{ik})} \quad (3.27)$$

Gl. (3.26) in Gl. 3.27 Einsetzen ergibt:

$$\begin{aligned} B_{ik} &= A_0 - A_0 e^{-i\varphi_{ik}} + A_0 e^{-i\varphi_{ik}} \frac{l_3}{l_1} e^{i(\psi_G + \psi_{ik})} \\ &= A_0 \left[1 - e^{-i\varphi_{ik}} + \frac{l_3}{l_1} e^{i(\psi_G + \psi_{ik} - \varphi_{ik})} \right] \end{aligned} \quad (3.28)$$

Unter Anwendung der kartesischen Darstellung $Z = x_Z + iy_Z$ und der Eulerschen Formeln $e^{i\varphi} = \cos \varphi + i \sin \varphi$, $e^{-i\varphi} = \cos \varphi - i \sin \varphi$ auf Gleichung (3.28) folgt daraus:

$$\begin{aligned} x_{Bik} + iy_{Bik} &= (x_{A0} + iy_{A0}) \left\{ 1 - (\cos \varphi_{ik} - i \sin \varphi_{ik}) + \frac{l_3}{l_1} [\cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right. \\ &\quad \left. + i \sin(\psi_G + \psi_{ik} - \varphi_{ik})] \right\} \end{aligned} \quad (3.29)$$

Hieraus sind die Koordinaten des Vektors B_{ik} (Arbeitskurve) ablesbar:

$$\begin{aligned} x_{Bik} = x_{A0} & \left[1 - \cos \varphi_{ik} + \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ & - y_{A0} \left[\sin \varphi_{ik} + \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.30)$$

$$\begin{aligned} y_{Bik} = x_{A0} & \left[\sin \varphi_{ik} + \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ & + y_{A0} \left[1 - \cos \varphi_{ik} + \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.31)$$

Zur Berechnung des Kurvenprofils (innere bzw. äußere Arbeitskurve) wird zunächst der Tangentenvektor B'_{ik} durch Differentiation von Gl. (3.28) nach φ_{ik} gebildet:

$$B'_{ik} = \frac{dB_{ik}}{d\varphi_{ik}} = iA_0 \left[e^{-i\varphi_{ik}} + \frac{l_3}{l_1} (\psi'_{ik} - 1) e^{i(\psi_G + \psi_{ik} - \varphi_{ik})} \right] \quad (3.32)$$

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

$$\begin{aligned} x'_{Bik} = x_{A0} & \left[\sin \varphi_{ik} - (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ & - y_{A0} \left[\cos \varphi_{ik} + (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.33)$$

$$\begin{aligned} y'_{Bik} = x_{A0} & \left[\cos \varphi_{ik} + (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \cos(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \\ & + y_{A0} \left[\sin \varphi_{ik} - (\psi'_{ik} - 1) \cdot \frac{l_3}{l_1} \sin(\psi_G + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.34)$$

Die Vektoren B_{iki} und B_{ika} von innerer und äußerer Arbeitskurve stehen senkrecht auf dem Tangentenvektor B'_{ik} (Abb. 18).

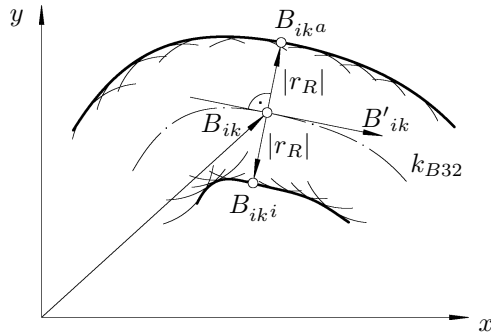


Abbildung 18: Grundfigur zur Berechnung der inneren und äußeren Arbeitskurve [9]

Die Gleichungen zur Berechnung von innerer und äußerer Arbeitskurve lauten:

$$B_{iki} = B_{ik} + ir_R \frac{B'_{ik}}{|B'_{ik}|} \quad (3.35)$$

$$B_{ika} = B_{ik} - ir_R \frac{B'_{ik}}{|B'_{ik}|} \quad (3.36)$$

Hierin ist r_R der Laufrollenradius.

Es folgen damit die Koordinaten von innerer und äußerer Arbeitskurve:

$$x_{iki} = x_{Bik} + r_R \frac{y'_{Bik}}{\sqrt{x'^2_{Bik} + y'^2_{Bik}}} \quad (3.37)$$

$$y_{iki} = y_{Bik} + r_R \frac{x'_{Bik}}{\sqrt{x'^2_{Bik} + y'^2_{Bik}}} \quad (3.38)$$

$$x_{ika} = x_{Bik} - r_R \frac{y'_{Bik}}{\sqrt{x'^2_{Bik} + y'^2_{Bik}}} \quad (3.39)$$

$$y_{ika} = y_{Bik} - r_R \frac{x'_{Bik}}{\sqrt{x'^2_{Bik} + y'^2_{Bik}}} \quad (3.40)$$

3.4.2 Gegenkurve

Die Koordinaten der Rollenmittelpunktskurve für die Gegenkurve werden analog zur Kurve nach Abbildung 19 berechnet.

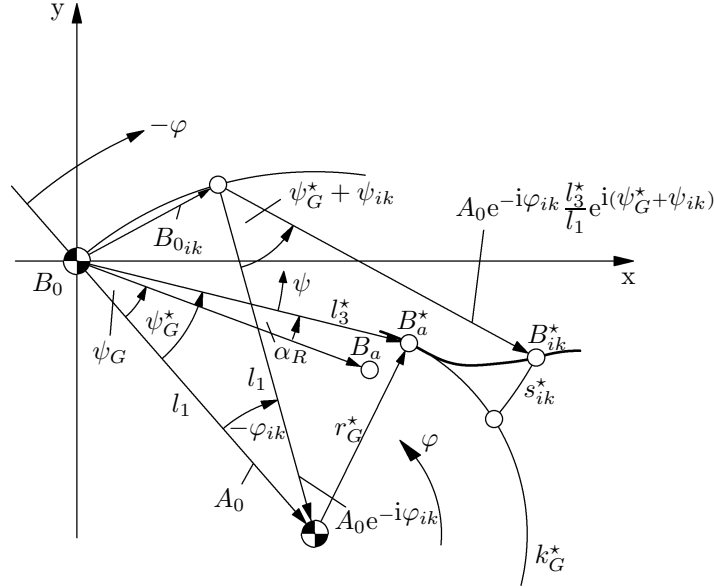


Abbildung 19: Grundfigur zur Berechnung der Rollenmittelpunktskurve der Gegenkurve

Der Grundkreiswinkel ψ_G^* und der Grundkreisradius r_G^* der Gegenkurve berechnen sich zu

$$\left. \begin{aligned} \psi_G^* &= \psi_G + \alpha_R \\ r_G^* &= \sqrt{l_1^2 + l_3^{*2} - 2l_1 l_3^* \cos \psi_G^*} \end{aligned} \right\} \quad (3.41)$$

Daraus ergeben sich die Koordinaten der Rollenmittelpunktskurve:

$$\begin{aligned} x_{B_{ik}}^* &= x_{A0} \left[1 - \cos \varphi_{ik} + \frac{l_3^*}{l_1} \cos(\psi_G^* + \psi_{ik} - \varphi_{ik}) \right] \\ &\quad - y_{A0} \left[\sin \varphi_{ik} + \frac{l_3^*}{l_1} \sin(\psi_G^* + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.42)$$

$$\begin{aligned} y_{B_{ik}}^* &= x_{A0} \left[\sin \varphi_{ik} + \frac{l_3^*}{l_1} \sin(\psi_G^* + \psi_{ik} - \varphi_{ik}) \right] \\ &\quad + y_{A0} \left[1 - \cos \varphi_{ik} + \frac{l_3^*}{l_1} \cos(\psi_G^* + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.43)$$

Die Rollenmittelpunktskoordinaten haben folgende Ableitungen:

$$\begin{aligned} x_{Bik}^{\star \prime} = & x_{A0} \left[\sin \varphi_{ik} - (\psi'_{ik} - 1) \cdot \frac{l_3^{\star}}{l_1} \sin(\psi_G^{\star} + \psi_{ik} - \varphi_{ik}) \right] \\ & - y_{A0} \left[\cos \varphi_{ik} + (\psi'_{ik} - 1) \cdot \frac{l_3^{\star}}{l_1} \cos(\psi_G^{\star} + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.44)$$

$$\begin{aligned} y_{Bik}^{\star \prime} = & x_{A0} \left[\cos \varphi_{ik} + (\psi'_{ik} - 1) \cdot \frac{l_3^{\star}}{l_1} \cos(\psi_G^{\star} + \psi_{ik} - \varphi_{ik}) \right] \\ & + y_{A0} \left[\sin \varphi_{ik} - (\psi'_{ik} - 1) \cdot \frac{l_3^{\star}}{l_1} \sin(\psi_G^{\star} + \psi_{ik} - \varphi_{ik}) \right] \end{aligned} \quad (3.45)$$

Schließlich lassen sich die Koordinaten von innerer und äußerer Arbeitskurve der Gegenkurve aufschreiben:

$$x_{iki}^{\star} = x_{Bik}^{\star} + r_R \frac{y_{Bik}^{\star}}{\sqrt{x_{Bik}^{\star \prime 2} + y_{Bik}^{\star \prime 2}}} \quad (3.46)$$

$$y_{iki}^{\star} = y_{Bik}^{\star} + r_R \frac{x_{Bik}^{\star}}{\sqrt{x_{Bik}^{\star \prime 2} + y_{Bik}^{\star \prime 2}}} \quad (3.47)$$

$$x_{iki}^{\star} = x_{Bik}^{\star} - r_R \frac{y_{Bik}^{\star}}{\sqrt{x_{Bik}^{\star \prime 2} + y_{Bik}^{\star \prime 2}}} \quad (3.48)$$

$$y_{iki}^{\star} = y_{Bik}^{\star} - r_R \frac{x_{Bik}^{\star}}{\sqrt{x_{Bik}^{\star \prime 2} + y_{Bik}^{\star \prime 2}}} \quad (3.49)$$

3.5 Grundkreiswinkel und Grundkreisradius

In Abschnitt 3.1 auf Seite 19 wurde bereits die Ermittlung der A_0 -Bereiche mit dem Verfahren nach *Flocke* beschrieben. Hat man sich die Koordinaten für den Kurvenscheibendrehpunkt vorgegeben, sind der Grundwinkel ψ_G und der Grundkreisradius r_G wie folgt berechenbar. Nach Abbildung 40 berechnet sich der Winkel α zu:

$$\alpha = \arctan \left(\frac{y_{A0}}{x_{A0}} \right) \quad (3.50)$$

Der Grundwinkel ψ_G ist die Differenz aus dem Winkel ψ_0 und dem Winkel α

$$\psi_G = |\psi_0| - |\alpha| \quad (3.51)$$

Zum Grundkreisradius gelangt man über den *Kosinussatz*:

$$r_G = \sqrt{l_1^2 + l_3^2 - 2 l_1 l_3 \cos \psi_G} \quad (3.52)$$

3.6 Übertragungswinkel

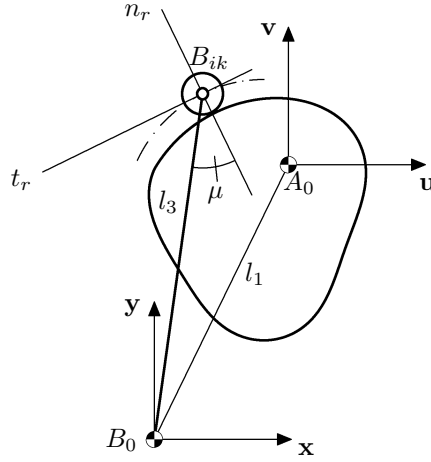


Abbildung 20: Übertragungswinkel μ am Kurvengetriebe

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

$$B_{ik} = u + iv \quad (3.53)$$

Durch Ableiten von Gl. (3.53) erhält man die Geschwindigkeit des Punktes B_{ik} in Richtung der Tangente t_r .

$$B_{ik}' = u' + iv' \quad (3.54)$$

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

$$i \cdot B_{ik}' = i \underbrace{(u' + iv')}_{B_{ik}'} = iu' - v' \quad (3.55)$$

Dieser Vektor und der Vektor des Schwinghebels $B_0 - B_{ik}$

$$B_0 - B_{ik} = -a_{0x} - u + i(-a_{0y} + v) \quad (3.56)$$

schließen den Übertragungswinkel μ ein. Er läßt sich über die Definitionsgleichung des skalaren Produktes berechnen.

$$\begin{aligned} \cos \mu &= \cos (B_0 - B_{ik}, iB_{ik}') \\ &= \frac{(B_0 - B_{ik}, iB_{ik}')}{|B_0 - B_{ik}| \cdot |iB_{ik}'|} \quad / \arccos \\ \mu &= \arccos \frac{(B_0 - B_{ik}, iB_{ik}')}{|B_0 - B_{ik}| \cdot |iB_{ik}'|} \end{aligned} \quad (3.57)$$

Für das Skalarprodukt¹¹ $(B_0 - B_{ik}, iB_{ik}')$ erhält man:

$$\begin{aligned}
(B_0 - B_{ik}, iB_{ik}') &= \frac{1}{2} \left((B_0 - B_{ik}) \cdot \overline{iB_{ik}'} + \overline{(B_0 - B_{ik})} \cdot iB_{ik}' \right) \\
&= \frac{1}{2} \left\{ \underbrace{[(-a_{0x} - u) + i(-a_{0y} + v)]}_{B_0 - B_{ik}} \cdot \underbrace{(u' - iv')}_{\overline{iB_{ik}'}} \right. \\
&\quad \left. + \underbrace{[(-a_{0x} - u) - i(-a_{0y} + v)]}_{\overline{B_0 - B_{ik}}} \cdot \underbrace{(u' + iv')}_{iB_{ik}'} \right\} \\
&= (-a_{0x} - u)u' + (-a_{0y} + v)v'
\end{aligned} \tag{3.58}$$

Die Beträge $|B_0 - B_{ik}|$ und $|iB_{ik}'|$ lauten:

$$|B_0 - B_{ik}| = \sqrt{(-a_{0x} - u)^2 - (-a_{0y} + v)^2} \tag{3.59}$$

$$|iB_{ik}'| = \sqrt{u'^2 + v'^2} \tag{3.60}$$

Einsetzen von Gl. (3.58) bis Gl. (3.60) in Gl. (3.57) ergibt:

$$\mu = \arccos \frac{(-a_{0x} - u)u' + (-a_{0y} + v)v'}{\sqrt{(-a_{0x} - u)^2 - (-a_{0y} + v)^2} \sqrt{u'^2 + v'^2}} \tag{3.61}$$

3.7 Krümmungsradius

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

$$r_{Kik} = \frac{\sqrt{(x_{Bik}'^2 + y_{Bik}'^2)^3}}{x_{Bik}'y_{Bik}'' - x_{Bik}''y_{Bik}'} \tag{3.62}$$

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

$$r_{K\min} \geq \frac{r_R}{0.7} \tag{3.63}$$

Ist der Krümmungsradius r_K gleich dem Rollenradius r_R , entsteht an der Kurvenkontur eine Spitze (Abb. 21). Wenn der Krümmungsradius r_K kleiner als der Rollenradius r_R ist, entsteht Hubverlust infolge von Unterschnitt.

¹¹Die Rechenregel $A \cdot B = \frac{1}{2} (\overline{AB} + \overline{BA})$ liefert eine reelle Zahl.

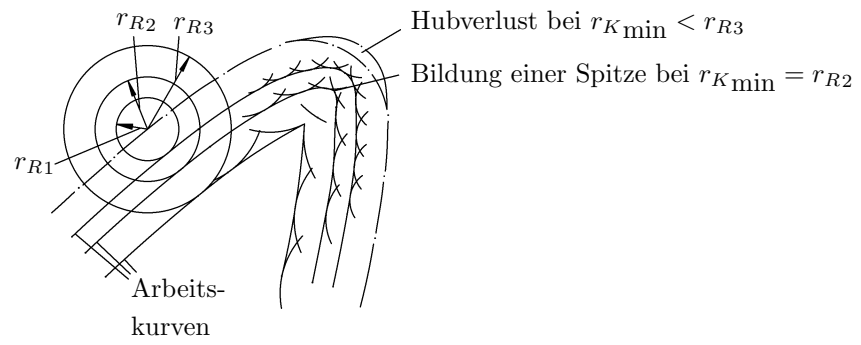


Abbildung 21: Spitzenbildung und Unterschnitt bei zu kleinem Krümmungsradius r_K [16]

4 Dynamisches Verhalten von Kurvengetrieben

Die bisherigen Betrachtungen gingen bei der Aufstellung der Übertragungsfunktionen von masselosen Getriebemodellen aus. Diese Vereinfachung ist nur erlaubt, wenn:

- das Getriebe aus starren Körpern besteht, die sich nicht relativ zueinander bewegen
- niedrige Arbeitsgeschwindigkeiten vorliegen
- niedrige Kräfte (z.B. Antriebs-, Massen- oder Lagerkräfte) wirken

Sind diese Bedingungen nicht mehr erfüllt, dann treten im System Schwingungen auf, die Abweichungen vom gewünschten Bewegungsverlauf hervorrufen. Bild 22 zeigt das Bewegungsdiagramm der allgemeinen Rastbewegung mit den maximalen Abweichungen Δs .

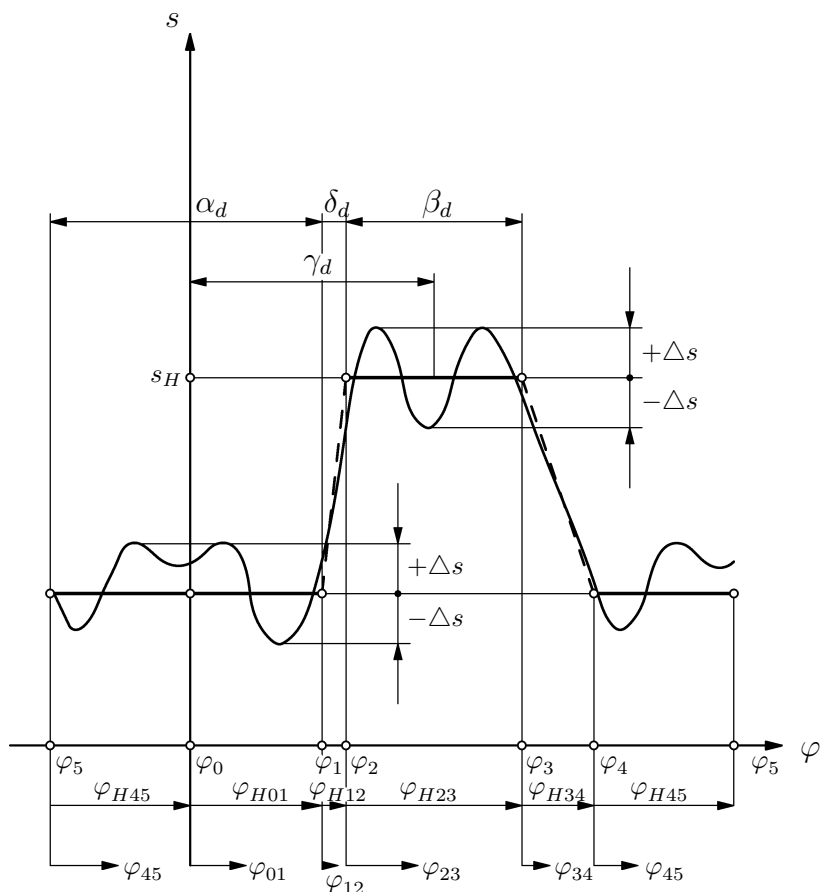


Abbildung 22: Bewegungsdiagramm der allgemeinen Rast-in-Rast-Bewegung [3]

Wie man für solche Problemstellungen die Übertragungsfunktionen finden kann, beschreiben die folgenden Abschnitte.

4.1 Modellbildung

Bild 23 zeigt ein Einmassenmodell für ein Kurvengetriebe. Es ist für Antriebssysteme geeignet, bei denen der Antrieb und die Kurvenscheibe eine hohe Steifigkeit aufweisen, die Abtriebsglieder jedoch Schwingungen ausführen. Bei diesen Systemen können die Masse m und die Steifigkeit c als konstante Größen betrachtet werden.

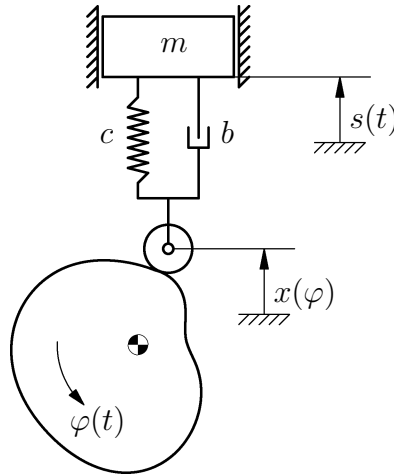


Abbildung 23: Einmassenmodell eines Kurvengetriebes [3]

Die Bewegungsgleichung für den Abtriebshub s lautet:

$$m\ddot{s} + b(\dot{s} - \dot{x}) + c(s - x) = 0 \quad (4.1)$$

Mit Gl. (2.1) von Seite 3 wird daraus:

$$\begin{aligned} m\ddot{s} + b\dot{s} + cs &= bx'\dot{\varphi} + cx \\ \ddot{s} + \frac{b}{m}\dot{s} + \frac{c}{m}s &= \frac{b}{m}x'\dot{\varphi} + \frac{c}{m}x \end{aligned} \quad \bigg/ \div m \quad (4.2)$$

4.1.1 Lösung der Differentialgleichung

Mit der Eigenkreisfrequenz $\omega_0 = \sqrt{c/m}$ und dem *Lehrschen* Dämpfungsmaß $\vartheta_L = b/2\sqrt{cm}$ erhält man eine inhomogene Differentialgleichung zweiter Ordnung:

$$\ddot{s} + 2\vartheta_L \omega_0 \dot{s} + \omega_0^2 s = 2\vartheta_L \omega_0 x' \dot{\phi} + \omega_0^2 x \quad (4.3)$$

Der periodische Bewegungsverlauf $x(\varphi)$ lässt sich in Form einer *Fourier-Reihe* darstellen:

$$\begin{aligned} x(\varphi) &= a_0 + \sum_{k=1}^{\infty} a_k \cos k\varphi + \sum_{k=1}^{\infty} b_k \sin k\varphi \\ &= a_0 + \sum_{k=1}^{\infty} a_k \cos k\Omega t + \sum_{k=1}^{\infty} b_k \sin k\Omega t \end{aligned} \quad (4.4)$$

Darin sind Ω die Erregerfrequenz der Kurvenscheibe ($\Omega = \dot{\varphi}$) und k die Ordnungszahl der Harmonischen. Durch die *Fourier-Reihe* wird also der Bewegungsverlauf in einzelne reine Schwingungen (Harmonische) zerlegt. Für den Bewegungsverlauf (4.4) erhält man folgende Ableitung

$$\begin{aligned} x'(\varphi) &= - \sum_{k=1}^{\infty} a_k k \sin k\varphi + \sum_{k=1}^{\infty} b_k k \cos k\varphi \\ &= - \sum_{k=1}^{\infty} a_k k \sin k\Omega t + \sum_{k=1}^{\infty} b_k k \cos k\Omega t \end{aligned} \quad (4.5)$$

Der Ansatz zur Lösung der DGL (4.3) wird ebenfalls in die Form einer *Fourier-Reihe* gebracht:

$$s = c_0 + \sum_{k=1}^{\infty} c_k \cos k\Omega t + \sum_{k=1}^{\infty} d_k \sin k\Omega t \quad (4.6)$$

Für den Ansatz (4.6) ergeben sich folgende Ableitungen:

$$\dot{s} = - \sum_{k=1}^{\infty} c_k k \Omega \sin k\Omega t + \sum_{k=1}^{\infty} d_k k \Omega \cos k\Omega t \quad (4.7)$$

$$\ddot{s} = - \sum_{k=1}^{\infty} c_k k^2 \Omega^2 \cos k\Omega t - \sum_{k=1}^{\infty} d_k k^2 \Omega^2 \sin k\Omega t \quad (4.8)$$

Einsetzen der Gleichungen (4.4) bis (4.8) in die DGL (4.3) ergibt:

$$\begin{aligned} &\omega_0^2 c_0 + [(\omega_0^2 - \Omega^2)c_1 + 2\vartheta_L \omega_0 \Omega d_1] \cos \Omega t \\ &\quad + [(\omega_0^2 - \Omega^2)d_1 - 2\vartheta_L \omega_0 \Omega c_1] \sin \Omega t \\ &\quad + [(\omega_0^2 - 4\Omega^2)c_2 + 2\vartheta_L \omega_0 2\Omega d_2] \cos \Omega t \\ &\quad + [(\omega_0^2 - 4\Omega^2)d_2 - 2\vartheta_L \omega_0 2\Omega c_2] \sin \Omega t \\ &\quad + \dots \\ &= \omega_0^2 a_0 + (\omega_0^2 a_1 + 2\vartheta_L \omega_0 \Omega b_1) \cos \Omega t \\ &\quad + (\omega_0^2 b_1 - 2\vartheta_L \omega_0 \Omega a_1) \sin \Omega t \\ &\quad + (\omega_0^2 a_2 + 2\vartheta_L \omega_0 2\Omega b_2) \cos \Omega t \\ &\quad + (\omega_0^2 b_2 - 2\vartheta_L \omega_0 2\Omega a_2) \sin \Omega t \\ &\quad + \dots \end{aligned} \quad (4.9)$$

Beim Vergleich der Koeffizienten von $\sin \Omega t$ und $\cos \Omega t$ erhält man folgendes Gleichungssystem zur Bestimmung der Unbekannten c_0 , c_k und d_k

$$\left. \begin{aligned} \omega_0^2 c_0 &= \omega_0^2 a_0 \\ (\omega_0^2 - k^2 \Omega^2) c_k + 2\vartheta_L \omega_0 k \Omega d_k &= \omega_0^2 a_k + 2\vartheta_L \omega_0 k \Omega b_k \\ (\omega_0^2 - k^2 \Omega^2) d_k - 2\vartheta_L \omega_0 k \Omega c_k &= \omega_0^2 b_k - 2\vartheta_L \omega_0 k \Omega a_k \end{aligned} \right\} \quad (4.10)$$

Daraus findet man

$$\left. \begin{aligned} c_0 &= a_0 \\ c_k &= \frac{\omega_0^2(4k^2\Omega^2\vartheta_L^2 - k^2\Omega^2 + \omega_0^2)a_k + 2\omega_0(k\Omega\vartheta_L\omega_0^2 - k^3\Omega^3\vartheta_L - k\Omega\vartheta_L\omega_0^2)b_k}{(\omega_0^2 - k^2\Omega^2)^2 + 4k^2\Omega^2\vartheta_L^2\omega_0^2} \\ d_k &= \frac{2\omega_0(k^3\Omega^3\vartheta_L - k\Omega\vartheta_L\omega_0^2 - k\Omega\vartheta_L\omega_0^2)a_k + \omega_0^2(4k^2\Omega^2\vartheta_L^2 - k^2\Omega^2 + \omega_0^2)b_k}{(\omega_0^2 - k^2\Omega^2)^2 + 4k^2\Omega^2\vartheta_L^2\omega_0^2} \end{aligned} \right\} \quad (4.11)$$

Bei Vernachlässigung der Dämpfung ($\vartheta_L = 0$) und Einführung des Abstimmungsverhältnisses ($\eta = \Omega/\omega_0$) ergibt sich

$$\left. \begin{aligned} c_0 &= a_0 \\ c_k &= \frac{\omega_0^2(\omega_0^2 - k^2\Omega^2)}{(\omega_0^2 - k^2\Omega^2)^2} a_k = \frac{\omega_0^2}{\omega_0^2 - k^2\Omega^2} a_k = \frac{1}{1 - k^2\eta^2} a_k \\ d_k &= \frac{\omega_0^2(\omega_0^2 - k^2\Omega^2)}{(\omega_0^2 - k^2\Omega^2)^2} b_k = \frac{\omega_0^2}{\omega_0^2 - k^2\Omega^2} b_k = \frac{1}{1 - k^2\eta^2} b_k \end{aligned} \right\} \quad (4.12)$$

Setzt man Gl. (4.20) in Gl. (4.6) ein, findet man die Istfunktion der Bewegung $s(\varphi)$

$$\begin{aligned} s(\varphi) &= a_0 + \sum_{k=1}^{n_H} \frac{1}{1 - k^2\eta^2} (a_k \cos k\Omega t + b_k \sin k\Omega t) \\ &= A(0) + \sum_{k=1}^{n_H} (A(k) \cos k\Omega t + B(k) \sin k\Omega t) \end{aligned} \quad (4.13)$$

In dieser Gleichung ist n_H die Anzahl der mitzuführenden sog. Hauptharmonischen und η das Abstimmungsverhältnis ($\eta = \Omega/\omega_0$). Die Koeffizienten $A(k)$ und $B(k)$ wurden in der vorliegenden Arbeit mit dem *Marquard-Levenberg* Algorithmus berechnet. Es sind aber auch andere Verfahren möglich, je nachdem welches Rechenwerkzeug verwendet wird.

4.1.2 Interpretation des Bewegungsgesetzes

Unter der Voraussetzung des schwach gedämpften Systems ($\vartheta_L < 0.2$) tritt bei folgenden Erregerfrequenzen Resonanz¹² auf:

$$\Omega = \frac{\omega_0}{k} \quad k = 1, 2, 3, \dots, n_H \quad (4.14)$$

Sind die Anteile der Harmonischen kleiner als die Eigenkreisfrequenz ($k\Omega < \omega_0$), wird der Nenner der Vergrößerungsfunktion in Gl. (4.13) positiv:

$$1 - k^2\eta^2 = 1 - \underbrace{k^2 \frac{\Omega^2}{\omega_0^2}}_{< 1} > 0 \quad (4.15)$$

¹²Auftreten maximaler Amplituden

Umgekehrt wird der Nenner negativ, wenn die Anteile der Harmonischen größer als die Eigenfrequenz sind ($k\Omega > \omega_0$):

$$1 - k^2 \eta^2 = 1 - k^2 \underbrace{\frac{\Omega^2}{\omega_0^2}}_{> 1} < 0 \quad (4.16)$$

In diesem Fall wird der Bewegungsverlauf $s(\varphi)$ gegenphasig erzeugt. Um gegenphasige Bewegungsverläufe und Resonanz zu vermeiden, ist deshalb immer die nachfolgende Bedingung einzuhalten:

$$n_H \ll \frac{\omega_0}{\Omega} \quad (4.17)$$

Weiterhin ist das Abstimmungsverhältnis η günstig zu wählen, d.h. nicht $\eta \rightarrow 0$ (Kriechgang) sondern $\eta = \eta_{opt} > 0$ (optimale Drehzahl kleiner als die Betriebsdrehzahl).

Nach [3] kann man für Rast-in-Rast-Bewegungen gemäß Bild 22 das optimale Abstimmungsverhältnis wie folgt berechnen

$$\eta_{opt}^2 = \frac{A(0) - s(\chi_d) + \sum_{k=1}^{n_H} [A(k) \cos k\chi_d + B(k) \sin k\chi_d]}{\sum_{k=1}^{n_H} k^2 [A(k) \cos k\chi_d + B(k) \sin k\chi_d]} \quad (4.18)$$

Hierin ist χ_d der Winkel zu Beginn der kürzeren Rast, bei dem die größten Abweichungen Δs zu erwarten sind.

Neben der Anzahl n_H der mitgeführten Harmonischen und dem Abstimmungsverhältnis η sind die erreichbaren Genauigkeiten der Kurvenprofile noch vom Abstand zwischen den benachbarten Rasten abhängig. Dies ist bei der allgemeinen Rast-in-Rastbewegung der Winkel δ_d (vgl. Bild 22 S. 33).

In Bild 24 sind die relativen Genauigkeiten $\Delta s/s_H$ in Abhängigkeit von diesem Abstand δ_d dargestellt.

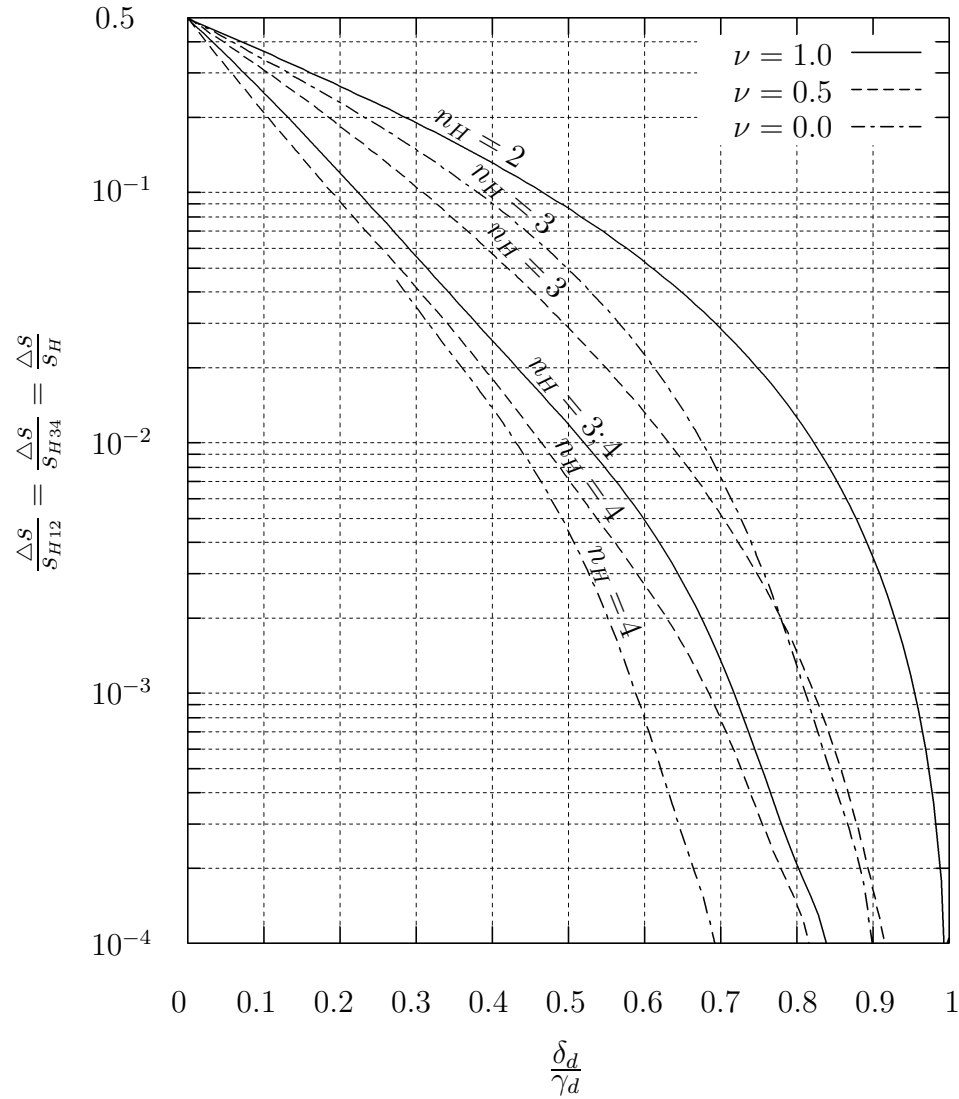


Abbildung 24: Relative Genauigkeiten für Rast-in-Rast-Bewegung [3]

Sonderfälle der allgemeinen Rast-in-Rast-Bewegung sind die symmetrische Rast-in-Rast-Bewegung und die Rast-Umkehr-Bewegung. Für ein Rastverhältnis¹³ von $\nu = 1$ liegt symmetrische Rast-in-Rast-Bewegung vor. Bei einem Rastverhältnis von $\nu = 0$ ist es eine Rast-Umkehr-Bewegung.

Für $n_H = 2$ wurde nur die Kurve von $\nu = 1$ dargestellt, weil sie mit den Kurven der anderen Rastverhältnisse nahezu übereinstimmt.

Das Diagramm von Bild 24 ist ein Hilfsmittel bei der Ermittlung von n_H . In Abschnitt 4.2 ab Seite 39 kann der Leser anhand von Beispielen die Wahl von n_H nachvollziehen.

¹³von oberer (kürzerer) zu unterer (längerer) Rast

4.2 Beispiele

4.2.1 Querhub

In Tabelle 3 sind die gegebenen Werte für die Querhubbewegung zusammengestellt.

Tabelle 3: Vorgaben für die Querhubbewegung

Größe	Wert	Größe	Wert
Drehwinkel φ_{H01}	70°	Hub s_H	5 mm
Drehwinkel φ_{H12}	40°	Abweichung $\triangle s$	0.05 mm
Drehwinkel φ_{H23}	190°		
Drehwinkel φ_{H34}	40°		
Drehwinkel φ_{H45}	20°		

Die Rastbreite δ_d des Übergangs von längerer zu kürzerer Rast entspricht dem Gesamtdrehwinkel φ_{H34} des 4. Abschnittes. Weiterhin werden die Verhältnisse $\triangle s/s_H$, δ_d/γ_d und $\nu = \beta_d/\alpha_d$ für die Bestimmung von n_H benötigt.

$$\left. \begin{aligned} \delta_d &= \varphi_{H34} = 40^\circ \\ \frac{\triangle s}{s_H} &= \frac{0.05}{5} = 0.01 \\ \frac{\delta_d}{\gamma_d} &= \frac{40^\circ}{180^\circ} = 0.2222 \\ \alpha_d &= \varphi_{H23} = 190^\circ \\ \beta_d &= \varphi_{H45} + \varphi_{H01} = 90^\circ \\ \nu &= \frac{\beta_d}{\alpha_d} = \frac{90^\circ}{190^\circ} = 0.4737 \end{aligned} \right\} \quad (4.19)$$

Mit diesen Werten liest man im Diagramm von Bild 24 ab, dass für diese Bewegungsaufgabe 4 Harmonische ($n_H = 4$) notwendig sind.

Die mit dem *Marquardt-Levenberg* Algorithmus gefundenen Koeffizienten $A(k)$ und $B(k)$ lauten für $\eta_{opt} = 0$:

$$\left. \begin{aligned} A(0) &= a_0 = 3.14895 \\ A(1) &= a_1 = -2.5767 & B(1) &= b_1 = -1.20153 \\ A(2) &= a_2 = -0.664103 & B(2) &= b_2 = -0.791447 \\ A(3) &= a_3 = 0.073979 & B(3) &= b_3 = 0.276093 \\ A(4) &= a_4 = -0.0848508 & B(4) &= b_4 = 0.481213 \end{aligned} \right\} \quad (4.20)$$

In Bild 25 ist der Bewegungsverlauf $s(\varphi)$ dargestellt.

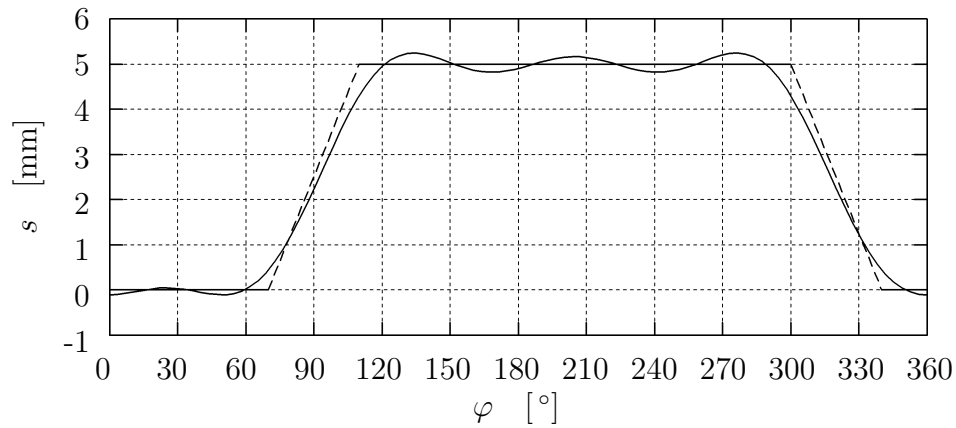
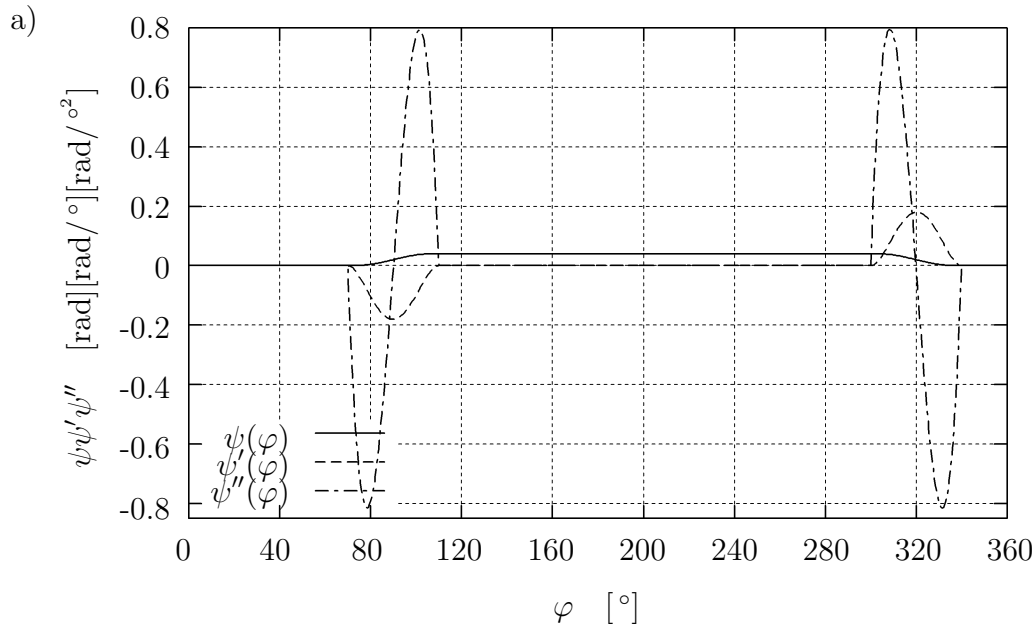


Abbildung 25: Bewegungsverlauf $s(\varphi)$ beim Querhub

Die für diesen schwingungsarmen Bewegungsverlauf nach Abschnitt 4.1.1 (S. 34) berechneten Übertragungsfunktionen sind in Bild 26 b) dargestellt. Zum besseren Vergleich wurde die gleiche Skalierung wie beim 3-4-5 Polynom verwendet (Abb. 26 a)). Die maximalen Beschleunigungen sind beim trigonometrischen Polynom deutlich niedriger als beim 3-4-5 Polynom. Damit sind auch die Massenkräfte geringer.

Für die Berechnungen kam das Programm *Mathematica* zum Einsatz. Die Datei mit den Berechnungen ist im Anhang ab Seite 76 aufgeführt.



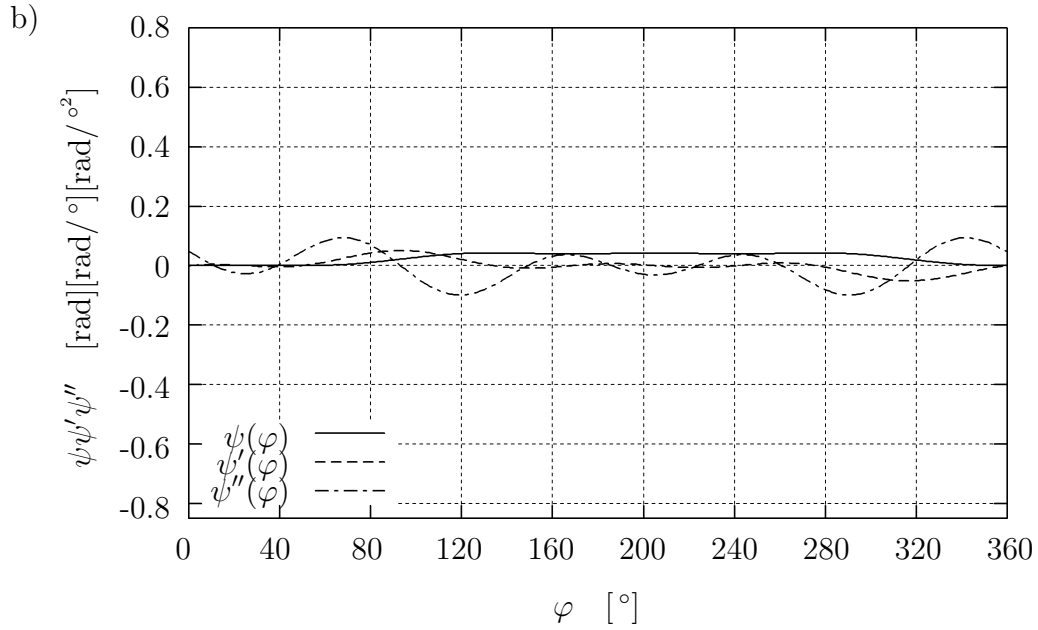


Abbildung 26: Vergleich der Querhubverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungssetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom

Bild 27 zeigt die Rollenmittelpunktskurve und die Arbeitskurven, wie sie sich gemäß Abschnitt 3.4 für die Querhubkurvenscheibe ergeben. Den vollständigen Rechengang findet man im Anhang ab Seite 77.

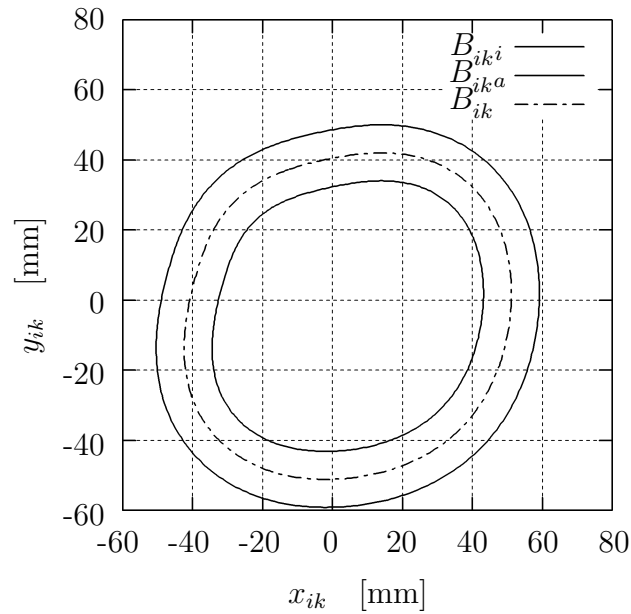


Abbildung 27: schwingungsarme Querhubkurvenscheibe

4.2.2 Schnittbewegung

Tabelle 4 enthält die Ausgangsdaten für die Schnittbewegung.

Tabelle 4: Vorgaben für die Schnittbewegung

Größe	Wert	Größe	Wert
Drehwinkel φ_{H01}	10°	Hub s_H	5.5 mm
Drehwinkel φ_{H12}	40°	Abweichung $\triangle s$	0.05 mm
Drehwinkel φ_{H23}	30°		
Drehwinkel φ_{H34}	280°		

Hier ist die Rastbreite δ_d des Übergangs von längerer zu kürzerer Rast der Winkel φ_{H12} . Die Verhältnisse $\triangle s/s_H$, δ_d/γ_d und $\nu = \beta_d/\alpha_d$ ergeben sich zu:

$$\left. \begin{aligned} \delta_d &= \varphi_{H12} = 40^\circ \\ \frac{\triangle s}{s_H} &= \frac{0.05}{5.5} = 0.0091 \\ \frac{\delta_d}{\gamma_d} &= \frac{40^\circ}{185^\circ} = 0.21622 \\ \alpha_d &= \varphi_{H45} + \varphi_{H01} = 290^\circ \\ \beta_d &= 0^\circ \\ \nu &= \frac{\beta_d}{\alpha_d} = \frac{0^\circ}{290^\circ} = 0 \end{aligned} \right\} \quad (4.21)$$

Ablesen in 24 ergibt 4 mitzuführende Harmonische. Die Koeffizienten $A(k)$ und $B(k)$ nehmen folgende Werte an:

$$\left. \begin{aligned} A(0) &= a_0 = 0.815935 \\ A(1) &= a_1 = 1.09325 & B(1) &= b_1 = 1.11299 \\ A(2) &= a_2 = -0.0259486 & B(2) &= b_2 = 1.35807 \\ A(3) &= a_3 = -0.776141 & B(3) &= b_3 = 0.727385 \\ A(4) &= a_4 = -0.72753 & B(4) &= b_4 = -0.0388622 \end{aligned} \right\} \quad (4.22)$$

Den daraus resultierenden Bewegungsverlauf zeigt Bild 28.

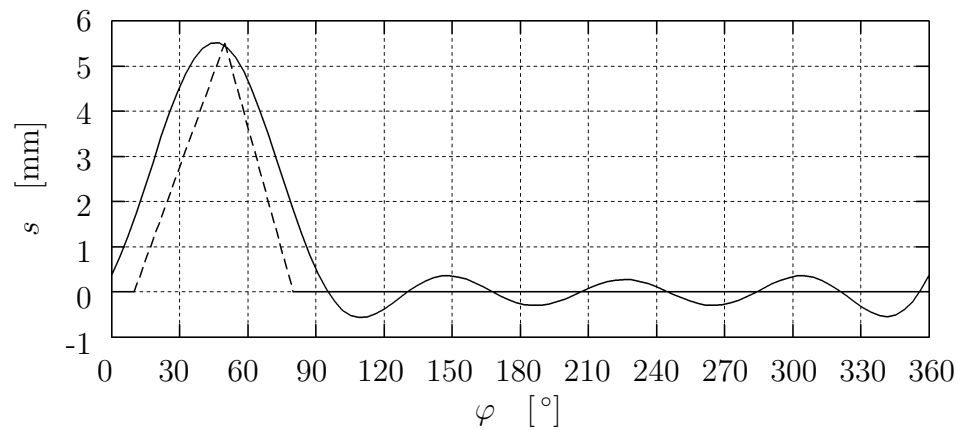
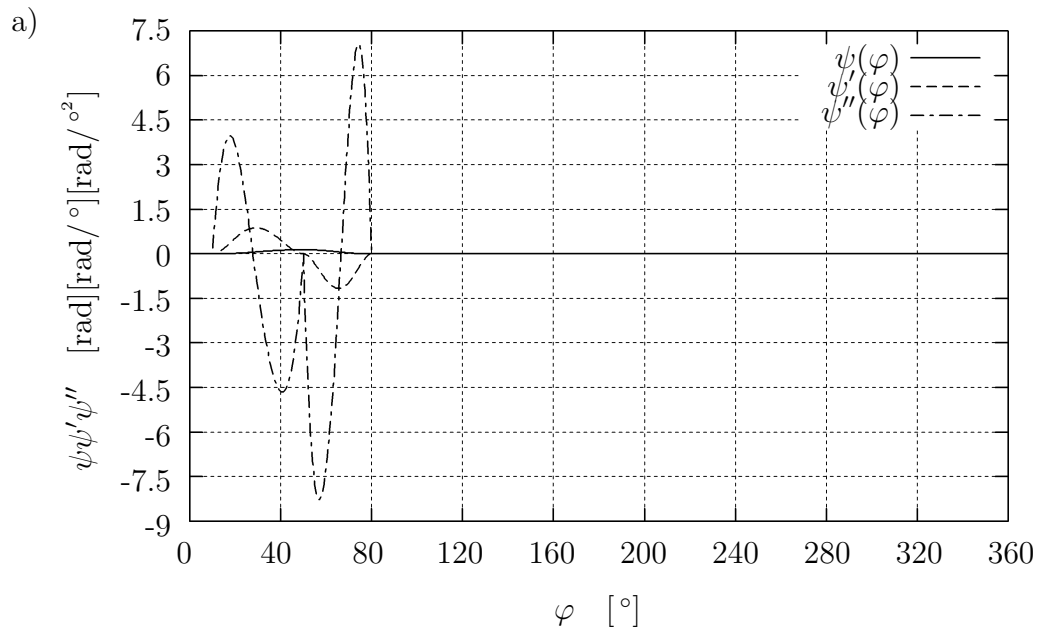


Abbildung 28: Bewegungsverlauf $s(\varphi)$ der Schnittbewegung

Beim Vergleich der Verläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ fällt wieder das trigonometrische Polynom durch deutlich niedrigere Beschleunigungen auf (Abb. 29 b)).



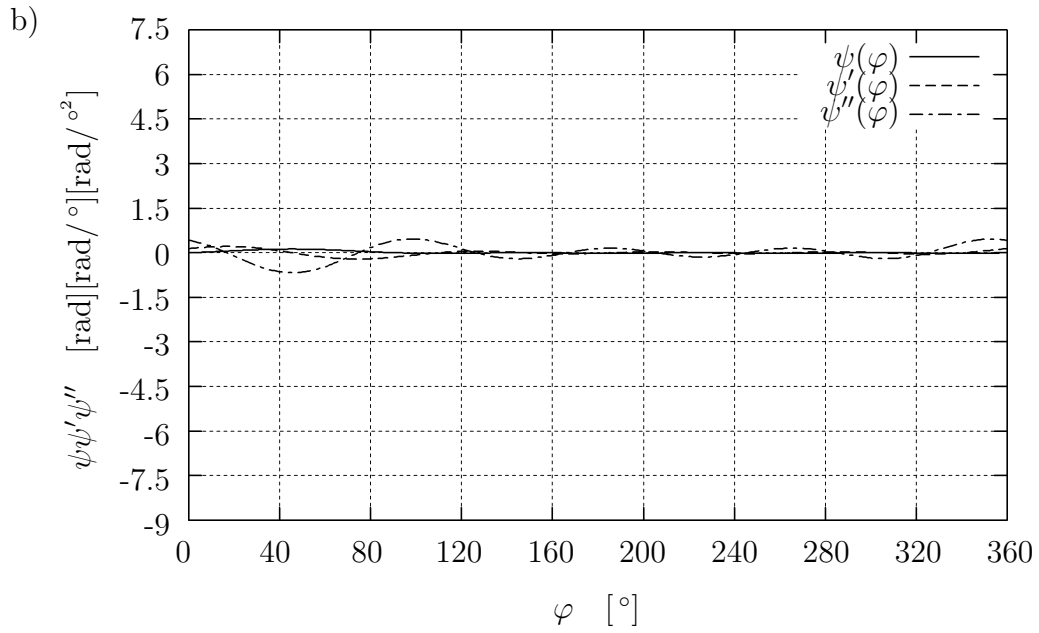


Abbildung 29: Vergleich der Schnittbewegungsverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungsgesetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom

Die errechnete Rollenmittelpunktskurve und die Arbeitskurven sind in Bild 30 dargestellt.

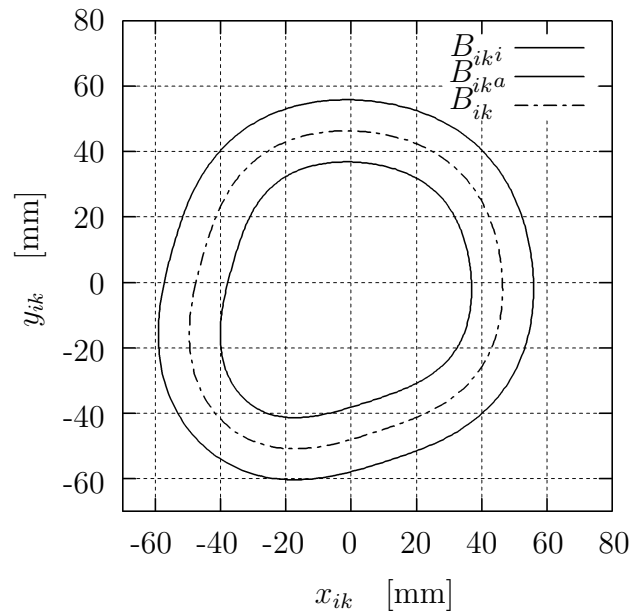


Abbildung 30: schwingungsarme Schnittkurvenscheibe

4.2.3 Setzhub

Tabelle 6 enthält die Ausgangsdaten für die Setzhubbewegung.

Tabelle 5: Vorgaben für die Setzhubbewegung

Größe	Wert	Größe	Wert
Drehwinkel φ_{H01}	110°	Hub s_H	58.5 mm
Drehwinkel φ_{H12}	80°	Abweichung $\triangle s$	0.05 mm
Drehwinkel φ_{H23}	20°		
Drehwinkel φ_{H34}	90°		
Drehwinkel φ_{H45}	60°		

Auch hier ist die Rastbreite δ_d gleich dem Winkel φ_{H12} . Die Verhältnisse $\triangle s/s_H$, δ_d/γ_d und $\nu = \beta_d/\alpha_d$ sind:

$$\left. \begin{aligned} \delta_d &= \varphi_{H12} = 80^\circ \\ \frac{\triangle s}{s_H} &= \frac{0.05}{58.5} = 0.0008547 \\ \frac{\delta_d}{\gamma_d} &= \frac{80^\circ}{135^\circ} = 0.59259 \\ \alpha_d &= \varphi_{H45} + \varphi_{H01} = 170^\circ \\ \beta_d &= \varphi_{H23} = 20^\circ \\ \nu &= \frac{\beta_d}{\alpha_d} = \frac{20^\circ}{170^\circ} = 0.11765 \end{aligned} \right\} \quad (4.23)$$

Die Anzahl der mitzuführenden Harmonischen ist wieder 4. Man erhält folgende Koeffizienten:

$$\left. \begin{aligned} A(0) &= a_0 = 16.9003 \\ A(1) &= a_1 = -25.064 & B(1) &= b_1 = -10.4783 \\ A(2) &= a_2 = 9.45316 & B(2) &= b_2 = 9.16597 \\ A(3) &= a_3 = -1.41296 & B(3) &= b_3 = -2.09477 \\ A(4) &= a_4 = 0.356817 & B(4) &= b_4 = -0.765526 \end{aligned} \right\} \quad (4.24)$$

Daraus ergibt sich der in Bild 31 dargestellte Bewegungsverlauf.

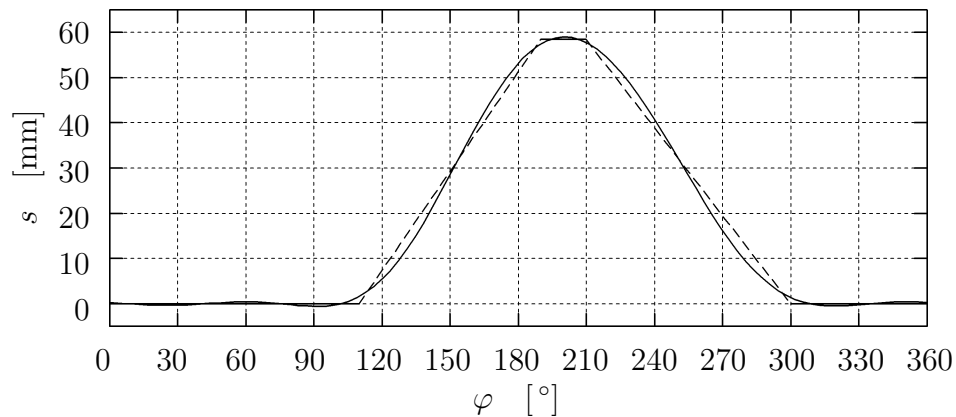
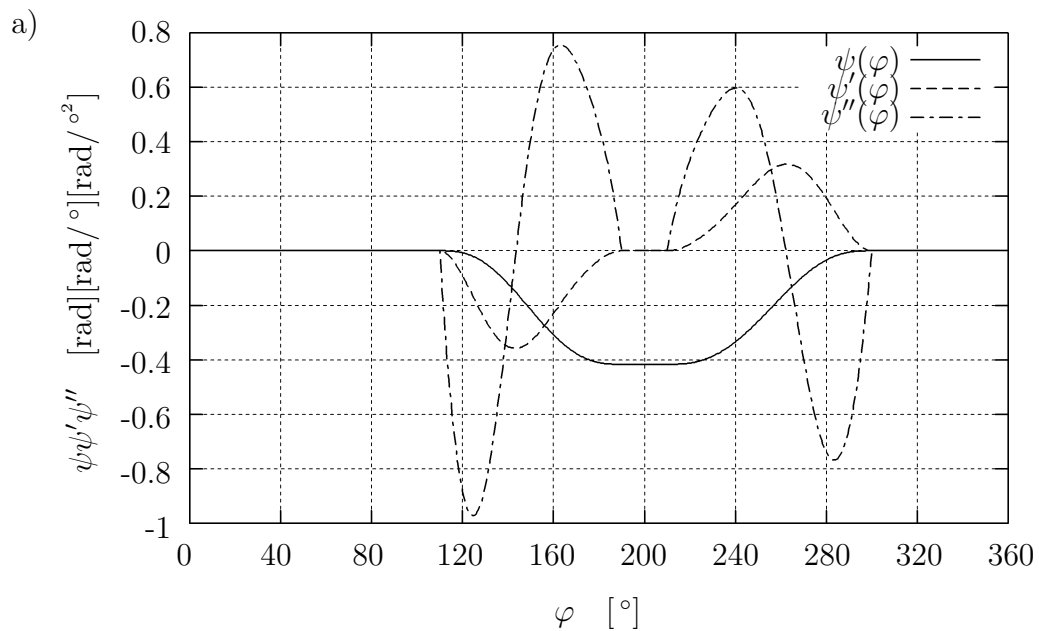


Abbildung 31: Bewegungsverlauf $s(\varphi)$ der Setzhubbewegung

Die Verläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ der Setzhubbewegung zeigt Bild 32. Die Beschleunigungen unterscheiden sich diesmal nicht so sehr. In diesem Fall ist genau zu prüfen, welches Bewegungsgesetz günstiger ist (3-4-5 Polynom oder trigonometrisches Polynom).



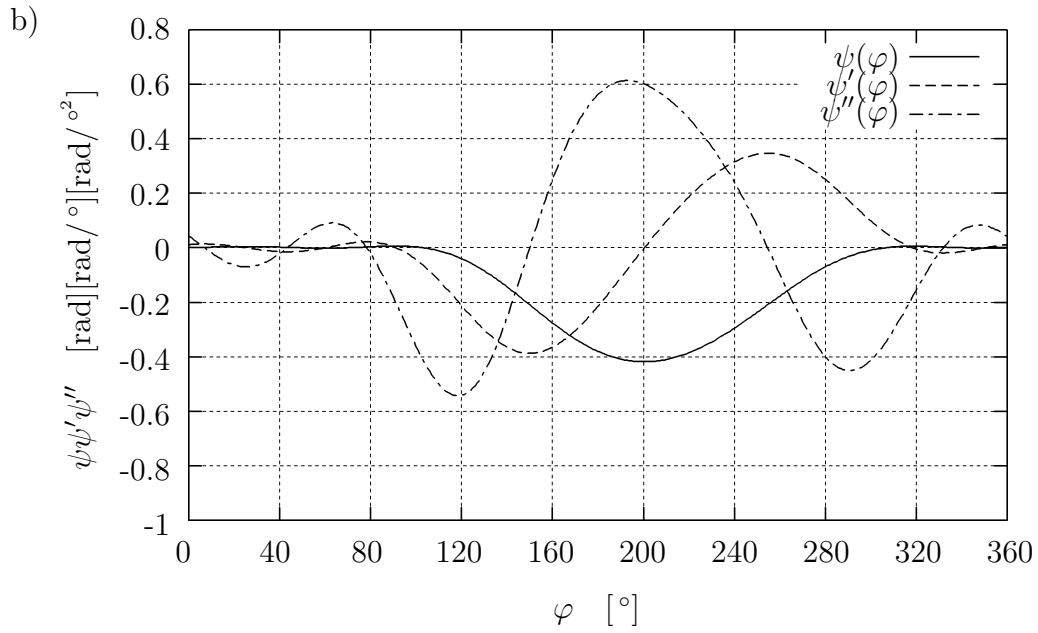


Abbildung 32: Vergleich der Setzhubbewegungsverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungsgesetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom

Die errechnete Rollenmittelpunktskurve und die Arbeitskurven sind in Bild 33 dargestellt.

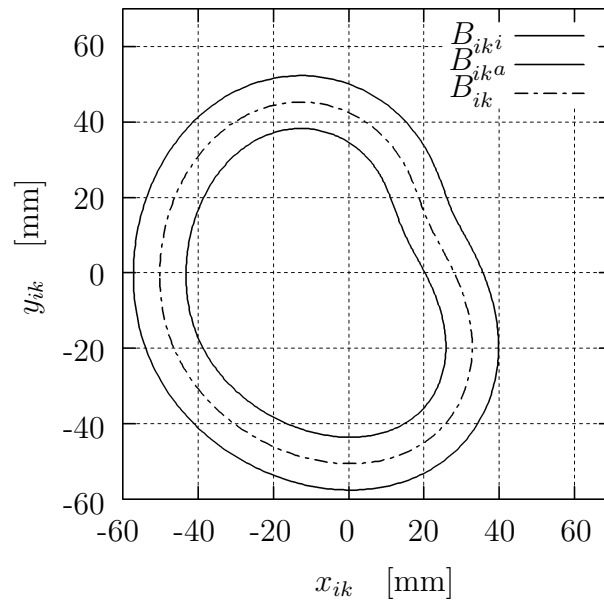


Abbildung 33: schwingungsarme Setzhubkurvenscheibe

4.2.4 Vorschub

Tabelle 6 enthält die Ausgangsdaten für die Vorschubbewegung.

Tabelle 6: Vorgaben für die Vorschubbewegung

Größe	Wert	Größe	Wert
Drehwinkel φ_{H01}	25°	Hub s_H	5 mm
Drehwinkel φ_{H12}	35°	Abweichung $\triangle s$	0.05 mm
Drehwinkel φ_{H23}	70°		
Drehwinkel φ_{H34}	60°		
Drehwinkel φ_{H45}	170°		

Die Rastbreite δ_d ist wieder der Winkel φ_{H12} . Die Verhältnisse $\triangle s/s_H$, δ_d/γ_d und $\nu = \beta_d/\alpha_d$ sind:

$$\left. \begin{aligned} \delta_d &= \varphi_{H12} = 35^\circ \\ \frac{\triangle s}{s_H} &= \frac{0.05}{5} = 0.001 \\ \frac{\delta_d}{\gamma_d} &= \frac{35^\circ}{132.5^\circ} = 0.264151 \\ \alpha_d &= \varphi_{H45} + \varphi_{H01} = 195^\circ \\ \beta_d &= \varphi_{H23} = 70^\circ \\ \nu &= \frac{\beta_d}{\alpha_d} = \frac{70^\circ}{195^\circ} = 0.359 \end{aligned} \right\} \quad (4.25)$$

Die sich daraus ergebende Anzahl der mitzuführenden Harmonischen ist wieder 4. Die Koeffizienten lauten:

$$\left. \begin{aligned} A(0) &= a_0 = 16.9003 \\ A(1) &= a_1 = -25.064 & B(1) &= b_1 = -10.4783 \\ A(2) &= a_2 = 9.45316 & B(2) &= b_2 = 9.16597 \\ A(3) &= a_3 = -1.41296 & B(3) &= b_3 = -2.09477 \\ A(4) &= a_4 = 0.356817 & B(4) &= b_4 = -0.765526 \end{aligned} \right\} \quad (4.26)$$

Der mit diesen Koeffizienten erzeugte Bewegungsverlauf ist in Bild 34 dargestellt.

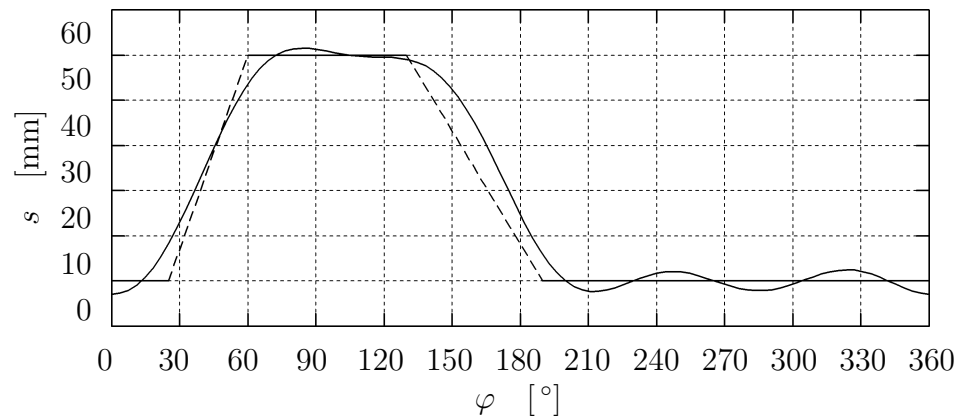
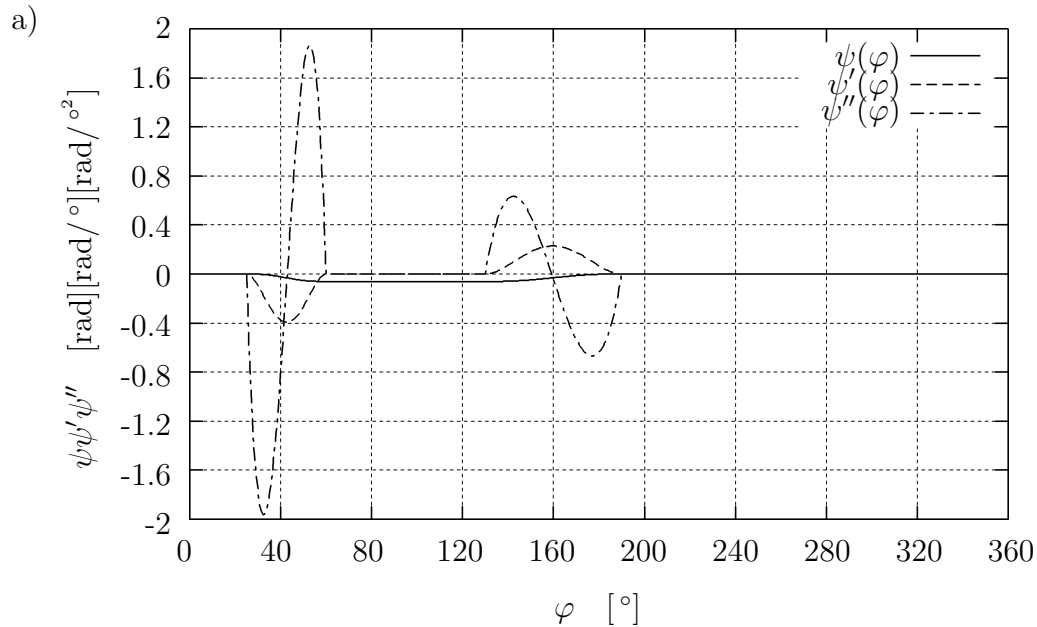


Abbildung 34: Bewegungsverlauf $s(\varphi)$ der Vorschubbewegung

Anhand von Bild 35 lassen sich die Bewegungsgrößen $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ von 3-4-5 Polynom und trigonometrischen Polynom für die Vorschubbewegung vergleichen. Durch den Einsatz des trigonometrischen Polynom können die Beschleunigungen des Abtriebshebels stark reduziert werden.



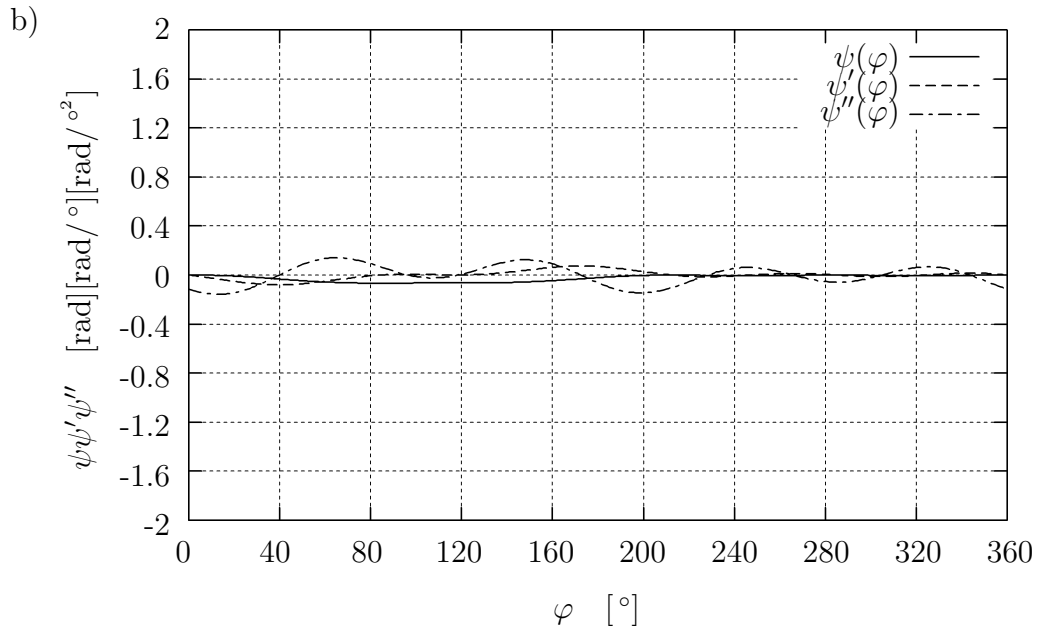


Abbildung 35: Vergleich der Vorschubbewegungsverläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$ bei den Bewegungsgesetzen: a) 3-4-5 Polynom b) trigonometrisches Approximationspolynom

Die errechnete Rollenmittelpunktskurve und die Arbeitskurven sind in Bild 36 dargestellt.

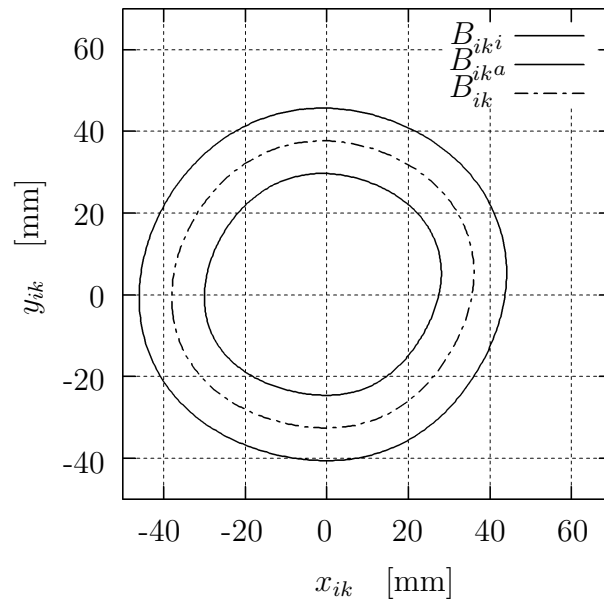


Abbildung 36: schwingungsarme Vorschubkurvenscheibe

5 Anwendungsbereiche der Modelle

Inhalt dieses Abschnittes ist, worin die Vor- und Nachteile der gezeigten Bewegungsgesetze für starre Modelle (Abschn. 3.2) und Modelle mit Elastizitäten im Abtrieb (Abschn. 4.1) bestehen.

Ein in eine Fourierreihe entwickeltes Bewegungsgesetz ist in allen Ableitungen stetig. Bei traditionellen Bewegungsgesetzen treten dagegen in höheren Ableitungen Unstetigkeiten auf. Dynamisch gesehen sind Bewegungsgesetze, die in eine Fourierreihe entwickelt werden günstiger, weil viele Erregerfrequenzen vermieden werden. Dadurch können höhere Arbeitsgeschwindigkeiten gefahren werden. Geräuschemission und Verschleiß sind geringer. Allerdings können unerwünschte Zusatzbewegungen auftreten, die nur durch zusätzliche Stützstellen gemindert oder vermieden werden können. Geradlinige Bewegungsverläufe sind nicht vorgebar. Bei den traditionellen Bewegungsgesetzen gibt es diese Probleme nicht. Jedoch können die geradlinigen Bewegungen wegen der Elastizitäten in den Gelenken bei höheren Drehzahlen nicht mehr eingehalten werden.

Die Fourierkoeffizienten sind schwer zu ermitteln. Die in der Literatur vorgeschlagenen Berechnungsalgorithmen sind oft unzureichend dokumentiert, was die Implementation in eigene Anwendungen erschwert.

Welches Bewegungsgesetz zum Einsatz kommen sollte, ist von Fall zu Fall zu entscheiden. Die wichtigsten Auswahlkriterien sind die Einhaltung des Bewegungsplanes sowie die in Abschn. 2.3.1 vorgestellten kinematischen Kennwerte C_v , C_a , C_j und C_{Mdyn} .

6 Die Anwendung

Opticurv ist ein Programm zur Berechnung der Kurvenscheibenkontur von Kurvenscheiben in kombinierten Getrieben. Es sind Kombinationen von Kurvenscheiben oder Doppelkurvenscheiben mit nachgeschalteten Koppelgetrieben möglich.

6.1 Installation

6.1.1 Windows

Für Windows wurde ein vorkompiliertes Programmsystem erstellt. Es läuft unter Windows 95/98/ME, Windows NT 4, Windows 2000 und Windows XP. Man braucht nur die Datei „`opticurvX.Y.Z.zip`“ in das gewünschte Verzeichnis zu kopieren und dort zu entpacken. Nun tragen Sie noch den Dokumentationspfad in die Datei „`opticurv.conf`“ ein. Z.B.:

```
E:\\Programme\\opticurv\\DOC\\
```

Nun ist das Programm lauffähig. Sie können jetzt die Datei „`opticurv.exe`“ ausführen.

6.1.2 Linux

Kopieren Sie als *root* die Datei „`opticurvX.Y.Z.tgz`“ in das gewünschte Verzeichnis z.B. `/usr/local`. Entpacken Sie die Datei in einer beliebigen Konsole:

```
tar -xvzf opticurvX.Y.Z.tgz
```

Öffnen Sie die Datei „`configure`“ in einem beliebigen Editor und tragen Sie in der Zeile 3 den Installationspfad ihrer Qt-Pakete ein. Z.B.:

```
/usr/local/qt
```

Speichern Sie diese Datei und führen Sie sie in der Konsole aus:

```
/usr/local/opticurv/configure
```

Nun wird das Programm übersetzt. Es kann ein paar Minuten dauern. Tragen Sie nun noch den Dokumentationspfad in die Datei „`opticurv.conf`“ ein. Z.B.:

```
/usr/local/opticurvX.Y.Z/doc
```

Nun ist das Programm lauffähig. Sie können jetzt die Datei „`opticurv`“ ausführen.

6.1.3 Andere Betriebssysteme

Das Programm wurde bisher nur unter oben aufgeführten Betriebssystemen getestet. Der Quellcode kann aber auf folgende Betriebssysteme direkt portiert werden:

- AIX, FreeBSD, HP-UX, IRIX, Solaris, Tru64, UnixWare 7, OpenUnix 8, Mac OS X

Auf dem System müssen nur die von der norwegischen Firma *Trolltech* entwickelte Qt-Klassenbibliothek und ein C++ Kompiler installiert sein.

6.2 Menüeinträge

6.2.1 Datei

Neu

Öffnet ein Dialogfenster zur Eingabe des Dateinamens. Das Programm speichert automatisch die Eingabedaten in dieser Datei.

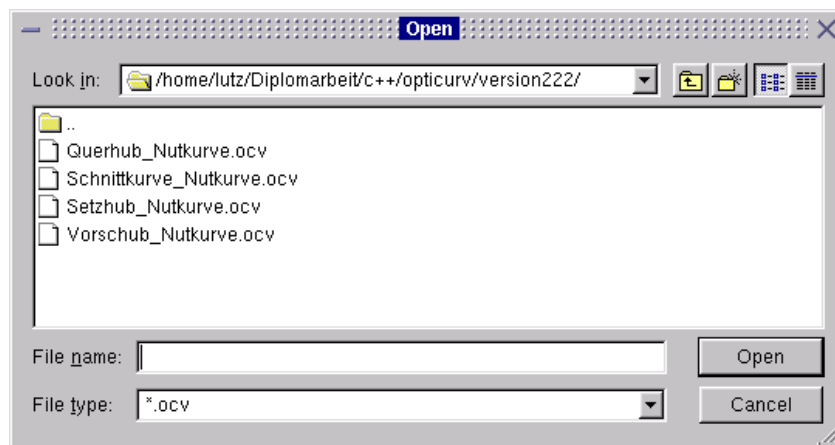


Abbildung 37: Dialogfenster

Nach der Eingabe des Dateinamens erscheint das Eingabeparameterfenster (Abb. 38). Für den Bewegungsverlauf sind die Anzahl der Bewegungsabschnitte, das Bewegungsgesetz, die Gesamtdrehwinkel φ_{Hik} und die Gesamthübe s_{Hik} anzugeben. Die aktuelle Gesamtgradzahl der Gesamtdrehwinkel wird in unterhalb der Maske *Bewegungsverlauf* angezeigt.

Eingabeparameter

Bewegungsverlauf | Geometriedaten | Vorzeichendefinitionen | Skalierung

Anzahl der Bewegungsabschnitte: 5

1. Abschnitt	Rast	phi_H01	[Grad]	70	s_H01	[mm]	0
2. Abschnitt	3-4-5 Polynom	phi_H12	[Grad]	40	s_H12	[mm]	5
3. Abschnitt	Rast	phi_H23	[Grad]	190	s_H23	[mm]	0
4. Abschnitt	3-4-5 Polynom	phi_H34	[Grad]	40	s_H34	[mm]	-5
5. Abschnitt	Rast	phi_H45	[Grad]	20	s_H45	[mm]	0
6. Abschnitt	Rast	phi_H56	[Grad]	1	s_H56	[mm]	1
7. Abschnitt	Rast	phi_H67	[Grad]	1	s_H67	[mm]	1
8. Abschnitt	Rast	phi_H78	[Grad]	1	s_H78	[mm]	1
9. Abschnitt	Rast	phi_H89	[Grad]	1	s_H89	[mm]	1
10. Abschnitt	Rast	phi_H910	[Grad]	1	s_H910	[mm]	1
11. Abschnitt	Rast	phi_H1011	[Grad]	1	s_H1011	[mm]	1
12. Abschnitt	Rast	phi_H1112	[Grad]	1	s_H1112	[mm]	1

Aktuelle Gesamtgradzahl der Gesamtdrehwinkel phi_Hik: 360

Ausführen Abbrechen

Abbildung 38: Definition des Bewegungsverlaufs

Die Größen beziehen sich auf den Bewegungsplan (Abb. 39).

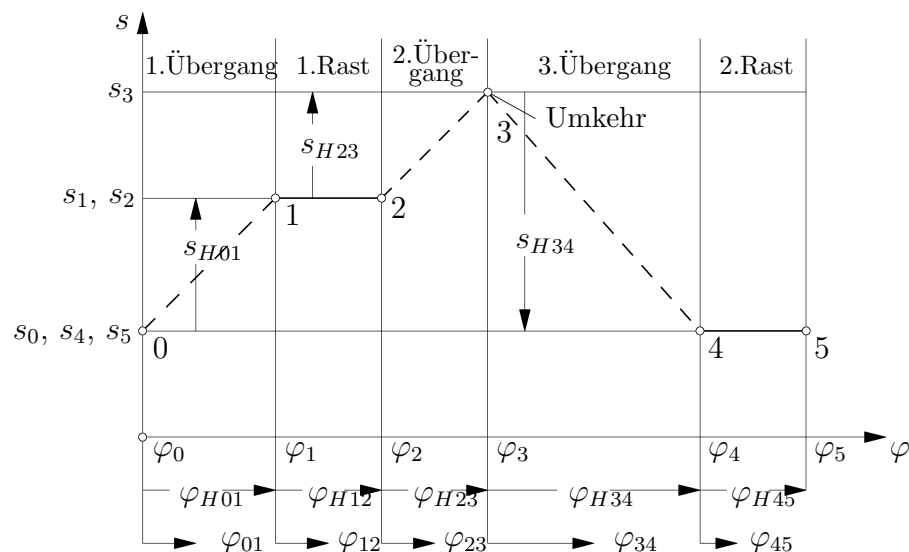


Abbildung 39: Bewegungsplan [14]

In der zweiten Eingabemaske werden die Geometriedaten eingelesen. Abbildung 40 zeigt die definierten Größen.



Neben den in Bild 40 angegebenen Größen werden in der Maske *Geometriedaten* noch die Ausführung der Kurvenscheibe, der Scheibenradius r_S , der Wellenradius r_W und die Schrittweite n des Drehwinkels eingelesen. Die Kurvenscheibe kann als Nutkurvenscheibe oder als Doppelkurvenscheibe ausgeführt werden.

1. Ist die Drehrichtung der Kurvenscheibe mathematisch positiv oder negativ?
2. Sind die Vorzeichen von Koordinate x_{S0} und Hub $s(\varphi)$ zu Beginn der Bewegung gleich?
3. In welchem Quadrant befindet sich der Punkt C_0 ?
4. Ist die Bewegung des Schwinghebels zu Beginn der Bewegung mathematisch positiv oder negativ?

55

Bei der zweiten Frage sind die Vorzeichen gleich, wenn entweder beide positiv oder beide negativ sind. Ungleich sind die Vorzeichen, wenn eines positiv und eines negativ ist. Die Fälle sind in Abbildung 41 dargestellt.

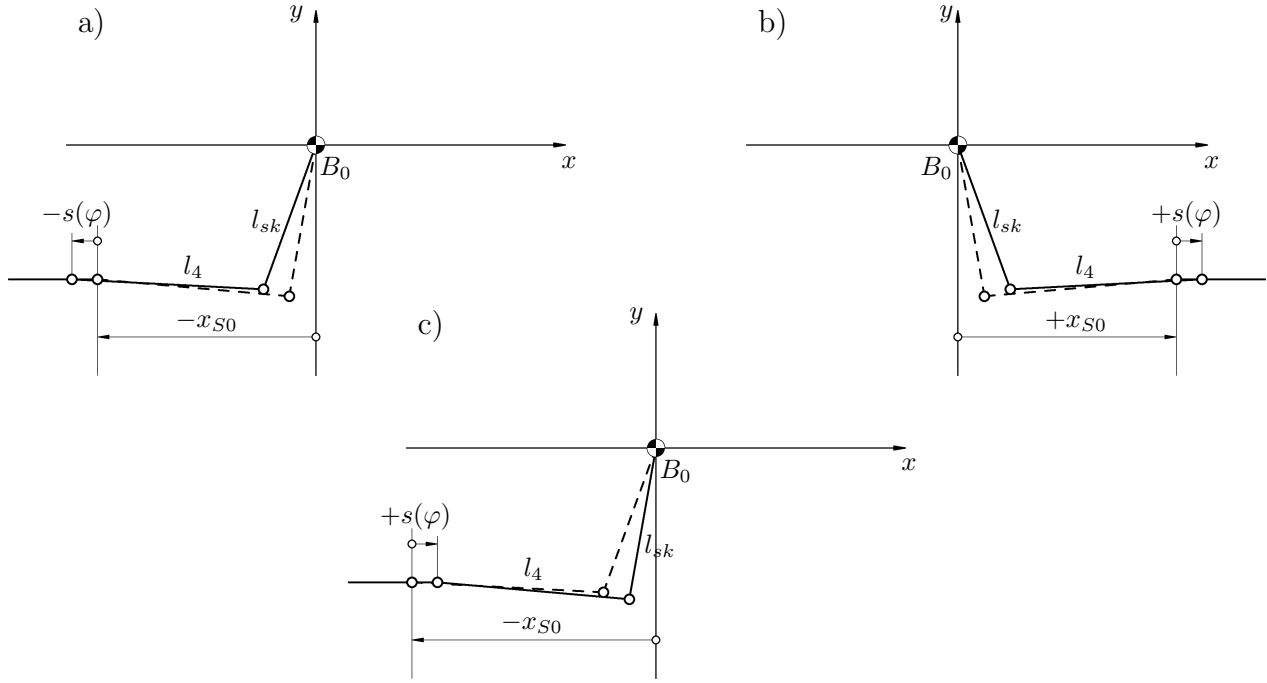


Abbildung 41: Vorzeichenbeispiele a) beide negativ b) beide positiv c) eines positiv und eines negativ

Die Quadranten der 3. Frage werden im mathematisch positiven Drehsinn gezählt (s. Abb. 40).

Die Bewegung des Schwinghebels von Frage 4 ist mathematisch positiv, wenn sich der Schwinghebel (Glied 3 mit der Länge l_3) mathematisch positiv um den Lagerpunkt B_0 dreht. Bei umgekehrter Drehrichtung ist die Bewegung des Schwinghebels mathematisch negativ.

In der vierten Eingabemaske kann die Skalierung der Plotfunktionen eingegeben werden. Im oberen Teil des Fensters können die Funktionen markiert werden, woraufhin diese mit einem Häkchen versehen und die Eingabefelder freigegeben werden. Es sind für die Skalierung die kleinsten Werte des Plotbereichs (x_{min} u. y_{min}), die größten Werte des Plotbereichs (x_{max} u. y_{max}) sowie die Abstände der Skalenwerte (dx u. dy) einzugeben. Ist eine Funktion nicht markiert, dann werden die eingegebenen Werte nicht vom Programm verwendet. In diesem Fall generiert das Programm die Skalierung automatisch.

Nach erfolgter Parametereingabe können die Berechnungen mit dem Button *Ausführen* ausgelöst werden. Daraufhin werden die Ergebnisse im Hauptfenster angezeigt und in ASCII-Dateien gespeichert.

- „Dateiname.ocv“ enthält alle Eingabeparameter
- „Dateiname.asc“ enthält alle von Drehwinkel abhängigen Ergebnisse
- „Dateiname_aussen.asc“ enthält die Koordinaten der äußeren Arbeitskurve
- „Dateiname_innen.asc“ enthält die Koordinaten der inneren Arbeitskurve
- „Dateiname_mitte.asc“ enthält die Koordinaten der Rollenmittelpunktskurve
- „Dateiname_s_phi.asc“ enthält den berechneten Hubverlauf $s(\varphi)$

Mit dem Button *Abbrechen* wird die Eingabe abgebrochen.

Öffnen

Das Menü Öffnen erlaubt dem Nutzer eine Projektdatei „Dateiname.ocv“ zu öffnen und im darauf folgenden Eingabefenster Änderungen an den Parametern vorzunehmen.

Drucken

Öffnet ein Dialogfenster zur Auswahl von Druckereinstellungen. Mit einem Klick auf den OK-Button wird der Druckauftrag gestartet.

Voreinstellungen

Unter *Voreinstellungen* kann der Anwender Einstellungen zur Arbeitsweise des Programms vornehmen. Dies sind bisher nur die Optionen:

- Vor dem Berechnungsstart immer nach dem Dateiname fragen
- Skalierfaktor der zu druckenden Plotfunktion

Wird die erste Option aktiviert, d.h. mit einem Häkchen versehen, dann wird das Dialogfenster zur Eingabe des Dateinamens (Abb. 37) vor dem Ausführen der Berechnungen eingeblendet. Dies ist dann sinnvoll, wenn man für ein Bewegungsgesetz mehrere Versionen von Kurvenscheiben erzeugen will, indem man die Eingabeparameter variiert und diese Varianten unter verschiedenen Versionsnamen speichert.

Die zweite Option ermöglicht dem Nutzer eine manuelle Anpassung der Plotfenstergröße und damit auch der Größe der beim Ausdrucken. Ein Skalierfaktor von 1 entspricht einer Bildgröße von 500×500 Pixeln. Bei Hochauflösenden Druckern kann deshalb eine Skalierung notwendig sein.

Schließen

Beendet das Programm.

6.2.2 Ausgabe

Eingabeparameter

Gibt die bei der zuletzt berechneten Kurvenscheibe verwendeten Eingabeparameter im Hauptfenster aus. Diese können zur Dokumentation auch ausgedruckt werden.

Grafik

Startet einen Funktionsplotter bestehend aus Steuerfenster und Grafikfenster (Abb. 42). Ermöglicht dem Nutzer die grafische Ausgabe der Ergebnisse auf dem Bildschirm.

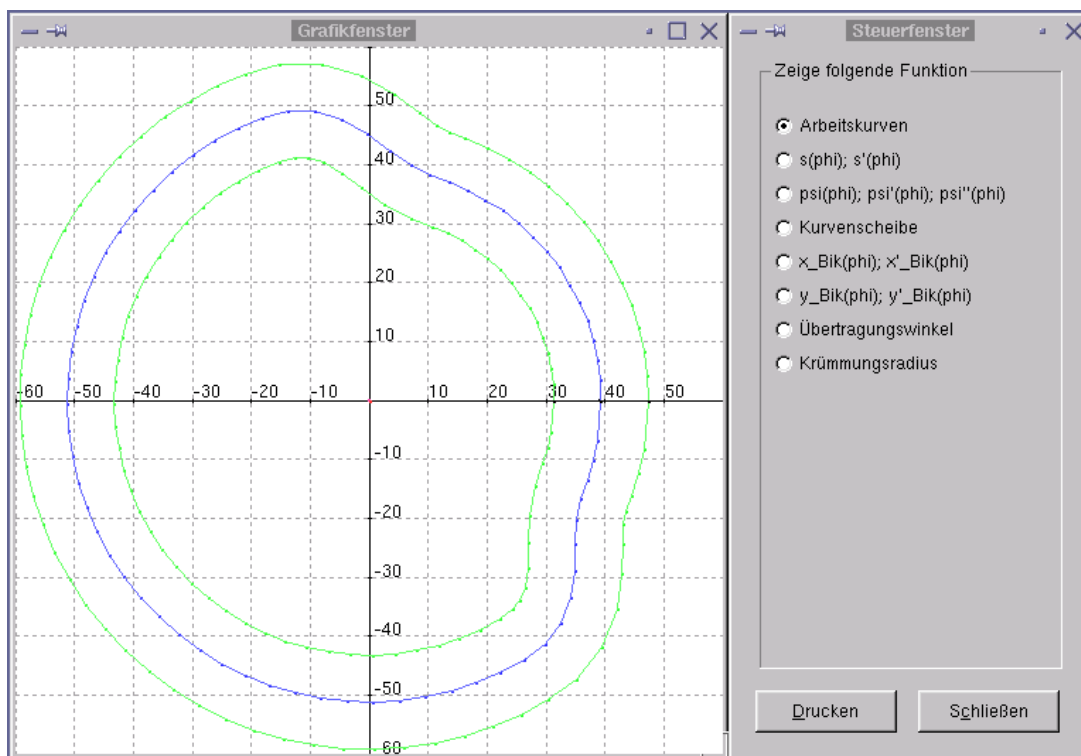


Abbildung 42: Funktionsplotter

Wurde die Kurvenscheibe als Nutkurvenscheibe ausgeführt, dann stehen folgende Plotfunktionen zur Verfügung:

- Arbeitskurven

blau ... Rollenmittelpunktskurve

grün ... innere und äußere Arbeitskurve

- Verläufe $s(\varphi)$ und $s'(\varphi)$

blau ... Hubverlauf $s(\varphi)$

grün ... Verlauf der Hubgeschwindigkeit $s'(\varphi)$

- Verläufe $\psi(\varphi)$, $\psi'(\varphi)$ und $\psi''(\varphi)$

blau ... Abtriebswinkelverlauf $\psi(\varphi)$

grün ... Verlauf der Abtriebswinkelgeschwindigkeit $\psi'(\varphi)$

rot ... Verlauf der Abtriebswinkelbeschleunigung $\psi''(\varphi)$

- Kurvenscheibe

blau ... Rollenmittelpunktskurve

grün ... innere und äußere Arbeitskurve

rot ... Wellendurchmesser und Kurvenscheibendurchmesser

- Verläufe $x_{Bik}(\varphi)$ und $x'_{Bik}(\varphi)$

blau ... Verlauf $x_{Bik}(\varphi)$

grün ... Verlauf $x'_{Bik}(\varphi)$

- Verläufe $y_{Bik}(\varphi)$ und $y'_{Bik}(\varphi)$

blau ... Verlauf $y_{Bik}(\varphi)$

grün ... Verlauf $y'_{Bik}(\varphi)$

- Übertragungswinkel μ

blau ... Verlauf des Übertragungswinkels μ

- Krümmungsradius r_K

blau ... Verlauf des Krümmungsradius r_K

Bei Doppelkurvenscheiben werden die gleichen Darstellungen wie bei den Nutkurvenscheiben verwendet. Lediglich die Darstellung der Arbeitskurven und der Kurvenscheibe wurde anders gewählt. Die Mittelpunktskurven sind ebenfalls verfügbar:

- Kurve und Gegenkurve

blau ... Rollenmittelpunktskurve der Kurve

grün ... Rollenmittelpunktskurve der Gegenkurve

lila ... Außen- und Innenkonturen (Arbeitskurven)

- Kurvenscheibe

blau ... Rollenmittelpunktskurve der Kurve

grün ... Rollenmittelpunktskurve der Gegenkurve

lila ... Außen- und Innenkonturen (Arbeitskurven)

rot ... Wellendurchmesser

- Mittelpunktskurven

blau ... Rollenmittelpunktskurve der Kurve

grün ... Rollenmittelpunktskurve der Gegenkurve

Tabelle

Zeigt im Hauptfenster alle berechneten Ergebnisse in tabellarischer Form an.

6.2.3 Hilfe

Manual

Zeigt die Seiten dieses Manuals an.

Über opticurv

Zeigt wesentliche Informationen zu „opticurv“ an.

6.3 Fragen und Antworten

1. Kann man aus den Zwischenergebnissen Aussagen über das dynamische Verhalten des Getriebes ableiten?

Nein. Dem Programm liegen rein kinematische Betrachtungen zugrunde.

7 Zusammenfassung

Mit dem Programm „opticurv“ sind die Rollenmittelpunktskurve und die Arbeitskurven für Kurvenkoppelgetriebe berechenbar. Im Programm sind die Übertragungsfunktionen 3-4-5 *Polynom* und *Bestehornsinoide* implementiert. Noch nicht im Programm enthalten sind die Übertragungsfunktionen, die in Kapitel 4 vorgestellt wurden. Die Ursache dafür sind die schwer zu ermittelnden Fourierkoeffizienten. In dieser Arbeit konnten sie nur als gute Annäherung an den Bewegungsplan gefunden werden. Jedoch sind auch noch das Abstimmungsverhältnis oder die Vermeidung von bestimmten Harmonischen wesentliche zu optimierende Variablen. Es ist also sehr wichtig, die Eigenfrequenzen des Systems zu kennen, wenn man dieses mehrdimensionale Optimierungsproblem lösen möchte.

Das weitere Vorgehen könnte wie folgt aussehen:

- Messung der Eigenfrequenzen
- Erweiterung des Programms um einen Optimierungsalgorithmus zur Berechnung der Fourierkoeffizienten
- Erweiterung des Programms um das dynamische Modell
- Berechnung des Gesamtsystems (Querhub-, Schnitt-, Setzhub- und Vorschubkurvenscheibe) im Programm ermöglichen, da sich die Kurvenscheiben gegenseitig beeinflussen und die Bewegungen besser überschaubar sind
- Verfeinerung des dynamischen Modells (z.B. Berücksichtigung von Gelenkspiel oder Elastizitäten im Antrieb)

Auf dem Gebiet der dynamischen Auslegung sind also die Möglichkeiten bei weitem noch nicht ausgeschöpft. Diese Arbeit bietet jedoch einen Einstieg in diese Thematik.

Literatur

- [1] Cărabăş, Iosif; Mesaroş-Anghel, Voicu; Lovasz, Erwin-Christian: *Proiectarea mecanismelor*. 1.Aufl., Mirton, 2000
- [2] Dankert, Jürgen: *Numerische Methoden der Mechanik*. 1.Aufl., Fachbuchverlag Leipzig, 1977
- [3] Dresig, Hans: *Forschungsergebnisse zur Dynamik von Kurvengetrieben*. TH Karl-Marx-Stadt, 1986, Wissenschaftliche Zeitschrift der TH Karl-Marx-Stadt 33, 34, 37, 38
- [4] Dresig, Hans: *Schwingungen mechanischer Antriebssysteme; Modellbildung, Berechnung, Analyse, Synthese*. 1.Aufl., Springer, 2001
- [5] Dresig, Hans; Vul'fson, I.I.: *Dynamik der Mechanismen*. 1. Aufl., Deutscher Verlag der Wissenschaften, 1989
- [6] Göldner, Hans; Holzweißig, Franz: *Leitfaden der Technischen Mechanik*. 11. Aufl., Fachbuchverlag Leipzig, 1989
- [7] Herold, Helmut: *Das Qt Buch; Portable GUI-Programmierung unter Linux/UNIX/Windows*. 1.Aufl., SuSE, 2001
- [8] Herold, Helmut: *Linux-Unix-Profitools; awk, sed, lex, yacc und make*. 3.Aufl., Addison-Wesley, 1999
- [9] Luck, Kurt; Modler, Karl-Heinz: *Getriebetechnik; Analyse, Synthese, Optimierung*. 2.Aufl., Springer, 1995 9, 12, 17, 18, 20, 25, 26
- [10] Lüder, Reinhard: *Zur Synthese periodischer Bewegungsgesetze von Mechanismen unter Berücksichtigung von Elastizität und Spiel*. Fortschritt-Berichte VDI Reihe 11 Nr. 225, VDI-Verlag, 1995
- [11] Obst, Peter; Heydt, Wolfgang: *Konstruktion und Fertigung von Kurvenmechanismen*. 1.Aufl., Verlag Technik, 1964 11
- [12] Rößler, Jürgen: *Dynamik von Mechanismen-Antriebssystemen im Textil- und Verarbeitungsmaschinenbau*. Diss., TH Karl-Marx-Stadt, 1985
- [13] VDI 2142 Blatt 1 *Auslegung ebener Kurvengetriebe; Grundlagen, Profilberechnung und Konstruktion*
- [14] VDI 2143 Blatt 1 *Bewegungsgesetze für Kurvengetriebe; Theoretische Grundlagen* 3, 5, 6, 7, 8, 54
- [15] Volmer, Johannes: *Getriebetechnik; Grundlagen*. 2.Aufl., Verlag Technik, 1995
- [16] Volmer, Johannes: *Getriebetechnik; Kurvengetriebe*. 2.Aufl., Verlag Technik, 1989 32
- [17] Willms, Gerhard: *C++; Das Grundlagenbuch*. 2.Aufl., DATA BECKER, 2001

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur angefertigt habe.

Dresden, den 01.12.2003

Stichwortverzeichnis

A_0 -Bereiche	
Ermittlung,	19
Abstimmungsverhältnis,	36, 37
Arbeitskurven	
Gegenkurve,	29
Kurve,	26
Beispiele	
Querhub,	68
Schnittkurve,	70
Setzhub,	72
Vorschub,	74
Beschleunigungskennwert,	10
<i>Bestehornsinoide</i> ,	21
Bewegungs-	
~aufgaben,	6
~diagramm,	5
Bewegungsaufgaben	
Kombination,	7
Bewegungsgesetze,	3
3-4-5 Polynom,	10, 12
normierte,	8
symmetrische,	9
unsymmetrische,	11
Bewegungsgleichung,	34
Bewegungsplan,	5
Bewegungsverlauf	
Kurvenkoppelgetriebe,	21
darstellende Größe,	19
Differentialgleichung,	34
Differenzenformeln	
zentrale,	24
Doppelkurvenscheiben,	15
Dynamisches Verhalten,	33
Eigenkreisfrequenz,	34
Einmassenmodell,	34
Erklärung	
eidesstattliche,	64
<i>Eulersche Formel</i> ,	22, 25
F-Kurvengetriebe,	18
Gegenlauf,	19
Gesamtdrehwinkel,	5
Gesamtweg,	5
Geschwindigkeitskennwert,	10
Gleichlauf,	19
Grundkreisradius,	28, 29
Grundkreiswinkel,	28, 29
Installation,	52
Kennwerte	
kinematische,	9
Kettenregel,	3
Koeffizienten,	35
Krümmungsradius,	31
Kurvenkoppelgetriebe,	1
Kurvenprofil	
Gegenkurve,	29
Kurve,	26
<i>Lehrschen Dämpfungsmaß</i> ,	34
Literatur,	63
Mathematica-Dateien	
Querhub_dyn.nb,	76
Schnitt_dyn.nb,	80
Setzhub_dyn.nb,	84
Vorschub_dyn.nb,	88
Modellbildung,	34
Momentenkennwert	
dynamischer \sim ,	10
statischer \sim ,	10
normierter Drehwinkel,	8
normierter Weg,	8
P-Kurvengetriebe,	18
Plotfunktionen	
Doppelkurvenscheibe,	60
Nutkurvenscheibe,	59
<i>3-4-5 Polynom</i> ,	21
Produktregel,	4
Quellcode,	89

functionplot.h,	89
opticurv273.cpp,	125
Resonanz,	36
Rollenmittelpunktskurve	
Gegenkurve,	28
Kurve,	24
Skalarprodukt,	30
Spitzenbildung,	31
Übertragungsfunktionen	
Kurvenkoppelgetriebe,	21
Übertragungsgetriebe,	1
Übertragungswinkel,	17, 30
Unterschnitt,	31
Verfahren nach <i>Flocke</i> ,	19
Vorzeichendefinitionen,	55

Anhang

A Beispiele für Doppelkurvenscheiben

A.1 Querhub

EINGABEPARAMETER

Opticurv Version 2.7.3 - TU DRESDEN

Die Nov 4 2003 - 15:14:41

/home/lutz/Diplomarbeit/c++/opticurv/version273/Querhub.ocv

Bewegungsverlauf

Anzahl der Bewegungsabschnitte: 5

1. Abschnitt	: Rast	phi_H01 = 70	s_H01 = 0
2. Abschnitt	: 3-4-5 Polynom	phi_H12 = 40	s_H12 = 5
3. Abschnitt	: Rast	phi_H23 = 190	s_H23 = 0
4. Abschnitt	: 3-4-5 Polynom	phi_H34 = 40	s_H34 = -5
5. Abschnitt	: Rast	phi_H45 = 20	s_H45 = 0

Geometriedaten

x_A0 = 290	r_G = 39.334	alpha_R = -17
y_A0 = 70	r_R = 8	L3stern = 297.29
x_S0 = 88.9011	r_S = 65	r_Rstern = 8
y_s0 = 124.829	r_W = 12.5	
L3 = 297.29	n = 0.5	
Lsk = 124.89		
L4 = 85		

Vorzeichendefinitionen

- Ist die Drehrichtung der Kurvenscheibe mathematisch positiv oder negativ?
-> mathematisch positiv
- Sind die Hubrichtungen von Koordinate x_S0 und Hub s(phi) zu Beginn der Bewegung gleich gerichtet?
-> Nein

- In welchem Quadrant befindet sich der Punkt C_a?
-> 1. bzw. 4. Quadrant
- Ist die Bewegung des Schwinghebels zu Beginn der Bewegung mathematisch positiv oder negativ?
-> mathematisch negativ

ERGEBNISSE

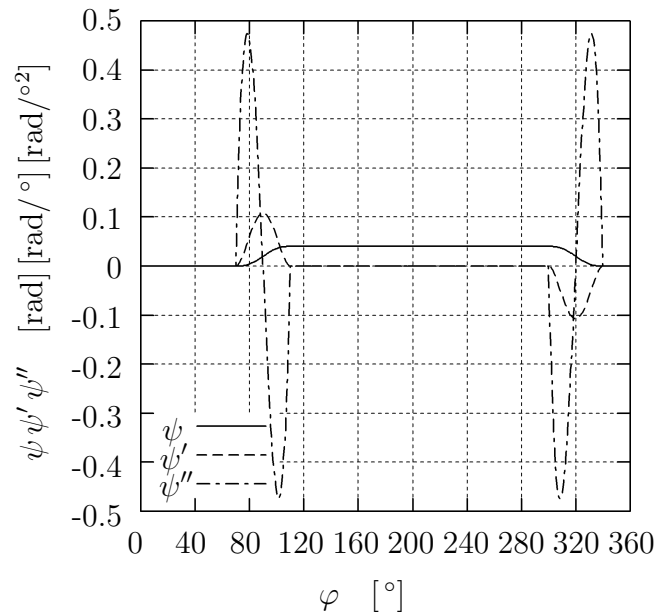


Abbildung 43: Verläufe ψ , ψ' und ψ'' beim Querhub

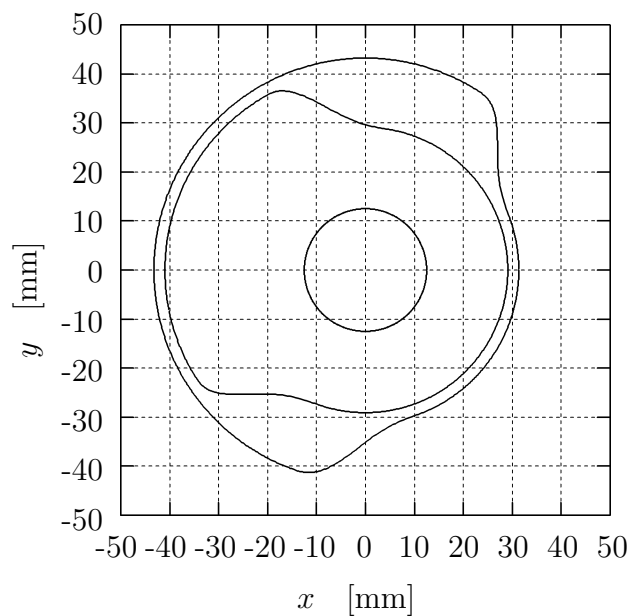


Abbildung 44: Doppelkurvenscheibe für die Querhubbewegung

A.2 Schnittkurve

EINGABEPARAMETER

Opticurv Version 2.7.3 - TU DRESDEN

Die Nov 4 2003 - 15:25:11

/home/lutz/Diplomarbeit/c++/opticurv/version273/Schnittkurve.ocv

Bewegungsverlauf

Anzahl der Bewegungsabschnitte: 5

1. Abschnitt	: Rast	phi_H01 = 10	s_H01 = 0
2. Abschnitt	: 3-4-5 Polynom	phi_H12 = 40	s_H12 = 5.5
3. Abschnitt	: Rast	phi_H23 = 0	s_H23 = 0
4. Abschnitt	: 3-4-5 Polynom	phi_H34 = 30	s_H34 = -5.5
5. Abschnitt	: Rast	phi_H45 = 280	s_H45 = 0

Geometriedaten

x_A0 = 34.81	r_G = 47.654	alpha_R = -70
y_A0 = 84.39	r_R = 9.5	L3stern = 84.9
x_S0 = 70.8912	r_S = 70	r_Rstern = 9.5
y_s0 = 41.7464	r_W = 12.5	
L3 = 84.9	n = 0.5	
Lsk = 42		
L4 = 75.5		

Vorzeichendefinitionen

- Ist die Drehrichtung der Kurvenscheibe mathematisch positiv oder negativ?
-> mathematisch positiv
- Sind die Hubrichtungen von Koordinate x_S0 und Hub s(phi) zu Beginn der Bewegung gleich gerichtet?
-> Ja
- In welchem Quadrant befindet sich der Punkt C_a?
-> 1. bzw. 4. Quadrant

- Ist die Bewegung des Schwinghebels zu Beginn der Bewegung mathematisch positiv oder negativ?
-> mathematisch positiv

ERGEBNISSE

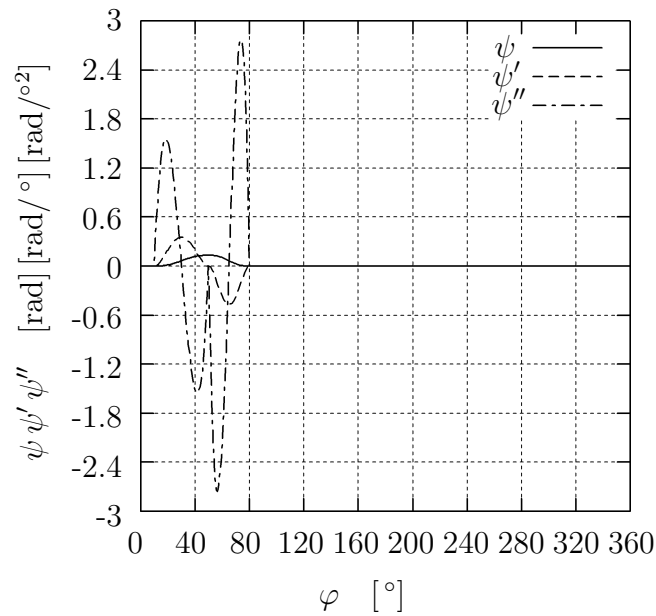


Abbildung 45: Verläufe ψ , ψ' und ψ'' bei der Schnittkurvenscheibe

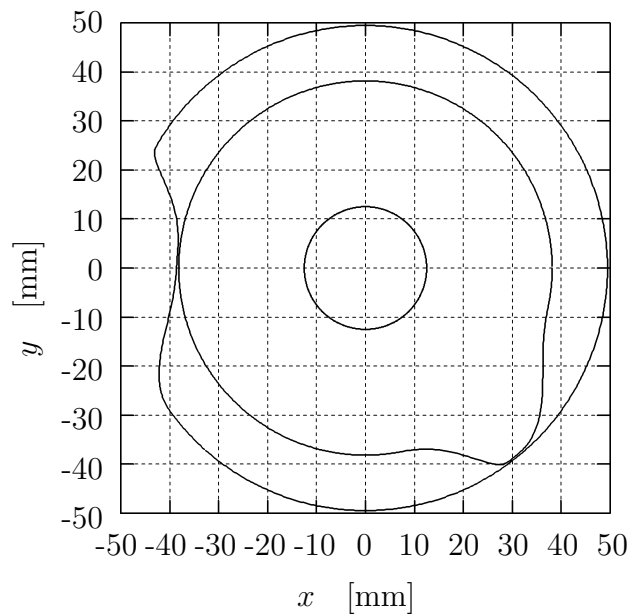


Abbildung 46: Doppelkurvenscheibe für die Schnittbewegung

A.3 Setzhub

EINGABEPARAMETER

Opticurv Version 2.7.3 - TU DRESDEN

Die Nov 4 2003 - 19:45:52

/home/lutz/Diplomarbeit/c++/opticurv/version273/Setzhub.ocv

Bewegungsverlauf

Anzahl der Bewegungsabschnitte: 5

1. Abschnitt	: Rast	phi_H01 = 110	s_H01 = 0
2. Abschnitt	: 3-4-5 Polynom	phi_H12 = 80	s_H12 = 58.5
3. Abschnitt	: Rast	phi_H23 = 20	s_H23 = 0
4. Abschnitt	: 3-4-5 Polynom	phi_H34 = 90	s_H34 = -58.5
5. Abschnitt	: Rast	phi_H45 = 60	s_H45 = 0

Geometriedaten

x_A0 = 40.46	r_G = 50.199	alpha_R = -80
y_A0 = 62	r_R = 7	L3stern = 61.29
x_S0 = -31.5467	r_S = 60	r_Rstern = 7
y_s0 = 131.986	r_W = 12.5	
L3 = 61.29	n = 0.5	
Lsk = 160.5		
L4 = 122.87		

Vorzeichendefinitionen

- Ist die Drehrichtung der Kurvenscheibe mathematisch positiv oder negativ?
-> mathematisch positiv
- Sind die Hubrichtungen von Koordinate x_S0 und Hub s(phi) zu Beginn der Bewegung gleich gerichtet?
-> Nein
- In welchem Quadrant befindet sich der Punkt C_a?
-> 2. bzw. 3. Quadrant

- Ist die Bewegung des Schwinghebels zu Beginn der Bewegung mathematisch positiv oder negativ?
-> mathematisch positiv

ERGEBNISSE

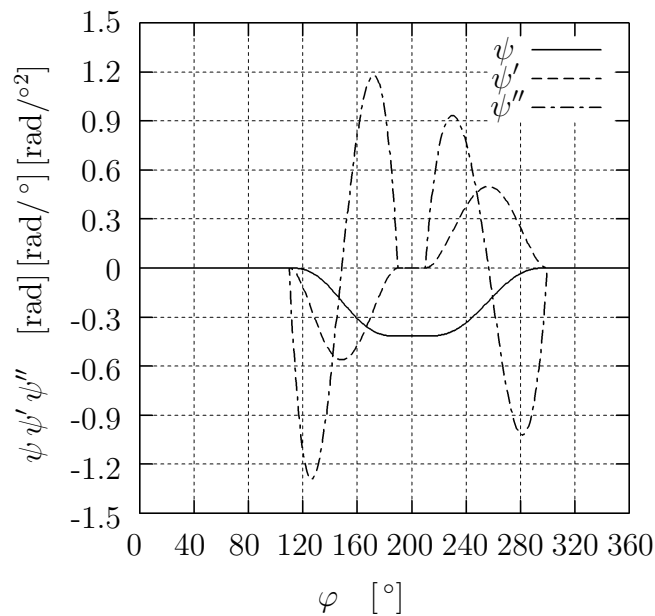


Abbildung 47: Verläufe ψ , ψ' und ψ'' beim Setzhub

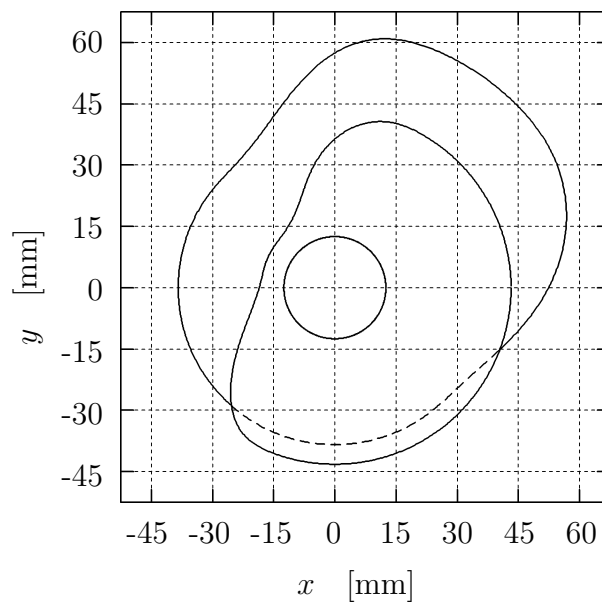


Abbildung 48: Doppelkurvenscheibe für die Setzhubbewegung

A.4 Vorschub

EINGABEPARAMETER

Opticurv Version 2.7.3 - TU DRESDEN

Die Nov 4 2003 - 19:51:56

/home/lutz/Diplomarbeit/c++/opticurv/version273/Vorschub.ocv

Bewegungsverlauf

Anzahl der Bewegungsabschnitte: 5

1. Abschnitt	: Rast	phi_H01 = 25	s_H01 = 0
2. Abschnitt	: 3-4-5 Polynom	phi_H12 = 35	s_H12 = 5
3. Abschnitt	: Rast	phi_H23 = 70	s_H23 = 0
4. Abschnitt	: 3-4-5 Polynom	phi_H34 = 60	s_H34 = -5
5. Abschnitt	: Rast	phi_H45 = 170	s_H45 = 0

Geometriedaten

x_A0 = 29.76	r_G = 38.006	alpha_R = -48
y_A0 = 85.37	r_R = 8	L3stern = 82.03
x_S0 = 86.0617	r_S = 50	r_Rstern = 8
y_s0 = 81.6329	r_W = 12.5	
L3 = 82.03	n = 0.5	
Lsk = 82.03		
L4 = 78		

Vorzeichendefinitionen

- Ist die Drehrichtung der Kurvenscheibe mathematisch positiv oder negativ?
-> mathematisch positiv
- Sind die Hubrichtungen von Koordinate x_S0 und Hub s(phi) zu Beginn der Bewegung gleich gerichtet?
-> Nein
- In welchem Quadrant befindet sich der Punkt C_a?
-> 1. bzw. 4. Quadrant

- Ist die Bewegung des Schwinghebels zu Beginn der Bewegung mathematisch positiv oder negativ?
-> mathematisch positiv

ERGEBNISSE

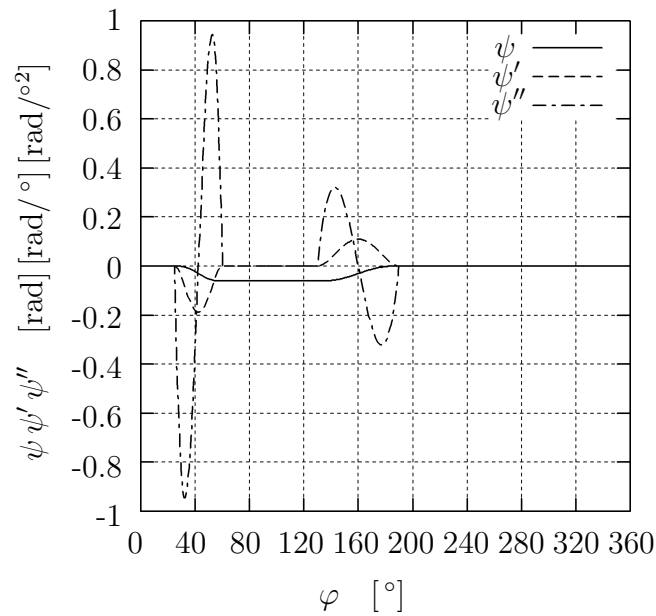


Abbildung 49: Verläufe ψ , ψ' und ψ'' beim Vorschub

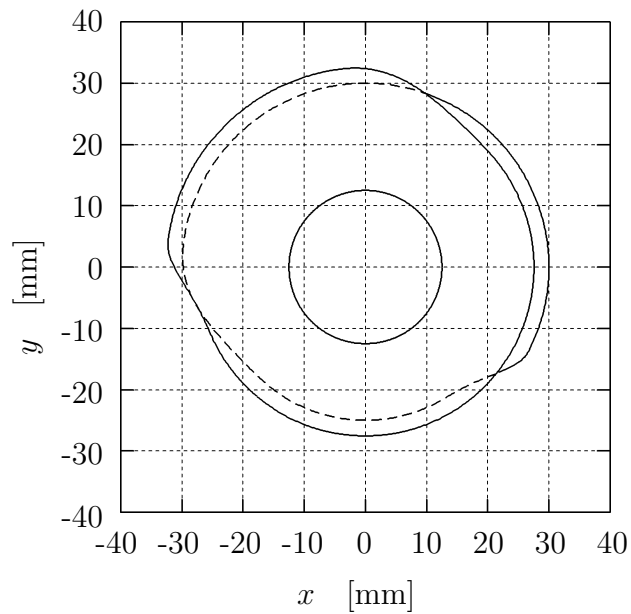


Abbildung 50: Doppelkurvenscheibe für die Vorschubbewegung

B Mathematica-Dateien

B.1 Querhub_dyn.nb

Übertragungsfunktionen

```
In[1]:= a0 := 3.14895

In[2]:= a1 := -2.5767

In[3]:= b1 := -1.20153

In[4]:= a2 := -0.664103

In[5]:= b2 := -0.791447

In[6]:= a3 := 0.073979

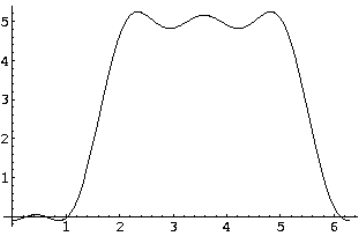
In[7]:= b3 := 0.276093

In[8]:= a4 := -0.0848508

In[9]:= b4 := 0.481213

In[10]:= x[phi_] := a0 + a1 * cos[phi] + b1 * sin[phi] + a2 * cos[2 * phi] + b2 * sin[2 * phi] +
          a3 * cos[3 * phi] + b3 * sin[3 * phi] + a4 * cos[4 * phi] + b4 * sin[4 * phi]

In[11]:= Plot[x[phi], {phi, 0°, 360°}]
```



```
Out[11]= - Graphics -

In[12]:= xs0 := 88.901

In[13]:= l3 := 297.29

In[14]:= l4 := 85

In[15]:= lsk := 124.89

In[16]:= ys0 := 124.829

In[17]:= AQ[phi_] := 2 * lsk * (xs0 - x[phi])

In[18]:= BQ[phi_] := -2 * lsk * ys0

In[19]:= CQ[phi_] := (xs0 - x[phi])^2 + ys0^2 - l4^2 + lsk^2
```

```

In[20]:= psisk[phi_] := 2 * arctan  $\left[ \frac{\left( BQ[\phi] + \sqrt{AQ[\phi]^2 + BQ[\phi]^2 - CQ[\phi]^2} \right)}{(AQ[\phi] - CQ[\phi])} \right]$ 

In[21]:=  $\psi$ [phi_] := -psisk[phi] + psisk[0°]

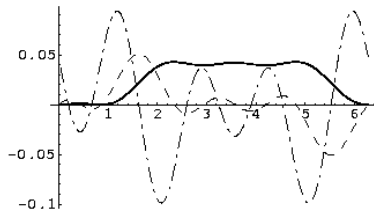
In[22]:= deltaphi := 0.001°

In[23]:= psi1[phi_] := ( $\psi$ [(phi + deltaphi)] -  $\psi$ [(phi - deltaphi)])/(2 * deltaphi)

In[24]:= psi2[phi_] :=
    ( $\psi$ [(phi + deltaphi)] - 2 *  $\psi$ [phi] +
      $\psi$ [(phi - deltaphi)])/(deltaphi ^2)

In[25]:= Plot[{ $\psi$ [phi], psi1[phi], psi2[phi]}, {phi, 0°, 360°}, PlotStyle -> {Thickness[0.005],
    Dashing[{0.03}], Dashing[{0.01, 0.02, 0.05, 0.02}]}]

```



```

Out[25]= -Graphics-

In[26]:= listel := Table[{phi,  $\psi$ [phi], psi1[phi], psi2[phi]}, {phi, 0°, 360°, 1°}]

In[27]:= Export[listelQ.txt``, listel, List``]

Out[27]= listelQ.txt

```

Koordinaten der Rollenmittelpunktsbahn

```

In[28]:= xA0 := 290

In[29]:= yA0 := 70

In[30]:= l1 := Sqrt[xA0 ^2 + yA0 ^2]

In[31]:= rG := 39.334

In[32]:= psiG :=
    arccos[(l1 ^2 + l3 ^2 - rG ^2)/(2 * l1 * l3)]

In[33]:= MQ[phi_] := (1 - cos[phi] + (l3 / l1) * cos[psiG +  $\psi$ [phi] + phi])

```



```

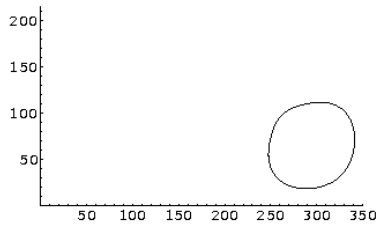
In[34]:= NQ[phi_] := (-sin[phi] + (13/11) * sin[psiG+psi[phi] + phi])

In[35]:= xBik[phi_] := (xA0 * MQ[phi] - yA0 * NQ[phi])

In[36]:= yBik[phi_] := (xA0 * NQ[phi] + yA0 * MQ[phi])

In[37]:= ParametricPlot[{xBik[phi], yBik[phi]}, {phi, 0°, 360°}, PlotRange -> {{0, 350}, {0, 215}}]

```



```
Out[37]= - Graphics -
```

```

In[38]:= xBik1[phi_] := (xA0 * sin[phi] + yA0 * cos[phi] - (13/11) * (psi1[phi] + 1) *
    (xA0 * sin[psiG+psi[phi] + phi] + yA0 * cos[psiG+psi[phi] + phi]))

In[39]:= yBik1[phi_] := (xA0 * cos[phi] - yA0 * sin[phi] - (13/11) * (psi1[phi] + 1) *
    (xA0 * cos[psiG+psi[phi] + phi] - yA0 * sin[psiG+psi[phi] + phi]))

```

```

In[40]:= liste2 := Table[{phi, xBik[phi], yBik[phi], xBik1[phi], yBik1[phi]}, {phi, 0°, 360°, 1°}]

In[41]:= Export[liste2Q.txt``, liste2, List``]

Out[41]= liste2Q.txt

```

Arbeitskurven

```

In[42]:= rR := 8

In[43]:= BetragB1[phi_] := Sqrt[xBik1[phi]^2 + yBik1[phi]^2]

In[44]:= xv[phi_] := xBik[phi] - xA0

In[45]:= yv[phi_] := yBik[phi] - yA0

In[46]:= xi[phi_] := xBik[phi] + rR * (yBik1[phi] / BetragB1[phi])

In[47]:= yi[phi_] := yBik[phi] + rR * (xBik1[phi] / BetragB1[phi])

In[48]:= xiv[phi_] := xi[phi] - xA0

In[49]:= yiv[phi_] := yi[phi] - yA0

In[50]:= xa[phi_] := xBik[phi] - rR * (yBik1[phi] / BetragB1[phi])

```

```

In[51]:= ya[phi_] := yBik[phi] - rR*(xBik1[phi]/BetragB1[phi])

In[52]:= xav[phi_] := xa[phi] - xA0

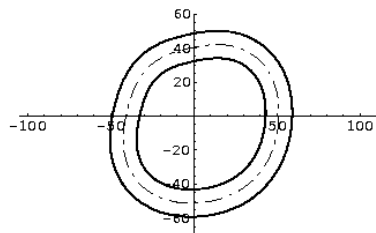
In[53]:= yav[phi_] := ya[phi] - yA0

In[54]:= ParametricPlot[{xv[phi], yv[phi]}, {xiv[phi], yiv[phi]}, {xav[phi], yav[phi]}, {phi, 0°, 360°},

    PlotStyle -> {Dashing[{0.01, 0.02, 0.05, 0.02}], Thickness[0.005], Thickness[0.005]},

    PlotRange -> {{-105, 110}, {-70, 60}}]

```



Out[54]= -Graphics-

```

In[55]:= liste3 := Table[{phi, xv[phi], yv[phi], xiv[phi], yiv[phi], xav[phi], yav[phi]}, {phi, 0°, 360°, 1°}]

```

```

In[56]:= Export[liste3Q.txt, liste3, List]

```

Out[56]= liste3Q.txt

B.2 Schnitt_dyn.nb

Übertragungsfunktionen

```
In[57]:= a0 := 0.815935
```

```
In[58]:= a1 := 1.09325
```

```
In[59]:= b1 := 1.11299
```

```
In[60]:= a2 := -0.0259486
```

```
In[61]:= b2 := 1.35807
```

```
In[62]:= a3 := -0.776141
```

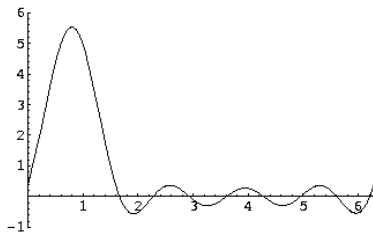
```
In[63]:= b3 := 0.727385
```

```
In[64]:= a4 := -0.72753
```

```
In[65]:= b4 := -0.0388622
```

```
In[66]:= x[phi_] := a0 + a1 * cos[phi] + b1 * sin[phi] + a2 * cos[2 * phi] + b2 * sin[2 * phi] +  
a3 * cos[3 * phi] + b3 * sin[3 * phi] + a4 * cos[4 * phi] + b4 * sin[4 * phi]
```

```
In[67]:= Plot[x[phi], {phi, 0°, 360°}, PlotRange -> {{0°, 360°}, {-1, 6}}]
```



```
Out[67]= -Graphics-
```

```
In[68]:= liste4 := Table[{phi, x[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[69]:= Export["liste4Sc.txt", liste4, "List"]
```

```
Out[69]= liste4Sc.txt
```

```
In[70]:= xs0 := 70.8912
```

```
In[71]:= l3 := 84.9
```

```
In[72]:= l4 := 75.5
```

```
In[73]:= lsk := 42
```

```
In[74]:= ys0 := 41.7464
```

```
In[75]:= ASc[phi_] := 2 * lsk * (xs0 + x[phi])
```

```

In[76]:= BSc[phi_] := -2 * lsk * ys0

In[77]:= CSc[phi_] := (xs0 + x[phi])^2 + ys0^2 - l4^2 + lsk^2

In[78]:= psisk[phi_] := 2 * arctan [ ( BSc[phi] + sqrt(Asc[phi]^2 + BSc[phi]^2 - CSc[phi]^2) ) /
                                     (Asc[phi] - CSc[phi]) ]

In[79]:= psi[phi_] := psisk[phi] - psisk[0°]

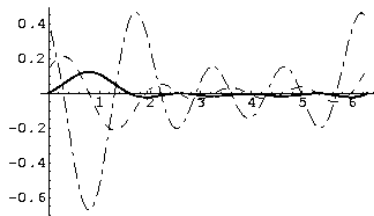
In[80]:= deltaphi := 0.001°

In[81]:= psil[phi_] := (psi[(phi + deltaphi)] - psi[(phi - deltaphi)]) / (2 * deltaphi)

In[82]:= psi2[phi_] := (psi[(phi + deltaphi)] - 2 * psi[phi] + psi[phi - deltaphi]) / (deltaphi^2)

In[83]:= Plot[{psi[phi], psil[phi], psi2[phi]}, {phi, 0°, 360°}, PlotStyle -> {Thickness[0.005],
                                     Dashing[{0.03}], Dashing[{0.01, 0.02, 0.05, 0.02}]}]

```



```
Out[83]= -Graphics-
```

```

In[84]:= listel := Table[{phi, psi[phi], psil[phi], psi2[phi]}, {phi, 0°, 360°, 1°}]

In[85]:= Export["listelSc.txt", listel, "List"]

Out[85]= listelSc.txt

```

Koordinaten der Rollenmittelpunktsbahn

```

In[86]:= xA0 := 34.81

In[87]:= yA0 := 84.39

In[88]:= l1 := Sqrt[xA0^2 + yA0^2]

In[89]:= rG := 47.654

In[90]:= psiG := arccos[(l1^2 + l3^2 - rG^2) / (2 * l1 * l3)]

In[91]:= MSc[phi_] := (1 - cos[phi] + (l3 / l1) * cos[psiG + psi[phi] + phi])

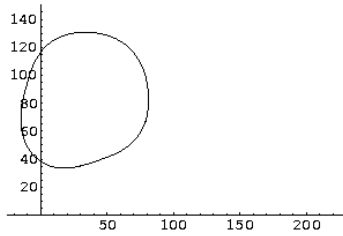
In[92]:= NSc[phi_] := (-sin[phi] + (l3 / l1) * sin[psiG + psi[phi] + phi])

```

```
In[93]:= xBik[phi_] := (xA0 * MSc[phi] - yA0 * NSc[phi])
```

```
In[94]:= yBik[phi_] := (xA0 * NSc[phi] + yA0 * MSc[phi])
```

```
In[95]:= ParametricPlot[{xBik[phi], yBik[phi]}, {phi, 0°, 360°}, PlotRange -> {{-25, 230}, {0, 150}}]
```



```
Out[95]= -Graphics-
```

```
In[96]:= xBik1[phi_] := (xA0 * sin[phi] + yA0 * cos[phi] - (13 / 11) * (psi1[phi] + 1) *
```

```
(xA0 * sin[psiG + psi[phi] + phi] + yA0 * cos[psiG + psi[phi] + phi]))
```

```
In[97]:= yBik1[phi_] := (xA0 * cos[phi] - yA0 * sin[phi] - (13 / 11) * (psi1[phi] + 1) *
```

```
(xA0 * cos[psiG + psi[phi] + phi] - yA0 * sin[psiG + psi[phi] + phi]))
```

```
In[98]:= liste2 := Table[{phi, xBik[phi], yBik[phi], xBik1[phi], yBik1[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[99]:= Export["liste2Sc.txt", liste2, "List"]
```

```
Out[99]= liste2Sc.txt
```

Arbeitskurven

```
In[100]:= rR := 9.5
```

```
In[101]:= BetragB1[phi_] := Sqrt[xBik1[phi]^2 + yBik1[phi]^2]
```

```
In[102]:= xv[phi_] := xBik[phi] - xA0
```

```
In[103]:= yv[phi_] := yBik[phi] - yA0
```

```
In[104]:= xi[phi_] := xBik[phi] + rR * (yBik1[phi] / BetragB1[phi])
```

```
In[105]:= yi[phi_] := yBik[phi] + rR * (xBik1[phi] / BetragB1[phi])
```

```
In[106]:= xiv[phi_] := xi[phi] - xA0
```

```
In[107]:= yiv[phi_] := yi[phi] - yA0
```

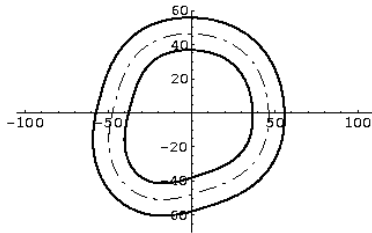
```
In[108]:= xa[phi_] := xBik[phi] - rR * (yBik1[phi] / BetragB1[phi])
```

```
In[109]:= ya[phi_] := yBik[phi] - rR * (xBik1[phi] / BetragB1[phi])
```

```
In[110]:= xav[phi_] := xa[phi] - xA0
```

```
In[111]:= yav[phi_] := ya[phi] - yA0
```

```
In[112]:= ParametricPlot[{ {xv[phi], yv[phi]}, {xiv[phi], yiv[phi]}, {xav[phi], yav[phi]}}, {phi, 0°, 360°},
    PlotStyle -> {Dashing[{0.01, 0.02, 0.05, 0.02}], Thickness[0.005], Thickness[0.005]},
    PlotRange -> {{-105, 110}, {-70, 60}}]
```



```
Out[112]= -Graphics-
```

```
In[113]:= liste3 := Table[{phi, xv[phi], yv[phi], xiv[phi], yiv[phi], xav[phi], yav[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[114]:= Export["liste3Sc.txt", liste3, "List"]
```

```
Out[114]= liste3Sc.txt
```

B.3 Setzhub_dyn.nb

Übertragungsfunktionen

```
In[115]:= a0 := 16.9003
```

```
In[116]:= a1 := -25.064
```

```
In[117]:= b1 := -10.4783
```

```
In[118]:= a2 := 9.45316
```

```
In[119]:= b2 := 9.16597
```

```
In[120]:= a3 := -1.41296
```

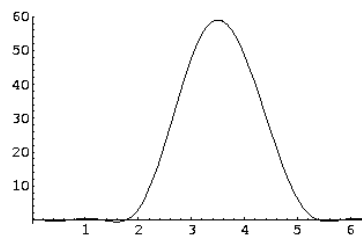
```
In[121]:= b3 := -2.09477
```

```
In[122]:= a4 := 0.356817
```

```
In[123]:= b4 := -0.765526
```

```
In[124]:= x[phi_] := a0 + a1 * cos[phi] + b1 * sin[phi] + a2 * cos[2 * phi] + b2 * sin[2 * phi] +  
a3 * cos[3 * phi] + b3 * sin[3 * phi] + a4 * cos[4 * phi] + b4 * sin[4 * phi]
```

```
In[125]:= Plot[x[phi], {phi, 0°, 360°}, PlotRange -> {{0°, 360°}, {-1, 60}}]
```



```
Out[125]= -Graphics-
```

```
In[126]:= liste4 := Table[{phi, x[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[127]:= Export["liste4Se.txt", liste4, "List"]
```

```
Out[127]= liste4Se.txt
```

```
In[128]:= xs0 := -31.5467
```

```
In[129]:= l3 := 61.29
```

```
In[130]:= l4 := 122.87
```

```
In[131]:= lsk := 160.5
```

```
In[132]:= ys0 := 131.986
```

```
In[133]:= ASe[phi_] := 2 * lsk * (xs0 - x[phi])
```

```
In[134]:= BSe[phi_] := -2 * lsk * ys0
```

```
In[135]:= CSe[phi_] := (xs0 - x[phi])^2 + ys0^2 - l4^2 + lsk^2
```

```
In[136]:= psisk[phi_] := 2 * arctan [ ( BSe[phi] - sqrt(ASe[phi]^2 + BSe[phi]^2 - CSe[phi]^2) ) /
(ASe[phi] - CSe[phi]) ]
```

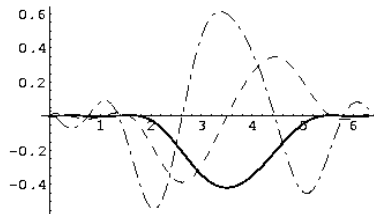
```
In[137]:= psi[phi_] := psisk[phi] - psisk[0°]
```

```
In[138]:= deltaphi := 0.001°
```

```
In[139]:= psil[phi_] := (psi[(phi + deltaphi)] - psi[(phi - deltaphi)]) / (2 * deltaphi)
```

```
In[140]:= psi2[phi_] := (psi[(phi + deltaphi)] - 2 * psi[phi] + psi[phi - deltaphi]) / (deltaphi^2)
```

```
In[141]:= Plot[{psi[phi], psil[phi], psi2[phi]}, {phi, 0°, 360°}, PlotStyle -> {Thickness[0.005],
Dashing[{0.03}], Dashing[{0.01, 0.02, 0.05, 0.02}]}]
```



```
Out[141]= -Graphics-
```

```
In[142]:= listel := Table[{phi, psi[phi], psil[phi], psi2[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[143]:= Export["listelSe.txt", listel, "List"]
```

```
Out[143]= listelSe.txt
```

Koordinaten der Rollenmittelpunktsbahn

```
In[144]:= xA0 := 40.46
```

```
In[145]:= yA0 := 62
```

```
In[146]:= l1 := Sqrt[xA0^2 + yA0^2]
```

```
In[147]:= rG := 50.199
```

```
In[148]:= psiG := arccos[(l1^2 + l3^2 - rG^2) / (2 * l1 * l3)]
```

```
In[149]:= MSe[phi_] := (1 - cos[phi] + (l3 / l1) * cos[psiG + psi[phi] + phi])
```

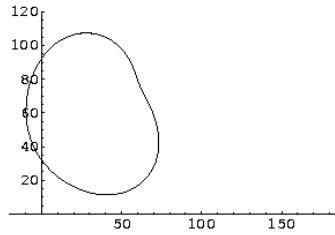
```
In[150]:= NSe[phi_] := (-sin[phi] + (l3 / l1) * sin[psiG + psi[phi] + phi])
```



```
In[151]:= xBik[phi_] := (xA0 * MSe[phi] - yA0 * NSe[phi])
```

```
In[152]:= yBik[phi_] := (xA0 * NSe[phi] + yA0 * MSe[phi])
```

```
In[153]:= ParametricPlot[{xBik[phi], yBik[phi]}, {phi, 0°, 360°}, PlotRange -> {{-20, 185}, {0, 120}}]
```



```
Out[153]= -Graphics-
```

```
In[154]:= xBik1[phi_] := (xA0 * sin[phi] + yA0 * cos[phi] - (13 / 11) * (psi1[phi] + 1) *
      (xA0 * sin[psiG + psi[phi] + phi] + yA0 * cos[psiG + psi[phi] + phi]))
```

```
In[155]:= yBik1[phi_] := (xA0 * cos[phi] - yA0 * sin[phi] - (13 / 11) * (psi1[phi] + 1) *
      (xA0 * cos[psiG + psi[phi] + phi] - yA0 * sin[psiG + psi[phi] + phi]))
```

```
In[156]:= liste2 :=
```

```
Table[{phi, xBik[phi], yBik[phi], xBik1[phi], yBik1[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[157]:= Export["liste2Se.txt", liste2, "List"]
```

```
Out[157]= liste2Se.txt
```

Arbeitskurven

```
In[158]:= rR := 7
```

```
In[159]:= BetragB1[phi_] := Sqrt[xBik1[phi]^2 + yBik1[phi]^2]
```

```
In[160]:= xv[phi_] := xBik[phi] - xA0
```

```
In[161]:= yv[phi_] := yBik[phi] - yA0
```

```
In[162]:= xi[phi_] := xBik[phi] + rR * (yBik1[phi] / BetragB1[phi])
```

```
In[163]:= yi[phi_] := yBik[phi] + rR * (xBik1[phi] / BetragB1[phi])
```

```
In[164]:= xiv[phi_] := xi[phi] - xA0
```

```
In[165]:= yiv[phi_] := yi[phi] - yA0
```

```
In[166]:= xa[phi_] := xBik[phi] - rR * (yBik1[phi] / BetragB1[phi])
```

```

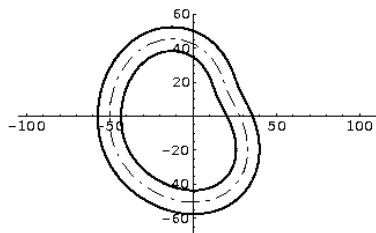
In[167]:= ya[phi_] := yBik[phi] - rR*(xBik1[phi]/BetragB1[phi])

In[168]:= xav[phi_] := xa[phi] - xA0

In[169]:= yav[phi_] := ya[phi] - yA0

In[170]:= ParametricPlot[{ {xv[phi], yv[phi]}, {xiv[phi], yiv[phi]}, {xav[phi], yav[phi]} }, {phi, 0°, 360°},
    PlotStyle -> {Dashing[{0.01, 0.02, 0.05, 0.02}], Thickness[0.005], Thickness[0.005]},
    PlotRange -> {{-105, 110}, {-70, 60}}]

```



```
Out[170]= -Graphics-
```

```

In[171]:= liste3 := Table[{phi, xv[phi], yv[phi], xiv[phi], yiv[phi], xav[phi], yav[phi]}, {phi, 0°, 360°, 1°}]

```

```

In[172]:= Export["liste3Se.txt", liste3, "List"]

```

```
Out[172]= liste3Se.txt
```

B.4 Vorschub_dyn.nb

Übertragungsfunktionen

```
In[173]:= a0 := 1.76736
```

```
In[174]:= a1 := -0.723446
```

```
In[175]:= b1 := 2.65826
```

```
In[176]:= a2 := -0.969136
```

```
In[177]:= b2 := -0.540282
```

```
In[178]:= a3 := -0.154056
```

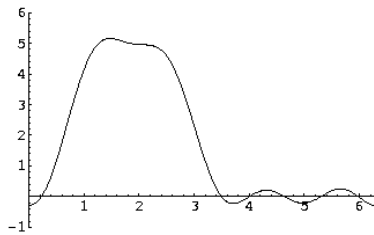
```
In[179]:= b3 := 0.0573191
```

```
In[180]:= a4 := -0.20547
```

```
In[181]:= b4 := -0.439409
```

```
In[182]:= x[phi_] := a0 + a1 * cos[phi] + b1 * sin[phi] + a2 * cos[2 * phi] + b2 * sin[2 * phi] +  
a3 * cos[3 * phi] + b3 * sin[3 * phi] + a4 * cos[4 * phi] + b4 * sin[4 * phi]
```

```
In[183]:= Plot[x[phi], {phi, 0°, 360°}, PlotRange -> {{0°, 360°}, {-1, 60}}]
```



```
Out[183]= - Graphics -
```

```
In[184]:= liste4 := Table[{phi, x[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[185]:= Export["liste4V.txt", liste4, "List"]
```

```
Out[185]= liste4V.txt
```

```
In[186]:= xs0 := -31.5467
```

```
In[187]:= l3 := 61.29
```

```
In[188]:= l4 := 122.87
```

```
In[189]:= lsk := 160.5
```

```
In[190]:= ys0 := 131.986
```

```
In[191]:= AV[phi_] := 2 * lsk * (xs0 - x[phi])
```

```

In[192]:= BV[phi_] := -2 * lsk * ys0

In[193]:= CV[phi_] := (xs0 - x[phi])^2 + ys0^2 - l4^2 + lsk^2

In[194]:= psisk[phi_] := 2 * arctan [ ( BV[phi] - sqrt(AV[phi]^2 + BV[phi]^2 - CV[phi]^2) ) /
      ( AV[phi] - CV[phi] ) ]

In[195]:= psi[phi_] := psisk[phi] - psisk[0°]

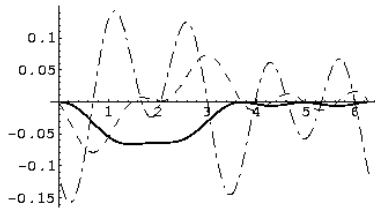
In[196]:= deltaphi := 0.001°

In[197]:= psil[phi_] := (psi[(phi + deltaphi)] - psi[(phi - deltaphi)]) / (2 * deltaphi)

In[198]:= psi2[phi_] := (psi[(phi + deltaphi)] - 2 * psi[phi] + psi[phi - deltaphi]) / (deltaphi^2)

In[199]:= Plot[{psi[phi], psil[phi], psi2[phi]}, {phi, 0°, 360°}, PlotStyle -> {Thickness[0.005],
      Dashing[{0.03}], Dashing[{0.01, 0.02, 0.05, 0.02}]}]

```



```

Out[199]= -Graphics-

In[200]:= listel := Table[{phi, psi[phi], psil[phi], psi2[phi]}, {phi, 0°, 360°, 1°}]

In[201]:= Export["listelV.txt", listel, "List"]

Out[201]= listelV.txt

```

Koordinaten der Rollenmittelpunktsbahn

```

In[202]:= xA0 := 40.46

In[203]:= yA0 := 62

In[204]:= l1 := Sqrt[xA0^2 + yA0^2]

In[205]:= rG := 50.199

In[206]:= psiG := arccos[(l1^2 + l3^2 - rG^2) / (2 * l1 * l3)]

In[207]:= MV[phi_] := (1 - cos[phi] + (l3 / l1) * cos[psiG + psi[phi] + phi])

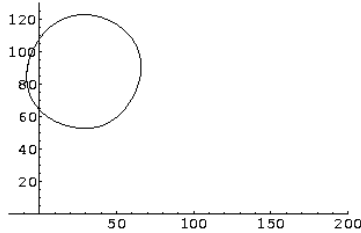
In[208]:= NV[phi_] := (-sin[phi] + (l3 / l1) * sin[psiG + psi[phi] + phi])

```

```
In[209]:= xBik[phi_] := (xA0 * MV[phi] - yA0 * NV[phi])
```

```
In[210]:= yBik[phi_] := (xA0 * NV[phi] + yA0 * MV[phi])
```

```
In[211]:= ParametricPlot[{xBik[phi], yBik[phi]}, {phi, 0°, 360°}, PlotRange -> {{-20, 185}, {0, 120}}]
```



```
Out[211]= - Graphics -
```

```
In[212]:= xBik1[phi_] := (xA0 * sin[phi] + yA0 * cos[phi] - (13 / 11) * (psi1[phi] + 1) *
      (xA0 * sin[psiG + psi[phi] + phi] + yA0 * cos[psiG + psi[phi] + phi]))
```

```
In[213]:= yBik1[phi_] := (xA0 * cos[phi] - yA0 * sin[phi] - (13 / 11) * (psi1[phi] + 1) *
      (xA0 * cos[psiG + psi[phi] + phi] - yA0 * sin[psiG + psi[phi] + phi]))
```

```
In[214]:= liste2 :=
```

```
Table[{phi, xBik[phi], yBik[phi], xBik1[phi], yBik1[phi]}, {phi, 0°, 360°, 1°}]
```

```
In[215]:= Export["liste2V.txt", liste2, "List"]
```

```
Out[215]= liste2V.txt
```

Arbeitskurven

```
In[216]:= rR := 7
```

```
In[217]:= BetragB1[phi_] := Sqrt[xBik1[phi]^2 + yBik1[phi]^2]
```

```
In[218]:= xv[phi_] := xBik[phi] - xA0
```

```
In[219]:= yv[phi_] := yBik[phi] - yA0
```

```
In[220]:= xi[phi_] := xBik[phi] + rR * (yBik1[phi] / BetragB1[phi])
```

```
In[221]:= yi[phi_] := yBik[phi] + rR * (xBik1[phi] / BetragB1[phi])
```

```
In[222]:= xiv[phi_] := xi[phi] - xA0
```

```
In[223]:= yiv[phi_] := yi[phi] - yA0
```

```
In[224]:= xa[phi_] := xBik[phi] - rR * (yBik1[phi] / BetragB1[phi])
```

```

In[225]:= ya[phi_] := yBik[phi] - rR*(xBik1[phi]/BetragB1[phi])

In[226]:= xav[phi_] := xa[phi] - xA0

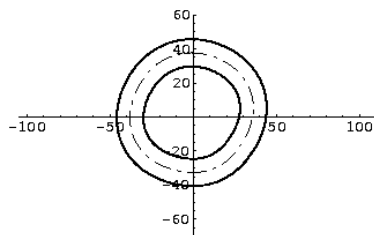
In[227]:= yav[phi_] := ya[phi] - yA0

In[228]:= ParametricPlot[{ {xv[phi], yv[phi]}, {xiv[phi], yiv[phi]}, {xav[phi], yav[phi]}}, {phi, 0°, 360°},

    PlotStyle -> {Dashing[{0.01, 0.02, 0.05, 0.02}], Thickness[0.005], Thickness[0.005]},

    PlotRange -> {{-105, 110}, {-70, 60}}]

```



```
Out[228]= -Graphics-
```

```

In[229]:= liste3 := Table[{phi, xv[phi], yv[phi], xiv[phi], yiv[phi], xav[phi], yav[phi]}, {phi, 0°, 360°, 1°}]

```

```

In[230]:= Export["liste3V.txt", liste3, "List"]

```

```
Out[230]= liste3V.txt
```

C C++ Quellcode

C.1 functionplot.h

```
1  /*****
2
3  functionplot.h
4
5  Diese Datei enthält Quellcode der Version 2.7.3 des Programms "opticurv",
6  das zur Berechnung der Kurvenscheibenform von Kurvenkoppelgetrieben
7  entwickelt wurde.
8
9  Copyright (C) 2003  Lutz Wirsig
10
11  Kontakt: Lutz Wirsig, Niederwiesaer Str.1, D-09577 Lichtenwalde
12  Email: lutz.wirsig@mailbox.tu-dresden.de
13
14  This program is free software; you can redistribute it and/or modify it
15  under the terms of the GNU General Public License as published by the
16  Free Software Foundation; either version 2 of the License, or (at your
17  option) any later version.
18
19  This program is distributed in the hope that it will be useful, but
20  WITHOUT ANY WARRANTY; without even the implied warranty of
21  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22  General Public License for more details.
23
24  You should have received a copy of the GNU General Public License
25  along with this program; if not, write to the Free Software Foundation,
26  Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
27
28  *****/
29
30  #ifndef _FUNCTIONPLOT_H
31  #define _FUNCTIONPLOT_H
32
33  #include <qapplication.h>
34  #include <qpainter.h>
35  #include <qpixmap.h>
36  #include <qwidget.h>
37
38  typedef struct
39  {
```

```

40     double  x, y;
41 }
42 valuePair;
43
44 //..... Klasse FunctionPlot
45 class FunctionPlot : public QWidget
46 {
47 public:
48     //... Konstruktor, dem die Anzahl der Wertepaare fuer
49     //... die zu plottende Funktion uebergeben wird
50     FunctionPlot( int n, double dPlotSkalierFaktor, QWidget *p=0,
51     const char *name=0 );
52     //... Destruktor; gibt den fuer die Wertepaare
53     //... reservierten Speicherplatz wieder frei
54     ~FunctionPlot()
55     {
56         delete [] valuesKurve1;
57         delete [] valuesKurve2;
58         delete [] valuesKurve3;
59         delete [] valuesKurve4;
60         delete [] valuesKurve5;
61         delete [] valuesKurve6;
62         delete [] valuesKurve7;
63         delete [] valuesKurve8;
64         delete [] valuesKurve9;
65         delete [] valuesKurve10;
66         delete [] valuesKurve11;
67     }
68
69     //... setzt das i.te Wertepaar auf die Werte v.x und v.y
70     void setValueKurve1( int i, valuePair v );
71     void setValueKurve2( int i, valuePair u );
72     void setValueKurve3( int i, valuePair w );
73     void setValueKurve4( int i, valuePair m );
74     void setValueKurve5( int i, valuePair u );
75     void setValueKurve6( int i, valuePair w );
76     void setValueKurve7( int i, valuePair m );
77     void setValueKurve8( int i, valuePair m );
78     void setValueKurve9( int i, valuePair r );
79     void setValueKurve10( int i, valuePair s );
80     void setValueKurve11( int i, valuePair t );
81
82     //... setzt die ersten n Wertepaare auf die Werte aus dem Array v
83     void setValuesKurve1( int n, valuePair v[] );

```



```

84     void setValuesKurve2( int n, valuePair u[] );
85     void setValuesKurve3( int n, valuePair w[] );
86     void setValuesKurve4( int n, valuePair m[] );
87     void setValuesKurve5( int n, valuePair u[] );
88     void setValuesKurve6( int n, valuePair w[] );
89     void setValuesKurve7( int n, valuePair m[] );
90     void setValuesKurve8( int n, valuePair m[] );
91     void setValuesKurve9( int n, valuePair r[] );
92     void setValuesKurve10( int n, valuePair s[] );
93     void setValuesKurve11( int n, valuePair t[] );
94
95     //... legt die minimalen und maximalen x- und y-Werte
96     //... des Bereichs fest, der von der Funktion anzuzeigen ist
97     void setPlotView( double minx, double miny, double maxx, double maxy );
98
99     //... bewirkt, dass die Funktion gezeichnet wird
100
101     void plotIt( void );
102     void plotIt_DK_aa( void );           // Doppelkurve mit Aussen-Aussen-Kontur
103     void plotIt_DK_ai( void );           // Doppelkurve mit Aussen-Innen-Kontur
104     void plotIt_DK_ia( void );           // Doppelkurve mit Innen-Aussen-Kontur
105     void plotIt_NK( void );               // Nutkurve
106     void plotIt_Hub_geschw( void );       // Hub; Geschwindigkeit
107     void plotIt_psi_geschw_beschl( void ); // psi ...
108     void plotIt_xBik( void );             // x_Bik, x_Bik'
109     void plotIt_yBik( void );             // y_Bik, y_Bik'
110     void plotIt_durchmesser( void );
111     void plotIt_mue( void );
112     void plotIt_rK( void );
113     void plotIt_MK( void );               // Mittelpunktskurven
114     void plotIt_DK_aa_durchmesser( void ); // Doppelkurve mit Wellendurchmesser
115     void plotIt_DK_ai_durchmesser( void );
116     void plotIt_DK_ia_durchmesser( void );
117
118     void druckeFunktion( const QString &filename, const QString &function,
119                         const QString &datetime );
120     void plotIt_manuell( double x_min, double x_max, double dx,    // manuelle
121                         double y_min, double y_max, double dy ); // Skalierung
122
123     QPixmap puffer; //(size());
124
125
126 protected:
127

```

```

128         //... bewirkt ein Vergroessern des Funktionsbereichs
129         //... "Cursor rechts": negative x-Achse wird verlaengert
130         //... "Cursor links" : positive x-Achse wird verlaengert
131         //... "Cursor hoch"  : negative y-Achse wird verlaengert
132         //... "Cursor tief"  : positive y-Achse wird verlaengert
133     virtual void keyPressEvent( QKeyEvent *ev );
134
135         //... leitet ein Zoomen (Verkleinern des Funktionsbereichs) ein
136     virtual void mousePressEvent( QMouseEvent *ev );
137
138         //... Zoomen (Verkleinern des Funktionsbereichs) findet gerade statt
139     virtual void mouseMoveEvent( QMouseEvent *ev );
140
141         //... Zoomen (Verkleinern des Funktionsbereichs) wird abgeschlossen
142     virtual void mouseReleaseEvent( QMouseEvent *ev );
143
144         //... bewirkt ein Neumalen der Funktion in dem aktuell
145         //... festgelegten Funktionsbereich
146     virtual void paintEvent( QPaintEvent *ev );
147
148 private:
149
150         //... zeichnen die x- bzw. y-Achse
151     void drawXScale( QPainter *p, double i, int yp );
152     void drawYScale( QPainter *p, double i, int xp );
153     void drawXScalepuffer( QPainter *ppuffer, double i, int yp );
154     void drawYScalepuffer( QPainter *ppuffer, double i, int xp );
155
156     int valueNo;    // Anzahl der Wertepaare der Funktion
157
158     double diffX, diffY; // Abstand der Skalenwerte
159
160     valuePair *valuesKurve1; // enthaelt die einzelnen Wertepaare
161     valuePair *valuesKurve2;
162     valuePair *valuesKurve3;
163     valuePair *valuesKurve4;
164     valuePair *valuesKurve5;
165     valuePair *valuesKurve6;
166     valuePair *valuesKurve7;
167     valuePair *valuesKurve8;
168     valuePair *valuesKurve9;
169     valuePair *valuesKurve10;
170     valuePair *valuesKurve11;
171

```

```

172     bool        plotViewSet; // zeigt an, ob explizit ein eigener
173                               // anzuzeigender Funktionsbereich festgelegt
174                               // wurde, also setPlotView() aufgerufen wurde.
175                               // plotViewSet=false, werden die minimalen
176                               // und maximalen x- und y-Werte des
177                               // anzuzeigenden Bereichs implizit bestimmt
178
179     double       minX, minY, maxX, maxY, // minimale und maximale x- und y-Werte
180                               // des anzuzeigenden Bereichs
181     xFactor, yFactor, // interne Projektionsfaktoren
182
183     SkalierFaktor; // Skalierfaktor beim Drucken
184
185     QPoint       startPos, letztePos, neuePos; // Mauspositionen, die fuer
186                               // das Zoomen mit der Maus
187                               // benoetigt werden.
188     bool         dragging, ersteMal; // fuer Ziehen der Maus benoetigt
189
190     QColor       farbe; // zeigt an, ob Zoomen aktiviert ist (gruen)
191                               // oder nicht (rot)
192 };
193
194 #endif
195
196 //..... Konstruktor FunctionPlot
197 FunctionPlot::FunctionPlot( int n, double dPlotSkalierFaktor, QWidget *p,
198     const char *name ): QWidget( p, name )
199 {
200     SkalierFaktor=dPlotSkalierFaktor;
201
202     setBackgroundColor( Qt::white );
203     valueNo = n;
204     valuesKurve1 = new valuePair [n];
205     valuesKurve2 = new valuePair [n];
206     valuesKurve3 = new valuePair [n];
207     valuesKurve4 = new valuePair [n];
208     valuesKurve5 = new valuePair [n];
209     valuesKurve6 = new valuePair [n];
210     valuesKurve7 = new valuePair [n];
211     valuesKurve8 = new valuePair [n];
212     valuesKurve9 = new valuePair [n];
213     valuesKurve10 = new valuePair [n];
214     valuesKurve11 = new valuePair [n];
215     for ( int i=1; i<(n-1); i++ )

```

```

216     {
217         valuesKurve1[i].x = valuesKurve1[i].y = 0.0;
218         valuesKurve2[i].x = valuesKurve2[i].y = 0.0;
219         valuesKurve3[i].x = valuesKurve3[i].y = 0.0;
220         valuesKurve4[i].x = valuesKurve4[i].y = 0.0;
221         valuesKurve5[i].x = valuesKurve5[i].y = 0.0;
222         valuesKurve6[i].x = valuesKurve6[i].y = 0.0;
223         valuesKurve7[i].x = valuesKurve7[i].y = 0.0;
224         valuesKurve8[i].x = valuesKurve8[i].y = 0.0;
225         valuesKurve9[i].x = valuesKurve9[i].y = 0.0;
226         valuesKurve10[i].x = valuesKurve10[i].y = 0.0;
227         valuesKurve11[i].x = valuesKurve11[i].y = 0.0;
228     }
229     plotViewSet = false;
230 }
231
232 /* setValueKurveX */
233
234 void FunctionPlot::setValueKurve1( int i, valuePair v )
235 {
236     if ( i >= valueNo || i < 0 )
237         return;
238     valuesKurve1[i] = v;
239 }
240 void FunctionPlot::setValueKurve2( int i, valuePair u )
241 {
242     if ( i >= valueNo || i < 0 )
243         return;
244     valuesKurve2[i] = u;
245 }
246 void FunctionPlot::setValueKurve3( int i, valuePair w )
247 {
248     if ( i >= valueNo || i < 0 )
249         return;
250     valuesKurve3[i] = w;
251 }
252 void FunctionPlot::setValueKurve4( int i, valuePair m )
253 {
254     if ( i >= valueNo || i < 0 )
255         return;
256     valuesKurve4[i] = m;
257 }
258 void FunctionPlot::setValueKurve5( int i, valuePair u )
259 {

```

```

260     if ( i >= valueNo || i < 0 )
261         return;
262     valuesKurve5[i] = u;
263 }
264 void FunctionPlot::setValueKurve6( int i, valuePair w )
265 {
266     if ( i >= valueNo || i < 0 )
267         return;
268     valuesKurve6[i] = w;
269 }
270 void FunctionPlot::setValueKurve7( int i, valuePair m )
271 {
272     if ( i >= valueNo || i < 0 )
273         return;
274     valuesKurve7[i] = m;
275 }
276 void FunctionPlot::setValueKurve8( int i, valuePair m )
277 {
278     if ( i >= valueNo || i < 0 )
279         return;
280     valuesKurve8[i] = m;
281 }
282 void FunctionPlot::setValueKurve9( int i, valuePair r )
283 {
284     if ( i >= valueNo || i < 0 )
285         return;
286     valuesKurve9[i] = r;
287 }
288 void FunctionPlot::setValueKurve10( int i, valuePair s )
289 {
290     if ( i >= valueNo || i < 0 )
291         return;
292     valuesKurve10[i] = s;
293 }
294 void FunctionPlot::setValueKurve11( int i, valuePair t )
295 {
296     if ( i >= valueNo || i < 0 )
297         return;
298     valuesKurve11[i] = t;
299 }
300
301 //..... setValuesKurveX
302 void FunctionPlot::setValuesKurve1( int n, valuePair v[] )
303 {

```

```

304     for ( int i=0; i<n; i++ )
305         setValueKurve1( i, v[i] );
306 }
307 void FunctionPlot::setValuesKurve2( int n, valuePair u[] )
308 {
309     for ( int i=0; i<n; i++ )
310         setValueKurve2( i, u[i] );
311 }
312 void FunctionPlot::setValuesKurve3( int n, valuePair w[] )
313 {
314     for ( int i=0; i<n; i++ )
315         setValueKurve3( i, w[i] );
316 }
317 void FunctionPlot::setValuesKurve4( int n, valuePair m[] )
318 {
319     for ( int i=0; i<n; i++ )
320         setValueKurve4( i, m[i] );
321 }
322 void FunctionPlot::setValuesKurve5( int n, valuePair u[] )
323 {
324     for ( int i=0; i<n; i++ )
325         setValueKurve5( i, u[i] );
326 }
327 void FunctionPlot::setValuesKurve6( int n, valuePair w[] )
328 {
329     for ( int i=0; i<n; i++ )
330         setValueKurve6( i, w[i] );
331 }
332 void FunctionPlot::setValuesKurve7( int n, valuePair m[] )
333 {
334     for ( int i=0; i<n; i++ )
335         setValueKurve7( i, m[i] );
336 }
337 void FunctionPlot::setValuesKurve8( int n, valuePair m[] )
338 {
339     for ( int i=0; i<n; i++ )
340         setValueKurve8( i, m[i] );
341 }
342 void FunctionPlot::setValuesKurve9( int n, valuePair r[] )
343 {
344     for ( int i=0; i<n; i++ )
345         setValueKurve9( i, r[i] );
346 }
347 void FunctionPlot::setValuesKurve10( int n, valuePair s[] )

```

```

348 {
349     for ( int i=0; i<n; i++ )
350         setValueKurve10( i, s[i] );
351 }
352 void FunctionPlot::setValuesKurve11( int n, valuePair t[] )
353 {
354     for ( int i=0; i<n; i++ )
355         setValueKurve11( i, t[i] );
356 }
357
358 //..... setPlotView
359 void FunctionPlot::setPlotView( double minx, double miny,
360                                 double maxx, double maxy )
361 {
362     plotViewSet = true;
363     minX = minx;
364     minY = miny;
365     maxX = maxx;
366     maxY = maxy;
367 }
368
369 /* Groesse des Plotfensters ermitteln */
370
371 void FunctionPlot::plotIt_DK_aa( void )
372 {
373     if ( !plotViewSet )
374     {
375         minX = maxX = 0;
376         minY = maxY = 0;
377         for ( int i=1; i<valueNo; i++ )
378         {
379             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
380             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
381             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
382             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
383             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
384             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
385             if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
386             if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
387             if ( valuesKurve9[i].x < minX ) minX = valuesKurve9[i].x;
388             if ( valuesKurve9[i].x > maxX ) maxX = valuesKurve9[i].x;
389             if ( valuesKurve9[i].y < minY ) minY = valuesKurve9[i].y;
390             if ( valuesKurve9[i].y > maxY ) maxY = valuesKurve9[i].y;
391             if ( valuesKurve10[i].x < minX ) minX = valuesKurve10[i].x;

```

```

392         if ( valuesKurve10[i].x > maxX ) maxX = valuesKurve10[i].x;
393         if ( valuesKurve10[i].y < minY ) minY = valuesKurve10[i].y;
394         if ( valuesKurve10[i].y > maxY ) maxY = valuesKurve10[i].y;
395     }
396     //.... Sicherstellen, dass Achsen sichtbar
397     if ( minX > 0 ) minX = 0;
398     if ( maxX < 0 ) maxX = 0;
399     if ( minY > 0 ) minY = 0;
400     if ( maxY < 0 ) maxY = 0;
401 }
402     diffX = (maxX-minX) / 10;
403     diffY = (maxY-minY) / 10;
404     repaint();
405 }
406 void FunctionPlot::plotIt_DK_ai( void )
407 {
408     if ( !plotViewSet )
409     {
410         minX = maxX = 0;
411         minY = maxY = 0;
412         for ( int i=1; i<valueNo; i++ )
413         {
414             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
415             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
416             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
417             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
418             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
419             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
420             if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
421             if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
422             if ( valuesKurve9[i].x < minX ) minX = valuesKurve9[i].x;
423             if ( valuesKurve9[i].x > maxX ) maxX = valuesKurve9[i].x;
424             if ( valuesKurve9[i].y < minY ) minY = valuesKurve9[i].y;
425             if ( valuesKurve9[i].y > maxY ) maxY = valuesKurve9[i].y;
426             if ( valuesKurve10[i].x < minX ) minX = valuesKurve10[i].x;
427             if ( valuesKurve10[i].x > maxX ) maxX = valuesKurve10[i].x;
428             if ( valuesKurve10[i].y < minY ) minY = valuesKurve10[i].y;
429             if ( valuesKurve10[i].y > maxY ) maxY = valuesKurve10[i].y;
430         }
431         //.... Sicherstellen, dass Achsen sichtbar
432         if ( minX > 0 ) minX = 0;
433         if ( maxX < 0 ) maxX = 0;
434         if ( minY > 0 ) minY = 0;
435         if ( maxY < 0 ) maxY = 0;

```



```

436     }
437     diffX = (maxX-minX) / 10;
438     diffY = (maxY-minY) / 10;
439     repaint();
440 }
441 void FunctionPlot::plotIt_DK_ia( void )
442 {
443     if ( !plotViewSet )
444     {
445         minX = maxX = 0;
446         minY = maxY = 0;
447         for ( int i=1; i<valueNo; i++ )
448         {
449             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
450             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
451             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
452             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
453             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
454             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
455             if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
456             if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
457             if ( valuesKurve9[i].x < minX ) minX = valuesKurve9[i].x;
458             if ( valuesKurve9[i].x > maxX ) maxX = valuesKurve9[i].x;
459             if ( valuesKurve9[i].y < minY ) minY = valuesKurve9[i].y;
460             if ( valuesKurve9[i].y > maxY ) maxY = valuesKurve9[i].y;
461             if ( valuesKurve10[i].x < minX ) minX = valuesKurve10[i].x;
462             if ( valuesKurve10[i].x > maxX ) maxX = valuesKurve10[i].x;
463             if ( valuesKurve10[i].y < minY ) minY = valuesKurve10[i].y;
464             if ( valuesKurve10[i].y > maxY ) maxY = valuesKurve10[i].y;
465         }
466         //.... Sicherstellen, dass Achsen sichtbar
467         if ( minX > 0 ) minX = 0;
468         if ( maxX < 0 ) maxX = 0;
469         if ( minY > 0 ) minY = 0;
470         if ( maxY < 0 ) maxY = 0;
471     }
472     diffX = (maxX-minX) / 10;
473     diffY = (maxY-minY) / 10;
474     repaint();
475 }
476 void FunctionPlot::plotIt_NK( void )
477 {
478     if ( !plotViewSet )
479     {

```

```

480     minX = maxX = 0;
481     minY = maxY = 0;
482     for ( int i=1; i<valueNo; i++ )
483     {
484         if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
485         if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
486         if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
487         if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
488         if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
489         if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
490         if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
491         if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
492         if ( valuesKurve7[i].x < minX ) minX = valuesKurve7[i].x;
493         if ( valuesKurve7[i].x > maxX ) maxX = valuesKurve7[i].x;
494         if ( valuesKurve7[i].y < minY ) minY = valuesKurve7[i].y;
495         if ( valuesKurve7[i].y > maxY ) maxY = valuesKurve7[i].y;
496     }
497     //.... Sicherstellen, dass Achsen sichtbar
498     if ( minX > 0 ) minX = 0;
499     if ( maxX < 0 ) maxX = 0;
500     if ( minY > 0 ) minY = 0;
501     if ( maxY < 0 ) maxY = 0;
502 }
503     diffX = (maxX-minX) / 10;
504     diffY = (maxY-minY) / 10;
505     repaint();
506 }
507 void FunctionPlot::plotIt_Hub_geschw( void )
508 {
509     if ( !plotViewSet )
510     {
511         minX = maxX = 0;
512         minY = maxY = 0;
513         for ( int i=1; i<valueNo; i++ )
514         {
515             if ( valuesKurve1[i].x < minX ) minX = valuesKurve1[i].x;
516             if ( valuesKurve1[i].x > maxX ) maxX = valuesKurve1[i].x;
517             if ( valuesKurve1[i].y < minY ) minY = valuesKurve1[i].y;
518             if ( valuesKurve1[i].y > maxY ) maxY = valuesKurve1[i].y;
519             if ( valuesKurve3[i].x < minX ) minX = valuesKurve3[i].x;
520             if ( valuesKurve3[i].x > maxX ) maxX = valuesKurve3[i].x;
521             if ( valuesKurve3[i].y < minY ) minY = valuesKurve3[i].y;
522             if ( valuesKurve3[i].y > maxY ) maxY = valuesKurve3[i].y;
523         }

```

```

524     //.... Sicherstellen, dass Achsen sichtbar
525     if ( minX > 0 ) minX = 0;
526     if ( maxX < 0 ) maxX = 0;
527     if ( minY > 0 ) minY = 0;
528     if ( maxY < 0 ) maxY = 0;
529 }
530     diffX = (maxX-minX) / 10;
531     diffY = (maxY-minY) / 10;
532     repaint();
533 }
534 void FunctionPlot::plotIt_psi_geschw_beschl( void )
535 {
536     if ( !plotViewSet )
537     {
538         minX = maxX = 0;
539         minY = maxY = 0;
540         for ( int i=1; i<valueNo; i++ )
541         {
542             if ( valuesKurve1[i].x < minX ) minX = valuesKurve1[i].x;
543             if ( valuesKurve1[i].x > maxX ) maxX = valuesKurve1[i].x;
544             if ( valuesKurve1[i].y < minY ) minY = valuesKurve1[i].y;
545             if ( valuesKurve1[i].y > maxY ) maxY = valuesKurve1[i].y;
546             if ( valuesKurve3[i].x < minX ) minX = valuesKurve3[i].x;
547             if ( valuesKurve3[i].x > maxX ) maxX = valuesKurve3[i].x;
548             if ( valuesKurve3[i].y < minY ) minY = valuesKurve3[i].y;
549             if ( valuesKurve3[i].y > maxY ) maxY = valuesKurve3[i].y;
550             if ( valuesKurve4[i].x < minX ) minX = valuesKurve4[i].x;
551             if ( valuesKurve4[i].x > maxX ) maxX = valuesKurve4[i].x;
552             if ( valuesKurve4[i].y < minY ) minY = valuesKurve4[i].y;
553             if ( valuesKurve4[i].y > maxY ) maxY = valuesKurve4[i].y;
554         }
555         //.... Sicherstellen, dass Achsen sichtbar
556         if ( minX > 0 ) minX = 0;
557         if ( maxX < 0 ) maxX = 0;
558         if ( minY > 0 ) minY = 0;
559         if ( maxY < 0 ) maxY = 0;
560     }
561     diffX = (maxX-minX) / 10;
562     diffY = (maxY-minY) / 10;
563     repaint();
564 }
565 void FunctionPlot::plotIt_xBik( void )
566 {
567     if ( !plotViewSet )

```

```

568     {
569         minX = maxX = 0;
570         minY = maxY = 0;
571         for ( int i=1; i<valueNo; i++ )
572         {
573             if ( valuesKurve1[i].x < minX ) minX = valuesKurve1[i].x;
574             if ( valuesKurve1[i].x > maxX ) maxX = valuesKurve1[i].x;
575             if ( valuesKurve1[i].y < minY ) minY = valuesKurve1[i].y;
576             if ( valuesKurve1[i].y > maxY ) maxY = valuesKurve1[i].y;
577             if ( valuesKurve3[i].x < minX ) minX = valuesKurve3[i].x;
578             if ( valuesKurve3[i].x > maxX ) maxX = valuesKurve3[i].x;
579             if ( valuesKurve3[i].y < minY ) minY = valuesKurve3[i].y;
580             if ( valuesKurve3[i].y > maxY ) maxY = valuesKurve3[i].y;
581         }
582         //.... Sicherstellen, dass Achsen sichtbar
583         if ( minX > 0 ) minX = 0;
584         if ( maxX < 0 ) maxX = 0;
585         if ( minY > 0 ) minY = 0;
586         if ( maxY < 0 ) maxY = 0;
587     }
588     diffX = (maxX-minX) / 10;
589     diffY = (maxY-minY) / 10;
590     repaint();
591 }
592 void FunctionPlot::plotIt_yBik( void )
593 {
594     if ( !plotViewSet )
595     {
596         minX = maxX = 0;
597         minY = maxY = 0;
598         for ( int i=1; i<valueNo; i++ )
599         {
600             if ( valuesKurve1[i].x < minX ) minX = valuesKurve1[i].x;
601             if ( valuesKurve1[i].x > maxX ) maxX = valuesKurve1[i].x;
602             if ( valuesKurve1[i].y < minY ) minY = valuesKurve1[i].y;
603             if ( valuesKurve1[i].y > maxY ) maxY = valuesKurve1[i].y;
604             if ( valuesKurve3[i].x < minX ) minX = valuesKurve3[i].x;
605             if ( valuesKurve3[i].x > maxX ) maxX = valuesKurve3[i].x;
606             if ( valuesKurve3[i].y < minY ) minY = valuesKurve3[i].y;
607             if ( valuesKurve3[i].y > maxY ) maxY = valuesKurve3[i].y;
608         }
609         //.... Sicherstellen, dass Achsen sichtbar
610         if ( minX > 0 ) minX = 0;
611         if ( maxX < 0 ) maxX = 0;

```

```

612     if ( minY > 0 ) minY = 0;
613     if ( maxY < 0 ) maxY = 0;
614 }
615     diffX = (maxX-minX) / 10;
616     diffY = (maxY-minY) / 10;
617     repaint();
618 }
619 void FunctionPlot::plotIt_durchmesser( void )
620 {
621     if ( !plotViewSet )
622     {
623         minX = maxX = 0;
624         minY = maxY = 0;
625         for ( int i=1; i<valueNo; i++ )
626         {
627             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
628             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
629             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
630             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
631             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
632             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
633             if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
634             if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
635             if ( valuesKurve7[i].x < minX ) minX = valuesKurve7[i].x;
636             if ( valuesKurve7[i].x > maxX ) maxX = valuesKurve7[i].x;
637             if ( valuesKurve7[i].y < minY ) minY = valuesKurve7[i].y;
638             if ( valuesKurve7[i].y > maxY ) maxY = valuesKurve7[i].y;
639             if ( valuesKurve8[i].x < minX ) minX = valuesKurve8[i].x;
640             if ( valuesKurve8[i].x > maxX ) maxX = valuesKurve8[i].x;
641             if ( valuesKurve8[i].y < minY ) minY = valuesKurve8[i].y;
642             if ( valuesKurve8[i].y > maxY ) maxY = valuesKurve8[i].y;
643             if ( valuesKurve11[i].x < minX ) minX = valuesKurve11[i].x;
644             if ( valuesKurve11[i].x > maxX ) maxX = valuesKurve11[i].x;
645             if ( valuesKurve11[i].y < minY ) minY = valuesKurve11[i].y;
646             if ( valuesKurve11[i].y > maxY ) maxY = valuesKurve11[i].y;
647         }
648         //.... Sicherstellen, dass Achsen sichtbar
649         if ( minX > 0 ) minX = 0;
650         if ( maxX < 0 ) maxX = 0;
651         if ( minY > 0 ) minY = 0;
652         if ( maxY < 0 ) maxY = 0;
653     }
654     diffX = (maxX-minX) / 10;
655     diffY = (maxY-minY) / 10;

```

```

656     repaint();
657 }
658 void FunctionPlot::plotIt_mue( void )
659 {
660     if ( !plotViewSet )
661     {
662         minX = maxX = 0;
663         minY = maxY = 0;
664         for ( int i=1; i<valueNo; i++ )
665         {
666             if ( valuesKurve1[i].x < minX ) minX = valuesKurve1[i].x;
667             if ( valuesKurve1[i].x > maxX ) maxX = valuesKurve1[i].x;
668             if ( valuesKurve1[i].y < minY ) minY = valuesKurve1[i].y;
669             if ( valuesKurve1[i].y > maxY ) maxY = valuesKurve1[i].y;
670         }
671         //.... Sicherstellen, dass Achsen sichtbar
672         if ( minX > 0 ) minX = 0;
673         if ( maxX < 0 ) maxX = 0;
674         if ( minY > 0 ) minY = 0;
675         if ( maxY < 0 ) maxY = 0;
676     }
677     diffX = (maxX-minX) / 10;
678     diffY = (maxY-minY) / 10;
679     repaint();
680 }
681 void FunctionPlot::plotIt_rK( void )
682 {
683     if ( !plotViewSet )
684     {
685         minX = maxX = 0;
686         minY = maxY = 0;
687         for ( int i=1; i<valueNo; i++ )
688         {
689             if ( valuesKurve1[i].x < minX ) minX = valuesKurve1[i].x;
690             if ( valuesKurve1[i].x > maxX ) maxX = valuesKurve1[i].x;
691             if ( valuesKurve1[i].y < minY ) minY = valuesKurve1[i].y;
692             if ( valuesKurve1[i].y > maxY ) maxY = valuesKurve1[i].y;
693         }
694         //.... Sicherstellen, dass Achsen sichtbar
695         if ( minX > 0 ) minX = 0;
696         if ( maxX < 0 ) maxX = 0;
697         if ( minY > 0 ) minY = 0;
698         if ( maxY < 0 ) maxY = 0;
699     }

```

```

700     diffX = (maxX-minX) / 10;
701     diffY = (maxY-minY) / 10;
702     repaint();
703 }
704 void FunctionPlot::plotIt_MK( void )
705 {
706     if ( !plotViewSet )
707     {
708         minX = maxX = 0;
709         minY = maxY = 0;
710         for ( int i=1; i<valueNo; i++ )
711         {
712             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
713             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
714             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
715             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
716             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
717             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
718             if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
719             if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
720         }
721         //.... Sicherstellen, dass Achsen sichtbar
722         if ( minX > 0 ) minX = 0;
723         if ( maxX < 0 ) maxX = 0;
724         if ( minY > 0 ) minY = 0;
725         if ( maxY < 0 ) maxY = 0;
726     }
727     diffX = (maxX-minX) / 10;
728     diffY = (maxY-minY) / 10;
729     repaint();
730 }
731 void FunctionPlot::plotIt_DK_aa_durchmesser( void )
732 {
733     if ( !plotViewSet )
734     {
735         minX = maxX = 0;
736         minY = maxY = 0;
737         for ( int i=1; i<valueNo; i++ )
738         {
739             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
740             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
741             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
742             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
743             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;

```

```

744     if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
745     if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
746     if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
747     if ( valuesKurve9[i].x < minX ) minX = valuesKurve9[i].x;
748     if ( valuesKurve9[i].x > maxX ) maxX = valuesKurve9[i].x;
749     if ( valuesKurve9[i].y < minY ) minY = valuesKurve9[i].y;
750     if ( valuesKurve9[i].y > maxY ) maxY = valuesKurve9[i].y;
751     if ( valuesKurve10[i].x < minX ) minX = valuesKurve10[i].x;
752     if ( valuesKurve10[i].x > maxX ) maxX = valuesKurve10[i].x;
753     if ( valuesKurve10[i].y < minY ) minY = valuesKurve10[i].y;
754     if ( valuesKurve10[i].y > maxY ) maxY = valuesKurve10[i].y;
755     if ( valuesKurve11[i].x < minX ) minX = valuesKurve11[i].x;
756     if ( valuesKurve11[i].x > maxX ) maxX = valuesKurve11[i].x;
757     if ( valuesKurve11[i].y < minY ) minY = valuesKurve11[i].y;
758     if ( valuesKurve11[i].y > maxY ) maxY = valuesKurve11[i].y;
759 }
760 //.... Sicherstellen, dass Achsen sichtbar
761 if ( minX > 0 ) minX = 0;
762 if ( maxX < 0 ) maxX = 0;
763 if ( minY > 0 ) minY = 0;
764 if ( maxY < 0 ) maxY = 0;
765 }
766 diffX = (maxX-minX) / 10;
767 diffY = (maxY-minY) / 10;
768 repaint();
769 }
770 void FunctionPlot::plotIt_DK_ai_durchmesser( void )
771 {
772     if ( !plotViewSet )
773     {
774         minX = maxX = 0;
775         minY = maxY = 0;
776         for ( int i=1; i<valueNo; i++ )
777         {
778             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
779             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
780             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
781             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
782             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
783             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
784             if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
785             if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
786             if ( valuesKurve9[i].x < minX ) minX = valuesKurve9[i].x;
787             if ( valuesKurve9[i].x > maxX ) maxX = valuesKurve9[i].x;

```



```

788         if ( valuesKurve9[i].y < minY ) minY = valuesKurve9[i].y;
789         if ( valuesKurve9[i].y > maxY ) maxY = valuesKurve9[i].y;
790         if ( valuesKurve10[i].x < minX ) minX = valuesKurve10[i].x;
791         if ( valuesKurve10[i].x > maxX ) maxX = valuesKurve10[i].x;
792         if ( valuesKurve10[i].y < minY ) minY = valuesKurve10[i].y;
793         if ( valuesKurve10[i].y > maxY ) maxY = valuesKurve10[i].y;
794         if ( valuesKurve11[i].x < minX ) minX = valuesKurve11[i].x;
795         if ( valuesKurve11[i].x > maxX ) maxX = valuesKurve11[i].x;
796         if ( valuesKurve11[i].y < minY ) minY = valuesKurve11[i].y;
797         if ( valuesKurve11[i].y > maxY ) maxY = valuesKurve11[i].y;
798     }
799     //.... Sicherstellen, dass Achsen sichtbar
800     if ( minX > 0 ) minX = 0;
801     if ( maxX < 0 ) maxX = 0;
802     if ( minY > 0 ) minY = 0;
803     if ( maxY < 0 ) maxY = 0;
804 }
805     diffX = (maxX-minX) / 10;
806     diffY = (maxY-minY) / 10;
807     repaint();
808 }
809 void FunctionPlot::plotIt_DK_ia_durchmesser( void )
810 {
811     if ( !plotViewSet )
812     {
813         minX = maxX = 0;
814         minY = maxY = 0;
815         for ( int i=1; i<valueNo; i++ )
816         {
817             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
818             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
819             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
820             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
821             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
822             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;
823             if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
824             if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
825             if ( valuesKurve9[i].x < minX ) minX = valuesKurve9[i].x;
826             if ( valuesKurve9[i].x > maxX ) maxX = valuesKurve9[i].x;
827             if ( valuesKurve9[i].y < minY ) minY = valuesKurve9[i].y;
828             if ( valuesKurve9[i].y > maxY ) maxY = valuesKurve9[i].y;
829             if ( valuesKurve10[i].x < minX ) minX = valuesKurve10[i].x;
830             if ( valuesKurve10[i].x > maxX ) maxX = valuesKurve10[i].x;
831             if ( valuesKurve10[i].y < minY ) minY = valuesKurve10[i].y;

```

```

832         if ( valuesKurve10[i].y > maxY ) maxY = valuesKurve10[i].y;
833         if ( valuesKurve11[i].x < minX ) minX = valuesKurve11[i].x;
834         if ( valuesKurve11[i].x > maxX ) maxX = valuesKurve11[i].x;
835         if ( valuesKurve11[i].y < minY ) minY = valuesKurve11[i].y;
836         if ( valuesKurve11[i].y > maxY ) maxY = valuesKurve11[i].y;
837     }
838     //.... Sicherstellen, dass Achsen sichtbar
839     if ( minX > 0 ) minX = 0;
840     if ( maxX < 0 ) maxX = 0;
841     if ( minY > 0 ) minY = 0;
842     if ( maxY < 0 ) maxY = 0;
843 }
844     diffX = (maxX-minX) / 10;
845     diffY = (maxY-minY) / 10;
846     repaint();
847 }
848
849 void FunctionPlot::plotIt( void ) {
850     if ( !plotViewSet ) {
851         minX = maxX = 0;
852         minY = maxY = 0;
853         for ( int i=1; i<valueNo; i++ ) {
854             if ( valuesKurve1[i].x < minX ) minX = valuesKurve1[i].x;
855             if ( valuesKurve1[i].x > maxX ) maxX = valuesKurve1[i].x;
856             if ( valuesKurve1[i].y < minY ) minY = valuesKurve1[i].y;
857             if ( valuesKurve1[i].y > maxY ) maxY = valuesKurve1[i].y;
858             if ( valuesKurve2[i].x < minX ) minX = valuesKurve2[i].x;
859             if ( valuesKurve2[i].x > maxX ) maxX = valuesKurve2[i].x;
860             if ( valuesKurve2[i].y < minY ) minY = valuesKurve2[i].y;
861             if ( valuesKurve2[i].y > maxY ) maxY = valuesKurve2[i].y;
862             if ( valuesKurve3[i].x < minX ) minX = valuesKurve3[i].x;
863             if ( valuesKurve3[i].x > maxX ) maxX = valuesKurve3[i].x;
864             if ( valuesKurve3[i].y < minY ) minY = valuesKurve3[i].y;
865             if ( valuesKurve3[i].y > maxY ) maxY = valuesKurve3[i].y;
866             if ( valuesKurve4[i].x < minX ) minX = valuesKurve4[i].x;
867             if ( valuesKurve4[i].x > maxX ) maxX = valuesKurve4[i].x;
868             if ( valuesKurve4[i].y < minY ) minY = valuesKurve4[i].y;
869             if ( valuesKurve4[i].y > maxY ) maxY = valuesKurve4[i].y;
870             if ( valuesKurve5[i].x < minX ) minX = valuesKurve5[i].x;
871             if ( valuesKurve5[i].x > maxX ) maxX = valuesKurve5[i].x;
872             if ( valuesKurve5[i].y < minY ) minY = valuesKurve5[i].y;
873             if ( valuesKurve5[i].y > maxY ) maxY = valuesKurve5[i].y;
874             if ( valuesKurve6[i].x < minX ) minX = valuesKurve6[i].x;
875             if ( valuesKurve6[i].x > maxX ) maxX = valuesKurve6[i].x;

```

```

876         if ( valuesKurve6[i].y < minY ) minY = valuesKurve6[i].y;
877         if ( valuesKurve6[i].y > maxY ) maxY = valuesKurve6[i].y;
878         if ( valuesKurve7[i].x < minX ) minX = valuesKurve7[i].x;
879         if ( valuesKurve7[i].x > maxX ) maxX = valuesKurve7[i].x;
880         if ( valuesKurve7[i].y < minY ) minY = valuesKurve7[i].y;
881         if ( valuesKurve7[i].y > maxY ) maxY = valuesKurve7[i].y;
882         if ( valuesKurve8[i].x < minX ) minX = valuesKurve8[i].x;
883         if ( valuesKurve8[i].x > maxX ) maxX = valuesKurve8[i].x;
884         if ( valuesKurve8[i].y < minY ) minY = valuesKurve8[i].y;
885         if ( valuesKurve8[i].y > maxY ) maxY = valuesKurve8[i].y;
886         if ( valuesKurve9[i].x < minX ) minX = valuesKurve9[i].x;
887         if ( valuesKurve9[i].x > maxX ) maxX = valuesKurve9[i].x;
888         if ( valuesKurve9[i].y < minY ) minY = valuesKurve9[i].y;
889         if ( valuesKurve9[i].y > maxY ) maxY = valuesKurve9[i].y;
890         if ( valuesKurve10[i].x < minX ) minX = valuesKurve10[i].x;
891         if ( valuesKurve10[i].x > maxX ) maxX = valuesKurve10[i].x;
892         if ( valuesKurve10[i].y < minY ) minY = valuesKurve10[i].y;
893         if ( valuesKurve10[i].y > maxY ) maxY = valuesKurve10[i].y;
894         if ( valuesKurve11[i].x < minX ) minX = valuesKurve11[i].x;
895         if ( valuesKurve11[i].x > maxX ) maxX = valuesKurve11[i].x;
896         if ( valuesKurve11[i].y < minY ) minY = valuesKurve11[i].y;
897         if ( valuesKurve11[i].y > maxY ) maxY = valuesKurve11[i].y;
898     }
899     //.... Sicherstellen, dass Achsen sichtbar
900     if ( minX > 0 ) minX = 0;
901     if ( maxX < 0 ) maxX = 0;
902     if ( minY > 0 ) minY = 0;
903     if ( maxY < 0 ) maxY = 0;
904 }
905 diffX = (maxX-minX) / 10;
906 diffY = (maxY-minY) / 10;
907 repaint();
908 }
909 //..... druckeFunktion
910 void FunctionPlot::druckeFunktion( const QString &filename,
911     const QString &function, const QString &datetime )
912 {
913     QPainter *meinDrucker;
914     meinDrucker = new QPainter;
915     // setBackgroundMode( NoBackground ); // beim Neumalen nicht loeschen
916
917     QString version = "Opticurv Version 2.7.3 - TU DRESDEN";
918
919     if( meinDrucker->setup( this ) )

```

```

920     {
921         QPainter p( meinDrucker );
922         p.drawText( SkalierFaktor*20, SkalierFaktor*20, version );
923         p.drawText( SkalierFaktor*20, SkalierFaktor*40, function );
924         p.drawText( SkalierFaktor*20, SkalierFaktor*60, datetime );
925         p.drawText( SkalierFaktor*20, SkalierFaktor*80, filename );
926         //.... Bild auf Drucker kopieren
927         p.drawPixmap( SkalierFaktor*20, SkalierFaktor*200, puffer );
928     }
929 }
930 //..... plotIt_manuell
931 void FunctionPlot::plotIt_manuell( double x_min, double x_max,
932     double dx, double y_min, double y_max, double dy )
933 {
934     minX = x_min;
935     maxX = x_max;
936     diffX = dx;
937
938     minY = y_min;
939     maxY = y_max;
940     diffY = dy;
941 }
942 //..... keyPressEvent
943 void FunctionPlot::keyPressEvent( QKeyEvent *ev )
944 {
945     double diffX = (maxX-minX) / 10,
946           diffY = (maxY-(minY+100)) / 10;
947     switch (ev->key() )
948     {
949         case Key_Right:  minX -= diffX; repaint(); break;
950         case Key_Left:   maxX += diffX; repaint(); break;
951         case Key_Up:     minY -= diffY; repaint(); break;
952         case Key_Down:   maxY += diffY; repaint(); break;
953     }
954 }
955
956 //..... mousePressEvent
957 void FunctionPlot::mousePressEvent( QMouseEvent *ev )
958 {
959     ersteMal = true;
960     dragging = false;
961     startPos = ev->pos();
962     repaint( false );
963 }

```

```

964
965 //..... mouseMoveEvent
966 void FunctionPlot::mouseMoveEvent( QMouseEvent *ev )
967 {
968     if ( !dragging )
969         //.... Dragging (Ziehen) einschalten, wenn
970         //.... neue Position mind. 20 Pixel entfernt ist
971         if ( QABS( startPos.x() - ev->pos().x() ) >= 20 ||
972             QABS( startPos.y() - ev->pos().y() ) >= 20 )
973             dragging = true;
974     if ( dragging )
975     {
976         neuePos = ev->pos();
977         //.... Farbe zeigt an, ob Zoom aktiviert ist
978         if ( QABS( startPos.x() - ev->pos().x() ) >= 50 &&
979             QABS( startPos.y() - ev->pos().y() ) >= 50 )
980             farbe = Qt::green.light( 60 );
981         else
982             farbe = Qt::red.light( 150 );
983         repaint( false );
984     }
985 }
986
987 //..... mouseReleaseEvent
988 void FunctionPlot::mouseReleaseEvent( QMouseEvent *ev )
989 {
990     //.... Bedingung soll versehentliches Zoomen verhindern
991     if ( QABS( startPos.x() - ev->pos().x() ) >= 50 &&
992         QABS( startPos.y() - ev->pos().y() ) >= 50 )
993     {
994         if ( ev->pos().x() > startPos.x() )
995         {
996             maxX = ev->pos().x() / xFactor + minX;
997             minX = startPos.x() / xFactor + minX;
998         }
999         else
1000         {
1001             maxX = startPos.x() / xFactor + minX;
1002             minX = ev->pos().x() / xFactor + minX;
1003         }
1004
1005         if ( ev->pos().y() > startPos.y() )
1006         {
1007             minY = maxY - ev->pos().y() / yFactor;

```

```

1008         maxY = maxY - startPos.y() / yFactor;
1009     }
1010     else
1011     {
1012         minY = maxY - startPos.y() / yFactor;
1013         maxY = maxY - ev->pos().y() / yFactor;
1014     }
1015 }
1016 dragging = false;
1017 repaint();
1018 }
1019
1020 //..... paintEvent
1021 void FunctionPlot::paintEvent( QPaintEvent * )
1022 {
1023     puffer.resize( SkalierFaktor*500, SkalierFaktor*500 );
1024     puffer.fill( white );
1025     QPainter p( this ); // erzeugt QPainter-Object zum Zeichnen im Window
1026     QPainter ppuffer;    // erzeugt QPainter-Object zum Zeichnen im Puffer
1027
1028     ppuffer.begin( &puffer );           // Start des Zeichnens im Puffer
1029
1030     double xp, yp, i;
1031
1032     xFactor = width() / (maxX -minX),
1033     yFactor = height() / (maxY -minY);
1034
1035     if ( dragging )
1036     {
1037         if ( !ersteMal )
1038             p.eraseRect( QRect( startPos, letztePos ) );
1039         ppuffer.eraseRect( QRect( startPos, letztePos ) );
1040         p.fillRect( QRect( startPos, neuePos ), farbe );
1041         ppuffer.fillRect( QRect( startPos, neuePos ), farbe );
1042         letztePos = neuePos;
1043         ersteMal = false;
1044     }
1045
1046     p.setPen( Qt::black );           //.... x- und y-Achsen zeichnen
1047     ppuffer.setPen( Qt::black );
1048     xp = -minX * xFactor;
1049     yp = maxY * yFactor;
1050     p.drawLine( 0, yp, width(), yp ); // x-Achse
1051     ppuffer.drawLine( 0, yp, width(), yp ); // x-Achse

```

```

1052     p.drawLine( xp, 0, xp, height() );          // y-Achse
1053     ppuffer.drawLine( xp, 0, xp, height() );     // y-Achse
1054
1055     ppuffer.drawRect( puffer.rect() );           // Rahmen zeichnen
1056
1057     /* x-Skalen zeichnen/beschriften */
1058
1059     for ( i = -diffX; i>=minX; i-=diffX )
1060     {
1061         drawXScale( &p, i, yp );
1062         drawXScalepuffer( &ppuffer, i, yp );
1063     }
1064     for ( i = diffX; i<=maxX; i+=diffX )
1065     {
1066         drawXScale( &p, i, yp );
1067         drawXScalepuffer( &ppuffer, i, yp );
1068     }
1069
1070
1071     /* y-Skalen zeichnen/beschriften */
1072     for ( i = -diffY; i>=minY; i-=diffY )
1073     {
1074         drawYScale( &p, i, xp );
1075         drawYScalepuffer( &ppuffer, i, xp );
1076     }
1077     for ( i = diffY; i<=maxY; i+=diffY )
1078     {
1079         drawYScale( &p, i, xp );
1080         drawYScalepuffer( &ppuffer, i, xp );
1081     }
1082
1083     /* Kurve 1 plotten */
1084
1085     xp = ( valuesKurve1[1].x - minX ) * xFactor;
1086     yp = ( maxY - valuesKurve1[1].y ) * yFactor;
1087     p.moveTo( xp, yp );
1088     p.setPen( Qt::black );
1089     p.setBrush( Qt::black );
1090     p.drawEllipse( xp-1, yp-1, 2, 2 );
1091     ppuffer.moveTo( xp, yp );
1092     ppuffer.setPen( Qt::black );
1093     ppuffer.setBrush( Qt::black );
1094     ppuffer.drawEllipse( xp-1, yp-1, 2, 2 );
1095

```

```

1096     for ( i=1; i<(valueNo-1); i++ )
1097     {
1098         xp = ( valuesKurve1[(int)i].x - minX ) * xFactor;
1099         yp = ( maxY - valuesKurve1[(int)i].y ) * yFactor;
1100         p.setPen( Qt::blue.light( 140 ) );
1101         p.lineTo( xp, yp );
1102         p.setPen( Qt::blue.light( 120 ) );
1103         p.drawEllipse( xp, yp, 2, 2 );
1104         ppuffer.setPen( Qt::blue.light( 140 ) );
1105         ppuffer.lineTo( xp, yp );
1106         ppuffer.setPen( Qt::blue.light( 120 ) );
1107         ppuffer.drawEllipse( xp, yp, 2, 2 );
1108     }
1109
1110     /* Kurve 2 plotten */
1111
1112     xp = ( valuesKurve2[1].x - minX ) * xFactor;
1113     yp = ( maxY - valuesKurve2[1].y ) * yFactor;
1114     p.moveTo( xp, yp );
1115     ppuffer.moveTo( xp, yp );
1116
1117     for ( i=1; i<(valueNo-1); i++ )
1118     {
1119         xp = ( valuesKurve2[(int)i].x - minX ) * xFactor;
1120         yp = ( maxY - valuesKurve2[(int)i].y ) * yFactor;
1121         p.setPen( Qt::green.light( 140 ) );
1122         p.lineTo( xp, yp );
1123         p.setPen( Qt::green.light( 120 ) );
1124         p.drawEllipse( xp, yp, 2, 2 );
1125         ppuffer.setPen( Qt::green.light( 140 ) );
1126         ppuffer.lineTo( xp, yp );
1127         ppuffer.setPen( Qt::green.light( 120 ) );
1128         ppuffer.drawEllipse( xp, yp, 2, 2 );
1129     }
1130
1131     /* Kurve 3 plotten */
1132
1133     xp = ( valuesKurve3[1].x - minX ) * xFactor;
1134     yp = ( maxY - valuesKurve3[1].y ) * yFactor;
1135     p.moveTo( xp, yp );
1136     ppuffer.moveTo( xp, yp );
1137
1138     for ( i=1; i<(valueNo-1); i++ )
1139     {

```



```

1140     xp = ( valuesKurve3[(int)i].x - minX ) * xFactor;
1141     yp = ( maxY - valuesKurve3[(int)i].y ) * yFactor;
1142     p.setPen( Qt::green.light( 140 ) );
1143     p.lineTo( xp, yp );
1144     p.setPen( Qt::green.light( 120 ) );
1145     p.drawEllipse( xp, yp, 2, 2 );
1146     ppuffer.setPen( Qt::green.light( 140 ) );
1147     ppuffer.lineTo( xp, yp );
1148     ppuffer.setPen( Qt::green.light( 120 ) );
1149     ppuffer.drawEllipse( xp, yp, 2, 2 );
1150 }
1151
1152 /* Kurve 4 plotten */
1153
1154 xp = ( valuesKurve4[1].x - minX ) * xFactor;
1155 yp = ( maxY - valuesKurve4[1].y ) * yFactor;
1156 p.moveTo( xp, yp );
1157 ppuffer.moveTo( xp, yp );
1158
1159 for ( i=1; i<(valueNo-1); i++ )
1160 {
1161     xp = ( valuesKurve4[(int)i].x - minX ) * xFactor;
1162     yp = ( maxY - valuesKurve4[(int)i].y ) * yFactor;
1163     p.setPen( Qt::red.light( 140 ) );
1164     p.lineTo( xp, yp );
1165     p.setPen( Qt::red.light( 120 ) );
1166     p.drawEllipse( xp, yp, 2, 2 );
1167     ppuffer.setPen( Qt::red.light( 140 ) );
1168     ppuffer.lineTo( xp, yp );
1169     ppuffer.setPen( Qt::red.light( 120 ) );
1170     ppuffer.drawEllipse( xp, yp, 2, 2 );
1171 }
1172
1173 /* Kurve 5 plotten */
1174
1175 xp = ( valuesKurve5[1].x - minX ) * xFactor;
1176 yp = ( maxY - valuesKurve5[1].y ) * yFactor;
1177 p.moveTo( xp, yp );
1178 p.setPen( Qt::black );
1179 p.setBrush( Qt::black );
1180 p.drawEllipse( xp-1, yp-1, 2, 2 );
1181 ppuffer.moveTo( xp, yp );
1182 ppuffer.setPen( Qt::black );
1183 ppuffer.setBrush( Qt::black );

```

```

1184     ppuffer.drawEllipse( xp-1, yp-1, 2, 2 );
1185
1186     for ( i=1; i<valueNo; i++ )
1187     {
1188         xp = ( valuesKurve5[(int)i].x - minX ) * xFactor;
1189         yp = ( maxY - valuesKurve5[(int)i].y ) * yFactor;
1190         p.setPen( Qt::blue.light( 140 ) );
1191         p.lineTo( xp, yp );
1192         p.setPen( Qt::blue.light( 120 ) );
1193         p.drawEllipse( xp, yp, 2, 2 );
1194         ppuffer.setPen( Qt::blue.light( 140 ) );
1195         ppuffer.lineTo( xp, yp );
1196         ppuffer.setPen( Qt::blue.light( 120 ) );
1197         ppuffer.drawEllipse( xp, yp, 2, 2 );
1198     }
1199     xp = ( valuesKurve5[1].x - minX ) * xFactor;
1200     yp = ( maxY - valuesKurve5[1].y ) * yFactor;
1201     p.setPen( Qt::blue.light( 140 ) );
1202     p.lineTo( xp, yp );
1203     p.setPen( Qt::blue.light( 120 ) );
1204     p.drawEllipse( xp, yp, 2, 2 );
1205     ppuffer.setPen( Qt::blue.light( 140 ) );
1206     ppuffer.lineTo( xp, yp );
1207     ppuffer.setPen( Qt::blue.light( 120 ) );
1208     ppuffer.drawEllipse( xp, yp, 2, 2 );
1209
1210     for ( i=1; i<2; i++ )
1211     {
1212         xp = ( valuesKurve5[(int)i].x - minX ) * xFactor;
1213         yp = ( maxY - valuesKurve5[(int)i].y ) * yFactor;
1214         p.setPen( Qt::black );
1215         p.lineTo( xp, yp );
1216         p.setPen( Qt::black );
1217         p.drawEllipse( xp, yp, 2.5, 2.5 );
1218         ppuffer.setPen( Qt::black );
1219         ppuffer.lineTo( xp, yp );
1220         ppuffer.setPen( Qt::black );
1221         ppuffer.drawEllipse( xp, yp, 2.5, 2.5 );
1222     }
1223
1224     /* Kurve 6 plotten */
1225
1226     xp = ( valuesKurve6[1].x - minX ) * xFactor;
1227     yp = ( maxY - valuesKurve6[1].y ) * yFactor;

```

```

1228     p.moveTo( xp, yp );
1229     ppuffer.moveTo( xp, yp );
1230
1231     for ( i=1; i<valueNo; i++ )
1232     {
1233         xp = ( valuesKurve6[(int)i].x - minX ) * xFactor;
1234         yp = ( maxY - valuesKurve6[(int)i].y ) * yFactor;
1235         p.setPen( Qt::green.light( 140 ) );
1236         p.lineTo( xp, yp );
1237         p.setPen( Qt::green.light( 120 ) );
1238         p.drawEllipse( xp, yp, 2, 2 );
1239         ppuffer.setPen( Qt::green.light( 140 ) );
1240         ppuffer.lineTo( xp, yp );
1241         ppuffer.setPen( Qt::green.light( 120 ) );
1242         ppuffer.drawEllipse( xp, yp, 2, 2 );
1243     }
1244     xp = ( valuesKurve6[1].x - minX ) * xFactor;
1245     yp = ( maxY - valuesKurve6[1].y ) * yFactor;
1246     p.setPen( Qt::green.light( 140 ) );
1247     p.lineTo( xp, yp );
1248     p.setPen( Qt::green.light( 120 ) );
1249     p.drawEllipse( xp, yp, 2, 2 );
1250     ppuffer.setPen( Qt::green.light( 140 ) );
1251     ppuffer.lineTo( xp, yp );
1252     ppuffer.setPen( Qt::green.light( 120 ) );
1253     ppuffer.drawEllipse( xp, yp, 2, 2 );
1254
1255     for ( i=1; i<2; i++ )
1256     {
1257         xp = ( valuesKurve6[(int)i].x - minX ) * xFactor;
1258         yp = ( maxY - valuesKurve6[(int)i].y ) * yFactor;
1259         p.setPen( Qt::black );
1260         p.lineTo( xp, yp );
1261         p.setPen( Qt::black );
1262         p.drawEllipse( xp, yp, 2.5, 2.5 );
1263         ppuffer.setPen( Qt::black );
1264         ppuffer.lineTo( xp, yp );
1265         ppuffer.setPen( Qt::black );
1266         ppuffer.drawEllipse( xp, yp, 2.5, 2.5 );
1267     }
1268
1269     /* Kurve 7 plotten */
1270
1271     xp = ( valuesKurve7[1].x - minX ) * xFactor;

```

```

1272     yp = ( maxY - valuesKurve7[1].y ) * yFactor;
1273     p.moveTo( xp, yp );
1274     ppuffer.moveTo( xp, yp );
1275     for ( i=1; i<valueNo; i++ )
1276     {
1277         xp = ( valuesKurve7[(int)i].x - minX ) * xFactor;
1278         yp = ( maxY - valuesKurve7[(int)i].y ) * yFactor;
1279         p.setPen( Qt::green.light( 140 ) );
1280         p.lineTo( xp, yp );
1281         p.setPen( Qt::green.light( 120 ) );
1282         p.drawEllipse( xp, yp, 2, 2 );
1283         ppuffer.setPen( Qt::green.light( 140 ) );
1284         ppuffer.lineTo( xp, yp );
1285         ppuffer.setPen( Qt::green.light( 120 ) );
1286         ppuffer.drawEllipse( xp, yp, 2, 2 );
1287     }
1288     xp = ( valuesKurve7[1].x - minX ) * xFactor;
1289     yp = ( maxY - valuesKurve7[1].y ) * yFactor;
1290     p.setPen( Qt::green.light( 140 ) );
1291     p.lineTo( xp, yp );
1292     p.setPen( Qt::green.light( 120 ) );
1293     p.drawEllipse( xp, yp, 2, 2 );
1294     ppuffer.setPen( Qt::green.light( 140 ) );
1295     ppuffer.lineTo( xp, yp );
1296     ppuffer.setPen( Qt::green.light( 120 ) );
1297     ppuffer.drawEllipse( xp, yp, 2, 2 );
1298
1299     for ( i=1; i<2; i++ )
1300     {
1301         xp = ( valuesKurve7[(int)i].x - minX ) * xFactor;
1302         yp = ( maxY - valuesKurve7[(int)i].y ) * yFactor;
1303         p.setPen( Qt::black );
1304         p.lineTo( xp, yp );
1305         p.setPen( Qt::black );
1306         p.drawEllipse( xp, yp, 2.5, 2.5 );
1307         ppuffer.setPen( Qt::black );
1308         ppuffer.lineTo( xp, yp );
1309         ppuffer.setPen( Qt::black );
1310         ppuffer.drawEllipse( xp, yp, 2.5, 2.5 );
1311     }
1312
1313     /* Kurve 8 plotten */
1314
1315     xp = ( valuesKurve8[1].x - minX ) * xFactor;

```

```

1316     yp = ( maxY - valuesKurve8[1].y ) * yFactor;
1317     p.moveTo( xp, yp );
1318     ppuffer.moveTo( xp, yp );
1319     for ( i=1; i<valueNo; i++ )
1320     {
1321         xp = ( valuesKurve8[(int)i].x - minX ) * xFactor;
1322         yp = ( maxY - valuesKurve8[(int)i].y ) * yFactor;
1323         p.setPen( Qt::red.light( 140 ) );
1324         p.lineTo( xp, yp );
1325         p.setPen( Qt::red.light( 120 ) );
1326         p.drawEllipse( xp, yp, 2, 2 );
1327         ppuffer.setPen( Qt::red.light( 140 ) );
1328         ppuffer.lineTo( xp, yp );
1329         ppuffer.setPen( Qt::red.light( 120 ) );
1330         ppuffer.drawEllipse( xp, yp, 2, 2 );
1331     }
1332     xp = ( valuesKurve8[1].x - minX ) * xFactor;
1333     yp = ( maxY - valuesKurve8[1].y ) * yFactor;
1334     p.setPen( Qt::red.light( 140 ) );
1335     p.lineTo( xp, yp );
1336     p.setPen( Qt::red.light( 120 ) );
1337     p.drawEllipse( xp, yp, 2, 2 );
1338     ppuffer.setPen( Qt::red.light( 140 ) );
1339     ppuffer.lineTo( xp, yp );
1340     ppuffer.setPen( Qt::red.light( 120 ) );
1341     ppuffer.drawEllipse( xp, yp, 2, 2 );
1342
1343     /* Kurve 9 plotten */
1344
1345     QColor lila( 240, 80, 180 );
1346
1347     xp = ( valuesKurve9[1].x - minX ) * xFactor;
1348     yp = ( maxY - valuesKurve9[1].y ) * yFactor;
1349     p.moveTo( xp, yp );
1350     p.setPen( lila );
1351     p.setBrush( lila );
1352     p.drawEllipse( xp-1, yp-1, 2, 2 );
1353     ppuffer.moveTo( xp, yp );
1354     ppuffer.setPen( lila );
1355     ppuffer.setBrush( lila );
1356     ppuffer.drawEllipse( xp-1, yp-1, 2, 2 );
1357
1358     for ( i=1; i<valueNo; i++ )
1359     {

```

```

1360     xp = ( valuesKurve9[(int)i].x - minX ) * xFactor;
1361     yp = ( maxY - valuesKurve9[(int)i].y ) * yFactor;
1362     p.setPen( lila );
1363     p.lineTo( xp, yp );
1364     p.setPen( lila );
1365     p.drawEllipse( xp, yp, 2, 2 );
1366     ppuffer.setPen( lila );
1367     ppuffer.lineTo( xp, yp );
1368     ppuffer.setPen( lila );
1369     ppuffer.drawEllipse( xp, yp, 2, 2 );
1370 }
1371 xp = ( valuesKurve9[1].x - minX ) * xFactor;
1372 yp = ( maxY - valuesKurve9[1].y ) * yFactor;
1373 p.setPen( lila );
1374 p.lineTo( xp, yp );
1375 p.setPen( lila );
1376 p.drawEllipse( xp, yp, 2, 2 );
1377 ppuffer.setPen( lila );
1378 ppuffer.lineTo( xp, yp );
1379 ppuffer.setPen( lila );
1380 ppuffer.drawEllipse( xp, yp, 2, 2 );
1381
1382 for ( i=1; i<2; i++ )
1383 {
1384     xp = ( valuesKurve9[(int)i].x - minX ) * xFactor;
1385     yp = ( maxY - valuesKurve9[(int)i].y ) * yFactor;
1386     p.setPen( lila );
1387     p.lineTo( xp, yp );
1388     p.setPen( lila );
1389     p.drawEllipse( xp, yp, 2.5, 2.5 );
1390     ppuffer.setPen( lila );
1391     ppuffer.lineTo( xp, yp );
1392     ppuffer.setPen( lila );
1393     ppuffer.drawEllipse( xp, yp, 2.5, 2.5 );
1394 }
1395
1396 /* Kurve 10 plotten */
1397
1398 QColor wgruen( 35, 200, 185 );
1399
1400 xp = ( valuesKurve10[1].x - minX ) * xFactor;
1401 yp = ( maxY - valuesKurve10[1].y ) * yFactor;
1402 p.moveTo( xp, yp );
1403 ppuffer.moveTo( xp, yp );

```

```

1404
1405     for ( i=1; i<valueNo; i++ )
1406     {
1407         xp = ( valuesKurve10[(int)i].x - minX ) * xFactor;
1408         yp = ( maxY - valuesKurve10[(int)i].y ) * yFactor;
1409         p.setPen( lila );
1410         p.lineTo( xp, yp );
1411         p.setPen( lila );
1412         p.drawEllipse( xp, yp, 2, 2 );
1413         ppuffer.setPen( lila );
1414         ppuffer.lineTo( xp, yp );
1415         ppuffer.setPen( lila );
1416         ppuffer.drawEllipse( xp, yp, 2, 2 );
1417     }
1418     xp = ( valuesKurve10[1].x - minX ) * xFactor;
1419     yp = ( maxY - valuesKurve10[1].y ) * yFactor;
1420     p.setPen( lila );
1421     p.lineTo( xp, yp );
1422     p.setPen( lila );
1423     p.drawEllipse( xp, yp, 2, 2 );
1424     ppuffer.setPen( lila );
1425     ppuffer.lineTo( xp, yp );
1426     ppuffer.setPen( lila );
1427     ppuffer.drawEllipse( xp, yp, 2, 2 );
1428
1429     for ( i=1; i<2; i++ )
1430     {
1431         xp = ( valuesKurve10[(int)i].x - minX ) * xFactor;
1432         yp = ( maxY - valuesKurve10[(int)i].y ) * yFactor;
1433         p.setPen( lila );
1434         p.lineTo( xp, yp );
1435         p.setPen( lila );
1436         p.drawEllipse( xp, yp, 2.5, 2.5 );
1437         ppuffer.setPen( lila );
1438         ppuffer.lineTo( xp, yp );
1439         ppuffer.setPen( lila );
1440         ppuffer.drawEllipse( xp, yp, 2.5, 2.5 );
1441     }
1442     /* Kurve 11 plotten */
1443
1444     xp = ( valuesKurve11[1].x - minX ) * xFactor;
1445     yp = ( maxY - valuesKurve11[1].y ) * yFactor;
1446     p.moveTo( xp, yp );
1447     ppuffer.moveTo( xp, yp );

```

```

1448     for ( i=1; i<valueNo; i++ )
1449     {
1450         xp = ( valuesKurve11[(int)i].x - minX ) * xFactor;
1451         yp = ( maxY - valuesKurve11[(int)i].y ) * yFactor;
1452         p.setPen( Qt::red.light( 140 ) );
1453         p.lineTo( xp, yp );
1454         p.setPen( Qt::red.light( 120 ) );
1455         p.drawEllipse( xp, yp, 2, 2 );
1456         ppuffer.setPen( Qt::red.light( 140 ) );
1457         ppuffer.lineTo( xp, yp );
1458         ppuffer.setPen( Qt::red.light( 120 ) );
1459         ppuffer.drawEllipse( xp, yp, 2, 2 );
1460     }
1461     xp = ( valuesKurve11[1].x - minX ) * xFactor;
1462     yp = ( maxY - valuesKurve11[1].y ) * yFactor;
1463     p.setPen( Qt::red.light( 140 ) );
1464     p.lineTo( xp, yp );
1465     p.setPen( Qt::red.light( 120 ) );
1466     p.drawEllipse( xp, yp, 2, 2 );
1467     ppuffer.setPen( Qt::red.light( 140 ) );
1468     ppuffer.lineTo( xp, yp );
1469     ppuffer.setPen( Qt::red.light( 120 ) );
1470     ppuffer.drawEllipse( xp, yp, 2, 2 );
1471
1472     ppuffer.end( );    // Beendigung des Zeichnens im Puffer
1473 }
1474
1475 //..... drawXScale
1476 void FunctionPlot::drawXScale( QPainter *p, double i, int yp )
1477 {
1478     QString text;
1479     double  xs = (i-minX)* xFactor;
1480
1481     text.sprintf( "%.3g", i );
1482     p->drawLine( xs, yp-2, xs, yp+2 );
1483     p->drawText( xs+1, yp-2, text );
1484     p->setPen( QPen( Qt::gray, 0, Qt::DotLine ) ); // Raster
1485     p->drawLine( xs, 0, xs, height() );
1486     p->setPen( QPen( Qt::black, 0, Qt::SolidLine ) );
1487 }
1488 void FunctionPlot::drawXScalepuffer( QPainter *ppuffer, double i, int yp )
1489 {
1490     QString text;
1491     double  xs = (i-minX) * xFactor;

```



```

1492
1493     text.sprintf( "%.3g", i );
1494     ppuffer->drawLine( xs, yp-2, xs, yp+2 );
1495     ppuffer->drawText( xs+1, yp-2, text );
1496     ppuffer->setPen( QPen( Qt::gray, 0, Qt::DotLine ) ); // Raster
1497     ppuffer->drawLine( xs, 0, xs, height() );
1498     ppuffer->setPen( QPen( Qt::black, 0, Qt::SolidLine ) );
1499 }
1500
1501 //..... drawYScale
1502 void FunctionPlot::drawYScale( QPainter *p, double i, int xp )
1503 {
1504     QString text;
1505     double ys = (maxY-i) * yFactor;
1506
1507     text.sprintf( "%.3g", i );
1508     p->drawLine( xp-2, ys, xp+2, ys );
1509     p->drawText( xp+4, ys, text );
1510     p->setPen( QPen( Qt::gray, 0, Qt::DotLine ) ); // Raster
1511     p->drawLine( 0, ys, width(), ys );
1512     p->setPen( QPen( Qt::black, 0, Qt::SolidLine ) );
1513 }
1514 void FunctionPlot::drawYScaleppuffer( QPainter *ppuffer, double i, int xp )
1515 {
1516     QString text;
1517     double ys = (maxY-i) * yFactor;
1518
1519     text.sprintf( "%.3g", i );
1520     ppuffer->drawLine( xp-2, ys, xp+2, ys );
1521     ppuffer->drawText( xp+4, ys, text );
1522     ppuffer->setPen( QPen( Qt::gray, 0, Qt::DotLine ) ); // Raster
1523     ppuffer->drawLine( 0, ys, width(), ys );
1524     ppuffer->setPen( QPen( Qt::black, 0, Qt::SolidLine ) );
1525 }
1526

```

C.2 opticurv273.cpp

```
1  /*****
2
3  opticurv273.cpp
4
5  Diese Datei enthält Quellcode der Version 2.7.3 des Programms "opticurv",
6  das zur Berechnung der Kurvenscheibenform von Kurvenkoppelgetrieben
7  entwickelt wurde.
8
9  Copyright (C) 2003  Lutz Wirsig
10
11  Kontakt: Lutz Wirsig, Niederwiesaer Str.1, D-09577 Lichtenwalde
12  Email: lutz.wirsig@mailbox.tu-dresden.de
13
14  This program is free software; you can redistribute it and/or modify it
15  under the terms of the GNU General Public License as published by the
16  Free Software Foundation; either version 2 of the License, or (at your
17  option) any later version.
18
19  This program is distributed in the hope that it will be useful, but
20  WITHOUT ANY WARRANTY; without even the implied warranty of
21  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22  General Public License for more details.
23
24  You should have received a copy of the GNU General Public License
25  along with this program; if not, write to the Free Software Foundation,
26  Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
27
28  *****/
29
30  #include <cmath>
31  #include <cstdlib>
32  #include <fstream.h>
33  #include <iostream.h>
34  #include <math.h>
35  #include <qapplication.h>
36  #include <qbuttongroup.h>
37  #include <qcheckbox.h>
38  #include <qcolor.h>
39  #include <qcombobox.h>
40  #include <qcstring.h>
41  #include <qdatetime.h>
42  #include <qdialog.h>
```

```

43  #include <qdir.h>
44  #include <qfile.h>
45  #include <qfiledialog.h>
46  #include <qframe.h>
47  #include <qimage.h>
48  #include <qkeycode.h>
49  #include <qlabel.h>
50  #include <qlayout.h>
51  #include <qlineedit.h>
52  #include <qmainwindow.h>
53  #include <qmap.h>
54  #include <qmenubar.h>
55  #include <qmessagebox.h>
56  #include <qmime.h>
57  #include <qmultilineedit.h>
58  #include <qpainter.h>
59  #include <qpaintdevicemetrics.h>
60  #include <qpixmap.h>
61  #include <qpopupmenu.h>
62  #include <qprinter.h>
63  #include <qpushbutton.h>
64  #include <qradiobutton.h>
65  #include <qscrollview.h>
66  #include <qsimplerichtext.h>
67  #include <qstatusbar.h>
68  #include <qstring.h>
69  #include <qstringlist.h>
70  #include <qtabdialog.h>
71  #include <qtextbrowser.h>
72  #include <qtextcodec.h>
73  #include <qtextstream.h>
74  #include <qtextview.h>
75  #include <qtooltip.h>
76  #include <qwidget.h>
77  #include <stdarg.h>
78  #include <stddef.h>
79  #include <stdio.h>
80  #include <stdlib.h>
81  #include <string.h>
82  #include "functionplot.h"
83
84  const long double pi = 3.14159265358979323846;
85  const long left = 2;
86  const long double deltaphi = 0.001;

```

```

87  const char *anzahlAbschnitte[] = {
88      "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12" };
89  const int anzahlNo = sizeof(anzahlAbschnitte) /
90      sizeof(anzahlAbschnitte[0]);
91  const char *bewegungsgesetz[] = { "Rast", "3-4-5 Polynom",
92      "Bestehornsinoide" };
93  const int bewgesetzNo = sizeof(bewegungsgesetz) /
94      sizeof(bewegungsgesetz[0]);
95  const char *auswahlKurvengetriebe[] = { "Nutkurvenscheibe",
96      "Doppelkurvenscheibe" };
97  const int auswahlKurveNo = sizeof(auswahlKurvengetriebe) /
98      sizeof(auswahlKurvengetriebe[0]);
99  const char *auswahlGegenkurvenkontur[] = { "Aussenkontur",
100      "Innenkontur" };
101  const int auswahlKonturNo = sizeof(auswahlGegenkurvenkontur) /
102      sizeof(auswahlGegenkurvenkontur[0]);
103  //----- class opticurv -----
104  class Opticurv : public QMainWindow    //friend of TabDialog
105  {
106      Q_OBJECT    // notwendig, da Opticurv Elementfunktionen enthaelt
107
108      public:
109
110      int index_zuweisung, indexe, indexAbschnitt, index1, index2,
111          index3, index4, index5, index6, index7, index8, index9,
112          index10, index11, index12, indexKurve, AnzeigenID, indexKontur;
113
114      uint i;
115
116      double dphi_H01, dphi_H12, dphi_H23, dphi_H34, dphi_H45,
117          dphi_H56, dphi_H67, dphi_H78, dphi_H89, dphi_H910,
118          dphi_H1011, dphi_H1112, ds_H01, ds_H12, ds_H23, ds_H34,
119          ds_H45, ds_H56, ds_H67, ds_H78, ds_H89, ds_H910, ds_H1011,
120          ds_H1112, dl_3, dx_s0, dl_4, dl_sk, dx_H, dy_s0_orig,
121          dy_H, dr_R, dx_A0, dy_A0, dbeta, dn_1, dn_2, dphi_R1fr,
122          dphi_R2fr, dergebnis, dr_G, dphi, ds, dpsi, dpsi_rad,
123          dpsi_strich_Grad, dpsi_zweistrich_Grad, dx_Bik, dy_Bik,
124          dx_Bik_strich, dy_Bik_strich, dx_i_versetzt,
125          dy_i_versetzt, dx_B_versetzt, dy_B_versetzt,
126          dx_a_versetzt, dy_a_versetzt, dr_Bik, dmue, dsmue, dr_K,
127          dphi_1, dphi_2, dphi_3, dphi_4, dphi_5, dphi_6, dphi_7,
128          dphi_8, dphi_9, dphi_10, dphi_11, dphi_12, ds_1, ds_2,
129          ds_3, ds_4, ds_5, ds_6, ds_7, ds_8, ds_9, ds_10, ds_11,
130          ds_12, dalpha, dgamma, dl_1, dy_s0, dpsi_sk0,

```

```

131     dpsi_0, dpsi_G, dhub_gleich, dquadrant,dri_scheibe,
132     dpsi_positiv, ds_strich, ds_zweistrich,
133     dabstand, psi_strich_analytisch, dr_S, dalpha_R,
134     dindexAbschnitt, dindex1, dindex2, dindex3, dindex4,
135     dindex5, dindex6, dindex7, dindex8, dindex9,
136     dindex10, dindex11, dindex12, dl_3star, dr_Rstar,
137     dpsi_Gstar, dindexKurve, dx_Bikstar, dy_Bikstar,
138     dpsistar, dpsistar_rad, dx_B_versetztstar,
139     dy_B_versetztstar, dx_Bik_strichstar, dy_Bik_strichstar,
140     dx_i_versetztstar, dy_i_versetztstar, dx_a_versetztstar,
141     dy_a_versetztstar, dDateinameAbfragen, dr_W,
142     dmanuelleArbeitskurven, dmanuelles_phi, dmanuellepsi_phi,
143     dmanuelleKurvenscheibe, dmanuellex_Bik, dmanuelley_Bik,
144     dmanuelleUebertragungswinkel, dmanuelleKruemmungsradius,
145     dmanuelleMittelpunktskurven, dx_minArbkurven,
146     dx_maxArbkurven, ddxArbkurven, dy_minArbkurven,
147     dy_maxArbkurven, ddyArbkurven, dx_mins_phi, dx_maxs_phi,
148     ddxs_phi, dy_mins_phi, dy_maxs_phi, ddys_phi,
149     dx_minpsi_phi, dx_maxpsi_phi, ddxpsi_phi, dy_minpsi_phi,
150     dy_maxpsi_phi, ddypsi_phi, dx_minKurvenscheibe,
151     dx_maxKurvenscheibe, ddxKurvenscheibe,
152     dy_minKurvenscheibe, dy_maxKurvenscheibe, ddyKurvenscheibe,
153     dx_minx_Bik, dx_maxx_Bik, ddx_Bik, dy_minx_Bik,
154     dy_maxx_Bik, ddyx_Bik, dx_miny_Bik, dx_maxy_Bik, ddx_Bik,
155     dy_miny_Bik, dy_maxy_Bik, ddy_Bik,
156     dx_minUebertragungswinkel, dx_maxUebertragungswinkel,
157     ddxUebertragungswinkel, dy_minUebertragungswinkel,
158     dy_maxUebertragungswinkel, ddyUebertragungswinkel,
159     dx_minKruemmungsradius, dx_maxKruemmungsradius, ddxKruemmungsradius,
160     dy_minKruemmungsradius, dy_maxKruemmungsradius, ddyKruemmungsradius,
161     dx_minMittelpunktskurven, dx_maxMittelpunktskurven,
162     ddxMittelpunktskurven, dy_minMittelpunktskurven,
163     dy_maxMittelpunktskurven, ddyMittelpunktskurven,
164     dPlotSkalierFaktor;
165
166     long double phi_H01, phi_H12, phi_H23, phi_H34, phi_H45,
167     phi_H56, phi_H67, phi_H78, phi_H89, phi_H910,
168     phi_H1011, phi_H1112, s_H01, s_H12, s_H23, s_H34,
169     s_H45, s_H56, s_H67, s_H78, s_H89, s_H910, s_H1011,
170     s_H1112, l_3, l_4, l_sk, x_H, y_H, r_R, y_s0_orig,
171     x_A0, y_A0, beta, n_1, n_2, phi_R1fr, phi_R2fr, ergebnis,
172     r_G, phi, alpha, gamma, x_s0, y_s0, s_1, s_2, s_3,
173     s_4, s_5, s_6, s_7, s_8, s_9, s_10, s_11, s_12,
174     A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R,

```

```

175     S, T, rad, psi_sk, psi_sk0, psi, psi_rad, A_0, x_Ba,
176     C_0, l_1, phi_1, phi_2, phi_3, phi_4, phi_5, phi_6,
177     phi_7, phi_8, phi_9, phi_10, phi_11, phi_12, s, s_strich,
178     s_zweistrich, r_Bik, mue, psi_0, s_1_strich, s_2_strich,
179     smue, r_Bik_strich, s_3_strich, s_4_strich, s_5_strich,
180     psi_strich, s_1_zweistrich, s_2_zweistrich, s_3_zweistrich,
181     s_4_zweistrich, s_5_zweistrich, Zaehler_psi_zweistrich,
182     Nenner_psi_zweistrich, psi_zweistrich, psi_strich_Grad,
183     psi_zweistrich_Grad, psi_G, alphan, x_Bik, y_Bik,
184     sinpsibeta, cospsibeta, x_Bik_strich, y_Bik_strich,
185     Betrag_B_strich, x_i, y_i, x_a, y_a, x_i_versetzt,
186     y_i_versetzt, x_B_versetzt, y_B_versetzt, x_a_versetzt,
187     y_a_versetzt, zaehler_mue, nenner_mue, x_Bik_zweistrich,
188     y_Bik_zweistrich, wurzelausdruck, potenz_wurzelausdruck,
189     r_K, zaehler_r_K, nenner_r_K, Abstand, s_plus_deltaphi,
190     s_minus_deltaphi, s_plus_deltaphi_2, s_minus_deltaphi_2,
191     s_plus_deltaphi_4, s_minus_deltaphi_4, A_plus_deltaphi,
192     A_minus_deltaphi, C_plus_deltaphi, C_minus_deltaphi,
193     psi_sk_plus_deltaphi, psi_sk_minus_deltaphi, r_S, x_s,
194     y_s, psi_plus_deltaphi, psi_minus_deltaphi, elemente,
195     alpha_R, l_3star, r_Gstar, psi_Gstar, x_Bikstar, y_Bikstar,
196     Mstar, Nstar, psistar, psistar_rad, x_B_versetztstar,
197     y_B_versetztstar, x_Bik_strichstar, y_Bik_strichstar,
198     Betrag_B_strichstar, r_Rstar, x_istar, y_istar, x_astar,
199     y_astar, x_i_versetztstar, y_i_versetztstar,
200     x_a_versetztstar, y_a_versetztstar, r_W, x_W, y_W,
201     drehWinkel, x_i_versetzt_old, y_i_versetzt_old,
202     x_B_versetzt_old, y_B_versetzt_old, x_a_versetzt_old,
203     y_a_versetzt_old, x_i_versetztstar_old, y_i_versetztstar_old,
204     x_B_versetztstar_old, y_B_versetztstar_old, x_a_versetztstar_old,
205     y_a_versetztstar_old;
206
207     double *z_phi;
208
209     long double *z_s, *z_psi, *z_psi_rad, *z_psi_strich_Grad,
210         *z_psi_zweistrich_Grad, *z_x_Bik, *z_y_Bik, *z_x_i_versetzt,
211         *z_y_i_versetzt, *z_x_B_versetzt, *z_y_B_versetzt, *z_x_a_versetzt,
212         *z_y_a_versetzt;
213
214     char s_richtung, Ba_Qudrant, positiv,
215         scheidendreh_richtung;
216
217     QString filename, filename_old, filename_asc, filename_innen,
218         filename_mitte, filename_aussen, text, dokufilename, configurefile,

```

```

219     dokupfad, texttwo, preferences, filename_s_phi, dt;
220
221     QTextBrowser* browser;
222
223     QMainWindow *mainWindow, *browserWindow;
224
225     QButtonGroup *buttonbox1, *buttonbox2, *buttonbox3, *buttonbox4,
226         *buttonbox5, *buttonbox6, *buttonboxAuswahl, *buttonboxkonturWahl,
227         *BG_Voreinstellungen, *BG_Skalierung;
228
229     QRadioButton *ja2, *nein2, *ja3, *nein3, *ja4, *nein4,
230         *ja1, *nein1,
231         *radiobuttonArbeitskurven,
232         *radiobuttonHub_geschw, *radiobuttonpsi_geschw_beschl,
233         *radiobuttonmue, *radiobuttonr_K, *radiobuttonx_Bik_x_Bikstrich,
234         *radiobuttony_Bik_y_Bikstrich, *radiobuttondurchmesser,
235         *radiobuttonGegenkurven, *konturAussenAussen, *konturAussenInnen,
236         *konturInnenAussen;
237
238     QCheckBox *CB_Dateiname, *CB_Arbeitskurven, *CB_s_phi, *CB_psi_phi,
239         *CB_Kurvenscheibe, *CB_x_Bik, *CB_y_Bik, *CB_Uebertragungswinkel,
240         *CB_Kruemmungsradius, *CB_Mittelpunktskurven;
241
242
243     QPushButton *konturFensterSchliessen;
244
245     QLabel *abfrage1, *abfrage2, *abfrage3, *abfrage4, *abfrage5,
246         *arbeitskurve, *labelH01, *labelH12, *labelH23, *labelH34,
247         *labelH45, *labelH56, *labelH67, *labelH78, *labelH89,
248         *labelH910, *labelH1011, *labelH1112, *labelAbschnitt1,
249         *labelAbschnitt2, *labelAbschnitt3, *labelAbschnitt4,
250         *labelAbschnitt5, *labelAbschnitt6, *labelAbschnitt7,
251         *labelAbschnitt8, *labelAbschnitt9, *labelAbschnitt10,
252         *labelAbschnitt11, *labelAbschnitt12, *labelsH01,
253         *labelsH12, *labelsH23, *labelsH34, *labelsH45, *labelsH56,
254         *labelsH67, *labelsH78, *labelsH89, *labelsH910, *abfrage6,
255         *labelsH1011, *labelsH1112, *labelalpha_R, *label1_3star,
256         *labelr_Rstar, *konturLabel, *konturLabel2, *konturLabel3,
257         *konturLabel4, *konturLabel5, *konturLabel6, *konturLabel7,
258         *konturLabel8, *konturLabelin, *QL_x_minArbkurven,
259         *QL_x_maxArbkurven, *QL_Arbeitskurven, *QL_dxArbkurven,
260         *QL_y_minArbkurven, *QL_y_maxArbkurven, *QL_dyArbkurven,
261         *QL_x_mins_phi, *QL_x_maxs_phi, *QL_s_phi, *QL_dxs_phi,
262         *QL_y_mins_phi, *QL_y_maxs_phi, *QL_dys_phi, *QL_x_minpsi_phi,

```

```

263     *QL_x_maxpsi_phi, *QL_psi_phi, *QL_dxpsi_phi,*QL_y_minpsi_phi,
264     *QL_y_maxpsi_phi, *QL_dypsi_phi, *QL_x_minKurvenscheibe,
265     *QL_x_maxKurvenscheibe, *QL_Kurvenscheibe, *QL_dxKurvenscheibe,
266     *QL_y_minKurvenscheibe, *QL_y_maxKurvenscheibe,
267     *QL_dyKurvenscheibe, *QL_x_minx_Bik, *QL_x_maxx_Bik,
268     *QL_x_Bik, *QL_dxx_Bik,*QL_y_minx_Bik, *QL_y_maxx_Bik, *QL_dyx_Bik,
269     *QL_x_miny_Bik, *QL_x_maxy_Bik, *QL_y_Bik, *QL_dxy_Bik,
270     *QL_y_miny_Bik, *QL_y_maxy_Bik, *QL_dyy_Bik,
271     *QL_x_minUebertragungswinkel, *QL_x_maxUebertragungswinkel,
272     *QL_Uebertragungswinkel, *QL_dxUebertragungswinkel,
273     *QL_y_minUebertragungswinkel, *QL_y_maxUebertragungswinkel,
274     *QL_dyUebertragungswinkel, *QL_x_minKruemmungsradius,
275     *QL_x_maxKruemmungsradius, *QL_Kruemmungsradius,
276     *QL_dxKruemmungsradius,*QL_y_minKruemmungsradius,
277     *QL_y_maxKruemmungsradius, *QL_dyKruemmungsradius,
278     *QL_x_minMittelpunktskurven, *QL_x_maxMittelpunktskurven,
279     *QL_Mittelpunktskurven, *QL_dxMittelpunktskurven,
280     *QL_y_minMittelpunktskurven, *QL_y_maxMittelpunktskurven,
281     *QL_dyMittelpunktskurven, *QL_PlotSkalierFaktor;
282
283     QComboBox *anzahlBewabschnitte, *comboAbschnitt1, *comboAbschnitt2,
284     *comboAbschnitt3, *comboAbschnitt4, *comboAbschnitt5,
285     *comboAbschnitt6, *comboAbschnitt7, *comboAbschnitt8,
286     *comboAbschnitt9, *comboAbschnitt10, *comboAbschnitt11,
287     *comboAbschnitt12, *ausfuehrungKurve;
288
289     QTime zeit;
290
291     QDate datum;
292
293     QWidget* firstpage;
294
295     /* Zeiger fuer dynamische Vektorfelder */
296
297     long double *phigrafisch, *x_B_verstzt, *y_B_verstzt,
298     *x_i_verstzt, *y_i_verstzt, *x_a_verstzt, *y_a_verstzt,
299     *sgrafisch, *s_strichgrafisch, *psigrafisch, *psi_strichgrafisch,
300     *psi_zweistrichgrafisch, *x_Bikgrafisch, *y_Bikgrafisch,
301     *x_Bik_strichgrafisch, *y_Bik_strichgrafisch, *x_sgrafisch,
302     *y_sgrafisch, *muegrafisch, *r_Kgrafisch, *x_Bikstargrafisch,
303     *y_Bikstargrafisch, *psistargrafisch, *psistar_radgrafisch,
304     *x_B_versetztstargrafisch, *y_B_versetztstargrafisch,
305     *x_Bik_strichstargrafisch, *y_Bik_strichstargrafisch,
306     *x_i_versetztstargrafisch, *y_i_versetztstargrafisch,

```



```

307     *x_a_versetztstargrafisch, *y_a_versetztstargrafisch,
308     *x_Wgrafisch, *y_Wgrafisch;
309
310     FunctionPlot* plotWindow;
311
312     Opticurv()
313     {
314         dateimenu = new QPopupMenu; //... erzeugt ein Datei-Menue
315         dateimenu->insertItem( "&Neu", this, SLOT( NeuSlot() ), CTRL+Key_N );
316         dateimenu->insertItem( "Ö&ffnen", this, SLOT( OeffnenSlot() ),
317                                CTRL+Key_O );
318         dateimenu->insertItem( "&Drucken", this, SLOT( print() ), CTRL+Key_P );
319         dateimenu->insertItem( "&Voreinstellungen", this,
320                                SLOT( VoreinstellungenSlot() ) );
321         dateimenu->insertItem( "S&chlieSSen", qApp, SLOT( quit() ), CTRL+Key_W );
322
323         ausgabemenu = new QPopupMenu; //... erzeugt ein Grafik-Menue
324         ausgabemenu->insertItem( "&Eingabeparameter", this,
325                                SLOT( ParameterSlot() ) );
326         AnzeigenID = ausgabemenu->insertItem( "&Grafik", this,
327                                                SLOT( GrafikSlot() ) );
328         ausgabemenu->setItemEnabled( AnzeigenID, FALSE );
329         ausgabemenu->insertItem( "&Tabelle", this, SLOT( TabelleSlot() ) );
330
331         hilfemenu = new QPopupMenu; //... erzeugt ein Hilfe-Menue
332         hilfemenu->insertItem( "&Manual", this, SLOT( HelpWindowSlot() ),
333                                Key_F1 );
334         hilfemenu->insertItem( "Ü&ber opticurv", this, SLOT( AboutSlot() ) );
335
336         menubalken = new QMenuBar( this ); //... erzeugt einen Menue-Balken
337         menubalken->insertItem( "&Datei", dateimenu );
338         menubalken->insertSeparator();
339         menubalken->insertItem( "&Ausgabe", ausgabemenu );
340         menubalken->insertSeparator();
341         menubalken->insertItem( "&Hilfe", hilfemenu );
342
343         printer = new QPrinter;
344
345         anzeige = new QTextView( this );
346         anzeige->setFocus();
347         this->setCentralWidget( anzeige );
348         statusBar()->message( "Opticurv Version 2.7.3 - TU DRESDEN" );
349
350         preferences = "preferences.conf";

```

```

351
352     dPlotSkalierFaktor = 1;
353
354
355     if( !preferences.isEmpty() )
356     {
357         double rEinstellungen[3];
358
359         QFile f(preferences);
360         f.open(IO_ReadOnly);
361         f.readBlock((char*)rEinstellungen,sizeof(rEinstellungen));
362         f.close();
363         dDateinameAbfragen=rEinstellungen[0];
364         dPlotSkalierFaktor=rEinstellungen[1];
365     }
366 }
367
368 ~Opticurv() { } // noch kein Code fuer den Destruktor
369
370 signals:
371
372 void comboDatenausDatei();
373
374 public slots:
375
376 void HelpWindowSlot()
377 {
378     browserWindow = new QMainWindow();
379     browserWindow->setGeometry( 0, 20, 700, 680 );
380     configurefile = "opticurv.conf";
381     QFile filepfad( configurefile );
382     if ( filepfad.open( IO_ReadOnly ) )
383     {
384         QTextStream streampfad( &filepfad );
385         dokupfad = streampfad.read();
386     }
387
388     browser = new QTextBrowser( browserWindow );
389     browser->mimeTypeFactory()->setExtensionType("html",
390         "text/html;charset=iso8859-1");
391     browser->setSource( dokupfad + "/index-1.html");
392     browser->setSource( dokupfad + "/index-2.html");
393     browser->setSource( dokupfad + "/index-3.html");
394     browser->setSource( dokupfad + "/index-4.html");

```

```

395         browser->setSource( dokupfad + "/index-5.html");
396         browser->setSource( dokupfad + "/index.html");
397
398         browserWindow->setCaption( "Online Manual" );
399         browserWindow->setCentralWidget( browser );
400         browserWindow->show();
401     }
402
403
404     protected:
405
406     QTextView* anzeige;
407
408     private slots:
409
410     void AboutSlot()
411     {
412         QMessageBox::information( this, "opticurv 1.1.0","Programm zur Auslegung"
413                                     " von\nKurvenkoppelgetrieben\n\n"
414                                     "Copyright 2003\n"
415                                     "Lutz Wirsig\n"
416                                     "lutz.wirsig@mailbox.tu-dresden.de");
417     }
418
419     void NeuSlot()
420     {
421         dphi_H01 = dphi_H12 = dphi_H23 = dphi_H34 = dphi_H45 = 1;
422         dphi_H56 = dphi_H67 = dphi_H78 = dphi_H89 = dphi_H910 = 1;
423         dphi_H1011 = dphi_H1112 = 1;
424         ds_H01 = ds_H12 = ds_H23 = ds_H34 = ds_H45 = 1;
425         ds_H56 = ds_H67 = ds_H78 = ds_H89 = ds_H910 = 1;
426         ds_H1011 = ds_H1112 = dalpha_R = dl_3star = dr_Rstar = 1;
427         dl_3 = dx_s0 = dy_s0 = dl_sk = dl_4 = 1;
428         dx_H = dy_H = dr_R = dx_A0 = dy_A0 = 1;
429         dbeta = dr_G = dn_1 = dr_S = dr_W =1;
430         dx_minArbkurven = dx_maxArbkurven = ddxArbkurven = 1;
431         dy_minArbkurven = dy_maxArbkurven = ddyArbkurven = 1;
432         dx_mins_phi = dx_maxs_phi = ddxs_phi = 1;
433         dy_mins_phi = dy_maxs_phi = ddys_phi = 1;
434         dx_minpsi_phi = dx_maxpsi_phi = ddxpsi_phi = 1;
435         dy_minpsi_phi = dy_maxpsi_phi = ddypsi_phi = 1;
436         dx_minKurvenscheibe = dx_maxKurvenscheibe = ddxKurvenscheibe = 1;
437         dy_minKurvenscheibe = dy_maxKurvenscheibe = ddyKurvenscheibe = 1;
438         dx_minx_Bik = dx_maxx_Bik = ddx_Bik = 1;

```

```

439     dy_minx_Bik = dy_maxx_Bik = ddyx_Bik = 1;
440     dx_miny_Bik = dx_maxy_Bik = ddx_Bik = 1;
441     dy_miny_Bik = dy_maxy_Bik = ddyy_Bik = 1;
442     dx_minUebertragungswinkel = dx_maxUebertragungswinkel = 1;
443     ddxUebertragungswinkel = dy_minUebertragungswinkel = 1;
444     dy_maxUebertragungswinkel = ddyUebertragungswinkel = 1;
445     dx_minKruemmungsradius = dx_maxKruemmungsradius = ddxKruemmungsradius = 1;
446     dy_minKruemmungsradius = dy_maxKruemmungsradius = ddyKruemmungsradius = 1;
447     dx_minMittelpunktskurven = dx_maxMittelpunktskurven = 1;
448     ddxMittelpunktskurven = dy_minMittelpunktskurven = 1;
449     dy_maxMittelpunktskurven = ddyMittelpunktskurven = 1;
450     dmanuelleArbeitskurven = dmanuelles_phi = dmanuellespsi_phi = 2;
451     dmanuelleKurvenscheibe = dmanuellex_Bik = dmanuelley_Bik = 2;
452     dmanuelleUebertragungswinkel = dmanuelleKruemmungsradius = 2;
453     dmanuelleMittelpunktskurven = 2;
454
455     void setFilter(const QString& ocv);
456
457     filename = QFileDialog::getSaveFileName( QString::null, "*.ocv", this );
458     if( !filename.isEmpty() )
459     {
460         filename_old = filename;
461         filename = filename + ".ocv";    // Endung .ocv an filename anhängen
462         filename_asc = filename_old + ".asc";    // Endung .asc an filename_asc
463                                         // anhängen
464         filename_innen = filename_old + "_innen.asc";
465         filename_mitte = filename_old + "_mitte.asc";
466         filename_aussen = filename_old + "_aussen.asc";
467         filename_s_phi = filename_old + "_s_phi.asc";
468
469         QFile ofl(filename);
470         ofl.open(IO_WriteOnly | IO_Truncate); // Datei für die Ausgabe öffnen
471         ofl.close();
472
473         QFile ofl_asc(filename_asc);
474         ofl_asc.open(IO_WriteOnly | IO_Truncate);
475         ofl_asc.close();
476
477         QFile ofl_asc2(filename_innen);
478         ofl_asc2.open(IO_WriteOnly | IO_Truncate);
479         ofl_asc2.close();
480
481         QFile ofl_asc3(filename_mitte);
482         ofl_asc3.open(IO_WriteOnly | IO_Truncate);

```

```

483     ofl_asc3.close();
484
485     QFile ofl_asc4(filename_aussen);
486     ofl_asc4.open(IO_WriteOnly | IO_Truncate);
487     ofl_asc4.close();
488
489     QFile ofl_asc5(filename_s_phi);
490     ofl_asc5.open(IO_WriteOnly | IO_Truncate);
491     ofl_asc5.close();
492
493     tabdialog = new QTabDialog(0,0,true);
494     tabdialog->setGeometry( 120, 120, 650, 450 );
495     tabdialog->setMinimumSize( 650, 450 );
496     tabdialog->setMaximumSize( 650, 450 );
497     tabdialog->setCaption( "Eingabeparameter" );
498     setupTab1();
499     setupTab2();
500     setupTab3();
501     setupTab4();
502     setup_ergAnz();
503     tabdialog->show();
504 }
505 }
506
507 void OeffnenSlot()
508 {
509     void setFilter(const QString& ocv);
510     filename = QFileDialog::getOpenFileName( QString::null, "*.ocv", this );
511     if( !filename.isEmpty() )
512     {
513         double rdaten[126];
514
515         QFile f(filename);
516         f.open(IO_ReadOnly);
517         f.readBlock((char*)rdaten,sizeof(rdaten));
518         f.close();
519         dphi_H01=rdaten[0];
520         dphi_H12=rdaten[1];
521         dphi_H23=rdaten[2];
522         dphi_H34=rdaten[3];
523         dphi_H45=rdaten[4];
524         dphi_H56=rdaten[5];
525         dphi_H67=rdaten[6];
526         dphi_H78=rdaten[7];

```

```

527     dphi_H89=rdaten[8];
528     dphi_H910=rdaten[9];
529     dphi_H1011=rdaten[10];
530     dphi_H1112=rdaten[11];
531     ds_H01=rdaten[12];
532     ds_H12=rdaten[13];
533     ds_H23=rdaten[14];
534     ds_H34=rdaten[15];
535     ds_H45=rdaten[16];
536     ds_H56=rdaten[17];
537     ds_H67=rdaten[18];
538     ds_H78=rdaten[19];
539     ds_H89=rdaten[20];
540     ds_H910=rdaten[21];
541     ds_H1011=rdaten[22];
542     ds_H1112=rdaten[23];
543     dl_3=rdaten[24];
544     dx_s0=rdaten[25];
545     dy_s0=rdaten[26];
546     dl_sk=rdaten[27];
547     dl_4=rdaten[28];
548     dx_H=rdaten[29];
549     dy_H=rdaten[30];
550     dr_R=rdaten[31];
551     dx_A0=rdaten[32];
552     dy_A0=rdaten[33];
553     dbeta=rdaten[34];
554     dr_G=rdaten[35];
555     dn_1=rdaten[36];
556     dri_scheibe=rdaten[37];
557     dhub_gleich=rdaten[38];
558     dquadrant=rdaten[39];
559     dpsi_positiv=rdaten[40];
560     dr_W=rdaten[41];
561     dr_S=rdaten[42];
562     dalpha_R=rdaten[43];
563     indexAbschnitt=(int)rdaten[44];
564     index1=(int)rdaten[45];
565     index2=(int)rdaten[46];
566     index3=(int)rdaten[47];
567     index4=(int)rdaten[48];
568     index5=(int)rdaten[49];
569     index6=(int)rdaten[50];
570     index7=(int)rdaten[51];

```

```

571     index8=(int)rdaten[52];
572     index9=(int)rdaten[53];
573     index10=(int)rdaten[54];
574     index11=(int)rdaten[55];
575     index12=(int)rdaten[56];
576     dl_3star=rdaten[57];
577     dr_Rstar=rdaten[58];
578     indexKurve=(int)rdaten[59];
579     indexKontur=(int)rdaten[60];
580     dmanuelleArbeitskurven=rdaten[62];
581     dmanuelles_phi=rdaten[63];
582     dmanuellespsi_phi=rdaten[64];
583     dmanuelleKurvenscheibe=rdaten[65];
584     dmanuellex_Bik=rdaten[66];
585     dmanuelley_Bik=rdaten[67];
586     dmanuelleUebertragungswinkel=rdaten[68];
587     dmanuelleKruemmungsradius=rdaten[69];
588     dx_minArbkurven=rdaten[70];
589     dx_maxArbkurven=rdaten[71];
590     ddxArbkurven=rdaten[72];
591     dy_minArbkurven=rdaten[73];
592     dy_maxArbkurven=rdaten[74];
593     ddyArbkurven=rdaten[75];
594     dx_mins_phi=rdaten[76];
595     dx_maxs_phi=rdaten[77];
596     ddxs_phi=rdaten[78];
597     dy_mins_phi=rdaten[79];
598     dy_maxs_phi=rdaten[80];
599     ddys_phi=rdaten[81];
600     dx_minpsi_phi=rdaten[82];
601     dx_maxpsi_phi=rdaten[83];
602     ddxpsi_phi=rdaten[84];
603     dy_minpsi_phi=rdaten[85];
604     dy_maxpsi_phi=rdaten[86];
605     ddypsi_phi=rdaten[87];
606     dx_minKurvenscheibe=rdaten[88];
607     dx_maxKurvenscheibe=rdaten[89];
608     ddxKurvenscheibe=rdaten[90];
609     dy_minKurvenscheibe=rdaten[91];
610     dy_maxKurvenscheibe=rdaten[92];
611     ddyKurvenscheibe=rdaten[93];
612     dx_minx_Bik=rdaten[94];
613     dx_maxx_Bik=rdaten[95];
614     ddxx_Bik=rdaten[96];

```

```

615     dy_minx_Bik=rdaten[97];
616     dy_maxx_Bik=rdaten[98];
617     ddyx_Bik=rdaten[99];
618     dx_miny_Bik=rdaten[100];
619     dx_maxy_Bik=rdaten[101];
620     ddxy_Bik=rdaten[102];
621     dy_miny_Bik=rdaten[103];
622     dy_maxy_Bik=rdaten[104];
623     ddyy_Bik=rdaten[105];
624     dx_minUebertragungswinkel=rdaten[106];
625     dx_maxUebertragungswinkel=rdaten[107];
626     ddxUebertragungswinkel=rdaten[108];
627     dy_minUebertragungswinkel=rdaten[109];
628     dy_maxUebertragungswinkel=rdaten[110];
629     ddyUebertragungswinkel=rdaten[111];
630     dx_minKruemmungsradius=rdaten[112];
631     dx_maxKruemmungsradius=rdaten[113];
632     ddxKruemmungsradius=rdaten[114];
633     dy_minKruemmungsradius=rdaten[115];
634     dy_maxKruemmungsradius=rdaten[116];
635     ddyKruemmungsradius=rdaten[117];
636     dmanuelleMittelpunktskurven=rdaten[118];
637     dx_minMittelpunktskurven=rdaten[119];
638     dx_maxMittelpunktskurven=rdaten[120];
639     ddxMittelpunktskurven=rdaten[121];
640     dy_minMittelpunktskurven=rdaten[122];
641     dy_maxMittelpunktskurven=rdaten[123];
642     ddyMittelpunktskurven=rdaten[124];
643
644     filename_old = filename;
645     filename_old.remove( filename_old.length()-4,4 );
646     filename_asc = filename_old + ".asc"; // Endung .asc an
647                                           // filename_asc anhängen
648     filename_innen = filename_old + "_innen.asc";
649     filename_mitte = filename_old + "_mitte.asc";
650     filename_aussen = filename_old + "_aussen.asc";
651     filename_s_phi = filename_old + "_s_phi.asc";
652
653     tabdialog = new QTabDialog(0,0,true);
654     tabdialog->setGeometry( 120, 120, 650, 450 );
655     tabdialog->setMinimumSize( 650, 450 );
656     tabdialog->setMaximumSize( 650, 450 );
657     tabdialog->setCaption( "Eingabeparameter" );
658     setupTab1();

```



```

659         gespeicherteWahl();
660         setupTab2();
661         setupTab3();
662         setupTab4();
663         setup_ergAnz();
664         tabdialog->show();
665     }
666 }
667 void VoreinstellungenSlot()
668 {
669     statusBar()->message( "Voreinstellungen...", 2000 );
670     einstellungendialog = new QTabDialog(0,0,true);
671     einstellungendialog->setGeometry( 120, 120, 450, 250 );
672     einstellungendialog->setMinimumSize( 450, 250 );
673     einstellungendialog->setMaximumSize( 450, 250 );
674     einstellungendialog->setCaption( "Voreinstellungen" );
675     setupEinst1();
676     einstellungendialog->show();
677 }
678
679 void ParameterSlot()
680 {
681     QString parameterText;
682     parameterText = texttwo;
683     anzeige->setText(parameterText);
684
685     statusBar()->message( "Opticurv Version 2.7.3 - TU DRESDEN" );
686 }
687
688 void TabelleSlot()
689 {
690     QString tabellenText;
691     tabellenText = text;
692     anzeige->setText(tabellenText);
693
694     statusBar()->message( "Opticurv Version 2.7.3 - TU DRESDEN" );
695 }
696
697 void GrafikSlot()
698 {
699     grafikAuswahl = new QWidget();
700
701     grafikAuswahl->setGeometry( 658, 150, 250, 500 );
702     grafikAuswahl->setMinimumSize( 250, 500 );

```

```

703 grafikAuswahl->setMaximumSize( 250, 500 );
704 grafikAuswahl->setCaption("Steuerfenster");
705
706 buttonboxAuswahl = new QButtonGroup( 1, Horizontal,
707                                     "Zeige folgende Funktion", grafikAuswahl);
708 buttonboxAuswahl->setGeometry( 18, 10, 214, 430 );
709 buttonboxAuswahl->setFrameStyle( QFrame::Panel | QFrame::Sunken );
710
711 arbeitskurve = new QLabel( buttonboxAuswahl );
712 arbeitskurve->setGeometry( 10, 10, 190, 10 );
713
714 if(indexKurve==0)      // nur bei Nutkurve
715 {
716     radiobuttonArbeitskurven = new QRadioButton( "Arbeitskurven",
717     buttonboxAuswahl );
718     radiobuttonArbeitskurven->setChecked( true );
719     radiobuttonHub_geschw = new QRadioButton( "s(phi); s'(phi)",
720     buttonboxAuswahl );
721     radiobuttonpsi_geschw_beschl = new QRadioButton(
722     "psi(phi); psi'(phi); psi''(phi)", buttonboxAuswahl );
723     radiobuttondurchmesser = new QRadioButton( "Kurvenscheibe",
724     buttonboxAuswahl );
725     radiobuttonx_Bik_x_Bikstrich = new QRadioButton(
726     "x_Bik(phi); x'_Bik(phi)", buttonboxAuswahl );
727     radiobuttoney_Bik_y_Bikstrich = new QRadioButton(
728     "y_Bik(phi); y'_Bik(phi)", buttonboxAuswahl );
729     radiobuttonmue = new QRadioButton( "Übertragungswinkel",
730     buttonboxAuswahl );
731     radiobuttonr_K = new QRadioButton( "Krümmungsradius",
732     buttonboxAuswahl );
733
734     connect( radiobuttonArbeitskurven, SIGNAL( clicked() ), this,
735     SLOT( plotSlot() ) );
736     connect( radiobuttonHub_geschw, SIGNAL( clicked() ), this,
737     SLOT( plotSlot() ) );
738     connect( radiobuttonpsi_geschw_beschl, SIGNAL( clicked() ), this,
739     SLOT( plotSlot() ) );
740     connect( radiobuttonx_Bik_x_Bikstrich, SIGNAL( clicked() ), this,
741     SLOT( plotSlot() ) );
742     connect( radiobuttoney_Bik_y_Bikstrich, SIGNAL( clicked() ), this,
743     SLOT( plotSlot() ) );
744     connect( radiobuttondurchmesser, SIGNAL( clicked() ), this,
745     SLOT( plotSlot() ) );
746     connect( radiobuttonmue, SIGNAL( clicked() ), this,

```

```

747         SLOT( plotSlot() ) );
748     connect( radiobuttonr_K, SIGNAL( clicked() ), this,
749         SLOT( plotSlot() ) );
750
751     plot0();
752 }
753
754 if(indexKurve==1)        // nur bei Doppelkurve
755 {
756     radiobuttonArbeitskurven = new QRadioButton( "Arbeits- u. Gegenkurven",
757         buttonboxAuswahl );
758     radiobuttonArbeitskurven->setChecked( true );
759     konturLabel = new QLabel( buttonboxAuswahl );
760     konturLabel->setGeometry( 10, 10, 10, 10 );
761     konturLabel2 = new QLabel( buttonboxAuswahl );
762     konturLabel2->setGeometry( 10, 10, 10, 10 );
763     konturLabel3 = new QLabel( buttonboxAuswahl );
764     konturLabel3->setGeometry( 10, 10, 10, 10 );
765     konturLabel4 = new QLabel( buttonboxAuswahl );
766     konturLabel4->setGeometry( 10, 10, 10, 10 );
767     konturLabel5 = new QLabel( buttonboxAuswahl );
768     konturLabel5->setGeometry( 10, 10, 10, 10 );
769     konturLabel6 = new QLabel( buttonboxAuswahl );
770     konturLabel6->setGeometry( 10, 10, 10, 10 );
771     konturLabel7 = new QLabel( buttonboxAuswahl );
772     konturLabel7->setGeometry( 10, 10, 10, 10 );
773     konturLabel8 = new QLabel( buttonboxAuswahl );
774     konturLabel8->setGeometry( 10, 10, 10, 10 );
775
776     buttonboxkonturWahl = new QButtonGroup( 1, Qt::Horizontal,
777         "Zeige Konturkombination", grafikAuswahl);
778     buttonboxkonturWahl->setGeometry( 33, 74, 185, 130 );
779     buttonboxkonturWahl->setFrameStyle( QFrame::NoFrame );
780
781     konturAussenAussen = new QRadioButton(
782         "Arbeitskurve: AuSSenkontur\nGegenkurve: AuSSenkontur",
783         buttonboxkonturWahl );
784     konturAussenAussen->setChecked( true );
785     konturAussenInnen = new QRadioButton(
786         "Arbeitskurve: AuSSenkontur\nGegenkurve: Innenkontur",
787         buttonboxkonturWahl );
788     konturInnenAussen = new QRadioButton(
789         "Arbeitskurve: Innenkontur\nGegenkurve: AuSSenkontur",
790         buttonboxkonturWahl );

```

```

791
792 radiobuttonHub_geschw = new QRadioButton( "s(phi); s'(phi)",
793     buttonboxAuswahl );
794 radiobuttonpsi_geschw_beschl = new QRadioButton(
795     "psi(phi); psi'(phi); psi''(phi)", buttonboxAuswahl );
796 radiobuttondurchmesser = new QRadioButton( "Kurvenscheibe",
797     buttonboxAuswahl );
798 radiobuttonx_Bik_x_Bikstrich = new QRadioButton(
799     "x_Bik(phi); x'_Bik(phi)", buttonboxAuswahl );
800 radiobuttony_Bik_y_Bikstrich = new QRadioButton(
801     "y_Bik(phi); y'_Bik(phi)", buttonboxAuswahl );
802 radiobuttonmue = new QRadioButton( "Übertragungswinkel",
803     buttonboxAuswahl );
804 radiobuttonr_K = new QRadioButton( "Krümmungsradius",
805     buttonboxAuswahl );
806 radiobuttonGegenkurven = new QRadioButton( "Mittelpunktskurven",
807     buttonboxAuswahl );
808
809 connect( radiobuttonArbeitskurven, SIGNAL( clicked() ), this,
810     SLOT( plotSlot() ) );
811 connect( radiobuttonHub_geschw, SIGNAL( clicked() ), this,
812     SLOT( plotSlot() ) );
813 connect( radiobuttonpsi_geschw_beschl, SIGNAL( clicked() ), this,
814     SLOT( plotSlot() ) );
815 connect( radiobuttonx_Bik_x_Bikstrich, SIGNAL( clicked() ), this,
816     SLOT( plotSlot() ) );
817 connect( radiobuttony_Bik_y_Bikstrich, SIGNAL( clicked() ), this,
818     SLOT( plotSlot() ) );
819 connect( radiobuttondurchmesser, SIGNAL( clicked() ), this,
820     SLOT( plotSlot() ) );
821 connect( radiobuttonmue, SIGNAL( clicked() ), this,
822     SLOT( plotSlot() ) );
823 connect( radiobuttonr_K, SIGNAL( clicked() ), this,
824     SLOT( plotSlot() ) );
825 connect( radiobuttonGegenkurven, SIGNAL( clicked() ), this,
826     SLOT( plotSlot() ) );
827 connect( konturAussenAussen, SIGNAL( clicked() ), this,
828     SLOT( plotKonturSlot() ) );
829 connect( konturAussenInnen, SIGNAL( clicked() ), this,
830     SLOT( plotKonturSlot() ) );
831 connect( konturInnenAussen, SIGNAL( clicked() ), this,
832     SLOT( plotKonturSlot() ) );
833
834 plot1();

```

```

835     }
836
837     QPushButton* grafikFensterSchliessen = new QPushButton( "S&chlieSSen",
838         grafikAuswahl);
839     grafikFensterSchliessen->setGeometry( 130, 455, 100, 30 );
840
841     QPushButton* grafikFensterDrucken = new QPushButton( "&Drucken",
842         grafikAuswahl);
843     grafikFensterDrucken->setGeometry( 15, 455, 100, 30 );
844
845     grafikAuswahl->show();
846
847     connect( grafikFensterSchliessen, SIGNAL( clicked() ), this,
848         SLOT( grafikfensterSchliessenSlot() ) );
849     connect( grafikFensterDrucken, SIGNAL( clicked() ), this,
850         SLOT( grafikfensterDruckenSlot() ) );
851 }
852
853 void plotSlot()
854 {
855     plotWindow->close();
856
857     if(indexKurve==0)        // nur bei Nutkurve
858     {
859         plot0();
860     }
861     if(indexKurve==1)        // nur bei Doppelkurve
862     {
863         plot1();
864     }
865 }
866
867 void plotKonturSlot()
868 {
869     plotWindow->close();
870
871     if ( konturAussenAussen->isChecked()||konturAussenInnen->isChecked()||
872         konturInnenAussen->isChecked() )
873     {
874         radiobuttonArbeitskurven->setChecked( true );
875     }
876     plot1();
877 }
878

```

```

879 void grafikfensterSchliessenSlot()
880 {
881     index_zuweisung = 0;
882     for(phi=0; phi<360; phi+=n_1)
883     {
884         index_zuweisung++;
885
886         phigrafisch[index_zuweisung] = 0;
887         x_B_verstzt[index_zuweisung] = 0;
888         y_B_verstzt[index_zuweisung] = 0;
889         x_i_verstzt[index_zuweisung] = 0;
890         y_i_verstzt[index_zuweisung] = 0;
891         x_a_verstzt[index_zuweisung] = 0;
892         y_a_verstzt[index_zuweisung] = 0;
893         sgrafisch[index_zuweisung] = 0;
894         s_strichgrafisch[index_zuweisung] = 0;
895         psigrafisch[index_zuweisung] = 0;
896         psi_strichgrafisch[index_zuweisung] = 0;
897         psi_zweistrichgrafisch[index_zuweisung] = 0;
898         x_Bikgrafisch[index_zuweisung] = 0;
899         y_Bikgrafisch[index_zuweisung] = 0;
900         x_Bik_strichgrafisch[index_zuweisung] = 0;
901         y_Bik_strichgrafisch[index_zuweisung] = 0;
902         x_sgrafisch[index_zuweisung] = 0;
903         y_sgrafisch[index_zuweisung] = 0;
904         x_Wgrafisch[index_zuweisung] = 0;
905         y_Wgrafisch[index_zuweisung] = 0;
906         muegrafisch[index_zuweisung] = 0;
907         r_Kgrafisch[index_zuweisung] = 0;
908
909         if(indexKurve==1)      // Zuweisung nur bei Doppelkurve
910         {
911             psistargrafisch[index_zuweisung] = 0;
912             x_Bikstargrafisch[index_zuweisung] = 0;
913             y_Bikstargrafisch[index_zuweisung] = 0;
914             x_Bik_strichstargrafisch[index_zuweisung] = 0;
915             y_Bik_strichstargrafisch[index_zuweisung] = 0;
916             x_B_versetztstargrafisch[index_zuweisung] = 0;
917             y_B_versetztstargrafisch[index_zuweisung] = 0;
918             x_i_versetztstargrafisch[index_zuweisung] = 0;
919             y_i_versetztstargrafisch[index_zuweisung] = 0;
920             x_a_versetztstargrafisch[index_zuweisung] = 0;
921             y_a_versetztstargrafisch[index_zuweisung] = 0;
922         }

```

```

923     }
924
925     free(phigrafisch);
926     free(x_B_verstzt);
927     free(y_B_verstzt);
928     free(x_i_verstzt);
929     free(y_i_verstzt);
930     free(x_a_verstzt);
931     free(y_a_verstzt);
932     free(sgrafisch);
933     free(s_strichgrafisch);
934     free(psigrafisch);
935     free(psi_strichgrafisch);
936     free(psi_zweistrichgrafisch);
937     free(x_Bikgrafisch);
938     free(y_Bikgrafisch);
939     free(x_Bik_strichgrafisch);
940     free(y_Bik_strichgrafisch);
941     free(x_sgrafisch);
942     free(y_sgrafisch);
943     free(x_Wgrafisch);
944     free(y_Wgrafisch);
945     free(muegrafisch);
946     free(r_Kgrafisch);
947     if(indexKurve==1)      // nur bei Doppelkurve
948     {
949         free(x_Bikstargrafisch);
950         free(y_Bikstargrafisch);
951         free(x_Bik_strichstargrafisch);
952         free(y_Bik_strichstargrafisch);
953         free(psistargrafisch);
954         free(psistar_radgrafisch);
955         free(x_B_versetztstargrafisch);
956         free(y_B_versetztstargrafisch);
957         free(x_i_versetztstargrafisch);
958         free(y_i_versetztstargrafisch);
959         free(x_a_versetztstargrafisch);
960         free(y_a_versetztstargrafisch);
961     }
962     plotWindow->close();
963     grafikAuswahl->close();
964     ausgabemenu->setItemEnabled( AnzeigenID, FALSE );
965 }
966

```

```

967 void grafikfensterDruckenSlot()
968 {
969     QString datumzeit;
970     datumzeit = datum.toString() + " - " + zeit.toString();
971
972     QString function;
973
974     if(radiobuttonArbeitskurven->isChecked())
975         function = "Grafikplot-Arbeitskurven";
976
977     if(radiobuttonHub_geschw->isChecked())
978         function = "Grafikplot-s(phi); s'(phi)";
979
980     if(radiobuttonpsi_geschw_beschl->isChecked())
981         function = "Grafikplot-psi(phi); psi'(phi); psi''(phi)";
982
983     if(radiobuttondurchmesser->isChecked())
984         function = "Grafikplot-Kurvenscheibe";
985
986     if(radiobuttonx_Bik_x_Bikstrich->isChecked())
987         function = "Grafikplot-x_Bik(phi); x'_Bik(phi)";
988
989     if(radiobuttony_Bik_y_Bikstrich->isChecked())
990         function = "Grafikplot-y_Bik(phi); y'_Bik(phi)";
991
992     if(radiobuttonmue->isChecked())
993         function = "Grafikplot-Übertragungswinkel";
994
995     if(radiobuttonr_K->isChecked())
996         function = "Grafikplot-Krümmungsradius";
997
998     plotWindow->druckeFunktion( filename, function, datumzeit );
999 }
1000
1001 void speichere()
1002 {
1003
1004     indexAbschnitt = anzahlBewabschnitte->currentItem();
1005     index1 = comboAbschnitt1->currentItem();
1006     index2 = comboAbschnitt2->currentItem();
1007     index3 = comboAbschnitt3->currentItem();
1008     index4 = comboAbschnitt4->currentItem();
1009     index5 = comboAbschnitt5->currentItem();
1010     index6 = comboAbschnitt6->currentItem();

```



```

1011     index7 = comboAbschnitt7->currentItem();
1012     index8 = comboAbschnitt8->currentItem();
1013     index9 = comboAbschnitt9->currentItem();
1014     index10 = comboAbschnitt10->currentItem();
1015     index11 = comboAbschnitt11->currentItem();
1016     index12 = comboAbschnitt12->currentItem();
1017     indexKurve = ausfuehrungKurve->currentItem();
1018
1019     sichereAbfrage();
1020
1021     if( dDateinameAbfragen == 1 )
1022     {
1023         saveAs();
1024     }
1025
1026
1027     QFile ofl(filename);
1028     ofl.open(IO_WriteOnly | IO_Truncate); // Datei für die Ausgabe öffnen
1029     ofl.close();
1030
1031     QFile ofl_asc(filename_asc);
1032     ofl_asc.open(IO_WriteOnly | IO_Truncate);
1033     ofl_asc.close();
1034
1035     QFile ofl_asc2(filename_innen);
1036     ofl_asc2.open(IO_WriteOnly | IO_Truncate);
1037
1038     ofl_asc2.close();
1039
1040     QFile ofl_asc3(filename_mitte);
1041     ofl_asc3.open(IO_WriteOnly | IO_Truncate);
1042     ofl_asc3.close();
1043
1044     QFile ofl_asc4(filename_aussen);
1045     ofl_asc4.open(IO_WriteOnly | IO_Truncate);
1046     ofl_asc4.close();
1047
1048     QFile ofl_asc5(filename_s_phi);
1049     ofl_asc5.open(IO_WriteOnly | IO_Truncate);
1050     ofl_asc5.close();
1051
1052     QFile f(filename);
1053     f.open(IO_WriteOnly);
1054     double wdaten[126] = {dphi_H01,dphi_H12,dphi_H23,dphi_H34,dphi_H45,

```

```

1055     dphi_H56,dphi_H67,dphi_H78,dphi_H89,dphi_H910,
1056     dphi_H1011,dphi_H1112,ds_H01,ds_H12,ds_H23,
1057     ds_H34,ds_H45,ds_H56,ds_H67,ds_H78,ds_H89,
1058     ds_H910,ds_H1011,ds_H1112,d1_3,dx_s0,dy_s0,
1059     d1_sk,d1_4,dx_H,dy_H,dr_R,dx_A0,dy_A0,dbeta,
1060     dr_G,dn_1,dri_scheibe,dhub_gleich,dquadrant,
1061     dpsi_positiv,dr_W,dr_S,dalpha_R,indexAbschnitt,
1062     index1,index2,index3,index4,index5,index6,
1063     index7,index8,index9,index10,index11,index12,
1064     d1_3star,dr_Rstar,indexKurve,indexKontur,
1065     indexKontur,dmanuelleArbeitskurven,dmanuelles_phi,
1066     dmanuellepsi_phi,dmanuelleKurvenscheibe,
1067     dmanuellex_Bik,dmanuelley_Bik,
1068     dmanuelleUebertragungswinkel,dmanuelleKruemmungsradius,
1069     dx_minArbkurven,dx_maxArbkurven,ddxArbkurven,
1070     dy_minArbkurven,dy_maxArbkurven,ddyArbkurven,
1071     dx_mins_phi,dx_maxs_phi,ddxs_phi,dy_mins_phi,
1072     dy_maxs_phi,ddys_phi,dx_minpsi_phi,dx_maxpsi_phi,
1073     ddxpsi_phi,dy_minpsi_phi,dy_maxpsi_phi,ddypsi_phi,
1074     dx_minKurvenscheibe,dx_maxKurvenscheibe,
1075     ddxKurvenscheibe,dy_minKurvenscheibe,
1076     dy_maxKurvenscheibe,ddyKurvenscheibe,dx_minx_Bik,
1077     dx_maxx_Bik,ddxx_Bik,dy_minx_Bik,dy_maxx_Bik,ddy_Bik,
1078     dx_miny_Bik,dx_maxy_Bik,ddxy_Bik,dy_miny_Bik,
1079     dy_maxy_Bik,ddy_Bik,dx_minUebertragungswinkel,
1080     dx_maxUebertragungswinkel,ddxUebertragungswinkel,
1081     dy_minUebertragungswinkel,dy_maxUebertragungswinkel,
1082     ddyUebertragungswinkel,dx_minKruemmungsradius,
1083     dx_maxKruemmungsradius,ddxKruemmungsradius,
1084     dy_minKruemmungsradius,dy_maxKruemmungsradius,
1085     ddyKruemmungsradius,dmanuelleMittelpunktskurven,
1086     dx_minMittelpunktskurven,dx_maxMittelpunktskurven,
1087     ddxMittelpunktskurven,dy_minMittelpunktskurven,
1088     dy_maxMittelpunktskurven,ddyMittelpunktskurven};
1089     f.writeBlock((char*)wdaten,sizeof(wdaten));
1090     f.close();
1091     calculate();
1092 }
1093
1094 void speicherePreferences()
1095 {
1096
1097     if( CB_Dateiname->isChecked() )
1098     {

```

```

1099     dDateinameAbfragen=1;
1100 }
1101 else if( !CB_Dateiname->isChecked() )
1102 {
1103     dDateinameAbfragen=2;
1104 }
1105
1106 QFile oflp(preferences);
1107 oflp.open(IO_WriteOnly | IO_Truncate); // Datei für die Ausgabe öffnen
1108 oflp.close();
1109
1110 QFile f(preferences);
1111 f.open(IO_WriteOnly);
1112 double wEinstellungen[3] = {dDateinameAbfragen,dPlotSkalierFaktor};
1113 f.writeBlock((char*)wEinstellungen,sizeof(wEinstellungen));
1114 f.close();
1115 }
1116
1117
1118 void neu_phi_H01( const QString& neuWert )
1119 {
1120     double phiH1 = neuWert.toDouble(&ok);
1121     if(ok == true)
1122         dphi_H01 = phiH1;
1123     berechne(phi_H01Eingabe);
1124 }
1125
1126 void neu_phi_H12( const QString& neuWert )
1127 {
1128     double phiH2 = neuWert.toDouble(&ok);
1129     if(ok == true)
1130         dphi_H12 = phiH2;
1131     berechne(phi_H12Eingabe);
1132 }
1133
1134 void neu_phi_H23( const QString& neuWert )
1135 {
1136     double phiH3 = neuWert.toDouble(&ok);
1137     if(ok == true)
1138         dphi_H23 = phiH3;
1139     berechne(phi_H23Eingabe);
1140 }
1141
1142 void neu_phi_H34( const QString& neuWert )

```

```

1143     {
1144         double phiH4 = neuWert.toDouble(&ok);
1145         if(ok == true)
1146             dphi_H34 = phiH4;
1147         berechne(phi_H34Eingabe);
1148     }
1149
1150 void neu_phi_H45( const QString& neuWert )
1151 {
1152     double phiH5 = neuWert.toDouble(&ok);
1153     if(ok == true)
1154         dphi_H45 = phiH5;
1155     berechne(phi_H45Eingabe);
1156 }
1157
1158 void neu_phi_H56( const QString& neuWert )
1159 {
1160     double phiH6 = neuWert.toDouble(&ok);
1161     if(ok == true)
1162         dphi_H56 = phiH6;
1163     berechne(phi_H56Eingabe);
1164 }
1165
1166 void neu_phi_H67( const QString& neuWert )
1167 {
1168     double phiH7 = neuWert.toDouble(&ok);
1169     if(ok == true)
1170         dphi_H67 = phiH7;
1171     berechne(phi_H67Eingabe);
1172 }
1173
1174 void neu_phi_H78( const QString& neuWert )
1175 {
1176     double phiH8 = neuWert.toDouble(&ok);
1177     if(ok == true)
1178         dphi_H78 = phiH8;
1179     berechne(phi_H78Eingabe);
1180 }
1181
1182 void neu_phi_H89( const QString& neuWert )
1183 {
1184     double phiH9 = neuWert.toDouble(&ok);
1185     if(ok == true)
1186         dphi_H89 = phiH9;

```

```

1187     berechne(phi_H89Eingabe);
1188 }
1189
1190 void neu_phi_H910( const QString& neuWert )
1191 {
1192     double phiH10 = neuWert.toDouble(&ok);
1193     if(ok == true)
1194         dphi_H910 = phiH10;
1195     berechne(phi_H910Eingabe);
1196 }
1197
1198 void neu_phi_H1011( const QString& neuWert )
1199 {
1200     double phiH11 = neuWert.toDouble(&ok);
1201     if(ok == true)
1202         dphi_H1011 = phiH11;
1203     berechne(phi_H1011Eingabe);
1204 }
1205
1206 void neu_phi_H1112( const QString& neuWert )
1207 {
1208     double phiH12 = neuWert.toDouble(&ok);
1209     if(ok == true)
1210         dphi_H1112 = phiH12;
1211     berechne(phi_H1112Eingabe);
1212 }
1213
1214 void neu_s_H01( const QString& neuWert )
1215 {
1216     double sH1 = neuWert.toDouble(&ok);
1217     if(ok == true)
1218         ds_H01 = sH1;
1219     berechne(s_H01Eingabe);
1220 }
1221
1222 void neu_s_H12( const QString& neuWert )
1223 {
1224     double sH2 = neuWert.toDouble(&ok);
1225     if(ok == true)
1226         ds_H12 = sH2;
1227     berechne(s_H12Eingabe);
1228 }
1229
1230 void neu_s_H23( const QString& neuWert )

```

```

1231     {
1232         double sH3 = neuWert.toDouble(&ok);
1233         if(ok == true)
1234             ds_H23 = sH3;
1235         berechne(s_H23Eingabe);
1236     }
1237
1238 void neu_s_H34( const QString& neuWert )
1239 {
1240     double sH4 = neuWert.toDouble(&ok);
1241     if(ok == true)
1242         ds_H34 = sH4;
1243     berechne(s_H34Eingabe);
1244 }
1245
1246 void neu_s_H45( const QString& neuWert )
1247 {
1248     double sH5 = neuWert.toDouble(&ok);
1249     if(ok == true)
1250         ds_H45 = sH5;
1251     berechne(s_H45Eingabe);
1252 }
1253
1254 void neu_s_H56( const QString& neuWert )
1255 {
1256     double sH6 = neuWert.toDouble(&ok);
1257     if(ok == true)
1258         ds_H56 = sH6;
1259     berechne(s_H56Eingabe);
1260 }
1261
1262 void neu_s_H67( const QString& neuWert )
1263 {
1264     double sH7 = neuWert.toDouble(&ok);
1265     if(ok == true)
1266         ds_H67 = sH7;
1267     berechne(s_H67Eingabe);
1268 }
1269
1270 void neu_s_H78( const QString& neuWert )
1271 {
1272     double sH8 = neuWert.toDouble(&ok);
1273     if(ok == true)
1274         ds_H78 = sH8;

```

```

1275     berechne(s_H78Eingabe);
1276 }
1277
1278 void neu_s_H89( const QString& neuWert )
1279 {
1280     double sH9 = neuWert.toDouble(&ok);
1281     if(ok == true)
1282         ds_H89 = sH9;
1283     berechne(s_H89Eingabe);
1284 }
1285
1286 void neu_s_H910( const QString& neuWert )
1287 {
1288     double sH10 = neuWert.toDouble(&ok);
1289     if(ok == true)
1290         ds_H910 = sH10;
1291     berechne(s_H910Eingabe);
1292 }
1293
1294 void neu_s_H1011( const QString& neuWert )
1295 {
1296     double sH11 = neuWert.toDouble(&ok);
1297     if(ok == true)
1298         ds_H1011 = sH11;
1299     berechne(s_H1011Eingabe);
1300 }
1301
1302 void neu_s_H1112( const QString& neuWert )
1303 {
1304     double sH12 = neuWert.toDouble(&ok);
1305     if(ok == true)
1306         ds_H1112 = sH12;
1307     berechne(s_H1112Eingabe);
1308 }
1309
1310 void neu_l_3( const QString& neuWert )
1311 {
1312     double l13 = neuWert.toDouble(&ok);
1313     if(ok == true)
1314         dl_3 = l13;
1315     berechne(l_3Eingabe);
1316 }
1317
1318 void neu_x_s0( const QString& neuWert )

```

```

1319     {
1320         double sH = neuWert.toDouble(&ok);
1321         if(ok == true)
1322             dx_s0 = sH;
1323         berechne(x_s0Eingabe);
1324     }
1325
1326 void neu_y_s0( const QString& neuWert )
1327 {
1328     double alphask = neuWert.toDouble(&ok);
1329     if(ok == true)
1330         dy_s0 = alphask;
1331     berechne(y_s0Eingabe);
1332 }
1333
1334 void neu_l_sk( const QString& neuWert )
1335 {
1336     double lsk = neuWert.toDouble(&ok);
1337     if(ok == true)
1338         dl_sk = lsk;
1339     berechne(l_skEingabe);
1340 }
1341
1342 void neu_l_4( const QString& neuWert )
1343 {
1344     double l4 = neuWert.toDouble(&ok);
1345     if(ok == true)
1346         dl_4 = l4;
1347     berechne(l_4Eingabe);
1348 }
1349 void neu_x_H( const QString& neuWert )
1350 {
1351     double xH = neuWert.toDouble(&ok);
1352     if(ok == true)
1353         dx_H = xH;
1354     berechne(x_HEingabe);
1355 }
1356
1357 void neu_y_H( const QString& neuWert )
1358 {
1359     double yH = neuWert.toDouble(&ok);
1360
1361     if(ok == true)
1362         dy_H = yH;

```



```

1363     berechne(y_HEingabe);
1364 }
1365
1366 void neu_r_R( const QString& neuWert )
1367 {
1368     double rR = neuWert.toDouble(&ok);
1369     if(ok == true)
1370         dr_R = rR;
1371     berechne(r_REingabe);
1372 }
1373
1374 void neu_x_A0( const QString& neuWert )
1375 {
1376     double xA0 = neuWert.toDouble(&ok);
1377     if(ok == true)
1378         dx_A0 = xA0;
1379     berechne(x_A0Eingabe);
1380 }
1381
1382 void neu_y_A0( const QString& neuWert )
1383 {
1384     double yA0 = neuWert.toDouble(&ok);
1385     if(ok == true)
1386         dy_A0 = yA0;
1387     berechne(y_A0Eingabe);
1388 }
1389
1390 void neu_beta( const QString& neuWert )
1391 {
1392     double bta = neuWert.toDouble(&ok);
1393     if(ok == true)
1394         dbeta = bta;
1395     berechne(betaEingabe);
1396 }
1397
1398 void neu_r_G( const QString& neuWert )
1399 {
1400     double rG = neuWert.toDouble(&ok);
1401     if(ok == true)
1402         dr_G = rG;
1403     berechne(r_GEingabe);
1404 }
1405
1406 void neu_n_1( const QString& neuWert )

```

```

1407     {
1408         double n1 = neuWert.toDouble(&ok);
1409         if(ok == true)
1410             dn_1 = n1;
1411         berechne(n_1Eingabe);
1412     }
1413
1414 void neu_r_S( const QString& neuWert )
1415 {
1416     double rS = neuWert.toDouble(&ok);
1417     if(ok == true)
1418         dr_S = rS;
1419     berechne(r_SEingabe);
1420 }
1421
1422 void neu_r_W( const QString& neuWert )
1423 {
1424     double rW = neuWert.toDouble(&ok);
1425     if(ok == true)
1426         dr_W = rW;
1427     berechne(r_WEingabe);
1428 }
1429
1430 void neu_alpha_R( const QString& neuWert )
1431 {
1432     double dalphaR = neuWert.toDouble(&ok);
1433     if(ok == true)
1434         dalpha_R = dalphaR;
1435     berechne(alpha_REingabe);
1436 }
1437
1438 void neu_l_3star( const QString& neuWert )
1439 {
1440     double dl3star = neuWert.toDouble(&ok);
1441     if(ok == true)
1442         dl_3star = dl3star;
1443     berechne(l_3starEingabe);
1444 }
1445
1446 void neu_r_Rstar( const QString& neuWert )
1447 {
1448     double drRstar = neuWert.toDouble(&ok);
1449     if(ok == true)
1450         dr_Rstar = drRstar;

```

```

1451     berechne(r_RstarEingabe);
1452 }
1453
1454 void neu_x_minArbkurven( const QString& neuWert )
1455 {
1456     double dxminArbkurven = neuWert.toDouble(&ok);
1457     if(ok == true)
1458         dx_minArbkurven = dxminArbkurven;
1459     berechne(QLE_x_minArbkurven);
1460 }
1461
1462 void neu_x_maxArbkurven( const QString& neuWert )
1463 {
1464     double dxmaxArbkurven = neuWert.toDouble(&ok);
1465     if(ok == true)
1466         dx_maxArbkurven = dxmaxArbkurven;
1467     berechne(QLE_x_maxArbkurven);
1468 }
1469
1470 void neu_dxArbkurven( const QString& neuWert )
1471 {
1472     double dxArbkurven = neuWert.toDouble(&ok);
1473     if(ok == true)
1474         ddxArbkurven = dxArbkurven;
1475     berechne(QLE_dxArbkurven);
1476 }
1477
1478 void neu_y_minArbkurven( const QString& neuWert )
1479 {
1480     double dyminArbkurven = neuWert.toDouble(&ok);
1481     if(ok == true)
1482         dy_minArbkurven = dyminArbkurven;
1483     berechne(QLE_y_minArbkurven);
1484 }
1485
1486 void neu_y_maxArbkurven( const QString& neuWert )
1487 {
1488     double dymaxArbkurven = neuWert.toDouble(&ok);
1489     if(ok == true)
1490         dy_maxArbkurven = dymaxArbkurven;
1491     berechne(QLE_x_maxArbkurven);
1492 }
1493
1494 void neu_dyArbkurven( const QString& neuWert )

```

```

1495     {
1496         double dyArbkurven = neuWert.toDouble(&ok);
1497         if(ok == true)
1498             ddyArbkurven = dyArbkurven;
1499         berechne(QLE_dyArbkurven);
1500     }
1501
1502 void neu_x_mins_phi( const QString& neuWert )
1503 {
1504     double dxmins_phi = neuWert.toDouble(&ok);
1505     if(ok == true)
1506         dx_mins_phi = dxmins_phi;
1507     berechne(QLE_x_mins_phi);
1508 }
1509
1510 void neu_x_maxs_phi( const QString& neuWert )
1511 {
1512     double dxmaxs_phi = neuWert.toDouble(&ok);
1513     if(ok == true)
1514         dx_maxs_phi = dxmaxs_phi;
1515     berechne(QLE_x_maxs_phi);
1516 }
1517
1518 void neu_dxs_phi( const QString& neuWert )
1519 {
1520     double dxs_phi = neuWert.toDouble(&ok);
1521     if(ok == true)
1522         ddxs_phi = dxs_phi;
1523     berechne(QLE_dxs_phi);
1524 }
1525
1526 void neu_y_mins_phi( const QString& neuWert )
1527 {
1528     double dymins_phi = neuWert.toDouble(&ok);
1529     if(ok == true)
1530         dy_mins_phi = dymins_phi;
1531     berechne(QLE_y_mins_phi);
1532 }
1533
1534 void neu_y_maxs_phi( const QString& neuWert )
1535 {
1536     double dymaxs_phi = neuWert.toDouble(&ok);
1537     if(ok == true)
1538         dy_maxs_phi = dymaxs_phi;

```

```

1539     berechne(QLE_x_maxs_phi);
1540 }
1541
1542 void neu_dys_phi( const QString& neuWert )
1543 {
1544     double dys_phi = neuWert.toDouble(&ok);
1545     if(ok == true)
1546         ddys_phi = dys_phi;
1547     berechne(QLE_dys_phi);
1548 }
1549
1550 void neu_x_minpsi_phi( const QString& neuWert )
1551 {
1552     double dxminpsi_phi = neuWert.toDouble(&ok);
1553     if(ok == true)
1554         dx_minpsi_phi = dxminpsi_phi;
1555     berechne(QLE_x_minpsi_phi);
1556 }
1557
1558 void neu_x_maxpsi_phi( const QString& neuWert )
1559 {
1560     double dxmaxpsi_phi = neuWert.toDouble(&ok);
1561     if(ok == true)
1562         dx_maxpsi_phi = dxmaxpsi_phi;
1563     berechne(QLE_x_maxpsi_phi);
1564 }
1565
1566 void neu_dxpsi_phi( const QString& neuWert )
1567 {
1568     double dxpsi_phi = neuWert.toDouble(&ok);
1569     if(ok == true)
1570         ddxpsi_phi = dxpsi_phi;
1571     berechne(QLE_dxpsi_phi);
1572 }
1573
1574 void neu_y_minpsi_phi( const QString& neuWert )
1575 {
1576     double dyminpsi_phi = neuWert.toDouble(&ok);
1577     if(ok == true)
1578         dy_minpsi_phi = dyminpsi_phi;
1579     berechne(QLE_y_minpsi_phi);
1580 }
1581
1582 void neu_y_maxpsi_phi( const QString& neuWert )

```

```

1583     {
1584         double dymaxpsi_phi = neuWert.toDouble(&ok);
1585         if(ok == true)
1586             dy_maxpsi_phi = dymaxpsi_phi;
1587         berechne(QLE_x_maxpsi_phi);
1588     }
1589
1590 void neu_dypsi_phi( const QString& neuWert )
1591 {
1592     double dypsi_phi = neuWert.toDouble(&ok);
1593     if(ok == true)
1594         ddypsi_phi = dypsi_phi;
1595     berechne(QLE_dypsi_phi);
1596 }
1597
1598 void neu_x_minKurvenscheibe( const QString& neuWert )
1599 {
1600     double dxminKurvenscheibe = neuWert.toDouble(&ok);
1601     if(ok == true)
1602         dx_minKurvenscheibe = dxminKurvenscheibe;
1603     berechne(QLE_x_minKurvenscheibe);
1604 }
1605
1606 void neu_x_maxKurvenscheibe( const QString& neuWert )
1607 {
1608     double dxmaxKurvenscheibe = neuWert.toDouble(&ok);
1609     if(ok == true)
1610         dx_maxKurvenscheibe = dxmaxKurvenscheibe;
1611     berechne(QLE_x_maxKurvenscheibe);
1612 }
1613
1614 void neu_dxKurvenscheibe( const QString& neuWert )
1615 {
1616     double dxKurvenscheibe = neuWert.toDouble(&ok);
1617     if(ok == true)
1618         ddxKurvenscheibe = dxKurvenscheibe;
1619     berechne(QLE_dxKurvenscheibe);
1620 }
1621
1622 void neu_y_minKurvenscheibe( const QString& neuWert )
1623 {
1624     double dyminKurvenscheibe = neuWert.toDouble(&ok);
1625     if(ok == true)
1626         dy_minKurvenscheibe = dyminKurvenscheibe;

```

```

1627     berechne(QLE_y_minKurvenscheibe);
1628 }
1629
1630 void neu_y_maxKurvenscheibe( const QString& neuWert )
1631 {
1632     double dymaxKurvenscheibe = neuWert.toDouble(&ok);
1633     if(ok == true)
1634         dy_maxKurvenscheibe = dymaxKurvenscheibe;
1635     berechne(QLE_x_maxKurvenscheibe);
1636 }
1637
1638 void neu_dyKurvenscheibe( const QString& neuWert )
1639 {
1640     double dyKurvenscheibe = neuWert.toDouble(&ok);
1641     if(ok == true)
1642         ddyKurvenscheibe = dyKurvenscheibe;
1643     berechne(QLE_dyKurvenscheibe);
1644 }
1645
1646 void neu_x_minx_Bik( const QString& neuWert )
1647 {
1648     double dxminx_Bik = neuWert.toDouble(&ok);
1649     if(ok == true)
1650         dx_minx_Bik = dxminx_Bik;
1651     berechne(QLE_x_minx_Bik);
1652 }
1653
1654 void neu_x_maxx_Bik( const QString& neuWert )
1655 {
1656     double dxmaxx_Bik = neuWert.toDouble(&ok);
1657     if(ok == true)
1658         dx_maxx_Bik = dxmaxx_Bik;
1659     berechne(QLE_x_maxx_Bik);
1660 }
1661
1662 void neu_dxx_Bik( const QString& neuWert )
1663 {
1664     double dxx_Bik = neuWert.toDouble(&ok);
1665     if(ok == true)
1666         ddx_Bik = dxx_Bik;
1667     berechne(QLE_dxx_Bik);
1668 }
1669
1670 void neu_y_minx_Bik( const QString& neuWert )

```

```

1671     {
1672         double dyminx_Bik = neuWert.toDouble(&ok);
1673         if(ok == true)
1674             dy_minx_Bik = dyminx_Bik;
1675         berechne(QLE_y_minx_Bik);
1676     }
1677
1678 void neu_y_maxx_Bik( const QString& neuWert )
1679 {
1680     double dymaxx_Bik = neuWert.toDouble(&ok);
1681     if(ok == true)
1682         dy_maxx_Bik = dymaxx_Bik;
1683     berechne(QLE_x_maxx_Bik);
1684 }
1685
1686 void neu_dyx_Bik( const QString& neuWert )
1687 {
1688     double dyx_Bik = neuWert.toDouble(&ok);
1689     if(ok == true)
1690         ddyx_Bik = dyx_Bik;
1691     berechne(QLE_dyx_Bik);
1692 }
1693
1694 void neu_x_miny_Bik( const QString& neuWert )
1695 {
1696     double dxminy_Bik = neuWert.toDouble(&ok);
1697     if(ok == true)
1698         dx_miny_Bik = dxminy_Bik;
1699     berechne(QLE_x_miny_Bik);
1700 }
1701
1702 void neu_x_maxy_Bik( const QString& neuWert )
1703 {
1704     double dxmaxy_Bik = neuWert.toDouble(&ok);
1705     if(ok == true)
1706         dx_maxy_Bik = dxmaxy_Bik;
1707     berechne(QLE_x_maxy_Bik);
1708 }
1709
1710 void neu_dxy_Bik( const QString& neuWert )
1711 {
1712     double dxy_Bik = neuWert.toDouble(&ok);
1713     if(ok == true)
1714         ddx_Bik = dxy_Bik;

```



```

1715     berechne(QLE_dxy_Bik);
1716 }
1717
1718 void neu_y_miny_Bik( const QString& neuWert )
1719 {
1720     double dyminy_Bik = neuWert.toDouble(&ok);
1721     if(ok == true)
1722         dy_miny_Bik = dyminy_Bik;
1723     berechne(QLE_y_miny_Bik);
1724 }
1725
1726 void neu_y_maxy_Bik( const QString& neuWert )
1727 {
1728     double dymaxy_Bik = neuWert.toDouble(&ok);
1729     if(ok == true)
1730         dy_maxy_Bik = dymaxy_Bik;
1731     berechne(QLE_x_maxy_Bik);
1732 }
1733
1734 void neu_dyy_Bik( const QString& neuWert )
1735 {
1736     double dyy_Bik = neuWert.toDouble(&ok);
1737     if(ok == true)
1738         ddy_Bik = dyy_Bik;
1739     berechne(QLE_dyy_Bik);
1740 }
1741
1742 void neu_x_minUebertragungswinkel( const QString& neuWert )
1743 {
1744     double dxminUebertragungswinkel = neuWert.toDouble(&ok);
1745     if(ok == true)
1746         dx_minUebertragungswinkel = dxminUebertragungswinkel;
1747     berechne(QLE_x_minUebertragungswinkel);
1748 }
1749
1750 void neu_x_maxUebertragungswinkel( const QString& neuWert )
1751 {
1752     double dxmaxUebertragungswinkel = neuWert.toDouble(&ok);
1753     if(ok == true)
1754         dx_maxUebertragungswinkel = dxmaxUebertragungswinkel;
1755     berechne(QLE_x_maxUebertragungswinkel);
1756 }
1757
1758 void neu_dxUebertragungswinkel( const QString& neuWert )

```

```

1759     {
1760         double dxUebertragungswinkel = neuWert.toDouble(&ok);
1761         if(ok == true)
1762             ddxUebertragungswinkel = dxUebertragungswinkel;
1763         berechne(QLE_dxUebertragungswinkel);
1764     }
1765
1766 void neu_y_minUebertragungswinkel( const QString& neuWert )
1767 {
1768     double dyminUebertragungswinkel = neuWert.toDouble(&ok);
1769     if(ok == true)
1770         dy_minUebertragungswinkel = dyminUebertragungswinkel;
1771     berechne(QLE_y_minUebertragungswinkel);
1772 }
1773
1774 void neu_y_maxUebertragungswinkel( const QString& neuWert )
1775 {
1776     double dymaxUebertragungswinkel = neuWert.toDouble(&ok);
1777     if(ok == true)
1778         dy_maxUebertragungswinkel = dymaxUebertragungswinkel;
1779     berechne(QLE_x_maxUebertragungswinkel);
1780 }
1781
1782 void neu_dyUebertragungswinkel( const QString& neuWert )
1783 {
1784     double dyUebertragungswinkel = neuWert.toDouble(&ok);
1785     if(ok == true)
1786         ddyUebertragungswinkel = dyUebertragungswinkel;
1787     berechne(QLE_dyUebertragungswinkel);
1788 }
1789
1790 void neu_x_minKruemmungsradius( const QString& neuWert )
1791 {
1792     double dxminKruemmungsradius = neuWert.toDouble(&ok);
1793     if(ok == true)
1794         dx_minKruemmungsradius = dxminKruemmungsradius;
1795     berechne(QLE_x_minKruemmungsradius);
1796 }
1797
1798 void neu_x_maxKruemmungsradius( const QString& neuWert )
1799 {
1800     double dxmaxKruemmungsradius = neuWert.toDouble(&ok);
1801     if(ok == true)
1802         dx_maxKruemmungsradius = dxmaxKruemmungsradius;

```

```

1803     berechne(QLE_x_maxKruemmungsradius);
1804 }
1805
1806 void neu_dxKruemmungsradius( const QString& neuWert )
1807 {
1808     double dxKruemmungsradius = neuWert.toDouble(&ok);
1809     if(ok == true)
1810         ddxKruemmungsradius = dxKruemmungsradius;
1811     berechne(QLE_dxKruemmungsradius);
1812 }
1813
1814 void neu_y_minKruemmungsradius( const QString& neuWert )
1815 {
1816     double dyminKruemmungsradius = neuWert.toDouble(&ok);
1817     if(ok == true)
1818         dy_minKruemmungsradius = dyminKruemmungsradius;
1819     berechne(QLE_y_minKruemmungsradius);
1820 }
1821
1822 void neu_y_maxKruemmungsradius( const QString& neuWert )
1823 {
1824     double dymaxKruemmungsradius = neuWert.toDouble(&ok);
1825     if(ok == true)
1826         dy_maxKruemmungsradius = dymaxKruemmungsradius;
1827     berechne(QLE_x_maxKruemmungsradius);
1828 }
1829
1830 void neu_dyKruemmungsradius( const QString& neuWert )
1831 {
1832     double dyKruemmungsradius = neuWert.toDouble(&ok);
1833     if(ok == true)
1834         ddyKruemmungsradius = dyKruemmungsradius;
1835     berechne(QLE_dyKruemmungsradius);
1836 }
1837
1838 void neu_dPlotSkalierFaktor( const QString& neuWert )
1839 {
1840     double ddPlotSkalierFaktor = neuWert.toDouble(&ok);
1841     if(ok == true)
1842         dPlotSkalierFaktor = ddPlotSkalierFaktor;
1843     berechneEinstellungen(QLE_PlotSkalierFaktor);
1844     speicherePreferences();
1845 }
1846

```

```

1847 void neu_x_minMittelpunktskurven( const QString& neuWert )
1848 {
1849     double dxminMittelpunktskurven = neuWert.toDouble(&ok);
1850     if(ok == true)
1851         dx_minMittelpunktskurven = dxminMittelpunktskurven;
1852     berechne(QLE_x_minMittelpunktskurven);
1853 }
1854
1855 void neu_x_maxMittelpunktskurven( const QString& neuWert )
1856 {
1857     double dxmaxMittelpunktskurven = neuWert.toDouble(&ok);
1858     if(ok == true)
1859         dx_maxMittelpunktskurven = dxmaxMittelpunktskurven;
1860     berechne(QLE_x_maxMittelpunktskurven);
1861 }
1862
1863 void neu_dxMittelpunktskurven( const QString& neuWert )
1864 {
1865     double dxMittelpunktskurven = neuWert.toDouble(&ok);
1866     if(ok == true)
1867         ddxMittelpunktskurven = dxMittelpunktskurven;
1868     berechne(QLE_dxMittelpunktskurven);
1869 }
1870
1871 void neu_y_minMittelpunktskurven( const QString& neuWert )
1872 {
1873     double dyminMittelpunktskurven = neuWert.toDouble(&ok);
1874     if(ok == true)
1875         dy_minMittelpunktskurven = dyminMittelpunktskurven;
1876     berechne(QLE_y_minMittelpunktskurven);
1877 }
1878
1879 void neu_y_maxMittelpunktskurven( const QString& neuWert )
1880 {
1881     double dymaxMittelpunktskurven = neuWert.toDouble(&ok);
1882     if(ok == true)
1883         dy_maxMittelpunktskurven = dymaxMittelpunktskurven;
1884     berechne(QLE_x_maxMittelpunktskurven);
1885 }
1886
1887 void neu_dyMittelpunktskurven( const QString& neuWert )
1888 {
1889     double dyMittelpunktskurven = neuWert.toDouble(&ok);
1890     if(ok == true)

```

```

1891         ddyMittelpunktskurven = dyMittelpunktskurven;
1892         berechne(QLE_dyMittelpunktskurven);
1893     }
1894
1895
1896     void zeigeWahl();
1897     void zeigeWahlSkalierung();
1898     void zeigeAlpha_R();
1899     void berechneHub1();
1900     void berechneHub2();
1901     void berechneHub3();
1902     void berechneHub4();
1903     void berechneHub5();
1904     void berechneHub6();
1905     void berechneHub7();
1906     void berechneHub8();
1907     void berechneHub9();
1908     void berechneHub10();
1909     void berechneHub11();
1910     void berechneHub12();
1911     void berechneHubaufrufen();
1912     void print();
1913
1914     private:
1915
1916     QCheckBox* _checkbox;
1917     QLabel *ergebnisAnzeige, *ergebnisAnzeigeLabel, *kontrollAnzeige;
1918     QLineEdit *phi_H01Eingabe, *phi_H12Eingabe, *phi_H23Eingabe,
1919         *phi_H34Eingabe, *phi_H45Eingabe, *phi_H56Eingabe,
1920         *phi_H67Eingabe, *phi_H78Eingabe, *phi_H89Eingabe,
1921         *phi_H910Eingabe, *phi_H1011Eingabe, *phi_H1112Eingabe,
1922         *l_3Eingabe, *x_s0Eingabe, *y_s0Eingabe, *l_skEingabe,
1923         *l_4Eingabe, *x_HEingabe, *y_HEingabe, *r_REingabe,
1924         *x_A0Eingabe, *y_A0Eingabe, *x_BaEingabe, *y_BaEingabe,
1925         *betaEingabe, *r_GEingabe, *n_1Eingabe, *r_SEingabe, *r_WEingabe,
1926         *s_H01Eingabe, *s_H12Eingabe, *s_H23Eingabe, *s_H34Eingabe,
1927         *s_H45Eingabe, *s_H56Eingabe, *s_H67Eingabe, *s_H78Eingabe,
1928         *s_H89Eingabe, *s_H910Eingabe, *s_H1011Eingabe,
1929         *s_H1112Eingabe, *alpha_REingabe, *l_3starEingabe,
1930         *r_RstarEingabe, *QLE_x_minArbkurven, *QLE_x_maxArbkurven,
1931         *QLE_dxArbkurven, *QLE_y_minArbkurven, *QLE_y_maxArbkurven,
1932         *QLE_dyArbkurven, *QLE_x_mins_phi, *QLE_x_maxs_phi,
1933         *QLE_dxs_phi, *QLE_y_mins_phi, *QLE_y_maxs_phi,
1934         *QLE_dys_phi, *QLE_x_minpsi_phi, *QLE_x_maxpsi_phi,

```

```

1935     *QLE_dxpsi_phi, *QLE_y_minpsi_phi, *QLE_y_maxpsi_phi,
1936     *QLE_dypsi_phi, *QLE_x_minKurvenscheibe, *QLE_x_maxKurvenscheibe,
1937     *QLE_dxKurvenscheibe, *QLE_y_minKurvenscheibe, *QLE_y_maxKurvenscheibe,
1938     *QLE_dyKurvenscheibe, *QLE_x_minx_Bik, *QLE_x_maxx_Bik,
1939     *QLE_dxx_Bik, *QLE_y_minx_Bik, *QLE_y_maxx_Bik, *QLE_dyx_Bik,
1940     *QLE_x_miny_Bik, *QLE_x_maxy_Bik, *QLE_dxy_Bik, *QLE_y_miny_Bik,
1941     *QLE_y_maxy_Bik, *QLE_dyy_Bik, *QLE_x_minUebertragungswinkel,
1942     *QLE_x_maxUebertragungswinkel, *QLE_dxUebertragungswinkel,
1943     *QLE_y_minUebertragungswinkel, *QLE_y_maxUebertragungswinkel,
1944     *QLE_dyUebertragungswinkel, *QLE_x_minKruemmungsradius,
1945     *QLE_x_maxKruemmungsradius, *QLE_dxKruemmungsradius,
1946     *QLE_y_minKruemmungsradius, *QLE_y_maxKruemmungsradius,
1947     *QLE_dyKruemmungsradius, *QLE_x_minMittelpunktskurven,
1948     *QLE_x_maxMittelpunktskurven, *QLE_dxMittelpunktskurven,
1949     *QLE_y_minMittelpunktskurven, *QLE_y_maxMittelpunktskurven,
1950     *QLE_dyMittelpunktskurven, *QLE_PlotSkalierFaktor;
1951 QMenuBar* menubalken;
1952 QMultiLineEdit* e;
1953 QPopupMenu *dateimenu, *ausgabemenu, *hilfemenu;
1954 QPrinter* printer;
1955 QScrollView* scrollview;
1956 QTabDialog* tabdialog, *einstellungendialog;
1957 QWidget *grafikAuswahl, *konturPlot;
1958
1959 bool ok;
1960
1961 void berechne( QLineEdit *eingabeWidget )
1962 {
1963     if(anzahlBewabschnitte->currentItem() == 0)
1964     {
1965         dergebnis = dphi_H01;
1966     }
1967     else if(anzahlBewabschnitte->currentItem() == 1)
1968     {
1969         dergebnis = dphi_H01 + dphi_H12;
1970     }
1971     else if(anzahlBewabschnitte->currentItem() == 2)
1972     {
1973         dergebnis = dphi_H01 + dphi_H12 + dphi_H23;
1974     }
1975     else if(anzahlBewabschnitte->currentItem() == 3)
1976     {
1977         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34;
1978     }

```

```

1979     else if(anzahlBewabschnitte->currentItem() == 3)
1980     {
1981         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34;
1982     }
1983     else if(anzahlBewabschnitte->currentItem() == 4)
1984     {
1985         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34 + dphi_H45;
1986     }
1987     else if(anzahlBewabschnitte->currentItem() == 5)
1988     {
1989         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34 + dphi_H45
1990                     + dphi_H56;
1991     }
1992     else
1993     {
1994         dergebnis = 1;
1995     }
1996
1997
1998     ergebnis = 1;
1999     if(ok == true)
2000     {
2001         QColor farbe( 0, 255, 255);
2002         ergebnisAnzeige->setNum(dergebnis);
2003         eingabeWidget->setPalette( QPalette( farbe, farbe ));
2004         ergebnisAnzeige->setPalette( QPalette( farbe, farbe ));
2005     }
2006     else
2007     {
2008         eingabeWidget->setPalette( QPalette( Qt::red, Qt::red ));
2009         ergebnisAnzeige->setPalette( QPalette( Qt::red, Qt::red ));
2010         ergebnisAnzeige->setText( "undefiniert" );
2011     }
2012     repaint();
2013 }
2014
2015 void berechneEinstellungen( QLineEdit *eingabeWidget )
2016 {
2017     if(ok == true)
2018     {
2019         QColor farbe( 0, 255, 255);
2020         eingabeWidget->setPalette( QPalette( farbe, farbe ));
2021     }
2022     else

```

```

2023     {
2024         eingabeWidget->setPalette( QPalette( Qt::red, Qt::red ));
2025     }
2026     repaint();
2027 }
2028
2029 void setupTab1();
2030 void setupTab2();
2031 void setupTab3();
2032 void setupTab4();
2033 void setupEinst1();
2034 void gespeicherteWahl();
2035 void sichereAbfrage();
2036 void setup_ergAnz();
2037 void calculate();
2038 void subcalculate();
2039 void speichereASCII();
2040 void weiseZu_konvertiereTyp();
2041 void ergebnisseMainWindow();
2042 void plot0();
2043 void plot1();
2044 void konturWahl();
2045 void plotArbeitsundGengenkurve();
2046 void saveAs();
2047 };
2048 void Opticurv::setupTab1()
2049 {
2050     firstpage = new QWidget( this );
2051     firstpage->setGeometry(10,10,730,430);
2052     tabdialog->addTab( firstpage, "Bewegungsverlauf");
2053     QColor farbe( 0, 255, 255 );
2054     ergebnis = 1;
2055
2056     QLabel* labelanzahlBewabschnitte = new QLabel(
2057         "Anzahl der Bewegungsabschnitte",firstpage );
2058     labelanzahlBewabschnitte->setGeometry( 20, 10, 190, 20);
2059     anzahlBewabschnitte = new QComboBox( false, firstpage );
2060     anzahlBewabschnitte->setGeometry( 220, 10, 50, 20 );
2061     for ( i=0; i<anzahlNo; i++)
2062         anzahlBewabschnitte->insertItem( anzahlAbschnitte[i] );
2063
2064     labelAbschnitt1 = new QLabel( "1. Abschnitt",firstpage );
2065     labelAbschnitt1->setGeometry( 20, 50, 100, 25);
2066     labelAbschnitt1->setEnabled(FALSE);

```



```

2067     comboAbschnitt1 = new QComboBox( false, firstpage );
2068     comboAbschnitt1->setGeometry( 100, 50, 120, 25 );
2069     for ( i=0; i<bewgesetzNo; i++)
2070     comboAbschnitt1->insertItem( bewegungsgesetz[i] );
2071     comboAbschnitt1->setEnabled(FALSE);
2072     labelH01 = new QLabel( "phi_H01      [Grad]",firstpage );
2073     labelH01->setGeometry( 240, 50, 120, 25);
2074     labelH01->setEnabled(FALSE);
2075     QToolTip::add( labelH01, "Gesamtdrehwinkel des 1. Bewegungsabschnittes" );
2076     phi_H01Eingabe = new QLineEdit( firstpage );
2077     phi_H01Eingabe->setGeometry( 350, 50, 80, 25);
2078     QString phi_H01ausgabe=phi_H01Eingabe->text();
2079     phi_H01ausgabe.setNum( dphi_H01 );
2080     phi_H01Eingabe->setText( phi_H01ausgabe );
2081     phi_H01Eingabe->setPalette( QPalette( farbe, farbe ) );
2082     phi_H01Eingabe->setEnabled(FALSE);
2083     labelsH01 = new QLabel( "s_H01      [mm]",firstpage );
2084     labelsH01->setGeometry( 440, 50, 100, 25);
2085     labelsH01->setEnabled(FALSE);
2086     QToolTip::add( labelsH01, "Gesamthub des 1. Bewegungsabschnittes" );
2087     s_H01Eingabe = new QLineEdit( firstpage );
2088     s_H01Eingabe->setGeometry( 530, 50, 80, 25);
2089     QString s_H01ausgabe=s_H01Eingabe->text();
2090     s_H01ausgabe.setNum( ds_H01 );
2091     s_H01Eingabe->setText( s_H01ausgabe );
2092     s_H01Eingabe->setPalette( QPalette( farbe, farbe ) );
2093     s_H01Eingabe->setEnabled(FALSE);
2094
2095     labelAbschnitt2 = new QLabel( "2. Abschnitt",firstpage );
2096     labelAbschnitt2->setGeometry( 20, 75, 100, 25);
2097     labelAbschnitt2->setEnabled(FALSE);
2098     comboAbschnitt2 = new QComboBox( false, firstpage );
2099     comboAbschnitt2->setGeometry( 100, 75, 120, 25 );
2100     for ( i=0; i<bewgesetzNo; i++)
2101     comboAbschnitt2->insertItem( bewegungsgesetz[i] );
2102     comboAbschnitt2->setEnabled(FALSE);
2103     labelH12 = new QLabel( "phi_H12      [Grad]",firstpage );
2104     labelH12->setGeometry( 240, 75, 120, 25);
2105     labelH12->setEnabled(FALSE);
2106     QToolTip::add( labelH12, "Gesamtdrehwinkel des 2. Bewegungsabschnittes" );
2107     phi_H12Eingabe = new QLineEdit( firstpage );
2108     phi_H12Eingabe->setGeometry( 350, 75, 80, 25);
2109     QString phi_H12ausgabe=phi_H12Eingabe->text();
2110     phi_H12ausgabe.setNum( dphi_H12 );

```

```

2111     phi_H12Eingabe->setText( phi_H12ausgabe );
2112     phi_H12Eingabe->setPalette( QPalette( farbe, farbe ) );
2113     phi_H12Eingabe->setEnabled(FALSE);
2114     labelsH12 = new QLabel( "s_H12      [mm]",firstpage );
2115     labelsH12->setGeometry( 440, 75, 100, 25);
2116     labelsH12->setEnabled(FALSE);
2117     QToolTip::add( labelsH12, "Gesamthub des 2. Bewegungsabschnittes" );
2118     s_H12Eingabe = new QLineEdit( firstpage );
2119     s_H12Eingabe->setGeometry( 530, 75, 80, 25);
2120     QString s_H12ausgabe=s_H12Eingabe->text();
2121     s_H12ausgabe.setNum( ds_H12 );
2122     s_H12Eingabe->setText( s_H12ausgabe );
2123     s_H12Eingabe->setPalette( QPalette( farbe, farbe ) );
2124     s_H12Eingabe->setEnabled(FALSE);
2125
2126     labelAbschnitt3 = new QLabel( "3. Abschnitt",firstpage );
2127     labelAbschnitt3->setGeometry( 20, 100, 100, 25);
2128     labelAbschnitt3->setEnabled(FALSE);
2129     comboAbschnitt3 = new QComboBox( false, firstpage );
2130     comboAbschnitt3->setGeometry( 100, 100, 120, 25 );
2131     for ( i=0; i<bewegesetzNo; i++)
2132         comboAbschnitt3->insertItem( bewegungsgesetz[i] );
2133     comboAbschnitt3->setEnabled(FALSE);
2134     labelH23 = new QLabel( "phi_H23      [Grad]",firstpage );
2135     labelH23->setGeometry( 240, 100, 120, 25);
2136     labelH23->setEnabled(FALSE);
2137     QToolTip::add( labelH23, "Gesamtdrehwinkel des 3. Bewegungsabschnittes" );
2138     phi_H23Eingabe = new QLineEdit( firstpage );
2139     phi_H23Eingabe->setGeometry( 350, 100, 80, 25);
2140     QString phi_H23ausgabe=phi_H23Eingabe->text();
2141     phi_H23ausgabe.setNum( dphi_H23 );
2142     phi_H23Eingabe->setText( phi_H23ausgabe );
2143     phi_H23Eingabe->setPalette( QPalette( farbe, farbe ) );
2144     phi_H23Eingabe->setEnabled(FALSE);
2145     labelsH23 = new QLabel( "s_H23      [mm]",firstpage );
2146     labelsH23->setGeometry( 440, 100, 100, 25);
2147     labelsH23->setEnabled(FALSE);
2148     QToolTip::add( labelsH23, "Gesamthub des 3. Bewegungsabschnittes" );
2149     s_H23Eingabe = new QLineEdit( firstpage );
2150     s_H23Eingabe->setGeometry( 530, 100, 80, 25);
2151     QString s_H23ausgabe=s_H23Eingabe->text();
2152     s_H23ausgabe.setNum( ds_H23 );
2153     s_H23Eingabe->setText( s_H23ausgabe );
2154     s_H23Eingabe->setPalette( QPalette( farbe, farbe ) );

```

```

2155         s_H23Eingabe->setEnabled(FALSE);
2156
2157     labelAbschnitt4 = new QLabel( "4. Abschnitt",firstpage );
2158     labelAbschnitt4->setGeometry( 20, 125, 100, 25);
2159     labelAbschnitt4->setEnabled(FALSE);
2160     labelAbschnitt4->setEnabled(FALSE);
2161     comboAbschnitt4 = new QComboBox( false, firstpage );
2162     comboAbschnitt4->setGeometry( 100, 125, 120, 25 );
2163     for ( i=0; i<bewegesetzNo; i++)
2164         comboAbschnitt4->insertItem( bewegungsgesetz[i] );
2165     comboAbschnitt4->setEnabled(FALSE);
2166     labelH34 = new QLabel( "phi_H34      [Grad]",firstpage );
2167     labelH34->setGeometry( 240, 125, 120, 25);
2168     labelH34->setEnabled(FALSE);
2169     QToolTip::add( labelH34, "Gesamtdrehwinkel des 4. Bewegungsabschnittes" );
2170     phi_H34Eingabe = new QLineEdit( firstpage );
2171     phi_H34Eingabe->setGeometry( 350, 125, 80, 25);
2172     QString phi_H34ausgabe=phi_H34Eingabe->text();
2173     phi_H34ausgabe.setNum( dphi_H34 );
2174     phi_H34Eingabe->setText( phi_H34ausgabe );
2175     phi_H34Eingabe->setPalette( QPalette( farbe, farbe ) );
2176     phi_H34Eingabe->setEnabled(FALSE);
2177     labelsH34 = new QLabel( "s_H34      [mm]",firstpage );
2178     labelsH34->setGeometry( 440, 125, 100, 25);
2179     labelsH34->setEnabled(FALSE);
2180     QToolTip::add( labelsH34, "Gesamthub des 4. Bewegungsabschnittes" );
2181     s_H34Eingabe = new QLineEdit( firstpage );
2182     s_H34Eingabe->setGeometry( 530, 125, 80, 25);
2183     QString s_H34ausgabe=s_H34Eingabe->text();
2184     s_H34ausgabe.setNum( ds_H34 );
2185     s_H34Eingabe->setText( s_H34ausgabe );
2186     s_H34Eingabe->setPalette( QPalette( farbe, farbe ) );
2187     s_H34Eingabe->setEnabled(FALSE);
2188
2189     labelAbschnitt5 = new QLabel( "5. Abschnitt",firstpage );
2190     labelAbschnitt5->setGeometry( 20, 150, 100, 25);
2191     labelAbschnitt5->setEnabled(FALSE);
2192     comboAbschnitt5 = new QComboBox( false, firstpage );
2193     comboAbschnitt5->setGeometry( 100, 150, 120, 25 );
2194     for ( i=0; i<bewegesetzNo; i++)
2195         comboAbschnitt5->insertItem( bewegungsgesetz[i] );
2196     comboAbschnitt5->setEnabled(FALSE);
2197     labelH45 = new QLabel( "phi_H45      [Grad]",firstpage );
2198     labelH45->setGeometry( 240, 150, 120, 25);

```

```

2199     labelH45->setEnabled(FALSE);
2200     QToolTip::add( labelH45, "Gesamtdrehwinkel des 5. Bewegungsabschnittes" );
2201     phi_H45Eingabe = new QLineEdit( firstpage );
2202     phi_H45Eingabe->setGeometry( 350, 150, 80, 25);
2203     QString phi_H45ausgabe=phi_H45Eingabe->text();
2204     phi_H45ausgabe.setNum( dphi_H45 );
2205     phi_H45Eingabe->setText( phi_H45ausgabe );
2206     phi_H45Eingabe->setPalette( QPalette( farbe, farbe ) );
2207     phi_H45Eingabe->setEnabled(FALSE);
2208     labelsH45 = new QLabel( "s_H45      [mm]",firstpage );
2209     labelsH45->setGeometry( 440, 150, 100, 25);
2210     labelsH45->setEnabled(FALSE);
2211     QToolTip::add( labelsH45, "Gesamthub des 5. Bewegungsabschnittes" );
2212     s_H45Eingabe = new QLineEdit( firstpage );
2213     s_H45Eingabe->setGeometry( 530, 150, 80, 25);
2214     QString s_H45ausgabe=s_H45Eingabe->text();
2215     s_H45ausgabe.setNum( ds_H45 );
2216     s_H45Eingabe->setText( s_H45ausgabe );
2217     s_H45Eingabe->setPalette( QPalette( farbe, farbe ) );
2218     s_H45Eingabe->setEnabled(FALSE);
2219
2220     labelAbschnitt6 = new QLabel( "6. Abschnitt",firstpage );
2221     labelAbschnitt6->setGeometry( 20, 175, 100, 25);
2222     labelAbschnitt6->setEnabled(FALSE);
2223     comboAbschnitt6 = new QComboBox( false, firstpage );
2224     comboAbschnitt6->setGeometry( 100, 175, 120, 25 );
2225     for ( i=0; i<bewgesetzNo; i++)
2226     comboAbschnitt6->insertItem( bewegungsgesetz[i] );
2227     comboAbschnitt6->setEnabled(FALSE);
2228     labelH56 = new QLabel( "phi_H56      [Grad]",firstpage );
2229     labelH56->setGeometry( 240, 175, 120, 25);
2230     labelH56->setEnabled(FALSE);
2231     QToolTip::add( labelH56, "Gesamtdrehwinkel des 6. Bewegungsabschnittes" );
2232     phi_H56Eingabe = new QLineEdit( firstpage );
2233     phi_H56Eingabe->setGeometry( 350, 175, 80, 25);
2234     QString phi_H56ausgabe=phi_H56Eingabe->text();
2235     phi_H56ausgabe.setNum( dphi_H56 );
2236     phi_H56Eingabe->setText( phi_H56ausgabe );
2237     phi_H56Eingabe->setPalette( QPalette( farbe, farbe ) );
2238     phi_H56Eingabe->setEnabled(FALSE);
2239     labelsH56 = new QLabel( "s_H56      [mm]",firstpage );
2240     labelsH56->setGeometry( 440, 175, 100, 25);
2241     labelsH56->setEnabled(FALSE);
2242     QToolTip::add( labelsH56, "Gesamthub des 6. Bewegungsabschnittes" );

```

```

2243     s_H56Eingabe = new QLineEdit( firstpage );
2244     s_H56Eingabe->setGeometry( 530, 175, 80, 25);
2245     QString s_H56ausgabe=s_H56Eingabe->text();
2246     s_H56ausgabe.setNum( ds_H56 );
2247     s_H56Eingabe->setText( s_H56ausgabe );
2248     s_H56Eingabe->setPalette( QPalette( farbe, farbe ) );
2249     s_H56Eingabe->setEnabled(FALSE);
2250
2251     labelAbschnitt7 = new QLabel( "7. Abschnitt",firstpage );
2252     labelAbschnitt7->setGeometry( 20, 200, 100, 25);
2253     labelAbschnitt7->setEnabled(FALSE);
2254     comboAbschnitt7 = new QComboBox( false, firstpage );
2255     comboAbschnitt7->setGeometry( 100, 200, 120, 25 );
2256     for ( i=0; i<bewegesetzNo; i++)
2257         comboAbschnitt7->insertItem( bewegungsgesetz[i] );
2258     comboAbschnitt7->setEnabled(FALSE);
2259     labelH67 = new QLabel( "phi_H67      [Grad]",firstpage );
2260     labelH67->setGeometry( 240, 200, 120, 25);
2261     labelH67->setEnabled(FALSE);
2262     QToolTip::add( labelH67, "Gesamtdrehwinkel des 7. Bewegungsabschnittes" );
2263     phi_H67Eingabe = new QLineEdit( firstpage );
2264     phi_H67Eingabe->setGeometry( 350, 200, 80, 25);
2265     QString phi_H67ausgabe=phi_H67Eingabe->text();
2266     phi_H67ausgabe.setNum( dphi_H67 );
2267     phi_H67Eingabe->setText( phi_H67ausgabe );
2268     phi_H67Eingabe->setPalette( QPalette( farbe, farbe ) );
2269     phi_H67Eingabe->setEnabled(FALSE);
2270     labelsH67 = new QLabel( "s_H67      [mm]",firstpage );
2271     labelsH67->setGeometry( 440, 200, 100, 25);
2272     labelsH67->setEnabled(FALSE);
2273     QToolTip::add( labelsH67, "Gesamthub des 7. Bewegungsabschnittes" );
2274     s_H67Eingabe = new QLineEdit( firstpage );
2275     s_H67Eingabe->setGeometry( 530, 200, 80, 25);
2276     QString s_H67ausgabe=s_H67Eingabe->text();
2277     s_H67ausgabe.setNum( ds_H67 );
2278     s_H67Eingabe->setText( s_H67ausgabe );
2279     s_H67Eingabe->setPalette( QPalette( farbe, farbe ) );
2280     s_H67Eingabe->setEnabled(FALSE);
2281
2282     labelAbschnitt8 = new QLabel( "8. Abschnitt",firstpage );
2283     labelAbschnitt8->setGeometry( 20, 225, 100, 25);
2284     labelAbschnitt8->setEnabled(FALSE);
2285     comboAbschnitt8 = new QComboBox( false, firstpage );
2286     comboAbschnitt8->setGeometry( 100, 225, 120, 25 );

```

```

2287         for ( i=0; i<bewgesetzNo; i++)
2288             comboAbschnitt8->insertItem( bewegungsgesetz[i] );
2289         comboAbschnitt8->setEnabled(FALSE);
2290         labelH78 = new QLabel( "phi_H78      [Grad]",firstpage );
2291         labelH78->setGeometry( 240, 225, 120, 25);
2292         labelH78->setEnabled(FALSE);
2293         QToolTip::add( labelH78, "Gesamtdrehwinkel des 8. Bewegungsabschnittes" );
2294         phi_H78Eingabe = new QLineEdit( firstpage );
2295         phi_H78Eingabe->setGeometry( 350, 225, 80, 25);
2296         QString phi_H78ausgabe=phi_H78Eingabe->text();
2297         phi_H78ausgabe.setNum( dphi_H78 );
2298         phi_H78Eingabe->setText( phi_H78ausgabe );
2299         phi_H78Eingabe->setPalette( QPalette( farbe, farbe ) );
2300         phi_H78Eingabe->setEnabled(FALSE);
2301         labelsH78 = new QLabel( "s_H78      [mm]",firstpage );
2302         labelsH78->setGeometry( 440, 225, 100, 25);
2303         labelsH78->setEnabled(FALSE);
2304         QToolTip::add( labelsH78, "Gesamthub des 8. Bewegungsabschnittes" );
2305         s_H78Eingabe = new QLineEdit( firstpage );
2306         s_H78Eingabe->setGeometry( 530, 225, 80, 25);
2307         QString s_H78ausgabe=s_H78Eingabe->text();
2308         s_H78ausgabe.setNum( ds_H78 );
2309         s_H78Eingabe->setText( s_H78ausgabe );
2310         s_H78Eingabe->setPalette( QPalette( farbe, farbe ) );
2311         s_H78Eingabe->setEnabled(FALSE);
2312
2313         labelAbschnitt9 = new QLabel( "9. Abschnitt",firstpage );
2314         labelAbschnitt9->setGeometry( 20, 250, 100, 25);
2315         labelAbschnitt9->setEnabled(FALSE);
2316         comboAbschnitt9 = new QComboBox( false, firstpage );
2317         comboAbschnitt9->setGeometry( 100, 250, 120, 25 );
2318         for ( i=0; i<bewgesetzNo; i++)
2319             comboAbschnitt9->insertItem( bewegungsgesetz[i] );
2320         comboAbschnitt9->setEnabled(FALSE);
2321         labelH89 = new QLabel( "phi_H89      [Grad]",firstpage );
2322         labelH89->setGeometry( 240, 250, 120, 25);
2323         labelH89->setEnabled(FALSE);
2324         QToolTip::add( labelH89, "Gesamtdrehwinkel des 9. Bewegungsabschnittes" );
2325         phi_H89Eingabe = new QLineEdit( firstpage );
2326         phi_H89Eingabe->setGeometry( 350, 250, 80, 25);
2327         QString phi_H89ausgabe=phi_H89Eingabe->text();
2328         phi_H89ausgabe.setNum( dphi_H89 );
2329         phi_H89Eingabe->setText( phi_H89ausgabe );
2330         phi_H89Eingabe->setPalette( QPalette( farbe, farbe ) );

```

```

2331     phi_H89Eingabe->setEnabled(FALSE);
2332     labelsH89 = new QLabel( "s_H89      [mm]",firstpage );
2333     labelsH89->setGeometry( 440, 250, 100, 25);
2334     labelsH89->setEnabled(FALSE);
2335     QToolTip::add( labelsH89, "Gesamthub des 9. Bewegungsabschnittes" );
2336     s_H89Eingabe = new QLineEdit( firstpage );
2337     s_H89Eingabe->setGeometry( 530, 250, 80, 25);
2338     QString s_H89ausgabe=s_H89Eingabe->text();
2339     s_H89ausgabe.setNum( ds_H89 );
2340     s_H89Eingabe->setText( s_H89ausgabe );
2341     s_H89Eingabe->setPalette( QPalette( farbe, farbe ) );
2342     s_H89Eingabe->setEnabled(FALSE);
2343
2344     labelAbschnitt10 = new QLabel( "10. Abschnitt",firstpage );
2345     labelAbschnitt10->setGeometry( 20, 275, 100, 25);
2346     labelAbschnitt10->setEnabled(FALSE);
2347     comboAbschnitt10 = new QComboBox( false, firstpage );
2348     comboAbschnitt10->setGeometry( 100, 275, 120, 25 );
2349     for ( i=0; i<bewegesetzNo; i++)
2350         comboAbschnitt10->insertItem( bewegungsgesetz[i] );
2351     comboAbschnitt10->setEnabled(FALSE);
2352     labelH910 = new QLabel( "phi_H910   [Grad]",firstpage );
2353     labelH910->setGeometry( 240, 275, 120, 25);
2354     labelH910->setEnabled(FALSE);
2355     QToolTip::add( labelH910, "Gesamtdrehwinkel des 10. Bewegungsabschnittes" );
2356     phi_H910Eingabe = new QLineEdit( firstpage );
2357     phi_H910Eingabe->setGeometry( 350, 275, 80, 25);
2358     QString phi_H910ausgabe=phi_H910Eingabe->text();
2359     phi_H910ausgabe.setNum( dphi_H910 );
2360     phi_H910Eingabe->setText( phi_H910ausgabe );
2361     phi_H910Eingabe->setPalette( QPalette( farbe, farbe ) );
2362     phi_H910Eingabe->setEnabled(FALSE);
2363     labelsH910 = new QLabel( "s_H910    [mm]",firstpage );
2364     labelsH910->setGeometry( 440, 275, 100, 25);
2365     labelsH910->setEnabled(FALSE);
2366     QToolTip::add( labelsH910, "Gesamthub des 10. Bewegungsabschnittes" );
2367     s_H910Eingabe = new QLineEdit( firstpage );
2368     s_H910Eingabe->setGeometry( 530, 275, 80, 25);
2369     QString s_H910ausgabe=s_H910Eingabe->text();
2370     s_H910ausgabe.setNum( ds_H910 );
2371     s_H910Eingabe->setText( s_H910ausgabe );
2372     s_H910Eingabe->setPalette( QPalette( farbe, farbe ) );
2373     s_H910Eingabe->setEnabled(FALSE);
2374

```

```

2375     labelAbschnitt11 = new QLabel( "11. Abschnitt",firstpage );
2376         labelAbschnitt11->setGeometry( 20, 300, 100, 25);
2377         labelAbschnitt11->setEnabled(FALSE);
2378     comboAbschnitt11 = new QComboBox( false, firstpage );
2379         comboAbschnitt11->setGeometry( 100, 300, 120, 25 );
2380         for ( i=0; i<bewgesetzNo; i++)
2381             comboAbschnitt11->insertItem( bewegungsgesetz[i] );
2382         comboAbschnitt11->setEnabled(FALSE);
2383     labelH1011 = new QLabel( "phi_H1011 [Grad]",firstpage );
2384         labelH1011->setGeometry( 240, 300, 120, 25);
2385         labelH1011->setEnabled(FALSE);
2386     QToolTip::add( labelH1011, "Gesamtdrehwinkel des 11. Bewegungsabschnittes" );
2387     phi_H1011Eingabe = new QLineEdit( firstpage );
2388         phi_H1011Eingabe->setGeometry( 350, 300, 80, 25);
2389         QString phi_H1011ausgabe=phi_H1011Eingabe->text();
2390         phi_H1011ausgabe.setNum( dphi_H1011 );
2391         phi_H1011Eingabe->setText( phi_H1011ausgabe );
2392         phi_H1011Eingabe->setPalette( QPalette( farbe, farbe ) );
2393         phi_H1011Eingabe->setEnabled(FALSE);
2394     labelsH1011 = new QLabel( "s_H1011 [mm]",firstpage );
2395         labelsH1011->setGeometry( 440, 300, 100, 25);
2396         labelsH1011->setEnabled(FALSE);
2397     QToolTip::add( labelsH1011, "Gesamthub des 11. Bewegungsabschnittes" );
2398     s_H1011Eingabe = new QLineEdit( firstpage );
2399         s_H1011Eingabe->setGeometry( 530, 300, 80, 25);
2400         QString s_H1011ausgabe=s_H1011Eingabe->text();
2401         s_H1011ausgabe.setNum( ds_H1011 );
2402         s_H1011Eingabe->setText( s_H1011ausgabe );
2403         s_H1011Eingabe->setPalette( QPalette( farbe, farbe ) );
2404         s_H1011Eingabe->setEnabled(FALSE);
2405
2406     labelAbschnitt12 = new QLabel( "12. Abschnitt",firstpage );
2407         labelAbschnitt12->setGeometry( 20, 325, 100, 25);
2408         labelAbschnitt12->setEnabled(FALSE);
2409     comboAbschnitt12 = new QComboBox( false, firstpage );
2410         comboAbschnitt12->setGeometry( 100, 325, 120, 25 );
2411         for ( i=0; i<bewgesetzNo; i++)
2412             comboAbschnitt12->insertItem( bewegungsgesetz[i] );
2413         comboAbschnitt12->setEnabled(FALSE);
2414     labelH1112 = new QLabel( "phi_H1112 [Grad]",firstpage );
2415         labelH1112->setGeometry( 240, 325, 120, 25);
2416         labelH1112->setEnabled(FALSE);
2417     QToolTip::add( labelH1112, "Gesamtdrehwinkel des 12. Bewegungsabschnittes" );
2418     phi_H1112Eingabe = new QLineEdit( firstpage );

```



```

2419     phi_H1112Eingabe->setGeometry( 350, 325, 80, 25);
2420     QString phi_H1112ausgabe=phi_H1112Eingabe->text();
2421     phi_H1112ausgabe.setNum( dphi_H1112 );
2422     phi_H1112Eingabe->setText( phi_H1112ausgabe );
2423     phi_H1112Eingabe->setPalette( QPalette( farbe, farbe ) );
2424     phi_H1112Eingabe->setEnabled(FALSE);
2425     labelsH1112 = new QLabel( "s_H1112 [mm]",firstpage );
2426     labelsH1112->setGeometry( 440, 325, 100, 25);
2427     labelsH1112->setEnabled(FALSE);
2428     QToolTip::add( labelsH1112, "Gesamthub des 12. Bewegungsabschnittes" );
2429     s_H1112Eingabe = new QLineEdit( firstpage );
2430     s_H1112Eingabe->setGeometry( 530, 325, 80, 25);
2431     QString s_H1112ausgabe=s_H1112Eingabe->text();
2432     s_H1112ausgabe.setNum( ds_H1112 );
2433     s_H1112Eingabe->setText( s_H1112ausgabe );
2434     s_H1112Eingabe->setPalette( QPalette( farbe, farbe ) );
2435     s_H1112Eingabe->setEnabled(FALSE);
2436
2437     QObject::connect( anzahlBewabschnitte, SIGNAL( activated( int ) ),
2438                      this, SLOT( zeigeWahl() ) );
2439
2440     QObject::connect(phi_H01Eingabe,
2441                     SIGNAL( textChanged( const QString & ) ),
2442                     this, SLOT( neu_phi_H01( const QString & ) ) );
2443
2444     QObject::connect(s_H01Eingabe,
2445                     SIGNAL( textChanged( const QString & ) ),
2446                     this, SLOT( neu_s_H01( const QString & ) ) );
2447
2448     QObject::connect(phi_H12Eingabe,
2449                     SIGNAL( textChanged( const QString & ) ),
2450                     this, SLOT( neu_phi_H12( const QString & ) ) );
2451
2452     QObject::connect(s_H12Eingabe,
2453                     SIGNAL( textChanged( const QString & ) ),
2454                     this, SLOT( neu_s_H12( const QString & ) ) );
2455
2456     QObject::connect(phi_H23Eingabe,
2457                     SIGNAL( textChanged( const QString & ) ),
2458                     this, SLOT( neu_phi_H23( const QString & ) ) );
2459
2460     QObject::connect(s_H23Eingabe,
2461                     SIGNAL( textChanged( const QString & ) ),
2462                     this, SLOT( neu_s_H23( const QString & ) ) );

```

```

2463
2464 QObject::connect(phi_H34Eingabe,
2465                 SIGNAL( textChanged( const QString & ) ),
2466                 this, SLOT( neu_phi_H34( const QString & ) ) );
2467
2468 QObject::connect(s_H34Eingabe,
2469                 SIGNAL( textChanged( const QString & ) ),
2470                 this, SLOT( neu_s_H34( const QString & ) ) );
2471
2472 QObject::connect(phi_H45Eingabe,
2473                 SIGNAL( textChanged( const QString & ) ),
2474                 this, SLOT( neu_phi_H45( const QString & ) ) );
2475
2476 QObject::connect(s_H45Eingabe,
2477                 SIGNAL( textChanged( const QString & ) ),
2478                 this, SLOT( neu_s_H45( const QString & ) ) );
2479
2480 QObject::connect(phi_H56Eingabe,
2481                 SIGNAL( textChanged( const QString & ) ),
2482                 this, SLOT( neu_phi_H56( const QString & ) ) );
2483
2484 QObject::connect(s_H56Eingabe,
2485                 SIGNAL( textChanged( const QString & ) ),
2486                 this, SLOT( neu_s_H56( const QString & ) ) );
2487
2488 QObject::connect(phi_H67Eingabe,
2489                 SIGNAL( textChanged( const QString & ) ),
2490                 this, SLOT( neu_phi_H67( const QString & ) ) );
2491
2492 QObject::connect(s_H67Eingabe,
2493                 SIGNAL( textChanged( const QString & ) ),
2494                 this, SLOT( neu_s_H67( const QString & ) ) );
2495
2496 QObject::connect(phi_H78Eingabe,
2497                 SIGNAL( textChanged( const QString & ) ),
2498                 this, SLOT( neu_phi_H78( const QString & ) ) );
2499
2500 QObject::connect(s_H78Eingabe,
2501                 SIGNAL( textChanged( const QString & ) ),
2502                 this, SLOT( neu_s_H78( const QString & ) ) );
2503
2504 QObject::connect(phi_H89Eingabe,
2505                 SIGNAL( textChanged( const QString & ) ),
2506                 this, SLOT( neu_phi_H89( const QString & ) ) );

```

```

2507
2508     QObject::connect(s_H89Eingabe,
2509                     SIGNAL( textChanged( const QString & ) ),
2510                     this, SLOT( neu_s_H89( const QString & ) ) );
2511
2512     QObject::connect(phi_H910Eingabe,
2513                     SIGNAL( textChanged( const QString & ) ),
2514                     this, SLOT( neu_phi_H910( const QString & ) ) );
2515
2516     QObject::connect(s_H910Eingabe,
2517                     SIGNAL( textChanged( const QString & ) ),
2518                     this, SLOT( neu_s_H910( const QString & ) ) );
2519
2520     QObject::connect(phi_H1011Eingabe,
2521                     SIGNAL( textChanged( const QString & ) ),
2522                     this, SLOT( neu_phi_H1011( const QString & ) ) );
2523
2524     QObject::connect(s_H1011Eingabe,
2525                     SIGNAL( textChanged( const QString & ) ),
2526                     this, SLOT( neu_s_H1011( const QString & ) ) );
2527
2528     QObject::connect(phi_H1112Eingabe,
2529                     SIGNAL( textChanged( const QString & ) ),
2530                     this, SLOT( neu_phi_H1112( const QString & ) ) );
2531
2532     QObject::connect(s_H1112Eingabe,
2533                     SIGNAL( textChanged( const QString & ) ),
2534                     this, SLOT( neu_s_H1112( const QString & ) ) );
2535 }
2536
2537 void Opticurv::gespeicherteWahl()
2538 {
2539     /* aus Datei gelesene Indizes den Comboboxen zuweisen und anzeigen */
2540
2541     anzahlBewabschnitte->setCurrentItem( indexAbschnitt );
2542     comboAbschnitt1->setCurrentItem( index1 );
2543     comboAbschnitt2->setCurrentItem( index2 );
2544     comboAbschnitt3->setCurrentItem( index3 );
2545     comboAbschnitt4->setCurrentItem( index4 );
2546     comboAbschnitt5->setCurrentItem( index5 );
2547     comboAbschnitt6->setCurrentItem( index6 );
2548     comboAbschnitt7->setCurrentItem( index7 );
2549     comboAbschnitt8->setCurrentItem( index8 );
2550     comboAbschnitt9->setCurrentItem( index9 );

```

```

2551     comboAbschnitt10->setCurrentItem( index10 );
2552     comboAbschnitt11->setCurrentItem( index11 );
2553     comboAbschnitt12->setCurrentItem( index12 );
2554
2555     zeigeWahl();
2556 }
2557 void Opticurv::zeigeWahl()
2558 {
2559     /* Trick, dass Gesamtgradzahl bei Betaetigen
2560        der Combobox aktualisiert wird */
2561
2562     ergebnis = 1;
2563     QColor farbe( 0, 255, 255 );
2564     QString str( anzahlBewabschnitte->currentText() );
2565     bool ok;
2566     int dec = str.toInt( &ok, 10 );
2567
2568     if(dec==1)
2569     {
2570
2571         labelAbschnitt1->setEnabled(TRUE);
2572         comboAbschnitt1->setEnabled(TRUE);
2573         labelH01->setEnabled(TRUE);
2574         phi_H01Eingabe->setEnabled(TRUE);
2575         labelsH01->setEnabled(TRUE);
2576         s_H01Eingabe->setEnabled(TRUE);
2577
2578         labelAbschnitt2->setEnabled(FALSE);
2579         comboAbschnitt2->setEnabled(FALSE);
2580         labelH12->setEnabled(FALSE);
2581         phi_H12Eingabe->setEnabled(FALSE);
2582         labelsH12->setEnabled(FALSE);
2583         s_H12Eingabe->setEnabled(FALSE);
2584
2585         labelAbschnitt3->setEnabled(FALSE);
2586         comboAbschnitt3->setEnabled(FALSE);
2587         labelH23->setEnabled(FALSE);
2588         phi_H23Eingabe->setEnabled(FALSE);
2589         labelsH23->setEnabled(FALSE);
2590         s_H23Eingabe->setEnabled(FALSE);
2591
2592         labelAbschnitt4->setEnabled(FALSE);
2593         comboAbschnitt4->setEnabled(FALSE);
2594         labelH34->setEnabled(FALSE);

```

```

2595     phi_H34Eingabe->setEnabled(FALSE);
2596     labelsH34->setEnabled(FALSE);
2597     s_H34Eingabe->setEnabled(FALSE);
2598
2599     labelAbschnitt5->setEnabled(FALSE);
2600     comboAbschnitt5->setEnabled(FALSE);
2601     labelH45->setEnabled(FALSE);
2602     phi_H45Eingabe->setEnabled(FALSE);
2603     labelsH45->setEnabled(FALSE);
2604     s_H45Eingabe->setEnabled(FALSE);
2605
2606     labelAbschnitt6->setEnabled(FALSE);
2607     comboAbschnitt6->setEnabled(FALSE);
2608     labelH56->setEnabled(FALSE);
2609     phi_H56Eingabe->setEnabled(FALSE);
2610     labelsH56->setEnabled(FALSE);
2611     s_H56Eingabe->setEnabled(FALSE);
2612
2613     labelAbschnitt7->setEnabled(FALSE);
2614     comboAbschnitt7->setEnabled(FALSE);
2615     labelH67->setEnabled(FALSE);
2616     phi_H67Eingabe->setEnabled(FALSE);
2617     labelsH67->setEnabled(FALSE);
2618     s_H67Eingabe->setEnabled(FALSE);
2619
2620     labelAbschnitt8->setEnabled(FALSE);
2621     comboAbschnitt8->setEnabled(FALSE);
2622     labelH78->setEnabled(FALSE);
2623     phi_H78Eingabe->setEnabled(FALSE);
2624     labelsH78->setEnabled(FALSE);
2625     s_H78Eingabe->setEnabled(FALSE);
2626
2627     labelAbschnitt9->setEnabled(FALSE);
2628     comboAbschnitt9->setEnabled(FALSE);
2629     labelH89->setEnabled(FALSE);
2630     phi_H89Eingabe->setEnabled(FALSE);
2631     labelsH89->setEnabled(FALSE);
2632     s_H89Eingabe->setEnabled(FALSE);
2633
2634     labelAbschnitt10->setEnabled(FALSE);
2635     comboAbschnitt10->setEnabled(FALSE);
2636     labelH910->setEnabled(FALSE);
2637     phi_H910Eingabe->setEnabled(FALSE);
2638     labelsH910->setEnabled(FALSE);

```

```

2639     s_H910Eingabe->setEnabled(FALSE);
2640
2641     labelAbschnitt11->setEnabled(FALSE);
2642     comboAbschnitt11->setEnabled(FALSE);
2643     labelH1011->setEnabled(FALSE);
2644     phi_H1011Eingabe->setEnabled(FALSE);
2645     labelsH1011->setEnabled(FALSE);
2646     s_H1011Eingabe->setEnabled(FALSE);
2647
2648     labelAbschnitt12->setEnabled(FALSE);
2649     comboAbschnitt12->setEnabled(FALSE);
2650     labelH1112->setEnabled(FALSE);
2651     phi_H1112Eingabe->setEnabled(FALSE);
2652     labelsH1112->setEnabled(FALSE);
2653     s_H1112Eingabe->setEnabled(FALSE);
2654
2655 }
2656
2657 if(dec==2)
2658 {
2659
2660     labelAbschnitt1->setEnabled(TRUE);
2661     comboAbschnitt1->setEnabled(TRUE);
2662     labelH01->setEnabled(TRUE);
2663     phi_H01Eingabe->setEnabled(TRUE);
2664     labelsH01->setEnabled(TRUE);
2665     s_H01Eingabe->setEnabled(TRUE);
2666
2667     labelAbschnitt2->setEnabled(TRUE);
2668     comboAbschnitt2->setEnabled(TRUE);
2669     labelH12->setEnabled(TRUE);
2670     phi_H12Eingabe->setEnabled(TRUE);
2671     labelsH12->setEnabled(TRUE);
2672     s_H12Eingabe->setEnabled(TRUE);
2673
2674     labelAbschnitt3->setEnabled(FALSE);
2675     comboAbschnitt3->setEnabled(FALSE);
2676     labelH23->setEnabled(FALSE);
2677     phi_H23Eingabe->setEnabled(FALSE);
2678     labelsH23->setEnabled(FALSE);
2679     s_H23Eingabe->setEnabled(FALSE);
2680
2681     labelAbschnitt4->setEnabled(FALSE);
2682     comboAbschnitt4->setEnabled(FALSE);

```

```

2683     labelH34->setEnabled(FALSE);
2684     phi_H34Eingabe->setEnabled(FALSE);
2685     labelsH34->setEnabled(FALSE);
2686     s_H34Eingabe->setEnabled(FALSE);
2687
2688     labelAbschnitt5->setEnabled(FALSE);
2689     comboAbschnitt5->setEnabled(FALSE);
2690     labelH45->setEnabled(FALSE);
2691     phi_H45Eingabe->setEnabled(FALSE);
2692     labelsH45->setEnabled(FALSE);
2693     s_H45Eingabe->setEnabled(FALSE);
2694
2695     labelAbschnitt6->setEnabled(FALSE);
2696     comboAbschnitt6->setEnabled(FALSE);
2697     labelH56->setEnabled(FALSE);
2698     phi_H56Eingabe->setEnabled(FALSE);
2699     labelsH56->setEnabled(FALSE);
2700     s_H56Eingabe->setEnabled(FALSE);
2701
2702     labelAbschnitt7->setEnabled(FALSE);
2703     comboAbschnitt7->setEnabled(FALSE);
2704     labelH67->setEnabled(FALSE);
2705     phi_H67Eingabe->setEnabled(FALSE);
2706     labelsH67->setEnabled(FALSE);
2707     s_H67Eingabe->setEnabled(FALSE);
2708
2709     labelAbschnitt8->setEnabled(FALSE);
2710     comboAbschnitt8->setEnabled(FALSE);
2711     labelH78->setEnabled(FALSE);
2712     phi_H78Eingabe->setEnabled(FALSE);
2713     labelsH78->setEnabled(FALSE);
2714     s_H78Eingabe->setEnabled(FALSE);
2715
2716     labelAbschnitt9->setEnabled(FALSE);
2717     comboAbschnitt9->setEnabled(FALSE);
2718     labelH89->setEnabled(FALSE);
2719     phi_H89Eingabe->setEnabled(FALSE);
2720     labelsH89->setEnabled(FALSE);
2721     s_H89Eingabe->setEnabled(FALSE);
2722
2723     labelAbschnitt10->setEnabled(FALSE);
2724     comboAbschnitt10->setEnabled(FALSE);
2725     labelH910->setEnabled(FALSE);
2726     phi_H910Eingabe->setEnabled(FALSE);

```

```

2727     labelsH910->setEnabled(FALSE);
2728     s_H910Eingabe->setEnabled(FALSE);
2729
2730     labelAbschnitt11->setEnabled(FALSE);
2731     comboAbschnitt11->setEnabled(FALSE);
2732     labelH1011->setEnabled(FALSE);
2733     phi_H1011Eingabe->setEnabled(FALSE);
2734     labelsH1011->setEnabled(FALSE);
2735     s_H1011Eingabe->setEnabled(FALSE);
2736
2737     labelAbschnitt12->setEnabled(FALSE);
2738     comboAbschnitt12->setEnabled(FALSE);
2739     labelH1112->setEnabled(FALSE);
2740     phi_H1112Eingabe->setEnabled(FALSE);
2741     labelsH1112->setEnabled(FALSE);
2742     s_H1112Eingabe->setEnabled(FALSE);
2743
2744 }
2745
2746 if(dec==3)
2747 {
2748
2749     labelAbschnitt1->setEnabled(TRUE);
2750     comboAbschnitt1->setEnabled(TRUE);
2751     labelH01->setEnabled(TRUE);
2752     phi_H01Eingabe->setEnabled(TRUE);
2753     labelsH01->setEnabled(TRUE);
2754     s_H01Eingabe->setEnabled(TRUE);
2755
2756     labelAbschnitt2->setEnabled(TRUE);
2757     comboAbschnitt2->setEnabled(TRUE);
2758     labelH12->setEnabled(TRUE);
2759     phi_H12Eingabe->setEnabled(TRUE);
2760     labelsH12->setEnabled(TRUE);
2761     s_H12Eingabe->setEnabled(TRUE);
2762
2763     labelAbschnitt3->setEnabled(TRUE);
2764     comboAbschnitt3->setEnabled(TRUE);
2765     labelH23->setEnabled(TRUE);
2766     phi_H23Eingabe->setEnabled(TRUE);
2767     labelsH23->setEnabled(TRUE);
2768     s_H23Eingabe->setEnabled(TRUE);
2769
2770     labelAbschnitt4->setEnabled(FALSE);

```



```

2771      comboAbschnitt4->setEnabled(FALSE);
2772      labelH34->setEnabled(FALSE);
2773      phi_H34Eingabe->setEnabled(FALSE);
2774      labelsH34->setEnabled(FALSE);
2775      s_H34Eingabe->setEnabled(FALSE);
2776
2777      labelAbschnitt5->setEnabled(FALSE);
2778      comboAbschnitt5->setEnabled(FALSE);
2779      labelH45->setEnabled(FALSE);
2780      phi_H45Eingabe->setEnabled(FALSE);
2781      labelsH45->setEnabled(FALSE);
2782      s_H45Eingabe->setEnabled(FALSE);
2783
2784      labelAbschnitt6->setEnabled(FALSE);
2785      comboAbschnitt6->setEnabled(FALSE);
2786      labelH56->setEnabled(FALSE);
2787      phi_H56Eingabe->setEnabled(FALSE);
2788      labelsH56->setEnabled(FALSE);
2789      s_H56Eingabe->setEnabled(FALSE);
2790
2791      labelAbschnitt7->setEnabled(FALSE);
2792      comboAbschnitt7->setEnabled(FALSE);
2793      labelH67->setEnabled(FALSE);
2794      phi_H67Eingabe->setEnabled(FALSE);
2795      labelsH67->setEnabled(FALSE);
2796      s_H67Eingabe->setEnabled(FALSE);
2797
2798      labelAbschnitt8->setEnabled(FALSE);
2799      comboAbschnitt8->setEnabled(FALSE);
2800      labelH78->setEnabled(FALSE);
2801      phi_H78Eingabe->setEnabled(FALSE);
2802      labelsH78->setEnabled(FALSE);
2803      s_H78Eingabe->setEnabled(FALSE);
2804
2805      labelAbschnitt9->setEnabled(FALSE);
2806      comboAbschnitt9->setEnabled(FALSE);
2807      labelH89->setEnabled(FALSE);
2808      phi_H89Eingabe->setEnabled(FALSE);
2809      labelsH89->setEnabled(FALSE);
2810      s_H89Eingabe->setEnabled(FALSE);
2811
2812      labelAbschnitt10->setEnabled(FALSE);
2813      comboAbschnitt10->setEnabled(FALSE);
2814      labelH910->setEnabled(FALSE);

```

```

2815     phi_H910Eingabe->setEnabled(FALSE);
2816     labelsH910->setEnabled(FALSE);
2817     s_H910Eingabe->setEnabled(FALSE);
2818
2819     labelAbschnitt11->setEnabled(FALSE);
2820     comboAbschnitt11->setEnabled(FALSE);
2821     labelH1011->setEnabled(FALSE);
2822     phi_H1011Eingabe->setEnabled(FALSE);
2823     labelsH1011->setEnabled(FALSE);
2824     s_H1011Eingabe->setEnabled(FALSE);
2825
2826     labelAbschnitt12->setEnabled(FALSE);
2827     comboAbschnitt12->setEnabled(FALSE);
2828     labelH1112->setEnabled(FALSE);
2829     phi_H1112Eingabe->setEnabled(FALSE);
2830     labelsH1112->setEnabled(FALSE);
2831     s_H1112Eingabe->setEnabled(FALSE);
2832
2833 }
2834
2835 if(dec==4)
2836 {
2837
2838     labelAbschnitt1->setEnabled(TRUE);
2839     comboAbschnitt1->setEnabled(TRUE);
2840     labelH01->setEnabled(TRUE);
2841     phi_H01Eingabe->setEnabled(TRUE);
2842     labelsH01->setEnabled(TRUE);
2843     s_H01Eingabe->setEnabled(TRUE);
2844
2845     labelAbschnitt2->setEnabled(TRUE);
2846     comboAbschnitt2->setEnabled(TRUE);
2847     labelH12->setEnabled(TRUE);
2848     phi_H12Eingabe->setEnabled(TRUE);
2849     labelsH12->setEnabled(TRUE);
2850     s_H12Eingabe->setEnabled(TRUE);
2851
2852     labelAbschnitt3->setEnabled(TRUE);
2853     comboAbschnitt3->setEnabled(TRUE);
2854     labelH23->setEnabled(TRUE);
2855     phi_H23Eingabe->setEnabled(TRUE);
2856     labelsH23->setEnabled(TRUE);
2857     s_H23Eingabe->setEnabled(TRUE);
2858

```

```

2859     labelAbschnitt4->setEnabled(TRUE);
2860     comboAbschnitt4->setEnabled(TRUE);
2861     labelH34->setEnabled(TRUE);
2862     phi_H34Eingabe->setEnabled(TRUE);
2863     labelsH34->setEnabled(TRUE);
2864     s_H34Eingabe->setEnabled(TRUE);
2865
2866     labelAbschnitt5->setEnabled(FALSE);
2867     comboAbschnitt5->setEnabled(FALSE);
2868     labelH45->setEnabled(FALSE);
2869     phi_H45Eingabe->setEnabled(FALSE);
2870     labelsH45->setEnabled(FALSE);
2871     s_H45Eingabe->setEnabled(FALSE);
2872
2873     labelAbschnitt6->setEnabled(FALSE);
2874     comboAbschnitt6->setEnabled(FALSE);
2875     labelH56->setEnabled(FALSE);
2876     phi_H56Eingabe->setEnabled(FALSE);
2877     labelsH56->setEnabled(FALSE);
2878     s_H56Eingabe->setEnabled(FALSE);
2879
2880     labelAbschnitt7->setEnabled(FALSE);
2881     comboAbschnitt7->setEnabled(FALSE);
2882     labelH67->setEnabled(FALSE);
2883     phi_H67Eingabe->setEnabled(FALSE);
2884     labelsH67->setEnabled(FALSE);
2885     s_H67Eingabe->setEnabled(FALSE);
2886
2887     labelAbschnitt8->setEnabled(FALSE);
2888     comboAbschnitt8->setEnabled(FALSE);
2889     labelH78->setEnabled(FALSE);
2890     phi_H78Eingabe->setEnabled(FALSE);
2891     labelsH78->setEnabled(FALSE);
2892     s_H78Eingabe->setEnabled(FALSE);
2893
2894     labelAbschnitt9->setEnabled(FALSE);
2895     comboAbschnitt9->setEnabled(FALSE);
2896     labelH89->setEnabled(FALSE);
2897     phi_H89Eingabe->setEnabled(FALSE);
2898     labelsH89->setEnabled(FALSE);
2899     s_H89Eingabe->setEnabled(FALSE);
2900
2901     labelAbschnitt10->setEnabled(FALSE);
2902     comboAbschnitt10->setEnabled(FALSE);

```

```

2903     labelH910->setEnabled(FALSE);
2904     phi_H910Eingabe->setEnabled(FALSE);
2905     labelsH910->setEnabled(FALSE);
2906     s_H910Eingabe->setEnabled(FALSE);
2907
2908     labelAbschnitt11->setEnabled(FALSE);
2909     comboAbschnitt11->setEnabled(FALSE);
2910     labelH1011->setEnabled(FALSE);
2911     phi_H1011Eingabe->setEnabled(FALSE);
2912     labelsH1011->setEnabled(FALSE);
2913     s_H1011Eingabe->setEnabled(FALSE);
2914
2915     labelAbschnitt12->setEnabled(FALSE);
2916     comboAbschnitt12->setEnabled(FALSE);
2917     labelH1112->setEnabled(FALSE);
2918     phi_H1112Eingabe->setEnabled(FALSE);
2919     labelsH1112->setEnabled(FALSE);
2920     s_H1112Eingabe->setEnabled(FALSE);
2921
2922 }
2923
2924 if(dec==5)
2925 {
2926
2927     labelAbschnitt1->setEnabled(TRUE);
2928     comboAbschnitt1->setEnabled(TRUE);
2929     labelH01->setEnabled(TRUE);
2930     phi_H01Eingabe->setEnabled(TRUE);
2931     labelsH01->setEnabled(TRUE);
2932     s_H01Eingabe->setEnabled(TRUE);
2933
2934     labelAbschnitt2->setEnabled(TRUE);
2935     comboAbschnitt2->setEnabled(TRUE);
2936     labelH12->setEnabled(TRUE);
2937     phi_H12Eingabe->setEnabled(TRUE);
2938     labelsH12->setEnabled(TRUE);
2939     s_H12Eingabe->setEnabled(TRUE);
2940
2941     labelAbschnitt3->setEnabled(TRUE);
2942     comboAbschnitt3->setEnabled(TRUE);
2943     labelH23->setEnabled(TRUE);
2944     phi_H23Eingabe->setEnabled(TRUE);
2945     labelsH23->setEnabled(TRUE);
2946     s_H23Eingabe->setEnabled(TRUE);

```

```

2947
2948     labelAbschnitt4->setEnabled(TRUE);
2949     comboAbschnitt4->setEnabled(TRUE);
2950     labelH34->setEnabled(TRUE);
2951     phi_H34Eingabe->setEnabled(TRUE);
2952     labelsH34->setEnabled(TRUE);
2953     s_H34Eingabe->setEnabled(TRUE);
2954
2955     labelAbschnitt5->setEnabled(TRUE);
2956     comboAbschnitt5->setEnabled(TRUE);
2957     labelH45->setEnabled(TRUE);
2958     phi_H45Eingabe->setEnabled(TRUE);
2959     labelsH45->setEnabled(TRUE);
2960     s_H45Eingabe->setEnabled(TRUE);
2961
2962     labelAbschnitt6->setEnabled(FALSE);
2963     comboAbschnitt6->setEnabled(FALSE);
2964     labelH56->setEnabled(FALSE);
2965     phi_H56Eingabe->setEnabled(FALSE);
2966     labelsH56->setEnabled(FALSE);
2967     s_H56Eingabe->setEnabled(FALSE);
2968
2969     labelAbschnitt7->setEnabled(FALSE);
2970     comboAbschnitt7->setEnabled(FALSE);
2971     labelH67->setEnabled(FALSE);
2972     phi_H67Eingabe->setEnabled(FALSE);
2973     labelsH67->setEnabled(FALSE);
2974     s_H67Eingabe->setEnabled(FALSE);
2975
2976     labelAbschnitt8->setEnabled(FALSE);
2977     comboAbschnitt8->setEnabled(FALSE);
2978     labelH78->setEnabled(FALSE);
2979     phi_H78Eingabe->setEnabled(FALSE);
2980     labelsH78->setEnabled(FALSE);
2981     s_H78Eingabe->setEnabled(FALSE);
2982
2983     labelAbschnitt9->setEnabled(FALSE);
2984     comboAbschnitt9->setEnabled(FALSE);
2985     labelH89->setEnabled(FALSE);
2986     phi_H89Eingabe->setEnabled(FALSE);
2987     labelsH89->setEnabled(FALSE);
2988     s_H89Eingabe->setEnabled(FALSE);
2989
2990     labelAbschnitt10->setEnabled(FALSE);

```

```

2991     comboAbschnitt10->setEnabled(FALSE);
2992     labelH910->setEnabled(FALSE);
2993     phi_H910Eingabe->setEnabled(FALSE);
2994     labelsH910->setEnabled(FALSE);
2995     s_H910Eingabe->setEnabled(FALSE);
2996
2997     labelAbschnitt11->setEnabled(FALSE);
2998     comboAbschnitt11->setEnabled(FALSE);
2999     labelH1011->setEnabled(FALSE);
3000     phi_H1011Eingabe->setEnabled(FALSE);
3001     labelsH1011->setEnabled(FALSE);
3002     s_H1011Eingabe->setEnabled(FALSE);
3003
3004     labelAbschnitt12->setEnabled(FALSE);
3005     comboAbschnitt12->setEnabled(FALSE);
3006     labelH1112->setEnabled(FALSE);
3007     phi_H1112Eingabe->setEnabled(FALSE);
3008     labelsH1112->setEnabled(FALSE);
3009     s_H1112Eingabe->setEnabled(FALSE);
3010
3011 }
3012
3013 if(dec==6)
3014 {
3015
3016     labelAbschnitt1->setEnabled(TRUE);
3017     comboAbschnitt1->setEnabled(TRUE);
3018     labelH01->setEnabled(TRUE);
3019     phi_H01Eingabe->setEnabled(TRUE);
3020     labelsH01->setEnabled(TRUE);
3021     s_H01Eingabe->setEnabled(TRUE);
3022
3023     labelAbschnitt2->setEnabled(TRUE);
3024     comboAbschnitt2->setEnabled(TRUE);
3025     labelH12->setEnabled(TRUE);
3026     phi_H12Eingabe->setEnabled(TRUE);
3027     labelsH12->setEnabled(TRUE);
3028     s_H12Eingabe->setEnabled(TRUE);
3029
3030     labelAbschnitt3->setEnabled(TRUE);
3031     comboAbschnitt3->setEnabled(TRUE);
3032     labelH23->setEnabled(TRUE);
3033     phi_H23Eingabe->setEnabled(TRUE);
3034     labelsH23->setEnabled(TRUE);

```

```

3035     s_H23Eingabe->setEnabled(TRUE);
3036
3037     labelAbschnitt4->setEnabled(TRUE);
3038     comboAbschnitt4->setEnabled(TRUE);
3039     labelH34->setEnabled(TRUE);
3040     phi_H34Eingabe->setEnabled(TRUE);
3041     labelsH34->setEnabled(TRUE);
3042     s_H34Eingabe->setEnabled(TRUE);
3043
3044     labelAbschnitt5->setEnabled(TRUE);
3045     comboAbschnitt5->setEnabled(TRUE);
3046     labelH45->setEnabled(TRUE);
3047     phi_H45Eingabe->setEnabled(TRUE);
3048     labelsH45->setEnabled(TRUE);
3049     s_H45Eingabe->setEnabled(TRUE);
3050
3051     labelAbschnitt6->setEnabled(TRUE);
3052     comboAbschnitt6->setEnabled(TRUE);
3053     labelH56->setEnabled(TRUE);
3054     phi_H56Eingabe->setEnabled(TRUE);
3055     labelsH56->setEnabled(TRUE);
3056     s_H56Eingabe->setEnabled(TRUE);
3057
3058     labelAbschnitt7->setEnabled(FALSE);
3059     comboAbschnitt7->setEnabled(FALSE);
3060     labelH67->setEnabled(FALSE);
3061     phi_H67Eingabe->setEnabled(FALSE);
3062     labelsH67->setEnabled(FALSE);
3063     s_H67Eingabe->setEnabled(FALSE);
3064
3065     labelAbschnitt8->setEnabled(FALSE);
3066     comboAbschnitt8->setEnabled(FALSE);
3067     labelH78->setEnabled(FALSE);
3068     phi_H78Eingabe->setEnabled(FALSE);
3069     labelsH78->setEnabled(FALSE);
3070     s_H78Eingabe->setEnabled(FALSE);
3071
3072     labelAbschnitt9->setEnabled(FALSE);
3073     comboAbschnitt9->setEnabled(FALSE);
3074     labelH89->setEnabled(FALSE);
3075     phi_H89Eingabe->setEnabled(FALSE);
3076     labelsH89->setEnabled(FALSE);
3077     s_H89Eingabe->setEnabled(FALSE);
3078

```

```

3079     labelAbschnitt10->setEnabled(FALSE);
3080     comboAbschnitt10->setEnabled(FALSE);
3081     labelH910->setEnabled(FALSE);
3082     phi_H910Eingabe->setEnabled(FALSE);
3083     labelsH910->setEnabled(FALSE);
3084     s_H910Eingabe->setEnabled(FALSE);
3085
3086     labelAbschnitt11->setEnabled(FALSE);
3087     comboAbschnitt11->setEnabled(FALSE);
3088     labelH1011->setEnabled(FALSE);
3089     phi_H1011Eingabe->setEnabled(FALSE);
3090     labelsH1011->setEnabled(FALSE);
3091     s_H1011Eingabe->setEnabled(FALSE);
3092
3093     labelAbschnitt12->setEnabled(FALSE);
3094     comboAbschnitt12->setEnabled(FALSE);
3095     labelH1112->setEnabled(FALSE);
3096     phi_H1112Eingabe->setEnabled(FALSE);
3097     labelsH1112->setEnabled(FALSE);
3098     s_H1112Eingabe->setEnabled(FALSE);
3099
3100 }
3101
3102 if(dec==7)
3103 {
3104
3105     labelAbschnitt1->setEnabled(TRUE);
3106     comboAbschnitt1->setEnabled(TRUE);
3107     labelH01->setEnabled(TRUE);
3108     phi_H01Eingabe->setEnabled(TRUE);
3109     labelsH01->setEnabled(TRUE);
3110     s_H01Eingabe->setEnabled(TRUE);
3111
3112     labelAbschnitt2->setEnabled(TRUE);
3113     comboAbschnitt2->setEnabled(TRUE);
3114     labelH12->setEnabled(TRUE);
3115     phi_H12Eingabe->setEnabled(TRUE);
3116     labelsH12->setEnabled(TRUE);
3117     s_H12Eingabe->setEnabled(TRUE);
3118
3119     labelAbschnitt3->setEnabled(TRUE);
3120     comboAbschnitt3->setEnabled(TRUE);
3121     labelH23->setEnabled(TRUE);
3122     phi_H23Eingabe->setEnabled(TRUE);

```



```

3123     labelsH23->setEnabled(TRUE);
3124     s_H23Eingabe->setEnabled(TRUE);
3125
3126     labelAbschnitt4->setEnabled(TRUE);
3127     comboAbschnitt4->setEnabled(TRUE);
3128     labelH34->setEnabled(TRUE);
3129     phi_H34Eingabe->setEnabled(TRUE);
3130     labelsH34->setEnabled(TRUE);
3131     s_H34Eingabe->setEnabled(TRUE);
3132
3133     labelAbschnitt5->setEnabled(TRUE);
3134     comboAbschnitt5->setEnabled(TRUE);
3135     labelH45->setEnabled(TRUE);
3136     phi_H45Eingabe->setEnabled(TRUE);
3137     labelsH45->setEnabled(TRUE);
3138     s_H45Eingabe->setEnabled(TRUE);
3139
3140     labelAbschnitt6->setEnabled(TRUE);
3141     comboAbschnitt6->setEnabled(TRUE);
3142     labelH56->setEnabled(TRUE);
3143     phi_H56Eingabe->setEnabled(TRUE);
3144     labelsH56->setEnabled(TRUE);
3145     s_H56Eingabe->setEnabled(TRUE);
3146
3147     labelAbschnitt7->setEnabled(TRUE);
3148     comboAbschnitt7->setEnabled(TRUE);
3149     labelH67->setEnabled(TRUE);
3150     phi_H67Eingabe->setEnabled(TRUE);
3151     labelsH67->setEnabled(TRUE);
3152     s_H67Eingabe->setEnabled(TRUE);
3153
3154     labelAbschnitt8->setEnabled(FALSE);
3155     comboAbschnitt8->setEnabled(FALSE);
3156     labelH78->setEnabled(FALSE);
3157     phi_H78Eingabe->setEnabled(FALSE);
3158     labelsH78->setEnabled(FALSE);
3159     s_H78Eingabe->setEnabled(FALSE);
3160
3161     labelAbschnitt9->setEnabled(FALSE);
3162     comboAbschnitt9->setEnabled(FALSE);
3163     labelH89->setEnabled(FALSE);
3164     phi_H89Eingabe->setEnabled(FALSE);
3165     labelsH89->setEnabled(FALSE);
3166     s_H89Eingabe->setEnabled(FALSE);

```

```

3167
3168     labelAbschnitt10->setEnabled(FALSE);
3169     comboAbschnitt10->setEnabled(FALSE);
3170     labelH910->setEnabled(FALSE);
3171     phi_H910Eingabe->setEnabled(FALSE);
3172     labelsH910->setEnabled(FALSE);
3173     s_H910Eingabe->setEnabled(FALSE);
3174
3175     labelAbschnitt11->setEnabled(FALSE);
3176     comboAbschnitt11->setEnabled(FALSE);
3177     labelH1011->setEnabled(FALSE);
3178     phi_H1011Eingabe->setEnabled(FALSE);
3179     labelsH1011->setEnabled(FALSE);
3180     s_H1011Eingabe->setEnabled(FALSE);
3181
3182     labelAbschnitt12->setEnabled(FALSE);
3183     comboAbschnitt12->setEnabled(FALSE);
3184     labelH1112->setEnabled(FALSE);
3185     phi_H1112Eingabe->setEnabled(FALSE);
3186     labelsH1112->setEnabled(FALSE);
3187     s_H1112Eingabe->setEnabled(FALSE);
3188
3189 }
3190
3191 if(dec==8)
3192 {
3193
3194     labelAbschnitt1->setEnabled(TRUE);
3195     comboAbschnitt1->setEnabled(TRUE);
3196     labelH01->setEnabled(TRUE);
3197     phi_H01Eingabe->setEnabled(TRUE);
3198     labelsH01->setEnabled(TRUE);
3199     s_H01Eingabe->setEnabled(TRUE);
3200
3201     labelAbschnitt2->setEnabled(TRUE);
3202     comboAbschnitt2->setEnabled(TRUE);
3203     labelH12->setEnabled(TRUE);
3204     phi_H12Eingabe->setEnabled(TRUE);
3205     labelsH12->setEnabled(TRUE);
3206     s_H12Eingabe->setEnabled(TRUE);
3207
3208     labelAbschnitt3->setEnabled(TRUE);
3209     comboAbschnitt3->setEnabled(TRUE);
3210     labelH23->setEnabled(TRUE);

```

```

3211     phi_H23Eingabe->setEnabled(TRUE);
3212     labelsH23->setEnabled(TRUE);
3213     s_H23Eingabe->setEnabled(TRUE);
3214
3215     labelAbschnitt4->setEnabled(TRUE);
3216     comboAbschnitt4->setEnabled(TRUE);
3217     labelH34->setEnabled(TRUE);
3218     phi_H34Eingabe->setEnabled(TRUE);
3219     labelsH34->setEnabled(TRUE);
3220     s_H34Eingabe->setEnabled(TRUE);
3221
3222     labelAbschnitt5->setEnabled(TRUE);
3223     comboAbschnitt5->setEnabled(TRUE);
3224     labelH45->setEnabled(TRUE);
3225     phi_H45Eingabe->setEnabled(TRUE);
3226     labelsH45->setEnabled(TRUE);
3227     s_H45Eingabe->setEnabled(TRUE);
3228
3229     labelAbschnitt6->setEnabled(TRUE);
3230     comboAbschnitt6->setEnabled(TRUE);
3231     labelH56->setEnabled(TRUE);
3232     phi_H56Eingabe->setEnabled(TRUE);
3233     labelsH56->setEnabled(TRUE);
3234     s_H56Eingabe->setEnabled(TRUE);
3235
3236     labelAbschnitt7->setEnabled(TRUE);
3237     comboAbschnitt7->setEnabled(TRUE);
3238     labelH67->setEnabled(TRUE);
3239     phi_H67Eingabe->setEnabled(TRUE);
3240     labelsH67->setEnabled(TRUE);
3241     s_H67Eingabe->setEnabled(TRUE);
3242
3243     labelAbschnitt8->setEnabled(TRUE);
3244     comboAbschnitt8->setEnabled(TRUE);
3245     labelH78->setEnabled(TRUE);
3246     phi_H78Eingabe->setEnabled(TRUE);
3247     labelsH78->setEnabled(TRUE);
3248     s_H78Eingabe->setEnabled(TRUE);
3249
3250     labelAbschnitt9->setEnabled(FALSE);
3251     comboAbschnitt9->setEnabled(FALSE);
3252     labelH89->setEnabled(FALSE);
3253     phi_H89Eingabe->setEnabled(FALSE);
3254     labelsH89->setEnabled(FALSE);

```

```

3255     s_H89Eingabe->setEnabled(FALSE);
3256
3257     labelAbschnitt10->setEnabled(FALSE);
3258     comboAbschnitt10->setEnabled(FALSE);
3259     labelH910->setEnabled(FALSE);
3260     phi_H910Eingabe->setEnabled(FALSE);
3261     labelsH910->setEnabled(FALSE);
3262     s_H910Eingabe->setEnabled(FALSE);
3263
3264     labelAbschnitt11->setEnabled(FALSE);
3265     comboAbschnitt11->setEnabled(FALSE);
3266     labelH1011->setEnabled(FALSE);
3267     phi_H1011Eingabe->setEnabled(FALSE);
3268     labelsH1011->setEnabled(FALSE);
3269     s_H1011Eingabe->setEnabled(FALSE);
3270
3271     labelAbschnitt12->setEnabled(FALSE);
3272     comboAbschnitt12->setEnabled(FALSE);
3273     labelH1112->setEnabled(FALSE);
3274     phi_H1112Eingabe->setEnabled(FALSE);
3275     labelsH1112->setEnabled(FALSE);
3276     s_H1112Eingabe->setEnabled(FALSE);
3277
3278 }
3279
3280 if(dec==9)
3281 {
3282
3283     labelAbschnitt1->setEnabled(TRUE);
3284     comboAbschnitt1->setEnabled(TRUE);
3285     labelH01->setEnabled(TRUE);
3286     phi_H01Eingabe->setEnabled(TRUE);
3287     labelsH01->setEnabled(TRUE);
3288     s_H01Eingabe->setEnabled(TRUE);
3289
3290     labelAbschnitt2->setEnabled(TRUE);
3291     comboAbschnitt2->setEnabled(TRUE);
3292     labelH12->setEnabled(TRUE);
3293     phi_H12Eingabe->setEnabled(TRUE);
3294     labelsH12->setEnabled(TRUE);
3295     s_H12Eingabe->setEnabled(TRUE);
3296
3297     labelAbschnitt3->setEnabled(TRUE);
3298     comboAbschnitt3->setEnabled(TRUE);

```

```

3299     labelH23->setEnabled(TRUE);
3300     phi_H23Eingabe->setEnabled(TRUE);
3301     labelsH23->setEnabled(TRUE);
3302     s_H23Eingabe->setEnabled(TRUE);
3303
3304     labelAbschnitt4->setEnabled(TRUE);
3305     comboAbschnitt4->setEnabled(TRUE);
3306     labelH34->setEnabled(TRUE);
3307     phi_H34Eingabe->setEnabled(TRUE);
3308     labelsH34->setEnabled(TRUE);
3309     s_H34Eingabe->setEnabled(TRUE);
3310
3311     labelAbschnitt5->setEnabled(TRUE);
3312     comboAbschnitt5->setEnabled(TRUE);
3313     labelH45->setEnabled(TRUE);
3314     phi_H45Eingabe->setEnabled(TRUE);
3315     labelsH45->setEnabled(TRUE);
3316     s_H45Eingabe->setEnabled(TRUE);
3317
3318     labelAbschnitt6->setEnabled(TRUE);
3319     comboAbschnitt6->setEnabled(TRUE);
3320     labelH56->setEnabled(TRUE);
3321     phi_H56Eingabe->setEnabled(TRUE);
3322     labelsH56->setEnabled(TRUE);
3323     s_H56Eingabe->setEnabled(TRUE);
3324
3325     labelAbschnitt7->setEnabled(TRUE);
3326     comboAbschnitt7->setEnabled(TRUE);
3327     labelH67->setEnabled(TRUE);
3328     phi_H67Eingabe->setEnabled(TRUE);
3329     labelsH67->setEnabled(TRUE);
3330     s_H67Eingabe->setEnabled(TRUE);
3331
3332     labelAbschnitt8->setEnabled(TRUE);
3333     comboAbschnitt8->setEnabled(TRUE);
3334     labelH78->setEnabled(TRUE);
3335     phi_H78Eingabe->setEnabled(TRUE);
3336     labelsH78->setEnabled(TRUE);
3337     s_H78Eingabe->setEnabled(TRUE);
3338
3339     labelAbschnitt9->setEnabled(TRUE);
3340     comboAbschnitt9->setEnabled(TRUE);
3341     labelH89->setEnabled(TRUE);
3342     phi_H89Eingabe->setEnabled(TRUE);

```

```

3343     labelsH89->setEnabled(TRUE);
3344     s_H89Eingabe->setEnabled(TRUE);
3345
3346     labelAbschnitt10->setEnabled(FALSE);
3347     comboAbschnitt10->setEnabled(FALSE);
3348     labelH910->setEnabled(FALSE);
3349     phi_H910Eingabe->setEnabled(FALSE);
3350     labelsH910->setEnabled(FALSE);
3351     s_H910Eingabe->setEnabled(FALSE);
3352
3353     labelAbschnitt11->setEnabled(FALSE);
3354     comboAbschnitt11->setEnabled(FALSE);
3355     labelH1011->setEnabled(FALSE);
3356     phi_H1011Eingabe->setEnabled(FALSE);
3357     labelsH1011->setEnabled(FALSE);
3358     s_H1011Eingabe->setEnabled(FALSE);
3359
3360     labelAbschnitt12->setEnabled(FALSE);
3361     comboAbschnitt12->setEnabled(FALSE);
3362     labelH1112->setEnabled(FALSE);
3363     phi_H1112Eingabe->setEnabled(FALSE);
3364     labelsH1112->setEnabled(FALSE);
3365     s_H1112Eingabe->setEnabled(FALSE);
3366
3367 }
3368
3369 if(dec==10)
3370 {
3371
3372     labelAbschnitt1->setEnabled(TRUE);
3373     comboAbschnitt1->setEnabled(TRUE);
3374     labelH01->setEnabled(TRUE);
3375     phi_H01Eingabe->setEnabled(TRUE);
3376     labelsH01->setEnabled(TRUE);
3377     s_H01Eingabe->setEnabled(TRUE);
3378
3379     labelAbschnitt2->setEnabled(TRUE);
3380     comboAbschnitt2->setEnabled(TRUE);
3381     labelH12->setEnabled(TRUE);
3382     phi_H12Eingabe->setEnabled(TRUE);
3383     labelsH12->setEnabled(TRUE);
3384     s_H12Eingabe->setEnabled(TRUE);
3385
3386     labelAbschnitt3->setEnabled(TRUE);

```

```

3387     comboAbschnitt3->setEnabled(TRUE);
3388     labelH23->setEnabled(TRUE);
3389     phi_H23Eingabe->setEnabled(TRUE);
3390     labelsH23->setEnabled(TRUE);
3391     s_H23Eingabe->setEnabled(TRUE);
3392
3393     labelAbschnitt4->setEnabled(TRUE);
3394     comboAbschnitt4->setEnabled(TRUE);
3395     labelH34->setEnabled(TRUE);
3396     phi_H34Eingabe->setEnabled(TRUE);
3397     labelsH34->setEnabled(TRUE);
3398     s_H34Eingabe->setEnabled(TRUE);
3399
3400     labelAbschnitt5->setEnabled(TRUE);
3401     comboAbschnitt5->setEnabled(TRUE);
3402     labelH45->setEnabled(TRUE);
3403     phi_H45Eingabe->setEnabled(TRUE);
3404     labelsH45->setEnabled(TRUE);
3405     s_H45Eingabe->setEnabled(TRUE);
3406
3407     labelAbschnitt6->setEnabled(TRUE);
3408     comboAbschnitt6->setEnabled(TRUE);
3409     labelH56->setEnabled(TRUE);
3410     phi_H56Eingabe->setEnabled(TRUE);
3411     labelsH56->setEnabled(TRUE);
3412     s_H56Eingabe->setEnabled(TRUE);
3413
3414     labelAbschnitt7->setEnabled(TRUE);
3415     comboAbschnitt7->setEnabled(TRUE);
3416     labelH67->setEnabled(TRUE);
3417     phi_H67Eingabe->setEnabled(TRUE);
3418     labelsH67->setEnabled(TRUE);
3419     s_H67Eingabe->setEnabled(TRUE);
3420
3421     labelAbschnitt8->setEnabled(TRUE);
3422     comboAbschnitt8->setEnabled(TRUE);
3423     labelH78->setEnabled(TRUE);
3424     phi_H78Eingabe->setEnabled(TRUE);
3425     labelsH78->setEnabled(TRUE);
3426     s_H78Eingabe->setEnabled(TRUE);
3427
3428     labelAbschnitt9->setEnabled(TRUE);
3429     comboAbschnitt9->setEnabled(TRUE);
3430     labelH89->setEnabled(TRUE);

```

```

3431     phi_H89Eingabe->setEnabled(TRUE);
3432     labelsH89->setEnabled(TRUE);
3433     s_H89Eingabe->setEnabled(TRUE);
3434
3435     labelAbschnitt10->setEnabled(TRUE);
3436     comboAbschnitt10->setEnabled(TRUE);
3437     labelH910->setEnabled(TRUE);
3438     phi_H910Eingabe->setEnabled(TRUE);
3439     labelsH910->setEnabled(TRUE);
3440     s_H910Eingabe->setEnabled(TRUE);
3441
3442     labelAbschnitt11->setEnabled(FALSE);
3443     comboAbschnitt11->setEnabled(FALSE);
3444     labelH1011->setEnabled(FALSE);
3445     phi_H1011Eingabe->setEnabled(FALSE);
3446     labelsH1011->setEnabled(FALSE);
3447     s_H1011Eingabe->setEnabled(FALSE);
3448
3449     labelAbschnitt12->setEnabled(FALSE);
3450     comboAbschnitt12->setEnabled(FALSE);
3451     labelH1112->setEnabled(FALSE);
3452     phi_H1112Eingabe->setEnabled(FALSE);
3453     labelsH1112->setEnabled(FALSE);
3454     s_H1112Eingabe->setEnabled(FALSE);
3455
3456 }
3457
3458 if(dec==11)
3459 {
3460
3461     labelAbschnitt1->setEnabled(TRUE);
3462     comboAbschnitt1->setEnabled(TRUE);
3463     labelH01->setEnabled(TRUE);
3464     phi_H01Eingabe->setEnabled(TRUE);
3465     labelsH01->setEnabled(TRUE);
3466     s_H01Eingabe->setEnabled(TRUE);
3467
3468     labelAbschnitt2->setEnabled(TRUE);
3469     comboAbschnitt2->setEnabled(TRUE);
3470     labelH12->setEnabled(TRUE);
3471     phi_H12Eingabe->setEnabled(TRUE);
3472     labelsH12->setEnabled(TRUE);
3473     s_H12Eingabe->setEnabled(TRUE);
3474

```



```

3475     labelAbschnitt3->setEnabled(TRUE);
3476     comboAbschnitt3->setEnabled(TRUE);
3477     labelH23->setEnabled(TRUE);
3478     phi_H23Eingabe->setEnabled(TRUE);
3479     labelsH23->setEnabled(TRUE);
3480     s_H23Eingabe->setEnabled(TRUE);
3481
3482     labelAbschnitt4->setEnabled(TRUE);
3483     comboAbschnitt4->setEnabled(TRUE);
3484     labelH34->setEnabled(TRUE);
3485     phi_H34Eingabe->setEnabled(TRUE);
3486     labelsH34->setEnabled(TRUE);
3487     s_H34Eingabe->setEnabled(TRUE);
3488
3489     labelAbschnitt5->setEnabled(TRUE);
3490     comboAbschnitt5->setEnabled(TRUE);
3491     labelH45->setEnabled(TRUE);
3492     phi_H45Eingabe->setEnabled(TRUE);
3493     labelsH45->setEnabled(TRUE);
3494     s_H45Eingabe->setEnabled(TRUE);
3495
3496     labelAbschnitt6->setEnabled(TRUE);
3497     comboAbschnitt6->setEnabled(TRUE);
3498     labelH56->setEnabled(TRUE);
3499     phi_H56Eingabe->setEnabled(TRUE);
3500     labelsH56->setEnabled(TRUE);
3501     s_H56Eingabe->setEnabled(TRUE);
3502
3503     labelAbschnitt7->setEnabled(TRUE);
3504     comboAbschnitt7->setEnabled(TRUE);
3505     labelH67->setEnabled(TRUE);
3506     phi_H67Eingabe->setEnabled(TRUE);
3507     labelsH67->setEnabled(TRUE);
3508     s_H67Eingabe->setEnabled(TRUE);
3509
3510     labelAbschnitt8->setEnabled(TRUE);
3511     comboAbschnitt8->setEnabled(TRUE);
3512     labelH78->setEnabled(TRUE);
3513     phi_H78Eingabe->setEnabled(TRUE);
3514     labelsH78->setEnabled(TRUE);
3515     s_H78Eingabe->setEnabled(TRUE);
3516
3517     labelAbschnitt9->setEnabled(TRUE);
3518     comboAbschnitt9->setEnabled(TRUE);

```

```

3519     labelH89->setEnabled(TRUE);
3520     phi_H89Eingabe->setEnabled(TRUE);
3521     labelsH89->setEnabled(TRUE);
3522     s_H89Eingabe->setEnabled(TRUE);
3523
3524     labelAbschnitt10->setEnabled(TRUE);
3525     comboAbschnitt10->setEnabled(TRUE);
3526     labelH910->setEnabled(TRUE);
3527     phi_H910Eingabe->setEnabled(TRUE);
3528     labelsH910->setEnabled(TRUE);
3529     s_H910Eingabe->setEnabled(TRUE);
3530
3531     labelAbschnitt11->setEnabled(TRUE);
3532     comboAbschnitt11->setEnabled(TRUE);
3533     labelH1011->setEnabled(TRUE);
3534     phi_H1011Eingabe->setEnabled(TRUE);
3535     labelsH1011->setEnabled(TRUE);
3536     s_H1011Eingabe->setEnabled(TRUE);
3537
3538     labelAbschnitt12->setEnabled(FALSE);
3539     comboAbschnitt12->setEnabled(FALSE);
3540     labelH1112->setEnabled(FALSE);
3541     phi_H1112Eingabe->setEnabled(FALSE);
3542     labelsH1112->setEnabled(FALSE);
3543     s_H1112Eingabe->setEnabled(FALSE);
3544
3545 }
3546
3547 if(dec==12)
3548 {
3549
3550     labelAbschnitt1->setEnabled(TRUE);
3551     comboAbschnitt1->setEnabled(TRUE);
3552     labelH01->setEnabled(TRUE);
3553     phi_H01Eingabe->setEnabled(TRUE);
3554     labelsH01->setEnabled(TRUE);
3555     s_H01Eingabe->setEnabled(TRUE);
3556
3557     labelAbschnitt2->setEnabled(TRUE);
3558     comboAbschnitt2->setEnabled(TRUE);
3559     labelH12->setEnabled(TRUE);
3560     phi_H12Eingabe->setEnabled(TRUE);
3561     labelsH12->setEnabled(TRUE);
3562     s_H12Eingabe->setEnabled(TRUE);

```

```
3563
3564     labelAbschnitt3->setEnabled(TRUE);
3565     comboAbschnitt3->setEnabled(TRUE);
3566     labelH23->setEnabled(TRUE);
3567     phi_H23Eingabe->setEnabled(TRUE);
3568     labelsH23->setEnabled(TRUE);
3569     s_H23Eingabe->setEnabled(TRUE);
3570
3571     labelAbschnitt4->setEnabled(TRUE);
3572     comboAbschnitt4->setEnabled(TRUE);
3573     labelH34->setEnabled(TRUE);
3574     phi_H34Eingabe->setEnabled(TRUE);
3575     labelsH34->setEnabled(TRUE);
3576     s_H34Eingabe->setEnabled(TRUE);
3577
3578     labelAbschnitt5->setEnabled(TRUE);
3579     comboAbschnitt5->setEnabled(TRUE);
3580     labelH45->setEnabled(TRUE);
3581     phi_H45Eingabe->setEnabled(TRUE);
3582     labelsH45->setEnabled(TRUE);
3583     s_H45Eingabe->setEnabled(TRUE);
3584
3585     labelAbschnitt6->setEnabled(TRUE);
3586     comboAbschnitt6->setEnabled(TRUE);
3587     labelH56->setEnabled(TRUE);
3588     phi_H56Eingabe->setEnabled(TRUE);
3589     labelsH56->setEnabled(TRUE);
3590     s_H56Eingabe->setEnabled(TRUE);
3591
3592     labelAbschnitt7->setEnabled(TRUE);
3593     comboAbschnitt7->setEnabled(TRUE);
3594     labelH67->setEnabled(TRUE);
3595     phi_H67Eingabe->setEnabled(TRUE);
3596     labelsH67->setEnabled(TRUE);
3597     s_H67Eingabe->setEnabled(TRUE);
3598
3599     labelAbschnitt8->setEnabled(TRUE);
3600     comboAbschnitt8->setEnabled(TRUE);
3601     labelH78->setEnabled(TRUE);
3602     phi_H78Eingabe->setEnabled(TRUE);
3603     labelsH78->setEnabled(TRUE);
3604     s_H78Eingabe->setEnabled(TRUE);
3605
3606     labelAbschnitt9->setEnabled(TRUE);
```

```

3607     comboAbschnitt9->setEnabled(TRUE);
3608     labelH89->setEnabled(TRUE);
3609     phi_H89Eingabe->setEnabled(TRUE);
3610     labelsH89->setEnabled(TRUE);
3611     s_H89Eingabe->setEnabled(TRUE);
3612
3613     labelAbschnitt10->setEnabled(TRUE);
3614     comboAbschnitt10->setEnabled(TRUE);
3615     labelH910->setEnabled(TRUE);
3616     phi_H910Eingabe->setEnabled(TRUE);
3617     labelsH910->setEnabled(TRUE);
3618     s_H910Eingabe->setEnabled(TRUE);
3619
3620     labelAbschnitt11->setEnabled(TRUE);
3621     comboAbschnitt11->setEnabled(TRUE);
3622     labelH1011->setEnabled(TRUE);
3623     phi_H1011Eingabe->setEnabled(TRUE);
3624     labelsH1011->setEnabled(TRUE);
3625     s_H1011Eingabe->setEnabled(TRUE);
3626
3627     labelAbschnitt12->setEnabled(TRUE);
3628     comboAbschnitt12->setEnabled(TRUE);
3629     labelH1112->setEnabled(TRUE);
3630     phi_H1112Eingabe->setEnabled(TRUE);
3631     labelsH1112->setEnabled(TRUE);
3632     s_H1112Eingabe->setEnabled(TRUE);
3633
3634 }
3635
3636 }
3637
3638 void Opticurv::setupTab2()
3639 {
3640     QWidget* secondpage = new QWidget( this );
3641     secondpage->setGeometry(10,10,260,310);
3642     tabdialog->addTab( secondpage, "Geometriedaten");
3643     QColor farbe( 0, 255, 255 );
3644     ergebnis = 1;
3645
3646     QLabel* labelausfuehrungKurve = new QLabel(
3647         "Ausführung des Kurvengetriebes",secondpage );
3648     labelausfuehrungKurve->setGeometry( 20, 10, 190, 20);
3649     ausfuehrungKurve = new QComboBox( false, secondpage );
3650     ausfuehrungKurve->setGeometry( 220, 10, 160, 20 );

```

```

3651     for ( i=0; i<auswahlKurveNo; i++)
3652         ausfuehrungKurve->insertItem( auswahlKurvengetriebe[i] );
3653 if(indexKurve==0 || indexKurve==1)
3654 {
3655     ausfuehrungKurve->setCurrentItem( indexKurve );
3656 }
3657 else
3658 {
3659     ausfuehrungKurve->setCurrentItem( 0 );
3660 }
3661
3662 x_A0Eingabe = new QLineEdit( secondpage );
3663 x_A0Eingabe->setGeometry( 120,50,95, 30);
3664 QString x_A0ausgabe=x_A0Eingabe->text();
3665 x_A0ausgabe.setNum( dx_A0 );
3666 x_A0Eingabe->setText( x_A0ausgabe );
3667 x_A0Eingabe->setPalette( QPalette( farbe, farbe ) );
3668 QLabel* labelx_A0 = new QLabel( "x_A0 [mm]",secondpage );
3669 labelx_A0->setGeometry( 20, 50, 100, 30);
3670 QToolTip::add( labelx_A0, "x-Koordinate des Kurvenscheibenlagerpunktes A_0" );
3671
3672 y_A0Eingabe = new QLineEdit( secondpage );
3673 y_A0Eingabe->setGeometry( 120, 80, 95,30);
3674 QString y_A0ausgabe=y_A0Eingabe->text();
3675 y_A0ausgabe.setNum( dy_A0 );
3676 y_A0Eingabe->setText( y_A0ausgabe );
3677 y_A0Eingabe->setPalette( QPalette( farbe, farbe ) );
3678 QLabel* labely_A0 = new QLabel( "y_A0 [mm]",secondpage );
3679 labely_A0->setGeometry( 20, 80, 100, 30);
3680 QToolTip::add( labely_A0, "y-Koordinate des Kurvenscheibenlagerpunktes A_0" );
3681
3682 x_s0Eingabe = new QLineEdit( secondpage );
3683 x_s0Eingabe->setGeometry( 120,110,95, 30);
3684 x_s0Eingabe->setEnabled(TRUE);
3685 QString x_s0ausgabe=x_s0Eingabe->text();
3686 x_s0ausgabe.setNum( dx_s0 );
3687 x_s0Eingabe->setText( x_s0ausgabe );
3688 x_s0Eingabe->setPalette( QPalette( farbe, farbe ) );
3689 QLabel* labelx_s0 = new QLabel( "x_s0 [mm]",secondpage );
3690 labelx_s0->setGeometry( 20, 110, 100, 30);
3691 labelx_s0->setEnabled(TRUE);
3692 QToolTip::add( labelx_s0, "x-Koordinate Nullhub" );
3693
3694 y_s0Eingabe = new QLineEdit( secondpage );

```

```

3695     y_s0Eingabe->setGeometry( 120, 140, 95,30);
3696     QString y_s0ausgabe=y_s0Eingabe->text();
3697     y_s0ausgabe.setNum( dy_s0 );
3698     y_s0Eingabe->setText( y_s0ausgabe );
3699     y_s0Eingabe->setPalette( QPalette( farbe, farbe ) );
3700     QLabel* labely_s0 = new QLabel( "y_s0          [mm]",secondpage );
3701     labely_s0->setGeometry( 20, 140, 100, 30);
3702     QToolTip::add( labely_s0, "y-Koordinate Nullhub" );
3703
3704     l_3Eingabe = new QLineEdit( secondpage );
3705     l_3Eingabe->setGeometry( 120,170,95, 30);
3706     QString l_3ausgabe=l_3Eingabe->text();
3707     l_3ausgabe.setNum( dl_3 );
3708     l_3Eingabe->setText( l_3ausgabe );
3709     l_3Eingabe->setPalette( QPalette( farbe, farbe ) );
3710     QLabel* labell_3 = new QLabel( "l_3          [mm]",secondpage );
3711     labell_3->setGeometry( 20, 170, 100, 30);
3712     QToolTip::add( labell_3, "Rollenhebellänge (Glieder 3)" );
3713
3714     l_skEingabe = new QLineEdit( secondpage );
3715     l_skEingabe->setGeometry( 120,200,95, 30);
3716     QString l_skausgabe=l_skEingabe->text();
3717     l_skausgabe.setNum( dl_sk );
3718     l_skEingabe->setText( l_skausgabe );
3719     l_skEingabe->setPalette( QPalette( farbe, farbe ) );
3720     QLabel* labell_sk = new QLabel( "l_sk          [mm]",secondpage );
3721     labell_sk->setGeometry( 20, 200, 100, 30);
3722     QToolTip::add( labell_sk, "Schubkurbellänge (Glieder 3)" );
3723
3724     l_4Eingabe = new QLineEdit( secondpage );
3725     l_4Eingabe->setGeometry( 120, 230, 95,30);
3726     QString l_4ausgabe=l_4Eingabe->text();
3727     l_4ausgabe.setNum( dl_4 );
3728     l_4Eingabe->setText( l_4ausgabe );
3729     l_4Eingabe->setPalette( QPalette( farbe, farbe ) );
3730     QLabel* labell_4 = new QLabel( "l_4          [mm]",secondpage );
3731     labell_4->setGeometry( 20, 230, 100, 30);
3732     QToolTip::add( labell_4, "Koppellänge (Glieder 4)" );
3733
3734     r_GEingabe = new QLineEdit( secondpage );
3735     r_GEingabe->setGeometry( 325,50,95, 30);
3736     QString r_Gausgabe=r_GEingabe->text();
3737     r_Gausgabe.setNum( dr_G );
3738     r_GEingabe->setText( r_Gausgabe );

```

```

3739     r_GEingabe->setPalette( QPalette( farbe, farbe ) );
3740     QLabel* labelr_G = new QLabel( "r_G           [mm]",secondpage );
3741     labelr_G->setGeometry( 225, 50, 100, 30);
3742     QToolTip::add( labelr_G, "Grundkreisradius" );
3743
3744     r_REingabe = new QLineEdit( secondpage );
3745     r_REingabe->setGeometry( 325, 80, 95,30);
3746     QString r_Rausgabe=r_REingabe->text();
3747     r_Rausgabe.setNum( dr_R );
3748     r_REingabe->setText( r_Rausgabe );
3749     r_REingabe->setPalette( QPalette( farbe, farbe ) );
3750     QLabel* labelr_R = new QLabel( "r_R           [mm]",secondpage );
3751     labelr_R->setGeometry( 225, 80, 100, 30);
3752     QToolTip::add( labelr_R, "Rollenradius" );
3753
3754     r_SEingabe = new QLineEdit( secondpage );
3755     r_SEingabe->setGeometry( 325,110,95, 30);
3756     QString r_Sausgabe=r_SEingabe->text();
3757     r_Sausgabe.setNum( dr_S );
3758     r_SEingabe->setText( r_Sausgabe );
3759     r_SEingabe->setPalette( QPalette( farbe, farbe ) );
3760     QLabel* labelr_S = new QLabel( "r_S           [mm]",secondpage );
3761     labelr_S->setGeometry( 225, 110, 100, 30);
3762     QToolTip::add( labelr_S, "Kurvenscheibenradius" );
3763
3764     r_WEingabe = new QLineEdit( secondpage );
3765     r_WEingabe->setGeometry( 325,140,95, 30);
3766     QString r_Wausgabe=r_WEingabe->text();
3767     r_Wausgabe.setNum( dr_W );
3768     r_WEingabe->setText( r_Wausgabe );
3769     r_WEingabe->setPalette( QPalette( farbe, farbe ) );
3770     QLabel* labelr_W = new QLabel( "r_W           [mm]",secondpage );
3771     labelr_W->setGeometry( 225, 140, 100, 30);
3772     QToolTip::add( labelr_W, "Kurvenscheibenradius" );
3773
3774     n_1Eingabe = new QLineEdit( secondpage );
3775     n_1Eingabe->setGeometry( 325, 170, 95,30);
3776     QString n_1ausgabe=n_1Eingabe->text();
3777     n_1ausgabe.setNum( dn_1 );
3778     n_1Eingabe->setText( n_1ausgabe );
3779     n_1Eingabe->setPalette( QPalette( farbe, farbe ) );
3780     QLabel* labeln_1 = new QLabel( "n           [Grad]",secondpage );
3781     labeln_1->setGeometry( 225, 170, 100, 30);
3782     QToolTip::add( labeln_1, "Schrittweite (des Drehwinkels)" );

```

```

3783
3784 QLineEdit *new6Eingabe = new QLineEdit( secondpage );
3785     new6Eingabe->setGeometry( 325,200,95, 30);
3786     new6Eingabe->setEnabled(FALSE);
3787
3788 QLineEdit *new7Eingabe = new QLineEdit( secondpage );
3789     new7Eingabe->setGeometry( 325, 230, 95,30);
3790     new7Eingabe->setEnabled(FALSE);
3791
3792 alpha_REingabe = new QLineEdit( secondpage );
3793     alpha_REingabe->setGeometry( 530,50,95, 30);
3794     QString alpha_Rausgabe=alpha_REingabe->text();
3795     alpha_Rausgabe.setNum( dalpha_R );
3796     alpha_REingabe->setText( alpha_Rausgabe );
3797     alpha_REingabe->setPalette( QPalette( farbe, farbe ) );
3798     labelalpha_R = new QLabel( "alpha_R    [Grad]",secondpage );
3799     labelalpha_R->setGeometry( 430, 50, 100, 30);
3800     QToolTip::add( labelalpha_R, "Doppelrollenhebelwinkel" );
3801
3802 l_3starEingabe = new QLineEdit( secondpage );
3803     l_3starEingabe->setGeometry( 530,80,95, 30);
3804     QString l_3starausgabe=l_3starEingabe->text();
3805     l_3starausgabe.setNum( dl_3star );
3806     l_3starEingabe->setText( l_3starausgabe );
3807     l_3starEingabe->setPalette( QPalette( farbe, farbe ) );
3808     labell_3star = new QLabel( "l_3stern    [mm]",secondpage );
3809     labell_3star->setGeometry( 430, 80, 100, 30);
3810     QToolTip::add( labell_3star, "Rollenhebelänge der Gegenkurve" );
3811
3812 r_RstarEingabe = new QLineEdit( secondpage );
3813     r_RstarEingabe->setGeometry( 530,110,95, 30);
3814     QString r_Rstarausgabe=r_RstarEingabe->text();
3815     r_Rstarausgabe.setNum( dr_Rstar );
3816     r_RstarEingabe->setText( r_Rstarausgabe );
3817     r_RstarEingabe->setPalette( QPalette( farbe, farbe ) );
3818     labelr_Rstar = new QLabel( "r_Rstern    [mm]",secondpage );
3819     labelr_Rstar->setGeometry( 430, 110, 100, 30);
3820     QToolTip::add( labelr_Rstar, "Rollenradius der Gegenkurve" );
3821
3822 /* Anzeige bei Doppelkurvenscheiben regeln */
3823
3824 if(indexKurve==0)
3825 {
3826     alpha_REingabe->setEnabled(FALSE);

```



```

3827     labelalpha_R->setEnabled(FALSE);
3828
3829     l_3starEingabe->setEnabled(FALSE);
3830     labell_3star->setEnabled(FALSE);
3831
3832     r_RstarEingabe->setEnabled(FALSE);
3833     labelr_Rstar->setEnabled(FALSE);
3834 }
3835 else if(indexKurve==1)
3836 {
3837     alpha_REingabe->setEnabled(TRUE);
3838     labelalpha_R->setEnabled(TRUE);
3839
3840     l_3starEingabe->setEnabled(TRUE);
3841     labell_3star->setEnabled(TRUE);
3842
3843     r_RstarEingabe->setEnabled(TRUE);
3844     labelr_Rstar->setEnabled(TRUE);
3845 }
3846 else
3847 {
3848     alpha_REingabe->setEnabled(FALSE);
3849     labelalpha_R->setEnabled(FALSE);
3850
3851     l_3starEingabe->setEnabled(FALSE);
3852     labell_3star->setEnabled(FALSE);
3853
3854     r_RstarEingabe->setEnabled(FALSE);
3855     labelr_Rstar->setEnabled(FALSE);
3856 }
3857
3858 QLineEdit *new2Eingabe = new QLineEdit( secondpage );
3859 new2Eingabe->setGeometry( 530,140,95, 30);
3860 new2Eingabe->setEnabled(FALSE);
3861
3862 QLineEdit *new3Eingabe = new QLineEdit( secondpage );
3863 new3Eingabe->setGeometry( 530,200,95, 30);
3864 new3Eingabe->setEnabled(FALSE);
3865
3866 QLineEdit *new4Eingabe = new QLineEdit( secondpage );
3867 new4Eingabe->setGeometry( 530,230,95, 30);
3868 new4Eingabe->setEnabled(FALSE);
3869
3870 QLineEdit *new5Eingabe = new QLineEdit( secondpage );

```

```

3871     new5Eingabe->setGeometry( 530,170,95, 30);
3872     new5Eingabe->setEnabled(FALSE);
3873
3874     QObject::connect( ausfuehrungKurve, SIGNAL( activated( int ) ),
3875                      this, SLOT( zeigeAlpha_R() ) );
3876
3877     QObject::connect(l_3Eingabe,
3878                      SIGNAL( textChanged( const QString & ) ),
3879                      this, SLOT( neu_l_3( const QString & ) ) );
3880
3881     QObject::connect(x_s0Eingabe,
3882                      SIGNAL( textChanged( const QString & ) ),
3883                      this, SLOT( neu_x_s0( const QString & ) ) );
3884
3885     QObject::connect(y_s0Eingabe,
3886                      SIGNAL( textChanged( const QString & ) ),
3887                      this, SLOT( neu_y_s0( const QString & ) ) );
3888
3889     QObject::connect(l_skEingabe,
3890                      SIGNAL( textChanged( const QString & ) ),
3891                      this, SLOT( neu_l_sk( const QString & ) ) );
3892
3893     QObject::connect(l_4Eingabe,
3894                      SIGNAL( textChanged( const QString & ) ),
3895                      this, SLOT( neu_l_4( const QString & ) ) );
3896
3897     QObject::connect(r_REingabe,
3898                      SIGNAL( textChanged( const QString & ) ),
3899                      this, SLOT( neu_r_R( const QString & ) ) );
3900
3901     QObject::connect(x_A0Eingabe,
3902                      SIGNAL( textChanged( const QString & ) ),
3903                      this, SLOT( neu_x_A0( const QString & ) ) );
3904
3905     QObject::connect(y_A0Eingabe,
3906                      SIGNAL( textChanged( const QString & ) ),
3907                      this, SLOT( neu_y_A0( const QString & ) ) );
3908
3909     QObject::connect(r_GEingabe,
3910                      SIGNAL( textChanged( const QString & ) ),
3911                      this, SLOT( neu_r_G( const QString & ) ) );
3912
3913     QObject::connect(n_1Eingabe,
3914                      SIGNAL( textChanged( const QString & ) ),

```

```

3915         this, SLOT( neu_n_1( const QString & ) ) );
3916
3917     QObject::connect(r_SEingabe,
3918         SIGNAL( textChanged( const QString & ) ),
3919         this, SLOT( neu_r_S( const QString & ) ) );
3920
3921     QObject::connect(r_WEingabe,
3922         SIGNAL( textChanged( const QString & ) ),
3923         this, SLOT( neu_r_W( const QString & ) ) );
3924
3925     QObject::connect(alpha_REingabe,
3926         SIGNAL( textChanged( const QString & ) ),
3927         this, SLOT( neu_alpha_R( const QString & ) ) );
3928
3929     QObject::connect(l_3starEingabe,
3930         SIGNAL( textChanged( const QString & ) ),
3931         this, SLOT( neu_l_3star( const QString & ) ) );
3932
3933     QObject::connect(r_RstarEingabe,
3934         SIGNAL( textChanged( const QString & ) ),
3935         this, SLOT( neu_r_Rstar( const QString & ) ) );
3936
3937 }
3938
3939 void Opticurv::zeigeAlpha_R()
3940 {
3941     ergebnis = 1;
3942     QColor farbe( 0, 255, 255 );
3943     int dec = ausfuehrungKurve->currentItem();
3944
3945     if(dec==0)
3946     {
3947         alpha_REingabe->setEnabled(FALSE);
3948         labelalpha_R->setEnabled(FALSE);
3949
3950         l_3starEingabe->setEnabled(FALSE);
3951         labell_3star->setEnabled(FALSE);
3952
3953         r_RstarEingabe->setEnabled(FALSE);
3954         labelr_Rstar->setEnabled(FALSE);
3955
3956         abfrage6->setEnabled(FALSE);
3957
3958         CB_Mittelpunktskurven->setEnabled(FALSE);

```

```

3959
3960     QL_Mittelpunktskurven->setEnabled(FALSE);
3961
3962     QL_x_minMittelpunktskurven->setEnabled(FALSE);
3963     QLE_x_minMittelpunktskurven->setEnabled(FALSE);
3964
3965     QL_x_maxMittelpunktskurven->setEnabled(FALSE);
3966     QLE_x_maxMittelpunktskurven->setEnabled(FALSE);
3967
3968     QL_dxMittelpunktskurven->setEnabled(FALSE);
3969     QLE_dxMittelpunktskurven->setEnabled(FALSE);
3970
3971     QL_y_minMittelpunktskurven->setEnabled(FALSE);
3972     QLE_y_minMittelpunktskurven->setEnabled(FALSE);
3973
3974     QL_y_maxMittelpunktskurven->setEnabled(FALSE);
3975     QLE_y_maxMittelpunktskurven->setEnabled(FALSE);
3976
3977     QL_dyMittelpunktskurven->setEnabled(FALSE);
3978     QLE_dyMittelpunktskurven->setEnabled(FALSE);
3979 }
3980 if(dec==1)
3981 {
3982     alpha_REingabe->setEnabled(TRUE);
3983     labelalpha_R->setEnabled(TRUE);
3984
3985     l_3starEingabe->setEnabled(TRUE);
3986     labell_3star->setEnabled(TRUE);
3987
3988     r_RstarEingabe->setEnabled(TRUE);
3989     labelr_Rstar->setEnabled(TRUE);
3990
3991     abfrage6->setEnabled(TRUE);
3992
3993     CB_Mittelpunktskurven->setEnabled(TRUE);
3994
3995     if( CB_Mittelpunktskurven->isChecked() )
3996     {
3997         QL_Mittelpunktskurven->setEnabled(TRUE);
3998
3999         QL_x_minMittelpunktskurven->setEnabled(TRUE);
4000         QLE_x_minMittelpunktskurven->setEnabled(TRUE);
4001
4002         QL_x_maxMittelpunktskurven->setEnabled(TRUE);

```

```

4003         QLE_x_maxMittelpunktskurven->setEnabled(TRUE);
4004
4005         QL_dxMittelpunktskurven->setEnabled(TRUE);
4006         QLE_dxMittelpunktskurven->setEnabled(TRUE);
4007
4008         QL_y_minMittelpunktskurven->setEnabled(TRUE);
4009         QLE_y_minMittelpunktskurven->setEnabled(TRUE);
4010
4011         QL_y_maxMittelpunktskurven->setEnabled(TRUE);
4012         QLE_y_maxMittelpunktskurven->setEnabled(TRUE);
4013
4014         QL_dyMittelpunktskurven->setEnabled(TRUE);
4015         QLE_dyMittelpunktskurven->setEnabled(TRUE);
4016     }
4017     if( !CB_Mittelpunktskurven->isChecked() )
4018     {
4019         QL_Mittelpunktskurven->setEnabled(FALSE);
4020
4021         QL_x_minMittelpunktskurven->setEnabled(FALSE);
4022         QLE_x_minMittelpunktskurven->setEnabled(FALSE);
4023
4024         QL_x_maxMittelpunktskurven->setEnabled(FALSE);
4025         QLE_x_maxMittelpunktskurven->setEnabled(FALSE);
4026
4027         QL_dxMittelpunktskurven->setEnabled(FALSE);
4028         QLE_dxMittelpunktskurven->setEnabled(FALSE);
4029
4030         QL_y_minMittelpunktskurven->setEnabled(FALSE);
4031         QLE_y_minMittelpunktskurven->setEnabled(FALSE);
4032
4033         QL_y_maxMittelpunktskurven->setEnabled(FALSE);
4034         QLE_y_maxMittelpunktskurven->setEnabled(FALSE);
4035
4036         QL_dyMittelpunktskurven->setEnabled(FALSE);
4037         QLE_dyMittelpunktskurven->setEnabled(FALSE);
4038     }
4039 }
4040 }
4041
4042
4043
4044 void Opticurv::setupTab3()
4045 {
4046     QWidget* thirdpage = new QWidget( this );

```

```

4047     thirdpage->setGeometry(10,10,260,310);
4048     tabdialog->addTab( thirdpage, "Vorzeichendefinitionen");
4049     QColor farbe( 0, 255, 255 );
4050     ergebnis = 1;
4051
4052     buttonbox1 = new QButtonGroup( 1, Horizontal,
4053         "Ist die Drehrichtung der Kurvenscheibe math.", thirdpage);
4054     buttonbox1->setGeometry( 20, 10, 290, 100 );
4055     buttonbox1->setFrameStyle( QFrame::Panel | QFrame::Sunken );
4056     abfrage1 = new QLabel( buttonbox1 );
4057     abfrage1->setText( "positiv oder negativ?" );
4058     abfrage1->setGeometry( 10, 120, 270, 100 );
4059     ja1 = new QRadioButton( "mathematisch positiv", buttonbox1 );
4060     nein1 = new QRadioButton( "mathematisch negativ", buttonbox1 );
4061
4062     if(dri_scheibe == 1)
4063     {
4064         ja1->setChecked( true );
4065     }
4066     else if(dri_scheibe == 2)
4067     {
4068         nein1->setChecked( true );
4069     }
4070     else
4071     {
4072         ja1->setChecked( true );
4073     }
4074
4075     buttonbox2 = new QButtonGroup( 1, Horizontal,
4076         "Sind Hubrichtungen von Koordinate x_S0 und", thirdpage);
4077     buttonbox2->setGeometry( 20, 125, 290, 100 );
4078     buttonbox2->setFrameStyle( QFrame::Panel | QFrame::Sunken );
4079     abfrage2 = new QLabel( buttonbox2 );
4080     abfrage2->setText( "Hub s( $\phi$ ) zu Beginn der Bew. gleich gerichtet?" );
4081     abfrage2->setGeometry( 10, 120, 270, 100 );
4082     ja2 = new QRadioButton( "Ja", buttonbox2 );
4083     nein2 = new QRadioButton( "Nein", buttonbox2 );
4084     if(dhub_gleich == 1)
4085     {
4086         ja2->setChecked( true );
4087     }
4088     else if(dhub_gleich == 2)
4089     {
4090         nein2->setChecked( true );

```

```

4091     }
4092     else
4093     {
4094         ja2->setChecked( true );
4095     }
4096
4097     buttonbox3 = new QButtonGroup( 1, Horizontal,
4098         "In welchem Quadrant befindet sich der", thirdpage);
4099     buttonbox3->setGeometry( 20, 240, 290, 100 );
4100     buttonbox3->setFrameStyle( QFrame::Panel | QFrame::Sunken );
4101     abfrage3 = new QLabel( buttonbox3 );
4102     abfrage3->setText( "Punkt C_0?" );
4103     abfrage3->setGeometry( 10, 120, 270, 100 );
4104     ja3 = new QRadioButton( "1. bzw. 4. Quadrant", buttonbox3 );
4105     nein3 = new QRadioButton( "2. bzw. 3. Quadrant", buttonbox3 );
4106     if(dquadrant == 1)
4107     {
4108         ja3->setChecked( true );
4109     }
4110     else if(dquadrant == 2)
4111     {
4112         nein3->setChecked( true );
4113     }
4114     else
4115     {
4116         ja3->setChecked( true );
4117     }
4118
4119     buttonbox4 = new QButtonGroup( 1, Horizontal,
4120         "Ist die Bewegung des Schwinghebels zu Beginn", thirdpage);
4121     buttonbox4->setGeometry( 325, 10, 290, 100 );
4122     buttonbox4->setFrameStyle( QFrame::Panel | QFrame::Sunken );
4123     abfrage4 = new QLabel( buttonbox4 );
4124     abfrage4->setText( "der Bewegung math. positiv oder negativ?" );
4125     abfrage4->setGeometry( 10, 120, 270, 100 );
4126     ja4 = new QRadioButton( "mathematisch positiv", buttonbox4 );
4127     nein4 = new QRadioButton( "mathematisch negativ", buttonbox4 );
4128     if(dpsi_positiv == 1)
4129     {
4130         ja4->setChecked( true );
4131     }
4132     else if(dpsi_positiv == 2)
4133     {
4134         nein4->setChecked( true );

```

```

4135     }
4136     else
4137     {
4138         ja4->setChecked( true );
4139     }
4140
4141     buttonbox5 = new QButtonGroup( 1, Horizontal, "", thirdpage);
4142     buttonbox5->setGeometry( 325, 130, 290, 95 );
4143     buttonbox5->setFrameStyle( QFrame::Panel | QFrame::Sunken );
4144     abfrage5 = new QLabel( buttonbox5 );
4145     abfrage5->setGeometry( 10, 120, 270, 100 );
4146     buttonbox6 = new QButtonGroup( 1, Horizontal, "", thirdpage);
4147     buttonbox6->setGeometry( 325, 245, 290, 95 );
4148     buttonbox6->setFrameStyle( QFrame::Panel | QFrame::Sunken );
4149     abfrage6 = new QLabel( buttonbox6 );
4150     abfrage6->setGeometry( 10, 120, 270, 100 );
4151 }
4152
4153 void Opticurv::setupTab4()
4154 {
4155     QWidget* fourthpage = new QWidget( this );
4156     fourthpage->setGeometry(10,10,730,430);
4157     tabdialog->addTab( fourthpage, "Skalierung");
4158     QColor farbe( 0, 255, 255 );
4159     ergebnis = 1;
4160
4161     BG_Skalierung = new QButtonGroup( 3, Vertical,
4162         "Verwende für folgende Funktionen manuelle Einstellwerte", fourthpage);
4163     BG_Skalierung->setGeometry( 18, 18, 595, 97 );
4164     BG_Skalierung->setFrameStyle( QFrame::Panel | QFrame::Sunken );
4165
4166     CB_Arbeitskurven = new QCheckBox( "1.) Arbeitskurven", BG_Skalierung );
4167
4168     if(dmanuelleArbeitskurven == 1)
4169     {
4170         CB_Arbeitskurven->setChecked( true );
4171     }
4172     else if(dmanuelleArbeitskurven == 2)
4173     {
4174         CB_Arbeitskurven->setChecked( false );
4175     }
4176     else
4177     {
4178         CB_Arbeitskurven->setChecked( false );

```



```

4179     }
4180
4181     CB_s_phi = new QCheckBox( "2.) s(phi); s'(phi)", BG_Skalierung );
4182
4183     if(dmanuelles_phi == 1)
4184     {
4185         CB_s_phi->setChecked( true );
4186     }
4187     else if(dmanuelles_phi == 2)
4188     {
4189         CB_s_phi->setChecked( false );
4190     }
4191     else
4192     {
4193         CB_s_phi->setChecked( false );
4194     }
4195
4196     CB_psi_phi = new QCheckBox( "3.) psi(phi); psi'(phi); psi''(phi)",
4197         BG_Skalierung );
4198
4199     if(dmanuellepsi_phi == 1)
4200     {
4201         CB_psi_phi->setChecked( true );
4202     }
4203     else if(dmanuellepsi_phi == 2)
4204     {
4205         CB_psi_phi->setChecked( false );
4206     }
4207     else
4208     {
4209         CB_psi_phi->setChecked( false );
4210     }
4211
4212     CB_Kurvenscheibe = new QCheckBox( "4.) Kurvenscheibe", BG_Skalierung );
4213
4214     if(dmanuelleKurvenscheibe == 1)
4215     {
4216         CB_Kurvenscheibe->setChecked( true );
4217     }
4218     else if(dmanuelleKurvenscheibe == 2)
4219     {
4220         CB_Kurvenscheibe->setChecked( false );
4221     }
4222     else

```

```

4223     {
4224         CB_Kurvenscheibe->setChecked( false );
4225     }
4226
4227     CB_x_Bik = new QCheckBox( "5.) x_Bik(phi); x'_Bik(phi)", BG_Skalierung );
4228
4229     if(dmanuellex_Bik == 1)
4230     {
4231         CB_x_Bik->setChecked( true );
4232     }
4233     else if(dmanuellex_Bik == 2)
4234     {
4235         CB_x_Bik->setChecked( false );
4236     }
4237     else
4238     {
4239         CB_x_Bik->setChecked( false );
4240     }
4241
4242     CB_y_Bik = new QCheckBox( "6.) y_Bik(phi); y'_Bik(phi)", BG_Skalierung );
4243
4244     if(dmanuelley_Bik == 1)
4245     {
4246         CB_y_Bik->setChecked( true );
4247     }
4248     else if(dmanuelley_Bik == 2)
4249     {
4250         CB_y_Bik->setChecked( false );
4251     }
4252     else
4253     {
4254         CB_y_Bik->setChecked( false );
4255     }
4256
4257     CB_Uebertragungswinkel = new QCheckBox( "7.) Übertragungswinkel",
4258         BG_Skalierung );
4259
4260     if(dmanuelleUebertragungswinkel == 1)
4261     {
4262         CB_Uebertragungswinkel->setChecked( true );
4263     }
4264     else if(dmanuelleUebertragungswinkel == 2)
4265     {
4266         CB_Uebertragungswinkel->setChecked( false );

```

```

4267     }
4268     else
4269     {
4270         CB_Uebertragungswinkel->setChecked( false );
4271     }
4272
4273     CB_Kruemmungsradius = new QCheckBox( "8.) Krümmungsradius", BG_Skalierung );
4274
4275     if(dmanuelleKruemmungsradius == 1)
4276     {
4277         CB_Kruemmungsradius->setChecked( true );
4278     }
4279     else if(dmanuelleKruemmungsradius == 2)
4280     {
4281         CB_Kruemmungsradius->setChecked( false );
4282     }
4283     else
4284     {
4285         CB_Kruemmungsradius->setChecked( false );
4286     }
4287
4288     CB_Mittelpunktskurven = new QCheckBox( "9.) Mittelpunktskurven",
4289         BG_Skalierung );
4290
4291     if(indexKurve==0)
4292     {
4293         CB_Mittelpunktskurven->setEnabled(FALSE);
4294
4295         if(dmanuelleMittelpunktskurven == 1)
4296         {
4297             CB_Mittelpunktskurven->setChecked( true );
4298         }
4299         else if(dmanuelleMittelpunktskurven == 2)
4300         {
4301             CB_Mittelpunktskurven->setChecked( false );
4302         }
4303         else
4304         {
4305             CB_Mittelpunktskurven->setChecked( false );
4306         }
4307     }
4308     else if(indexKurve==1)
4309     {
4310         CB_Mittelpunktskurven->setEnabled(TRUE);

```

```

4311
4312     if(dmanuelleMittelpunktskurven == 1)
4313     {
4314         CB_Mittelpunktskurven->setChecked( true );
4315     }
4316     else if(dmanuelleMittelpunktskurven == 2)
4317     {
4318         CB_Mittelpunktskurven->setChecked( false );
4319     }
4320     else
4321     {
4322         CB_Mittelpunktskurven->setChecked( false );
4323     }
4324 }
4325 else
4326 {
4327     CB_Mittelpunktskurven->setEnabled(FALSE);
4328
4329     if(dmanuelleMittelpunktskurven == 1)
4330     {
4331         CB_Mittelpunktskurven->setChecked( true );
4332     }
4333     else if(dmanuelleMittelpunktskurven == 2)
4334     {
4335         CB_Mittelpunktskurven->setChecked( false );
4336     }
4337     else
4338     {
4339         CB_Mittelpunktskurven->setChecked( false );
4340     }
4341 }
4342
4343     QL_Arbeitskurven = new QLabel( "1.",fourthpage );
4344     QL_Arbeitskurven->setGeometry( 15, 125, 20, 25);
4345     QL_Arbeitskurven->setEnabled(FALSE);
4346
4347     QL_x_minArbkurven = new QLabel( "x_min",fourthpage );
4348     QL_x_minArbkurven->setGeometry( 41, 125, 70, 25);
4349     QL_x_minArbkurven->setEnabled(FALSE);
4350     QToolTip::add( QL_x_minArbkurven, "unterer Plotwert der x-Koordinate" );
4351     QLE_x_minArbkurven = new QLineEdit( fourthpage );
4352     QLE_x_minArbkurven->setGeometry( 78, 125, 54, 25);
4353     QString QS_x_minArbkurven=QLE_x_minArbkurven->text();
4354     QS_x_minArbkurven.setNum( dx_minArbkurven );

```

```

4355     QLE_x_minArbkurven->setText( QS_x_minArbkurven );
4356     QLE_x_minArbkurven->setPalette( QPalette( farbe, farbe ) );
4357     QLE_x_minArbkurven->setEnabled(FALSE);
4358
4359     QL_x_maxArbkurven = new QLabel( "x_max",fourthpage );
4360     QL_x_maxArbkurven->setGeometry( 143, 125, 70, 25);
4361     QL_x_maxArbkurven->setEnabled(FALSE);
4362     QToolTip::add( QL_x_maxArbkurven, "oberer Plotwert der x-Koordinate" );
4363     QLE_x_maxArbkurven = new QLineEdit( fourthpage );
4364     QLE_x_maxArbkurven->setGeometry( 187, 125, 54, 25);
4365     QString QS_x_maxArbkurven=QLE_x_maxArbkurven->text();
4366     QS_x_maxArbkurven.setNum( dx_maxArbkurven );
4367     QLE_x_maxArbkurven->setText( QS_x_maxArbkurven );
4368     QLE_x_maxArbkurven->setPalette( QPalette( farbe, farbe ) );
4369     QLE_x_maxArbkurven->setEnabled(FALSE);
4370
4371     QL_dxArbkurven = new QLabel( "dx",fourthpage );
4372     QL_dxArbkurven->setGeometry( 252, 125, 70, 25);
4373     QL_dxArbkurven->setEnabled(FALSE);
4374     QToolTip::add( QL_dxArbkurven, "unterer Plotwert der x-Koordinate" );
4375     QLE_dxArbkurven = new QLineEdit( fourthpage );
4376     QLE_dxArbkurven->setGeometry( 270, 125, 54, 25);
4377     QString QS_dxArbkurven=QLE_dxArbkurven->text();
4378     QS_dxArbkurven.setNum( ddxArbkurven );
4379     QLE_dxArbkurven->setText( QS_dxArbkurven );
4380     QLE_dxArbkurven->setPalette( QPalette( farbe, farbe ) );
4381     QLE_dxArbkurven->setEnabled(FALSE);
4382
4383     QL_y_minArbkurven = new QLabel( "y_min",fourthpage );
4384     QL_y_minArbkurven->setGeometry( 329, 125, 70, 25);
4385     QL_y_minArbkurven->setEnabled(FALSE);
4386     QToolTip::add( QL_y_minArbkurven, "unterer Plotwert der x-Koordinate" );
4387     QLE_y_minArbkurven = new QLineEdit( fourthpage );
4388     QLE_y_minArbkurven->setGeometry( 366, 125, 54, 25);
4389     QString QS_y_minArbkurven=QLE_y_minArbkurven->text();
4390     QS_y_minArbkurven.setNum( dy_minArbkurven );
4391     QLE_y_minArbkurven->setText( QS_y_minArbkurven );
4392     QLE_y_minArbkurven->setPalette( QPalette( farbe, farbe ) );
4393     QLE_y_minArbkurven->setEnabled(FALSE);
4394
4395     QL_y_maxArbkurven = new QLabel( "y_max",fourthpage );
4396     QL_y_maxArbkurven->setGeometry( 431, 125, 70, 25);
4397     QL_y_maxArbkurven->setEnabled(FALSE);
4398     QToolTip::add( QL_y_maxArbkurven, "oberer Plotwert der x-Koordinate" );

```

```

4399     QLE_y_maxArbkurven = new QLineEdit( fourthpage );
4400     QLE_y_maxArbkurven->setGeometry( 475, 125, 54, 25);
4401     QString QS_y_maxArbkurven=QLE_y_maxArbkurven->text();
4402     QS_y_maxArbkurven.setNum( dy_maxArbkurven );
4403     QLE_y_maxArbkurven->setText( QS_y_maxArbkurven );
4404     QLE_y_maxArbkurven->setPalette( QPalette( farbe, farbe ) );
4405     QLE_y_maxArbkurven->setEnabled(FALSE);
4406
4407     QL_dyArbkurven = new QLabel( "dy",fourthpage );
4408     QL_dyArbkurven->setGeometry( 540, 125, 70, 25);
4409     QL_dyArbkurven->setEnabled(FALSE);
4410     QToolTip::add( QL_dyArbkurven, "unterer Plotwert der x-Koordinate" );
4411     QLE_dyArbkurven = new QLineEdit( fourthpage );
4412     QLE_dyArbkurven->setGeometry( 558, 125, 54, 25);
4413     QString QS_dyArbkurven=QLE_dyArbkurven->text();
4414     QS_dyArbkurven.setNum( ddyArbkurven );
4415     QLE_dyArbkurven->setText( QS_dyArbkurven );
4416     QLE_dyArbkurven->setPalette( QPalette( farbe, farbe ) );
4417     QLE_dyArbkurven->setEnabled(FALSE);
4418
4419     //.....
4420     QL_s_phi = new QLabel( "2.)",fourthpage );
4421     QL_s_phi->setGeometry( 15, 150, 20, 25);
4422     QL_s_phi->setEnabled(FALSE);
4423
4424     QL_x_mins_phi = new QLabel( "x_min",fourthpage );
4425     QL_x_mins_phi->setGeometry( 41, 150, 70, 25);
4426     QL_x_mins_phi->setEnabled(FALSE);
4427     QToolTip::add( QL_x_mins_phi, "unterer Plotwert der x-Koordinate" );
4428     QLE_x_mins_phi = new QLineEdit( fourthpage );
4429     QLE_x_mins_phi->setGeometry( 78, 150, 54, 25);
4430     QString QS_x_mins_phi=QLE_x_mins_phi->text();
4431     QS_x_mins_phi.setNum( dx_mins_phi );
4432     QLE_x_mins_phi->setText( QS_x_mins_phi );
4433     QLE_x_mins_phi->setPalette( QPalette( farbe, farbe ) );
4434     QLE_x_mins_phi->setEnabled(FALSE);
4435
4436     QL_x_maxs_phi = new QLabel( "x_max",fourthpage );
4437     QL_x_maxs_phi->setGeometry( 143, 150, 70, 25);
4438     QL_x_maxs_phi->setEnabled(FALSE);
4439     QToolTip::add( QL_x_maxs_phi, "oberer Plotwert der x-Koordinate" );
4440     QLE_x_maxs_phi = new QLineEdit( fourthpage );
4441     QLE_x_maxs_phi->setGeometry( 187, 150, 54, 25);
4442     QString QS_x_maxs_phi=QLE_x_maxs_phi->text();

```

```

4443     QS_x_maxs_phi.setNum( dx_maxs_phi );
4444     QLE_x_maxs_phi->setText( QS_x_maxs_phi );
4445     QLE_x_maxs_phi->setPalette( QPalette( farbe, farbe ) );
4446     QLE_x_maxs_phi->setEnabled(FALSE);
4447
4448     QL_dxs_phi = new QLabel( "dx",fourthpage );
4449     QL_dxs_phi->setGeometry( 252, 150, 70, 25);
4450     QL_dxs_phi->setEnabled(FALSE);
4451     QToolTip::add( QL_dxs_phi, "unterer Plotwert der x-Koordinate" );
4452     QLE_dxs_phi = new QLineEdit( fourthpage );
4453     QLE_dxs_phi->setGeometry( 270, 150, 54, 25);
4454     QString QS_dxs_phi=QLE_dxs_phi->text();
4455     QS_dxs_phi.setNum( ddxs_phi );
4456     QLE_dxs_phi->setText( QS_dxs_phi );
4457     QLE_dxs_phi->setPalette( QPalette( farbe, farbe ) );
4458     QLE_dxs_phi->setEnabled(FALSE);
4459
4460     QL_y_mins_phi = new QLabel( "y_min",fourthpage );
4461     QL_y_mins_phi->setGeometry( 329, 150, 70, 25);
4462     QL_y_mins_phi->setEnabled(FALSE);
4463     QToolTip::add( QL_y_mins_phi, "unterer Plotwert der x-Koordinate" );
4464     QLE_y_mins_phi = new QLineEdit( fourthpage );
4465     QLE_y_mins_phi->setGeometry( 366, 150, 54, 25);
4466     QString QS_y_mins_phi=QLE_y_mins_phi->text();
4467     QS_y_mins_phi.setNum( dy_mins_phi );
4468     QLE_y_mins_phi->setText( QS_y_mins_phi );
4469     QLE_y_mins_phi->setPalette( QPalette( farbe, farbe ) );
4470     QLE_y_mins_phi->setEnabled(FALSE);
4471
4472     QL_y_maxs_phi = new QLabel( "y_max",fourthpage );
4473     QL_y_maxs_phi->setGeometry( 431, 150, 70, 25);
4474     QL_y_maxs_phi->setEnabled(FALSE);
4475     QToolTip::add( QL_y_maxs_phi, "oberer Plotwert der x-Koordinate" );
4476     QLE_y_maxs_phi = new QLineEdit( fourthpage );
4477     QLE_y_maxs_phi->setGeometry( 475, 150, 54, 25);
4478     QString QS_y_maxs_phi=QLE_y_maxs_phi->text();
4479     QS_y_maxs_phi.setNum( dy_maxs_phi );
4480     QLE_y_maxs_phi->setText( QS_y_maxs_phi );
4481     QLE_y_maxs_phi->setPalette( QPalette( farbe, farbe ) );
4482     QLE_y_maxs_phi->setEnabled(FALSE);
4483
4484     QL_dys_phi = new QLabel( "dy",fourthpage );
4485     QL_dys_phi->setGeometry( 540, 150, 70, 25);
4486     QL_dys_phi->setEnabled(FALSE);

```

```

4487     QToolTip::add( QL_dys_phi, "unterer Plotwert der x-Koordinate" );
4488     QLE_dys_phi = new QLineEdit( fourthpage );
4489     QLE_dys_phi->setGeometry( 558, 150, 54, 25);
4490     QString QS_dys_phi=QLE_dys_phi->text();
4491     QS_dys_phi.setNum( ddys_phi );
4492     QLE_dys_phi->setText( QS_dys_phi );
4493     QLE_dys_phi->setPalette( QPalette( farbe, farbe ) );
4494     QLE_dys_phi->setEnabled(FALSE);
4495
4496     //.....
4497     QL_psi_phi = new QLabel( "3.",fourthpage );
4498     QL_psi_phi->setGeometry( 15, 175, 20, 25);
4499     QL_psi_phi->setEnabled(FALSE);
4500
4501     QL_x_minpsi_phi = new QLabel( "x_min",fourthpage );
4502     QL_x_minpsi_phi->setGeometry( 41, 175, 70, 25);
4503     QL_x_minpsi_phi->setEnabled(FALSE);
4504     QToolTip::add( QL_x_minpsi_phi, "unterer Plotwert der x-Koordinate" );
4505     QLE_x_minpsi_phi = new QLineEdit( fourthpage );
4506     QLE_x_minpsi_phi->setGeometry( 78, 175, 54, 25);
4507     QString QS_x_minpsi_phi=QLE_x_minpsi_phi->text();
4508     QS_x_minpsi_phi.setNum( dx_minpsi_phi );
4509     QLE_x_minpsi_phi->setText( QS_x_minpsi_phi );
4510     QLE_x_minpsi_phi->setPalette( QPalette( farbe, farbe ) );
4511     QLE_x_minpsi_phi->setEnabled(FALSE);
4512
4513     QL_x_maxpsi_phi = new QLabel( "x_max",fourthpage );
4514     QL_x_maxpsi_phi->setGeometry( 143, 175, 70, 25);
4515     QL_x_maxpsi_phi->setEnabled(FALSE);
4516     QToolTip::add( QL_x_maxpsi_phi, "oberer Plotwert der x-Koordinate" );
4517     QLE_x_maxpsi_phi = new QLineEdit( fourthpage );
4518     QLE_x_maxpsi_phi->setGeometry( 187, 175, 54, 25);
4519     QString QS_x_maxpsi_phi=QLE_x_maxpsi_phi->text();
4520     QS_x_maxpsi_phi.setNum( dx_maxpsi_phi );
4521     QLE_x_maxpsi_phi->setText( QS_x_maxpsi_phi );
4522     QLE_x_maxpsi_phi->setPalette( QPalette( farbe, farbe ) );
4523     QLE_x_maxpsi_phi->setEnabled(FALSE);
4524
4525     QL_dxpsi_phi = new QLabel( "dx",fourthpage );
4526     QL_dxpsi_phi->setGeometry( 252, 175, 70, 25);
4527     QL_dxpsi_phi->setEnabled(FALSE);
4528     QToolTip::add( QL_dxpsi_phi, "unterer Plotwert der x-Koordinate" );
4529     QLE_dxpsi_phi = new QLineEdit( fourthpage );
4530     QLE_dxpsi_phi->setGeometry( 270, 175, 54, 25);

```



```

4531     QString QS_dxpsi_phi=QLE_dxpsi_phi->text();
4532     QS_dxpsi_phi.setNum( ddxpsi_phi );
4533     QLE_dxpsi_phi->setText( QS_dxpsi_phi );
4534     QLE_dxpsi_phi->setPalette( QPalette( farbe, farbe ) );
4535     QLE_dxpsi_phi->setEnabled(FALSE);
4536
4537     QL_y_minpsi_phi = new QLabel( "y_min",fourthpage );
4538     QL_y_minpsi_phi->setGeometry( 329, 175, 70, 25);
4539     QL_y_minpsi_phi->setEnabled(FALSE);
4540     QToolTip::add( QL_y_minpsi_phi, "unterer Plotwert der x-Koordinate" );
4541     QLE_y_minpsi_phi = new QLineEdit( fourthpage );
4542     QLE_y_minpsi_phi->setGeometry( 366, 175, 54, 25);
4543     QString QS_y_minpsi_phi=QLE_y_minpsi_phi->text();
4544     QS_y_minpsi_phi.setNum( dy_minpsi_phi );
4545     QLE_y_minpsi_phi->setText( QS_y_minpsi_phi );
4546     QLE_y_minpsi_phi->setPalette( QPalette( farbe, farbe ) );
4547     QLE_y_minpsi_phi->setEnabled(FALSE);
4548
4549     QL_y_maxpsi_phi = new QLabel( "y_max",fourthpage );
4550     QL_y_maxpsi_phi->setGeometry( 431, 175, 70, 25);
4551     QL_y_maxpsi_phi->setEnabled(FALSE);
4552     QToolTip::add( QL_y_maxpsi_phi, "oberer Plotwert der x-Koordinate" );
4553     QLE_y_maxpsi_phi = new QLineEdit( fourthpage );
4554     QLE_y_maxpsi_phi->setGeometry( 475, 175, 54, 25);
4555     QString QS_y_maxpsi_phi=QLE_y_maxpsi_phi->text();
4556     QS_y_maxpsi_phi.setNum( dy_maxpsi_phi );
4557     QLE_y_maxpsi_phi->setText( QS_y_maxpsi_phi );
4558     QLE_y_maxpsi_phi->setPalette( QPalette( farbe, farbe ) );
4559     QLE_y_maxpsi_phi->setEnabled(FALSE);
4560
4561     QL_dypsi_phi = new QLabel( "dy",fourthpage );
4562     QL_dypsi_phi->setGeometry( 540, 175, 70, 25);
4563     QL_dypsi_phi->setEnabled(FALSE);
4564     QToolTip::add( QL_dypsi_phi, "unterer Plotwert der x-Koordinate" );
4565     QLE_dypsi_phi = new QLineEdit( fourthpage );
4566     QLE_dypsi_phi->setGeometry( 558, 175, 54, 25);
4567     QString QS_dypsi_phi=QLE_dypsi_phi->text();
4568     QS_dypsi_phi.setNum( ddypsi_phi );
4569     QLE_dypsi_phi->setText( QS_dypsi_phi );
4570     QLE_dypsi_phi->setPalette( QPalette( farbe, farbe ) );
4571     QLE_dypsi_phi->setEnabled(FALSE);
4572
4573     //.....
4574     QL_Kurvenscheibe = new QLabel( "4.",fourthpage );

```

```

4575     QL_Kurvenscheibe->setGeometry( 15, 200, 20, 25);
4576     QL_Kurvenscheibe->setEnabled(FALSE);
4577
4578     QL_x_minKurvenscheibe = new QLabel( "x_min",fourthpage );
4579     QL_x_minKurvenscheibe->setGeometry( 41, 200, 70, 25);
4580     QL_x_minKurvenscheibe->setEnabled(FALSE);
4581     QToolTip::add( QL_x_minKurvenscheibe, "unterer Plotwert der x-Koordinate" );
4582     QLE_x_minKurvenscheibe = new QLineEdit( fourthpage );
4583     QLE_x_minKurvenscheibe->setGeometry( 78, 200, 54, 25);
4584     QString QS_x_minKurvenscheibe=QLE_x_minKurvenscheibe->text();
4585     QS_x_minKurvenscheibe.setNum( dx_minKurvenscheibe );
4586     QLE_x_minKurvenscheibe->setText( QS_x_minKurvenscheibe );
4587     QLE_x_minKurvenscheibe->setPalette( QPalette( farbe, farbe ) );
4588     QLE_x_minKurvenscheibe->setEnabled(FALSE);
4589
4590     QL_x_maxKurvenscheibe = new QLabel( "x_max",fourthpage );
4591     QL_x_maxKurvenscheibe->setGeometry( 143, 200, 70, 25);
4592     QL_x_maxKurvenscheibe->setEnabled(FALSE);
4593     QToolTip::add( QL_x_maxKurvenscheibe, "oberer Plotwert der x-Koordinate" );
4594     QLE_x_maxKurvenscheibe = new QLineEdit( fourthpage );
4595     QLE_x_maxKurvenscheibe->setGeometry( 187, 200, 54, 25);
4596     QString QS_x_maxKurvenscheibe=QLE_x_maxKurvenscheibe->text();
4597     QS_x_maxKurvenscheibe.setNum( dx_maxKurvenscheibe );
4598     QLE_x_maxKurvenscheibe->setText( QS_x_maxKurvenscheibe );
4599     QLE_x_maxKurvenscheibe->setPalette( QPalette( farbe, farbe ) );
4600     QLE_x_maxKurvenscheibe->setEnabled(FALSE);
4601
4602     QL_dxKurvenscheibe = new QLabel( "dx",fourthpage );
4603     QL_dxKurvenscheibe->setGeometry( 252, 200, 70, 25);
4604     QL_dxKurvenscheibe->setEnabled(FALSE);
4605     QToolTip::add( QL_dxKurvenscheibe, "unterer Plotwert der x-Koordinate" );
4606     QLE_dxKurvenscheibe = new QLineEdit( fourthpage );
4607     QLE_dxKurvenscheibe->setGeometry( 270, 200, 54, 25);
4608     QString QS_dxKurvenscheibe=QLE_dxKurvenscheibe->text();
4609     QS_dxKurvenscheibe.setNum( ddxKurvenscheibe );
4610     QLE_dxKurvenscheibe->setText( QS_dxKurvenscheibe );
4611     QLE_dxKurvenscheibe->setPalette( QPalette( farbe, farbe ) );
4612     QLE_dxKurvenscheibe->setEnabled(FALSE);
4613
4614     QL_y_minKurvenscheibe = new QLabel( "y_min",fourthpage );
4615     QL_y_minKurvenscheibe->setGeometry( 329, 200, 70, 25);
4616     QL_y_minKurvenscheibe->setEnabled(FALSE);
4617     QToolTip::add( QL_y_minKurvenscheibe, "unterer Plotwert der x-Koordinate" );
4618     QLE_y_minKurvenscheibe = new QLineEdit( fourthpage );

```

```

4619     QLE_y_minKurvenscheibe->setGeometry( 366, 200, 54, 25);
4620     QString QS_y_minKurvenscheibe=QLE_y_minKurvenscheibe->text();
4621     QS_y_minKurvenscheibe.setNum( dy_minKurvenscheibe );
4622     QLE_y_minKurvenscheibe->setText( QS_y_minKurvenscheibe );
4623     QLE_y_minKurvenscheibe->setPalette( QPalette( farbe, farbe ) );
4624     QLE_y_minKurvenscheibe->setEnabled(FALSE);
4625
4626     QL_y_maxKurvenscheibe = new QLabel( "y_max",fourthpage );
4627     QL_y_maxKurvenscheibe->setGeometry( 431, 200, 70, 25);
4628     QL_y_maxKurvenscheibe->setEnabled(FALSE);
4629     QToolTip::add( QL_y_maxKurvenscheibe, "oberer Plotwert der x-Koordinate" );
4630     QLE_y_maxKurvenscheibe = new QLineEdit( fourthpage );
4631     QLE_y_maxKurvenscheibe->setGeometry( 475, 200, 54, 25);
4632     QString QS_y_maxKurvenscheibe=QLE_y_maxKurvenscheibe->text();
4633     QS_y_maxKurvenscheibe.setNum( dy_maxKurvenscheibe );
4634     QLE_y_maxKurvenscheibe->setText( QS_y_maxKurvenscheibe );
4635     QLE_y_maxKurvenscheibe->setPalette( QPalette( farbe, farbe ) );
4636     QLE_y_maxKurvenscheibe->setEnabled(FALSE);
4637
4638     QL_dyKurvenscheibe = new QLabel( "dy",fourthpage );
4639     QL_dyKurvenscheibe->setGeometry( 540, 200, 70, 25);
4640     QL_dyKurvenscheibe->setEnabled(FALSE);
4641     QToolTip::add( QL_dyKurvenscheibe, "unterer Plotwert der x-Koordinate" );
4642     QLE_dyKurvenscheibe = new QLineEdit( fourthpage );
4643     QLE_dyKurvenscheibe->setGeometry( 558, 200, 54, 25);
4644     QString QS_dyKurvenscheibe=QLE_dyKurvenscheibe->text();
4645     QS_dyKurvenscheibe.setNum( ddyKurvenscheibe );
4646     QLE_dyKurvenscheibe->setText( QS_dyKurvenscheibe );
4647     QLE_dyKurvenscheibe->setPalette( QPalette( farbe, farbe ) );
4648     QLE_dyKurvenscheibe->setEnabled(FALSE);
4649
4650     //.....
4651     QL_x_Bik = new QLabel( "5.",fourthpage );
4652     QL_x_Bik->setGeometry( 15, 225, 20, 25);
4653     QL_x_Bik->setEnabled(FALSE);
4654
4655     QL_x_minx_Bik = new QLabel( "x_min",fourthpage );
4656     QL_x_minx_Bik->setGeometry( 41, 225, 70, 25);
4657     QL_x_minx_Bik->setEnabled(FALSE);
4658     QToolTip::add( QL_x_minx_Bik, "unterer Plotwert der x-Koordinate" );
4659     QLE_x_minx_Bik = new QLineEdit( fourthpage );
4660     QLE_x_minx_Bik->setGeometry( 78, 225, 54, 25);
4661     QString QS_x_minx_Bik=QLE_x_minx_Bik->text();
4662     QS_x_minx_Bik.setNum( dx_minx_Bik );

```

```

4663     QLE_x_minx_Bik->setText( QS_x_minx_Bik );
4664     QLE_x_minx_Bik->setPalette( QPalette( farbe, farbe ) );
4665     QLE_x_minx_Bik->setEnabled(FALSE);
4666
4667     QL_x_maxx_Bik = new QLabel( "x_max",fourthpage );
4668     QL_x_maxx_Bik->setGeometry( 143, 225, 70, 25);
4669     QL_x_maxx_Bik->setEnabled(FALSE);
4670     QToolTip::add( QL_x_maxx_Bik, "oberer Plotwert der x-Koordinate" );
4671     QLE_x_maxx_Bik = new QLineEdit( fourthpage );
4672     QLE_x_maxx_Bik->setGeometry( 187, 225, 54, 25);
4673     QString QS_x_maxx_Bik=QLE_x_maxx_Bik->text();
4674     QS_x_maxx_Bik.setNum( dx_maxx_Bik );
4675     QLE_x_maxx_Bik->setText( QS_x_maxx_Bik );
4676     QLE_x_maxx_Bik->setPalette( QPalette( farbe, farbe ) );
4677     QLE_x_maxx_Bik->setEnabled(FALSE);
4678
4679     QL_dxx_Bik = new QLabel( "dx",fourthpage );
4680     QL_dxx_Bik->setGeometry( 252, 225, 70, 25);
4681     QL_dxx_Bik->setEnabled(FALSE);
4682     QToolTip::add( QL_dxx_Bik, "unterer Plotwert der x-Koordinate" );
4683     QLE_dxx_Bik = new QLineEdit( fourthpage );
4684     QLE_dxx_Bik->setGeometry( 270, 225, 54, 25);
4685     QString QS_dxx_Bik=QLE_dxx_Bik->text();
4686     QS_dxx_Bik.setNum( ddxx_Bik );
4687     QLE_dxx_Bik->setText( QS_dxx_Bik );
4688     QLE_dxx_Bik->setPalette( QPalette( farbe, farbe ) );
4689     QLE_dxx_Bik->setEnabled(FALSE);
4690
4691     QL_y_minx_Bik = new QLabel( "y_min",fourthpage );
4692     QL_y_minx_Bik->setGeometry( 329, 225, 70, 25);
4693     QL_y_minx_Bik->setEnabled(FALSE);
4694     QToolTip::add( QL_y_minx_Bik, "unterer Plotwert der x-Koordinate" );
4695     QLE_y_minx_Bik = new QLineEdit( fourthpage );
4696     QLE_y_minx_Bik->setGeometry( 366, 225, 54, 25);
4697     QString QS_y_minx_Bik=QLE_y_minx_Bik->text();
4698     QS_y_minx_Bik.setNum( dy_minx_Bik );
4699     QLE_y_minx_Bik->setText( QS_y_minx_Bik );
4700     QLE_y_minx_Bik->setPalette( QPalette( farbe, farbe ) );
4701     QLE_y_minx_Bik->setEnabled(FALSE);
4702
4703     QL_y_maxx_Bik = new QLabel( "y_max",fourthpage );
4704     QL_y_maxx_Bik->setGeometry( 431, 225, 70, 25);
4705     QL_y_maxx_Bik->setEnabled(FALSE);
4706     QToolTip::add( QL_y_maxx_Bik, "oberer Plotwert der x-Koordinate" );

```

```

4707     QLE_y_maxx_Bik = new QLineEdit( fourthpage );
4708     QLE_y_maxx_Bik->setGeometry( 475, 225, 54, 25);
4709     QString QS_y_maxx_Bik=QLE_y_maxx_Bik->text();
4710     QS_y_maxx_Bik.setNum( dy_maxx_Bik );
4711     QLE_y_maxx_Bik->setText( QS_y_maxx_Bik );
4712     QLE_y_maxx_Bik->setPalette( QPalette( farbe, farbe ) );
4713     QLE_y_maxx_Bik->setEnabled(FALSE);
4714
4715     QL_dyx_Bik = new QLabel( "dy",fourthpage );
4716     QL_dyx_Bik->setGeometry( 540, 225, 70, 25);
4717     QL_dyx_Bik->setEnabled(FALSE);
4718     QToolTip::add(QL_dyx_Bik, "unterer Plotwert der x-Koordinate" );
4719     QLE_dyx_Bik = new QLineEdit( fourthpage );
4720     QLE_dyx_Bik->setGeometry( 558, 225, 54, 25);
4721     QString QS_dyx_Bik=QLE_dyx_Bik->text();
4722     QS_dyx_Bik.setNum( ddyx_Bik );
4723     QLE_dyx_Bik->setText( QS_dyx_Bik );
4724     QLE_dyx_Bik->setPalette( QPalette( farbe, farbe ) );
4725     QLE_dyx_Bik->setEnabled(FALSE);
4726
4727     //.....
4728     QL_y_Bik = new QLabel( "6.)",fourthpage );
4729     QL_y_Bik->setGeometry( 15, 250, 20, 25);
4730     QL_y_Bik->setEnabled(FALSE);
4731
4732     QL_x_miny_Bik = new QLabel( "x_min",fourthpage );
4733     QL_x_miny_Bik->setGeometry( 41, 250, 70, 25);
4734     QL_x_miny_Bik->setEnabled(FALSE);
4735     QToolTip::add( QL_x_miny_Bik, "unterer Plotwert der x-Koordinate" );
4736     QLE_x_miny_Bik = new QLineEdit( fourthpage );
4737     QLE_x_miny_Bik->setGeometry( 78, 250, 54, 25);
4738     QString QS_x_miny_Bik=QLE_x_miny_Bik->text();
4739     QS_x_miny_Bik.setNum( dx_miny_Bik );
4740     QLE_x_miny_Bik->setText( QS_x_miny_Bik );
4741     QLE_x_miny_Bik->setPalette( QPalette( farbe, farbe ) );
4742     QLE_x_miny_Bik->setEnabled(FALSE);
4743
4744     QL_x_maxy_Bik = new QLabel( "x_max",fourthpage );
4745     QL_x_maxy_Bik->setGeometry( 143, 250, 70, 25);
4746     QL_x_maxy_Bik->setEnabled(FALSE);
4747     QToolTip::add( QL_x_maxy_Bik, "oberer Plotwert der x-Koordinate" );
4748     QLE_x_maxy_Bik = new QLineEdit( fourthpage );
4749     QLE_x_maxy_Bik->setGeometry( 187, 250, 54, 25);
4750     QString QS_x_maxy_Bik=QLE_x_maxy_Bik->text();

```

```

4751     QS_x_maxy_Bik.setNum( dx_maxy_Bik );
4752     QLE_x_maxy_Bik->setText( QS_x_maxy_Bik );
4753     QLE_x_maxy_Bik->setPalette( QPalette( farbe, farbe ) );
4754     QLE_x_maxy_Bik->setEnabled(FALSE);
4755
4756     QL_dxy_Bik = new QLabel( "dx",fourthpage );
4757     QL_dxy_Bik->setGeometry( 252, 250, 70, 25);
4758     QL_dxy_Bik->setEnabled(FALSE);
4759     QToolTip::add( QL_dxy_Bik, "unterer Plotwert der x-Koordinate" );
4760     QLE_dxy_Bik = new QLineEdit( fourthpage );
4761     QLE_dxy_Bik->setGeometry( 270, 250, 54, 25);
4762     QString QS_dxy_Bik=QLE_dxy_Bik->text();
4763     QS_dxy_Bik.setNum( ddx_Bik );
4764     QLE_dxy_Bik->setText( QS_dxy_Bik );
4765     QLE_dxy_Bik->setPalette( QPalette( farbe, farbe ) );
4766     QLE_dxy_Bik->setEnabled(FALSE);
4767
4768     QL_y_miny_Bik = new QLabel( "y_min",fourthpage );
4769     QL_y_miny_Bik->setGeometry( 329, 250, 70, 25);
4770     QL_y_miny_Bik->setEnabled(FALSE);
4771     QToolTip::add( QL_y_miny_Bik, "unterer Plotwert der x-Koordinate" );
4772     QLE_y_miny_Bik = new QLineEdit( fourthpage );
4773     QLE_y_miny_Bik->setGeometry( 366, 250, 54, 25);
4774     QString QS_y_miny_Bik=QLE_y_miny_Bik->text();
4775     QS_y_miny_Bik.setNum( dy_miny_Bik );
4776     QLE_y_miny_Bik->setText( QS_y_miny_Bik );
4777     QLE_y_miny_Bik->setPalette( QPalette( farbe, farbe ) );
4778     QLE_y_miny_Bik->setEnabled(FALSE);
4779
4780     QL_y_maxy_Bik = new QLabel( "y_max",fourthpage );
4781     QL_y_maxy_Bik->setGeometry( 431, 250, 70, 25);
4782     QL_y_maxy_Bik->setEnabled(FALSE);
4783     QToolTip::add( QL_y_maxy_Bik, "oberer Plotwert der x-Koordinate" );
4784     QLE_y_maxy_Bik = new QLineEdit( fourthpage );
4785     QLE_y_maxy_Bik->setGeometry( 475, 250, 54, 25);
4786     QString QS_y_maxy_Bik=QLE_y_maxy_Bik->text();
4787     QS_y_maxy_Bik.setNum( dy_maxy_Bik );
4788     QLE_y_maxy_Bik->setText( QS_y_maxy_Bik );
4789     QLE_y_maxy_Bik->setPalette( QPalette( farbe, farbe ) );
4790     QLE_y_maxy_Bik->setEnabled(FALSE);
4791
4792     QL_dyy_Bik = new QLabel( "dy",fourthpage );
4793     QL_dyy_Bik->setGeometry( 540, 250, 70, 25);
4794     QL_dyy_Bik->setEnabled(FALSE);

```

```

4795     QToolTip::add( QL_dyy_Bik, "unterer Plotwert der x-Koordinate" );
4796     QLE_dyy_Bik = new QLineEdit( fourthpage );
4797     QLE_dyy_Bik->setGeometry( 558, 250, 54, 25);
4798     QString QS_dyy_Bik=QLE_dyy_Bik->text();
4799     QS_dyy_Bik.setNum( ddy_Bik );
4800     QLE_dyy_Bik->setText( QS_dyy_Bik );
4801     QLE_dyy_Bik->setPalette( QPalette( farbe, farbe ) );
4802     QLE_dyy_Bik->setEnabled(FALSE);
4803
4804     //.....
4805     QL_Uebertragungswinkel = new QLabel( "7.",fourthpage );
4806     QL_Uebertragungswinkel->setGeometry( 15, 275, 20, 25);
4807     QL_Uebertragungswinkel->setEnabled(FALSE);
4808
4809     QL_x_minUebertragungswinkel = new QLabel( "x_min",fourthpage );
4810     QL_x_minUebertragungswinkel->setGeometry( 41, 275, 70, 25);
4811     QL_x_minUebertragungswinkel->setEnabled(FALSE);
4812     QToolTip::add( QL_x_minUebertragungswinkel,
4813     "unterer Plotwert der x-Koordinate" );
4814     QLE_x_minUebertragungswinkel = new QLineEdit( fourthpage );
4815     QLE_x_minUebertragungswinkel->setGeometry( 78, 275, 54, 25);
4816     QString QS_x_minUebertragungswinkel=QLE_x_minUebertragungswinkel->text();
4817     QS_x_minUebertragungswinkel.setNum( dx_minUebertragungswinkel );
4818     QLE_x_minUebertragungswinkel->setText( QS_x_minUebertragungswinkel );
4819     QLE_x_minUebertragungswinkel->setPalette( QPalette( farbe, farbe ) );
4820     QLE_x_minUebertragungswinkel->setEnabled(FALSE);
4821
4822     QL_x_maxUebertragungswinkel = new QLabel( "x_max",fourthpage );
4823     QL_x_maxUebertragungswinkel->setGeometry( 143, 275, 70, 25);
4824     QL_x_maxUebertragungswinkel->setEnabled(FALSE);
4825     QToolTip::add( QL_x_maxUebertragungswinkel,
4826     "oberer Plotwert der x-Koordinate" );
4827     QLE_x_maxUebertragungswinkel = new QLineEdit( fourthpage );
4828     QLE_x_maxUebertragungswinkel->setGeometry( 187, 275, 54, 25);
4829     QString QS_x_maxUebertragungswinkel=QLE_x_maxUebertragungswinkel->text();
4830     QS_x_maxUebertragungswinkel.setNum( dx_maxUebertragungswinkel );
4831     QLE_x_maxUebertragungswinkel->setText( QS_x_maxUebertragungswinkel );
4832     QLE_x_maxUebertragungswinkel->setPalette( QPalette( farbe, farbe ) );
4833     QLE_x_maxUebertragungswinkel->setEnabled(FALSE);
4834
4835     QL_dxUebertragungswinkel = new QLabel( "dx",fourthpage );
4836     QL_dxUebertragungswinkel->setGeometry( 252, 275, 70, 25);
4837     QL_dxUebertragungswinkel->setEnabled(FALSE);
4838     QToolTip::add( QL_dxUebertragungswinkel,

```

```

4839     "unterer Plotwert der x-Koordinate" );
4840     QLE_dxUebertragungswinkel = new QLineEdit( fourthpage );
4841     QLE_dxUebertragungswinkel->setGeometry( 270, 275, 54, 25);
4842     QString QS_dxUebertragungswinkel=QLE_dxUebertragungswinkel->text();
4843     QS_dxUebertragungswinkel.setNum( ddxUebertragungswinkel );
4844     QLE_dxUebertragungswinkel->setText( QS_dxUebertragungswinkel );
4845     QLE_dxUebertragungswinkel->setPalette( QPalette( farbe, farbe ) );
4846     QLE_dxUebertragungswinkel->setEnabled(FALSE);
4847
4848     QL_y_minUebertragungswinkel = new QLabel( "y_min",fourthpage );
4849     QL_y_minUebertragungswinkel->setGeometry( 329, 275, 70, 25);
4850     QL_y_minUebertragungswinkel->setEnabled(FALSE);
4851     QToolTip::add( QL_y_minUebertragungswinkel,
4852     "unterer Plotwert der x-Koordinate" );
4853     QLE_y_minUebertragungswinkel = new QLineEdit( fourthpage );
4854     QLE_y_minUebertragungswinkel->setGeometry( 366, 275, 54, 25);
4855     QString QS_y_minUebertragungswinkel=QLE_y_minUebertragungswinkel->text();
4856     QS_y_minUebertragungswinkel.setNum( dy_minUebertragungswinkel );
4857     QLE_y_minUebertragungswinkel->setText( QS_y_minUebertragungswinkel );
4858     QLE_y_minUebertragungswinkel->setPalette( QPalette( farbe, farbe ) );
4859     QLE_y_minUebertragungswinkel->setEnabled(FALSE);
4860
4861     QL_y_maxUebertragungswinkel = new QLabel( "y_max",fourthpage );
4862     QL_y_maxUebertragungswinkel->setGeometry( 431, 275, 70, 25);
4863     QL_y_maxUebertragungswinkel->setEnabled(FALSE);
4864     QToolTip::add( QL_y_maxUebertragungswinkel,
4865     "oberer Plotwert der x-Koordinate" );
4866     QLE_y_maxUebertragungswinkel = new QLineEdit( fourthpage );
4867     QLE_y_maxUebertragungswinkel->setGeometry( 475, 275, 54, 25);
4868     QString QS_y_maxUebertragungswinkel=QLE_y_maxUebertragungswinkel->text();
4869     QS_y_maxUebertragungswinkel.setNum( dy_maxUebertragungswinkel );
4870     QLE_y_maxUebertragungswinkel->setText( QS_y_maxUebertragungswinkel );
4871     QLE_y_maxUebertragungswinkel->setPalette( QPalette( farbe, farbe ) );
4872     QLE_y_maxUebertragungswinkel->setEnabled(FALSE);
4873
4874     QL_dyUebertragungswinkel = new QLabel( "dy",fourthpage );
4875     QL_dyUebertragungswinkel->setGeometry( 540, 275, 70, 25);
4876     QL_dyUebertragungswinkel->setEnabled(FALSE);
4877     QToolTip::add( QL_dyUebertragungswinkel,
4878     "unterer Plotwert der x-Koordinate" );
4879     QLE_dyUebertragungswinkel = new QLineEdit( fourthpage );
4880     QLE_dyUebertragungswinkel->setGeometry( 558, 275, 54, 25);
4881     QString QS_dyUebertragungswinkel=QLE_dyUebertragungswinkel->text();
4882     QS_dyUebertragungswinkel.setNum( ddyUebertragungswinkel );

```



```

4883     QLE_dyUebertragungswinkel->setText( QS_dyUebertragungswinkel );
4884     QLE_dyUebertragungswinkel->setPalette( QPalette( farbe, farbe ) );
4885     QLE_dyUebertragungswinkel->setEnabled(FALSE);
4886
4887     //.....
4888     QL_Kruemmungsradius = new QLabel( "8.",fourthpage );
4889     QL_Kruemmungsradius->setGeometry( 15, 300, 20, 25);
4890     QL_Kruemmungsradius->setEnabled(FALSE);
4891
4892     QL_x_minKruemmungsradius = new QLabel( "x_min",fourthpage );
4893     QL_x_minKruemmungsradius->setGeometry( 41, 300, 70, 25);
4894     QL_x_minKruemmungsradius->setEnabled(FALSE);
4895     QToolTip::add( QL_x_minKruemmungsradius,
4896         "unterer Plotwert der x-Koordinate" );
4897     QLE_x_minKruemmungsradius = new QLineEdit( fourthpage );
4898     QLE_x_minKruemmungsradius->setGeometry( 78, 300, 54, 25);
4899     QString QS_x_minKruemmungsradius=QLE_x_minKruemmungsradius->text();
4900     QS_x_minKruemmungsradius.setNum( dx_minKruemmungsradius );
4901     QLE_x_minKruemmungsradius->setText( QS_x_minKruemmungsradius );
4902     QLE_x_minKruemmungsradius->setPalette( QPalette( farbe, farbe ) );
4903     QLE_x_minKruemmungsradius->setEnabled(FALSE);
4904
4905     QL_x_maxKruemmungsradius = new QLabel( "x_max",fourthpage );
4906     QL_x_maxKruemmungsradius->setGeometry( 143, 300, 70, 25);
4907     QL_x_maxKruemmungsradius->setEnabled(FALSE);
4908     QToolTip::add( QL_x_maxKruemmungsradius,
4909         "oberer Plotwert der x-Koordinate" );
4910     QLE_x_maxKruemmungsradius = new QLineEdit( fourthpage );
4911     QLE_x_maxKruemmungsradius->setGeometry( 187, 300, 54, 25);
4912     QString QS_x_maxKruemmungsradius=QLE_x_maxKruemmungsradius->text();
4913     QS_x_maxKruemmungsradius.setNum( dx_maxKruemmungsradius );
4914     QLE_x_maxKruemmungsradius->setText( QS_x_maxKruemmungsradius );
4915     QLE_x_maxKruemmungsradius->setPalette( QPalette( farbe, farbe ) );
4916     QLE_x_maxKruemmungsradius->setEnabled(FALSE);
4917
4918     QL_dxKruemmungsradius = new QLabel( "dx",fourthpage );
4919     QL_dxKruemmungsradius->setGeometry( 252, 300, 70, 25);
4920     QL_dxKruemmungsradius->setEnabled(FALSE);
4921     QToolTip::add( QL_dxKruemmungsradius,
4922         "unterer Plotwert der x-Koordinate" );
4923     QLE_dxKruemmungsradius = new QLineEdit( fourthpage );
4924     QLE_dxKruemmungsradius->setGeometry( 270, 300, 54, 25);
4925     QString QS_dxKruemmungsradius=QLE_dxKruemmungsradius->text();
4926     QS_dxKruemmungsradius.setNum( ddxKruemmungsradius );

```

```

4927     QLE_dxKruemmungsradius->setText( QS_dxKruemmungsradius );
4928     QLE_dxKruemmungsradius->setPalette( QPalette( farbe, farbe ) );
4929     QLE_dxKruemmungsradius->setEnabled(FALSE);
4930
4931     QL_y_minKruemmungsradius = new QLabel( "y_min",fourthpage );
4932     QL_y_minKruemmungsradius->setGeometry( 329, 300, 70, 25);
4933     QL_y_minKruemmungsradius->setEnabled(FALSE);
4934     QToolTip::add( QL_y_minKruemmungsradius,
4935         "unterer Plotwert der x-Koordinate" );
4936     QLE_y_minKruemmungsradius = new QLineEdit( fourthpage );
4937     QLE_y_minKruemmungsradius->setGeometry( 366, 300, 54, 25);
4938     QString QS_y_minKruemmungsradius=QLE_y_minKruemmungsradius->text();
4939     QS_y_minKruemmungsradius.setNum( dy_minKruemmungsradius );
4940     QLE_y_minKruemmungsradius->setText( QS_y_minKruemmungsradius );
4941     QLE_y_minKruemmungsradius->setPalette( QPalette( farbe, farbe ) );
4942     QLE_y_minKruemmungsradius->setEnabled(FALSE);
4943
4944     QL_y_maxKruemmungsradius = new QLabel( "y_max",fourthpage );
4945     QL_y_maxKruemmungsradius->setGeometry( 431, 300, 70, 25);
4946     QL_y_maxKruemmungsradius->setEnabled(FALSE);
4947     QToolTip::add( QL_y_maxKruemmungsradius,
4948         "oberer Plotwert der x-Koordinate" );
4949     QLE_y_maxKruemmungsradius = new QLineEdit( fourthpage );
4950     QLE_y_maxKruemmungsradius->setGeometry( 475, 300, 54, 25);
4951     QString QS_y_maxKruemmungsradius=QLE_y_maxKruemmungsradius->text();
4952     QS_y_maxKruemmungsradius.setNum( dy_maxKruemmungsradius );
4953     QLE_y_maxKruemmungsradius->setText( QS_y_maxKruemmungsradius );
4954     QLE_y_maxKruemmungsradius->setPalette( QPalette( farbe, farbe ) );
4955     QLE_y_maxKruemmungsradius->setEnabled(FALSE);
4956
4957     QL_dyKruemmungsradius = new QLabel( "dy",fourthpage );
4958     QL_dyKruemmungsradius->setGeometry( 540, 300, 70, 25);
4959     QL_dyKruemmungsradius->setEnabled(FALSE);
4960     QToolTip::add( QL_dyKruemmungsradius,
4961         "unterer Plotwert der x-Koordinate" );
4962     QLE_dyKruemmungsradius = new QLineEdit( fourthpage );
4963     QLE_dyKruemmungsradius->setGeometry( 558, 300, 54, 25);
4964     QString QS_dyKruemmungsradius=QLE_dyKruemmungsradius->text();
4965     QS_dyKruemmungsradius.setNum( ddyKruemmungsradius );
4966     QLE_dyKruemmungsradius->setText( QS_dyKruemmungsradius );
4967     QLE_dyKruemmungsradius->setPalette( QPalette( farbe, farbe ) );
4968     QLE_dyKruemmungsradius->setEnabled(FALSE);
4969
4970     //.....

```

```

4971     QL_Mittelpunktskurven = new QLabel( "9.",fourthpage );
4972     QL_Mittelpunktskurven->setGeometry( 15, 325, 20, 25);
4973     QL_Mittelpunktskurven->setEnabled(FALSE);
4974
4975     QL_x_minMittelpunktskurven = new QLabel( "x_min",fourthpage );
4976     QL_x_minMittelpunktskurven->setGeometry( 41, 325, 70, 25);
4977     QL_x_minMittelpunktskurven->setEnabled(FALSE);
4978     QToolTip::add( QL_x_minMittelpunktskurven,
4979         "unterer Plotwert der x-Koordinate" );
4980     QLE_x_minMittelpunktskurven = new QLineEdit( fourthpage );
4981     QLE_x_minMittelpunktskurven->setGeometry( 78, 325, 54, 25);
4982     QString QS_x_minMittelpunktskurven=QLE_x_minMittelpunktskurven->text();
4983     QS_x_minMittelpunktskurven.setNum( dx_minMittelpunktskurven );
4984     QLE_x_minMittelpunktskurven->setText( QS_x_minMittelpunktskurven );
4985     QLE_x_minMittelpunktskurven->setPalette( QPalette( farbe, farbe ) );
4986     QLE_x_minMittelpunktskurven->setEnabled(FALSE);
4987
4988     QL_x_maxMittelpunktskurven = new QLabel( "x_max",fourthpage );
4989     QL_x_maxMittelpunktskurven->setGeometry( 143, 325, 70, 25);
4990     QL_x_maxMittelpunktskurven->setEnabled(FALSE);
4991     QToolTip::add( QL_x_maxMittelpunktskurven,
4992         "oberer Plotwert der x-Koordinate" );
4993     QLE_x_maxMittelpunktskurven = new QLineEdit( fourthpage );
4994     QLE_x_maxMittelpunktskurven->setGeometry( 187, 325, 54, 25);
4995     QString QS_x_maxMittelpunktskurven=QLE_x_maxMittelpunktskurven->text();
4996     QS_x_maxMittelpunktskurven.setNum( dx_maxMittelpunktskurven );
4997     QLE_x_maxMittelpunktskurven->setText( QS_x_maxMittelpunktskurven );
4998     QLE_x_maxMittelpunktskurven->setPalette( QPalette( farbe, farbe ) );
4999     QLE_x_maxMittelpunktskurven->setEnabled(FALSE);
5000
5001     QL_dxMittelpunktskurven = new QLabel( "dx",fourthpage );
5002     QL_dxMittelpunktskurven->setGeometry( 252, 325, 70, 25);
5003     QL_dxMittelpunktskurven->setEnabled(FALSE);
5004     QToolTip::add( QL_dxMittelpunktskurven,
5005         "unterer Plotwert der x-Koordinate" );
5006     QLE_dxMittelpunktskurven = new QLineEdit( fourthpage );
5007     QLE_dxMittelpunktskurven->setGeometry( 270, 325, 54, 25);
5008     QString QS_dxMittelpunktskurven=QLE_dxMittelpunktskurven->text();
5009     QS_dxMittelpunktskurven.setNum( ddxMittelpunktskurven );
5010     QLE_dxMittelpunktskurven->setText( QS_dxMittelpunktskurven );
5011     QLE_dxMittelpunktskurven->setPalette( QPalette( farbe, farbe ) );
5012     QLE_dxMittelpunktskurven->setEnabled(FALSE);
5013
5014     QL_y_minMittelpunktskurven = new QLabel( "y_min",fourthpage );

```

```

5015     QL_y_minMittelpunktskurven->setGeometry( 329, 325, 70, 25);
5016     QL_y_minMittelpunktskurven->setEnabled(FALSE);
5017     QToolTip::add( QL_y_minMittelpunktskurven,
5018         "unterer Plotwert der x-Koordinate" );
5019     QLE_y_minMittelpunktskurven = new QLineEdit( fourthpage );
5020     QLE_y_minMittelpunktskurven->setGeometry( 366, 325, 54, 25);
5021     QString QS_y_minMittelpunktskurven=QLE_y_minMittelpunktskurven->text();
5022     QS_y_minMittelpunktskurven.setNum( dy_minMittelpunktskurven );
5023     QLE_y_minMittelpunktskurven->setText( QS_y_minMittelpunktskurven );
5024     QLE_y_minMittelpunktskurven->setPalette( QPalette( farbe, farbe ) );
5025     QLE_y_minMittelpunktskurven->setEnabled(FALSE);
5026
5027     QL_y_maxMittelpunktskurven = new QLabel( "y_max",fourthpage );
5028     QL_y_maxMittelpunktskurven->setGeometry( 431, 325, 70, 25);
5029     QL_y_maxMittelpunktskurven->setEnabled(FALSE);
5030     QToolTip::add( QL_y_maxMittelpunktskurven,
5031         "oberer Plotwert der x-Koordinate" );
5032     QLE_y_maxMittelpunktskurven = new QLineEdit( fourthpage );
5033     QLE_y_maxMittelpunktskurven->setGeometry( 475, 325, 54, 25);
5034     QString QS_y_maxMittelpunktskurven=QLE_y_maxMittelpunktskurven->text();
5035     QS_y_maxMittelpunktskurven.setNum( dy_maxMittelpunktskurven );
5036     QLE_y_maxMittelpunktskurven->setText( QS_y_maxMittelpunktskurven );
5037     QLE_y_maxMittelpunktskurven->setPalette( QPalette( farbe, farbe ) );
5038     QLE_y_maxMittelpunktskurven->setEnabled(FALSE);
5039
5040     QL_dyMittelpunktskurven = new QLabel( "dy",fourthpage );
5041     QL_dyMittelpunktskurven->setGeometry( 540, 325, 70, 25);
5042     QL_dyMittelpunktskurven->setEnabled(FALSE);
5043     QToolTip::add( QL_dyMittelpunktskurven,
5044         "unterer Plotwert der x-Koordinate" );
5045     QLE_dyMittelpunktskurven = new QLineEdit( fourthpage );
5046     QLE_dyMittelpunktskurven->setGeometry( 558, 325, 54, 25);
5047     QString QS_dyMittelpunktskurven=QLE_dyMittelpunktskurven->text();
5048     QS_dyMittelpunktskurven.setNum( ddyMittelpunktskurven );
5049     QLE_dyMittelpunktskurven->setText( QS_dyMittelpunktskurven );
5050     QLE_dyMittelpunktskurven->setPalette( QPalette( farbe, farbe ) );
5051     QLE_dyMittelpunktskurven->setEnabled(FALSE);
5052
5053     zeigeWahlSkalierung();
5054
5055     QObject::connect( CB_s_phi, SIGNAL( clicked() ),
5056         this, SLOT( zeigeWahlSkalierung() ) );
5057
5058     QObject::connect( CB_Arbeitskurven, SIGNAL( clicked() ),

```

```

5059             this, SLOT( zeigeWahlSkalierung() ) );
5060
5061     QObject::connect( CB_s_phi, SIGNAL( clicked() ),
5062                     this, SLOT( zeigeWahlSkalierung() ) );
5063
5064     QObject::connect( CB_psi_phi, SIGNAL( clicked() ),
5065                     this, SLOT( zeigeWahlSkalierung() ) );
5066
5067     QObject::connect( CB_Kurvenscheibe, SIGNAL( clicked() ),
5068                     this, SLOT( zeigeWahlSkalierung() ) );
5069
5070     QObject::connect( CB_x_Bik, SIGNAL( clicked() ),
5071                     this, SLOT( zeigeWahlSkalierung() ) );
5072
5073     QObject::connect( CB_y_Bik, SIGNAL( clicked() ),
5074                     this, SLOT( zeigeWahlSkalierung() ) );
5075
5076     QObject::connect( CB_Uebertragungswinkel, SIGNAL( clicked() ),
5077                     this, SLOT( zeigeWahlSkalierung() ) );
5078
5079     QObject::connect( CB_Kruemmungsradius, SIGNAL( clicked() ),
5080                     this, SLOT( zeigeWahlSkalierung() ) );
5081
5082     QObject::connect( CB_Mittelpunktskurven, SIGNAL( clicked() ),
5083                     this, SLOT( zeigeWahlSkalierung() ) );
5084
5085     //.....
5086     QObject::connect(QLE_x_minArbkurven,
5087                     SIGNAL( textChanged( const QString & ) ),
5088                     this, SLOT( neu_x_minArbkurven( const QString & ) ) );
5089
5090     QObject::connect(QLE_x_maxArbkurven,
5091                     SIGNAL( textChanged( const QString & ) ),
5092                     this, SLOT( neu_x_maxArbkurven( const QString & ) ) );
5093
5094     QObject::connect(QLE_dxArbkurven,
5095                     SIGNAL( textChanged( const QString & ) ),
5096                     this, SLOT( neu_dxArbkurven( const QString & ) ) );
5097
5098     QObject::connect(QLE_y_minArbkurven,
5099                     SIGNAL( textChanged( const QString & ) ),
5100                     this, SLOT( neu_y_minArbkurven( const QString & ) ) );
5101
5102     QObject::connect(QLE_y_maxArbkurven,

```

```

5103         SIGNAL( textChanged( const QString & ) ),
5104         this, SLOT( neu_y_maxArbkurven( const QString & ) ) );
5105
5106     QObject::connect(QLE_dyArbkurven,
5107         SIGNAL( textChanged( const QString & ) ),
5108         this, SLOT( neu_dyArbkurven( const QString & ) ) );
5109 //.....
5110     QObject::connect(QLE_x_mins_phi,
5111         SIGNAL( textChanged( const QString & ) ),
5112         this, SLOT( neu_x_mins_phi( const QString & ) ) );
5113
5114     QObject::connect(QLE_x_maxs_phi,
5115         SIGNAL( textChanged( const QString & ) ),
5116         this, SLOT( neu_x_maxs_phi( const QString & ) ) );
5117
5118     QObject::connect(QLE_dxs_phi,
5119         SIGNAL( textChanged( const QString & ) ),
5120         this, SLOT( neu_dxs_phi( const QString & ) ) );
5121
5122     QObject::connect(QLE_y_mins_phi,
5123         SIGNAL( textChanged( const QString & ) ),
5124         this, SLOT( neu_y_mins_phi( const QString & ) ) );
5125
5126     QObject::connect(QLE_y_maxs_phi,
5127         SIGNAL( textChanged( const QString & ) ),
5128         this, SLOT( neu_y_maxs_phi( const QString & ) ) );
5129
5130     QObject::connect(QLE_dys_phi,
5131         SIGNAL( textChanged( const QString & ) ),
5132         this, SLOT( neu_dys_phi( const QString & ) ) );
5133 //.....
5134     QObject::connect(QLE_x_minpsi_phi,
5135         SIGNAL( textChanged( const QString & ) ),
5136         this, SLOT( neu_x_minpsi_phi( const QString & ) ) );
5137
5138     QObject::connect(QLE_x_maxpsi_phi,
5139         SIGNAL( textChanged( const QString & ) ),
5140         this, SLOT( neu_x_maxpsi_phi( const QString & ) ) );
5141
5142     QObject::connect(QLE_dxpsi_phi,
5143         SIGNAL( textChanged( const QString & ) ),
5144         this, SLOT( neu_dxpsi_phi( const QString & ) ) );
5145
5146     QObject::connect(QLE_y_minpsi_phi,

```

```

5147         SIGNAL( textChanged( const QString & ) ),
5148         this, SLOT( neu_y_minpsi_phi( const QString & ) ) );
5149
5150     QObject::connect(QLE_y_maxpsi_phi,
5151         SIGNAL( textChanged( const QString & ) ),
5152         this, SLOT( neu_y_maxpsi_phi( const QString & ) ) );
5153
5154     QObject::connect(QLE_dypsi_phi,
5155         SIGNAL( textChanged( const QString & ) ),
5156         this, SLOT( neu_dypsi_phi( const QString & ) ) );
5157 //.....
5158     QObject::connect(QLE_x_minKurvenscheibe,
5159         SIGNAL( textChanged( const QString & ) ),
5160         this, SLOT( neu_x_minKurvenscheibe( const QString & ) ) );
5161
5162     QObject::connect(QLE_x_maxKurvenscheibe,
5163         SIGNAL( textChanged( const QString & ) ),
5164         this, SLOT( neu_x_maxKurvenscheibe( const QString & ) ) );
5165
5166     QObject::connect(QLE_dxKurvenscheibe,
5167         SIGNAL( textChanged( const QString & ) ),
5168         this, SLOT( neu_dxKurvenscheibe( const QString & ) ) );
5169
5170     QObject::connect(QLE_y_minKurvenscheibe,
5171         SIGNAL( textChanged( const QString & ) ),
5172         this, SLOT( neu_y_minKurvenscheibe( const QString & ) ) );
5173
5174     QObject::connect(QLE_y_maxKurvenscheibe,
5175         SIGNAL( textChanged( const QString & ) ),
5176         this, SLOT( neu_y_maxKurvenscheibe( const QString & ) ) );
5177
5178     QObject::connect(QLE_dyKurvenscheibe,
5179         SIGNAL( textChanged( const QString & ) ),
5180         this, SLOT( neu_dyKurvenscheibe( const QString & ) ) );
5181 //.....
5182     QObject::connect(QLE_x_minx_Bik,
5183         SIGNAL( textChanged( const QString & ) ),
5184         this, SLOT( neu_x_minx_Bik( const QString & ) ) );
5185
5186     QObject::connect(QLE_x_maxx_Bik,
5187         SIGNAL( textChanged( const QString & ) ),
5188         this, SLOT( neu_x_maxx_Bik( const QString & ) ) );
5189
5190     QObject::connect(QLE_dxx_Bik,

```

```

5191         SIGNAL( textChanged( const QString & ) ),
5192         this, SLOT( neu_dxx_Bik( const QString & ) ) );
5193
5194     QObject::connect(QLE_y_minx_Bik,
5195         SIGNAL( textChanged( const QString & ) ),
5196         this, SLOT( neu_y_minx_Bik( const QString & ) ) );
5197
5198     QObject::connect(QLE_y_maxx_Bik,
5199         SIGNAL( textChanged( const QString & ) ),
5200         this, SLOT( neu_y_maxx_Bik( const QString & ) ) );
5201
5202     QObject::connect(QLE_dyx_Bik,
5203         SIGNAL( textChanged( const QString & ) ),
5204         this, SLOT( neu_dyx_Bik( const QString & ) ) );
5205     //.....
5206     QObject::connect(QLE_x_miny_Bik,
5207         SIGNAL( textChanged( const QString & ) ),
5208         this, SLOT( neu_x_miny_Bik( const QString & ) ) );
5209
5210     QObject::connect(QLE_x_maxy_Bik,
5211         SIGNAL( textChanged( const QString & ) ),
5212         this, SLOT( neu_x_maxy_Bik( const QString & ) ) );
5213
5214     QObject::connect(QLE_dxy_Bik,
5215         SIGNAL( textChanged( const QString & ) ),
5216         this, SLOT( neu_dxy_Bik( const QString & ) ) );
5217
5218     QObject::connect(QLE_y_miny_Bik,
5219         SIGNAL( textChanged( const QString & ) ),
5220         this, SLOT( neu_y_miny_Bik( const QString & ) ) );
5221
5222     QObject::connect(QLE_y_maxy_Bik,
5223         SIGNAL( textChanged( const QString & ) ),
5224         this, SLOT( neu_y_maxy_Bik( const QString & ) ) );
5225
5226     QObject::connect(QLE_dyy_Bik,
5227         SIGNAL( textChanged( const QString & ) ),
5228         this, SLOT( neu_dyy_Bik( const QString & ) ) );
5229     //.....
5230     QObject::connect(QLE_x_minUebertragungswinkel,
5231         SIGNAL( textChanged( const QString & ) ),
5232         this, SLOT( neu_x_minUebertragungswinkel( const QString & ) ) );
5233
5234     QObject::connect(QLE_x_maxUebertragungswinkel,

```



```

5235         SIGNAL( textChanged( const QString & ) ),
5236         this, SLOT( neu_x_maxUebertragungswinkel( const QString & ) ) );
5237
5238     QObject::connect(QLE_dxUebertragungswinkel,
5239         SIGNAL( textChanged( const QString & ) ),
5240         this, SLOT( neu_dxUebertragungswinkel( const QString & ) ) );
5241
5242     QObject::connect(QLE_y_minUebertragungswinkel,
5243         SIGNAL( textChanged( const QString & ) ),
5244         this, SLOT( neu_y_minUebertragungswinkel( const QString & ) ) );
5245
5246     QObject::connect(QLE_y_maxUebertragungswinkel,
5247         SIGNAL( textChanged( const QString & ) ),
5248         this, SLOT( neu_y_maxUebertragungswinkel( const QString & ) ) );
5249
5250     QObject::connect(QLE_dyUebertragungswinkel,
5251         SIGNAL( textChanged( const QString & ) ),
5252         this, SLOT( neu_dyUebertragungswinkel( const QString & ) ) );
5253 //.....
5254     QObject::connect(QLE_x_minKruemmungsradius,
5255         SIGNAL( textChanged( const QString & ) ),
5256         this, SLOT( neu_x_minKruemmungsradius( const QString & ) ) );
5257
5258     QObject::connect(QLE_x_maxKruemmungsradius,
5259         SIGNAL( textChanged( const QString & ) ),
5260         this, SLOT( neu_x_maxKruemmungsradius( const QString & ) ) );
5261
5262     QObject::connect(QLE_dxKruemmungsradius,
5263         SIGNAL( textChanged( const QString & ) ),
5264         this, SLOT( neu_dxKruemmungsradius( const QString & ) ) );
5265
5266     QObject::connect(QLE_y_minKruemmungsradius,
5267         SIGNAL( textChanged( const QString & ) ),
5268         this, SLOT( neu_y_minKruemmungsradius( const QString & ) ) );
5269
5270     QObject::connect(QLE_y_maxKruemmungsradius,
5271         SIGNAL( textChanged( const QString & ) ),
5272         this, SLOT( neu_y_maxKruemmungsradius( const QString & ) ) );
5273
5274     QObject::connect(QLE_dyKruemmungsradius,
5275         SIGNAL( textChanged( const QString & ) ),
5276         this, SLOT( neu_dyKruemmungsradius( const QString & ) ) );
5277 //.....
5278     QObject::connect(QLE_x_minMittelpunktskurven,

```

```

5279         SIGNAL( textChanged( const QString & ) ),
5280         this, SLOT( neu_x_minMittelpunktskurven( const QString & ) ) );
5281
5282     QObject::connect(QLE_x_maxMittelpunktskurven,
5283         SIGNAL( textChanged( const QString & ) ),
5284         this, SLOT( neu_x_maxMittelpunktskurven( const QString & ) ) );
5285
5286     QObject::connect(QLE_dxMittelpunktskurven,
5287         SIGNAL( textChanged( const QString & ) ),
5288         this, SLOT( neu_dxMittelpunktskurven( const QString & ) ) );
5289
5290     QObject::connect(QLE_y_minMittelpunktskurven,
5291         SIGNAL( textChanged( const QString & ) ),
5292         this, SLOT( neu_y_minMittelpunktskurven( const QString & ) ) );
5293
5294     QObject::connect(QLE_y_maxMittelpunktskurven,
5295         SIGNAL( textChanged( const QString & ) ),
5296         this, SLOT( neu_y_maxMittelpunktskurven( const QString & ) ) );
5297
5298     QObject::connect(QLE_dyMittelpunktskurven,
5299         SIGNAL( textChanged( const QString & ) ),
5300         this, SLOT( neu_dyMittelpunktskurven( const QString & ) ) );
5301
5302 }
5303 void Opticurv::zeigeWahlSkalierung()
5304 {
5305     if( CB_Arbeitskurven->isChecked() )
5306     {
5307         QL_Arbeitskurven->setEnabled(TRUE);
5308
5309         QL_x_minArbkurven->setEnabled(TRUE);
5310         QLE_x_minArbkurven->setEnabled(TRUE);
5311
5312         QL_x_maxArbkurven->setEnabled(TRUE);
5313         QLE_x_maxArbkurven->setEnabled(TRUE);
5314
5315         QL_dxArbkurven->setEnabled(TRUE);
5316         QLE_dxArbkurven->setEnabled(TRUE);
5317
5318         QL_y_minArbkurven->setEnabled(TRUE);
5319         QLE_y_minArbkurven->setEnabled(TRUE);
5320
5321         QL_y_maxArbkurven->setEnabled(TRUE);
5322         QLE_y_maxArbkurven->setEnabled(TRUE);

```

```

5323
5324     QL_dyArbkurven->setEnabled(TRUE);
5325     QLE_dyArbkurven->setEnabled(TRUE);
5326 }
5327 if( !CB_Arbeitskurven->isChecked() )
5328 {
5329     QL_Arbeitskurven->setEnabled(FALSE);
5330
5331     QL_x_minArbkurven->setEnabled(FALSE);
5332     QLE_x_minArbkurven->setEnabled(FALSE);
5333
5334     QL_x_maxArbkurven->setEnabled(FALSE);
5335     QLE_x_maxArbkurven->setEnabled(FALSE);
5336
5337     QL_dxArbkurven->setEnabled(FALSE);
5338     QLE_dxArbkurven->setEnabled(FALSE);
5339
5340     QL_y_minArbkurven->setEnabled(FALSE);
5341     QLE_y_minArbkurven->setEnabled(FALSE);
5342
5343     QL_y_maxArbkurven->setEnabled(FALSE);
5344     QLE_y_maxArbkurven->setEnabled(FALSE);
5345
5346     QL_dyArbkurven->setEnabled(FALSE);
5347     QLE_dyArbkurven->setEnabled(FALSE);
5348 }
5349 if( CB_s_phi->isChecked() )
5350 {
5351     QL_s_phi->setEnabled(TRUE);
5352
5353     QL_x_mins_phi->setEnabled(TRUE);
5354     QLE_x_mins_phi->setEnabled(TRUE);
5355
5356     QL_x_maxs_phi->setEnabled(TRUE);
5357     QLE_x_maxs_phi->setEnabled(TRUE);
5358
5359     QL_dxs_phi->setEnabled(TRUE);
5360     QLE_dxs_phi->setEnabled(TRUE);
5361
5362     QL_y_mins_phi->setEnabled(TRUE);
5363     QLE_y_mins_phi->setEnabled(TRUE);
5364
5365     QL_y_maxs_phi->setEnabled(TRUE);
5366     QLE_y_maxs_phi->setEnabled(TRUE);

```

```

5367
5368     QL_dys_phi->setEnabled(TRUE);
5369     QLE_dys_phi->setEnabled(TRUE);
5370 }
5371 if( !CB_s_phi->isChecked() )
5372 {
5373     QL_s_phi->setEnabled(FALSE);
5374
5375     QL_x_mins_phi->setEnabled(FALSE);
5376     QLE_x_mins_phi->setEnabled(FALSE);
5377
5378     QL_x_maxs_phi->setEnabled(FALSE);
5379     QLE_x_maxs_phi->setEnabled(FALSE);
5380
5381     QL_dxs_phi->setEnabled(FALSE);
5382     QLE_dxs_phi->setEnabled(FALSE);
5383
5384     QL_y_mins_phi->setEnabled(FALSE);
5385     QLE_y_mins_phi->setEnabled(FALSE);
5386
5387     QL_y_maxs_phi->setEnabled(FALSE);
5388     QLE_y_maxs_phi->setEnabled(FALSE);
5389
5390     QL_dys_phi->setEnabled(FALSE);
5391     QLE_dys_phi->setEnabled(FALSE);
5392 }
5393 if( CB_psi_phi->isChecked() )
5394 {
5395     QL_psi_phi->setEnabled(TRUE);
5396
5397     QL_x_minpsi_phi->setEnabled(TRUE);
5398     QLE_x_minpsi_phi->setEnabled(TRUE);
5399
5400     QL_x_maxpsi_phi->setEnabled(TRUE);
5401     QLE_x_maxpsi_phi->setEnabled(TRUE);
5402
5403     QL_dxpsi_phi->setEnabled(TRUE);
5404     QLE_dxpsi_phi->setEnabled(TRUE);
5405
5406     QL_y_minpsi_phi->setEnabled(TRUE);
5407     QLE_y_minpsi_phi->setEnabled(TRUE);
5408
5409     QL_y_maxpsi_phi->setEnabled(TRUE);
5410     QLE_y_maxpsi_phi->setEnabled(TRUE);

```

```

5411
5412     QL_dypsi_phi->setEnabled(TRUE);
5413     QLE_dypsi_phi->setEnabled(TRUE);
5414 }
5415 if( !CB_psi_phi->isChecked() )
5416 {
5417     QL_psi_phi->setEnabled(FALSE);
5418
5419     QL_x_minpsi_phi->setEnabled(FALSE);
5420     QLE_x_minpsi_phi->setEnabled(FALSE);
5421
5422     QL_x_maxpsi_phi->setEnabled(FALSE);
5423     QLE_x_maxpsi_phi->setEnabled(FALSE);
5424
5425     QL_dxpsi_phi->setEnabled(FALSE);
5426     QLE_dxpsi_phi->setEnabled(FALSE);
5427
5428     QL_y_minpsi_phi->setEnabled(FALSE);
5429     QLE_y_minpsi_phi->setEnabled(FALSE);
5430
5431     QL_y_maxpsi_phi->setEnabled(FALSE);
5432     QLE_y_maxpsi_phi->setEnabled(FALSE);
5433
5434     QL_dypsi_phi->setEnabled(FALSE);
5435     QLE_dypsi_phi->setEnabled(FALSE);
5436 }
5437 if( CB_Kurvenscheibe->isChecked() )
5438 {
5439     QL_Kurvenscheibe->setEnabled(TRUE);
5440
5441     QL_x_minKurvenscheibe->setEnabled(TRUE);
5442     QLE_x_minKurvenscheibe->setEnabled(TRUE);
5443
5444     QL_x_maxKurvenscheibe->setEnabled(TRUE);
5445     QLE_x_maxKurvenscheibe->setEnabled(TRUE);
5446
5447     QL_dxKurvenscheibe->setEnabled(TRUE);
5448     QLE_dxKurvenscheibe->setEnabled(TRUE);
5449
5450     QL_y_minKurvenscheibe->setEnabled(TRUE);
5451     QLE_y_minKurvenscheibe->setEnabled(TRUE);
5452
5453     QL_y_maxKurvenscheibe->setEnabled(TRUE);
5454     QLE_y_maxKurvenscheibe->setEnabled(TRUE);

```

```

5455
5456     QL_dyKurvenscheibe->setEnabled(TRUE);
5457     QLE_dyKurvenscheibe->setEnabled(TRUE);
5458 }
5459 if( !CB_Kurvenscheibe->isChecked() )
5460 {
5461     QL_Kurvenscheibe->setEnabled(FALSE);
5462
5463     QL_x_minKurvenscheibe->setEnabled(FALSE);
5464     QLE_x_minKurvenscheibe->setEnabled(FALSE);
5465
5466     QL_x_maxKurvenscheibe->setEnabled(FALSE);
5467     QLE_x_maxKurvenscheibe->setEnabled(FALSE);
5468
5469     QL_dxKurvenscheibe->setEnabled(FALSE);
5470     QLE_dxKurvenscheibe->setEnabled(FALSE);
5471
5472     QL_y_minKurvenscheibe->setEnabled(FALSE);
5473     QLE_y_minKurvenscheibe->setEnabled(FALSE);
5474
5475     QL_y_maxKurvenscheibe->setEnabled(FALSE);
5476     QLE_y_maxKurvenscheibe->setEnabled(FALSE);
5477
5478     QL_dyKurvenscheibe->setEnabled(FALSE);
5479     QLE_dyKurvenscheibe->setEnabled(FALSE);
5480 }
5481 if( CB_x_Bik->isChecked() )
5482 {
5483     QL_x_Bik->setEnabled(TRUE);
5484
5485     QL_x_minx_Bik->setEnabled(TRUE);
5486     QLE_x_minx_Bik->setEnabled(TRUE);
5487
5488     QL_x_maxx_Bik->setEnabled(TRUE);
5489     QLE_x_maxx_Bik->setEnabled(TRUE);
5490
5491     QL_dxx_Bik->setEnabled(TRUE);
5492     QLE_dxx_Bik->setEnabled(TRUE);
5493
5494     QL_y_minx_Bik->setEnabled(TRUE);
5495     QLE_y_minx_Bik->setEnabled(TRUE);
5496
5497     QL_y_maxx_Bik->setEnabled(TRUE);
5498     QLE_y_maxx_Bik->setEnabled(TRUE);

```

```

5499
5500     QL_dyx_Bik->setEnabled(TRUE);
5501     QLE_dyx_Bik->setEnabled(TRUE);
5502 }
5503 if( !CB_x_Bik->isChecked() )
5504 {
5505     QL_x_Bik->setEnabled(FALSE);
5506
5507     QL_x_minx_Bik->setEnabled(FALSE);
5508     QLE_x_minx_Bik->setEnabled(FALSE);
5509
5510     QL_x_maxx_Bik->setEnabled(FALSE);
5511     QLE_x_maxx_Bik->setEnabled(FALSE);
5512
5513     QL_dxx_Bik->setEnabled(FALSE);
5514     QLE_dxx_Bik->setEnabled(FALSE);
5515
5516     QL_y_minx_Bik->setEnabled(FALSE);
5517     QLE_y_minx_Bik->setEnabled(FALSE);
5518
5519     QL_y_maxx_Bik->setEnabled(FALSE);
5520     QLE_y_maxx_Bik->setEnabled(FALSE);
5521
5522     QL_dyx_Bik->setEnabled(FALSE);
5523     QLE_dyx_Bik->setEnabled(FALSE);
5524 }
5525 if( CB_y_Bik->isChecked() )
5526 {
5527     QL_y_Bik->setEnabled(TRUE);
5528
5529     QL_x_miny_Bik->setEnabled(TRUE);
5530     QLE_x_miny_Bik->setEnabled(TRUE);
5531
5532     QL_x_maxy_Bik->setEnabled(TRUE);
5533     QLE_x_maxy_Bik->setEnabled(TRUE);
5534
5535     QL_dxy_Bik->setEnabled(TRUE);
5536     QLE_dxy_Bik->setEnabled(TRUE);
5537
5538     QL_y_miny_Bik->setEnabled(TRUE);
5539     QLE_y_miny_Bik->setEnabled(TRUE);
5540
5541     QL_y_maxy_Bik->setEnabled(TRUE);
5542     QLE_y_maxy_Bik->setEnabled(TRUE);

```

```

5543
5544     QL_dyy_Bik->setEnabled(TRUE);
5545     QLE_dyy_Bik->setEnabled(TRUE);
5546 }
5547 if( !CB_y_Bik->isChecked() )
5548 {
5549     QL_y_Bik->setEnabled(FALSE);
5550
5551     QL_x_miny_Bik->setEnabled(FALSE);
5552     QLE_x_miny_Bik->setEnabled(FALSE);
5553
5554     QL_x_maxy_Bik->setEnabled(FALSE);
5555     QLE_x_maxy_Bik->setEnabled(FALSE);
5556
5557     QL_dxy_Bik->setEnabled(FALSE);
5558     QLE_dxy_Bik->setEnabled(FALSE);
5559
5560     QL_y_miny_Bik->setEnabled(FALSE);
5561     QLE_y_miny_Bik->setEnabled(FALSE);
5562
5563     QL_y_maxy_Bik->setEnabled(FALSE);
5564     QLE_y_maxy_Bik->setEnabled(FALSE);
5565
5566     QL_dyy_Bik->setEnabled(FALSE);
5567     QLE_dyy_Bik->setEnabled(FALSE);
5568 }
5569 if( CB_Uebertragungswinkel->isChecked() )
5570 {
5571     QL_Uebertragungswinkel->setEnabled(TRUE);
5572
5573     QL_x_minUebertragungswinkel->setEnabled(TRUE);
5574     QLE_x_minUebertragungswinkel->setEnabled(TRUE);
5575
5576     QL_x_maxUebertragungswinkel->setEnabled(TRUE);
5577     QLE_x_maxUebertragungswinkel->setEnabled(TRUE);
5578
5579     QL_dxUebertragungswinkel->setEnabled(TRUE);
5580     QLE_dxUebertragungswinkel->setEnabled(TRUE);
5581
5582     QL_y_minUebertragungswinkel->setEnabled(TRUE);
5583     QLE_y_minUebertragungswinkel->setEnabled(TRUE);
5584
5585     QL_y_maxUebertragungswinkel->setEnabled(TRUE);
5586     QLE_y_maxUebertragungswinkel->setEnabled(TRUE);

```



```

5587
5588     QL_dyUebertragungswinkel->setEnabled(TRUE);
5589     QLE_dyUebertragungswinkel->setEnabled(TRUE);
5590 }
5591 if( !CB_Uebertragungswinkel->isChecked() )
5592 {
5593     QL_Uebertragungswinkel->setEnabled(FALSE);
5594
5595     QL_x_minUebertragungswinkel->setEnabled(FALSE);
5596     QLE_x_minUebertragungswinkel->setEnabled(FALSE);
5597
5598     QL_x_maxUebertragungswinkel->setEnabled(FALSE);
5599     QLE_x_maxUebertragungswinkel->setEnabled(FALSE);
5600
5601     QL_dxUebertragungswinkel->setEnabled(FALSE);
5602     QLE_dxUebertragungswinkel->setEnabled(FALSE);
5603
5604     QL_y_minUebertragungswinkel->setEnabled(FALSE);
5605     QLE_y_minUebertragungswinkel->setEnabled(FALSE);
5606
5607     QL_y_maxUebertragungswinkel->setEnabled(FALSE);
5608     QLE_y_maxUebertragungswinkel->setEnabled(FALSE);
5609
5610     QL_dyUebertragungswinkel->setEnabled(FALSE);
5611     QLE_dyUebertragungswinkel->setEnabled(FALSE);
5612 }
5613 if( CB_Kruemmungsradius->isChecked() )
5614 {
5615     QL_Kruemmungsradius->setEnabled(TRUE);
5616
5617     QL_x_minKruemmungsradius->setEnabled(TRUE);
5618     QLE_x_minKruemmungsradius->setEnabled(TRUE);
5619
5620     QL_x_maxKruemmungsradius->setEnabled(TRUE);
5621     QLE_x_maxKruemmungsradius->setEnabled(TRUE);
5622
5623     QL_dxKruemmungsradius->setEnabled(TRUE);
5624     QLE_dxKruemmungsradius->setEnabled(TRUE);
5625
5626     QL_y_minKruemmungsradius->setEnabled(TRUE);
5627     QLE_y_minKruemmungsradius->setEnabled(TRUE);
5628
5629     QL_y_maxKruemmungsradius->setEnabled(TRUE);
5630     QLE_y_maxKruemmungsradius->setEnabled(TRUE);

```

```

5631
5632     QL_dyKruemmungsradius->setEnabled(TRUE);
5633     QLE_dyKruemmungsradius->setEnabled(TRUE);
5634 }
5635 if( !CB_Kruemmungsradius->isChecked() )
5636 {
5637     QL_Kruemmungsradius->setEnabled(FALSE);
5638
5639     QL_x_minKruemmungsradius->setEnabled(FALSE);
5640     QLE_x_minKruemmungsradius->setEnabled(FALSE);
5641
5642     QL_x_maxKruemmungsradius->setEnabled(FALSE);
5643     QLE_x_maxKruemmungsradius->setEnabled(FALSE);
5644
5645     QL_dxKruemmungsradius->setEnabled(FALSE);
5646     QLE_dxKruemmungsradius->setEnabled(FALSE);
5647
5648     QL_y_minKruemmungsradius->setEnabled(FALSE);
5649     QLE_y_minKruemmungsradius->setEnabled(FALSE);
5650
5651     QL_y_maxKruemmungsradius->setEnabled(FALSE);
5652     QLE_y_maxKruemmungsradius->setEnabled(FALSE);
5653
5654     QL_dyKruemmungsradius->setEnabled(FALSE);
5655     QLE_dyKruemmungsradius->setEnabled(FALSE);
5656 }
5657 if(indexKurve==0)
5658 {
5659     QL_Mittelpunktskurven->setEnabled(FALSE);
5660
5661     QL_x_minMittelpunktskurven->setEnabled(FALSE);
5662     QLE_x_minMittelpunktskurven->setEnabled(FALSE);
5663
5664     QL_x_maxMittelpunktskurven->setEnabled(FALSE);
5665     QLE_x_maxMittelpunktskurven->setEnabled(FALSE);
5666
5667     QL_dxMittelpunktskurven->setEnabled(FALSE);
5668     QLE_dxMittelpunktskurven->setEnabled(FALSE);
5669
5670     QL_y_minMittelpunktskurven->setEnabled(FALSE);
5671     QLE_y_minMittelpunktskurven->setEnabled(FALSE);
5672
5673     QL_y_maxMittelpunktskurven->setEnabled(FALSE);
5674     QLE_y_maxMittelpunktskurven->setEnabled(FALSE);

```

```

5675
5676     QL_dyMittelpunktskurven->setEnabled(FALSE);
5677     QLE_dyMittelpunktskurven->setEnabled(FALSE);
5678 }
5679 else if(indexKurve==1)
5680 {
5681     if( CB_Mittelpunktskurven->isChecked() )
5682     {
5683         QL_Mittelpunktskurven->setEnabled(TRUE);
5684
5685         QL_x_minMittelpunktskurven->setEnabled(TRUE);
5686         QLE_x_minMittelpunktskurven->setEnabled(TRUE);
5687
5688         QL_x_maxMittelpunktskurven->setEnabled(TRUE);
5689         QLE_x_maxMittelpunktskurven->setEnabled(TRUE);
5690
5691         QL_dxMittelpunktskurven->setEnabled(TRUE);
5692         QLE_dxMittelpunktskurven->setEnabled(TRUE);
5693
5694         QL_y_minMittelpunktskurven->setEnabled(TRUE);
5695         QLE_y_minMittelpunktskurven->setEnabled(TRUE);
5696
5697         QL_y_maxMittelpunktskurven->setEnabled(TRUE);
5698         QLE_y_maxMittelpunktskurven->setEnabled(TRUE);
5699
5700         QL_dyMittelpunktskurven->setEnabled(TRUE);
5701         QLE_dyMittelpunktskurven->setEnabled(TRUE);
5702     }
5703     if( !CB_Mittelpunktskurven->isChecked() )
5704     {
5705         QL_Mittelpunktskurven->setEnabled(FALSE);
5706
5707         QL_x_minMittelpunktskurven->setEnabled(FALSE);
5708         QLE_x_minMittelpunktskurven->setEnabled(FALSE);
5709
5710         QL_x_maxMittelpunktskurven->setEnabled(FALSE);
5711         QLE_x_maxMittelpunktskurven->setEnabled(FALSE);
5712
5713         QL_dxMittelpunktskurven->setEnabled(FALSE);
5714         QLE_dxMittelpunktskurven->setEnabled(FALSE);
5715
5716         QL_y_minMittelpunktskurven->setEnabled(FALSE);
5717         QLE_y_minMittelpunktskurven->setEnabled(FALSE);
5718

```

```

5719         QL_y_maxMittelpunktskurven->setEnabled(FALSE);
5720         QLE_y_maxMittelpunktskurven->setEnabled(FALSE);
5721
5722         QL_dyMittelpunktskurven->setEnabled(FALSE);
5723         QLE_dyMittelpunktskurven->setEnabled(FALSE);
5724     }
5725 }
5726 }
5727
5728
5729 void Opticurv::setupEinst1()
5730 {
5731     QColor farbe( 0, 255, 255 );
5732     QWidget* page1 = new QWidget( this );
5733
5734     einstellungendialog->addTab( page1, "Allgemeines" );
5735
5736     QButtonGroup *BG_Voreinstellungen = new QButtonGroup( 1, Qt::Horizontal,
5737         "", page1 );
5738     BG_Voreinstellungen->setGeometry(10,10,410,30);
5739     BG_Voreinstellungen->setFrameShape( QFrame::NoFrame );
5740
5741     CB_Dateiname = new QCheckBox(
5742         "Vor dem Berechnungsstart immer nach Dateiname fragen",
5743         BG_Voreinstellungen );
5744
5745     if(dDateinameAbfragen == 1)
5746     {
5747         CB_Dateiname->setChecked( true );
5748     }
5749     else if(dDateinameAbfragen == 2)
5750     {
5751         CB_Dateiname->setChecked( false );
5752     }
5753     else
5754     {
5755         CB_Dateiname->setChecked( true );
5756         dDateinameAbfragen = 1;
5757     }
5758
5759     QL_PlotSkalierFaktor = new QLabel(
5760         "Skalierfaktor der zu druckenden Plotfunktion",page1 );
5761     QL_PlotSkalierFaktor->setGeometry( 20, 70, 250, 25);
5762     QToolTip::add( QL_PlotSkalierFaktor,

```

```

5763     "Skalierfaktor der zu druckenden Plotfunktion" );
5764     QLE_PlotSkalierFaktor = new QLineEdit( page1 );
5765     QLE_PlotSkalierFaktor->setGeometry( 320, 70, 70, 25);
5766     QString QS_PlotSkalierFaktor=QLE_PlotSkalierFaktor->text();
5767     QS_PlotSkalierFaktor.setNum( dPlotSkalierFaktor );
5768     QLE_PlotSkalierFaktor->setText( QS_PlotSkalierFaktor );
5769     QLE_PlotSkalierFaktor->setPalette( QPalette( farbe, farbe ) );
5770
5771     QObject::connect(CB_Dateiname,SIGNAL( clicked () ),
5772                     this, SLOT( speicherePreferences() ) );
5773
5774     QObject::connect(QLE_PlotSkalierFaktor,
5775                     SIGNAL( textChanged( const QString & ) ),
5776                     this, SLOT( neu_dPlotSkalierFaktor( const QString & ) ) );
5777 }
5778
5779 void Opticurv::sichereAbfrage()
5780 {
5781     if ( ja1->isChecked() )scheibendreh_richtung = 'p';
5782     if ( nein1->isChecked() )scheibendreh_richtung = 'n';
5783     if ( ja2->isChecked() )s_richtung = 'y';
5784     if ( nein2->isChecked() )s_richtung = 'n';
5785     if ( ja3->isChecked() )Ba_Qudrant = 'y';
5786     if ( nein3->isChecked() )Ba_Qudrant = 'n';
5787     if ( ja4->isChecked() )positiv = 'y';
5788     if ( nein4->isChecked() )positiv = 'n';
5789
5790     if( CB_Arbeitskurven->isChecked() ) dmanuelleArbeitskurven=1;
5791     else if( !CB_Arbeitskurven->isChecked() ) dmanuelleArbeitskurven=2;
5792
5793     if( CB_s_phi->isChecked() ) dmanuelles_phi=1;
5794     else if( !CB_s_phi->isChecked() ) dmanuelles_phi=2;
5795
5796     if( CB_psi_phi->isChecked() ) dmanuellepsi_phi=1;
5797     else if( !CB_psi_phi->isChecked() ) dmanuellepsi_phi=2;
5798
5799     if( CB_Kurvenscheibe->isChecked() ) dmanuelleKurvenscheibe=1;
5800     else if( !CB_Kurvenscheibe->isChecked() ) dmanuelleKurvenscheibe=2;
5801
5802     if( CB_x_Bik->isChecked() ) dmanuellex_Bik=1;
5803     else if( !CB_x_Bik->isChecked() ) dmanuellex_Bik=2;
5804
5805     if( CB_y_Bik->isChecked() ) dmanuelley_Bik=1;
5806     else if( !CB_y_Bik->isChecked() ) dmanuelley_Bik=2;

```

```

5807
5808     if( CB_Uebertragungswinkel->isChecked() ) dmanuelleUebertragungswinkel=1;
5809     else if( !CB_Uebertragungswinkel->isChecked() ) dmanuelleUebertragungswinkel=2;
5810
5811     if( CB_Kruemmungsradius->isChecked() ) dmanuelleKruemmungsradius=1;
5812     else if( !CB_Kruemmungsradius->isChecked() ) dmanuelleKruemmungsradius=2;
5813
5814     if( CB_Mittelpunktskurven->isChecked() ) dmanuelleMittelpunktskurven=1;
5815     else if( !CB_Mittelpunktskurven->isChecked() ) dmanuelleMittelpunktskurven=2;
5816
5817     if(scheibendreh_richtung == 'p')
5818     {
5819         dri_scheibe = 1;
5820     }
5821     else if(scheibendreh_richtung == 'n')
5822     {
5823         dri_scheibe = 2;
5824     }
5825     if(s_richtung == 'y')
5826     {
5827         dhub_gleich = 1;
5828     }
5829     else if(s_richtung == 'n')
5830     {
5831         dhub_gleich = 2;
5832     }
5833     if(Ba_Qudrant == 'y')
5834     {
5835         dquadrant = 1;
5836     }
5837     else if(Ba_Qudrant == 'n')
5838     {
5839         dquadrant = 2;
5840     }
5841     if(positiv == 'y')
5842     {
5843         dps_i_positiv = 1;
5844     }
5845     else if(positiv == 'n')
5846     {
5847         dps_i_positiv = 2;
5848     }
5849 }
5850

```

```

5851 void Opticurv::saveAs()
5852 {
5853     void setFilter(const QString& ocv);
5854
5855     filename = QFileDialog::getSaveFileName( QString::null, "*.ocv", this );
5856     if( !filename.isEmpty() )
5857     {
5858         filename_old = filename;
5859         filename_old.remove( filename_old.length()-4,4 );
5860         filename_asc = filename_old + ".asc"; // Endung .asc an
5861                                           // filename_asc anhängen
5862         filename_innen = filename_old + "_innen.asc";
5863         filename_mitte = filename_old + "_mitte.asc";
5864         filename_aussen = filename_old + "_aussen.asc";
5865         filename_s_phi  = filename_old + "_s_phi .asc";
5866     }
5867 }
5868 void Opticurv::setup_ergAnz()
5869 {
5870     if(anzahlBewabschnitte->currentItem() == 0)
5871     {
5872         dergebnis = dphi_H01;
5873     }
5874     else if(anzahlBewabschnitte->currentItem() == 1)
5875     {
5876         dergebnis = dphi_H01 + dphi_H12;
5877     }
5878     else if(anzahlBewabschnitte->currentItem() == 2)
5879     {
5880         dergebnis = dphi_H01 + dphi_H12 + dphi_H23;
5881     }
5882     else if(anzahlBewabschnitte->currentItem() == 3)
5883     {
5884         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34;
5885     }
5886     else if(anzahlBewabschnitte->currentItem() == 3)
5887     {
5888         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34;
5889     }
5890     else if(anzahlBewabschnitte->currentItem() == 4)
5891     {
5892         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34 + dphi_H45;
5893     }
5894     else if(anzahlBewabschnitte->currentItem() == 5)

```

```

5895     {
5896         dergebnis = dphi_H01 + dphi_H12 + dphi_H23 + dphi_H34 + dphi_H45
5897             + dphi_H56;
5898     }
5899     else
5900     {
5901         dergebnis = 1;
5902     }
5903
5904
5905     QColor farbe( 0, 255, 255 );
5906     ergebnisAnzeige = new QLabel( tabdialog );
5907     ergebnisAnzeige->setGeometry( 360, 415, 70, 30);
5908     ergebnisAnzeige->setNum( dergebnis );
5909     ergebnisAnzeige->setAlignment( QLabel::AlignCenter );
5910     ergebnisAnzeige->setPalette( QPalette( farbe, farbe ) );
5911     ergebnisAnzeige->show();
5912     ergebnisAnzeigeLabel = new QLabel( tabdialog );
5913     ergebnisAnzeigeLabel->setText(
5914         "Aktuelle Gesamtgradzahl der Gesamtdrehwinkel phi_Hik" );
5915     ergebnisAnzeigeLabel->setGeometry( 30, 415, 330, 30);
5916     tabdialog->setOkButton( "&Ausfuehren" );
5917     tabdialog->setCancelButton( "A&bbrechen" );
5918
5919     connect( tabdialog, SIGNAL( applyButtonPressed() ), this,
5920             SLOT( speichere() ) );
5921 }
5922
5923 void Opticurv::berechneHub1()
5924 {
5925     if( index1==0 )
5926     {
5927         /* Hub in den Bewegungsabschnitten von phi */
5928
5929         s = s_1;
5930         s_plus_deltaphi = s_1;
5931         s_minus_deltaphi = s_1;
5932
5933         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
5934
5935         s_strich = 0;
5936
5937         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
5938

```



```

5939     s_zweistrich = 0;
5940 }
5941 else if( index1==1 )
5942 {
5943     /* Hub in den Bewegungsabschnitten von phi */
5944
5945     s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
5946         (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
5947
5948     s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
5949         (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
5950         (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
5951         (pow((phi+deltaphi), 5))));
5952
5953     s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
5954         (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
5955         (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
5956         (pow((phi-deltaphi), 5))));
5957
5958     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
5959
5960     s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
5961         (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
5962         (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4))));
5963
5964     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
5965
5966     s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
5967         (pow((phi),2)))+(120/(pow(phi_H01,2)))*
5968         (pow((phi),3))));
5969 }
5970 else if( index1==2 )
5971 {
5972     /* Bestehornsinoide */
5973
5974     s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
5975
5976     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
5977         sin(2*pi*((phi+deltaphi)/phi_H01))));
5978
5979     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
5980         sin(2*pi*((phi-deltaphi)/phi_H01))));
5981
5982     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));

```

```

5983
5984     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
5985 }
5986 }
5987
5988 void Opticurv::berechneHub2()
5989 {
5990     /* erster Abschnitt */
5991     if( phi>=0 && phi<phi_1 )
5992     {
5993         if( index1==0 )
5994         {
5995             /* Hub in den Bewegungsabschnitten von phi */
5996
5997             s = s_1;
5998             s_plus_deltaphi = s_1;
5999             s_minus_deltaphi = s_1;
6000
6001             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6002
6003             s_strich = 0;
6004
6005             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6006
6007             s_zweistrich = 0;
6008         }
6009         else if( index1==1 )
6010         {
6011             /* Hub in den Bewegungsabschnitten von phi */
6012
6013             s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
6014                 (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
6015
6016             s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6017                 (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
6018                 (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6019                 (pow((phi+deltaphi), 5))));
6020
6021             s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6022                 (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
6023                 (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6024                 (pow((phi-deltaphi), 5))));
6025
6026             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */

```

```

6027
6028     s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
6029             (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
6030             (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4))));
6031
6032     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6033
6034     s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
6035             (pow((phi),2)))+(120/(pow(phi_H01,2))*
6036             (pow((phi),3)))));
6037 }
6038 else if( index1==2 )
6039 {
6040     /* Bestehornsinoide */
6041
6042     s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
6043
6044     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
6045             sin(2*pi*((phi+deltaphi)/phi_H01))));
6046
6047     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
6048             sin(2*pi*((phi-deltaphi)/phi_H01))));
6049
6050     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
6051
6052     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
6053 }
6054 }
6055 /* zweiter Abschnitt */
6056 else
6057 {
6058     if( index2==0 )
6059     {
6060         /* Hub in den Bewegungsabschnitten von phi */
6061
6062         s = s_2;
6063         s_plus_deltaphi = s_2;
6064         s_minus_deltaphi = s_2;
6065
6066         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6067
6068         s_strich = 0;
6069
6070         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */

```

```

6071
6072     s_zweistrich = 0;
6073 }
6074 else if( index2==1 )
6075 {
6076     /* Hub in den Bewegungsabschnitten von phi */
6077
6078     s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
6079         (15/phi_H12)*(pow(phi-phi_1, 4))+6/(pow(phi_H12, 2)))*
6080         (pow(phi-phi_1, 5)))));
6081
6082     s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6083         (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
6084         (pow((phi-phi_1+deltaphi), 4))+6/(pow(phi_H12, 2)))*
6085         (pow((phi-phi_1+deltaphi), 5)))));
6086
6087     s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6088         (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
6089         (pow((phi-phi_1-deltaphi), 4))+6/(pow(phi_H12, 2)))*
6090         (pow((phi-phi_1-deltaphi), 5)))));
6091
6092     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6093
6094     s_strich=((s_H12/(pow((phi_H12*rad),3)))*(30*(pow(((phi-phi_1)*rad),2))-
6095         (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad),3))+
6096         (30/(pow((phi_H12*rad),2)))*(pow(((phi-phi_1)*rad),4))));
6097
6098     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6099
6100     s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
6101         (pow((phi-phi_1),2)))+((120/(pow(phi_H12,2)))*
6102         (pow((phi-phi_1),3))));
6103 }
6104 else if( index2==2 )
6105 {
6106     /* Bestehornsinoide */
6107
6108     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
6109         ((phi-phi_1)/phi_H12))));
6110
6111     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
6112         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
6113         phi_H12))));
6114

```

```

6115         s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
6116             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
6117                 phi_H12))));
6118
6119         s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
6120
6121         s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
6122     }
6123 }
6124 }
6125
6126 void Opticurv::berechneHub3()
6127 {
6128     /* erster Abschnitt */
6129     if( phi>=0 && phi<phi_1 )
6130     {
6131         if( index1==0 )
6132         {
6133             /* Hub in den Bewegungsabschnitten von phi */
6134
6135             s = s_1;
6136             s_plus_deltaphi = s_1;
6137             s_minus_deltaphi = s_1;
6138
6139             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6140
6141             s_strich = 0;
6142
6143             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6144
6145             s_zweistrich = 0;
6146         }
6147     else if( index1==1 )
6148     {
6149         /* Hub in den Bewegungsabschnitten von phi */
6150
6151         s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
6152             (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
6153
6154         s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6155             (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
6156             (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6157             (pow((phi+deltaphi), 5))));
6158

```

```

6159     s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6160         (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
6161         (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6162         (pow((phi-deltaphi), 5))));
6163
6164     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6165
6166     s_strich=((s_H01/(pow((phi_H01*rad), 3)))*(30*(pow((phi)*rad), 2))-
6167         (60/(phi_H01*rad))*(pow((phi)*rad), 3))+
6168         (30/(pow((phi_H01*rad), 2)))*(pow((phi)*rad), 4)));
6169
6170     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6171
6172     s_zweistrich=((s_H01/(pow(phi_H01, 3)))*(60*(phi)-((180/phi_H01)*
6173         (pow((phi), 2)))+(120/(pow(phi_H01, 2)))*
6174         (pow((phi), 3))));
6175 }
6176 else if( index1==2 )
6177 {
6178     /* Bestehornsinoide */
6179
6180     s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
6181
6182     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
6183         sin(2*pi*((phi+deltaphi)/phi_H01))));
6184
6185     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
6186         sin(2*pi*((phi-deltaphi)/phi_H01))));
6187
6188     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
6189
6190     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
6191 }
6192 }
6193 /* zweiter Abschnitt */
6194 else if(phi>= phi_1 && phi<phi_2)
6195 {
6196     if( index2==0 )
6197     {
6198         /* Hub in den Bewegungsabschnitten von phi */
6199
6200         s = s_2;
6201         s_plus_deltaphi = s_2;
6202         s_minus_deltaphi = s_2;

```

```

6203
6204     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6205
6206     s_strich = 0;
6207
6208     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6209
6210     s_zweistrich = 0;
6211 }
6212 else if( index2==1 )
6213 {
6214     /* Hub in den Bewegungsabschnitten von phi */
6215
6216     s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
6217         (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
6218         (pow(phi-phi_1, 5))))));
6219
6220     s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6221         (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
6222         (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
6223         (pow((phi-phi_1+deltaphi), 5))))));
6224
6225     s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6226         (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
6227         (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
6228         (pow((phi-phi_1-deltaphi), 5))))));
6229
6230     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6231
6232     s_strich=((s_H12/(pow((phi_H12*rad), 3)))*(30*(pow(((phi-phi_1)*rad), 2))-
6233         (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad), 3))+
6234         (30/(pow((phi_H12*rad), 2)))*(pow(((phi-phi_1)*rad), 4))));
6235
6236     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6237
6238     s_zweistrich=((s_H12/(pow(phi_H12, 3)))*(60*(phi-phi_1)-((180/phi_H12)*
6239         (pow((phi-phi_1), 2)))+((120/(pow(phi_H12, 2)))*
6240         (pow((phi-phi_1), 3))));
6241 }
6242 else if( index2==2 )
6243 {
6244     /* Bestehornsinoide */
6245
6246     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*

```

```

6247         ((phi-phi_1)/phi_H12)))));
6248
6249     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
6250         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
6251             phi_H12))));
6252
6253     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
6254         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
6255             phi_H12))));
6256
6257     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
6258
6259     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
6260 }
6261 }
6262 /* dritter Abschnitt */
6263 else
6264 {
6265     if( index3==0 )
6266     {
6267         /* Hub in den Bewegungsabschnitten von phi */
6268
6269         s = s_3;
6270         s_plus_deltaphi = s_3;
6271         s_minus_deltaphi = s_3;
6272
6273         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6274
6275         s_strich = 0;
6276
6277         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6278
6279         s_zweistrich = 0;
6280     }
6281     else if( index3==1 )
6282     {
6283         /* Hub in den Bewegungsabschnitten von phi */
6284
6285         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
6286             (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
6287             (pow(phi-phi_2, 5))))));
6288
6289         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
6290             (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*

```



```

6291         (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
6292         (pow((phi-phi_2+deltaphi), 5))));
6293
6294     s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
6295         (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
6296         (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
6297         (pow((phi-phi_2-deltaphi), 5))));
6298
6299     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6300
6301     s_strich=((s_H23/(pow((phi_H23*rad), 3)))*(30*(pow(((phi-phi_2)*rad), 2))-
6302         (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad), 3))+
6303         (30/(pow((phi_H23*rad), 2)))*(pow(((phi-phi_2)*rad), 4))));
6304
6305     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6306
6307     s_zweistrich=((s_H23/(pow(phi_H23, 3)))*(60*(phi-phi_2)-((180/phi_H23)*
6308         (pow((phi-phi_2), 2)))+(120/(pow(phi_H23, 2)))*
6309         (pow((phi-phi_2), 3))));
6310 }
6311 else if( index3==2 )
6312 {
6313     /* Bestehornsinoide */
6314
6315     s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
6316         ((phi-phi_2)/phi_H23))));
6317
6318     s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
6319         (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
6320         phi_H23))));
6321
6322     s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
6323         (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
6324         phi_H23))));
6325
6326     s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
6327
6328     s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
6329 }
6330 }
6331 }
6332
6333 void Opticurv::berechneHub4()
6334 {

```

```

6335      /* erster Abschnitt */
6336      if( phi>=0 && phi<phi_1 )
6337      {
6338          if( index1==0 )
6339          {
6340              /* Hub in den Bewegungsabschnitten von phi */
6341
6342              s = s_1;
6343              s_plus_deltaphi = s_1;
6344              s_minus_deltaphi = s_1;
6345
6346              /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6347
6348              s_strich = 0;
6349
6350              /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6351
6352              s_zweistrich = 0;
6353          }
6354          else if( index1==1 )
6355          {
6356              /* Hub in den Bewegungsabschnitten von phi */
6357
6358              s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
6359                  (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
6360
6361              s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6362                  (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
6363                  (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6364                  (pow((phi+deltaphi), 5))));
6365
6366              s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6367                  (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
6368                  (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6369                  (pow((phi-deltaphi), 5))));
6370
6371              /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6372
6373              s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
6374                  (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
6375                  (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4))));
6376
6377              /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6378

```

```

6379         s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
6380                     (pow((phi),2)))+((120/(pow(phi_H01,2)))*
6381                     (pow((phi),3)))));
6382     }
6383     else if( index1==2 )
6384     {
6385         /* Bestehornsinoide */
6386
6387         s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
6388
6389         s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
6390                     sin(2*pi*((phi+deltaphi)/phi_H01))));
6391
6392         s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
6393                     sin(2*pi*((phi-deltaphi)/phi_H01))));
6394
6395         s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
6396
6397         s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
6398     }
6399 }
6400 /* zweiter Abschnitt */
6401 else if(phi>= phi_1 && phi<phi_2)
6402 {
6403     if( index2==0 )
6404     {
6405         /* Hub in den Bewegungsabschnitten von phi */
6406
6407         s = s_2;
6408         s_plus_deltaphi = s_2;
6409         s_minus_deltaphi = s_2;
6410
6411         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6412
6413         s_strich = 0;
6414
6415         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6416
6417         s_zweistrich = 0;
6418     }
6419     else if( index2==1 )
6420     {
6421         /* Hub in den Bewegungsabschnitten von phi */
6422

```

```

6423     s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
6424         (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
6425         (pow(phi-phi_1, 5)))));
6426
6427     s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6428         (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
6429         (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
6430         (pow((phi-phi_1+deltaphi), 5)))));
6431
6432     s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6433         (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
6434         (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
6435         (pow((phi-phi_1-deltaphi), 5)))));
6436
6437     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6438
6439     s_strich=((s_H12/(pow((phi_H12*rad),3)))*(30*(pow(((phi-phi_1)*rad),2))-
6440         (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad),3))+
6441         (30/(pow((phi_H12*rad),2))*(pow(((phi-phi_1)*rad),4)))));
6442
6443     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6444
6445     s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
6446         (pow((phi-phi_1),2)))+(120/(pow(phi_H12,2)))*
6447         (pow((phi-phi_1),3))));
6448 }
6449 else if( index2==2 )
6450 {
6451     /* Bestehornsinoide */
6452
6453     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
6454         ((phi-phi_1)/phi_H12))));
6455
6456     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
6457         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
6458         phi_H12))));
6459
6460     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
6461         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
6462         phi_H12))));
6463
6464     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
6465
6466     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));

```

```

6467     }
6468 }
6469 /* dritter Abschnitt */
6470 else if(phi>=phi_2 && phi<phi_3)
6471 {
6472     if( index3==0 )
6473     {
6474         /* Hub in den Bewegungsabschnitten von phi */
6475
6476         s = s_3;
6477         s_plus_deltaphi = s_3;
6478         s_minus_deltaphi = s_3;
6479
6480         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6481
6482         s_strich = 0;
6483
6484         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6485
6486         s_zweistrich = 0;
6487     }
6488     else if( index3==1 )
6489     {
6490         /* Hub in den Bewegungsabschnitten von phi */
6491
6492         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
6493             (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
6494             (pow(phi-phi_2, 5))))));
6495
6496         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
6497             (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
6498             (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
6499             (pow((phi-phi_2+deltaphi), 5))))));
6500
6501         s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
6502             (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
6503             (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
6504             (pow((phi-phi_2-deltaphi), 5))))));
6505
6506         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6507
6508         s_strich=((s_H23/(pow((phi_H23*rad),3)))*(30*(pow(((phi-phi_2)*rad),2))-
6509             (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad),3))+
6510             (30/(pow((phi_H23*rad),2)))*(pow(((phi-phi_2)*rad),4))));

```

```

6511
6512      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6513
6514      s_zweistrich=((s_H23/(pow(phi_H23,3)))*(60*(phi-phi_2)-((180/phi_H23)*
6515          (pow((phi-phi_2),2)))+((120/(pow(phi_H23,2)))*
6516          (pow((phi-phi_2),3)))));
6517  }
6518  else if( index3==2 )
6519  {
6520      /* Bestehornsinoide */
6521
6522      s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
6523          ((phi-phi_2)/phi_H23))));
6524
6525      s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
6526          (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
6527          phi_H23))));
6528
6529      s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
6530          (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
6531          phi_H23))));
6532
6533      s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
6534
6535      s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
6536  }
6537 }
6538 /* vierter Abschnitt */
6539 else
6540 {
6541     if( index4==0 )
6542     {
6543         /* Hub in den Bewegungsabschnitten von phi */
6544
6545         s = s_4;
6546         s_plus_deltaphi = s_4;
6547         s_minus_deltaphi = s_4;
6548
6549         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6550
6551         s_strich = 0;
6552
6553         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6554

```

```

6555     s_zweistrich = 0;
6556 }
6557 else if( index4==1 )
6558 {
6559     /* Hub in den Bewegungsabschnitten von phi */
6560
6561     s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
6562         (15/phi_H34)*(pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*
6563         (pow(phi-phi_3, 5))))));
6564
6565     s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
6566         (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
6567         (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
6568         (pow((phi-phi_3+deltaphi), 5))))));
6569
6570     s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
6571         (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
6572         (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
6573         (pow((phi-phi_3-deltaphi), 5))))));
6574
6575     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6576
6577     s_strich=((s_H34/(pow((phi_H34*rad), 3)))*(30*(pow(((phi-phi_3)*rad), 2))-
6578         (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad), 3))+
6579         (30/(pow((phi_H34*rad), 2)))*(pow(((phi-phi_3)*rad), 4))));
6580
6581     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6582
6583     s_zweistrich=((s_H34/(pow(phi_H34, 3)))*(60*(phi-phi_3)-((180/phi_H34)*
6584         (pow((phi-phi_3), 2)))+((120/(pow(phi_H34, 2)))*
6585         (pow((phi-phi_3), 3))));
6586 }
6587 else if( index4==2 )
6588 {
6589     /* Bestehornsinoide */
6590
6591     s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
6592         ((phi-phi_3)/phi_H34))));
6593
6594     s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
6595         (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
6596         phi_H34))));
6597
6598     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-

```

```

6599             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
6600             phi_H34))));
6601
6602     s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
6603
6604     s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
6605 }
6606 }
6607 }
6608
6609 void Opticurv::berechneHub5()
6610 {
6611     /* erster Abschnitt */
6612     if( phi>=0 && phi<phi_1 )
6613     {
6614         if( index1==0 )
6615         {
6616             /* Hub in den Bewegungsabschnitten von phi */
6617
6618             s = s_1;
6619             s_plus_deltaphi = s_1;
6620             s_minus_deltaphi = s_1;
6621
6622             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6623
6624             s_strich = 0;
6625
6626             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6627
6628             s_zweistrich = 0;
6629         }
6630     else if( index1==1 )
6631     {
6632         /* Hub in den Bewegungsabschnitten von phi */
6633
6634         s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
6635             (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
6636
6637         s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6638             (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
6639             (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6640             (pow((phi+deltaphi), 5))));
6641
6642         s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*

```



```

6643         (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
6644         (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6645         (pow((phi-deltaphi), 5))));
6646
6647     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6648
6649     s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
6650             (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
6651             (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4))));
6652
6653     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6654
6655     s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
6656             (pow((phi),2)))+(120/(pow(phi_H01,2)))*
6657             (pow((phi),3))));
6658 }
6659 else if( index1==2 )
6660 {
6661     /* Bestehornsinoide */
6662
6663     s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
6664
6665     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
6666             sin(2*pi*((phi+deltaphi)/phi_H01))));
6667
6668     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
6669             sin(2*pi*((phi-deltaphi)/phi_H01))));
6670
6671     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
6672
6673     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
6674 }
6675 }
6676 /* zweiter Abschnitt */
6677 else if(phi>= phi_1 && phi<phi_2)
6678 {
6679     if( index2==0 )
6680     {
6681         /* Hub in den Bewegungsabschnitten von phi */
6682
6683         s = s_2;
6684         s_plus_deltaphi = s_2;
6685         s_minus_deltaphi = s_2;
6686

```

```

6687      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6688
6689      s_strich = 0;
6690
6691      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6692
6693      s_zweistrich = 0;
6694  }
6695  else if( index2==1 )
6696  {
6697      /* Hub in den Bewegungsabschnitten von phi */
6698
6699      s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
6700          (15/phi_H12)*(pow(phi-phi_1, 4))+6/(pow(phi_H12, 2)))*
6701          (pow(phi-phi_1, 5)))));
6702
6703      s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6704          (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
6705          (pow((phi-phi_1+deltaphi), 4))+6/(pow(phi_H12, 2)))*
6706          (pow((phi-phi_1+deltaphi), 5)))));
6707
6708      s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
6709          (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
6710          (pow((phi-phi_1-deltaphi), 4))+6/(pow(phi_H12, 2)))*
6711          (pow((phi-phi_1-deltaphi), 5)))));
6712
6713      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6714
6715      s_strich=((s_H12/(pow((phi_H12*rad),3)))*(30*(pow(((phi-phi_1)*rad),2))-
6716          (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad),3))+
6717          (30/(pow((phi_H12*rad),2)))*(pow(((phi-phi_1)*rad),4))));
6718
6719      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6720
6721      s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
6722          (pow((phi-phi_1),2)))+(120/(pow(phi_H12,2)))*
6723          (pow((phi-phi_1),3))));
6724  }
6725  else if( index2==2 )
6726  {
6727      /* Bestehornsinoide */
6728
6729      s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
6730          ((phi-phi_1)/phi_H12))));

```

```

6731
6732     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
6733         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
6734             phi_H12))));
6735
6736     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
6737         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
6738             phi_H12))));
6739
6740     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
6741
6742     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
6743 }
6744 }
6745 /* dritter Abschnitt */
6746 else if(phi>=phi_2 && phi<phi_3)
6747 {
6748     if( index3==0 )
6749     {
6750         /* Hub in den Bewegungsabschnitten von phi */
6751
6752         s = s_3;
6753         s_plus_deltaphi = s_3;
6754         s_minus_deltaphi = s_3;
6755
6756         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6757
6758         s_strich = 0;
6759
6760         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6761
6762         s_zweistrich = 0;
6763     }
6764     else if( index3==1 )
6765     {
6766         /* Hub in den Bewegungsabschnitten von phi */
6767
6768         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
6769             (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2))))*
6770             (pow(phi-phi_2, 5)))));
6771
6772         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
6773             (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
6774             (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2))))*

```

```

6775         (pow((phi-phi_2+deltaphi), 5))));
6776
6777     s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
6778         (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
6779         (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
6780         (pow((phi-phi_2-deltaphi), 5))));
6781
6782     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6783
6784     s_strich=((s_H23/(pow((phi_H23*rad),3)))*(30*(pow(((phi-phi_2)*rad),2))-
6785         (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad),3))+
6786         (30/(pow((phi_H23*rad),2))*(pow(((phi-phi_2)*rad),4))));
6787
6788     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6789
6790     s_zweistrich=((s_H23/(pow(phi_H23,3)))*(60*(phi-phi_2)-((180/phi_H23)*
6791         (pow((phi-phi_2),2)))+((120/(pow(phi_H23,2)))*
6792         (pow((phi-phi_2),3))));
6793 }
6794 else if( index3==2 )
6795 {
6796     /* Bestehornsinoide */
6797
6798     s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
6799         ((phi-phi_2)/phi_H23))));
6800
6801     s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
6802         (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
6803         phi_H23))));
6804
6805     s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
6806         (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
6807         phi_H23))));
6808
6809     s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
6810
6811     s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
6812 }
6813 }
6814 /* vierter Abschnitt */
6815 else if(phi>=phi_3 && phi<phi_4)
6816 {
6817     if( index4==0 )
6818     {

```

```

6819      /* Hub in den Bewegungsabschnitten von phi */
6820
6821      s = s_4;
6822      s_plus_deltaphi = s_4;
6823      s_minus_deltaphi = s_4;
6824
6825      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6826
6827      s_strich = 0;
6828
6829      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6830
6831      s_zweistrich = 0;
6832  }
6833  else if( index4==1 )
6834  {
6835      /* Hub in den Bewegungsabschnitten von phi */
6836
6837      s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
6838          (15/phi_H34)*(pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*
6839          (pow(phi-phi_3, 5))))));
6840
6841      s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
6842          (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
6843          (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
6844          (pow((phi-phi_3+deltaphi), 5))))));
6845
6846      s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
6847          (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
6848          (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
6849          (pow((phi-phi_3-deltaphi), 5))))));
6850
6851      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6852
6853      s_strich=((s_H34/(pow((phi_H34*rad), 3)))*(30*(pow(((phi-phi_3)*rad), 2))-
6854          (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad), 3))+
6855          (30/(pow((phi_H34*rad), 2)))*(pow(((phi-phi_3)*rad), 4))));
6856
6857      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6858
6859      s_zweistrich=((s_H34/(pow(phi_H34, 3)))*(60*(phi-phi_3)-((180/phi_H34)*
6860          (pow((phi-phi_3), 2)))+((120/(pow(phi_H34, 2)))*
6861          (pow((phi-phi_3), 3)))));
6862  }

```

```

6863     else if( index4==2 )
6864     {
6865         /* Bestehornsinoide */
6866
6867         s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
6868             ((phi-phi_3)/phi_H34))));
6869
6870         s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
6871             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
6872                 phi_H34))));
6873
6874         s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
6875             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
6876                 phi_H34))));
6877
6878         s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
6879
6880         s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
6881     }
6882 }
6883 /* fuenfter Abschnitt */
6884 else
6885 {
6886     if( index5==0 )
6887     {
6888         /* Hub in den Bewegungsabschnitten von phi */
6889
6890         s = s_5;
6891         s_plus_deltaphi = s_5;
6892         s_minus_deltaphi = s_5;
6893
6894         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6895
6896         s_strich = 0;
6897
6898         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6899
6900         s_zweistrich = 0;
6901     }
6902     else if( index5==1 )
6903     {
6904         /* Hub in den Bewegungsabschnitten von phi */
6905
6906         s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-

```

```

6907         (15/phi_H45)*(pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2)))*
6908         (pow(phi-phi_4, 5))));
6909
6910     s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
6911         (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
6912         (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2)))*
6913         (pow((phi-phi_4+deltaphi), 5))));
6914
6915     s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
6916         (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*
6917         (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
6918         (pow((phi-phi_4-deltaphi), 5))));
6919
6920     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6921
6922     s_strich=((s_H45/(pow((phi_H45*rad), 3)))*(30*(pow(((phi-phi_4)*rad), 2))-
6923         (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad), 3))+
6924         (30/(pow((phi_H45*rad), 2)))*(pow(((phi-phi_4)*rad), 4))));
6925
6926     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6927
6928     s_zweistrich=((s_H45/(pow(phi_H45, 3)))*(60*(phi-phi_4)-((180/phi_H45)*
6929         (pow((phi-phi_4), 2)))+(120/(pow(phi_H45, 2)))*
6930         (pow((phi-phi_4), 3))));
6931 }
6932 else if( index5==2 )
6933 {
6934     /* Bestehornsinoide */
6935
6936     s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
6937         ((phi-phi_4)/phi_H45))));
6938
6939     s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-
6940         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/
6941         phi_H45))));
6942
6943     s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
6944         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
6945         phi_H45))));
6946
6947     s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
6948
6949     s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
6950 }

```

```

6951     }
6952 }
6953
6954 void Opticurv::berechneHub6()
6955 {
6956     /* erster Abschnitt */
6957     if( phi>=0 && phi<phi_1 )
6958     {
6959         if( index1==0 )
6960         {
6961             /* Hub in den Bewegungsabschnitten von phi */
6962
6963             s = s_1;
6964             s_plus_deltaphi = s_1;
6965             s_minus_deltaphi = s_1;
6966
6967             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6968
6969             s_strich = 0;
6970
6971             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6972
6973             s_zweistrich = 0;
6974         }
6975         else if( index1==1 )
6976         {
6977             /* Hub in den Bewegungsabschnitten von phi */
6978
6979             s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
6980                 (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
6981
6982             s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6983                 (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
6984                 (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6985                 (pow((phi+deltaphi), 5))));
6986
6987             s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
6988                 (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
6989                 (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
6990                 (pow((phi-deltaphi), 5))));
6991
6992             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
6993
6994             s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-

```



```

6995             (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
6996             (30/(pow((phi_H01*rad),2))*(pow(((phi)*rad),4))));
6997
6998     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
6999
7000     s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
7001             (pow((phi),2)))+((120/(pow(phi_H01,2)))*
7002             (pow((phi),3)))));
7003 }
7004 else if( index1==2 )
7005 {
7006     /* Bestehornsinoide */
7007
7008     s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
7009
7010     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
7011             sin(2*pi*((phi+deltaphi)/phi_H01))));
7012
7013     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
7014             sin(2*pi*((phi-deltaphi)/phi_H01))));
7015
7016     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
7017
7018     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
7019 }
7020 }
7021 /* zweiter Abschnitt */
7022 else if(phi>= phi_1 && phi<phi_2)
7023 {
7024     if( index2==0 )
7025     {
7026         /* Hub in den Bewegungsabschnitten von phi */
7027
7028         s = s_2;
7029         s_plus_deltaphi = s_2;
7030         s_minus_deltaphi = s_2;
7031
7032         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7033
7034         s_strich = 0;
7035
7036         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7037
7038         s_zweistrich = 0;

```

```

7039     }
7040     else if( index2==1 )
7041     {
7042         /* Hub in den Bewegungsabschnitten von phi */
7043
7044         s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
7045             (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
7046             (pow(phi-phi_1, 5)))));
7047
7048         s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
7049             (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
7050             (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
7051             (pow((phi-phi_1+deltaphi), 5)))));
7052
7053         s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
7054             (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
7055             (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
7056             (pow((phi-phi_1-deltaphi), 5)))));
7057
7058         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7059
7060         s_strich=((s_H12/(pow((phi_H12*rad),3)))*(30*(pow(((phi-phi_1)*rad),2))-
7061             (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad),3))+
7062             (30/(pow((phi_H12*rad),2)))*(pow(((phi-phi_1)*rad),4))));
7063
7064         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7065
7066         s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
7067             (pow((phi-phi_1),2)))+((120/(pow(phi_H12,2)))*
7068             (pow((phi-phi_1),3))));
7069     }
7070     else if( index2==2 )
7071     {
7072         /* Bestehornsinoide */
7073
7074         s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
7075             ((phi-phi_1)/phi_H12))));
7076
7077         s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
7078             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
7079             phi_H12))));
7080
7081         s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
7082             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/

```

```

7083             phi_H12))));
7084
7085     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
7086
7087     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
7088 }
7089 }
7090 /* dritter Abschnitt */
7091 else if(phi>=phi_2 && phi<phi_3)
7092 {
7093     if( index3==0 )
7094     {
7095         /* Hub in den Bewegungsabschnitten von phi */
7096
7097         s = s_3;
7098         s_plus_deltaphi = s_3;
7099         s_minus_deltaphi = s_3;
7100
7101         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7102
7103         s_strich = 0;
7104
7105         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7106
7107         s_zweistrich = 0;
7108     }
7109     else if( index3==1 )
7110     {
7111         /* Hub in den Bewegungsabschnitten von phi */
7112
7113         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
7114             (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
7115             (pow(phi-phi_2, 5))))));
7116
7117         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
7118             (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
7119             (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
7120             (pow((phi-phi_2+deltaphi), 5))))));
7121
7122         s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
7123             (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
7124             (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
7125             (pow((phi-phi_2-deltaphi), 5))))));
7126

```

```

7127      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7128
7129      s_strich=((s_H23/(pow((phi_H23*rad),3)))*(30*(pow(((phi-phi_2)*rad),2))-
7130              (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad),3))+
7131              (30/(pow((phi_H23*rad),2)))*(pow(((phi-phi_2)*rad),4))));
7132
7133      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7134
7135      s_zweistrich=((s_H23/(pow(phi_H23,3)))*(60*(phi-phi_2)-((180/phi_H23)*
7136              (pow((phi-phi_2),2)))+((120/(pow(phi_H23,2)))*
7137              (pow((phi-phi_2),3)))));
7138  }
7139  else if( index3==2 )
7140  {
7141      /* Bestehornsinoide */
7142
7143      s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
7144              ((phi-phi_2)/phi_H23))));
7145
7146      s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
7147              (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
7148              phi_H23))));
7149
7150      s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
7151              (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
7152              phi_H23))));
7153
7154      s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
7155
7156      s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
7157  }
7158  }
7159  /* vierter Abschnitt */
7160  else if(phi>=phi_3 && phi<phi_4)
7161  {
7162      if( index4==0 )
7163      {
7164          /* Hub in den Bewegungsabschnitten von phi */
7165
7166          s = s_4;
7167          s_plus_deltaphi = s_4;
7168          s_minus_deltaphi = s_4;
7169
7170          /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */

```

```

7171
7172     s_strich = 0;
7173
7174     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7175
7176     s_zweistrich = 0;
7177 }
7178 else if( index4==1 )
7179 {
7180     /* Hub in den Bewegungsabschnitten von phi */
7181
7182     s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
7183         (15/phi_H34)*(pow(phi-phi_3, 4))+6/(pow(phi_H34, 2)))*
7184         (pow(phi-phi_3, 5)))));
7185
7186     s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
7187         (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
7188         (pow((phi-phi_3+deltaphi), 4))+6/(pow(phi_H34, 2)))*
7189         (pow((phi-phi_3+deltaphi), 5)))));
7190
7191     s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
7192         (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
7193         (pow((phi-phi_3-deltaphi), 4))+6/(pow(phi_H34, 2)))*
7194         (pow((phi-phi_3-deltaphi), 5)))));
7195
7196     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7197
7198     s_strich=((s_H34/(pow((phi_H34*rad), 3)))*(30*(pow((phi-phi_3)*rad, 2))-
7199         (60/(phi_H34*rad))*(pow((phi-phi_3)*rad, 3))+
7200         (30/(pow((phi_H34*rad), 2)))*(pow((phi-phi_3)*rad, 4))));
7201
7202     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7203
7204     s_zweistrich=((s_H34/(pow(phi_H34, 3)))*(60*(phi-phi_3)-((180/phi_H34)*
7205         (pow((phi-phi_3), 2)))+((120/(pow(phi_H34, 2)))*
7206         (pow((phi-phi_3), 3))));
7207 }
7208 else if( index4==2 )
7209 {
7210     /* Bestehornsinoide */
7211
7212     s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
7213         ((phi-phi_3)/phi_H34))));
7214

```

```

7215     s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
7216                             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
7217                             phi_H34))));
7218
7219     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
7220                             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
7221                             phi_H34))));
7222
7223     s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
7224
7225     s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
7226 }
7227 }
7228 /* fuerter Abschnitt */
7229 else if(phi>=phi_4 && phi<phi_5)
7230 {
7231     if( index5==0 )
7232     {
7233         /* Hub in den Bewegungsabschnitten von phi */
7234
7235         s = s_5;
7236         s_plus_deltaphi = s_5;
7237         s_minus_deltaphi = s_5;
7238
7239         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7240
7241         s_strich = 0;
7242
7243         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7244
7245         s_zweistrich = 0;
7246     }
7247 else if( index5==1 )
7248 {
7249     /* Hub in den Bewegungsabschnitten von phi */
7250
7251     s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-
7252         (15/phi_H45)*(pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2)))*
7253         (pow(phi-phi_4, 5))))));
7254
7255     s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
7256         (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
7257         (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2)))*
7258         (pow((phi-phi_4+deltaphi), 5))))));

```

```

7259
7260     s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
7261         (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*
7262         (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
7263         (pow((phi-phi_4-deltaphi), 5)))));
7264
7265     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7266
7267     s_strich=((s_H45/(pow((phi_H45*rad), 3)))*(30*(pow(((phi-phi_4)*rad), 2))-
7268         (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad), 3))+
7269         (30/(pow((phi_H45*rad), 2)))*(pow(((phi-phi_4)*rad), 4))));
7270
7271     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7272
7273     s_zweistrich=((s_H45/(pow(phi_H45, 3)))*(60*(phi-phi_4)-((180/phi_H45)*
7274         (pow((phi-phi_4), 2)))+(120/(pow(phi_H45, 2)))*
7275         (pow((phi-phi_4), 3))));
7276 }
7277 else if( index5==2 )
7278 {
7279     /* Bestehornsinoide */
7280
7281     s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
7282         ((phi-phi_4)/phi_H45))));
7283
7284     s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-
7285         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/
7286         phi_H45))));
7287
7288     s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
7289         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
7290         phi_H45))));
7291
7292     s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
7293
7294     s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
7295 }
7296 }
7297 /* sechster Abschnitt */
7298 else
7299 {
7300     if( index6==0 )
7301     {
7302         /* Hub in den Bewegungsabschnitten von phi */

```

```

7303
7304     s = s_6;
7305     s_plus_deltaphi = s_6;
7306     s_minus_deltaphi = s_6;
7307
7308     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7309
7310     s_strich = 0;
7311
7312     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7313
7314     s_zweistrich = 0;
7315 }
7316 else if( index6==1 )
7317 {
7318     /* Hub in den Bewegungsabschnitten von phi */
7319
7320     s = (s_5+((s_H56/(pow(phi_H56, 3)))*(10*(pow(phi-phi_5, 3))-
7321         (15/phi_H56)*(pow(phi-phi_5, 4))+(6/(pow(phi_H56, 2)))*
7322         (pow(phi-phi_5, 5))))));
7323
7324     s_plus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
7325         (10*(pow((phi-phi_5+deltaphi), 3))-(15/phi_H56)*
7326         (pow((phi-phi_5+deltaphi), 4))+(6/(pow(phi_H56, 2)))*
7327         (pow((phi-phi_5+deltaphi), 5))))));
7328
7329     s_minus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
7330         (10*(pow((phi-phi_5-deltaphi), 3))-(15/phi_H56)*
7331         (pow((phi-phi_5-deltaphi), 4))+(6/(pow(phi_H56, 2)))*
7332         (pow((phi-phi_5-deltaphi), 5))))));
7333
7334     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7335
7336     s_strich=((s_H56/(pow((phi_H56*rad),3)))*(30*(pow(((phi-phi_5)*rad),2))-
7337         (60/(phi_H56*rad))*(pow(((phi-phi_5)*rad),3))+
7338         (30/(pow((phi_H56*rad),2)))*(pow(((phi-phi_5)*rad),4))));
7339
7340     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7341
7342     s_zweistrich=((s_H56/(pow(phi_H56,3)))*(60*(phi-phi_5)-((180/phi_H56)*
7343         (pow((phi-phi_5),2)))+((120/(pow(phi_H56,2)))*
7344         (pow((phi-phi_5),3))));
7345 }
7346 else if( index6==2 )

```



```

7347     {
7348         /* Bestehornsinoide */
7349
7350         s = s_5+((s_H56/phi_H56)*(phi-phi_5)-(s_H56/(2*pi)*sin(2*pi*
7351             ((phi-phi_5)/phi_H56))));
7352
7353         s_plus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5+deltaphi)-
7354             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5+deltaphi)/
7355                 phi_H56))));
7356
7357         s_minus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5-deltaphi)-
7358             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5-deltaphi)/
7359                 phi_H56))));
7360
7361         s_strich=s_H56*(1-cos(2*pi*((phi-phi_5)/phi_H56)));
7362
7363         s_zweistrich=2*pi*sin(2*pi*((phi-phi_5)/phi_H56));
7364     }
7365 }
7366 }
7367
7368 void Opticurv::berechneHub7()
7369 {
7370     /* erster Abschnitt */
7371     if( phi>=0 && phi<phi_1 )
7372     {
7373         if( index1==0 )
7374         {
7375             /* Hub in den Bewegungsabschnitten von phi */
7376
7377             s = s_1;
7378             s_plus_deltaphi =s_1;
7379             s_minus_deltaphi = s_1;
7380
7381             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7382
7383             s_strich = 0;
7384
7385             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7386
7387             s_zweistrich = 0;
7388         }
7389         else if( index1==1 )
7390         {

```

```

7391      /* Hub in den Bewegungsabschnitten von phi */
7392
7393      s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
7394          (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
7395
7396      s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
7397          (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
7398          (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
7399          (pow((phi+deltaphi), 5))));
7400
7401      s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
7402          (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
7403          (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
7404          (pow((phi-deltaphi), 5))));
7405
7406      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7407
7408      s_strich=((s_H01/(pow((phi_H01*rad), 3)))*(30*(pow((phi)*rad), 2))-
7409          (60/(phi_H01*rad))*(pow((phi)*rad), 3))+
7410          (30/(pow((phi_H01*rad), 2)))*(pow((phi)*rad), 4)));
7411
7412      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7413
7414      s_zweistrich=((s_H01/(pow(phi_H01, 3)))*(60*(phi)-((180/phi_H01)*
7415          (pow((phi), 2)))+(120/(pow(phi_H01, 2)))*
7416          (pow((phi), 3))));
7417  }
7418  else if( index1==2 )
7419  {
7420      /* Bestehornsinoide */
7421
7422      s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
7423
7424      s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
7425          sin(2*pi*((phi+deltaphi)/phi_H01))));
7426
7427      s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
7428          sin(2*pi*((phi-deltaphi)/phi_H01))));
7429
7430      s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
7431
7432      s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
7433  }
7434  }

```

```

7435     /* zweiter Abschnitt */
7436     else if(phi>= phi_1 && phi<phi_2)
7437     {
7438         if( index2==0 )
7439         {
7440             /* Hub in den Bewegungsabschnitten von phi */
7441
7442             s = s_2;
7443             s_plus_deltaphi = s_2;
7444             s_minus_deltaphi = s_2;
7445
7446             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7447
7448             s_strich = 0;
7449
7450             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7451
7452             s_zweistrich = 0;
7453         }
7454     else if( index2==1 )
7455     {
7456         /* Hub in den Bewegungsabschnitten von phi */
7457
7458         s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
7459             (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
7460             (pow(phi-phi_1, 5))))));
7461
7462         s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
7463             (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
7464             (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
7465             (pow((phi-phi_1+deltaphi), 5))))));
7466
7467         s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
7468             (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
7469             (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
7470             (pow((phi-phi_1-deltaphi), 5))))));
7471
7472         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7473
7474         s_strich=((s_H12/(pow((phi_H12*rad), 3)))*(30*(pow(((phi-phi_1)*rad), 2))-
7475             (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad), 3))+
7476             (30/(pow((phi_H12*rad), 2)))*(pow(((phi-phi_1)*rad), 4))));
7477
7478         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */

```

```

7479
7480     s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
7481         (pow((phi-phi_1),2)))+((120/(pow(phi_H12,2)))*
7482         (pow((phi-phi_1),3)))));
7483 }
7484 else if( index2==2 )
7485 {
7486     /* Bestehornsinoide */
7487
7488     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
7489         ((phi-phi_1)/phi_H12))));
7490
7491     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
7492         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
7493         phi_H12))));
7494
7495     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
7496         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
7497         phi_H12))));
7498
7499     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
7500
7501     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
7502 }
7503 }
7504 /* dritter Abschnitt */
7505 else if(phi>=phi_2 && phi<phi_3)
7506 {
7507     if( index3==0 )
7508     {
7509         /* Hub in den Bewegungsabschnitten von phi */
7510
7511         s = s_3;
7512         s_plus_deltaphi = s_3;
7513         s_minus_deltaphi = s_3;
7514
7515         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7516
7517         s_strich = 0;
7518
7519         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7520
7521         s_zweistrich = 0;
7522     }

```

```

7523     else if( index3==1 )
7524     {
7525         /* Hub in den Bewegungsabschnitten von phi */
7526
7527         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
7528             (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
7529             (pow(phi-phi_2, 5)))));
7530
7531         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
7532             (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
7533             (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
7534             (pow((phi-phi_2+deltaphi), 5)))));
7535
7536         s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
7537             (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
7538             (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
7539             (pow((phi-phi_2-deltaphi), 5)))));
7540
7541         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7542
7543         s_strich=((s_H23/(pow((phi_H23*rad), 3)))*(30*(pow(((phi-phi_2)*rad), 2))-
7544             (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad), 3))+
7545             (30/(pow((phi_H23*rad), 2)))*(pow(((phi-phi_2)*rad), 4))));
7546
7547         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7548
7549         s_zweistrich=((s_H23/(pow(phi_H23, 3)))*(60*(phi-phi_2)-((180/phi_H23)*
7550             (pow((phi-phi_2), 2)))+(120/(pow(phi_H23, 2)))*
7551             (pow((phi-phi_2), 3))));
7552     }
7553     else if( index3==2 )
7554     {
7555         /* Bestehornsinoide */
7556
7557         s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
7558             ((phi-phi_2)/phi_H23))));
7559
7560         s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
7561             (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
7562             phi_H23))));
7563
7564         s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
7565             (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
7566             phi_H23))));

```

```

7567
7568     s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
7569
7570     s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
7571 }
7572 }
7573 /* vierter Abschnitt */
7574 else if(phi>=phi_3 && phi<phi_4)
7575 {
7576     if( index4==0 )
7577     {
7578         /* Hub in den Bewegungsabschnitten von phi */
7579
7580         s = s_4;
7581         s_plus_deltaphi = s_4;
7582         s_minus_deltaphi = s_4;
7583
7584         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7585
7586         s_strich = 0;
7587
7588         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7589
7590         s_zweistrich = 0;
7591     }
7592     else if( index4==1 )
7593     {
7594         /* Hub in den Bewegungsabschnitten von phi */
7595
7596         s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
7597             (15/phi_H34)*(pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*
7598             (pow(phi-phi_3, 5))))));
7599
7600         s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
7601             (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
7602             (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
7603             (pow((phi-phi_3+deltaphi), 5))))));
7604
7605         s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
7606             (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
7607             (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
7608             (pow((phi-phi_3-deltaphi), 5))))));
7609
7610         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */

```

```

7611
7612     s_strich=((s_H34/(pow((phi_H34*rad),3)))*(30*(pow(((phi-phi_3)*rad),2))-
7613             (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad),3))+
7614             (30/(pow((phi_H34*rad),2)))*(pow(((phi-phi_3)*rad),4))));
7615
7616     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7617
7618     s_zweistrich=((s_H34/(pow(phi_H34,3)))*(60*(phi-phi_3)-((180/phi_H34)*
7619             (pow((phi-phi_3),2)))+(120/(pow(phi_H34,2)))*
7620             (pow((phi-phi_3),3))));
7621 }
7622 else if( index4==2 )
7623 {
7624     /* Bestehornsinoide */
7625
7626     s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
7627             ((phi-phi_3)/phi_H34))));
7628
7629     s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
7630             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
7631             phi_H34))));
7632
7633     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
7634             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
7635             phi_H34))));
7636
7637     s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
7638
7639     s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
7640 }
7641 }
7642 /* fuerter Abschnitt */
7643 else if(phi>=phi_4 && phi<phi_5)
7644 {
7645     if( index5==0 )
7646     {
7647         /* Hub in den Bewegungsabschnitten von phi */
7648
7649         s = s_5;
7650         s_plus_deltaphi = s_5;
7651         s_minus_deltaphi = s_5;
7652
7653         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7654

```

```

7655     s_strich = 0;
7656
7657     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7658
7659     s_zweistrich = 0;
7660 }
7661 else if( index5==1 )
7662 {
7663     /* Hub in den Bewegungsabschnitten von phi */
7664
7665     s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-
7666         (15/phi_H45)*(pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2)))*
7667         (pow(phi-phi_4, 5))))));
7668
7669     s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
7670         (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
7671         (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2)))*
7672         (pow((phi-phi_4+deltaphi), 5))))));
7673
7674     s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
7675         (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*
7676         (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
7677         (pow((phi-phi_4-deltaphi), 5))))));
7678
7679     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7680
7681     s_strich=((s_H45/(pow((phi_H45*rad),3)))*(30*(pow(((phi-phi_4)*rad),2))-
7682         (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad),3))+
7683         (30/(pow((phi_H45*rad),2)))*(pow(((phi-phi_4)*rad),4))));
7684
7685     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7686
7687     s_zweistrich=((s_H45/(pow(phi_H45,3)))*(60*(phi-phi_4)-((180/phi_H45)*
7688         (pow((phi-phi_4),2)))+((120/(pow(phi_H45,2)))*
7689         (pow((phi-phi_4),3)))));
7690 }
7691 else if( index5==2 )
7692 {
7693     /* Bestehornsinoide */
7694
7695     s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
7696         ((phi-phi_4)/phi_H45))));
7697
7698     s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-

```



```

7699             (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/
7700             phi_H45)))));
7701
7702     s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
7703             (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
7704             phi_H45)))));
7705
7706     s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
7707
7708     s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
7709 }
7710 }
7711 /* sechster Abschnitt */
7712 else if(phi>=phi_5 && phi<phi_6)
7713 {
7714     if( index6==0 )
7715     {
7716         /* Hub in den Bewegungsabschnitten von phi */
7717
7718         s = s_6;
7719         s_plus_deltaphi = s_6;
7720         s_minus_deltaphi = s_6;
7721
7722         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7723
7724         s_strich = 0;
7725
7726         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7727
7728         s_zweistrich = 0;
7729     }
7730 else if( index6==1 )
7731 {
7732     /* Hub in den Bewegungsabschnitten von phi */
7733
7734     s = (s_5+((s_H56/(pow(phi_H56, 3)))*(10*(pow(phi-phi_5, 3))-
7735             (15/phi_H56)*(pow(phi-phi_5, 4))+(6/(pow(phi_H56, 2)))*
7736             (pow(phi-phi_5, 5))))));
7737
7738     s_plus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
7739             (10*(pow((phi-phi_5+deltaphi), 3))-(15/phi_H56)*
7740             (pow((phi-phi_5+deltaphi), 4))+(6/(pow(phi_H56, 2)))*
7741             (pow((phi-phi_5+deltaphi), 5))))));
7742

```

```

7743     s_minus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
7744         (10*(pow((phi-phi_5-deltaphi), 3))-(15/phi_H56)*
7745         (pow((phi-phi_5-deltaphi), 4))+(6/(pow(phi_H56, 2)))*
7746         (pow((phi-phi_5-deltaphi), 5)))));
7747
7748     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7749
7750     s_strich=((s_H56/(pow((phi_H56*rad), 3)))*(30*(pow(((phi-phi_5)*rad), 2))-
7751         (60/(phi_H56*rad))*(pow(((phi-phi_5)*rad), 3))+
7752         (30/(pow((phi_H56*rad), 2)))*(pow(((phi-phi_5)*rad), 4))));
7753
7754     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7755
7756     s_zweistrich=((s_H56/(pow(phi_H56, 3)))*(60*(phi-phi_5)-((180/phi_H56)*
7757         (pow((phi-phi_5), 2)))+((120/(pow(phi_H56, 2)))*
7758         (pow((phi-phi_5), 3))));
7759 }
7760 else if( index6==2 )
7761 {
7762     /* Bestehornsinoide */
7763
7764     s = s_5+((s_H56/phi_H56)*(phi-phi_5)-(s_H56/(2*pi)*sin(2*pi*
7765         ((phi-phi_5)/phi_H56))));
7766
7767     s_plus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5+deltaphi)-
7768         (s_H56/(2*pi)*sin(2*pi*((phi-phi_5+deltaphi)/
7769         phi_H56))));
7770
7771     s_minus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5-deltaphi)-
7772         (s_H56/(2*pi)*sin(2*pi*((phi-phi_5-deltaphi)/
7773         phi_H56))));
7774
7775     s_strich=s_H56*(1-cos(2*pi*((phi-phi_5)/phi_H56)));
7776
7777     s_zweistrich=2*pi*sin(2*pi*((phi-phi_5)/phi_H56));
7778 }
7779 }
7780 /* siebter Abschnitt */
7781 else
7782 {
7783     if( index7==0 )
7784     {
7785         /* Hub in den Bewegungsabschnitten von phi */
7786

```

```

7787     s = s_7;
7788     s_plus_deltaphi = s_7;
7789     s_minus_deltaphi = s_7;
7790
7791     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7792
7793     s_strich = 0;
7794
7795     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7796
7797     s_zweistrich = 0;
7798 }
7799 else if( index7==1 )
7800 {
7801     /* Hub in den Bewegungsabschnitten von phi */
7802
7803     s = (s_6+((s_H67/(pow(phi_H67, 3)))*(10*(pow(phi-phi_6, 3))-
7804         (15/phi_H67)*(pow(phi-phi_6, 4))+(6/(pow(phi_H67, 2)))*
7805         (pow(phi-phi_6, 5))))));
7806
7807     s_plus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
7808         (10*(pow((phi-phi_6+deltaphi), 3))-(15/phi_H67)*
7809         (pow((phi-phi_6+deltaphi), 4))+(6/(pow(phi_H67, 2)))*
7810         (pow((phi-phi_6+deltaphi), 5))))));
7811
7812     s_minus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
7813         (10*(pow((phi-phi_6-deltaphi), 3))-(15/phi_H67)*
7814         (pow((phi-phi_6-deltaphi), 4))+(6/(pow(phi_H67, 2)))*
7815         (pow((phi-phi_6-deltaphi), 5))))));
7816
7817     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7818
7819     s_strich=((s_H67/(pow((phi_H67*rad), 3)))*(30*(pow(((phi-phi_6)*rad), 2))-
7820         (60/(phi_H67*rad))*(pow(((phi-phi_6)*rad), 3))+
7821         (30/(pow((phi_H67*rad), 2)))*(pow(((phi-phi_6)*rad), 4))));
7822
7823     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7824
7825     s_zweistrich=((s_H67/(pow(phi_H67, 3)))*(60*(phi-phi_6)-((180/phi_H67)*
7826         (pow((phi-phi_6), 2)))+((120/(pow(phi_H67, 2)))*
7827         (pow((phi-phi_6), 3))));
7828 }
7829 else if( index7==2 )
7830 {

```

```

7831      /* Bestehornsinoide */
7832
7833      s = s_6+((s_H67/phi_H67)*(phi-phi_6)-(s_H67/(2*pi)*sin(2*pi*
7834          ((phi-phi_6)/phi_H67))));
7835
7836      s_plus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6+deltaphi)-
7837          (s_H67/(2*pi)*sin(2*pi*((phi-phi_6+deltaphi)/
7838              phi_H67))));
7839
7840      s_minus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6-deltaphi)-
7841          (s_H67/(2*pi)*sin(2*pi*((phi-phi_6-deltaphi)/
7842              phi_H67))));
7843
7844      s_strich=s_H67*(1-cos(2*pi*((phi-phi_6)/phi_H67)));
7845
7846      s_zweistrich=2*pi*sin(2*pi*((phi-phi_6)/phi_H67));
7847  }
7848  }
7849  }
7850
7851  void Opticurv::berechneHub8()
7852  {
7853      /* erster Abschnitt */
7854      if( phi>=0 && phi<phi_1 )
7855      {
7856          if( index1==0 )
7857          {
7858              /* Hub in den Bewegungsabschnitten von phi */
7859
7860              s = s_1;
7861              s_plus_deltaphi = s_1;
7862              s_minus_deltaphi = s_1;
7863
7864              /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7865
7866              s_strich = 0;
7867
7868              /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7869
7870              s_zweistrich = 0;
7871          }
7872          else if( index1==1 )
7873          {
7874              /* Hub in den Bewegungsabschnitten von phi */

```

```

7875
7876 s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
7877      (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
7878
7879 s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
7880      (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
7881      (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
7882      (pow((phi+deltaphi), 5))));
7883
7884 s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
7885      (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
7886      (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
7887      (pow((phi-deltaphi), 5))));
7888
7889 /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7890
7891 s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
7892      (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
7893      (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4))));
7894
7895 /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7896
7897 s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
7898      (pow((phi),2)))+(120/(pow(phi_H01,2)))*
7899      (pow((phi),3))));
7900 }
7901 else if( index1==2 )
7902 {
7903     /* Bestehornsinoide */
7904
7905     s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
7906
7907     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
7908         sin(2*pi*((phi+deltaphi)/phi_H01))));
7909
7910     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
7911         sin(2*pi*((phi-deltaphi)/phi_H01))));
7912
7913     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
7914
7915     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
7916 }
7917 }
7918 /* zweiter Abschnitt */

```

```

7919 else if(phi>= phi_1 && phi<phi_2)
7920 {
7921     if( index2==0 )
7922     {
7923         /* Hub in den Bewegungsabschnitten von phi */
7924
7925         s = s_2;
7926         s_plus_deltaphi = s_2;
7927         s_minus_deltaphi = s_2;
7928
7929         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7930
7931         s_strich = 0;
7932
7933         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7934
7935         s_zweistrich = 0;
7936     }
7937 else if( index2==1 )
7938 {
7939     /* Hub in den Bewegungsabschnitten von phi */
7940
7941     s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
7942         (15/phi_H12)*
7943         (pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*(pow(phi-phi_1, 5))))));
7944
7945     s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
7946         (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
7947         (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
7948         (pow((phi-phi_1+deltaphi), 5))))));
7949
7950     s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
7951         (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
7952         (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
7953         (pow((phi-phi_1-deltaphi), 5))))));
7954
7955     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7956
7957     s_strich=((s_H12/(pow((phi_H12*rad),3)))*(30*(pow(((phi-phi_1)*rad),2))-
7958         (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad),3))+
7959         (30/(pow((phi_H12*rad),2)))*(pow(((phi-phi_1)*rad),4))));
7960
7961     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
7962

```

```

7963     s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
7964         (pow((phi-phi_1),2)))+((120/(pow(phi_H12,2)))*
7965         (pow((phi-phi_1),3)))));
7966 }
7967 else if( index2==2 )
7968 {
7969     /* Bestehornsinoide */
7970
7971     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
7972         ((phi-phi_1)/phi_H12))));
7973
7974     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
7975         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
7976         phi_H12))));
7977
7978     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
7979         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
7980         phi_H12))));
7981
7982     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
7983
7984     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
7985 }
7986 }
7987 /* dritter Abschnitt */
7988 else if(phi>=phi_2 && phi<phi_3)
7989 {
7990     if( index3==0 )
7991     {
7992         /* Hub in den Bewegungsabschnitten von phi */
7993
7994         s = s_3;
7995         s_plus_deltaphi = s_3;
7996         s_minus_deltaphi = s_3;
7997
7998         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
7999
8000         s_strich = 0;
8001
8002         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8003
8004         s_zweistrich = 0;
8005     }
8006     else if( index3==1 )

```

```

8007 {
8008     /* Hub in den Bewegungsabschnitten von phi */
8009
8010     s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
8011         (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
8012         (pow(phi-phi_2, 5)))));
8013
8014     s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
8015         (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
8016         (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
8017         (pow((phi-phi_2+deltaphi), 5)))));
8018
8019     s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
8020         (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
8021         (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
8022         (pow((phi-phi_2-deltaphi), 5)))));
8023
8024     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8025
8026     s_strich=((s_H23/(pow((phi_H23*rad), 3)))*(30*(pow(((phi-phi_2)*rad), 2))-
8027         (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad), 3))+
8028         (30/(pow((phi_H23*rad), 2)))*(pow(((phi-phi_2)*rad), 4))));
8029
8030     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8031
8032     s_zweistrich=((s_H23/(pow(phi_H23, 3)))*(60*(phi-phi_2)-((180/phi_H23)*
8033         (pow((phi-phi_2), 2)))+((120/(pow(phi_H23, 2)))*
8034         (pow((phi-phi_2), 3))));
8035 }
8036 else if( index3==2 )
8037 {
8038     /* Bestehornsinoide */
8039
8040     s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
8041         ((phi-phi_2)/phi_H23))));
8042
8043     s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
8044         (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
8045         phi_H23))));
8046
8047     s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
8048         (s_H23/(2*pi)*
8049         sin(2*pi*((phi-phi_2-deltaphi)/phi_H23))));
8050

```



```

8051         s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
8052
8053         s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
8054     }
8055 }
8056 /* vierter Abschnitt */
8057 else if(phi>=phi_3 && phi<phi_4)
8058 {
8059     if( index4==0 )
8060     {
8061         /* Hub in den Bewegungsabschnitten von phi */
8062
8063         s = s_4;
8064         s_plus_deltaphi = s_4;
8065         s_minus_deltaphi = s_4;
8066
8067         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8068
8069         s_strich = 0;
8070
8071         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8072
8073         s_zweistrich = 0;
8074     }
8075     else if( index4==1 )
8076     {
8077         /* Hub in den Bewegungsabschnitten von phi */
8078
8079         s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
8080             (15/phi_H34)*(pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*
8081             (pow(phi-phi_3, 5))))));
8082
8083         s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
8084             (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
8085             (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
8086             (pow((phi-phi_3+deltaphi), 5))))));
8087
8088         s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
8089             (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
8090             (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
8091             (pow((phi-phi_3-deltaphi), 5))))));
8092
8093         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8094

```

```

8095     s_strich=((s_H34/(pow((phi_H34*rad),3)))*(30*(pow(((phi-phi_3)*rad),2))-
8096             (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad),3))+
8097             (30/(pow((phi_H34*rad),2))*(pow(((phi-phi_3)*rad),4))));
8098
8099     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8100
8101     s_zweistrich=((s_H34/(pow(phi_H34,3)))*(60*(phi-phi_3)-((180/phi_H34)*
8102             (pow((phi-phi_3),2)))+((120/(pow(phi_H34,2)))*
8103             (pow((phi-phi_3),3)))));
8104 }
8105 else if( index4==2 )
8106 {
8107     /* Bestehornsinoide */
8108
8109     s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
8110             ((phi-phi_3)/phi_H34))));
8111
8112     s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
8113             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
8114             phi_H34))));
8115
8116     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
8117             (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
8118             phi_H34))));
8119
8120     s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
8121
8122     s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
8123 }
8124 }
8125 /* fuerfter Abschnitt */
8126 else if(phi>=phi_4 && phi<phi_5)
8127 {
8128     if( index5==0 )
8129     {
8130         /* Hub in den Bewegungsabschnitten von phi */
8131
8132         s = s_5;
8133         s_plus_deltaphi = s_5;
8134         s_minus_deltaphi = s_5;
8135
8136         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8137
8138         s_strich = 0;

```

```

8139
8140     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8141
8142     s_zweistrich = 0;
8143 }
8144 else if( index5==1 )
8145 {
8146     /* Hub in den Bewegungsabschnitten von phi */
8147
8148     s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-
8149         (15/phi_H45)*(pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2)))*
8150         (pow(phi-phi_4, 5))))));
8151
8152     s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
8153         (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
8154         (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2)))*
8155         (pow((phi-phi_4+deltaphi), 5))))));
8156
8157     s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
8158         (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*
8159         (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
8160         (pow((phi-phi_4-deltaphi), 5))))));
8161
8162     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8163
8164     s_strich=((s_H45/(pow((phi_H45*rad), 3)))*(30*(pow(((phi-phi_4)*rad), 2))-
8165         (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad), 3))+
8166         (30/(pow((phi_H45*rad), 2)))*(pow(((phi-phi_4)*rad), 4))));
8167
8168     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8169
8170     s_zweistrich=((s_H45/(pow(phi_H45, 3)))*(60*(phi-phi_4)-((180/phi_H45)*
8171         (pow((phi-phi_4), 2)))+(120/(pow(phi_H45, 2)))*
8172         (pow((phi-phi_4), 3))));
8173 }
8174 else if( index5==2 )
8175 {
8176     /* Bestehornsinoide */
8177
8178     s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
8179         ((phi-phi_4)/phi_H45))));
8180
8181     s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-
8182         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/

```

```

8183         phi_H45)))));
8184
8185     s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
8186         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
8187             phi_H45))));
8188
8189     s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
8190
8191     s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
8192 }
8193 }
8194 /* sechster Abschnitt */
8195 else if(phi>=phi_5 && phi<phi_6)
8196 {
8197     if( index6==0 )
8198     {
8199         /* Hub in den Bewegungsabschnitten von phi */
8200
8201         s = s_6;
8202         s_plus_deltaphi = s_6;
8203         s_minus_deltaphi = s_6;
8204
8205         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8206
8207         s_strich = 0;
8208
8209         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8210
8211         s_zweistrich = 0;
8212     }
8213     else if( index6==1 )
8214     {
8215         /* Hub in den Bewegungsabschnitten von phi */
8216
8217         s = (s_5+((s_H56/(pow(phi_H56, 3)))*(10*(pow(phi-phi_5, 3))-
8218             (15/phi_H56)*(pow(phi-phi_5, 4))+6/(pow(phi_H56, 2)))*
8219             (pow(phi-phi_5, 5)))));
8220
8221         s_plus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
8222             (10*(pow((phi-phi_5+deltaphi), 3))-(15/phi_H56)*
8223             (pow((phi-phi_5+deltaphi), 4))+6/(pow(phi_H56, 2)))*
8224             (pow((phi-phi_5+deltaphi), 5)))));
8225
8226         s_minus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*

```

```

8227         (10*(pow((phi-phi_5-deltaphi), 3))-(15/phi_H56)*
8228         (pow((phi-phi_5-deltaphi), 4))+(6/(pow(phi_H56, 2)))*
8229         (pow((phi-phi_5-deltaphi), 5))));
8230
8231     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8232
8233     s_strich=((s_H56/(pow((phi_H56*rad),3)))*(30*(pow(((phi-phi_5)*rad),2))-
8234             (60/(phi_H56*rad))*(pow(((phi-phi_5)*rad),3))+
8235             (30/(pow((phi_H56*rad),2))*(pow(((phi-phi_5)*rad),4)))));
8236
8237     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8238
8239     s_zweistrich=((s_H56/(pow(phi_H56,3)))*(60*(phi-phi_5)-((180/phi_H56)*
8240             (pow((phi-phi_5),2)))+((120/(pow(phi_H56,2)))*
8241             (pow((phi-phi_5),3))));
8242 }
8243 else if( index6==2 )
8244 {
8245     /* Bestehornsinoide */
8246
8247     s = s_5+((s_H56/phi_H56)*(phi-phi_5)-(s_H56/(2*pi)*sin(2*pi*
8248             ((phi-phi_5)/phi_H56))));
8249
8250     s_plus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5+deltaphi)-
8251             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5+deltaphi)/
8252             phi_H56))));
8253
8254     s_minus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5-deltaphi)-
8255             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5-deltaphi)/
8256             phi_H56))));
8257
8258     s_strich=s_H56*(1-cos(2*pi*((phi-phi_5)/phi_H56)));
8259
8260     s_zweistrich=2*pi*sin(2*pi*((phi-phi_5)/phi_H56));
8261 }
8262 }
8263 /* siebter Abschnitt */
8264 else if(phi>=phi_6 && phi<phi_7)
8265 {
8266     if( index7==0 )
8267     {
8268         /* Hub in den Bewegungsabschnitten von phi */
8269
8270         s = s_7;

```

```

8271     s_plus_deltaphi = s_7;
8272     s_minus_deltaphi = s_7;
8273
8274     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8275
8276     s_strich = 0;
8277
8278     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8279
8280     s_zweistrich = 0;
8281 }
8282 else if( index7==1 )
8283 {
8284     /* Hub in den Bewegungsabschnitten von phi */
8285
8286     s = (s_6+((s_H67/(pow(phi_H67, 3)))*(10*(pow(phi-phi_6, 3))-
8287         (15/phi_H67)*(pow(phi-phi_6, 4))+(6/(pow(phi_H67, 2)))*
8288         (pow(phi-phi_6, 5))))));
8289
8290     s_plus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
8291         (10*(pow((phi-phi_6+deltaphi), 3))-(15/phi_H67)*
8292         (pow((phi-phi_6+deltaphi), 4))+(6/(pow(phi_H67, 2)))*
8293         (pow((phi-phi_6+deltaphi), 5))))));
8294
8295     s_minus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
8296         (10*(pow((phi-phi_6-deltaphi), 3))-(15/phi_H67)*
8297         (pow((phi-phi_6-deltaphi), 4))+(6/(pow(phi_H67, 2)))*
8298         (pow((phi-phi_6-deltaphi), 5))))));
8299
8300     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8301
8302     s_strich=((s_H67/(pow((phi_H67*rad),3)))*(30*(pow(((phi-phi_6)*rad),2))-
8303         (60/(phi_H67*rad))*(pow(((phi-phi_6)*rad),3))+
8304         (30/(pow((phi_H67*rad),2)))*(pow(((phi-phi_6)*rad),4))));
8305
8306     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8307
8308     s_zweistrich=((s_H67/(pow(phi_H67,3)))*(60*(phi-phi_6)-((180/phi_H67)*
8309         (pow((phi-phi_6),2)))+((120/(pow(phi_H67,2)))*
8310         (pow((phi-phi_6),3)))));
8311 }
8312 else if( index7==2 )
8313 {
8314     /* Bestehornsinoide */

```

```

8315
8316     s = s_6+((s_H67/phi_H67)*(phi-phi_6)-(s_H67/(2*pi)*sin(2*pi*
8317         ((phi-phi_6)/phi_H67))));
8318
8319     s_plus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6+deltaphi)-
8320         (s_H67/(2*pi)*sin(2*pi*((phi-phi_6+deltaphi)/
8321             phi_H67))));
8322
8323     s_minus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6-deltaphi)-
8324         (s_H67/(2*pi)*sin(2*pi*((phi-phi_6-deltaphi)/
8325             phi_H67))));
8326
8327     s_strich=s_H67*(1-cos(2*pi*((phi-phi_6)/phi_H67)));
8328
8329     s_zweistrich=2*pi*sin(2*pi*((phi-phi_6)/phi_H67));
8330 }
8331 }
8332 /* achter Abschnitt */
8333 else
8334 {
8335     if( index8==0 )
8336     {
8337         /* Hub in den Bewegungsabschnitten von phi */
8338
8339         s = s_8;
8340         s_plus_deltaphi = s_8;
8341         s_minus_deltaphi = s_8;
8342
8343         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8344
8345         s_strich = 0;
8346
8347         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8348
8349         s_zweistrich = 0;
8350     }
8351     else if( index8==1 )
8352     {
8353         /* Hub in den Bewegungsabschnitten von phi */
8354
8355         s = (s_7+((s_H78/(pow(phi_H78, 3)))*(10*(pow(phi-phi_7, 3))-
8356             (15/phi_H78)*(pow(phi-phi_7, 4))+(6/(pow(phi_H78, 2)))*
8357             (pow(phi-phi_7, 5))))));
8358

```

```

8359     s_plus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
8360         (10*(pow((phi-phi_7+deltaphi), 3))-(15/phi_H78)*
8361         (pow((phi-phi_7+deltaphi), 4))+(6/(pow(phi_H78, 2)))*
8362         (pow((phi-phi_7+deltaphi), 5)))));
8363
8364     s_minus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
8365         (10*(pow((phi-phi_7-deltaphi), 3))-(15/phi_H78)*
8366         (pow((phi-phi_7-deltaphi), 4))+(6/(pow(phi_H78, 2)))*
8367         (pow((phi-phi_7-deltaphi), 5)))));
8368
8369     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8370
8371     s_strich=((s_H78/(pow((phi_H78*rad), 3)))*(30*(pow(((phi-phi_7)*rad), 2))-
8372         (60/(phi_H78*rad))*(pow(((phi-phi_7)*rad), 3))+
8373         (30/(pow((phi_H78*rad), 2)))*(pow(((phi-phi_7)*rad), 4))));
8374
8375     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8376
8377     s_zweistrich=((s_H78/(pow(phi_H78, 3)))*(60*(phi-phi_7)-((180/phi_H67)*
8378         (pow((phi-phi_6), 2)))+((120/(pow(phi_H67, 2)))*
8379         (pow((phi-phi_6), 3))));
8380 }
8381 else if( index8==2 )
8382 {
8383     /* Bestehornsinoide */
8384
8385     s = s_7+((s_H78/phi_H78)*(phi-phi_7)-(s_H78/(2*pi)*sin(2*pi*
8386         ((phi-phi_7)/phi_H78))));
8387
8388     s_plus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7+deltaphi)-
8389         (s_H78/(2*pi)*sin(2*pi*((phi-phi_7+deltaphi)/
8390         phi_H78))));
8391
8392     s_minus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7-deltaphi)-
8393         (s_H78/(2*pi)*sin(2*pi*((phi-phi_7-deltaphi)/
8394         phi_H78))));
8395
8396     s_strich=s_H78*(1-cos(2*pi*((phi-phi_7)/phi_H78)));
8397
8398     s_zweistrich=2*pi*sin(2*pi*((phi-phi_7)/phi_H78));
8399 }
8400 }
8401 }
8402

```



```

8403 void Opticurv::berechneHub9()
8404 {
8405     /* erster Abschnitt */
8406     if( phi>=0 && phi<phi_1 )
8407     {
8408         if( index1==0 )
8409         {
8410             /* Hub in den Bewegungsabschnitten von phi */
8411
8412             s = s_1;
8413             s_plus_deltaphi = s_1;
8414             s_minus_deltaphi = s_1;
8415
8416             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8417
8418             s_strich = 0;
8419
8420             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8421
8422             s_zweistrich = 0;
8423         }
8424     else if( index1==1 )
8425     {
8426         /* Hub in den Bewegungsabschnitten von phi */
8427
8428         s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
8429             (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
8430
8431         s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
8432             (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
8433             (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
8434             (pow((phi+deltaphi), 5))));
8435
8436         s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
8437             (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
8438             (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
8439             (pow((phi-deltaphi), 5))));
8440
8441         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8442
8443         s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
8444             (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
8445             (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4))));
8446

```

```

8447      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8448
8449      s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
8450                  (pow((phi),2)))+(120/(pow(phi_H01,2)))*
8451                  (pow((phi),3)))));
8452  }
8453  else if( index1==2 )
8454  {
8455      /* Bestehornsinoide */
8456
8457      s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
8458
8459      s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
8460                  sin(2*pi*((phi+deltaphi)/phi_H01))));
8461
8462      s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
8463                  sin(2*pi*((phi-deltaphi)/phi_H01))));
8464
8465      s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
8466
8467      s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
8468  }
8469  }
8470  /* zweiter Abschnitt */
8471  else if(phi>= phi_1 && phi<phi_2)
8472  {
8473      if( index2==0 )
8474      {
8475          /* Hub in den Bewegungsabschnitten von phi */
8476
8477          s = s_2;
8478          s_plus_deltaphi =s_2;
8479          s_minus_deltaphi = s_2;
8480
8481          /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8482
8483          s_strich = 0;
8484
8485          /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8486
8487          s_zweistrich = 0;
8488      }
8489      else if( index2==1 )
8490      {

```

```

8491      /* Hub in den Bewegungsabschnitten von phi */
8492
8493      s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
8494          (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
8495          (pow(phi-phi_1, 5))))));
8496
8497      s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
8498          (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
8499          (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
8500          (pow((phi-phi_1+deltaphi), 5))))));
8501
8502      s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
8503          (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
8504          (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
8505          (pow((phi-phi_1-deltaphi), 5))))));
8506
8507      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8508
8509      s_strich=((s_H12/(pow((phi_H12*rad), 3)))*(30*(pow(((phi-phi_1)*rad), 2))-
8510          (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad), 3))+
8511          (30/(pow((phi_H12*rad), 2)))*(pow(((phi-phi_1)*rad), 4))));
8512
8513      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8514
8515      s_zweistrich=((s_H12/(pow(phi_H12, 3)))*(60*(phi-phi_1)-((180/phi_H12)*
8516          (pow((phi-phi_1), 2)))+(120/(pow(phi_H12, 2)))*
8517          (pow((phi-phi_1), 3))));
8518  }
8519  else if( index2==2 )
8520  {
8521      /* Bestehornsinoide */
8522
8523      s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
8524          ((phi-phi_1)/phi_H12))));
8525
8526      s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
8527          (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
8528          phi_H12))));
8529
8530      s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
8531          (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
8532          phi_H12))));
8533
8534      s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));

```

```

8535
8536     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
8537 }
8538 }
8539 /* dritter Abschnitt */
8540 else if(phi>=phi_2 && phi<phi_3)
8541 {
8542     if( index3==0 )
8543     {
8544         /* Hub in den Bewegungsabschnitten von phi */
8545
8546         s = s_3;
8547         s_plus_deltaphi = s_3;
8548         s_minus_deltaphi = s_3;
8549
8550         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8551
8552         s_strich = 0;
8553
8554         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8555
8556         s_zweistrich = 0;
8557     }
8558     else if( index3==1 )
8559     {
8560         /* Hub in den Bewegungsabschnitten von phi */
8561
8562         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
8563             (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2))))*
8564             (pow(phi-phi_2, 5)))));
8565
8566         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
8567             (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
8568             (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2))))*
8569             (pow((phi-phi_2+deltaphi), 5)))));
8570
8571         s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
8572             (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
8573             (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2))))*
8574             (pow((phi-phi_2-deltaphi), 5)))));
8575
8576         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8577
8578         s_strich=((s_H23/(pow((phi_H23*rad),3)))*(30*(pow(((phi-phi_2)*rad),2))-

```

```

8579             (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad),3))+
8580             (30/(pow((phi_H23*rad),2))*(pow(((phi-phi_2)*rad),4))));
8581
8582     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8583
8584     s_zweistrich=((s_H23/(pow(phi_H23,3)))*(60*(phi-phi_2)-((180/phi_H23)*
8585             (pow((phi-phi_2),2)))+((120/(pow(phi_H23,2)))*
8586             (pow((phi-phi_2),3)))));
8587 }
8588 else if( index3==2 )
8589 {
8590     /* Bestehornsinoide */
8591
8592     s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
8593             ((phi-phi_2)/phi_H23))));
8594
8595     s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
8596             (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
8597             phi_H23))));
8598
8599     s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
8600             (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
8601             phi_H23))));
8602
8603     s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
8604
8605     s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
8606 }
8607 }
8608 /* vierter Abschnitt */
8609 else if(phi>=phi_3 && phi<phi_4)
8610 {
8611     if( index4==0 )
8612     {
8613         /* Hub in den Bewegungsabschnitten von phi */
8614
8615         s = s_4;
8616         s_plus_deltaphi = s_4;
8617         s_minus_deltaphi = s_4;
8618
8619         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8620
8621         s_strich = 0;
8622

```

```

8623      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8624
8625      s_zweistrich = 0;
8626  }
8627  else if( index4==1 )
8628  {
8629      /* Hub in den Bewegungsabschnitten von phi */
8630
8631      s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
8632          (15/phi_H34)*(pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*
8633          (pow(phi-phi_3, 5))))));
8634
8635      s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
8636          (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
8637          (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
8638          (pow((phi-phi_3+deltaphi), 5))))));
8639
8640      s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
8641          (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
8642          (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
8643          (pow((phi-phi_3-deltaphi), 5))))));
8644
8645      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8646
8647      s_strich=((s_H34/(pow((phi_H34*rad), 3)))*(30*(pow(((phi-phi_3)*rad), 2))-
8648          (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad), 3))+
8649          (30/(pow((phi_H34*rad), 2)))*(pow(((phi-phi_3)*rad), 4))));
8650
8651      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8652
8653      s_zweistrich=((s_H34/(pow(phi_H34, 3)))*(60*(phi-phi_3)-((180/phi_H34)*
8654          (pow((phi-phi_3), 2)))+((120/(pow(phi_H34, 2)))*
8655          (pow((phi-phi_3), 3)))));
8656  }
8657  else if( index4==2 )
8658  {
8659      /* Bestehornsinoide */
8660
8661      s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
8662          ((phi-phi_3)/phi_H34))));
8663
8664      s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
8665          (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
8666              phi_H34))));

```

```

8667
8668     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
8669         (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
8670             phi_H34))));
8671
8672     s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
8673
8674     s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
8675 }
8676 }
8677 /* fuerfter Abschnitt */
8678 else if(phi>=phi_4 && phi<phi_5)
8679 {
8680     if( index5==0 )
8681     {
8682         /* Hub in den Bewegungsabschnitten von phi */
8683
8684         s = s_5;
8685         s_plus_deltaphi = s_5;
8686         s_minus_deltaphi = s_5;
8687
8688         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8689
8690         s_strich = 0;
8691
8692         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8693
8694         s_zweistrich = 0;
8695     }
8696     else if( index5==1 )
8697     {
8698         /* Hub in den Bewegungsabschnitten von phi */
8699
8700         s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-
8701             (15/phi_H45)*(pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2)))*
8702             (pow(phi-phi_4, 5))))));
8703
8704         s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
8705             (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
8706             (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2)))*
8707             (pow((phi-phi_4+deltaphi), 5))))));
8708
8709         s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
8710             (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*

```

```

8711         (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
8712         (pow((phi-phi_4-deltaphi), 5))));
8713
8714     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8715
8716     s_strich=((s_H45/(pow((phi_H45*rad), 3)))*(30*(pow(((phi-phi_4)*rad), 2))-
8717         (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad), 3))+
8718         (30/(pow((phi_H45*rad), 2)))*(pow(((phi-phi_4)*rad), 4))));
8719
8720     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8721
8722     s_zweistrich=((s_H45/(pow(phi_H45, 3)))*(60*(phi-phi_4)-((180/phi_H45)*
8723         (pow((phi-phi_4), 2)))+(120/(pow(phi_H45, 2)))*
8724         (pow((phi-phi_4), 3))));
8725 }
8726 else if( index5==2 )
8727 {
8728     /* Bestehornsinoide */
8729
8730     s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
8731         ((phi-phi_4)/phi_H45))));
8732
8733     s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-
8734         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/
8735         phi_H45))));
8736
8737     s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
8738         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
8739         phi_H45))));
8740
8741     s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
8742
8743     s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
8744 }
8745 }
8746 /* sechster Abschnitt */
8747 else if(phi>=phi_5 && phi<phi_6)
8748 {
8749     if( index6==0 )
8750     {
8751         /* Hub in den Bewegungsabschnitten von phi */
8752
8753         s = s_6;
8754         s_plus_deltaphi = s_6;

```



```

8755     s_minus_deltaphi = s_6;
8756
8757     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8758
8759     s_strich = 0;
8760
8761     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8762
8763     s_zweistrich = 0;
8764 }
8765 else if( index6==1 )
8766 {
8767     /* Hub in den Bewegungsabschnitten von phi */
8768
8769     s = (s_5+((s_H56/(pow(phi_H56, 3)))*(10*(pow(phi-phi_5, 3))-
8770         (15/phi_H56)*(pow(phi-phi_5, 4))+(6/(pow(phi_H56, 2)))*
8771         (pow(phi-phi_5, 5))))));
8772
8773     s_plus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
8774         (10*(pow((phi-phi_5+deltaphi), 3))-(15/phi_H56)*
8775         (pow((phi-phi_5+deltaphi), 4))+(6/(pow(phi_H56, 2)))*
8776         (pow((phi-phi_5+deltaphi), 5))))));
8777
8778     s_minus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
8779         (10*(pow((phi-phi_5-deltaphi), 3))-(15/phi_H56)*
8780         (pow((phi-phi_5-deltaphi), 4))+(6/(pow(phi_H56, 2)))*
8781         (pow((phi-phi_5-deltaphi), 5))))));
8782
8783     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8784
8785     s_strich=((s_H56/(pow((phi_H56*rad), 3)))*(30*(pow(((phi-phi_5)*rad), 2))-
8786         (60/(phi_H56*rad))*(pow(((phi-phi_5)*rad), 3))+
8787         (30/(pow((phi_H56*rad), 2)))*(pow(((phi-phi_5)*rad), 4))));
8788
8789     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8790
8791     s_zweistrich=((s_H56/(pow(phi_H56, 3)))*(60*(phi-phi_5)-((180/phi_H56)*
8792         (pow((phi-phi_5), 2)))+(120/(pow(phi_H56, 2)))*
8793         (pow((phi-phi_5), 3))));
8794 }
8795 else if( index6==2 )
8796 {
8797     /* Bestehornsinoide */
8798

```

```

8799     s = s_5+((s_H56/phi_H56)*(phi-phi_5)-(s_H56/(2*pi)*sin(2*pi*
8800         ((phi-phi_5)/phi_H56))));
8801
8802     s_plus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5+deltaphi)-
8803         (s_H56/(2*pi)*sin(2*pi*((phi-phi_5+deltaphi)/
8804             phi_H56))));
8805
8806     s_minus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5-deltaphi)-
8807         (s_H56/(2*pi)*sin(2*pi*((phi-phi_5-deltaphi)/
8808             phi_H56))));
8809
8810     s_strich=s_H56*(1-cos(2*pi*((phi-phi_5)/phi_H56)));
8811
8812     s_zweistrich=2*pi*sin(2*pi*((phi-phi_5)/phi_H56));
8813 }
8814 }
8815 /* siebter Abschnitt */
8816 else if(phi>=phi_6 && phi<phi_7)
8817 {
8818     if( index7==0 )
8819     {
8820         /* Hub in den Bewegungsabschnitten von phi */
8821
8822         s = s_7;
8823         s_plus_deltaphi = s_7;
8824         s_minus_deltaphi = s_7;
8825
8826         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8827
8828         s_strich = 0;
8829
8830         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8831
8832         s_zweistrich = 0;
8833     }
8834     else if( index7==1 )
8835     {
8836         /* Hub in den Bewegungsabschnitten von phi */
8837
8838         s = (s_6+((s_H67/(pow(phi_H67, 3)))*(10*(pow(phi-phi_6, 3))-
8839             (15/phi_H67)*(pow(phi-phi_6, 4))+(6/(pow(phi_H67, 2)))*
8840             (pow(phi-phi_6, 5))))));
8841
8842         s_plus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*

```

```

8843         (10*(pow((phi-phi_6+deltaphi), 3))-(15/phi_H67)*
8844         (pow((phi-phi_6+deltaphi), 4))+(6/(pow(phi_H67, 2)))*
8845         (pow((phi-phi_6+deltaphi), 5))));
8846
8847     s_minus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
8848     (10*(pow((phi-phi_6-deltaphi), 3))-(15/phi_H67)*
8849     (pow((phi-phi_6-deltaphi), 4))+(6/(pow(phi_H67, 2)))*
8850     (pow((phi-phi_6-deltaphi), 5))));
8851
8852     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8853
8854     s_strich=((s_H67/(pow((phi_H67*rad),3)))*(30*(pow(((phi-phi_6)*rad),2))-
8855     (60/(phi_H67*rad))*(pow(((phi-phi_6)*rad),3))+
8856     (30/(pow((phi_H67*rad),2)))*(pow(((phi-phi_6)*rad),4))));
8857
8858     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8859
8860     s_zweistrich=((s_H67/(pow(phi_H67,3)))*(60*(phi-phi_6)-((180/phi_H67)*
8861     (pow((phi-phi_6),2)))+((120/(pow(phi_H67,2)))*
8862     (pow((phi-phi_6),3))));
8863 }
8864 else if( index7==2 )
8865 {
8866     /* Bestehornsinoide */
8867
8868     s = s_6+((s_H67/phi_H67)*(phi-phi_6)-(s_H67/(2*pi)*sin(2*pi*
8869     ((phi-phi_6)/phi_H67))));
8870
8871     s_plus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6+deltaphi)-
8872     (s_H67/(2*pi)*sin(2*pi*((phi-phi_6+deltaphi)/
8873     phi_H67))));
8874
8875     s_minus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6-deltaphi)-
8876     (s_H67/(2*pi)*sin(2*pi*((phi-phi_6-deltaphi)/
8877     phi_H67))));
8878
8879     s_strich=s_H67*(1-cos(2*pi*((phi-phi_6)/phi_H67)));
8880
8881     s_zweistrich=2*pi*sin(2*pi*((phi-phi_6)/phi_H67));
8882 }
8883 }
8884 /* achter Abschnitt */
8885 else if(phi>=phi_7 && phi<phi_8)
8886 {

```

```

8887     if( index8==0 )
8888     {
8889         /* Hub in den Bewegungsabschnitten von phi */
8890
8891         s = s_8;
8892         s_plus_deltaphi = s_8;
8893         s_minus_deltaphi = s_8;
8894
8895         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8896
8897         s_strich = 0;
8898
8899         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8900
8901         s_zweistrich = 0;
8902     }
8903     else if( index8==1 )
8904     {
8905         /* Hub in den Bewegungsabschnitten von phi */
8906
8907         s = (s_7+((s_H78/(pow(phi_H78, 3)))*(10*(pow(phi-phi_7, 3))-
8908             (15/phi_H78)*(pow(phi-phi_7, 4))+(6/(pow(phi_H78, 2)))*
8909             (pow(phi-phi_7, 5))))));
8910
8911         s_plus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
8912             (10*(pow((phi-phi_7+deltaphi), 3))-(15/phi_H78)*
8913             (pow((phi-phi_7+deltaphi), 4))+(6/(pow(phi_H78, 2)))*
8914             (pow((phi-phi_7+deltaphi), 5))))));
8915
8916         s_minus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
8917             (10*(pow((phi-phi_7-deltaphi), 3))-(15/phi_H78)*
8918             (pow((phi-phi_7-deltaphi), 4))+(6/(pow(phi_H78, 2)))*
8919             (pow((phi-phi_7-deltaphi), 5))))));
8920
8921         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8922
8923         s_strich=((s_H78/(pow((phi_H78*rad),3)))*(30*(pow(((phi-phi_7)*rad),2))-
8924             (60/(phi_H78*rad))*(pow(((phi-phi_7)*rad),3))+
8925             (30/(pow((phi_H78*rad),2)))*(pow(((phi-phi_7)*rad),4))));
8926
8927         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8928
8929         s_zweistrich=((s_H78/(pow(phi_H78,3)))*(60*(phi-phi_7)-((180/phi_H78)*
8930             (pow((phi-phi_7),2)))+((120/(pow(phi_H78,2)))*

```

```

8931             (pow((phi-phi_7),3)))));
8932     }
8933     else if( index8==2 )
8934     {
8935         /* Bestehornsinoide */
8936
8937         s = s_7+((s_H78/phi_H78)*(phi-phi_7)-(s_H78/(2*pi)*sin(2*pi*
8938             ((phi-phi_7)/phi_H78))));
8939
8940         s_plus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7+deltaphi)-
8941             (s_H78/(2*pi)*sin(2*pi*((phi-phi_7+deltaphi)/
8942             phi_H78))));
8943
8944         s_minus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7-deltaphi)-
8945             (s_H78/(2*pi)*sin(2*pi*((phi-phi_7-deltaphi)/
8946             phi_H78))));
8947
8948         s_strich=s_H78*(1-cos(2*pi*((phi-phi_7)/phi_H78)));
8949
8950         s_zweistrich=2*pi*sin(2*pi*((phi-phi_7)/phi_H78));
8951     }
8952 }
8953 /* neunter Abschnitt */
8954 else
8955 {
8956     if( index9==0 )
8957     {
8958         /* Hub in den Bewegungsabschnitten von phi */
8959
8960         s = s_9;
8961         s_plus_deltaphi = s_9;
8962         s_minus_deltaphi = s_9;
8963
8964         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8965
8966         s_strich = 0;
8967
8968         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8969
8970         s_zweistrich = 0;
8971     }
8972     else if( index9==1 )
8973     {
8974         /* Hub in den Bewegungsabschnitten von phi */

```

```

8975
8976 s = (s_8+((s_H89/(pow(phi_H89, 3)))*(10*(pow(phi-phi_8, 3))-
8977 (15/phi_H89)*(pow(phi-phi_8, 4))+(6/(pow(phi_H89, 2)))*
8978 (pow(phi-phi_8, 5)))));
8979
8980 s_plus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
8981 (10*(pow((phi-phi_8+deltaphi), 3))-(15/phi_H89)*
8982 (pow((phi-phi_8+deltaphi), 4))+(6/(pow(phi_H89, 2)))*
8983 (pow((phi-phi_8+deltaphi), 5)))));
8984
8985 s_minus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
8986 (10*(pow((phi-phi_8-deltaphi), 3))-(15/phi_H89)*
8987 (pow((phi-phi_8-deltaphi), 4))+(6/(pow(phi_H89, 2)))*
8988 (pow((phi-phi_8-deltaphi), 5)))));
8989
8990 /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
8991
8992 s_strich=((s_H89/(pow((phi_H89*rad), 3)))*(30*(pow(((phi-phi_8)*rad), 2))-
8993 (60/(phi_H89*rad))*(pow(((phi-phi_8)*rad), 3))+
8994 (30/(pow((phi_H89*rad), 2)))*(pow(((phi-phi_8)*rad), 4))));
8995
8996 /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
8997
8998 s_zweistrich=((s_H89/(pow(phi_H89, 3)))*(60*(phi-phi_8)-((180/phi_H89)*
8999 (pow((phi-phi_8), 2)))+((120/(pow(phi_H89, 2)))*
9000 (pow((phi-phi_8), 3))));
9001 }
9002 else if( index9==2 )
9003 {
9004     /* Bestehornsinoide */
9005
9006     s = s_8+((s_H89/phi_H89)*(phi-phi_8)-(s_H89/(2*pi)*sin(2*pi*
9007 ((phi-phi_8)/phi_H89))));
9008
9009     s_plus_deltaphi = s_8+((s_H89/phi_H89)*(phi-phi_8+deltaphi)-
9010 (s_H89/(2*pi)*sin(2*pi*((phi-phi_8+deltaphi)/
9011 phi_H89))));
9012
9013     s_minus_deltaphi = s_8+((s_H89/phi_H89)*(phi-phi_8-deltaphi)-
9014 (s_H89/(2*pi)*sin(2*pi*((phi-phi_8-deltaphi)/
9015 phi_H89))));
9016
9017     s_strich=s_H89*(1-cos(2*pi*((phi-phi_8)/phi_H89)));
9018

```

```

9019         s_zweistrich=2*pi*sin(2*pi*((phi-phi_8)/phi_H89));
9020     }
9021 }
9022 }
9023 void Opticurv::berechneHub10()
9024 {
9025     /* erster Abschnitt */
9026     if( phi>=0 && phi<phi_1 )
9027     {
9028         if( index1==0 )
9029         {
9030             /* Hub in den Bewegungsabschnitten von phi */
9031
9032             s = s_1;
9033             s_plus_deltaphi = s_1;
9034             s_minus_deltaphi = s_1;
9035
9036             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9037
9038             s_strich = 0;
9039
9040             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9041
9042             s_zweistrich = 0;
9043         }
9044         else if( index1==1 )
9045         {
9046             /* Hub in den Bewegungsabschnitten von phi */
9047
9048             s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
9049                 (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
9050
9051             s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
9052                 (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
9053                 (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
9054                 (pow((phi+deltaphi), 5))));
9055
9056             s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
9057                 (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
9058                 (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
9059                 (pow((phi-deltaphi), 5))));
9060
9061             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9062

```

```

9063     s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow((phi)*rad),2))-
9064             (60/(phi_H01*rad))*(pow((phi)*rad),3))+
9065             (30/(pow((phi_H01*rad),2))*(pow((phi)*rad),4))));
9066
9067     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9068
9069     s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
9070             (pow((phi),2)))+((120/(pow(phi_H01,2)))*
9071             (pow((phi),3)))));
9072 }
9073 else if( index1==2 )
9074 {
9075     /* Bestehornsinoide */
9076
9077     s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
9078
9079     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
9080             sin(2*pi*((phi+deltaphi)/phi_H01))));
9081
9082     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
9083             sin(2*pi*((phi-deltaphi)/phi_H01))));
9084
9085     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
9086
9087     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
9088 }
9089 }
9090 /* zweiter Abschnitt */
9091 else if(phi>= phi_1 && phi<phi_2)
9092 {
9093     if( index2==0 )
9094     {
9095         /* Hub in den Bewegungsabschnitten von phi */
9096
9097         s = s_2;
9098         s_plus_deltaphi = s_2;
9099         s_minus_deltaphi = s_2;
9100
9101         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9102
9103         s_strich = 0;
9104
9105         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9106

```



```

9107     s_zweistrich = 0;
9108 }
9109 else if( index2==1 )
9110 {
9111     /* Hub in den Bewegungsabschnitten von phi */
9112
9113     s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
9114         (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
9115         (pow(phi-phi_1, 5))))));
9116
9117     s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
9118         (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
9119         (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
9120         (pow((phi-phi_1+deltaphi), 5))))));
9121
9122     s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
9123         (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
9124         (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
9125         (pow((phi-phi_1-deltaphi), 5))))));
9126
9127     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9128
9129     s_strich=((s_H12/(pow((phi_H12*rad), 3)))*(30*(pow(((phi-phi_1)*rad), 2))-
9130         (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad), 3))+
9131         (30/(pow((phi_H12*rad), 2)))*(pow(((phi-phi_1)*rad), 4))));
9132
9133     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9134
9135     s_zweistrich=((s_H12/(pow(phi_H12, 3)))*(60*(phi-phi_1)-((180/phi_H12)*
9136         (pow((phi-phi_1), 2)))+((120/(pow(phi_H12, 2)))*
9137         (pow((phi-phi_1), 3))));
9138 }
9139 else if( index2==2 )
9140 {
9141     /* Bestehornsinoide */
9142
9143     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
9144         ((phi-phi_1)/phi_H12))));
9145
9146     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
9147         (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
9148         phi_H12))));
9149
9150     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-

```

```

9151             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
9152             phi_H12)))));
9153
9154     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
9155
9156     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
9157 }
9158 }
9159 /* dritter Abschnitt */
9160 else if(phi>=phi_2 && phi<phi_3)
9161 {
9162     if( index3==0 )
9163     {
9164         /* Hub in den Bewegungsabschnitten von phi */
9165
9166         s = s_3;
9167         s_plus_deltaphi = s_3;
9168         s_minus_deltaphi = s_3;
9169
9170         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9171
9172         s_strich = 0;
9173
9174         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9175
9176         s_zweistrich = 0;
9177     }
9178     else if( index3==1 )
9179     {
9180         /* Hub in den Bewegungsabschnitten von phi */
9181
9182         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
9183         (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
9184         (pow(phi-phi_2, 5))))));
9185
9186         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
9187         (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
9188         (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
9189         (pow((phi-phi_2+deltaphi), 5))))));
9190
9191         s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
9192         (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
9193         (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
9194         (pow((phi-phi_2-deltaphi), 5))))));

```

```

9195
9196     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9197
9198     s_strich=((s_H23/(pow((phi_H23*rad),3)))*(30*(pow(((phi-phi_2)*rad),2))-
9199             (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad),3))+
9200             (30/(pow((phi_H23*rad),2)))*(pow(((phi-phi_2)*rad),4))));
9201
9202     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9203
9204     s_zweistrich=((s_H23/(pow(phi_H23,3)))*(60*(phi-phi_2)-((180/phi_H23)*
9205             (pow((phi-phi_2),2)))+((120/(pow(phi_H23,2)))*
9206             (pow((phi-phi_2),3)))));
9207 }
9208 else if( index3==2 )
9209 {
9210     /* Bestehornsinoide */
9211
9212     s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
9213             ((phi-phi_2)/phi_H23))));
9214
9215     s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
9216             (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
9217             phi_H23))));
9218
9219     s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
9220             (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
9221             phi_H23))));
9222
9223     s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
9224
9225     s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
9226 }
9227 }
9228 /* vierter Abschnitt */
9229 else if(phi>=phi_3 && phi<phi_4)
9230 {
9231     if( index4==0 )
9232     {
9233         /* Hub in den Bewegungsabschnitten von phi */
9234
9235         s = s_4;
9236         s_plus_deltaphi = s_4;
9237         s_minus_deltaphi = s_4;
9238

```

```

9239      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9240
9241      s_strich = 0;
9242
9243      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9244
9245      s_zweistrich = 0;
9246  }
9247  else if( index4==1 )
9248  {
9249      /* Hub in den Bewegungsabschnitten von phi */
9250
9251      s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
9252          (15/phi_H34)*(pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*
9253          (pow(phi-phi_3, 5))))));
9254
9255      s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
9256          (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
9257          (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
9258          (pow((phi-phi_3+deltaphi), 5))))));
9259
9260      s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
9261          (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
9262          (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
9263          (pow((phi-phi_3-deltaphi), 5))))));
9264
9265      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9266
9267      s_strich=((s_H34/(pow((phi_H34*rad),3)))*(30*(pow(((phi-phi_3)*rad),2))-
9268          (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad),3))+
9269          (30/(pow((phi_H34*rad),2))*(pow(((phi-phi_3)*rad),4)))));
9270
9271      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9272
9273      s_zweistrich=((s_H34/(pow(phi_H34,3)))*(60*(phi-phi_3)-((180/phi_H34)*
9274          (pow((phi-phi_3),2)))+(120/(pow(phi_H34,2))*
9275          (pow((phi-phi_3),3)))));
9276  }
9277  else if( index4==2 )
9278  {
9279      /* Bestehornsinoide */
9280
9281      s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
9282          ((phi-phi_3)/phi_H34))));

```

```

9283
9284     s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
9285                          (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
9286                          phi_H34))));
9287
9288     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
9289                          (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
9290                          phi_H34))));
9291
9292     s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
9293
9294     s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
9295 }
9296 }
9297 /* fuerfter Abschnitt */
9298 else if(phi>=phi_4 && phi<phi_5)
9299 {
9300     if( index5==0 )
9301     {
9302         /* Hub in den Bewegungsabschnitten von phi */
9303
9304         s = s_5;
9305         s_plus_deltaphi = s_5;
9306         s_minus_deltaphi = s_5;
9307
9308         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9309
9310         s_strich = 0;
9311
9312         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9313
9314         s_zweistrich = 0;
9315     }
9316     else if( index5==1 )
9317     {
9318         /* Hub in den Bewegungsabschnitten von phi */
9319
9320         s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-
9321             (15/phi_H45)*(pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2))))*
9322             (pow(phi-phi_4, 5))));
9323
9324         s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
9325             (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
9326             (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2))))*

```

```

9327         (pow((phi-phi_4+deltaphi), 5))));
9328
9329     s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
9330         (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*
9331         (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
9332         (pow((phi-phi_4-deltaphi), 5))));
9333
9334     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9335
9336     s_strich=((s_H45/(pow((phi_H45*rad),3)))*(30*(pow(((phi-phi_4)*rad),2))-
9337         (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad),3))+
9338         (30/(pow((phi_H45*rad),2))*(pow(((phi-phi_4)*rad),4))));
9339
9340     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9341
9342     s_zweistrich=((s_H45/(pow(phi_H45,3)))*(60*(phi-phi_4)-((180/phi_H45)*
9343         (pow((phi-phi_4),2)))+((120/(pow(phi_H45,2)))*
9344         (pow((phi-phi_4),3))));
9345 }
9346 else if( index5==2 )
9347 {
9348     /* Bestehornsinoide */
9349
9350     s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
9351         ((phi-phi_4)/phi_H45))));
9352
9353     s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-
9354         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/
9355         phi_H45))));
9356
9357     s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
9358         (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
9359         phi_H45))));
9360
9361     s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
9362
9363     s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
9364 }
9365 }
9366 /* sechster Abschnitt */
9367 else if(phi>=phi_5 && phi<phi_6)
9368 {
9369     if( index6==0 )
9370     {

```

```

9371      /* Hub in den Bewegungsabschnitten von phi */
9372
9373      s = s_6;
9374      s_plus_deltaphi = s_6;
9375      s_minus_deltaphi = s_6;
9376
9377      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9378
9379      s_strich = 0;
9380
9381      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9382
9383      s_zweistrich = 0;
9384  }
9385  else if( index6==1 )
9386  {
9387      /* Hub in den Bewegungsabschnitten von phi */
9388
9389      s = (s_5+((s_H56/(pow(phi_H56, 3)))*(10*(pow(phi-phi_5, 3))-
9390          (15/phi_H56)*(pow(phi-phi_5, 4))+(6/(pow(phi_H56, 2)))*
9391          (pow(phi-phi_5, 5))))));
9392
9393      s_plus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
9394          (10*(pow((phi-phi_5+deltaphi), 3))-(15/phi_H56)*
9395          (pow((phi-phi_5+deltaphi), 4))+(6/(pow(phi_H56, 2)))*
9396          (pow((phi-phi_5+deltaphi), 5))))));
9397
9398      s_minus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
9399          (10*(pow((phi-phi_5-deltaphi), 3))-(15/phi_H56)*
9400          (pow((phi-phi_5-deltaphi), 4))+(6/(pow(phi_H56, 2)))*
9401          (pow((phi-phi_5-deltaphi), 5))))));
9402
9403      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9404
9405      s_strich=((s_H56/(pow((phi_H56*rad), 3)))*(30*(pow(((phi-phi_5)*rad), 2))-
9406          (60/(phi_H56*rad))*(pow(((phi-phi_5)*rad), 3))+
9407          (30/(pow((phi_H56*rad), 2)))*(pow(((phi-phi_5)*rad), 4))));
9408
9409      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9410
9411      s_zweistrich=((s_H56/(pow(phi_H56, 3)))*(60*(phi-phi_5)-((180/phi_H56)*
9412          (pow((phi-phi_5), 2)))+((120/(pow(phi_H56, 2)))*
9413          (pow((phi-phi_5), 3)))));
9414  }

```

```

9415     else if( index6==2 )
9416     {
9417         /* Bestehornsinoide */
9418
9419         s = s_5+((s_H56/phi_H56)*(phi-phi_5)-(s_H56/(2*pi)*sin(2*pi*
9420             ((phi-phi_5)/phi_H56))));
9421
9422         s_plus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5+deltaphi)-
9423             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5+deltaphi)/
9424                 phi_H56))));
9425
9426         s_minus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5-deltaphi)-
9427             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5-deltaphi)/
9428                 phi_H56))));
9429
9430         s_strich=s_H56*(1-cos(2*pi*((phi-phi_5)/phi_H56)));
9431
9432         s_zweistrich=2*pi*sin(2*pi*((phi-phi_5)/phi_H56));
9433     }
9434 }
9435 /* siebter Abschnitt */
9436 else if(phi>=phi_6 && phi<phi_7)
9437 {
9438     if( index7==0 )
9439     {
9440         /* Hub in den Bewegungsabschnitten von phi */
9441
9442         s = s_7;
9443         s_plus_deltaphi = s_7;
9444         s_minus_deltaphi = s_7;
9445
9446         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9447
9448         s_strich = 0;
9449
9450         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9451
9452         s_zweistrich = 0;
9453     }
9454     else if( index7==1 )
9455     {
9456         /* Hub in den Bewegungsabschnitten von phi */
9457
9458         s = (s_6+((s_H67/(pow(phi_H67, 3)))*(10*(pow(phi-phi_6, 3))-

```



```

9459         (15/phi_H67)*(pow(phi-phi_6, 4))+(6/(pow(phi_H67, 2)))*
9460         (pow(phi-phi_6, 5))));
9461
9462     s_plus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
9463         (10*(pow((phi-phi_6+deltaphi), 3))-(15/phi_H67)*
9464         (pow((phi-phi_6+deltaphi), 4))+(6/(pow(phi_H67, 2)))*
9465         (pow((phi-phi_6+deltaphi), 5))));
9466
9467     s_minus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
9468         (10*(pow((phi-phi_6-deltaphi), 3))-(15/phi_H67)*
9469         (pow((phi-phi_6-deltaphi), 4))+(6/(pow(phi_H67, 2)))*
9470         (pow((phi-phi_6-deltaphi), 5))));
9471
9472     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9473
9474     s_strich=((s_H67/(pow((phi_H67*rad), 3)))*(30*(pow(((phi-phi_6)*rad), 2))-
9475         (60/(phi_H67*rad))*(pow(((phi-phi_6)*rad), 3))+
9476         (30/(pow((phi_H67*rad), 2)))*(pow(((phi-phi_6)*rad), 4))));
9477
9478     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9479
9480     s_zweistrich=((s_H67/(pow(phi_H67, 3)))*(60*(phi-phi_6)-((180/phi_H67)*
9481         (pow((phi-phi_6), 2)))+(120/(pow(phi_H67, 2)))*
9482         (pow((phi-phi_6), 3))));
9483 }
9484 else if( index7==2 )
9485 {
9486     /* Bestehornsinoide */
9487
9488     s = s_6+((s_H67/phi_H67)*(phi-phi_6)-(s_H67/(2*pi)*sin(2*pi*
9489         ((phi-phi_6)/phi_H67))));
9490
9491     s_plus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6+deltaphi)-
9492         (s_H67/(2*pi)*sin(2*pi*((phi-phi_6+deltaphi)/
9493         phi_H67))));
9494
9495     s_minus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6-deltaphi)-
9496         (s_H67/(2*pi)*sin(2*pi*((phi-phi_6-deltaphi)/
9497         phi_H67))));
9498
9499     s_strich=s_H67*(1-cos(2*pi*((phi-phi_6)/phi_H67)));
9500
9501     s_zweistrich=2*pi*sin(2*pi*((phi-phi_6)/phi_H67));
9502 }

```

```

9503     }
9504     /* achter Abschnitt */
9505     else if(phi>=phi_7 && phi<phi_8)
9506     {
9507         if( index8==0 )
9508         {
9509             /* Hub in den Bewegungsabschnitten von phi */
9510
9511             s = s_8;
9512             s_plus_deltaphi = s_8;
9513             s_minus_deltaphi = s_8;
9514
9515             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9516
9517             s_strich = 0;
9518
9519             /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9520
9521             s_zweistrich = 0;
9522         }
9523     else if( index8==1 )
9524     {
9525         /* Hub in den Bewegungsabschnitten von phi */
9526
9527         s = (s_7+((s_H78/(pow(phi_H78, 3)))*(10*(pow(phi-phi_7, 3))-
9528             (15/phi_H78)*(pow(phi-phi_7, 4))+(6/(pow(phi_H78, 2)))*
9529             (pow(phi-phi_7, 5))))));
9530
9531         s_plus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
9532             (10*(pow((phi-phi_7+deltaphi), 3))-(15/phi_H78)*
9533             (pow((phi-phi_7+deltaphi), 4))+(6/(pow(phi_H78, 2)))*
9534             (pow((phi-phi_7+deltaphi), 5))))));
9535
9536         s_minus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
9537             (10*(pow((phi-phi_7-deltaphi), 3))-(15/phi_H78)*
9538             (pow((phi-phi_7-deltaphi), 4))+(6/(pow(phi_H78, 2)))*
9539             (pow((phi-phi_7-deltaphi), 5))))));
9540
9541         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9542
9543         s_strich=((s_H78/(pow((phi_H78*rad), 3)))*(30*(pow(((phi-phi_7)*rad), 2))-
9544             (60/(phi_H78*rad))*(pow(((phi-phi_7)*rad), 3))+
9545             (30/(pow((phi_H78*rad), 2)))*(pow(((phi-phi_7)*rad), 4))));
9546

```

```

9547      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9548
9549      s_zweistrich=((s_H78/(pow(phi_H78,3)))*(60*(phi-phi_7)-((180/phi_H78)*
9550                  (pow((phi-phi_7),2)))+((120/(pow(phi_H78,2)))*
9551                  (pow((phi-phi_7),3)))));
9552  }
9553  else if( index8==2 )
9554  {
9555      /* Bestehornsinoide */
9556
9557      s = s_7+((s_H78/phi_H78)*(phi-phi_7)-(s_H78/(2*pi)*sin(2*pi*
9558          ((phi-phi_7)/phi_H78))));
9559
9560      s_plus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7+deltaphi)-
9561          (s_H78/(2*pi)*sin(2*pi*((phi-phi_7+deltaphi)/
9562              phi_H78))));
9563
9564      s_minus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7-deltaphi)-
9565          (s_H78/(2*pi)*sin(2*pi*((phi-phi_7-deltaphi)/
9566              phi_H78))));
9567
9568      s_strich=s_H78*(1-cos(2*pi*((phi-phi_7)/phi_H78)));
9569
9570      s_zweistrich=2*pi*sin(2*pi*((phi-phi_7)/phi_H78));
9571  }
9572  }
9573  /* neunter Abschnitt */
9574  else if(phi>=phi_8 && phi<phi_9)
9575  {
9576      if( index9==0 )
9577      {
9578          /* Hub in den Bewegungsabschnitten von phi */
9579
9580          s = s_9;
9581          s_plus_deltaphi = s_9;
9582          s_minus_deltaphi = s_9;
9583
9584          /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9585
9586          s_strich = 0;
9587
9588          /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9589
9590          s_zweistrich = 0;

```

```

9591     }
9592     else if( index9==1 )
9593     {
9594         /* Hub in den Bewegungsabschnitten von phi */
9595
9596         s = (s_8+((s_H89/(pow(phi_H89, 3)))*(10*(pow(phi-phi_8, 3))-(15/phi_H89)*
9597             (pow(phi-phi_8, 4))+6/(pow(phi_H89, 2)))*(pow(phi-phi_8, 5)))));
9598
9599         s_plus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
9600             (10*(pow((phi-phi_8+deltaphi), 3))-(15/phi_H89)*
9601             (pow((phi-phi_8+deltaphi), 4))+6/(pow(phi_H89, 2)))*
9602             (pow((phi-phi_8+deltaphi), 5)))));
9603
9604         s_minus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
9605             (10*(pow((phi-phi_8-deltaphi), 3))-(15/phi_H89)*
9606             (pow((phi-phi_8-deltaphi), 4))+6/(pow(phi_H89, 2)))*
9607             (pow((phi-phi_8-deltaphi), 5)))));
9608
9609         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9610
9611         s_strich=((s_H89/(pow((phi_H89*rad), 3)))*(30*(pow(((phi-phi_8)*rad), 2))-
9612             (60/(phi_H89*rad))*(pow(((phi-phi_8)*rad), 3))+
9613             (30/(pow((phi_H89*rad), 2)))*(pow(((phi-phi_8)*rad), 4))));
9614
9615         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9616
9617         s_zweistrich=((s_H89/(pow(phi_H89, 3)))*(60*(phi-phi_8)-((180/phi_H89)*
9618             (pow((phi-phi_8), 2)))+(120/(pow(phi_H89, 2)))*
9619             (pow((phi-phi_8), 3))));
9620     }
9621     else if( index9==2 )
9622     {
9623         /* Bestehornsinoide */
9624
9625         s = s_8+((s_H89/phi_H89)*(phi-phi_8)-(s_H89/(2*pi)*sin(2*pi*
9626             ((phi-phi_8)/phi_H89))));
9627
9628         s_plus_deltaphi = s_8+((s_H89/phi_H89)*(phi-phi_8+deltaphi)-
9629             (s_H89/(2*pi)*sin(2*pi*((phi-phi_8+deltaphi)/
9630             phi_H89))));
9631
9632         s_minus_deltaphi = s_8+((s_H89/phi_H89)*(phi-phi_8-deltaphi)-
9633             (s_H89/(2*pi)*sin(2*pi*((phi-phi_8-deltaphi)/
9634             phi_H89))));

```

```

9635
9636     s_strich=s_H89*(1-cos(2*pi*((phi-phi_8)/phi_H89)));
9637
9638     s_zweistrich=2*pi*sin(2*pi*((phi-phi_8)/phi_H89));
9639 }
9640 }
9641 /* zehnter Abschnitt */
9642 else
9643 {
9644     if( index10==0 )
9645     {
9646         /* Hub in den Bewegungsabschnitten von phi */
9647
9648         s = s_10;
9649         s_plus_deltaphi = s_10;
9650         s_minus_deltaphi = s_10;
9651
9652         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9653
9654         s_strich = 0;
9655
9656         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9657
9658         s_zweistrich = 0;
9659     }
9660     else if( index10==1 )
9661     {
9662         /* Hub in den Bewegungsabschnitten von phi */
9663
9664         s = (s_9+((s_H910/(pow(phi_H910, 3)))*(10*(pow(phi-phi_9, 3))-
9665             (15/phi_H910)*(pow(phi-phi_9, 4))+(6/(pow(phi_H910, 2)))*
9666             (pow(phi-phi_9, 5))))));
9667
9668         s_plus_deltaphi = (s_9+((s_H910/(pow(phi_H910, 3)))*
9669             (10*(pow((phi-phi_9+deltaphi), 3))-(15/phi_H910)*
9670             (pow((phi-phi_9+deltaphi), 4))+(6/(pow(phi_H910, 2)))*
9671             (pow((phi-phi_9+deltaphi), 5))))));
9672
9673         s_minus_deltaphi = (s_9+((s_H910/(pow(phi_H910, 3)))*
9674             (10*(pow((phi-phi_9-deltaphi), 3))-(15/phi_H910)*
9675             (pow((phi-phi_9-deltaphi), 4))+(6/(pow(phi_H910, 2)))*
9676             (pow((phi-phi_9-deltaphi), 5))))));
9677
9678         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */

```

```

9679
9680     s_strich=((s_H910/(pow((phi_H910*rad),3)))*(30*(pow(((phi-phi_9)*rad),2))-
9681             (60/(phi_H910*rad))*(pow(((phi-phi_9)*rad),3))+
9682             (30/(pow((phi_H910*rad),2)))*(pow(((phi-phi_9)*rad),4))));
9683
9684     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9685
9686     s_zweistrich=((s_H910/(pow(phi_H910,3)))*(60*(phi-phi_9)-((180/phi_H910)*
9687             (pow((phi-phi_9),2)))+(120/(pow(phi_H910,2)))*
9688             (pow((phi-phi_9),3))));
9689 }
9690 else if( index10==2 )
9691 {
9692     /* Bestehornsinoide */
9693
9694     s = s_9+((s_H910/phi_H910)*(phi-phi_9)-(s_H910/(2*pi)*sin(2*pi*
9695             ((phi-phi_9)/phi_H910))));
9696
9697     s_plus_deltaphi = s_9+((s_H910/phi_H910)*(phi-phi_9+deltaphi)-
9698             (s_H910/(2*pi)*sin(2*pi*((phi-phi_9+deltaphi)/
9699             phi_H910))));
9700
9701     s_minus_deltaphi = s_9+((s_H910/phi_H910)*(phi-phi_9-deltaphi)-
9702             (s_H910/(2*pi)*sin(2*pi*((phi-phi_9-deltaphi)/
9703             phi_H910))));
9704
9705     s_strich=s_H910*(1-cos(2*pi*((phi-phi_9)/phi_H910)));
9706
9707     s_zweistrich=2*pi*sin(2*pi*((phi-phi_9)/phi_H910));
9708 }
9709 }
9710 }
9711 void Opticurv::berechneHub11()
9712 {
9713     /* erster Abschnitt */
9714     if( phi>=0 && phi<phi_1 )
9715     {
9716         if( index1==0 )
9717         {
9718             /* Hub in den Bewegungsabschnitten von phi */
9719
9720             s = s_1;
9721             s_plus_deltaphi = s_1;
9722             s_minus_deltaphi = s_1;

```

```

9723
9724      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9725
9726      s_strich = 0;
9727
9728      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9729
9730      s_zweistrich = 0;
9731  }
9732  else if( index1==1 )
9733  {
9734      /* Hub in den Bewegungsabschnitten von phi */
9735
9736      s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
9737          (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5))));
9738
9739      s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
9740          (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
9741          (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
9742          (pow((phi+deltaphi), 5))));
9743
9744      s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
9745          (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
9746          (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
9747          (pow((phi-deltaphi), 5))));
9748
9749      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9750
9751      s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
9752          (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
9753          (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4))));
9754
9755      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9756
9757      s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
9758          (pow((phi),2)))+((120/(pow(phi_H01,2)))*
9759          (pow((phi),3)))));
9760  }
9761  else if( index1==2 )
9762  {
9763      /* Bestehornsinoide */
9764
9765      s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
9766

```

```

9767     s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
9768         sin(2*pi*((phi+deltaphi)/phi_H01))));
9769
9770     s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
9771         sin(2*pi*((phi-deltaphi)/phi_H01))));
9772
9773     s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
9774
9775     s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
9776 }
9777 }
9778 /* zweiter Abschnitt */
9779 else if(phi>= phi_1 && phi<phi_2)
9780 {
9781     if( index2==0 )
9782     {
9783         /* Hub in den Bewegungsabschnitten von phi */
9784
9785         s = s_2;
9786         s_plus_deltaphi = s_2;
9787         s_minus_deltaphi = s_2;
9788
9789         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9790
9791         s_strich = 0;
9792
9793         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9794
9795         s_zweistrich = 0;
9796     }
9797     else if( index2==1 )
9798     {
9799         /* Hub in den Bewegungsabschnitten von phi */
9800
9801         s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
9802             (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
9803             (pow(phi-phi_1, 5))))));
9804
9805         s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
9806             (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
9807             (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
9808             (pow((phi-phi_1+deltaphi), 5))))));
9809
9810         s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*

```



```

9811         (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
9812         (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
9813         (pow((phi-phi_1-deltaphi), 5))));
9814
9815     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9816
9817     s_strich=((s_H12/(pow((phi_H12*rad),3)))*(30*(pow(((phi-phi_1)*rad),2))-
9818             (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad),3))+
9819             (30/(pow((phi_H12*rad),2)))*(pow(((phi-phi_1)*rad),4))));
9820
9821     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9822
9823     s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
9824             (pow((phi-phi_1),2)))+((120/(pow(phi_H12,2)))*
9825             (pow((phi-phi_1),3))));
9826 }
9827 else if( index2==2 )
9828 {
9829     /* Bestehornsinoide */
9830
9831     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
9832             ((phi-phi_1)/phi_H12))));
9833
9834     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
9835             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
9836             phi_H12))));
9837
9838     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
9839             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
9840             phi_H12))));
9841
9842     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
9843
9844     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));
9845 }
9846 }
9847 /* dritter Abschnitt */
9848 else if(phi>=phi_2 && phi<phi_3)
9849 {
9850     if( index3==0 )
9851     {
9852         /* Hub in den Bewegungsabschnitten von phi */
9853
9854         s = s_3;

```

```

9855     s_plus_deltaphi = s_3;
9856     s_minus_deltaphi = s_3;
9857
9858     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9859
9860     s_strich = 0;
9861
9862     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9863
9864     s_zweistrich = 0;
9865 }
9866 else if( index3==1 )
9867 {
9868     /* Hub in den Bewegungsabschnitten von phi */
9869
9870     s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
9871         (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
9872         (pow(phi-phi_2, 5))))));
9873
9874     s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
9875         (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
9876         (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
9877         (pow((phi-phi_2+deltaphi), 5))))));
9878
9879     s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
9880         (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
9881         (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
9882         (pow((phi-phi_2-deltaphi), 5))))));
9883
9884     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9885
9886     s_strich=((s_H23/(pow((phi_H23*rad),3)))*(30*(pow(((phi-phi_2)*rad),2))-
9887         (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad),3))+
9888         (30/(pow((phi_H23*rad),2)))*(pow(((phi-phi_2)*rad),4))));
9889
9890     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9891
9892     s_zweistrich=((s_H23/(pow(phi_H23,3)))*(60*(phi-phi_2)-((180/phi_H23)*
9893         (pow((phi-phi_2),2)))+((120/(pow(phi_H23,2)))*
9894         (pow((phi-phi_2),3)))));
9895 }
9896 else if( index3==2 )
9897 {
9898     /* Bestehornsinoide */

```

```

9899
9900     s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
9901         ((phi-phi_2)/phi_H23))));
9902
9903     s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
9904         (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
9905             phi_H23))));
9906
9907     s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
9908         (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
9909             phi_H23))));
9910
9911     s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
9912
9913     s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
9914 }
9915 }
9916 /* vierter Abschnitt */
9917 else if(phi>=phi_3 && phi<phi_4)
9918 {
9919     if( index4==0 )
9920     {
9921         /* Hub in den Bewegungsabschnitten von phi */
9922
9923         s = s_4;
9924         s_plus_deltaphi = s_4;
9925         s_minus_deltaphi = s_4;
9926
9927         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9928
9929         s_strich = 0;
9930
9931         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9932
9933         s_zweistrich = 0;
9934     }
9935     else if( index4==1 )
9936     {
9937         /* Hub in den Bewegungsabschnitten von phi */
9938
9939         s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-
9940             (15/phi_H34)*(pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*
9941             (pow(phi-phi_3, 5))))));
9942

```

```

9943     s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
9944         (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
9945         (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
9946         (pow((phi-phi_3+deltaphi), 5)))));
9947
9948     s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
9949         (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
9950         (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
9951         (pow((phi-phi_3-deltaphi), 5)))));
9952
9953     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9954
9955     s_strich=((s_H34/(pow((phi_H34*rad), 3)))*(30*(pow(((phi-phi_3)*rad), 2))-
9956         (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad), 3))+
9957         (30/(pow((phi_H34*rad), 2)))*(pow(((phi-phi_3)*rad), 4))));
9958
9959     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
9960
9961     s_zweistrich=((s_H34/(pow(phi_H34, 3)))*(60*(phi-phi_3)-((180/phi_H34)*
9962         (pow((phi-phi_3), 2)))+((120/(pow(phi_H34, 2)))*
9963         (pow((phi-phi_3), 3))));
9964 }
9965 else if( index4==2 )
9966 {
9967     /* Bestehornsinoide */
9968
9969     s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
9970         ((phi-phi_3)/phi_H34))));
9971
9972     s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
9973         (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
9974         phi_H34))));
9975
9976     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
9977         (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/
9978         phi_H34))));
9979
9980     s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
9981
9982     s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
9983 }
9984 }
9985 /* fuenfter Abschnitt */
9986 else if(phi>=phi_4 && phi<phi_5)

```

```

9987     {
9988         if( index5==0 )
9989         {
9990             /* Hub in den Bewegungsabschnitten von phi */
9991
9992             s = s_5;
9993             s_plus_deltaphi = s_5;
9994             s_minus_deltaphi = s_5;
9995
9996             /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
9997
9998             s_strich = 0;
9999
10000            /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10001
10002            s_zweistrich = 0;
10003        }
10004        else if( index5==1 )
10005        {
10006            /* Hub in den Bewegungsabschnitten von phi */
10007
10008            s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-(15/phi_H45)*
10009                (pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2)))*(pow(phi-phi_4, 5))))));
10010
10011            s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
10012                (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
10013                (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2)))*
10014                (pow((phi-phi_4+deltaphi), 5))))));
10015
10016            s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
10017                (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*
10018                (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
10019                (pow((phi-phi_4-deltaphi), 5))))));
10020
10021            /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10022
10023            s_strich=((s_H45/(pow((phi_H45*rad),3)))*(30*(pow(((phi-phi_4)*rad),2))-
10024                (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad),3))+
10025                (30/(pow((phi_H45*rad),2)))*(pow(((phi-phi_4)*rad),4))));
10026
10027            /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10028
10029            s_zweistrich=((s_H45/(pow(phi_H45,3)))*(60*(phi-phi_4)-((180/phi_H45)*
10030                (pow((phi-phi_4),2)))+(120/(pow(phi_H45,2)))*

```

```

10031             (pow((phi-phi_4),3))));
10032     }
10033     else if( index5==2 )
10034     {
10035         /* Bestehornsinoide */
10036
10037         s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
10038             ((phi-phi_4)/phi_H45))));
10039
10040         s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-
10041             (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/
10042                 phi_H45))));
10043
10044         s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
10045             (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
10046                 phi_H45))));
10047
10048         s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
10049
10050         s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
10051     }
10052 }
10053 /* sechster Abschnitt */
10054 else if(phi>=phi_5 && phi<phi_6)
10055 {
10056     if( index6==0 )
10057     {
10058         /* Hub in den Bewegungsabschnitten von phi */
10059
10060         s = s_6;
10061         s_plus_deltaphi = s_6;
10062         s_minus_deltaphi = s_6;
10063
10064         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10065
10066         s_strich = 0;
10067
10068         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10069
10070         s_zweistrich = 0;
10071     }
10072     else if( index6==1 )
10073     {
10074         /* Hub in den Bewegungsabschnitten von phi */

```

```

10075
10076 s = (s_5+((s_H56/(pow(phi_H56, 3)))*(10*(pow(phi-phi_5, 3))-
10077 (15/phi_H56)*(pow(phi-phi_5, 4))+(6/(pow(phi_H56, 2)))*
10078 (pow(phi-phi_5, 5)))));
10079
10080 s_plus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
10081 (10*(pow((phi-phi_5+deltaphi), 3))-(15/phi_H56)*
10082 (pow((phi-phi_5+deltaphi), 4))+(6/(pow(phi_H56, 2)))*
10083 (pow((phi-phi_5+deltaphi), 5)))));
10084
10085 s_minus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
10086 (10*(pow((phi-phi_5-deltaphi), 3))-(15/phi_H56)*
10087 (pow((phi-phi_5-deltaphi), 4))+(6/(pow(phi_H56, 2)))*
10088 (pow((phi-phi_5-deltaphi), 5)))));
10089
10090 /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10091
10092 s_strich=((s_H56/(pow((phi_H56*rad), 3)))*(30*(pow(((phi-phi_5)*rad), 2))-
10093 (60/(phi_H56*rad))*(pow(((phi-phi_5)*rad), 3))+
10094 (30/(pow((phi_H56*rad), 2)))*(pow(((phi-phi_5)*rad), 4))));
10095
10096 /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10097
10098 s_zweistrich=((s_H56/(pow(phi_H56, 3)))*(60*(phi-phi_5)-((180/phi_H56)*
10099 (pow((phi-phi_5), 2)))+((120/(pow(phi_H56, 2)))*
10100 (pow((phi-phi_5), 3))));
10101 }
10102 else if( index6==2 )
10103 {
10104 /* Bestehornsinoide */
10105
10106 s = s_5+((s_H56/phi_H56)*(phi-phi_5)-(s_H56/(2*pi)*sin(2*pi*
10107 ((phi-phi_5)/phi_H56))));
10108
10109 s_plus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5+deltaphi)-
10110 (s_H56/(2*pi)*sin(2*pi*((phi-phi_5+deltaphi)/
10111 phi_H56))));
10112
10113 s_minus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5-deltaphi)-
10114 (s_H56/(2*pi)*sin(2*pi*((phi-phi_5-deltaphi)/
10115 phi_H56))));
10116
10117 s_strich=s_H56*(1-cos(2*pi*((phi-phi_5)/phi_H56)));
10118

```

```

10119     s_zweistrich=2*pi*sin(2*pi*((phi-phi_5)/phi_H56));
10120 }
10121 }
10122 /* siebter Abschnitt */
10123 else if(phi>=phi_6 && phi<phi_7)
10124 {
10125     if( index7==0 )
10126     {
10127         /* Hub in den Bewegungsabschnitten von phi */
10128
10129         s = s_7;
10130         s_plus_deltaphi = s_7;
10131         s_minus_deltaphi = s_7;
10132
10133         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10134
10135         s_strich = 0;
10136
10137         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10138
10139         s_zweistrich = 0;
10140     }
10141 else if( index7==1 )
10142 {
10143     /* Hub in den Bewegungsabschnitten von phi */
10144
10145     s = (s_6+((s_H67/(pow(phi_H67, 3)))*(10*(pow(phi-phi_6, 3))-
10146         (15/phi_H67)*(pow(phi-phi_6, 4))+(6/(pow(phi_H67, 2)))*
10147         (pow(phi-phi_6, 5))))));
10148
10149     s_plus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
10150         (10*(pow((phi-phi_6+deltaphi), 3))-(15/phi_H67)*
10151         (pow((phi-phi_6+deltaphi), 4))+(6/(pow(phi_H67, 2)))*
10152         (pow((phi-phi_6+deltaphi), 5))))));
10153
10154     s_minus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
10155         (10*(pow((phi-phi_6-deltaphi), 3))-(15/phi_H67)*
10156         (pow((phi-phi_6-deltaphi), 4))+(6/(pow(phi_H67, 2)))*
10157         (pow((phi-phi_6-deltaphi), 5))))));
10158
10159     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10160
10161     s_strich=((s_H67/(pow((phi_H67*rad),3)))*(30*(pow(((phi-phi_6)*rad),2))-
10162         (60/(phi_H67*rad))*pow(((phi-phi_6)*rad),3))+

```



```

10163             (30/(pow((phi_H67*rad),2)))*(pow(((phi-phi_6)*rad),4))));
10164
10165     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10166
10167     s_zweistrich=((s_H67/(pow(phi_H67,3)))*(60*(phi-phi_6)-((180/phi_H67)*
10168             (pow((phi-phi_6),2)))+((120/(pow(phi_H67,2)))*
10169             (pow((phi-phi_6),3)))));
10170 }
10171 else if( index7==2 )
10172 {
10173     /* Bestehornsinoide */
10174
10175     s = s_6+((s_H67/phi_H67)*(phi-phi_6)-(s_H67/(2*pi)*sin(2*pi*
10176             ((phi-phi_6)/phi_H67))));
10177
10178     s_plus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6+deltaphi)-
10179             (s_H67/(2*pi)*sin(2*pi*((phi-phi_6+deltaphi)/
10180             phi_H67))));
10181
10182     s_minus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6-deltaphi)-
10183             (s_H67/(2*pi)*sin(2*pi*((phi-phi_6-deltaphi)/
10184             phi_H67))));
10185
10186     s_strich=s_H67*(1-cos(2*pi*((phi-phi_6)/phi_H67)));
10187
10188     s_zweistrich=2*pi*sin(2*pi*((phi-phi_6)/phi_H67));
10189 }
10190 }
10191 /* achter Abschnitt */
10192 else if(phi>=phi_7 && phi<phi_8)
10193 {
10194     if( index8==0 )
10195     {
10196         /* Hub in den Bewegungsabschnitten von phi */
10197
10198         s = s_8;
10199         s_plus_deltaphi = s_8;
10200         s_minus_deltaphi = s_8;
10201
10202         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10203
10204         s_strich = 0;
10205
10206         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */

```

```

10207
10208     s_zweistrich = 0;
10209 }
10210 else if( index8==1 )
10211 {
10212     /* Hub in den Bewegungsabschnitten von phi */
10213
10214     s = (s_7+((s_H78/(pow(phi_H78, 3)))*(10*(pow(phi-phi_7, 3))-
10215         (15/phi_H78)*(pow(phi-phi_7, 4))+6/(pow(phi_H78, 2)))*
10216         (pow(phi-phi_7, 5)))));
10217
10218     s_plus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
10219         (10*(pow((phi-phi_7+deltaphi), 3))-(15/phi_H78)*
10220         (pow((phi-phi_7+deltaphi), 4))+6/(pow(phi_H78, 2)))*
10221         (pow((phi-phi_7+deltaphi), 5)))));
10222
10223     s_minus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
10224         (10*(pow((phi-phi_7-deltaphi), 3))-(15/phi_H78)*
10225         (pow((phi-phi_7-deltaphi), 4))+6/(pow(phi_H78, 2)))*
10226         (pow((phi-phi_7-deltaphi), 5)))));
10227
10228     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10229
10230     s_strich=((s_H78/(pow((phi_H78*rad),3)))*(30*(pow(((phi-phi_7)*rad),2))-
10231         (60/(phi_H78*rad))*(pow(((phi-phi_7)*rad),3))+
10232         (30/(pow((phi_H78*rad),2)))*(pow(((phi-phi_7)*rad),4))));
10233
10234     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10235
10236     s_zweistrich=((s_H78/(pow(phi_H78,3)))*(60*(phi-phi_7)-((180/phi_H78)*
10237         (pow((phi-phi_7),2)))+((120/(pow(phi_H78,2)))*
10238         (pow((phi-phi_7),3))));
10239 }
10240 else if( index8==2 )
10241 {
10242     /* Bestehornsinoide */
10243
10244     s = s_7+((s_H78/phi_H78)*(phi-phi_7)-(s_H78/(2*pi)*sin(2*pi*
10245         ((phi-phi_7)/phi_H78))));
10246
10247     s_plus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7+deltaphi)-
10248         (s_H78/(2*pi)*sin(2*pi*((phi-phi_7+deltaphi)/
10249         phi_H78))));
10250

```

```

10251     s_minus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7-deltaphi)-
10252                               (s_H78/(2*pi)*sin(2*pi*((phi-phi_7-deltaphi)/
10253                               phi_H78))));
10254
10255     s_strich=s_H78*(1-cos(2*pi*((phi-phi_7)/phi_H78)));
10256
10257     s_zweistrich=2*pi*sin(2*pi*((phi-phi_7)/phi_H78));
10258 }
10259 }
10260 /* neunter Abschnitt */
10261 else if(phi>=phi_8 && phi<phi_9)
10262 {
10263     if( index9==0 )
10264     {
10265         /* Hub in den Bewegungsabschnitten von phi */
10266
10267         s = s_9;
10268         s_plus_deltaphi = s_9;
10269         s_minus_deltaphi = s_9;
10270
10271         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10272
10273         s_strich = 0;
10274
10275         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10276
10277         s_zweistrich = 0;
10278     }
10279     else if( index9==1 )
10280     {
10281         /* Hub in den Bewegungsabschnitten von phi */
10282
10283         s = (s_8+((s_H89/(pow(phi_H89, 3)))*(10*(pow(phi-phi_8, 3))-
10284             (15/phi_H89)*(pow(phi-phi_8, 4))+(6/(pow(phi_H89, 2)))*
10285             (pow(phi-phi_8, 5))))));
10286
10287         s_plus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
10288             (10*(pow((phi-phi_8+deltaphi), 3))-(15/phi_H89)*
10289             (pow((phi-phi_8+deltaphi), 4))+(6/(pow(phi_H89, 2)))*
10290             (pow((phi-phi_8+deltaphi), 5))))));
10291
10292         s_minus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
10293             (10*(pow((phi-phi_8-deltaphi), 3))-(15/phi_H89)*
10294             (pow((phi-phi_8-deltaphi), 4))+(6/(pow(phi_H89, 2)))*

```

```

10295         (pow((phi-phi_8-deltaphi), 5))));
10296
10297     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10298
10299     s_strich=((s_H89/(pow((phi_H89*rad),3)))*(30*(pow(((phi-phi_8)*rad),2))-
10300         (60/(phi_H89*rad))*(pow(((phi-phi_8)*rad),3))+
10301         (30/(pow((phi_H89*rad),2)))*(pow(((phi-phi_8)*rad),4))));
10302
10303     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10304
10305     s_zweistrich=((s_H89/(pow(phi_H89,3)))*(60*(phi-phi_8)-((180/phi_H89)*
10306         (pow((phi-phi_8),2)))+((120/(pow(phi_H89,2)))*
10307         (pow((phi-phi_8),3))));
10308 }
10309 else if( index9==2 )
10310 {
10311     /* Bestehornsinoide */
10312
10313     s = s_8+((s_H89/phi_H89)*(phi-phi_8)-(s_H89/(2*pi)*sin(2*pi*
10314         ((phi-phi_8)/phi_H89))));
10315
10316     s_plus_deltaphi = s_8+((s_H89/phi_H89)*(phi-phi_8+deltaphi)-
10317         (s_H89/(2*pi)*sin(2*pi*((phi-phi_8+deltaphi)/
10318         phi_H89))));
10319
10320     s_minus_deltaphi = s_8+((s_H89/phi_H89)*(phi-phi_8-deltaphi)-
10321         (s_H89/(2*pi)*sin(2*pi*((phi-phi_8-deltaphi)/
10322         phi_H89))));
10323
10324     s_strich=s_H89*(1-cos(2*pi*((phi-phi_8)/phi_H89)));
10325
10326     s_zweistrich=2*pi*sin(2*pi*((phi-phi_8)/phi_H89));
10327 }
10328 }
10329 /* zehnter Abschnitt */
10330 else if(phi>=phi_9 && phi<phi_10)
10331 {
10332     if( index10==0 )
10333     {
10334         /* Hub in den Bewegungsabschnitten von phi */
10335
10336         s = s_10;
10337         s_plus_deltaphi = s_10;
10338         s_minus_deltaphi = s_10;

```

```

10339
10340      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10341
10342      s_strich = 0;
10343
10344      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10345
10346      s_zweistrich = 0;
10347  }
10348  else if( index10==1 )
10349  {
10350      /* Hub in den Bewegungsabschnitten von phi */
10351
10352      s = (s_9+((s_H910/(pow(phi_H910, 3)))*(10*(pow(phi-phi_9, 3))-
10353          (15/phi_H910)*(pow(phi-phi_9, 4))+(6/(pow(phi_H910, 2)))*
10354          (pow(phi-phi_9, 5))))));
10355
10356      s_plus_deltaphi = (s_9+((s_H910/(pow(phi_H910, 3)))*
10357          (10*(pow((phi-phi_9+deltaphi), 3))-(15/phi_H910)*
10358          (pow((phi-phi_9+deltaphi), 4))+(6/(pow(phi_H910, 2)))*
10359          (pow((phi-phi_9+deltaphi), 5))))));
10360
10361      s_minus_deltaphi = (s_9+((s_H910/(pow(phi_H910, 3)))*
10362          (10*(pow((phi-phi_9-deltaphi), 3))-(15/phi_H910)*
10363          (pow((phi-phi_9-deltaphi), 4))+(6/(pow(phi_H910, 2)))*
10364          (pow((phi-phi_9-deltaphi), 5))))));
10365
10366      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10367
10368      s_strich=((s_H910/(pow((phi_H910*rad),3)))*(30*(pow(((phi-phi_9)*rad),2))-
10369          (60/(phi_H910*rad))*(pow(((phi-phi_9)*rad),3))+
10370          (30/(pow((phi_H910*rad),2)))*(pow(((phi-phi_9)*rad),4))));
10371
10372      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10373
10374      s_zweistrich=((s_H910/(pow(phi_H910,3)))*(60*(phi-phi_9)-((180/phi_H910)*
10375          (pow((phi-phi_9),2)))+((120/(pow(phi_H910,2)))*
10376          (pow((phi-phi_9),3)))));
10377  }
10378  else if( index10==2 )
10379  {
10380      /* Bestehornsinoide */
10381
10382      s = s_9+((s_H910/phi_H910)*(phi-phi_9)-(s_H910/(2*pi)*sin(2*pi*

```

```

10383         ((phi-phi_9)/phi_H910)))));
10384
10385     s_plus_deltaphi = s_9+((s_H910/phi_H910)*(phi-phi_9+deltaphi)-
10386         (s_H910/(2*pi)*sin(2*pi*((phi-phi_9+deltaphi)/
10387             phi_H910))));
10388
10389     s_minus_deltaphi = s_9+((s_H910/phi_H910)*(phi-phi_9-deltaphi)-
10390         (s_H910/(2*pi)*sin(2*pi*((phi-phi_9-deltaphi)/
10391             phi_H910))));
10392
10393     s_strich=s_H910*(1-cos(2*pi*((phi-phi_9)/phi_H910)));
10394
10395     s_zweistrich=2*pi*sin(2*pi*((phi-phi_9)/phi_H910));
10396 }
10397 }
10398 /* elfter Abschnitt */
10399 else
10400 {
10401     if( index11==0 )
10402     {
10403         /* Hub in den Bewegungsabschnitten von phi */
10404
10405         s = s_11;
10406         s_plus_deltaphi = s_11;
10407         s_minus_deltaphi = s_11;
10408
10409         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10410
10411         s_strich = 0;
10412
10413         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10414
10415         s_zweistrich = 0;
10416     }
10417     else if( index11==1 )
10418     {
10419         /* Hub in den Bewegungsabschnitten von phi */
10420
10421         s = (s_10+((s_H1011/(pow(phi_H1011, 3)))*(10*(pow(phi-phi_10, 3))-
10422             (15/phi_H1011)*(pow(phi-phi_10, 4))+(6/(pow(phi_H1011, 2)))*
10423             (pow(phi-phi_10, 5)))));
10424
10425         s_plus_deltaphi = (s_10+((s_H1011/(pow(phi_H1011, 3)))*
10426             (10*(pow((phi-phi_10+deltaphi), 3))-(15/phi_H1011)*

```

```

10427         (pow((phi-phi_10+deltaphi), 4))+(6/(pow(phi_H1011, 2)))*
10428         (pow((phi-phi_10+deltaphi), 5))));
10429
10430     s_minus_deltaphi = (s_10+((s_H1011/(pow(phi_H1011, 3)))*
10431         (10*(pow((phi-phi_10-deltaphi), 3))-(15/phi_H1011)*
10432         (pow((phi-phi_10-deltaphi), 4))+(6/(pow(phi_H1011, 2)))*
10433         (pow((phi-phi_10-deltaphi), 5))));
10434
10435     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10436
10437     s_strich=((s_H1011/(pow((phi_H1011*rad),3)))*(30*
10438         (pow(((phi-phi_10)*rad),2))-(60/(phi_H1011*rad))*
10439         (pow(((phi-phi_10)*rad),3))+(30/(pow((phi_H1011*rad),2)))*
10440         (pow(((phi-phi_10)*rad),4))));
10441
10442     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10443
10444     s_zweistrich=((s_H1011/(pow(phi_H1011,3)))*(60*(phi-phi_10)-
10445         ((180/phi_H1011)*(pow((phi-phi_10),2)))+
10446         ((120/(pow(phi_H1011,2)))*(pow((phi-phi_10),3))));
10447 }
10448 else if( index11==2 )
10449 {
10450     /* Bestehornsinoide */
10451
10452     s = s_10+((s_H1011/phi_H1011)*(phi-phi_10)-(s_H1011/(2*pi)*sin(2*pi*
10453         ((phi-phi_10)/phi_H1011))));
10454
10455     s_plus_deltaphi = s_10+((s_H1011/phi_H1011)*(phi-phi_10+deltaphi)-
10456         (s_H1011/(2*pi)*sin(2*pi*((phi-phi_10+deltaphi)/
10457         phi_H1011))));
10458
10459     s_minus_deltaphi = s_10+((s_H1011/phi_H1011)*(phi-phi_10-deltaphi)-
10460         (s_H1011/(2*pi)*sin(2*pi*((phi-phi_10-deltaphi)/
10461         phi_H1011))));
10462
10463     s_strich=s_H1011*(1-cos(2*pi*((phi-phi_10)/phi_H1011)));
10464
10465     s_zweistrich=2*pi*sin(2*pi*((phi-phi_10)/phi_H1011));
10466 }
10467 }
10468 }
10469 void Opticurv::berechneHub12()
10470 {

```

```

10471      /* erster Abschnitt */
10472      if( phi>=0 && phi<phi_1 )
10473      {
10474          if( index1==0 )
10475          {
10476              /* Hub in den Bewegungsabschnitten von phi */
10477
10478              s = s_1;
10479              s_plus_deltaphi = s_1;
10480              s_minus_deltaphi = s_1;
10481
10482              /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10483
10484              s_strich = 0;
10485
10486              /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10487
10488              s_zweistrich = 0;
10489          }
10490      else if( index1==1 )
10491      {
10492          /* Hub in den Bewegungsabschnitten von phi */
10493
10494          s = ((s_H01/(pow(phi_H01, 3)))*(10*(pow(phi, 3))-(15/phi_H01)*
10495              (pow(phi, 4))+(6/(pow(phi_H01, 2)))*(pow(phi, 5)))));
10496
10497          s_plus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
10498              (10*(pow((phi+deltaphi), 3))-(15/phi_H01)*
10499              (pow((phi+deltaphi), 4))+(6/(pow(phi_H01, 2)))*
10500              (pow((phi+deltaphi), 5)))));
10501
10502          s_minus_deltaphi = ((s_H01/(pow(phi_H01, 3)))*
10503              (10*(pow((phi-deltaphi), 3))-(15/phi_H01)*
10504              (pow((phi-deltaphi), 4))+(6/(pow(phi_H01, 2)))*
10505              (pow((phi-deltaphi), 5)))));
10506
10507          /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10508
10509          s_strich=((s_H01/(pow((phi_H01*rad),3)))*(30*(pow(((phi)*rad),2))-
10510              (60/(phi_H01*rad))*(pow(((phi)*rad),3))+
10511              (30/(pow((phi_H01*rad),2)))*(pow(((phi)*rad),4)))));
10512
10513          /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10514

```



```

10515         s_zweistrich=((s_H01/(pow(phi_H01,3)))*(60*(phi)-((180/phi_H01)*
10516                     (pow((phi),2)))+((120/(pow(phi_H01,2)))*
10517                     (pow((phi),3)))));
10518     }
10519     else if( index1==2 )
10520     {
10521         /* Bestehornsinoide */
10522
10523         s = ((s_H01/phi_H01)*phi-(s_H01/(2*pi)*sin(2*pi*(phi/phi_H01))));
10524
10525         s_plus_deltaphi = ((s_H01/phi_H01)*(phi+deltaphi)-(s_H01/(2*pi)*
10526                     sin(2*pi*((phi+deltaphi)/phi_H01))));
10527
10528         s_minus_deltaphi = ((s_H01/phi_H01)*(phi-deltaphi)-(s_H01/(2*pi)*
10529                     sin(2*pi*((phi-deltaphi)/phi_H01))));
10530
10531         s_strich=s_H01*(1-cos(2*pi*((phi)/phi_H01)));
10532
10533         s_zweistrich=2*pi*sin(2*pi*((phi)/phi_H01));
10534     }
10535 }
10536 /* zweiter Abschnitt */
10537 else if(phi>= phi_1 && phi<phi_2)
10538 {
10539     if( index2==0 )
10540     {
10541         /* Hub in den Bewegungsabschnitten von phi */
10542
10543         s = s_2;
10544         s_plus_deltaphi = s_2;
10545         s_minus_deltaphi = s_2;
10546
10547         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10548
10549         s_strich = 0;
10550
10551         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10552
10553         s_zweistrich = 0;
10554     }
10555     else if( index2==1 )
10556     {
10557         /* Hub in den Bewegungsabschnitten von phi */
10558

```

```

10559     s = (s_1+((s_H12/(pow(phi_H12, 3)))*(10*(pow(phi-phi_1, 3))-
10560             (15/phi_H12)*(pow(phi-phi_1, 4))+(6/(pow(phi_H12, 2)))*
10561             (pow(phi-phi_1, 5)))));
10562
10563     s_plus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
10564             (10*(pow((phi-phi_1+deltaphi), 3))-(15/phi_H12)*
10565             (pow((phi-phi_1+deltaphi), 4))+(6/(pow(phi_H12, 2)))*
10566             (pow((phi-phi_1+deltaphi), 5)))));
10567
10568     s_minus_deltaphi = (s_1+((s_H12/(pow(phi_H12, 3)))*
10569             (10*(pow((phi-phi_1-deltaphi), 3))-(15/phi_H12)*
10570             (pow((phi-phi_1-deltaphi), 4))+(6/(pow(phi_H12, 2)))*
10571             (pow((phi-phi_1-deltaphi), 5)))));
10572
10573     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10574
10575     s_strich=((s_H12/(pow((phi_H12*rad),3)))*(30*(pow(((phi-phi_1)*rad),2))-
10576             (60/(phi_H12*rad))*(pow(((phi-phi_1)*rad),3))+
10577             (30/(pow((phi_H12*rad),2))*(pow(((phi-phi_1)*rad),4)))));
10578
10579     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10580
10581     s_zweistrich=((s_H12/(pow(phi_H12,3)))*(60*(phi-phi_1)-((180/phi_H12)*
10582             (pow((phi-phi_1),2)))+(120/(pow(phi_H12,2))*
10583             (pow((phi-phi_1),3))));
10584 }
10585 else if( index2==2 )
10586 {
10587     /* Bestehornsinoide */
10588
10589     s = s_1+((s_H12/phi_H12)*(phi-phi_1)-(s_H12/(2*pi)*sin(2*pi*
10590             ((phi-phi_1)/phi_H12))));
10591
10592     s_plus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1+deltaphi)-
10593             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1+deltaphi)/
10594             phi_H12))));
10595
10596     s_minus_deltaphi = s_1+((s_H12/phi_H12)*(phi-phi_1-deltaphi)-
10597             (s_H12/(2*pi)*sin(2*pi*((phi-phi_1-deltaphi)/
10598             phi_H12))));
10599
10600     s_strich=s_H12*(1-cos(2*pi*((phi-phi_1)/phi_H12)));
10601
10602     s_zweistrich=2*pi*sin(2*pi*((phi-phi_1)/phi_H12));

```

```

10603     }
10604 }
10605 /* dritter Abschnitt */
10606 else if(phi>=phi_2 && phi<phi_3)
10607 {
10608     if( index3==0 )
10609     {
10610         /* Hub in den Bewegungsabschnitten von phi */
10611
10612         s = s_3;
10613         s_plus_deltaphi = s_3;
10614         s_minus_deltaphi = s_3;
10615
10616         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10617
10618         s_strich = 0;
10619
10620         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10621
10622         s_zweistrich = 0;
10623     }
10624     else if( index3==1 )
10625     {
10626         /* Hub in den Bewegungsabschnitten von phi */
10627
10628         s = (s_2+((s_H23/(pow(phi_H23, 3)))*(10*(pow(phi-phi_2, 3))-
10629             (15/phi_H23)*(pow(phi-phi_2, 4))+(6/(pow(phi_H23, 2)))*
10630             (pow(phi-phi_2, 5))))));
10631
10632         s_plus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
10633             (10*(pow((phi-phi_2+deltaphi), 3))-(15/phi_H23)*
10634             (pow((phi-phi_2+deltaphi), 4))+(6/(pow(phi_H23, 2)))*
10635             (pow((phi-phi_2+deltaphi), 5))))));
10636
10637         s_minus_deltaphi = (s_2+((s_H23/(pow(phi_H23, 3)))*
10638             (10*(pow((phi-phi_2-deltaphi), 3))-(15/phi_H23)*
10639             (pow((phi-phi_2-deltaphi), 4))+(6/(pow(phi_H23, 2)))*
10640             (pow((phi-phi_2-deltaphi), 5))))));
10641
10642         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10643
10644         s_strich=((s_H23/(pow((phi_H23*rad),3)))*(30*(pow(((phi-phi_2)*rad),2))-
10645             (60/(phi_H23*rad))*(pow(((phi-phi_2)*rad),3))+
10646             (30/(pow((phi_H23*rad),2)))*(pow(((phi-phi_2)*rad),4))));

```

```

10647
10648      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10649
10650      s_zweistrich=((s_H23/(pow(phi_H23,3)))*(60*(phi-phi_2)-((180/phi_H23)*
10651          (pow((phi-phi_2),2)))+((120/(pow(phi_H23,2)))*
10652          (pow((phi-phi_2),3)))));
10653  }
10654  else if( index3==2 )
10655  {
10656      /* Bestehornsinoide */
10657
10658      s = s_2+((s_H23/phi_H23)*(phi-phi_2)-(s_H23/(2*pi)*sin(2*pi*
10659          ((phi-phi_2)/phi_H23))));
10660
10661      s_plus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2+deltaphi)-
10662          (s_H23/(2*pi)*sin(2*pi*((phi-phi_2+deltaphi)/
10663          phi_H23))));
10664
10665      s_minus_deltaphi = s_2+((s_H23/phi_H23)*(phi-phi_2-deltaphi)-
10666          (s_H23/(2*pi)*sin(2*pi*((phi-phi_2-deltaphi)/
10667          phi_H23))));
10668
10669      s_strich=s_H23*(1-cos(2*pi*((phi-phi_2)/phi_H23)));
10670
10671      s_zweistrich=2*pi*sin(2*pi*((phi-phi_2)/phi_H23));
10672  }
10673 }
10674 /* vierter Abschnitt */
10675 else if(phi>=phi_3 && phi<phi_4)
10676 {
10677     if( index4==0 )
10678     {
10679         /* Hub in den Bewegungsabschnitten von phi */
10680
10681         s = s_4;
10682         s_plus_deltaphi = s_4;
10683         s_minus_deltaphi = s_4;
10684
10685         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10686
10687         s_strich = 0;
10688
10689         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10690

```

```

10691     s_zweistrich = 0;
10692 }
10693 else if( index4==1 )
10694 {
10695     /* Hub in den Bewegungsabschnitten von phi */
10696
10697     s = (s_3+((s_H34/(pow(phi_H34, 3)))*(10*(pow(phi-phi_3, 3))-(15/phi_H34)*
10698         (pow(phi-phi_3, 4))+(6/(pow(phi_H34, 2)))*(pow(phi-phi_3, 5))))));
10699
10700     s_plus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
10701         (10*(pow((phi-phi_3+deltaphi), 3))-(15/phi_H34)*
10702         (pow((phi-phi_3+deltaphi), 4))+(6/(pow(phi_H34, 2)))*
10703         (pow((phi-phi_3+deltaphi), 5))))));
10704
10705     s_minus_deltaphi = (s_3+((s_H34/(pow(phi_H34, 3)))*
10706         (10*(pow((phi-phi_3-deltaphi), 3))-(15/phi_H34)*
10707         (pow((phi-phi_3-deltaphi), 4))+(6/(pow(phi_H34, 2)))*
10708         (pow((phi-phi_3-deltaphi), 5))))));
10709
10710     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10711
10712     s_strich=((s_H34/(pow((phi_H34*rad), 3)))*(30*(pow(((phi-phi_3)*rad), 2))-
10713         (60/(phi_H34*rad))*(pow(((phi-phi_3)*rad), 3))+
10714         (30/(pow((phi_H34*rad), 2)))*(pow(((phi-phi_3)*rad), 4))));
10715
10716     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10717
10718     s_zweistrich=((s_H34/(pow(phi_H34, 3)))*(60*(phi-phi_3)-((180/phi_H34)*
10719         (pow((phi-phi_3), 2)))+((120/(pow(phi_H34, 2)))*
10720         (pow((phi-phi_3), 3))));
10721 }
10722 else if( index4==2 )
10723 {
10724     /* Bestehornsinoide */
10725
10726     s = s_3+((s_H34/phi_H34)*(phi-phi_3)-(s_H34/(2*pi)*sin(2*pi*
10727         ((phi-phi_3)/phi_H34))));
10728
10729     s_plus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3+deltaphi)-
10730         (s_H34/(2*pi)*sin(2*pi*((phi-phi_3+deltaphi)/
10731         phi_H34))));
10732
10733     s_minus_deltaphi = s_3+((s_H34/phi_H34)*(phi-phi_3-deltaphi)-
10734         (s_H34/(2*pi)*sin(2*pi*((phi-phi_3-deltaphi)/

```

```

10735             phi_H34))));
10736
10737         s_strich=s_H34*(1-cos(2*pi*((phi-phi_3)/phi_H34)));
10738
10739         s_zweistrich=2*pi*sin(2*pi*((phi-phi_3)/phi_H34));
10740     }
10741 }
10742 /* fuerfter Abschnitt */
10743 else if(phi>=phi_4 && phi<phi_5)
10744 {
10745     if( index5==0 )
10746     {
10747         /* Hub in den Bewegungsabschnitten von phi */
10748
10749         s = s_5;
10750         s_plus_deltaphi = s_5;
10751         s_minus_deltaphi = s_5;
10752
10753         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10754
10755         s_strich = 0;
10756
10757         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10758
10759         s_zweistrich = 0;
10760     }
10761 else if( index5==1 )
10762 {
10763     /* Hub in den Bewegungsabschnitten von phi */
10764
10765     s = (s_4+((s_H45/(pow(phi_H45, 3)))*(10*(pow(phi-phi_4, 3))-
10766         (15/phi_H45)*(pow(phi-phi_4, 4))+(6/(pow(phi_H45, 2)))*
10767         (pow(phi-phi_4, 5))))));
10768
10769     s_plus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
10770         (10*(pow((phi-phi_4+deltaphi), 3))-(15/phi_H45)*
10771         (pow((phi-phi_4+deltaphi), 4))+(6/(pow(phi_H45, 2)))*
10772         (pow((phi-phi_4+deltaphi), 5))))));
10773
10774     s_minus_deltaphi = (s_4+((s_H45/(pow(phi_H45, 3)))*
10775         (10*(pow((phi-phi_4-deltaphi), 3))-(15/phi_H45)*
10776         (pow((phi-phi_4-deltaphi), 4))+(6/(pow(phi_H45, 2)))*
10777         (pow((phi-phi_4-deltaphi), 5))))));
10778

```

```

10779      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10780
10781      s_strich=((s_H45/(pow((phi_H45*rad),3)))*(30*(pow(((phi-phi_4)*rad),2))-
10782              (60/(phi_H45*rad))*(pow(((phi-phi_4)*rad),3))+
10783              (30/(pow((phi_H45*rad),2)))*(pow(((phi-phi_4)*rad),4))));
10784
10785      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10786
10787      s_zweistrich=((s_H45/(pow(phi_H45,3)))*(60*(phi-phi_4)-((180/phi_H45)*
10788              (pow((phi-phi_4),2)))+((120/(pow(phi_H45,2)))*
10789              (pow((phi-phi_4),3)))));
10790  }
10791  else if( index5==2 )
10792  {
10793      /* Bestehornsinoide */
10794
10795      s = s_4+((s_H45/phi_H45)*(phi-phi_4)-(s_H45/(2*pi)*sin(2*pi*
10796              ((phi-phi_4)/phi_H45))));
10797
10798      s_plus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4+deltaphi)-
10799              (s_H45/(2*pi)*sin(2*pi*((phi-phi_4+deltaphi)/
10800              phi_H45))));
10801
10802      s_minus_deltaphi = s_4+((s_H45/phi_H45)*(phi-phi_4-deltaphi)-
10803              (s_H45/(2*pi)*sin(2*pi*((phi-phi_4-deltaphi)/
10804              phi_H45))));
10805
10806      s_strich=s_H45*(1-cos(2*pi*((phi-phi_4)/phi_H45)));
10807
10808      s_zweistrich=2*pi*sin(2*pi*((phi-phi_4)/phi_H45));
10809  }
10810 }
10811 /* sechster Abschnitt */
10812 else if(phi>=phi_5 && phi<phi_6)
10813 {
10814     if( index6==0 )
10815     {
10816         /* Hub in den Bewegungsabschnitten von phi */
10817
10818         s = s_6;
10819         s_plus_deltaphi = s_6;
10820         s_minus_deltaphi = s_6;
10821
10822         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */

```

```

10823
10824     s_strich = 0;
10825
10826     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10827
10828     s_zweistrich = 0;
10829 }
10830 else if( index6==1 )
10831 {
10832     /* Hub in den Bewegungsabschnitten von phi */
10833
10834     s = (s_5+((s_H56/(pow(phi_H56, 3)))*(10*(pow(phi-phi_5, 3))-
10835         (15/phi_H56)*(pow(phi-phi_5, 4))+(6/(pow(phi_H56, 2)))*
10836         (pow(phi-phi_5, 5))))));
10837
10838     s_plus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
10839         (10*(pow((phi-phi_5+deltaphi), 3))-(15/phi_H56)*
10840         (pow((phi-phi_5+deltaphi), 4))+(6/(pow(phi_H56, 2)))*
10841         (pow((phi-phi_5+deltaphi), 5))))));
10842
10843     s_minus_deltaphi = (s_5+((s_H56/(pow(phi_H56, 3)))*
10844         (10*(pow((phi-phi_5-deltaphi), 3))-(15/phi_H56)*
10845         (pow((phi-phi_5-deltaphi), 4))+(6/(pow(phi_H56, 2)))*
10846         (pow((phi-phi_5-deltaphi), 5))))));
10847
10848     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10849
10850     s_strich=((s_H56/(pow((phi_H56*rad), 3)))*(30*(pow(((phi-phi_5)*rad), 2))-
10851         (60/(phi_H56*rad))*(pow(((phi-phi_5)*rad), 3))+
10852         (30/(pow((phi_H56*rad), 2)))*(pow(((phi-phi_5)*rad), 4))));
10853
10854     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10855
10856     s_zweistrich=((s_H56/(pow(phi_H56, 3)))*(60*(phi-phi_5)-((180/phi_H56)*
10857         (pow((phi-phi_5), 2)))+((120/(pow(phi_H56, 2)))*
10858         (pow((phi-phi_5), 3))));
10859 }
10860 else if( index6==2 )
10861 {
10862     /* Bestehornsinoide */
10863
10864     s = s_5+((s_H56/phi_H56)*(phi-phi_5)-(s_H56/(2*pi)*sin(2*pi*
10865         ((phi-phi_5)/phi_H56))));
10866

```



```

10867     s_plus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5+deltaphi)-
10868                             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5+deltaphi)/
10869                             phi_H56))));
10870
10871     s_minus_deltaphi = s_5+((s_H56/phi_H56)*(phi-phi_5-deltaphi)-
10872                             (s_H56/(2*pi)*sin(2*pi*((phi-phi_5-deltaphi)/
10873                             phi_H56))));
10874
10875     s_strich=s_H56*(1-cos(2*pi*((phi-phi_5)/phi_H56)));
10876
10877     s_zweistrich=2*pi*sin(2*pi*((phi-phi_5)/phi_H56));
10878 }
10879 }
10880 /* siebter Abschnitt */
10881 else if(phi>=phi_6 && phi<phi_7)
10882 {
10883     if( index7==0 )
10884     {
10885         /* Hub in den Bewegungsabschnitten von phi */
10886
10887         s = s_7;
10888         s_plus_deltaphi = s_7;
10889         s_minus_deltaphi = s_7;
10890
10891         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10892
10893         s_strich = 0;
10894
10895         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10896
10897         s_zweistrich = 0;
10898     }
10899 else if( index7==1 )
10900 {
10901     /* Hub in den Bewegungsabschnitten von phi */
10902
10903     s = (s_6+((s_H67/(pow(phi_H67, 3)))*(10*(pow(phi-phi_6, 3))-
10904         (15/phi_H67)*(pow(phi-phi_6, 4))+(6/(pow(phi_H67, 2)))*
10905         (pow(phi-phi_6, 5))))));
10906
10907     s_plus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
10908         (10*(pow((phi-phi_6+deltaphi), 3))-(15/phi_H67)*
10909         (pow((phi-phi_6+deltaphi), 4))+(6/(pow(phi_H67, 2)))*
10910         (pow((phi-phi_6+deltaphi), 5))))));

```

```

10911
10912     s_minus_deltaphi = (s_6+((s_H67/(pow(phi_H67, 3)))*
10913         (10*(pow((phi-phi_6-deltaphi), 3))-(15/phi_H67)*
10914         (pow((phi-phi_6-deltaphi), 4))+(6/(pow(phi_H67, 2)))*
10915         (pow((phi-phi_6-deltaphi), 5)))));
10916
10917     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10918
10919     s_strich=((s_H67/(pow((phi_H67*rad), 3)))*(30*(pow(((phi-phi_6)*rad), 2))-
10920         (60/(phi_H67*rad))*(pow(((phi-phi_6)*rad), 3))+
10921         (30/(pow((phi_H67*rad), 2))*(pow(((phi-phi_6)*rad), 4)))));
10922
10923     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10924
10925     s_zweistrich=((s_H67/(pow(phi_H67, 3)))*(60*(phi-phi_6)-((180/phi_H67)*
10926         (pow((phi-phi_6), 2)))+(120/(pow(phi_H67, 2))*(
10927         (pow((phi-phi_6), 3)))));
10928 }
10929 else if( index7==2 )
10930 {
10931     /* Bestehornsinoide */
10932
10933     s = s_6+((s_H67/phi_H67)*(phi-phi_6)-(s_H67/(2*pi)*sin(2*pi*
10934         ((phi-phi_6)/phi_H67))));
10935
10936     s_plus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6+deltaphi)-
10937         (s_H67/(2*pi)*sin(2*pi*((phi-phi_6+deltaphi)/
10938         phi_H67))));
10939
10940     s_minus_deltaphi = s_6+((s_H67/phi_H67)*(phi-phi_6-deltaphi)-
10941         (s_H67/(2*pi)*sin(2*pi*((phi-phi_6-deltaphi)/
10942         phi_H67))));
10943
10944     s_strich=s_H67*(1-cos(2*pi*((phi-phi_6)/phi_H67)));
10945
10946     s_zweistrich=2*pi*sin(2*pi*((phi-phi_6)/phi_H67));
10947 }
10948 }
10949 /* achter Abschnitt */
10950 else if(phi>=phi_7 && phi<phi_8)
10951 {
10952     if( index8==0 )
10953     {
10954         /* Hub in den Bewegungsabschnitten von phi */

```

```

10955
10956     s = s_8;
10957     s_plus_deltaphi = s_8;
10958     s_minus_deltaphi = s_8;
10959
10960     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10961
10962     s_strich = 0;
10963
10964     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10965
10966     s_zweistrich = 0;
10967 }
10968 else if( index8==1 )
10969 {
10970     /* Hub in den Bewegungsabschnitten von phi */
10971
10972     s = (s_7+((s_H78/(pow(phi_H78, 3)))*(10*(pow(phi-phi_7, 3))-
10973         (15/phi_H78)*(pow(phi-phi_7, 4))+(6/(pow(phi_H78, 2)))*
10974         (pow(phi-phi_7, 5))))));
10975
10976     s_plus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
10977         (10*(pow((phi-phi_7+deltaphi), 3))-(15/phi_H78)*
10978         (pow((phi-phi_7+deltaphi), 4))+(6/(pow(phi_H78, 2)))*
10979         (pow((phi-phi_7+deltaphi), 5))))));
10980
10981     s_minus_deltaphi = (s_7+((s_H78/(pow(phi_H78, 3)))*
10982         (10*(pow((phi-phi_7-deltaphi), 3))-(15/phi_H78)*
10983         (pow((phi-phi_7-deltaphi), 4))+(6/(pow(phi_H78, 2)))*
10984         (pow((phi-phi_7-deltaphi), 5))))));
10985
10986     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
10987
10988     s_strich=((s_H78/(pow((phi_H78*rad),3)))*(30*(pow(((phi-phi_7)*rad),2))-
10989         (60/(phi_H78*rad))*(pow(((phi-phi_7)*rad),3))+
10990         (30/(pow((phi_H78*rad),2)))*(pow(((phi-phi_7)*rad),4))));
10991
10992     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
10993
10994     s_zweistrich=((s_H78/(pow(phi_H78,3)))*(60*(phi-phi_7)-((180/phi_H78)*
10995         (pow((phi-phi_7),2)))+((120/(pow(phi_H78,2)))*
10996         (pow((phi-phi_7),3))));
10997 }
10998 else if( index8==2 )

```

```

10999 {
11000     /* Bestehornsinoide */
11001
11002     s = s_7+((s_H78/phi_H78)*(phi-phi_7)-(s_H78/(2*pi)*sin(2*pi*
11003         ((phi-phi_7)/phi_H78))));
11004
11005     s_plus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7+deltaphi)-
11006         (s_H78/(2*pi)*sin(2*pi*((phi-phi_7+deltaphi)/
11007             phi_H78))));
11008
11009     s_minus_deltaphi = s_7+((s_H78/phi_H78)*(phi-phi_7-deltaphi)-
11010         (s_H78/(2*pi)*sin(2*pi*((phi-phi_7-deltaphi)/
11011             phi_H78))));
11012
11013     s_strich=s_H78*(1-cos(2*pi*((phi-phi_7)/phi_H78)));
11014
11015     s_zweistrich=2*pi*sin(2*pi*((phi-phi_7)/phi_H78));
11016 }
11017 }
11018 /* neunter Abschnitt */
11019 else if(phi>=phi_8 && phi<phi_9)
11020 {
11021     if( index9==0 )
11022     {
11023         /* Hub in den Bewegungsabschnitten von phi */
11024
11025         s = s_9;
11026         s_plus_deltaphi = s_9;
11027         s_minus_deltaphi = s_9;
11028
11029         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11030
11031         s_strich = 0;
11032
11033         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
11034
11035         s_zweistrich = 0;
11036     }
11037     else if( index9==1 )
11038     {
11039         /* Hub in den Bewegungsabschnitten von phi */
11040
11041         s = (s_8+((s_H89/(pow(phi_H89, 3)))*(10*(pow(phi-phi_8, 3))-
11042             (15/phi_H89)*(pow(phi-phi_8, 4))+(6/(pow(phi_H89, 2)))*

```

```

11043         (pow(phi-phi_8, 5))));
11044
11045     s_plus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
11046         (10*(pow((phi-phi_8+deltaphi), 3))-(15/phi_H89)*
11047         (pow((phi-phi_8+deltaphi), 4))+(6/(pow(phi_H89, 2)))*
11048         (pow((phi-phi_8+deltaphi), 5))));
11049
11050     s_minus_deltaphi = (s_8+((s_H89/(pow(phi_H89, 3)))*
11051         (10*(pow((phi-phi_8-deltaphi), 3))-(15/phi_H89)*
11052         (pow((phi-phi_8-deltaphi), 4))+(6/(pow(phi_H89, 2)))*
11053         (pow((phi-phi_8-deltaphi), 5))));
11054
11055     /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11056
11057     s_strich=((s_H89/(pow((phi_H89*rad), 3)))*(30*(pow(((phi-phi_8)*rad), 2))-
11058         (60/(phi_H89*rad))*(pow(((phi-phi_8)*rad), 3))+
11059         (30/(pow((phi_H89*rad), 2)))*(pow(((phi-phi_8)*rad), 4))));
11060
11061     /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
11062
11063     s_zweistrich=((s_H89/(pow(phi_H89, 3)))*(60*(phi-phi_8)-((180/phi_H89)*
11064         (pow((phi-phi_8), 2)))+((120/(pow(phi_H89, 2)))*
11065         (pow((phi-phi_8), 3))));
11066 }
11067 else if( index9==2 )
11068 {
11069     /* Bestehornsinoide */
11070
11071     s = s_8+((s_H89/phi_H89)*(phi-phi_8)-(s_H89/(2*pi)*sin(2*pi*
11072         ((phi-phi_8)/phi_H89))));
11073
11074     s_plus_deltaphi =s_8+((s_H89/phi_H89)*(phi-phi_8+deltaphi)-
11075         (s_H89/(2*pi)*sin(2*pi*((phi-phi_8+deltaphi)/
11076         phi_H89))));
11077
11078     s_minus_deltaphi = s_8+((s_H89/phi_H89)*(phi-phi_8-deltaphi)-
11079         (s_H89/(2*pi)*sin(2*pi*((phi-phi_8-deltaphi)/
11080         phi_H89))));
11081
11082     s_strich=s_H89*(1-cos(2*pi*((phi-phi_8)/phi_H89)));
11083
11084     s_zweistrich=2*pi*sin(2*pi*((phi-phi_8)/phi_H89));
11085 }
11086 }

```

```

11087      /* zehnter Abschnitt */
11088      else if(phi>=phi_9 && phi<phi_10)
11089      {
11090          if( index10==0 )
11091          {
11092              /* Hub in den Bewegungsabschnitten von phi */
11093
11094              s = s_10;
11095              s_plus_deltaphi = s_10;
11096              s_minus_deltaphi = s_10;
11097
11098              /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11099
11100              s_strich = 0;
11101
11102              /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
11103
11104              s_zweistrich = 0;
11105          }
11106      else if( index10==1 )
11107      {
11108          /* Hub in den Bewegungsabschnitten von phi */
11109
11110          s = (s_9+((s_H910/(pow(phi_H910, 3)))*(10*(pow(phi-phi_9, 3))-
11111              (15/phi_H910)*(pow(phi-phi_9, 4))+(6/(pow(phi_H910, 2)))*
11112              (pow(phi-phi_9, 5))))));
11113
11114          s_plus_deltaphi = (s_9+((s_H910/(pow(phi_H910, 3)))*
11115              (10*(pow((phi-phi_9+deltaphi), 3))-(15/phi_H910)*
11116              (pow((phi-phi_9+deltaphi), 4))+(6/(pow(phi_H910, 2)))*
11117              (pow((phi-phi_9+deltaphi), 5))))));
11118
11119          s_minus_deltaphi = (s_9+((s_H910/(pow(phi_H910, 3)))*
11120              (10*(pow((phi-phi_9-deltaphi), 3))-(15/phi_H910)*
11121              (pow((phi-phi_9-deltaphi), 4))+(6/(pow(phi_H910, 2)))*
11122              (pow((phi-phi_9-deltaphi), 5))))));
11123
11124          /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11125
11126          s_strich=((s_H910/(pow((phi_H910*rad), 3)))*(30*(pow(((phi-phi_9)*rad), 2))-
11127              (60/(phi_H910*rad))*(pow(((phi-phi_9)*rad), 3))+
11128              (30/(pow((phi_H910*rad), 2)))*(pow(((phi-phi_9)*rad), 4))));
11129
11130          /* Hubbeschleunigung in den Bewegungsabschnitten von phi */

```

```

11131
11132     s_zweistrich=((s_H910/(pow(phi_H910,3)))*(60*(phi-phi_9)-((180/phi_H910)*
11133         (pow((phi-phi_9),2)))+(120/(pow(phi_H910,2))*
11134         (pow((phi-phi_9),3)))));
11135 }
11136 else if( index10==2 )
11137 {
11138     /* Bestehornsinoide */
11139
11140     s = s_9+((s_H910/phi_H910)*(phi-phi_9)-(s_H910/(2*pi)*sin(2*pi*
11141         ((phi-phi_9)/phi_H910))));
11142
11143     s_plus_deltaphi = s_9+((s_H910/phi_H910)*(phi-phi_9+deltaphi)-
11144         (s_H910/(2*pi)*sin(2*pi*((phi-phi_9+deltaphi)/
11145         phi_H910))));
11146
11147     s_minus_deltaphi = s_9+((s_H910/phi_H910)*(phi-phi_9-deltaphi)-
11148         (s_H910/(2*pi)*sin(2*pi*((phi-phi_9-deltaphi)/
11149         phi_H910))));
11150
11151     s_strich=s_H910*(1-cos(2*pi*((phi-phi_9)/phi_H910)));
11152
11153     s_zweistrich=2*pi*sin(2*pi*((phi-phi_9)/phi_H910));
11154 }
11155 }
11156 /* elfter Abschnitt */
11157 else if(phi>=phi_10 && phi<phi_11)
11158 {
11159     if( index11==0 )
11160     {
11161         /* Hub in den Bewegungsabschnitten von phi */
11162
11163         s = s_11;
11164         s_plus_deltaphi = s_11;
11165         s_minus_deltaphi = s_11;
11166
11167         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11168
11169         s_strich = 0;
11170
11171         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
11172
11173         s_zweistrich = 0;
11174     }

```

```

11175     else if( index11==1 )
11176     {
11177         /* Hub in den Bewegungsabschnitten von phi */
11178
11179         s = (s_10+((s_H1011/(pow(phi_H1011, 3)))*(10*(pow(phi-phi_10, 3))-
11180             (15/phi_H1011)*(pow(phi-phi_10, 4))+(6/(pow(phi_H1011, 2)))*
11181             (pow(phi-phi_10, 5)))));
11182
11183         s_plus_deltaphi = (s_10+((s_H1011/(pow(phi_H1011, 3)))*
11184             (10*(pow((phi-phi_10+deltaphi), 3))-(15/phi_H1011)*
11185             (pow((phi-phi_10+deltaphi), 4))+(6/(pow(phi_H1011, 2)))*
11186             (pow((phi-phi_10+deltaphi), 5)))));
11187
11188         s_minus_deltaphi = (s_10+((s_H1011/(pow(phi_H1011, 3)))*
11189             (10*(pow((phi-phi_10-deltaphi), 3))-(15/phi_H1011)*
11190             (pow((phi-phi_10-deltaphi), 4))+(6/(pow(phi_H1011, 2)))*
11191             (pow((phi-phi_10-deltaphi), 5)))));
11192
11193         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11194
11195         s_strich=((s_H1011/(pow((phi_H1011*rad),3)))*(30*
11196             (pow(((phi-phi_10)*rad),2))-(60/(phi_H1011*rad)))*
11197             (pow(((phi-phi_10)*rad),3))+(30/(pow((phi_H1011*rad),2)))*
11198             (pow(((phi-phi_10)*rad),4))));
11199
11200         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
11201
11202         s_zweistrich=((s_H1011/(pow(phi_H1011,3)))*(60*(phi-phi_10)-
11203             ((180/phi_H1011)*(pow((phi-phi_10),2)))+(120/
11204             (pow(phi_H1011,2)))*(pow((phi-phi_10),3))));
11205     }
11206     else if( index11==2 )
11207     {
11208         /* Bestehornsinoide */
11209
11210         s = s_10+((s_H1011/phi_H1011)*(phi-phi_10)-(s_H1011/(2*pi)*sin(2*pi*
11211             ((phi-phi_10)/phi_H1011))));
11212
11213         s_plus_deltaphi = s_10+((s_H1011/phi_H1011)*(phi-phi_10+deltaphi)-
11214             (s_H1011/(2*pi)*sin(2*pi*((phi-phi_10+deltaphi)/
11215             phi_H1011))));
11216
11217         s_minus_deltaphi = s_10+((s_H1011/phi_H1011)*(phi-phi_10-deltaphi)-
11218             (s_H1011/(2*pi)*sin(2*pi*((phi-phi_10-deltaphi)/

```



```

11219             phi_H1011)))));
11220
11221         s_strich=s_H1011*(1-cos(2*pi*((phi-phi_10)/phi_H1011)));
11222
11223         s_zweistrich=2*pi*sin(2*pi*((phi-phi_10)/phi_H1011));
11224     }
11225 }
11226 /* zwoelfter Abschnitt */
11227 else
11228 {
11229     if( index12==0 )
11230     {
11231         /* Hub in den Bewegungsabschnitten von phi */
11232
11233         s = s_12;
11234         s_plus_deltaphi = s_12;
11235         s_minus_deltaphi = s_12;
11236
11237         /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11238
11239         s_strich = 0;
11240
11241         /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
11242
11243         s_zweistrich = 0;
11244     }
11245     else if( index12==1 )
11246     {
11247         /* Hub in den Bewegungsabschnitten von phi */
11248
11249         s = (s_11+((s_H1112/(pow(phi_H1112, 3)))*(10*(pow(phi-phi_11, 3))-
11250             (15/phi_H1112)*(pow(phi-phi_11, 4))+(6/(pow(phi_H1112, 2)))*
11251             (pow(phi-phi_11, 5))))));
11252
11253         s_plus_deltaphi = (s_11+((s_H1112/(pow(phi_H1112, 3)))*
11254             (10*(pow((phi-phi_11+deltaphi), 3))-(15/phi_H1112)*
11255             (pow((phi-phi_11+deltaphi), 4))+(6/(pow(phi_H1112, 2)))*
11256             (pow((phi-phi_11+deltaphi), 5))))));
11257
11258         s_minus_deltaphi = (s_11+((s_H1112/(pow(phi_H1112, 3)))*
11259             (10*(pow((phi-phi_11-deltaphi), 3))-(15/phi_H1112)*
11260             (pow((phi-phi_11-deltaphi), 4))+(6/(pow(phi_H1112, 2)))*
11261             (pow((phi-phi_11-deltaphi), 5))))));
11262

```

```

11263      /* Hubgeschwindigkeit in den Bewegungsabschnitten von phi */
11264
11265      s_strich=((s_H1112/(pow((phi_H1112*rad),3)))*(30*
11266              (pow(((phi-phi_11)*rad),2))-(60/(phi_H1112*rad))*
11267              (pow(((phi-phi_11)*rad),3)))+(30/(pow((phi_H1112*rad),2)))*
11268              (pow(((phi-phi_11)*rad),4))));
11269
11270      /* Hubbeschleunigung in den Bewegungsabschnitten von phi */
11271
11272      s_zweistrich=((s_H1112/(pow(phi_H1112,3)))*(60*(phi-phi_11)-
11273              ((180/phi_H1112)*(pow((phi-phi_11),2)))+((120/
11274              (pow(phi_H1112,2)))*(pow((phi-phi_11),3)))));
11275  }
11276  else if( index12==2 )
11277  {
11278      /* Bestehornsinoide */
11279
11280      s = s_11+((s_H1112/phi_H1112)*(phi-phi_11)-(s_H1112/(2*pi)*sin(2*pi*
11281              ((phi-phi_11)/phi_H1112))));
11282
11283      s_plus_deltaphi = s_11+((s_H1112/phi_H1112)*(phi-phi_11+deltaphi)-
11284              (s_H1112/(2*pi)*sin(2*pi*((phi-phi_11+deltaphi)/
11285              phi_H1112))));
11286
11287      s_minus_deltaphi = s_11+((s_H1112/phi_H1112)*(phi-phi_11-deltaphi)-
11288              (s_H1112/(2*pi)*sin(2*pi*((phi-phi_11-deltaphi)/
11289              phi_H1112))));
11290
11291      s_strich=s_H1112*(1-cos(2*pi*((phi-phi_11)/phi_H1112)));
11292
11293      s_zweistrich=2*pi*sin(2*pi*((phi-phi_11)/phi_H1112));
11294  }
11295  }
11296  }
11297
11298
11299  void Opticurv::berechneHubaufrufen()
11300  {
11301      if(indexAbschnitt == 0)
11302      {
11303          berechneHub1();
11304      }
11305      else if(indexAbschnitt == 1)
11306      {

```

```

11307     berechneHub2();
11308 }
11309 else if(indexAbschnitt == 2)
11310 {
11311     berechneHub3();
11312 }
11313 else if(indexAbschnitt == 3)
11314 {
11315     berechneHub4();
11316 }
11317 else if(indexAbschnitt == 4)
11318 {
11319     berechneHub5();
11320 }
11321 else if(indexAbschnitt == 5)
11322 {
11323     berechneHub6();
11324 }
11325 else if(indexAbschnitt == 6)
11326 {
11327     berechneHub7();
11328 }
11329 else if(indexAbschnitt == 7)
11330 {
11331     berechneHub8();
11332 }
11333 else if(indexAbschnitt == 8)
11334 {
11335     berechneHub9();
11336 }
11337 else if(indexAbschnitt == 9)
11338 {
11339     berechneHub10();
11340 }
11341 else if(indexAbschnitt == 10)
11342 {
11343     berechneHub11();
11344 }
11345 else if(indexAbschnitt == 11)
11346 {
11347     berechneHub12();
11348 }
11349 }
11350

```

```

11351 void Opticurv::calculate()
11352 {
11353     statusBar()->message( "Berechne Koordinaten...", 2000 );
11354
11355     ausgabemenu->setItemEnabled( AnzeigenID, TRUE );
11356
11357     rad = (pi/180);
11358
11359     /* implizite Typkonvertierungen */
11360
11361     phi_H01 = dphi_H01;
11362     phi_H12 = dphi_H12;
11363     phi_H23 = dphi_H23;
11364     phi_H34 = dphi_H34;
11365     phi_H45 = dphi_H45;
11366     phi_H56 = dphi_H56;
11367     phi_H67 = dphi_H67;
11368     phi_H78 = dphi_H78;
11369     phi_H89 = dphi_H89;
11370     phi_H910 = dphi_H910;
11371     phi_H1011 = dphi_H1011;
11372     phi_H1112 = dphi_H1112;
11373     s_H01 = ds_H01;
11374     s_H12 = ds_H12;
11375     s_H23 = ds_H23;
11376     s_H34 = ds_H34;
11377     s_H45 = ds_H45;
11378     s_H56 = ds_H56;
11379     s_H67 = ds_H67;
11380     s_H78 = ds_H78;
11381     s_H89 = ds_H89;
11382     s_H910 = ds_H910;
11383     s_H1011 = ds_H1011;
11384     s_H1112 = ds_H1112;
11385     l_3 = dl_3;
11386     x_s0 = dx_s0;
11387     y_s0 = dy_s0;
11388     l_sk = dl_sk;
11389     l_4 = dl_4;
11390     x_H = dx_H;
11391     y_H = dy_H;
11392     r_R = dr_R;
11393     x_A0 = dx_A0;
11394     y_A0 = dy_A0;

```

```

11395     beta = dbeta;
11396     n_1 = dn_1;
11397     n_2 = dn_2;
11398     phi_R1fr = dphi_R1fr;
11399     phi_R2fr = dphi_R2fr;
11400     ergebnis = dergebnis;
11401     r_G = dr_G;
11402     phi = dphi;
11403     r_S = dr_S;
11404     r_W = dr_W;
11405     alpha_R = dalpha_R;
11406     l_3star = dl_3star;
11407     r_Rstar = dr_Rstar;
11408
11409     alpha = ((atan(y_A0/x_A0))/rad);
11410
11411     static const char* text_Grad[] = {" Grad",0};
11412     static const char* text_mm[] = {" mm",0};
11413
11414     text = "";
11415     texttwo = "";
11416
11417     gamma = ((asin(y_H/l_3))/rad);
11418
11419
11420     /* Berechnung der Gestelllaenge l_1 */
11421
11422     l_1 = (sqrt((pow(x_A0,2))+(pow(y_A0,2))));
11423
11424     /* Bezeichnungen */
11425
11426     phi_1 = phi_H01;
11427     phi_2 = phi_H01 + phi_H12;
11428     phi_3 = phi_H01 + phi_H12 + phi_H23;
11429     phi_4 = phi_H01 + phi_H12 + phi_H23 + phi_H34;
11430     phi_5 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45;
11431     phi_6 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45 + phi_H56;
11432     phi_7 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45 + phi_H56 +
11433         phi_H67;
11434     phi_8 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45 + phi_H56 +
11435         phi_H67 + phi_H78;
11436     phi_9 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45 + phi_H56 +
11437         phi_H67 + phi_H78 + phi_H89;
11438     phi_10 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45 + phi_H56 +

```

```

11439         phi_H67 + phi_H78 + phi_H89 + phi_H910;
11440     phi_11 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45 + phi_H56 +
11441         phi_H67 + phi_H78 + phi_H89 + phi_H910 + phi_H1011;
11442     phi_12 = phi_H01 + phi_H12 + phi_H23 + phi_H34 + phi_H45 + phi_H56 +
11443         phi_H67 + phi_H78 + phi_H89 + phi_H910 + phi_H1011 + phi_H1112;
11444
11445     s_1 = s_H01;
11446     s_2 = s_H01 + s_H12;
11447     s_3 = s_H01 + s_H12 + s_H23;
11448     s_4 = s_H01 + s_H12 + s_H23 + s_H34;
11449     s_5 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45;
11450     s_6 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45 + s_H56;
11451     s_7 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45 + s_H56 + s_H67;
11452     s_8 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45 + s_H56 + s_H67 + s_H78;
11453     s_9 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45 + s_H56 + s_H67 + s_H78 +
11454         s_H89;
11455     s_10 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45 + s_H56 + s_H67 + s_H78 +
11456         s_H89 + s_H910;
11457     s_11 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45 + s_H56 + s_H67 + s_H78 +
11458         s_H89 + s_H910 + s_H1011;
11459     s_12 = s_H01 + s_H12 + s_H23 + s_H34 + s_H45 + s_H56 + s_H67 + s_H78 +
11460         s_H89 + s_H910 + s_H1011 + s_H1112;
11461
11462     /* Berechnung von x_s0 */
11463
11464     y_s0_orig = y_s0;
11465
11466     if(x_s0 >= 0 && y_s0 < 0)
11467     {
11468         y_s0 = (-1)*y_s0;
11469     }
11470
11471     /* implizite Typkonvertierungen */
11472
11473     dphi_1 = phi_1;
11474     dphi_2 = phi_2;
11475     dphi_3 = phi_3;
11476     dphi_4 = phi_4;
11477     dphi_5 = phi_5;
11478     dphi_6 = phi_6;
11479     dphi_7 = phi_7;
11480     dphi_8 = phi_8;
11481     dphi_9 = phi_9;
11482     dphi_10 = phi_10;

```

```

11483     dphi_11 = phi_11;
11484     dphi_12 = phi_12;
11485     dalpha = alpha;
11486     dgamma = gamma;
11487     dx_s0 = x_s0;
11488     dl_1 = l_1;
11489     dy_s0 = y_s0;
11490     dy_s0_orig = y_s0_orig;
11491
11492     /* Vektorfelder fuer grafische Ausgabe anlegen */
11493
11494     index_zuweisung = 0;
11495     for(phi=0; phi<360; phi+=n_1)
11496         index_zuweisung++;
11497     int i;
11498     i = ((index_zuweisung + 5 ));
11499     index_zuweisung = 0;
11500
11501     phigrafisch = (long double *) malloc(i * sizeof(long double));
11502     x_B_verstzt = (long double *) malloc(i * sizeof(long double));
11503     y_B_verstzt = (long double *) malloc(i * sizeof(long double));
11504     x_i_verstzt = (long double *) malloc(i * sizeof(long double));
11505     y_i_verstzt = (long double *) malloc(i * sizeof(long double));
11506     x_a_verstzt = (long double *) malloc(i * sizeof(long double));
11507     y_a_verstzt = (long double *) malloc(i * sizeof(long double));
11508     sgrafisch = (long double *) malloc(i * sizeof(long double));
11509     s_strichgrafisch = (long double *) malloc(i * sizeof(long double));
11510     psigrafisch = (long double *) malloc(i * sizeof(long double));
11511     psi_strichgrafisch = (long double *) malloc(i * sizeof(long double));
11512     psi_zweistrichgrafisch = (long double *) malloc(i * sizeof(long double));
11513     x_Bikgrafisch = (long double *) malloc(i * sizeof(long double));
11514     y_Bikgrafisch = (long double *) malloc(i * sizeof(long double));
11515     x_Bik_strichgrafisch = (long double *) malloc(i * sizeof(long double));
11516     y_Bik_strichgrafisch = (long double *) malloc(i * sizeof(long double));
11517     x_sgrafisch = (long double *) malloc(i * sizeof(long double));
11518     y_sgrafisch = (long double *) malloc(i * sizeof(long double));
11519     x_Wgrafisch = (long double *) malloc(i * sizeof(long double));
11520     y_Wgrafisch = (long double *) malloc(i * sizeof(long double));
11521     muegrafisch = (long double *) malloc(i * sizeof(long double));
11522     r_Kgrafisch = (long double *) malloc(i * sizeof(long double));
11523
11524     if(indexKurve==1)      // nur bei Doppelkurve
11525     {
11526         x_Bikstargrafisch = (long double *) malloc(i * sizeof(long double));

```

```

11527     y_Bikstargrafisch = (long double *) malloc(i * sizeof(long double));
11528     x_Bik_strichstargrafisch = (long double *) malloc(i * sizeof(long double));
11529     y_Bik_strichstargrafisch = (long double *) malloc(i * sizeof(long double));
11530     psistargrafisch = (long double *) malloc(i * sizeof(long double));
11531     psistar_radgrafisch = (long double *) malloc(i * sizeof(long double));
11532     x_B_versetztstargrafisch = (long double *) malloc(i * sizeof(long double));
11533     y_B_versetztstargrafisch = (long double *) malloc(i * sizeof(long double));
11534     x_i_versetztstargrafisch = (long double *) malloc(i * sizeof(long double));
11535     y_i_versetztstargrafisch = (long double *) malloc(i * sizeof(long double));
11536     x_a_versetztstargrafisch = (long double *) malloc(i * sizeof(long double));
11537     y_a_versetztstargrafisch = (long double *) malloc(i * sizeof(long double));
11538 }
11539
11540 /* Debugging */
11541
11542 QString version = "Opticurv Version 2.7.3 - TU DRESDEN";
11543 datum = QDate::currentDate();
11544 zeit = QTime::currentTime();
11545 dt = filename + "\n" + version + " - " + datum.toString() +
11546     " - " + zeit.toString();
11547 QString datetime = "<br>" + version + "<br>" +
11548     datum.toString() + " - " + zeit.toString() +
11549     "<br>" + filename + "<br><br>";
11550
11551 qDebug( dt );
11552 qDebug( "phi_1 = %Lg; phi_2 = %Lg; phi_3 = %Lg; phi_4 = %Lg; phi_5 = %Lg",
11553     phi_1, phi_2, phi_3, phi_4, phi_5);
11554 qDebug( "alpha = %Lg; gamma = %Lg; x_s0 = %Lg; l_1 = %Lg; y_s0 = %Lg",
11555     alpha, gamma, x_s0, l_1, y_s0);
11556
11557 QString erg_ausgabe_phi_1=anzeige->text();
11558 erg_ausgabe_phi_1 = "<br><br> phi_1 = " +
11559     erg_ausgabe_phi_1.setNum( dphi_1 ) + " Grad";
11560
11561 QString erg_ausgabe_phi_2=anzeige->text();
11562 erg_ausgabe_phi_2 = "<br> phi_2 = " + erg_ausgabe_phi_2.setNum( dphi_2 ) +
11563     " Grad";
11564
11565 QString erg_ausgabe_phi_3=anzeige->text();
11566 erg_ausgabe_phi_3 = "<br> phi_3 = " + erg_ausgabe_phi_3.setNum( dphi_3 ) +
11567     " Grad";
11568
11569 QString erg_ausgabe_phi_4=anzeige->text();
11570 erg_ausgabe_phi_4 = "<br> phi_4 = " + erg_ausgabe_phi_4.setNum( dphi_4 ) +

```



```

11571         " Grad";
11572
11573     QString erg_ausgabe_phi_5=anzeige->text();
11574     erg_ausgabe_phi_5 = "<br> phi_5 = " + erg_ausgabe_phi_5.setNum( dphi_5 ) +
11575         " Grad";
11576
11577     QString erg_ausgabe_alpha=anzeige->text();
11578     erg_ausgabe_alpha = "<br> alpha = " + erg_ausgabe_alpha.setNum( dalpha ) +
11579         " Grad";
11580
11581     QString erg_ausgabe_gamma=anzeige->text();
11582     erg_ausgabe_gamma = "<br> gamma = " + erg_ausgabe_gamma.setNum( dgamma ) +
11583         " Grad";
11584
11585     QString erg_ausgabe_x_s0=anzeige->text();
11586     erg_ausgabe_x_s0 = "<br> x_s0 = " + erg_ausgabe_x_s0.setNum( dx_s0 ) +
11587         " mm";
11588
11589     QString erg_ausgabe_l_1=anzeige->text();
11590     erg_ausgabe_l_1 = "<br> l_1 = " + erg_ausgabe_l_1.setNum( dl_1 ) +
11591         " mm";
11592
11593     QString erg_ausgabe_y_s0=anzeige->text();
11594     erg_ausgabe_y_s0 = "<br> y_s0 = " + erg_ausgabe_y_s0.setNum( dy_s0_orig ) +
11595         " mm";
11596     static const char* parameterkopf[]={"<html><head></head>"
11597         "<body><br><b><h3>EINGABEPARAMETER</h3></b><p>",0};
11598
11599     static const char* bewegungsverlauf[]={"<h4>Bewegungsverlauf</h4><p>",0};
11600
11601     texttwo+=parameterkopf[0] + datetime + bewegungsverlauf[0];
11602
11603     QString anzahlderAbschnitte, bewegungsverlauf, bewegungsGesetz1,
11604         bewegungsGesetz2, bewegungsGesetz3, bewegungsGesetz4, bewegungsGesetz5,
11605         bewegungsGesetz6, bewegungsGesetz7, bewegungsGesetz8, bewegungsGesetz9,
11606         bewegungsGesetz10, bewegungsGesetz11, bewegungsGesetz12,
11607         geometrieDaten, vorzeichenDefinitionen;
11608
11609     anzahlderAbschnitte = "Anzahl der Bewegungsabschnitte: ";
11610
11611     bewegungsverlauf="<TABLE border=\"0\" cellspacing=\"0\" cellpadding=\"4\" >"
11612         "<TR><TD>1. Abschnitt</TD>"
11613         "<TD>:</TD><TD>";
11614

```

```

11615     if( indexAbschnitt == 0 )
11616     {
11617         anzahlderAbschnitte+="1<p>";
11618
11619         if( index1 == 0 )
11620         {
11621             bewegungsGesetz1 = "Rast";
11622         }
11623         else if( index1 == 1 )
11624         {
11625             bewegungsGesetz1 = "3-4-5 Polynom";
11626         }
11627         else if( index1 == 2 )
11628         {
11629             bewegungsGesetz1 = "Bestehornsinoide";
11630         }
11631         QString erg_dphi_H01=anzeige->text();
11632         erg_dphi_H01.setNum( dphi_H01 );
11633
11634         QString erg_ds_H01=anzeige->text();
11635         erg_ds_H01.setNum( ds_H01 );
11636
11637         bewegungsVerlauf+= bewegungsGesetz1 +
11638             "</TD><TD><font size=\"5\">⌘124;</TD><TD>"
11639             "phi_H01</TD><TD>=</TD><TD>" +
11640             erg_dphi_H01 + "</TD><TD><font size=\"5\">⌘124;</TD>"
11641             "<TD>s_H01</TD><TD>=</TD><TD>" +
11642             erg_ds_H01 + "</TD></TR>";
11643     }
11644     if( indexAbschnitt == 1 )
11645     {
11646         anzahlderAbschnitte+="2<p>";
11647
11648         if( index1 == 0 )
11649         {
11650             bewegungsGesetz1 = "Rast";
11651         }
11652         else if( index1 == 1 )
11653         {
11654             bewegungsGesetz1 = "3-4-5 Polynom";
11655         }
11656         else if( index1 == 2 )
11657         {
11658             bewegungsGesetz1 = "Bestehornsinoide";

```

```

11659     }
11660     QString erg_dphi_H01=anzeige->text();
11661     erg_dphi_H01.setNum( dphi_H01 );
11662
11663     QString erg_ds_H01=anzeige->text();
11664     erg_ds_H01.setNum( ds_H01 );
11665
11666     if( index2 == 0 )
11667     {
11668         bewegungsGesetz2 = "Rast";
11669     }
11670     else if( index2 == 1 )
11671     {
11672         bewegungsGesetz2 = "3-4-5 Polynom";
11673     }
11674     else if( index2 == 2 )
11675     {
11676         bewegungsGesetz2 = "Bestehornsinoide";
11677     }
11678     QString erg_dphi_H12=anzeige->text();
11679     erg_dphi_H12.setNum( dphi_H12 );
11680
11681     QString erg_ds_H12=anzeige->text();
11682     erg_ds_H12.setNum( ds_H12 );
11683
11684
11685     bewegungsVerlauf+= bewegungsGesetz1 +
11686         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11687         "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
11688         "</TD><TD><font size=\"5\">#124;</TD>"
11689         "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
11690         "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
11691         bewegungsGesetz2 +
11692         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11693         "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
11694         "</TD><TD><font size=\"5\">#124;</TD>"
11695         "<TD>s_H12</TD><TD>=</TD><TD>" +
11696         erg_ds_H12 + "</TD></TR>";
11697 }
11698 if( indexAbschnitt == 2 )
11699 {
11700     anzahlderAbschnitte+="3<p>";
11701
11702     if( index1 == 0 )

```

```

11703     {
11704         bewegungsGesetz1 = "Rast";
11705     }
11706     else if( index1 == 1 )
11707     {
11708         bewegungsGesetz1 = "3-4-5 Polynom";
11709     }
11710     else if( index1 == 2 )
11711     {
11712         bewegungsGesetz1 = "Bestehornsinoide";
11713     }
11714     QString erg_dphi_H01=anzeige->text();
11715     erg_dphi_H01.setNum( dphi_H01 );
11716
11717     QString erg_ds_H01=anzeige->text();
11718     erg_ds_H01.setNum( ds_H01 );
11719
11720     if( index2 == 0 )
11721     {
11722         bewegungsGesetz2 = "Rast";
11723     }
11724     else if( index2 == 1 )
11725     {
11726         bewegungsGesetz2 = "3-4-5 Polynom";
11727     }
11728     else if( index2 == 2 )
11729     {
11730         bewegungsGesetz2 = "Bestehornsinoide";
11731     }
11732     QString erg_dphi_H12=anzeige->text();
11733     erg_dphi_H12.setNum( dphi_H12 );
11734
11735     QString erg_ds_H12=anzeige->text();
11736     erg_ds_H12.setNum( ds_H12 );
11737
11738     if( index3 == 0 )
11739     {
11740         bewegungsGesetz3 = "Rast";
11741     }
11742     else if( index3 == 1 )
11743     {
11744         bewegungsGesetz3 = "3-4-5 Polynom";
11745     }
11746     else if( index3 == 2 )

```

```

11747 {
11748     bewegungsGesetz3 = "Bestehornsinoide";
11749 }
11750 QString erg_dphi_H23=anzeige->text();
11751 erg_dphi_H23.setNum( dphi_H23 );
11752
11753 QString erg_ds_H23=anzeige->text();
11754 erg_ds_H23.setNum( ds_H23 );
11755
11756
11757 bewegungsVerlauf+= bewegungsGesetz1 +
11758     "</TD><TD><font size=\"5\">#124;</TD><TD>"
11759     "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
11760     "</TD><TD><font size=\"5\">#124;</TD>"
11761     "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
11762     "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
11763     bewegungsGesetz2 +
11764     "</TD><TD><font size=\"5\">#124;</TD><TD>"
11765     "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
11766     "</TD><TD><font size=\"5\">#124;</TD>"
11767     "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
11768     "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
11769     bewegungsGesetz3 +
11770     "</TD><TD><font size=\"5\">#124;</TD><TD>"
11771     "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
11772     "</TD><TD><font size=\"5\">#124;</TD>"
11773     "<TD>s_H23</TD><TD>=</TD><TD>" +
11774     erg_ds_H23 + "</TD></TR>";
11775 }
11776 if( indexAbschnitt == 3 )
11777 {
11778     anzahlderAbschnitte+="4<p>";
11779
11780     if( index1 == 0 )
11781     {
11782         bewegungsGesetz1 = "Rast";
11783     }
11784     else if( index1 == 1 )
11785     {
11786         bewegungsGesetz1 = "3-4-5 Polynom";
11787     }
11788     else if( index1 == 2 )
11789     {
11790         bewegungsGesetz1 = "Bestehornsinoide";

```

```

11791     }
11792     QString erg_dphi_H01=anzeige->text();
11793     erg_dphi_H01.setNum( dphi_H01 );
11794
11795     QString erg_ds_H01=anzeige->text();
11796     erg_ds_H01.setNum( ds_H01 );
11797
11798     if( index2 == 0 )
11799     {
11800         bewegungsGesetz2 = "Rast";
11801     }
11802     else if( index2 == 1 )
11803     {
11804         bewegungsGesetz2 = "3-4-5 Polynom";
11805     }
11806     else if( index2 == 2 )
11807     {
11808         bewegungsGesetz2 = "Bestehornsinoide";
11809     }
11810     QString erg_dphi_H12=anzeige->text();
11811     erg_dphi_H12.setNum( dphi_H12 );
11812
11813     QString erg_ds_H12=anzeige->text();
11814     erg_ds_H12.setNum( ds_H12 );
11815
11816     if( index3 == 0 )
11817     {
11818         bewegungsGesetz3 = "Rast";
11819     }
11820     else if( index3 == 1 )
11821     {
11822         bewegungsGesetz3 = "3-4-5 Polynom";
11823     }
11824     else if( index3 == 2 )
11825     {
11826         bewegungsGesetz3 = "Bestehornsinoide";
11827     }
11828     QString erg_dphi_H23=anzeige->text();
11829     erg_dphi_H23.setNum( dphi_H23 );
11830
11831     QString erg_ds_H23=anzeige->text();
11832     erg_ds_H23.setNum( ds_H23 );
11833
11834     if( index4 == 0 )

```

```

11835     {
11836         bewegungsGesetz4 = "Rast";
11837     }
11838     else if( index4 == 1 )
11839     {
11840         bewegungsGesetz4 = "3-4-5 Polynom";
11841     }
11842     else if( index4 == 2 )
11843     {
11844         bewegungsGesetz4 = "Bestehornsinoide";
11845     }
11846     QString erg_dphi_H34=anzeige->text();
11847     erg_dphi_H34.setNum( dphi_H34 );
11848
11849     QString erg_ds_H34=anzeige->text();
11850     erg_ds_H34.setNum( ds_H34 );
11851
11852     bewegungsVerlauf+= bewegungsGesetz1 +
11853         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11854         "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
11855         "</TD><TD><font size=\"5\">#124;</TD>"
11856         "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
11857         "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
11858         bewegungsGesetz2 +
11859         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11860         "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
11861         "</TD><TD><font size=\"5\">#124;</TD>"
11862         "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
11863         "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
11864         bewegungsGesetz3 +
11865         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11866         "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
11867         "</TD><TD><font size=\"5\">#124;</TD>"
11868         "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
11869         "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
11870         bewegungsGesetz4 +
11871         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11872         "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
11873         "</TD><TD><font size=\"5\">#124;</TD>"
11874         "<TD>s_H34</TD><TD>=</TD><TD>" +
11875         erg_ds_H34 + "</TD></TR>";
11876     }
11877     if( indexAbschnitt == 4 )
11878     {

```

```

11879      anzahlDerAbschnitte+="5<p>";
11880
11881      if( index1 == 0 )
11882      {
11883          bewegungsGesetz1 = "Rast";
11884      }
11885      else if( index1 == 1 )
11886      {
11887          bewegungsGesetz1 = "3-4-5 Polynom";
11888      }
11889      else if( index1 == 2 )
11890      {
11891          bewegungsGesetz1 = "Bestehornsinoide";
11892      }
11893      QString erg_dphi_H01=anzeige->text();
11894      erg_dphi_H01.setNum( dphi_H01 );
11895
11896      QString erg_ds_H01=anzeige->text();
11897      erg_ds_H01.setNum( ds_H01 );
11898
11899      if( index2 == 0 )
11900      {
11901          bewegungsGesetz2 = "Rast";
11902      }
11903      else if( index2 == 1 )
11904      {
11905          bewegungsGesetz2 = "3-4-5 Polynom";
11906      }
11907      else if( index2 == 2 )
11908      {
11909          bewegungsGesetz2 = "Bestehornsinoide";
11910      }
11911      QString erg_dphi_H12=anzeige->text();
11912      erg_dphi_H12.setNum( dphi_H12 );
11913
11914      QString erg_ds_H12=anzeige->text();
11915      erg_ds_H12.setNum( ds_H12 );
11916
11917      if( index3 == 0 )
11918      {
11919          bewegungsGesetz3 = "Rast";
11920      }
11921      else if( index3 == 1 )
11922      {

```



```

11923     bewegungsGesetz3 = "3-4-5 Polynom";
11924 }
11925 else if( index3 == 2 )
11926 {
11927     bewegungsGesetz3 = "Bestehornsinoide";
11928 }
11929 QString erg_dphi_H23=anzeige->text();
11930 erg_dphi_H23.setNum( dphi_H23 );
11931
11932 QString erg_ds_H23=anzeige->text();
11933 erg_ds_H23.setNum( ds_H23 );
11934
11935 if( index4 == 0 )
11936 {
11937     bewegungsGesetz4 = "Rast";
11938 }
11939 else if( index4 == 1 )
11940 {
11941     bewegungsGesetz4 = "3-4-5 Polynom";
11942 }
11943 else if( index4 == 2 )
11944 {
11945     bewegungsGesetz4 = "Bestehornsinoide";
11946 }
11947 QString erg_dphi_H34=anzeige->text();
11948 erg_dphi_H34.setNum( dphi_H34 );
11949
11950 QString erg_ds_H34=anzeige->text();
11951 erg_ds_H34.setNum( ds_H34 );
11952
11953 if( index5 == 0 )
11954 {
11955     bewegungsGesetz5 = "Rast";
11956 }
11957 else if( index5 == 1 )
11958 {
11959     bewegungsGesetz5 = "3-4-5 Polynom";
11960 }
11961 else if( index5 == 2 )
11962 {
11963     bewegungsGesetz5 = "Bestehornsinoide";
11964 }
11965 QString erg_dphi_H45=anzeige->text();
11966 erg_dphi_H45.setNum( dphi_H45 );

```

```

11967
11968     QString erg_ds_H45=anzeige->text();
11969     erg_ds_H45.setNum( ds_H45 );
11970
11971     bewegungsVerlauf+= bewegungsGesetz1 +
11972         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11973         "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
11974         "</TD><TD><font size=\"5\">#124;</TD>"
11975         "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
11976         "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
11977         bewegungsGesetz2 +
11978         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11979         "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
11980         "</TD><TD><font size=\"5\">#124;</TD>"
11981         "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
11982         "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
11983         bewegungsGesetz3 +
11984         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11985         "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
11986         "</TD><TD><font size=\"5\">#124;</TD>"
11987         "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
11988         "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
11989         bewegungsGesetz4 +
11990         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11991         "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
11992         "</TD><TD><font size=\"5\">#124;</TD>"
11993         "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +
11994         "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +
11995         bewegungsGesetz5 +
11996         "</TD><TD><font size=\"5\">#124;</TD><TD>"
11997         "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +
11998         "</TD><TD><font size=\"5\">#124;</TD>"
11999         "<TD>s_H45</TD><TD>=</TD><TD>" +
12000         erg_ds_H45 + "</TD></TR>";
12001 }
12002 if( indexAbschnitt == 5 )
12003 {
12004     anzahlderAbschnitte+="6<p>";
12005
12006     if( index1 == 0 )
12007     {
12008         bewegungsGesetz1 = "Rast";
12009     }
12010     else if( index1 == 1 )

```

```

12011     {
12012         bewegungsGesetz1 = "3-4-5 Polynom";
12013     }
12014     else if( index1 == 2 )
12015     {
12016         bewegungsGesetz1 = "Bestehornsinoide";
12017     }
12018     QString erg_dphi_H01=anzeige->text();
12019     erg_dphi_H01.setNum( dphi_H01 );
12020
12021     QString erg_ds_H01=anzeige->text();
12022     erg_ds_H01.setNum( ds_H01 );
12023
12024     if( index2 == 0 )
12025     {
12026         bewegungsGesetz2 = "Rast";
12027     }
12028     else if( index2 == 1 )
12029     {
12030         bewegungsGesetz2 = "3-4-5 Polynom";
12031     }
12032     else if( index2 == 2 )
12033     {
12034         bewegungsGesetz2 = "Bestehornsinoide";
12035     }
12036     QString erg_dphi_H12=anzeige->text();
12037     erg_dphi_H12.setNum( dphi_H12 );
12038
12039     QString erg_ds_H12=anzeige->text();
12040     erg_ds_H12.setNum( ds_H12 );
12041
12042     if( index3 == 0 )
12043     {
12044         bewegungsGesetz3 = "Rast";
12045     }
12046     else if( index3 == 1 )
12047     {
12048         bewegungsGesetz3 = "3-4-5 Polynom";
12049     }
12050     else if( index3 == 2 )
12051     {
12052         bewegungsGesetz3 = "Bestehornsinoide";
12053     }
12054     QString erg_dphi_H23=anzeige->text();

```

```

12055     erg_dphi_H23.setNum( dphi_H23 );
12056
12057     QString erg_ds_H23=anzeige->text();
12058     erg_ds_H23.setNum( ds_H23 );
12059
12060     if( index4 == 0 )
12061     {
12062         bewegungsGesetz4 = "Rast";
12063     }
12064     else if( index4 == 1 )
12065     {
12066         bewegungsGesetz4 = "3-4-5 Polynom";
12067     }
12068     else if( index4 == 2 )
12069     {
12070         bewegungsGesetz4 = "Bestehornsinoide";
12071     }
12072     QString erg_dphi_H34=anzeige->text();
12073     erg_dphi_H34.setNum( dphi_H34 );
12074
12075     QString erg_ds_H34=anzeige->text();
12076     erg_ds_H34.setNum( ds_H34 );
12077
12078     if( index5 == 0 )
12079     {
12080         bewegungsGesetz5 = "Rast";
12081     }
12082     else if( index5 == 1 )
12083     {
12084         bewegungsGesetz5 = "3-4-5 Polynom";
12085     }
12086     else if( index5 == 2 )
12087     {
12088         bewegungsGesetz5 = "Bestehornsinoide";
12089     }
12090     QString erg_dphi_H45=anzeige->text();
12091     erg_dphi_H45.setNum( dphi_H45 );
12092
12093     QString erg_ds_H45=anzeige->text();
12094     erg_ds_H45.setNum( ds_H45 );
12095
12096     if( index6 == 0 )
12097     {
12098         bewegungsGesetz6 = "Rast";

```

```

12099     }
12100     else if( index6 == 1 )
12101     {
12102         bewegungsGesetz6 = "3-4-5 Polynom";
12103     }
12104     else if( index6 == 2 )
12105     {
12106         bewegungsGesetz6 = "Bestehornsinoide";
12107     }
12108     QString erg_dphi_H56=anzeige->text();
12109     erg_dphi_H56.setNum( dphi_H56 );
12110
12111     QString erg_ds_H56=anzeige->text();
12112     erg_ds_H56.setNum( ds_H56 );
12113
12114     bewegungsVerlauf+= bewegungsGesetz1 +
12115         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12116         "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
12117         "</TD><TD><font size=\"5\">#124;</TD>"
12118         "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
12119         "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
12120         bewegungsGesetz2 +
12121         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12122         "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
12123         "</TD><TD><font size=\"5\">#124;</TD>"
12124         "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
12125         "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
12126         bewegungsGesetz3 +
12127         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12128         "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
12129         "</TD><TD><font size=\"5\">#124;</TD>"
12130         "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
12131         "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
12132         bewegungsGesetz4 +
12133         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12134         "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
12135         "</TD><TD><font size=\"5\">#124;</TD>"
12136         "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +
12137         "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +
12138         bewegungsGesetz5 +
12139         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12140         "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +
12141         "</TD><TD><font size=\"5\">#124;</TD>"
12142         "<TD>s_H45</TD><TD>=</TD><TD>" + erg_ds_H45 +

```

```

12143         "</TD></TR><TR><TD>6. Abschnitt</TD><TD>:</TD><TD>" +
12144         bewegungsGesetz6 +
12145         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12146         "phi_H56</TD><TD>=</TD><TD>" + erg_dphi_H56 +
12147         "</TD><TD><font size=\"5\">#124;</TD>"
12148         "<TD>s_H56</TD><TD>=</TD><TD>" +
12149         erg_ds_H56 + "</TD></TR>";
12150     }
12151     if( indexAbschnitt == 6 )
12152     {
12153         anzahlderAbschnitte+="7<p>";
12154
12155         if( index1 == 0 )
12156         {
12157             bewegungsGesetz1 = "Rast";
12158         }
12159         else if( index1 == 1 )
12160         {
12161             bewegungsGesetz1 = "3-4-5 Polynom";
12162         }
12163         else if( index1 == 2 )
12164         {
12165             bewegungsGesetz1 = "Bestehornsinoide";
12166         }
12167         QString erg_dphi_H01=anzeige->text();
12168         erg_dphi_H01.setNum( dphi_H01 );
12169
12170         QString erg_ds_H01=anzeige->text();
12171         erg_ds_H01.setNum( ds_H01 );
12172
12173         if( index2 == 0 )
12174         {
12175             bewegungsGesetz2 = "Rast";
12176         }
12177         else if( index2 == 1 )
12178         {
12179             bewegungsGesetz2 = "3-4-5 Polynom";
12180         }
12181         else if( index2 == 2 )
12182         {
12183             bewegungsGesetz2 = "Bestehornsinoide";
12184         }
12185         QString erg_dphi_H12=anzeige->text();
12186         erg_dphi_H12.setNum( dphi_H12 );

```

```

12187
12188     QString erg_ds_H12=anzeige->text();
12189     erg_ds_H12.setNum( ds_H12 );
12190
12191     if( index3 == 0 )
12192     {
12193         bewegungsGesetz3 = "Rast";
12194     }
12195     else if( index3 == 1 )
12196     {
12197         bewegungsGesetz3 = "3-4-5 Polynom";
12198     }
12199     else if( index3 == 2 )
12200     {
12201         bewegungsGesetz3 = "Bestehornsinoide";
12202     }
12203     QString erg_dphi_H23=anzeige->text();
12204     erg_dphi_H23.setNum( dphi_H23 );
12205
12206     QString erg_ds_H23=anzeige->text();
12207     erg_ds_H23.setNum( ds_H23 );
12208
12209     if( index4 == 0 )
12210     {
12211         bewegungsGesetz4 = "Rast";
12212     }
12213     else if( index4 == 1 )
12214     {
12215         bewegungsGesetz4 = "3-4-5 Polynom";
12216     }
12217     else if( index4 == 2 )
12218     {
12219         bewegungsGesetz4 = "Bestehornsinoide";
12220     }
12221     QString erg_dphi_H34=anzeige->text();
12222     erg_dphi_H34.setNum( dphi_H34 );
12223
12224     QString erg_ds_H34=anzeige->text();
12225     erg_ds_H34.setNum( ds_H34 );
12226
12227     if( index5 == 0 )
12228     {
12229         bewegungsGesetz5 = "Rast";
12230     }

```

```

12231     else if( index5 == 1 )
12232     {
12233         bewegungsGesetz5 = "3-4-5 Polynom";
12234     }
12235     else if( index5 == 2 )
12236     {
12237         bewegungsGesetz5 = "Bestehornsinoide";
12238     }
12239     QString erg_dphi_H45=anzeige->text();
12240     erg_dphi_H45.setNum( dphi_H45 );
12241
12242     QString erg_ds_H45=anzeige->text();
12243     erg_ds_H45.setNum( ds_H45 );
12244
12245     if( index6 == 0 )
12246     {
12247         bewegungsGesetz6 = "Rast";
12248     }
12249     else if( index6 == 1 )
12250     {
12251         bewegungsGesetz6 = "3-4-5 Polynom";
12252     }
12253     else if( index6 == 2 )
12254     {
12255         bewegungsGesetz6 = "Bestehornsinoide";
12256     }
12257     QString erg_dphi_H56=anzeige->text();
12258     erg_dphi_H56.setNum( dphi_H56 );
12259
12260     QString erg_ds_H56=anzeige->text();
12261     erg_ds_H56.setNum( ds_H56 );
12262
12263     if( index7 == 0 )
12264     {
12265         bewegungsGesetz7 = "Rast";
12266     }
12267     else if( index7 == 1 )
12268     {
12269         bewegungsGesetz7 = "3-4-5 Polynom";
12270     }
12271     else if( index7 == 2 )
12272     {
12273         bewegungsGesetz7 = "Bestehornsinoide";
12274     }

```



```

12275   QString erg_dphi_H67=anzeige->text();
12276   erg_dphi_H67.setNum( dphi_H67 );
12277
12278   QString erg_ds_H67=anzeige->text();
12279   erg_ds_H67.setNum( ds_H67 );
12280
12281   bewegungsVerlauf+= bewegungsGesetz1 +
12282                       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12283                       "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
12284                       "</TD><TD><font size=\"5\">#124;</TD>"
12285                       "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
12286                       "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
12287                       bewegungsGesetz2 +
12288                       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12289                       "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
12290                       "</TD><TD><font size=\"5\">#124;</TD>"
12291                       "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
12292                       "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
12293                       bewegungsGesetz3 +
12294                       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12295                       "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
12296                       "</TD><TD><font size=\"5\">#124;</TD>"
12297                       "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
12298                       "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
12299                       bewegungsGesetz4 +
12300                       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12301                       "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
12302                       "</TD><TD><font size=\"5\">#124;</TD>"
12303                       "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +
12304                       "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +
12305                       bewegungsGesetz5 +
12306                       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12307                       "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +
12308                       "</TD><TD><font size=\"5\">#124;</TD>"
12309                       "<TD>s_H45</TD><TD>=</TD><TD>" + erg_ds_H45 +
12310                       "</TD></TR><TR><TD>6. Abschnitt</TD><TD>:</TD><TD>" +
12311                       bewegungsGesetz6 +
12312                       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12313                       "phi_H56</TD><TD>=</TD><TD>" + erg_dphi_H56 +
12314                       "</TD><TD><font size=\"5\">#124;</TD>"
12315                       "<TD>s_H56</TD><TD>=</TD><TD>" + erg_ds_H56 +
12316                       "</TD></TR><TR><TD>7. Abschnitt</TD><TD>:</TD><TD>" +
12317                       bewegungsGesetz7 +
12318                       "</TD><TD><font size=\"5\">#124;</TD><TD>"

```

```

12319         "phi_H67</TD><TD>=</TD><TD>" + erg_dphi_H67 +
12320         "</TD><TD><font size=\"5\">#124;</TD>"
12321         "<TD>s_H67</TD><TD>=</TD><TD>" +
12322         erg_ds_H67 + "</TD></TR>";
12323     }
12324     if( indexAbschnitt == 7 )
12325     {
12326         anzahlderAbschnitte+="8<p>";
12327
12328         if( index1 == 0 )
12329         {
12330             bewegungsGesetz1 = "Rast";
12331         }
12332         else if( index1 == 1 )
12333         {
12334             bewegungsGesetz1 = "3-4-5 Polynom";
12335         }
12336         else if( index1 == 2 )
12337         {
12338             bewegungsGesetz1 = "Bestehornsinoide";
12339         }
12340         QString erg_dphi_H01=anzeige->text();
12341         erg_dphi_H01.setNum( dphi_H01 );
12342
12343         QString erg_ds_H01=anzeige->text();
12344         erg_ds_H01.setNum( ds_H01 );
12345
12346         if( index2 == 0 )
12347         {
12348             bewegungsGesetz2 = "Rast";
12349         }
12350         else if( index2 == 1 )
12351         {
12352             bewegungsGesetz2 = "3-4-5 Polynom";
12353         }
12354         else if( index2 == 2 )
12355         {
12356             bewegungsGesetz2 = "Bestehornsinoide";
12357         }
12358         QString erg_dphi_H12=anzeige->text();
12359         erg_dphi_H12.setNum( dphi_H12 );
12360
12361         QString erg_ds_H12=anzeige->text();
12362         erg_ds_H12.setNum( ds_H12 );

```

```

12363
12364     if( index3 == 0 )
12365     {
12366         bewegungsGesetz3 = "Rast";
12367     }
12368     else if( index3 == 1 )
12369     {
12370         bewegungsGesetz3 = "3-4-5 Polynom";
12371     }
12372     else if( index3 == 2 )
12373     {
12374         bewegungsGesetz3 = "Bestehornsinoide";
12375     }
12376     QString erg_dphi_H23=anzeige->text();
12377     erg_dphi_H23.setNum( dphi_H23 );
12378
12379     QString erg_ds_H23=anzeige->text();
12380     erg_ds_H23.setNum( ds_H23 );
12381
12382     if( index4 == 0 )
12383     {
12384         bewegungsGesetz4 = "Rast";
12385     }
12386     else if( index4 == 1 )
12387     {
12388         bewegungsGesetz4 = "3-4-5 Polynom";
12389     }
12390     else if( index4 == 2 )
12391     {
12392         bewegungsGesetz4 = "Bestehornsinoide";
12393     }
12394     QString erg_dphi_H34=anzeige->text();
12395     erg_dphi_H34.setNum( dphi_H34 );
12396
12397     QString erg_ds_H34=anzeige->text();
12398     erg_ds_H34.setNum( ds_H34 );
12399
12400     if( index5 == 0 )
12401     {
12402         bewegungsGesetz5 = "Rast";
12403     }
12404     else if( index5 == 1 )
12405     {
12406         bewegungsGesetz5 = "3-4-5 Polynom";

```

```

12407     }
12408     else if( index5 == 2 )
12409     {
12410         bewegungsGesetz5 = "Bestehornsinoide";
12411     }
12412     QString erg_dphi_H45=anzeige->text();
12413     erg_dphi_H45.setNum( dphi_H45 );
12414
12415     QString erg_ds_H45=anzeige->text();
12416     erg_ds_H45.setNum( ds_H45 );
12417
12418     if( index6 == 0 )
12419     {
12420         bewegungsGesetz6 = "Rast";
12421     }
12422     else if( index6 == 1 )
12423     {
12424         bewegungsGesetz6 = "3-4-5 Polynom";
12425     }
12426     else if( index6 == 2 )
12427     {
12428         bewegungsGesetz6 = "Bestehornsinoide";
12429     }
12430     QString erg_dphi_H56=anzeige->text();
12431     erg_dphi_H56.setNum( dphi_H56 );
12432
12433     QString erg_ds_H56=anzeige->text();
12434     erg_ds_H56.setNum( ds_H56 );
12435
12436     if( index7 == 0 )
12437     {
12438         bewegungsGesetz7 = "Rast";
12439     }
12440     else if( index7 == 1 )
12441     {
12442         bewegungsGesetz7 = "3-4-5 Polynom";
12443     }
12444     else if( index7 == 2 )
12445     {
12446         bewegungsGesetz7 = "Bestehornsinoide";
12447     }
12448     QString erg_dphi_H67=anzeige->text();
12449     erg_dphi_H67.setNum( dphi_H67 );
12450

```

```

12451   QString erg_ds_H67=anzeige->text();
12452   erg_ds_H67.setNum( ds_H67 );
12453
12454   if( index8 == 0 )
12455   {
12456       bewegungsGesetz8 = "Rast";
12457   }
12458   else if( index8 == 1 )
12459   {
12460       bewegungsGesetz8 = "3-4-5 Polynom";
12461   }
12462   else if( index8 == 2 )
12463   {
12464       bewegungsGesetz8 = "Bestehornsinoide";
12465   }
12466   QString erg_dphi_H78=anzeige->text();
12467   erg_dphi_H78.setNum( dphi_H78 );
12468
12469   QString erg_ds_H78=anzeige->text();
12470   erg_ds_H78.setNum( ds_H78 );
12471
12472   bewegungsVerlauf+= bewegungsGesetz1 +
12473       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12474       "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
12475       "</TD><TD><font size=\"5\">#124;</TD>"
12476       "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
12477       "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
12478       bewegungsGesetz2 +
12479       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12480       "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
12481       "</TD><TD><font size=\"5\">#124;</TD>"
12482       "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
12483       "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
12484       bewegungsGesetz3 +
12485       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12486       "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
12487       "</TD><TD><font size=\"5\">#124;</TD>"
12488       "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
12489       "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
12490       bewegungsGesetz4 +
12491       "</TD><TD><font size=\"5\">#124;</TD><TD>"
12492       "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
12493       "</TD><TD><font size=\"5\">#124;</TD>"
12494       "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +

```

```

12495      "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +
12496      bewegungsGesetz5 +
12497      "</TD><TD><font size=\"5\">#124;</TD><TD>"
12498      "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +
12499      "</TD><TD><font size=\"5\">#124;</TD>"
12500      "<TD>s_H45</TD><TD>=</TD><TD>" + erg_ds_H45 +
12501      "</TD></TR><TR><TD>6. Abschnitt</TD><TD>:</TD><TD>" +
12502      bewegungsGesetz6 +
12503      "</TD><TD><font size=\"5\">#124;</TD><TD>"
12504      "phi_H56</TD><TD>=</TD><TD>" + erg_dphi_H56 +
12505      "</TD><TD><font size=\"5\">#124;</TD>"
12506      "<TD>s_H56</TD><TD>=</TD><TD>" + erg_ds_H56 +
12507      "</TD></TR><TR><TD>7. Abschnitt</TD><TD>:</TD><TD>" +
12508      bewegungsGesetz7 +
12509      "</TD><TD><font size=\"5\">#124;</TD><TD>"
12510      "phi_H67</TD><TD>=</TD><TD>" + erg_dphi_H67 +
12511      "</TD><TD><font size=\"5\">#124;</TD>"
12512      "<TD>s_H67</TD><TD>=</TD><TD>" + erg_ds_H67 +
12513      "</TD></TR><TR><TD>8. Abschnitt</TD><TD>:</TD><TD>" +
12514      bewegungsGesetz8 +
12515      "</TD><TD><font size=\"5\">#124;</TD><TD>"
12516      "phi_H78</TD><TD>=</TD><TD>" + erg_dphi_H78 +
12517      "</TD><TD><font size=\"5\">#124;</TD>"
12518      "<TD>s_H78</TD><TD>=</TD><TD>" +
12519      erg_ds_H78 + "</TD></TR>";
12520  }
12521  if( indexAbschnitt == 8 )
12522  {
12523      anzahlderAbschnitte+="9<p>";
12524
12525      if( index1 == 0 )
12526      {
12527          bewegungsGesetz1 = "Rast";
12528      }
12529      else if( index1 == 1 )
12530      {
12531          bewegungsGesetz1 = "3-4-5 Polynom";
12532      }
12533      else if( index1 == 2 )
12534      {
12535          bewegungsGesetz1 = "Bestehornsinoide";
12536      }
12537      QString erg_dphi_H01=anzeige->text();
12538      erg_dphi_H01.setNum( dphi_H01 );

```

```

12539
12540     QString erg_ds_H01=anzeige->text();
12541     erg_ds_H01.setNum( ds_H01 );
12542
12543     if( index2 == 0 )
12544     {
12545         bewegungsGesetz2 = "Rast";
12546     }
12547     else if( index2 == 1 )
12548     {
12549         bewegungsGesetz2 = "3-4-5 Polynom";
12550     }
12551     else if( index2 == 2 )
12552     {
12553         bewegungsGesetz2 = "Bestehornsinoide";
12554     }
12555     QString erg_dphi_H12=anzeige->text();
12556     erg_dphi_H12.setNum( dphi_H12 );
12557
12558     QString erg_ds_H12=anzeige->text();
12559     erg_ds_H12.setNum( ds_H12 );
12560
12561     if( index3 == 0 )
12562     {
12563         bewegungsGesetz3 = "Rast";
12564     }
12565     else if( index3 == 1 )
12566     {
12567         bewegungsGesetz3 = "3-4-5 Polynom";
12568     }
12569     else if( index3 == 2 )
12570     {
12571         bewegungsGesetz3 = "Bestehornsinoide";
12572     }
12573     QString erg_dphi_H23=anzeige->text();
12574     erg_dphi_H23.setNum( dphi_H23 );
12575
12576     QString erg_ds_H23=anzeige->text();
12577     erg_ds_H23.setNum( ds_H23 );
12578
12579     if( index4 == 0 )
12580     {
12581         bewegungsGesetz4 = "Rast";
12582     }

```

```

12583     else if( index4 == 1 )
12584     {
12585         bewegungsGesetz4 = "3-4-5 Polynom";
12586     }
12587     else if( index4 == 2 )
12588     {
12589         bewegungsGesetz4 = "Bestehornsinoide";
12590     }
12591     QString erg_dphi_H34=anzeige->text();
12592     erg_dphi_H34.setNum( dphi_H34 );
12593
12594     QString erg_ds_H34=anzeige->text();
12595     erg_ds_H34.setNum( ds_H34 );
12596
12597     if( index5 == 0 )
12598     {
12599         bewegungsGesetz5 = "Rast";
12600     }
12601     else if( index5 == 1 )
12602     {
12603         bewegungsGesetz5 = "3-4-5 Polynom";
12604     }
12605     else if( index5 == 2 )
12606     {
12607         bewegungsGesetz5 = "Bestehornsinoide";
12608     }
12609     QString erg_dphi_H45=anzeige->text();
12610     erg_dphi_H45.setNum( dphi_H45 );
12611
12612     QString erg_ds_H45=anzeige->text();
12613     erg_ds_H45.setNum( ds_H45 );
12614
12615     if( index6 == 0 )
12616     {
12617         bewegungsGesetz6 = "Rast";
12618     }
12619     else if( index6 == 1 )
12620     {
12621         bewegungsGesetz6 = "3-4-5 Polynom";
12622     }
12623     else if( index6 == 2 )
12624     {
12625         bewegungsGesetz6 = "Bestehornsinoide";
12626     }

```



```

12627     QString erg_dphi_H56=anzeige->text();
12628     erg_dphi_H56.setNum( dphi_H56 );
12629
12630     QString erg_ds_H56=anzeige->text();
12631     erg_ds_H56.setNum( ds_H56 );
12632
12633     if( index7 == 0 )
12634     {
12635         bewegungsGesetz7 = "Rast";
12636     }
12637     else if( index7 == 1 )
12638     {
12639         bewegungsGesetz7 = "3-4-5 Polynom";
12640     }
12641     else if( index7 == 2 )
12642     {
12643         bewegungsGesetz7 = "Bestehornsinoide";
12644     }
12645     QString erg_dphi_H67=anzeige->text();
12646     erg_dphi_H67.setNum( dphi_H67 );
12647
12648     QString erg_ds_H67=anzeige->text();
12649     erg_ds_H67.setNum( ds_H67 );
12650
12651     if( index8 == 0 )
12652     {
12653         bewegungsGesetz8 = "Rast";
12654     }
12655     else if( index8 == 1 )
12656     {
12657         bewegungsGesetz8 = "3-4-5 Polynom";
12658     }
12659     else if( index8 == 2 )
12660     {
12661         bewegungsGesetz8 = "Bestehornsinoide";
12662     }
12663     QString erg_dphi_H78=anzeige->text();
12664     erg_dphi_H78.setNum( dphi_H78 );
12665
12666     QString erg_ds_H78=anzeige->text();
12667     erg_ds_H78.setNum( ds_H78 );
12668
12669     if( index9 == 0 )
12670     {

```

```

12671     bewegungsGesetz9 = "Rast";
12672 }
12673 else if( index9 == 1 )
12674 {
12675     bewegungsGesetz9 = "3-4-5 Polynom";
12676 }
12677 else if( index9 == 2 )
12678 {
12679     bewegungsGesetz9 = "Bestehornsinoide";
12680 }
12681 QString erg_dphi_H89=anzeige->text();
12682 erg_dphi_H89.setNum( dphi_H89 );
12683
12684 QString erg_ds_H89=anzeige->text();
12685 erg_ds_H89.setNum( ds_H89 );
12686
12687 bewegungsVerlauf+= bewegungsGesetz1 +
12688     "</TD><TD><font size=\"5\">#124;</TD><TD>"
12689     "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
12690     "</TD><TD><font size=\"5\">#124;</TD>"
12691     "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
12692     "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
12693     bewegungsGesetz2 +
12694     "</TD><TD><font size=\"5\">#124;</TD><TD>"
12695     "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
12696     "</TD><TD><font size=\"5\">#124;</TD>"
12697     "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
12698     "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
12699     bewegungsGesetz3 +
12700     "</TD><TD><font size=\"5\">#124;</TD><TD>"
12701     "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
12702     "</TD><TD><font size=\"5\">#124;</TD>"
12703     "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
12704     "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
12705     bewegungsGesetz4 +
12706     "</TD><TD><font size=\"5\">#124;</TD><TD>"
12707     "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
12708     "</TD><TD><font size=\"5\">#124;</TD>"
12709     "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +
12710     "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +
12711     bewegungsGesetz5 +
12712     "</TD><TD><font size=\"5\">#124;</TD><TD>"
12713     "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +
12714     "</TD><TD><font size=\"5\">#124;</TD>"

```

```

12715 "<TD>s_H45</TD><TD>=</TD><TD>" + erg_ds_H45 +
12716 "</TD></TR><TR><TD>6. Abschnitt</TD><TD>:</TD><TD>" +
12717 bewegungsGesetz6 +
12718 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12719 "phi_H56</TD><TD>=</TD><TD>" + erg_dphi_H56 +
12720 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12721 "<TD>s_H56</TD><TD>=</TD><TD>" + erg_ds_H56 +
12722 "</TD></TR><TR><TD>7. Abschnitt</TD><TD>:</TD><TD>" +
12723 bewegungsGesetz7 +
12724 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12725 "phi_H67</TD><TD>=</TD><TD>" + erg_dphi_H67 +
12726 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12727 "<TD>s_H67</TD><TD>=</TD><TD>" + erg_ds_H67 +
12728 "</TD></TR><TR><TD>8. Abschnitt</TD><TD>:</TD><TD>" +
12729 bewegungsGesetz8 +
12730 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12731 "phi_H78</TD><TD>=</TD><TD>" + erg_dphi_H78 +
12732 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12733 "<TD>s_H78</TD><TD>=</TD><TD>" + erg_ds_H78 +
12734 "</TD></TR><TR><TD>9. Abschnitt</TD><TD>:</TD><TD>" +
12735 bewegungsGesetz9 +
12736 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12737 "phi_H89</TD><TD>=</TD><TD>" + erg_dphi_H89 +
12738 "</TD><TD><font size=\"5\">#124;</TD><TD>"
12739 "<TD>s_H89</TD><TD>=</TD><TD>" +
12740 erg_ds_H89 + "</TD></TR>";
12741 }
12742 if( indexAbschnitt == 9 )
12743 {
12744     anzahlderAbschnitte+="10<p>";
12745
12746     if( index1 == 0 )
12747     {
12748         bewegungsGesetz1 = "Rast";
12749     }
12750     else if( index1 == 1 )
12751     {
12752         bewegungsGesetz1 = "3-4-5 Polynom";
12753     }
12754     else if( index1 == 2 )
12755     {
12756         bewegungsGesetz1 = "Bestehornsinoide";
12757     }
12758     QString erg_dphi_H01=anzeige->text();

```

```

12759     erg_dphi_H01.setNum( dphi_H01 );
12760
12761     QString erg_ds_H01=anzeige->text();
12762     erg_ds_H01.setNum( ds_H01 );
12763
12764     if( index2 == 0 )
12765     {
12766         bewegungsGesetz2 = "Rast";
12767     }
12768     else if( index2 == 1 )
12769     {
12770         bewegungsGesetz2 = "3-4-5 Polynom";
12771     }
12772     else if( index2 == 2 )
12773     {
12774         bewegungsGesetz2 = "Bestehornsinoide";
12775     }
12776     QString erg_dphi_H12=anzeige->text();
12777     erg_dphi_H12.setNum( dphi_H12 );
12778
12779     QString erg_ds_H12=anzeige->text();
12780     erg_ds_H12.setNum( ds_H12 );
12781
12782     if( index3 == 0 )
12783     {
12784         bewegungsGesetz3 = "Rast";
12785     }
12786     else if( index3 == 1 )
12787     {
12788         bewegungsGesetz3 = "3-4-5 Polynom";
12789     }
12790     else if( index3 == 2 )
12791     {
12792         bewegungsGesetz3 = "Bestehornsinoide";
12793     }
12794     QString erg_dphi_H23=anzeige->text();
12795     erg_dphi_H23.setNum( dphi_H23 );
12796
12797     QString erg_ds_H23=anzeige->text();
12798     erg_ds_H23.setNum( ds_H23 );
12799
12800     if( index4 == 0 )
12801     {
12802         bewegungsGesetz4 = "Rast";

```

```

12803     }
12804     else if( index4 == 1 )
12805     {
12806         bewegungsGesetz4 = "3-4-5 Polynom";
12807     }
12808     else if( index4 == 2 )
12809     {
12810         bewegungsGesetz4 = "Bestehornsinoide";
12811     }
12812     QString erg_dphi_H34=anzeige->text();
12813     erg_dphi_H34.setNum( dphi_H34 );
12814
12815     QString erg_ds_H34=anzeige->text();
12816     erg_ds_H34.setNum( ds_H34 );
12817
12818     if( index5 == 0 )
12819     {
12820         bewegungsGesetz5 = "Rast";
12821     }
12822     else if( index5 == 1 )
12823     {
12824         bewegungsGesetz5 = "3-4-5 Polynom";
12825     }
12826     else if( index5 == 2 )
12827     {
12828         bewegungsGesetz5 = "Bestehornsinoide";
12829     }
12830     QString erg_dphi_H45=anzeige->text();
12831     erg_dphi_H45.setNum( dphi_H45 );
12832
12833     QString erg_ds_H45=anzeige->text();
12834     erg_ds_H45.setNum( ds_H45 );
12835
12836     if( index6 == 0 )
12837     {
12838         bewegungsGesetz6 = "Rast";
12839     }
12840     else if( index6 == 1 )
12841     {
12842         bewegungsGesetz6 = "3-4-5 Polynom";
12843     }
12844     else if( index6 == 2 )
12845     {
12846         bewegungsGesetz6 = "Bestehornsinoide";

```

```

12847     }
12848     QString erg_dphi_H56=anzeige->text();
12849     erg_dphi_H56.setNum( dphi_H56 );
12850
12851     QString erg_ds_H56=anzeige->text();
12852     erg_ds_H56.setNum( ds_H56 );
12853
12854     if( index7 == 0 )
12855     {
12856         bewegungsGesetz7 = "Rast";
12857     }
12858     else if( index7 == 1 )
12859     {
12860         bewegungsGesetz7 = "3-4-5 Polynom";
12861     }
12862     else if( index7 == 2 )
12863     {
12864         bewegungsGesetz7 = "Bestehornsinoide";
12865     }
12866     QString erg_dphi_H67=anzeige->text();
12867     erg_dphi_H67.setNum( dphi_H67 );
12868
12869     QString erg_ds_H67=anzeige->text();
12870     erg_ds_H67.setNum( ds_H67 );
12871
12872     if( index8 == 0 )
12873     {
12874         bewegungsGesetz8 = "Rast";
12875     }
12876     else if( index8 == 1 )
12877     {
12878         bewegungsGesetz8 = "3-4-5 Polynom";
12879     }
12880     else if( index8 == 2 )
12881     {
12882         bewegungsGesetz8 = "Bestehornsinoide";
12883     }
12884     QString erg_dphi_H78=anzeige->text();
12885     erg_dphi_H78.setNum( dphi_H78 );
12886
12887     QString erg_ds_H78=anzeige->text();
12888     erg_ds_H78.setNum( ds_H78 );
12889
12890     if( index9 == 0 )

```

```

12891     {
12892         bewegungsGesetz9 = "Rast";
12893     }
12894     else if( index9 == 1 )
12895     {
12896         bewegungsGesetz9 = "3-4-5 Polynom";
12897     }
12898     else if( index9 == 2 )
12899     {
12900         bewegungsGesetz9 = "Bestehornsinoide";
12901     }
12902     QString erg_dphi_H89=anzeige->text();
12903     erg_dphi_H89.setNum( dphi_H89 );
12904
12905     QString erg_ds_H89=anzeige->text();
12906     erg_ds_H89.setNum( ds_H89 );
12907
12908     if( index10 == 0 )
12909     {
12910         bewegungsGesetz10 = "Rast";
12911     }
12912     else if( index10 == 1 )
12913     {
12914         bewegungsGesetz10 = "3-4-5 Polynom";
12915     }
12916     else if( index10 == 2 )
12917     {
12918         bewegungsGesetz10 = "Bestehornsinoide";
12919     }
12920     QString erg_dphi_H910=anzeige->text();
12921     erg_dphi_H910.setNum( dphi_H910 );
12922
12923     QString erg_ds_H910=anzeige->text();
12924     erg_ds_H910.setNum( ds_H910 );
12925
12926     bewegungsVerlauf+= bewegungsGesetz1 +
12927         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12928         "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
12929         "</TD><TD><font size=\"5\">#124;</TD>"
12930         "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
12931         "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
12932         bewegungsGesetz2 +
12933         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12934         "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +

```

12935 "</TD><TD>#124;</TD>"

12936 "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +

12937 "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +

12938 bewegungsGesetz3 +

12939 "</TD><TD>#124;</TD><TD>"

12940 "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +

12941 "</TD><TD>#124;</TD>"

12942 "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +

12943 "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +

12944 bewegungsGesetz4 +

12945 "</TD><TD>#124;</TD><TD>"

12946 "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +

12947 "</TD><TD>#124;</TD>"

12948 "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +

12949 "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +

12950 bewegungsGesetz5 +

12951 "</TD><TD>#124;</TD><TD>"

12952 "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +

12953 "</TD><TD>#124;</TD>"

12954 "<TD>s_H45</TD><TD>=</TD><TD>" + erg_ds_H45 +

12955 "</TD></TR><TR><TD>6. Abschnitt</TD><TD>:</TD><TD>" +

12956 bewegungsGesetz6 +

12957 "</TD><TD>#124;</TD><TD>"

12958 "phi_H56</TD><TD>=</TD><TD>" + erg_dphi_H56 +

12959 "</TD><TD>#124;</TD>"

12960 "<TD>s_H56</TD><TD>=</TD><TD>" + erg_ds_H56 +

12961 "</TD></TR><TR><TD>7. Abschnitt</TD><TD>:</TD><TD>" +

12962 bewegungsGesetz7 +

12963 "</TD><TD>#124;</TD><TD>"

12964 "phi_H67</TD><TD>=</TD><TD>" + erg_dphi_H67 +

12965 "</TD><TD>#124;</TD>"

12966 "<TD>s_H67</TD><TD>=</TD><TD>" + erg_ds_H67 +

12967 "</TD></TR><TR><TD>8. Abschnitt</TD><TD>:</TD><TD>" +

12968 bewegungsGesetz8 +

12969 "</TD><TD>#124;</TD><TD>"

12970 "phi_H78</TD><TD>=</TD><TD>" + erg_dphi_H78 +

12971 "</TD><TD>#124;</TD>"

12972 "<TD>s_H78</TD><TD>=</TD><TD>" + erg_ds_H78 +

12973 "</TD></TR><TR><TD>9. Abschnitt</TD><TD>:</TD><TD>" +

12974 bewegungsGesetz9 +

12975 "</TD><TD>#124;</TD><TD>"

12976 "phi_H89</TD><TD>=</TD><TD>" + erg_dphi_H89 +

12977 "</TD><TD>#124;</TD>"

12978 "<TD>s_H89</TD><TD>=</TD><TD>" + erg_ds_H89 +


```

12979         "</TD></TR><TR><TD>10. Abschnitt</TD><TD>:</TD><TD>" +
12980         bewegungsGesetz10 +
12981         "</TD><TD><font size=\"5\">#124;</TD><TD>"
12982         "phi_H910</TD><TD>=</TD><TD>" + erg_dphi_H910 +
12983         "</TD><TD><font size=\"5\">#124;</TD>"
12984         "<TD>s_H910</TD><TD>=</TD><TD>" +
12985         erg_ds_H910 + "</TD></TR>";
12986     }
12987     if( indexAbschnitt == 10 )
12988     {
12989         anzahlderAbschnitte+="11<p>";
12990
12991         if( index1 == 0 )
12992         {
12993             bewegungsGesetz1 = "Rast";
12994         }
12995         else if( index1 == 1 )
12996         {
12997             bewegungsGesetz1 = "3-4-5 Polynom";
12998         }
12999         else if( index1 == 2 )
13000         {
13001             bewegungsGesetz1 = "Bestehornsinoide";
13002         }
13003         QString erg_dphi_H01=anzeige->text();
13004         erg_dphi_H01.setNum( dphi_H01 );
13005
13006         QString erg_ds_H01=anzeige->text();
13007         erg_ds_H01.setNum( ds_H01 );
13008
13009         if( index2 == 0 )
13010         {
13011             bewegungsGesetz2 = "Rast";
13012         }
13013         else if( index2 == 1 )
13014         {
13015             bewegungsGesetz2 = "3-4-5 Polynom";
13016         }
13017         else if( index2 == 2 )
13018         {
13019             bewegungsGesetz2 = "Bestehornsinoide";
13020         }
13021         QString erg_dphi_H12=anzeige->text();
13022         erg_dphi_H12.setNum( dphi_H12 );

```

```

13023
13024     QString erg_ds_H12=anzeige->text();
13025     erg_ds_H12.setNum( ds_H12 );
13026
13027     if( index3 == 0 )
13028     {
13029         bewegungsGesetz3 = "Rast";
13030     }
13031     else if( index3 == 1 )
13032     {
13033         bewegungsGesetz3 = "3-4-5 Polynom";
13034     }
13035     else if( index3 == 2 )
13036     {
13037         bewegungsGesetz3 = "Bestehornsinoide";
13038     }
13039     QString erg_dphi_H23=anzeige->text();
13040     erg_dphi_H23.setNum( dphi_H23 );
13041
13042     QString erg_ds_H23=anzeige->text();
13043     erg_ds_H23.setNum( ds_H23 );
13044
13045     if( index4 == 0 )
13046     {
13047         bewegungsGesetz4 = "Rast";
13048     }
13049     else if( index4 == 1 )
13050     {
13051         bewegungsGesetz4 = "3-4-5 Polynom";
13052     }
13053     else if( index4 == 2 )
13054     {
13055         bewegungsGesetz4 = "Bestehornsinoide";
13056     }
13057     QString erg_dphi_H34=anzeige->text();
13058     erg_dphi_H34.setNum( dphi_H34 );
13059
13060     QString erg_ds_H34=anzeige->text();
13061     erg_ds_H34.setNum( ds_H34 );
13062
13063     if( index5 == 0 )
13064     {
13065         bewegungsGesetz5 = "Rast";
13066     }

```

```

13067     else if( index5 == 1 )
13068     {
13069         bewegungsGesetz5 = "3-4-5 Polynom";
13070     }
13071     else if( index5 == 2 )
13072     {
13073         bewegungsGesetz5 = "Bestehornsinoide";
13074     }
13075     QString erg_dphi_H45=anzeige->text();
13076     erg_dphi_H45.setNum( dphi_H45 );
13077
13078     QString erg_ds_H45=anzeige->text();
13079     erg_ds_H45.setNum( ds_H45 );
13080
13081     if( index6 == 0 )
13082     {
13083         bewegungsGesetz6 = "Rast";
13084     }
13085     else if( index6 == 1 )
13086     {
13087         bewegungsGesetz6 = "3-4-5 Polynom";
13088     }
13089     else if( index6 == 2 )
13090     {
13091         bewegungsGesetz6 = "Bestehornsinoide";
13092     }
13093     QString erg_dphi_H56=anzeige->text();
13094     erg_dphi_H56.setNum( dphi_H56 );
13095
13096     QString erg_ds_H56=anzeige->text();
13097     erg_ds_H56.setNum( ds_H56 );
13098
13099     if( index7 == 0 )
13100     {
13101         bewegungsGesetz7 = "Rast";
13102     }
13103     else if( index7 == 1 )
13104     {
13105         bewegungsGesetz7 = "3-4-5 Polynom";
13106     }
13107     else if( index7 == 2 )
13108     {
13109         bewegungsGesetz7 = "Bestehornsinoide";
13110     }

```

```

13111   QString erg_dphi_H67=anzeige->text();
13112   erg_dphi_H67.setNum( dphi_H67 );
13113
13114   QString erg_ds_H67=anzeige->text();
13115   erg_ds_H67.setNum( ds_H67 );
13116
13117   if( index8 == 0 )
13118   {
13119       bewegungsGesetz8 = "Rast";
13120   }
13121   else if( index8 == 1 )
13122   {
13123       bewegungsGesetz8 = "3-4-5 Polynom";
13124   }
13125   else if( index8 == 2 )
13126   {
13127       bewegungsGesetz8 = "Bestehornsinoide";
13128   }
13129   QString erg_dphi_H78=anzeige->text();
13130   erg_dphi_H78.setNum( dphi_H78 );
13131
13132   QString erg_ds_H78=anzeige->text();
13133   erg_ds_H78.setNum( ds_H78 );
13134
13135   if( index9 == 0 )
13136   {
13137       bewegungsGesetz9 = "Rast";
13138   }
13139   else if( index9 == 1 )
13140   {
13141       bewegungsGesetz9 = "3-4-5 Polynom";
13142   }
13143   else if( index9 == 2 )
13144   {
13145       bewegungsGesetz9 = "Bestehornsinoide";
13146   }
13147   QString erg_dphi_H89=anzeige->text();
13148   erg_dphi_H89.setNum( dphi_H89 );
13149
13150   QString erg_ds_H89=anzeige->text();
13151   erg_ds_H89.setNum( ds_H89 );
13152
13153   if( index10 == 0 )
13154   {

```

```

13155     bewegungsGesetz10 = "Rast";
13156 }
13157 else if( index10 == 1 )
13158 {
13159     bewegungsGesetz10 = "3-4-5 Polynom";
13160 }
13161 else if( index10 == 2 )
13162 {
13163     bewegungsGesetz10 = "Bestehornsinoide";
13164 }
13165 QString erg_dphi_H910=anzeige->text();
13166 erg_dphi_H910.setNum( dphi_H910 );
13167
13168 QString erg_ds_H910=anzeige->text();
13169 erg_ds_H910.setNum( ds_H910 );
13170
13171 if( index11 == 0 )
13172 {
13173     bewegungsGesetz11 = "Rast";
13174 }
13175 else if( index11 == 1 )
13176 {
13177     bewegungsGesetz11 = "3-4-5 Polynom";
13178 }
13179 else if( index11 == 2 )
13180 {
13181     bewegungsGesetz11 = "Bestehornsinoide";
13182 }
13183 QString erg_dphi_H1011=anzeige->text();
13184 erg_dphi_H1011.setNum( dphi_H1011 );
13185
13186 QString erg_ds_H1011=anzeige->text();
13187 erg_ds_H1011.setNum( ds_H1011 );
13188
13189 bewegungsVerlauf+= bewegungsGesetz1 +
13190     "</TD><TD><font size=\"5\">#124;</TD><TD>"
13191     "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
13192     "</TD><TD><font size=\"5\">#124;</TD>"
13193     "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
13194     "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
13195     bewegungsGesetz2 +
13196     "</TD><TD><font size=\"5\">#124;</TD><TD>"
13197     "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
13198     "</TD><TD><font size=\"5\">#124;</TD>"

```

13199 "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
13200 "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
13201 bewegungsGesetz3 +
13202 "</TD><TD>#124;</TD><TD>"
13203 "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
13204 "</TD><TD>#124;</TD><TD>"
13205 "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
13206 "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
13207 bewegungsGesetz4 +
13208 "</TD><TD>#124;</TD><TD>"
13209 "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
13210 "</TD><TD>#124;</TD><TD>"
13211 "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +
13212 "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +
13213 bewegungsGesetz5 +
13214 "</TD><TD>#124;</TD><TD>"
13215 "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +
13216 "</TD><TD>#124;</TD><TD>"
13217 "<TD>s_H45</TD><TD>=</TD><TD>" + erg_ds_H45 +
13218 "</TD></TR><TR><TD>6. Abschnitt</TD><TD>:</TD><TD>" +
13219 bewegungsGesetz6 +
13220 "</TD><TD>#124;</TD><TD>"
13221 "phi_H56</TD><TD>=</TD><TD>" + erg_dphi_H56 +
13222 "</TD><TD>#124;</TD><TD>"
13223 "<TD>s_H56</TD><TD>=</TD><TD>" + erg_ds_H56 +
13224 "</TD></TR><TR><TD>7. Abschnitt</TD><TD>:</TD><TD>" +
13225 bewegungsGesetz7 +
13226 "</TD><TD>#124;</TD><TD>"
13227 "phi_H67</TD><TD>=</TD><TD>" + erg_dphi_H67 +
13228 "</TD><TD>#124;</TD><TD>"
13229 "<TD>s_H67</TD><TD>=</TD><TD>" + erg_ds_H67 +
13230 "</TD></TR><TR><TD>8. Abschnitt</TD><TD>:</TD><TD>" +
13231 bewegungsGesetz8 +
13232 "</TD><TD>#124;</TD><TD>"
13233 "phi_H78</TD><TD>=</TD><TD>" + erg_dphi_H78 +
13234 "</TD><TD>#124;</TD><TD>"
13235 "<TD>s_H78</TD><TD>=</TD><TD>" + erg_ds_H78 +
13236 "</TD></TR><TR><TD>9. Abschnitt</TD><TD>:</TD><TD>" +
13237 bewegungsGesetz9 +
13238 "</TD><TD>#124;</TD><TD>"
13239 "phi_H89</TD><TD>=</TD><TD>" + erg_dphi_H89 +
13240 "</TD><TD>#124;</TD><TD>"
13241 "<TD>s_H89</TD><TD>=</TD><TD>" + erg_ds_H89 +
13242 "</TD></TR><TR><TD>10. Abschnitt</TD><TD>:</TD><TD>" +

```

13243      bewegungsGesetz10 +
13244      "</TD><TD><font size=\"5\">#124;</TD><TD>"
13245      "phi_H910</TD><TD>=</TD><TD>" + erg_dphi_H910 +
13246      "</TD><TD><font size=\"5\">#124;</TD>"
13247      "<TD>s_H910</TD><TD>=</TD><TD>" + erg_ds_H910 +
13248      "</TD></TR><TR><TD>11. Abschnitt</TD><TD>:</TD><TD>" +
13249      bewegungsGesetz11 +
13250      "</TD><TD><font size=\"5\">#124;</TD><TD>"
13251      "phi_H1011</TD><TD>=</TD><TD>" + erg_dphi_H1011 +
13252      "</TD><TD><font size=\"5\">#124;</TD>"
13253      "<TD>s_H1011</TD><TD>=</TD><TD>" +
13254      erg_ds_H1011 + "</TD></TR>";
13255  }
13256  if( indexAbschnitt == 11 )
13257  {
13258      anzahlderAbschnitte+="12<p>";
13259
13260      if( index1 == 0 )
13261      {
13262          bewegungsGesetz1 = "Rast";
13263      }
13264      else if( index1 == 1 )
13265      {
13266          bewegungsGesetz1 = "3-4-5 Polynom";
13267      }
13268      else if( index1 == 2 )
13269      {
13270          bewegungsGesetz1 = "Bestehornsinoide";
13271      }
13272      QString erg_dphi_H01=anzeige->text();
13273      erg_dphi_H01.setNum( dphi_H01 );
13274
13275      QString erg_ds_H01=anzeige->text();
13276      erg_ds_H01.setNum( ds_H01 );
13277
13278      if( index2 == 0 )
13279      {
13280          bewegungsGesetz2 = "Rast";
13281      }
13282      else if( index2 == 1 )
13283      {
13284          bewegungsGesetz2 = "3-4-5 Polynom";
13285      }
13286      else if( index2 == 2 )

```

```

13287     {
13288         bewegungsGesetz2 = "Bestehornsinoide";
13289     }
13290     QString erg_dphi_H12=anzeige->text();
13291     erg_dphi_H12.setNum( dphi_H12 );
13292
13293     QString erg_ds_H12=anzeige->text();
13294     erg_ds_H12.setNum( ds_H12 );
13295
13296     if( index3 == 0 )
13297     {
13298         bewegungsGesetz3 = "Rast";
13299     }
13300     else if( index3 == 1 )
13301     {
13302         bewegungsGesetz3 = "3-4-5 Polynom";
13303     }
13304     else if( index3 == 2 )
13305     {
13306         bewegungsGesetz3 = "Bestehornsinoide";
13307     }
13308     QString erg_dphi_H23=anzeige->text();
13309     erg_dphi_H23.setNum( dphi_H23 );
13310
13311     QString erg_ds_H23=anzeige->text();
13312     erg_ds_H23.setNum( ds_H23 );
13313
13314     if( index4 == 0 )
13315     {
13316         bewegungsGesetz4 = "Rast";
13317     }
13318     else if( index4 == 1 )
13319     {
13320         bewegungsGesetz4 = "3-4-5 Polynom";
13321     }
13322     else if( index4 == 2 )
13323     {
13324         bewegungsGesetz4 = "Bestehornsinoide";
13325     }
13326     QString erg_dphi_H34=anzeige->text();
13327     erg_dphi_H34.setNum( dphi_H34 );
13328
13329     QString erg_ds_H34=anzeige->text();
13330     erg_ds_H34.setNum( ds_H34 );

```



```

13331
13332     if( index5 == 0 )
13333     {
13334         bewegungsGesetz5 = "Rast";
13335     }
13336     else if( index5 == 1 )
13337     {
13338         bewegungsGesetz5 = "3-4-5 Polynom";
13339     }
13340     else if( index5 == 2 )
13341     {
13342         bewegungsGesetz5 = "Bestehornsinoide";
13343     }
13344     QString erg_dphi_H45=anzeige->text();
13345     erg_dphi_H45.setNum( dphi_H45 );
13346
13347     QString erg_ds_H45=anzeige->text();
13348     erg_ds_H45.setNum( ds_H45 );
13349
13350     if( index6 == 0 )
13351     {
13352         bewegungsGesetz6 = "Rast";
13353     }
13354     else if( index6 == 1 )
13355     {
13356         bewegungsGesetz6 = "3-4-5 Polynom";
13357     }
13358     else if( index6 == 2 )
13359     {
13360         bewegungsGesetz6 = "Bestehornsinoide";
13361     }
13362     QString erg_dphi_H56=anzeige->text();
13363     erg_dphi_H56.setNum( dphi_H56 );
13364
13365     QString erg_ds_H56=anzeige->text();
13366     erg_ds_H56.setNum( ds_H56 );
13367
13368     if( index7 == 0 )
13369     {
13370         bewegungsGesetz7 = "Rast";
13371     }
13372     else if( index7 == 1 )
13373     {
13374         bewegungsGesetz7 = "3-4-5 Polynom";

```

```

13375     }
13376     else if( index7 == 2 )
13377     {
13378         bewegungsGesetz7 = "Bestehornsinoide";
13379     }
13380     QString erg_dphi_H67=anzeige->text();
13381     erg_dphi_H67.setNum( dphi_H67 );
13382
13383     QString erg_ds_H67=anzeige->text();
13384     erg_ds_H67.setNum( ds_H67 );
13385
13386     if( index8 == 0 )
13387     {
13388         bewegungsGesetz8 = "Rast";
13389     }
13390     else if( index8 == 1 )
13391     {
13392         bewegungsGesetz8 = "3-4-5 Polynom";
13393     }
13394     else if( index8 == 2 )
13395     {
13396         bewegungsGesetz8 = "Bestehornsinoide";
13397     }
13398     QString erg_dphi_H78=anzeige->text();
13399     erg_dphi_H78.setNum( dphi_H78 );
13400
13401     QString erg_ds_H78=anzeige->text();
13402     erg_ds_H78.setNum( ds_H78 );
13403
13404     if( index9 == 0 )
13405     {
13406         bewegungsGesetz9 = "Rast";
13407     }
13408     else if( index9 == 1 )
13409     {
13410         bewegungsGesetz9 = "3-4-5 Polynom";
13411     }
13412     else if( index9 == 2 )
13413     {
13414         bewegungsGesetz9 = "Bestehornsinoide";
13415     }
13416     QString erg_dphi_H89=anzeige->text();
13417     erg_dphi_H89.setNum( dphi_H89 );
13418

```

```

13419     QString erg_ds_H89=anzeige->text();
13420     erg_ds_H89.setNum( ds_H89 );
13421
13422     if( index10 == 0 )
13423     {
13424         bewegungsGesetz10 = "Rast";
13425     }
13426     else if( index10 == 1 )
13427     {
13428         bewegungsGesetz10 = "3-4-5 Polynom";
13429     }
13430     else if( index10 == 2 )
13431     {
13432         bewegungsGesetz10 = "Bestehornsinoide";
13433     }
13434     QString erg_dphi_H910=anzeige->text();
13435     erg_dphi_H910.setNum( dphi_H910 );
13436
13437     QString erg_ds_H910=anzeige->text();
13438     erg_ds_H910.setNum( ds_H910 );
13439
13440     if( index11 == 0 )
13441     {
13442         bewegungsGesetz11 = "Rast";
13443     }
13444     else if( index11 == 1 )
13445     {
13446         bewegungsGesetz11 = "3-4-5 Polynom";
13447     }
13448     else if( index11 == 2 )
13449     {
13450         bewegungsGesetz11= "Bestehornsinoide";
13451     }
13452     QString erg_dphi_H1011=anzeige->text();
13453     erg_dphi_H1011.setNum( dphi_H1011 );
13454
13455     QString erg_ds_H1011=anzeige->text();
13456     erg_ds_H1011.setNum( ds_H1011 );
13457
13458     if( index12 == 0 )
13459     {
13460         bewegungsGesetz12 = "Rast";
13461     }
13462     else if( index12 == 1 )

```

```

13463 {
13464     bewegungsGesetz12 = "3-4-5 Polynom";
13465 }
13466 else if( index12 == 2 )
13467 {
13468     bewegungsGesetz12 = "Bestehornsinoide";
13469 }
13470 QString erg_dphi_H1112=anzeige->text();
13471 erg_dphi_H1112.setNum( dphi_H1112 );
13472
13473 QString erg_ds_H1112=anzeige->text();
13474 erg_ds_H1112.setNum( ds_H1112 );
13475
13476 bewegungsVerlauf+= bewegungsGesetz1 +
13477     "</TD><TD><font size=\"5\">#124;</TD><TD>"
13478     "phi_H01</TD><TD>=</TD><TD>" + erg_dphi_H01 +
13479     "</TD><TD><font size=\"5\">#124;</TD>"
13480     "<TD>s_H01</TD><TD>=</TD><TD>" + erg_ds_H01 +
13481     "</TD></TR><TR><TD>2. Abschnitt</TD><TD>:</TD><TD>" +
13482     bewegungsGesetz2 +
13483     "</TD><TD><font size=\"5\">#124;</TD><TD>"
13484     "phi_H12</TD><TD>=</TD><TD>" + erg_dphi_H12 +
13485     "</TD><TD><font size=\"5\">#124;</TD>"
13486     "<TD>s_H12</TD><TD>=</TD><TD>" + erg_ds_H12 +
13487     "</TD></TR><TR><TD>3. Abschnitt</TD><TD>:</TD><TD>" +
13488     bewegungsGesetz3 +
13489     "</TD><TD><font size=\"5\">#124;</TD><TD>"
13490     "phi_H23</TD><TD>=</TD><TD>" + erg_dphi_H23 +
13491     "</TD><TD><font size=\"5\">#124;</TD>"
13492     "<TD>s_H23</TD><TD>=</TD><TD>" + erg_ds_H23 +
13493     "</TD></TR><TR><TD>4. Abschnitt</TD><TD>:</TD><TD>" +
13494     bewegungsGesetz4 +
13495     "</TD><TD><font size=\"5\">#124;</TD><TD>"
13496     "phi_H34</TD><TD>=</TD><TD>" + erg_dphi_H34 +
13497     "</TD><TD><font size=\"5\">#124;</TD>"
13498     "<TD>s_H34</TD><TD>=</TD><TD>" + erg_ds_H34 +
13499     "</TD></TR><TR><TD>5. Abschnitt</TD><TD>:</TD><TD>" +
13500     bewegungsGesetz5 +
13501     "</TD><TD><font size=\"5\">#124;</TD><TD>"
13502     "phi_H45</TD><TD>=</TD><TD>" + erg_dphi_H45 +
13503     "</TD><TD><font size=\"5\">#124;</TD>"
13504     "<TD>s_H45</TD><TD>=</TD><TD>" + erg_ds_H45 +
13505     "</TD></TR><TR><TD>6. Abschnitt</TD><TD>:</TD><TD>" +
13506     bewegungsGesetz6 +

```

```

13507 " </TD><TD><font size=\"5\">#124;</TD><TD>"
13508 "phi_H56</TD><TD>=</TD><TD>" + erg_dphi_H56 +
13509 " </TD><TD><font size=\"5\">#124;</TD>"
13510 "<TD>s_H56</TD><TD>=</TD><TD>" + erg_ds_H56 +
13511 " </TD></TR><TR><TD>7. Abschnitt</TD><TD>:</TD><TD>" +
13512 bewegungsGesetz7 +
13513 " </TD><TD><font size=\"5\">#124;</TD><TD>"
13514 "phi_H67</TD><TD>=</TD><TD>" + erg_dphi_H67 +
13515 " </TD><TD><font size=\"5\">#124;</TD>"
13516 "<TD>s_H67</TD><TD>=</TD><TD>" + erg_ds_H67 +
13517 " </TD></TR><TR><TD>8. Abschnitt</TD><TD>:</TD><TD>" +
13518 bewegungsGesetz8 +
13519 " </TD><TD><font size=\"5\">#124;</TD><TD>"
13520 "phi_H78</TD><TD>=</TD><TD>" + erg_dphi_H78 +
13521 " </TD><TD><font size=\"5\">#124;</TD>"
13522 "<TD>s_H78</TD><TD>=</TD><TD>" + erg_ds_H78 +
13523 " </TD></TR><TR><TD>9. Abschnitt</TD><TD>:</TD><TD>" +
13524 bewegungsGesetz9 +
13525 " </TD><TD><font size=\"5\">#124;</TD><TD>"
13526 "phi_H89</TD><TD>=</TD><TD>" + erg_dphi_H89 +
13527 " </TD><TD><font size=\"5\">#124;</TD>"
13528 "<TD>s_H89</TD><TD>=</TD><TD>" + erg_ds_H89 +
13529 " </TD></TR><TR><TD>10. Abschnitt</TD><TD>:</TD><TD>" +
13530 bewegungsGesetz10 +
13531 " </TD><TD><font size=\"5\">#124;</TD><TD>"
13532 "phi_H910</TD><TD>=</TD><TD>" + erg_dphi_H910 +
13533 " </TD><TD><font size=\"5\">#124;</TD>"
13534 "<TD>s_H910</TD><TD>=</TD><TD>" + erg_ds_H910 +
13535 " </TD></TR><TR><TD>11. Abschnitt</TD><TD>:</TD><TD>" +
13536 bewegungsGesetz11 +
13537 " </TD><TD><font size=\"5\">#124;</TD><TD>"
13538 "phi_H1011</TD><TD>=</TD><TD>" + erg_dphi_H1011 +
13539 " </TD><TD><font size=\"5\">#124;</TD>"
13540 "<TD>s_H1011</TD><TD>=</TD><TD>" + erg_ds_H1011 +
13541 " </TD></TR><TR><TD>12. Abschnitt</TD><TD>:</TD><TD>" +
13542 bewegungsGesetz12 +
13543 " </TD><TD><font size=\"5\">#124;</TD><TD>"
13544 "phi_H1112</TD><TD>=</TD><TD>" + erg_dphi_H1112 +
13545 " </TD><TD><font size=\"5\">#124;</TD>"
13546 "<TD>s_H1112</TD><TD>=</TD><TD>" +
13547 erg_ds_H1112 + " </TD></TR>";
13548 }
13549
13550 bewegungsVerlauf+= "</TABLE><p>";

```

```

13551    geometrieDaten="<h4>Geometriedaten</h4><p>"
13552        "<TABLE border=\"0\" cellspacing=\"0\" cellpadding=\"4\" >";
13553
13554    QString erg_dl_3=anzeige->text();
13555    erg_dl_3.setNum( dl_3 );
13556
13557    QString erg_dr_R=anzeige->text();
13558    erg_dr_R.setNum( dr_R );
13559
13560    QString erg_dl_4=anzeige->text();
13561    erg_dl_4.setNum( dl_4 );
13562
13563    QString erg_dx_A0=anzeige->text();
13564    erg_dx_A0.setNum( dx_A0 );
13565
13566    QString erg_dl_sk=anzeige->text();
13567    erg_dl_sk.setNum( dl_sk );
13568
13569    QString erg_dy_A0=anzeige->text();
13570    erg_dy_A0.setNum( dy_A0 );
13571
13572    QString erg_dbeta=anzeige->text();
13573    erg_dbeta.setNum( dbeta );
13574
13575    QString erg_dx_s0=anzeige->text();
13576    erg_dx_s0.setNum( dx_s0 );
13577
13578    QString erg_dy_s0=anzeige->text();
13579    erg_dy_s0.setNum( dy_s0_orig );
13580
13581    QString erg_dr_G=anzeige->text();
13582    erg_dr_G.setNum( dr_G );
13583
13584    QString erg_dx_H=anzeige->text();
13585    erg_dx_H.setNum( dx_H );
13586
13587    QString erg_dn=anzeige->text();
13588    erg_dn.setNum( dn_1 );
13589
13590    QString erg_dy_H=anzeige->text();
13591    erg_dy_H.setNum( dy_H );
13592
13593    QString erg_dr_S=anzeige->text();
13594    erg_dr_S.setNum( dr_S );

```

```

13595
13596 QString erg_dr_W=anzeige->text();
13597 erg_dr_W.setNum( dr_W );
13598
13599 QString erg_dalpha_R=anzeige->text();
13600 erg_dalpha_R.setNum( dalpha_R );
13601
13602 QString erg_dl_3stern=anzeige->text();
13603 erg_dl_3stern.setNum( dl_3star );
13604
13605 QString erg_dr_Rstern=anzeige->text();
13606 erg_dr_Rstern.setNum( dr_Rstar );
13607
13608 if(indexKurve==0)
13609 {
13610     static const char* tabellenkopf0[]={ "<html><head></head>"
13611     "<body><table><thead><tr><th>phi</th><th>s(phi)</th>"
13612     "<th>s_strich</th><th>psi(phi)</th><th>psi(phi)</th>"
13613     "<th>psi'(phi)</th><th>psi''(phi)</th><th>x_Bik</th>"
13614     "<th>y_Bik</th><th>x_Bik_strich</th><th>y_Bik_strich</th>"
13615     "<th>x_i_vers</th><th>y_i_vers</th><th>x_B_vers</th>"
13616     "<th>y_B_vers</th><th>x_a_vers</th><th>y_a_vers</th>"
13617     "<th>Abstand</th><th>mue</th><th>r_K</th></tr></thead>"
13618     "<tbody><tr><td>[Grad]</td><td>[mm]</td><td>[mm/Grad]</td>"
13619     "<td>[Grad]</td><td>[rad]</td><td>[rad]</td><td>[rad]</td>"
13620     "<td>[mm]</td><td>[mm]</td><td>[mm]/Grad</td>"
13621     "<td>[mm]/Grad</td><td>[mm]</td><td>[mm]</td><td>[mm]</td>"
13622     "<td>[mm]</td><td>[mm]</td><td>[mm]</td><td>[mm]</td>"
13623     "<td>[Grad]</td><td>[mm]</td></tr>",0};
13624     text+=datetime + tabellenkopf0[0];
13625
13626     geometrieDaten+="<TR><TD>x_A0</TD><TD>=</TD><TD>" + erg_dx_A0 +
13627     "</TD><TD><font size=\"5\">#124;</TD>"
13628     "<TD>r_G</TD><TD>=</TD><TD>" + erg_dr_G +
13629     "</TD><TD><font size=\"5\">#124;</TD>"
13630     "<TD></TD><TD></TD><TD></TD></TR>"
13631     "<TR><TD>y_A0</TD><TD>=</TD><TD>" + erg_dy_A0 +
13632     "</TD><TD><font size=\"5\">#124;</TD>"
13633     "<TD>r_R</TD><TD>=</TD><TD>" + erg_dr_R +
13634     "</TD><TD><font size=\"5\">#124;</TD>"
13635     "<TD></TD><TD></TD><TD></TD></TR>"
13636     "<TR><TD>x_s0</TD><TD>=</TD><TD>" + erg_dx_s0 +
13637     "</TD><TD><font size=\"5\">#124;</TD>"
13638     "<TD>r_S</TD><TD>=</TD><TD>" + erg_dr_S +

```

```

13639         "</TD><TD><font size=\"5\">⌵;</TD>"
13640         "<TD></TD><TD></TD><TD></TD></TR>"
13641         "<TR><TD>y_s0</TD><TD>=</TD><TD>" + erg_dy_s0 +
13642         "</TD><TD><font size=\"5\">⌵;</TD>"
13643         "<TD>r_W</TD><TD>=</TD><TD>" + erg_dr_W +
13644         "</TD><TD><font size=\"5\">⌵;</TD>"
13645         "<TD></TD><TD></TD><TD></TD></TR>"
13646         "<TR><TD>l_3</TD><TD>=</TD><TD>" + erg_dl_3 +
13647         "</TD><TD><font size=\"5\">⌵;</TD>"
13648         "<TD>n</TD><TD>=</TD><TD>" + erg_dn +
13649         "</TD><TD><font size=\"5\">⌵;</TD>"
13650         "<TD></TD><TD></TD><TD></TD></TR>"
13651         "<TR><TD>l_sk</TD><TD>=</TD><TD>" + erg_dl_sk +
13652         "</TD><TD><font size=\"5\">⌵;</TD>"
13653         "<TD></TD><TD></TD><TD>"
13654         "</TD><TD><font size=\"5\">⌵;</TD>"
13655         "<TD></TD><TD></TD><TD></TD></TR>"
13656         "<TR><TD>l_4</TD><TD>=</TD><TD>" + erg_dl_4 +
13657         "</TD><TD><font size=\"5\">⌵;</TD>"
13658         "<TD></TD><TD></TD><TD>"
13659         "</TD><TD><font size=\"5\">⌵;</TD>"
13660         "<TD></TD><TD></TD><TD></TD></TR>";
13661     }
13662     if(indexKurve==1)
13663     {
13664         static const char* tabellenkopf1[]={ "<html><head></head>"
13665         "<body><table><thead><tr><th>phi</th><th>s(phi)</th>"
13666         "<th>s_strich</th><th>psi(phi)</th><th>psi(phi)</th>"
13667         "<th>psi'(phi)</th><th>psi''(phi)</th><th>x_Bik</th>"
13668         "<th>y_Bik</th><th>x_Bik_strich</th><th>y_Bik_strich</th>"
13669         "<th>x_i_vers</th><th>y_i_vers</th><th>x_B_vers</th>"
13670         "<th>y_B_vers</th><th>x_a_vers</th><th>y_a_vers</th>"
13671         "<th>psistar</th><th>psistar_rad</th><th>x_Bikstar</th><th>y_Bikstar</th>"
13672         "<th>x_Bik_strichstar</th><th>y_Bik_strichstar</th>"
13673         "<th>x_i_versstar</th><th>y_i_versstar</th><th>x_B_versstar</th>"
13674         "<th>y_B_versstar</th><th>x_a_versstar</th><th>y_a_versstar</th>"
13675         "<th>Abstand</th><th>mue</th><th>r_K</th></tr></thead>"
13676         "<tbody><tr><td>[Grad]</td><td>[mm]</td><td>[mm/Grad]</td>"
13677         "<td>[Grad]</td><td>[rad]</td><td>[rad]</td><td>[rad]</td>"
13678         "<td>[mm]</td><td>[mm]</td><td>[mm]/Grad</td>"
13679         "<td>[mm]/Grad</td><td>[mm]</td><td>[mm]</td><td>[mm]</td>"
13680         "<td>[mm]</td><td>[mm]</td><td>[mm]</td>"
13681         "<td>[Grad]</td><td>[rad]</td><td>[mm]</td><td>[mm]</td>"
13682         "<td>[rad]</td><td>[rad]</td>"}

```



```

13683         "<td>[mm]</td><td>[mm]</td><td>[mm]</td>"
13684         "<td>[mm]</td><td>[mm]</td><td>[mm]</td>"
13685         "<td>[mm]</td><td>[Grad]</td><td>[mm]</td></tr>",0};
13686     text+=datetime + tabellenkopf1[0];
13687
13688     geometrieDaten+="<TR><TD>x_A0</TD><TD>=</TD><TD>" + erg_dx_A0 +
13689         "</TD><TD><font size=\"5\">#124;</TD>"
13690         "<TD>r_G</TD><TD>=</TD><TD>" + erg_dr_G +
13691         "</TD><TD><font size=\"5\">#124;</TD>"
13692         "<TD>alpha_R</TD><TD>=</TD><TD>" + erg_dalpha_R +
13693         "</TD></TR><TR><TD>y_A0</TD><TD>=</TD><TD>" + erg_dy_A0 +
13694         "</TD><TD><font size=\"5\">#124;</TD>"
13695         "<TD>r_R</TD><TD>=</TD><TD>" + erg_dr_R +
13696         "</TD><TD><font size=\"5\">#124;</TD>"
13697         "<TD>l_3stern</TD><TD>=</TD><TD>" + erg_dl_3stern +
13698         "</TD></TR><TR><TD>x_S0</TD><TD>=</TD><TD>" + erg_dx_s0 +
13699         "</TD><TD><font size=\"5\">#124;</TD>"
13700         "<TD>r_S</TD><TD>=</TD><TD>" + erg_dr_S +
13701         "</TD><TD><font size=\"5\">#124;</TD>"
13702         "<TD>r_Rstern</TD><TD>=</TD><TD>" + erg_dr_Rstern +
13703         "</TD></TR><TR><TD>y_s0</TD><TD>=</TD><TD>" + erg_dy_s0 +
13704         "</TD><TD><font size=\"5\">#124;</TD>"
13705         "<TD>r_W</TD><TD>=</TD><TD>" + erg_dr_W +
13706         "</TD><TD><font size=\"5\">#124;</TD>"
13707         "<TD></TD><TD></TD><TD></TD></TR>"
13708         "<TR><TD>l_3</TD><TD>=</TD><TD>" + erg_dl_3 +
13709         "</TD><TD><font size=\"5\">#124;</TD>"
13710         "<TD>n</TD><TD>=</TD><TD>" + erg_dn +
13711         "</TD><TD><font size=\"5\">#124;</TD>"
13712         "<TD></TD><TD></TD><TD></TD></TR>"
13713         "<TR><TD>l_sk</TD><TD>=</TD><TD>" + erg_dl_sk +
13714         "</TD><TD><font size=\"5\">#124;</TD>"
13715         "<TD></TD><TD></TD><TD>"
13716         "</TD><TD><font size=\"5\">#124;</TD>"
13717         "<TD></TD><TD></TD><TD></TD></TR>"
13718         "<TR><TD>l_4</TD><TD>=</TD><TD>" + erg_dl_4 +
13719         "</TD><TD><font size=\"5\">#124;</TD>"
13720         "<TD></TD><TD></TD><TD>"
13721         "</TD><TD><font size=\"5\">#124;</TD>"
13722         "<TD></TD><TD></TD><TD></TD></TR>";
13723     }
13724
13725     geometrieDaten+= "</TABLE><p>";
13726     static const char* parameterrest[]={"</body></html>",0};

```

```

13727
13728 vorzeichenDefinitionen="<h4>Vorzeichendefinitionen</h4><p>"
13729         "<ul><li>Ist die Drehrichtung der Kurvenscheibe"
13730         " mathematisch positiv oder negativ?";
13731
13732     if(scheibendreh_richtung == 'p')
13733     {
13734         vorzeichenDefinitionen+="<br>-> mathematisch positiv";
13735     }
13736     else if(scheibendreh_richtung == 'n')
13737     {
13738         vorzeichenDefinitionen+="<br>-> mathematisch negativ";
13739     }
13740
13741     vorzeichenDefinitionen+="<li>Sind die Hubrichtungen von Koordinate"
13742         " x_S0 und Hub s(phi) zu Beginn der"
13743         "<br>Bewegung gleich gerichtet?";
13744
13745     if(s_richtung == 'y')
13746     {
13747         vorzeichenDefinitionen+="<br>-> Ja";
13748     }
13749     if(s_richtung == 'n')
13750     {
13751         vorzeichenDefinitionen+="<br>-> Nein";
13752     }
13753
13754     vorzeichenDefinitionen+="<li>In welchem Quadrant befindet sich der Punkt C_0?";
13755
13756     if(Ba_Qudrant == 'y')
13757     {
13758         vorzeichenDefinitionen+="<br>-> 1. bzw. 4. Quadrant";
13759     }
13760     if(Ba_Qudrant == 'n')
13761     {
13762         vorzeichenDefinitionen+="<br>-> 2. bzw. 3. Quadrant";
13763     }
13764
13765     vorzeichenDefinitionen+="<li>Ist die Bewegung des Schwinghebels zu Beginn"
13766         " der Bewegung mathematisch"
13767         "<br>positiv oder negativ?";
13768
13769     if(positiv == 'y')
13770     {

```

```

13771     vorzeichenDefinitionen+="  
>-> mathematisch positiv";
13772 }
13773 if(positiv == 'n')
13774 {
13775     vorzeichenDefinitionen+="  
>-> mathematisch negativ";
13776 }
13777
13778 vorzeichenDefinitionen+="  
><p><p>&#160;";
13779
13780
13781 texttwo+= anzahlderAbschnitte + bewegungsverlauf + geometriedaten +
13782     vorzeichenDefinitionen + parameterrest[0];
13783
13784 if(s_richtung == 'y')
13785 {
13786     for(phi=0; phi<360; phi+=n_1)
13787     {
13788         /* M, Schleifen für Bewegungsabschnitte von phi */
13789
13790         berechneHubaufrufen();
13791
13792         /* Substitutionen für psi */
13793
13794         A = (2*l_sk*(x_s0+s));
13795         A_plus_deltaphi = (2*l_sk*(x_s0+s_plus_deltaphi));
13796         A_minus_deltaphi = (2*l_sk*(x_s0+s_minus_deltaphi));
13797         B = (-2*l_sk*y_s0);
13798         C = (pow((x_s0+s),2))+(pow(y_s0,2))-(pow(l_4,2))+(pow(l_sk,2));
13799         C_plus_deltaphi = (pow((x_s0+s_plus_deltaphi),2))+(pow(y_s0,2))-
13800             (pow(l_4,2))+(pow(l_sk,2));
13801         C_minus_deltaphi = (pow((x_s0+s_minus_deltaphi),2))+(pow(y_s0,2))-
13802             (pow(l_4,2))+(pow(l_sk,2));
13803
13804         subcalculate();
13805         speichereASCII();
13806         weiseZu_konvertiereTyp();
13807         ergebnisseMainWindow();
13808     }
13809 }
13810 else if(s_richtung == 'n')
13811 {
13812     for(phi=0; phi<360; phi+=n_1)
13813     {
13814         /* Schleifen für Bewegungsabschnitte von phi */

```

```

13815
13816     berechneHubaufrufen();
13817
13818     /* Substitutionen für psi */
13819
13820     A = (2*l_sk*(x_s0-s));
13821     A_plus_deltaphi = (2*l_sk*(x_s0-s_plus_deltaphi));
13822     A_minus_deltaphi = (2*l_sk*(x_s0-s_minus_deltaphi));
13823     B = (-2*l_sk*y_s0);
13824     C = (pow((x_s0-s),2))+(pow(y_s0,2))-(pow(l_4,2))+(pow(l_sk,2));
13825     C_plus_deltaphi = (pow((x_s0-s_plus_deltaphi),2))+
13826                     (pow(y_s0,2))-(pow(l_4,2))+(pow(l_sk,2));
13827     C_minus_deltaphi = (pow((x_s0-s_minus_deltaphi),2))+
13828                     (pow(y_s0,2))-(pow(l_4,2))+(pow(l_sk,2));
13829
13830     subcalculate();
13831     speichereASCII();
13832     weiseZu_konvertiereTyp();
13833     ergebnisseMainWindow();
13834 }
13835 }
13836 static const char* tabellenrest[] = {"</tbody>"
13837                                     "</table>"
13838                                     "</body>"
13839                                     "</html>",0};
13840
13841     /* implizite Typkonvertierungen für Ausgabe notw. */
13842
13843     dpsi_sk0 = psi_sk0;
13844     dpsi_0 = psi_0;
13845     dpsi_G = psi_G;
13846
13847     /* Debugging */
13848
13849     qDebug( "psi_sk0 = %Lg; psi_0 = %Lg; psi_G = %Lg;",
13850            psi_sk0, psi_0, psi_G);
13851
13852     QString erg_ausgabe_psi_sk0=anzeige->text();
13853     erg_ausgabe_psi_sk0 = "<br> psi_sk0 = " + erg_ausgabe_psi_sk0.setNum(dpsi_sk0)
13854                       + " Grad";
13855
13856     QString erg_ausgabe_psi_0=anzeige->text();
13857     erg_ausgabe_psi_0 = "<br> psi_0 = " + erg_ausgabe_psi_0.setNum( dpsi_0 ) +
13858                       " Grad";

```

```

13859
13860 QString erg_ausgabe_psi_G=anzeige->text();
13861 erg_ausgabe_psi_G = "<br> psi_G = " + erg_ausgabe_psi_G.setNum( dpsi_G ) +
13862     " Grad<br><br><br><br>";
13863
13864 text+=tabellenrest[0] + erg_ausgabe_phi_1 + erg_ausgabe_phi_2 +
13865     erg_ausgabe_phi_3 + erg_ausgabe_phi_4 + erg_ausgabe_phi_5 +
13866     erg_ausgabe_alpha + erg_ausgabe_gamma + erg_ausgabe_x_s0 +
13867     erg_ausgabe_l_1 + erg_ausgabe_y_s0 + erg_ausgabe_psi_sk0 +
13868     erg_ausgabe_psi_0 + erg_ausgabe_psi_G;
13869
13870 anzeige->setText(text);
13871
13872 statusBar()->message( "Berechnen beendet", 2000 );
13873 }
13874
13875 void Opticurv::subcalculate()
13876 {
13877     if(Ba_Qudrant == 'y')
13878     {
13879         psi_sk = (2*(atan((B+(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/
13880             (A-C)))-(beta*rad));
13881         psi_sk_plus_deltaphi = (2*(atan((B+(sqrt(pow(A_plus_deltaphi,2)+
13882             pow(B,2)-pow(C_plus_deltaphi,2))))/
13883             (A_plus_deltaphi-C_plus_deltaphi)))-(beta*rad));
13884         psi_sk_minus_deltaphi = (2*(atan((B+(sqrt(pow(A_minus_deltaphi,2)+
13885             pow(B,2)-pow(C_minus_deltaphi,2))))/
13886             (A_minus_deltaphi-C_minus_deltaphi)))-(beta*rad));
13887     }
13888     else if(Ba_Qudrant == 'n')
13889     {
13890         psi_sk = (2*(atan((B-(sqrt(pow(A,2)+pow(B,2)-pow(C,2))))/
13891             (A-C)))-(beta*rad));
13892         psi_sk_plus_deltaphi = (2*(atan((B-(sqrt(pow(A_plus_deltaphi,2)+
13893             pow(B,2)-pow(C_plus_deltaphi,2))))/
13894             (A_plus_deltaphi-C_plus_deltaphi)))-(beta*rad));
13895         psi_sk_minus_deltaphi = (2*(atan((B-(sqrt(pow(A_minus_deltaphi,2)+
13896             pow(B,2)-pow(C_minus_deltaphi,2))))/
13897             (A_minus_deltaphi-C_minus_deltaphi)))-(beta*rad));
13898     }
13899     if(Ba_Qudrant == 'y')
13900     {
13901         psi_sk0 = (2*(atan((B+(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/
13902             (A_0-C_0)))-(beta*rad));

```

```

13903     }
13904     else if(Ba_Qudrant == 'n')
13905     {
13906         psi_sk0 = (2*(atan((B-(sqrt(pow(A_0,2)+pow(B,2)-pow(C_0,2))))/
13907             (A_0-C_0)))-(beta*rad));
13908     }
13909
13910     /* Einfluss der Bewegung des Schwinghebels */
13911
13912     if(positiv == 'y')
13913     {
13914         psi_rad = (psi_sk - psi_sk0);
13915         psi = psi_rad/rad;
13916         psi_plus_deltaphi = (psi_sk_plus_deltaphi - psi_sk0)/rad;
13917         psi_minus_deltaphi = (psi_sk_minus_deltaphi - psi_sk0)/rad;
13918     }
13919     else if(positiv == 'n')
13920     {
13921         psi_rad = (-psi_sk + psi_sk0);
13922         psi = psi_rad/rad;
13923         psi_plus_deltaphi = (-psi_sk_plus_deltaphi + psi_sk0)/rad;
13924         psi_minus_deltaphi = (-psi_sk_minus_deltaphi + psi_sk0)/rad;
13925     }
13926     psi_strich = ((psi_plus_deltaphi-psi_minus_deltaphi)/(2*deltaphi));
13927
13928     psi_zweistrich = ((psi_plus_deltaphi-2*psi+psi_minus_deltaphi)/
13929         ((pow((deltaphi),2))*rad));
13930
13931
13932     /* Berechnung von psi_0 */
13933
13934     psi_0 = (psi_sk0/rad);
13935
13936     /* Berechnung des Grundkreiswinkels psi_G */
13937
13938     psi_G = ((acos(((pow(l_3,2))+pow(l_1,2))-(pow(r_G,2)))/(2*l_1*l_3)))/rad);
13939
13940     if(indexKurve==1)      // Berechnung nur bei Doppelkurve
13941     {
13942         /* Berechnung des Abtriebswinkels psistar */
13943
13944         psistar = psi + alpha_R;
13945         psistar_rad = psi_rad + alpha_R*rad;
13946

```

```

13947      /* Berechnung des Grundkreiswinkels psi_Gstar */
13948
13949      psi_Gstar = psi_G + alpha_R;
13950
13951      /* Berechnung des Grundkreisradius r_Gstern */
13952
13953      r_Gstar = sqrt((pow(l_1,2))+(pow(l_3star,2))-2*l_1*l_3star*
13954                    (cos(psi_Gstar*rad)));
13955  }
13956
13957  /* Substitutionen für Koordinaten des Rollenmittelpunktes */
13958
13959  M = (1-cos((phi)*rad)+(l_3/l_1)*cos((psi_G+psi+phi)*rad));
13960  N = (-sin((phi)*rad)+(l_3/l_1)*sin((psi_G+psi+phi)*rad));
13961
13962  /* Koordinaten des Rollenmittelpunktes */
13963
13964  x_Bik = (x_A0*M-y_A0*N);
13965  y_Bik = (x_A0*N+y_A0*M);
13966
13967  if(indexKurve==1)      // Berechnung nur bei Doppelkurve
13968  {
13969      /* Substitutionen für Koordinaten des Rollenmittelpunktes der Gegenkurve */
13970
13971      Mstar = (1-cos((phi)*rad)+(l_3star/l_1)*cos((psi_Gstar+psi+phi)*rad));
13972      Nstar = (-sin((phi)*rad)+(l_3star/l_1)*sin((psi_Gstar+psi+phi)*rad));
13973
13974      /* Koordinaten des Rollenmittelpunktes der Gegenkurve */
13975
13976      x_Bikstar = (x_A0*Mstar-y_A0*Nstar);
13977      y_Bikstar = (x_A0*Nstar+y_A0*Mstar);
13978  }
13979
13980  /* Ableitungen x'_Bik und y'_Bik */
13981
13982  x_Bik_strich = ((x_A0*sin((phi)*rad)+y_A0*cos((phi)*rad)-(l_3/l_1)*
13983                (psi_strich+1)*(x_A0*sin((psi_G+psi+phi)*rad)+
13984                y_A0*cos((psi_G+psi+phi)*rad))));
13985  y_Bik_strich = ((x_A0*cos((phi)*rad)-y_A0*sin((phi)*rad)-(l_3/l_1)*
13986                (psi_strich+1)*(x_A0*cos((psi_G+psi+phi)*rad)-
13987                y_A0*sin((psi_G+psi+phi)*rad))));
13988
13989  if(indexKurve==1)      // Berechnung nur bei Doppelkurve
13990  {

```

```

13991      /* Ableitungen x'_Bik und y'_Bik der Gegenkurve */
13992
13993      x_Bik_strichstar = ((x_A0*sin((phi)*rad)+y_A0*cos((phi)*rad)-(l_3star/l_1)*
13994                          (psi_strich+1)*(x_A0*sin((psi_Gstar+psi+phi)*rad)+
13995                          y_A0*cos((psi_Gstar+psi+phi)*rad))));
13996      y_Bik_strichstar = ((x_A0*cos((phi)*rad)-y_A0*sin((phi)*rad)-(l_3star/l_1)*
13997                          (psi_strich+1)*(x_A0*cos((psi_Gstar+psi+phi)*rad)-
13998                          y_A0*sin((psi_Gstar+psi+phi)*rad))));
13999  }
14000
14001      /* Betrag des Tangentenvektors B' */
14002
14003      Betrag_B_strich = (sqrt((pow((fabs(x_Bik_strich)),2))+
14004                          (pow((fabs(y_Bik_strich)),2))));
14005
14006      if(indexKurve==1)      // Berechnung nur bei Doppelkurve
14007      {
14008          /* Betrag des Tangentenvektors B' der Gegenkurve */
14009
14010          Betrag_B_strichstar = (sqrt((pow((fabs(x_Bik_strichstar)),2))+
14011                                      (pow((fabs(y_Bik_strichstar)),2))));
14012      }
14013
14014
14015      /* innere Koordinaten */
14016
14017      x_i = (x_Bik+r_R*(y_Bik_strich/Betrag_B_strich));
14018      y_i = (y_Bik+r_R*(x_Bik_strich/Betrag_B_strich));
14019
14020      /* äüSSere Koordinaten */
14021
14022      x_a = (x_Bik-r_R*(y_Bik_strich/Betrag_B_strich));
14023      y_a = (y_Bik-r_R*(x_Bik_strich/Betrag_B_strich));
14024
14025      if(indexKurve==1)      // Berechnung nur bei Doppelkurve
14026      {
14027          /* innere Koordinaten der Gegenkurve */
14028
14029          x_istar = (x_Bikstar+r_Rstar*(y_Bik_strichstar/Betrag_B_strichstar));
14030          y_istar = (y_Bikstar+r_Rstar*(x_Bik_strichstar/Betrag_B_strichstar));
14031
14032          /* äüSSere Koordinaten der Gegenkurve */
14033
14034          x_astar = (x_Bikstar-r_Rstar*(y_Bik_strichstar/Betrag_B_strichstar));

```



```

14035     y_aster = (y_Bikstar-r_Rstar*(x_Bik_strichstar/Betrag_B_strichstar));
14036 }
14037
14038
14039 /* Koordinatensystem versetzen */
14040
14041 x_i_versetzt = (x_i-x_A0);
14042 y_i_versetzt = (y_i-y_A0);
14043 x_B_versetzt = (x_Bik-x_A0);
14044 y_B_versetzt = (y_Bik-y_A0);
14045 x_a_versetzt = (x_a-x_A0);
14046 y_a_versetzt = (y_a-y_A0);
14047
14048 if(indexKurve==1)      // Berechnung nur bei Doppelkurve
14049 {
14050     x_i_versetztstar = (x_istar-x_A0);
14051     y_i_versetztstar = (y_istar-y_A0);
14052     x_B_versetztstar = (x_Bikstar-x_A0);
14053     y_B_versetztstar = (y_Bikstar-y_A0);
14054     x_a_versetztstar = (x_aster-x_A0);
14055     y_a_versetztstar = (y_aster-y_A0);
14056 }
14057
14058 /* Berechnung von psi_0 */
14059
14060 psi_0 = (psi_sk0/rad);
14061
14062 /* Berechnung von mue */
14063
14064 r_Bik = (sqrt((pow((fabs(x_B_versetzt)),2))+
14065               (pow((fabs(y_B_versetzt)),2))));
14066
14067 r_Bik_strich = (sqrt((pow((fabs(x_Bik_strich)),2))+
14068                    (pow((fabs(y_Bik_strich)),2))));
14069
14070 zaehler_mue = (x_Bik_strich*y_B_versetzt+x_B_versetzt*y_Bik_strich);
14071
14072 nenner_mue = (r_Bik*r_Bik_strich);
14073
14074 smue = (fabs(zaehler_mue/nenner_mue));
14075
14076 mue = (fabs((((-pi)/2)+acos(smue))/rad));
14077
14078

```

```

14079      /* Berechnung von r_K */
14080
14081      Q = (cos(phi*rad)-(l_3/l_1)*(cos((psi_G+psi+phi)*rad))*
14082           (pow((fabs(psi_strich+1)),2))-(l_3/l_1)*
14083           (sin((psi_G+psi+phi)*rad))*psi_zweistrich);
14084
14085      R = (sin(phi*rad)-(l_3/l_1)*(sin((psi_G+psi+phi)*rad))*
14086           (pow((fabs(psi_strich+1)),2))-(l_3/l_1)*
14087           (cos((psi_G+psi+phi)*rad))*psi_zweistrich);
14088
14089      x_Bik_zweistrich = (x_A0*Q-y_A0*R);
14090
14091      S = (sin(phi*rad)-(l_3/l_1)*(sin((psi_G+psi+phi)*rad))*
14092           (pow((fabs(psi_strich+1)),2))+(l_3/l_1)*
14093           (cos((psi_G+psi+phi)*rad))*psi_zweistrich);
14094
14095      T = (cos(phi*rad)-(l_3/l_1)*(cos((psi_G+psi+phi)*rad))*
14096           (pow((fabs(psi_strich+1)),2))+(l_3/l_1)*
14097           (sin((psi_G+psi+phi)*rad))*psi_zweistrich);
14098
14099      y_Bik_zweistrich = (x_A0*S+y_A0*T);
14100
14101      wurzelausdruck = ((pow(x_Bik_strich, 2))+
14102                        (pow(y_Bik_strich, 2)));
14103
14104      potenz_wurzelausdruck = (pow(wurzelausdruck, 3));
14105
14106      zaehler_r_K = (sqrt(fabs(potenz_wurzelausdruck)));
14107
14108      nenner_r_K = (x_Bik_strich*y_Bik_zweistrich-
14109                   x_Bik_zweistrich*y_Bik_strich);
14110
14111      r_K = (zaehler_r_K/nenner_r_K);
14112
14113      Abstand = sqrt(pow((x_a_versetzt-x_i_versetzt),2)+
14114                    pow((y_a_versetzt-y_i_versetzt),2));
14115
14116      if( phi == 0 )
14117      {
14118          x_Ba = x_Bik;
14119      }
14120
14121      /* an der x-Achse spiegeln und um 180 Grad drehen */
14122

```

```

14123     if(scheibendreh_richtung == 'p')
14124     {
14125         /* Spiegeln */
14126
14127         x_i_versetzt = (-1)*x_i_versetzt;
14128         x_B_versetzt = (-1)*x_B_versetzt;
14129         x_a_versetzt = (-1)*x_a_versetzt;
14130
14131         if(indexKurve==1)      // Berechnung nur bei Doppelkurve
14132         {
14133             /* Spiegeln */
14134
14135             x_i_versetztstar = (-1)*x_i_versetztstar;
14136             x_B_versetztstar = (-1)*x_B_versetztstar;
14137             x_a_versetztstar = (-1)*x_a_versetztstar;
14138
14139         }
14140     }
14141
14142     /* Drehwinkel ermitteln */
14143
14144     if( phi == 0 )
14145     {
14146         /* 1. und 4. Quadrant */
14147
14148         if( ( (x_B_versetzt>=0) && (y_B_versetzt>=0) ) || ( (x_B_versetzt>=0) &&
14149             (y_B_versetzt<0) ) )
14150         {
14151             drehWinkel = (atan(y_B_versetzt/x_B_versetzt));
14152         }
14153
14154         /* 2. und 3. Quadrant */
14155
14156         if( ( (x_B_versetzt<0) && (y_B_versetzt>=0) ) || ( (x_B_versetzt<0) &&
14157             (y_B_versetzt<0) ) )
14158         {
14159             drehWinkel = (atan(y_B_versetzt/x_B_versetzt) + pi);
14160         }
14161     }
14162
14163     /* lokales Koordinatensystem in Lagerpunkt A_0 drehen */
14164
14165     x_i_versetzt_old = x_i_versetzt;
14166     y_i_versetzt_old = y_i_versetzt;

```

```

14167
14168     x_B_versetzt_old = x_B_versetzt;
14169     y_B_versetzt_old = y_B_versetzt;
14170
14171     x_a_versetzt_old = x_a_versetzt;
14172     y_a_versetzt_old = y_a_versetzt;
14173
14174     x_i_versetzt = x_i_versetzt_old*cos(drehWinkel) + y_i_versetzt_old*
14175                   sin(drehWinkel);
14176     y_i_versetzt = (-1)*x_i_versetzt_old*sin(drehWinkel) + y_i_versetzt_old*
14177                   cos(drehWinkel);
14178
14179     x_B_versetzt = x_B_versetzt_old*cos(drehWinkel) + y_B_versetzt_old*
14180                   sin(drehWinkel);
14181     y_B_versetzt = (-1)*x_B_versetzt_old*sin(drehWinkel) + y_B_versetzt_old*
14182                   cos(drehWinkel);
14183
14184     x_a_versetzt = x_a_versetzt_old*cos(drehWinkel) + y_a_versetzt_old*
14185                   sin(drehWinkel);
14186     y_a_versetzt = (-1)*x_a_versetzt_old*sin(drehWinkel) + y_a_versetzt_old*
14187                   cos(drehWinkel);
14188
14189     if(indexKurve==1)      // Berechnung nur bei Doppelkurve
14190     {
14191         x_i_versetztstar_old = x_i_versetztstar;
14192         y_i_versetztstar_old = y_i_versetztstar;
14193
14194         x_B_versetztstar_old = x_B_versetztstar;
14195         y_B_versetztstar_old = y_B_versetztstar;
14196
14197         x_a_versetztstar_old = x_a_versetztstar;
14198         y_a_versetztstar_old = y_a_versetztstar;
14199
14200         x_i_versetztstar = x_i_versetztstar_old*cos(drehWinkel) +
14201                           y_i_versetztstar_old*sin(drehWinkel);
14202         y_i_versetztstar = (-1)*x_i_versetztstar_old*sin(drehWinkel) +
14203                           y_i_versetztstar_old*cos(drehWinkel);
14204
14205         x_B_versetztstar = x_B_versetztstar_old*cos(drehWinkel) +
14206                           y_B_versetztstar_old*sin(drehWinkel);
14207         y_B_versetztstar = (-1)*x_B_versetztstar_old*sin(drehWinkel) +
14208                           y_B_versetztstar_old*cos(drehWinkel);
14209
14210         x_a_versetztstar = x_a_versetztstar_old*cos(drehWinkel) +

```

```

14211             y_a_versetztstar_old*sin(drehWinkel);
14212     y_a_versetztstar = (-1)*x_a_versetztstar_old*sin(drehWinkel) +
14213             y_a_versetztstar_old*cos(drehWinkel);
14214 }
14215
14216     /* Berechnung der Scheibendurchmesserkoordinaten */
14217
14218     x_s = r_S*cos(phi*rad);
14219     y_s = r_S*sin(phi*rad);
14220
14221     /* Berechnung der Wellendurchmesserkoordinaten */
14222
14223     x_W = r_W*cos(phi*rad);
14224     y_W = r_W*sin(phi*rad);
14225 }
14226
14227 void Opticurv::speichereASCII()
14228 {
14229     ofstream ofl;
14230
14231     if(indexKurve==0)
14232     {
14233         /* Datei für die Ausgabe öffnen ios::app --> neue Daten werden
14234             stets am Dateende angefügt */
14235         ofl.open(filename_asc, ios::app);
14236
14237         ofl << phi << ","
14238             << s << ","
14239             << s_strich << ","
14240             << psi << ","
14241             << psi_rad << ","
14242             << psi_strich << ","
14243             << psi_zweistrich << ","
14244             << x_Bik << ","
14245             << y_Bik << ","
14246             << x_Bik_strich << ","
14247             << y_Bik_strich << ","
14248             << x_i_versetzt << ","
14249             << y_i_versetzt << ","
14250             << x_B_versetzt << ","
14251             << y_B_versetzt << ","
14252             << x_a_versetzt << ","
14253             << y_a_versetzt << ","
14254             << Abstand << ","

```

```

14255         << mue << ","
14256         << r_K << ","
14257         << x_s << ","
14258         << y_s << ","
14259         << x_W << ","
14260         << y_W << endl;    // endl : Manipulator, erzeugt Zeilenvorschub
14261     ofl.close();
14262
14263     ofl.open(filename_innen, ios::app);
14264     ofl << x_i_versetzt << ","
14265         << y_i_versetzt << endl;
14266     ofl.close();
14267
14268     ofl.open(filename_mitte, ios::app);
14269     ofl << x_B_versetzt << ","
14270         << y_B_versetzt << endl;
14271     ofl.close();
14272
14273     ofl.open(filename_aussen, ios::app);
14274     ofl << x_a_versetzt << ","
14275         << y_a_versetzt << endl;
14276     ofl.close();
14277
14278     ofl.open(filename_s_phi, ios::app);
14279     ofl << phi << ","
14280         << s << endl;
14281     ofl.close();
14282 }
14283
14284 if(indexKurve==1)    // Berechnung nur bei Doppelkurve
14285 {
14286     /* Datei für die Ausgabe öffnen ios::app --> neue Daten werden
14287                                     stets am Dateiende angefügt */
14288     ofl.open(filename_asc, ios::app);
14289
14290     ofl << phi << ","
14291         << s << ","
14292         << s_strich << ","
14293         << psi << ","
14294         << psi_rad << ","
14295         << psi_strich << ","
14296         << psi_zweistrich << ","
14297         << x_Bik << ","
14298         << y_Bik << ","

```

```

14299         << x_Bik_strich << ","
14300         << y_Bik_strich << ","
14301         << x_i_versetzt << ","
14302         << y_i_versetzt << ","
14303         << x_B_versetzt << ","
14304         << y_B_versetzt << ","
14305         << x_a_versetzt << ","
14306         << y_a_versetzt << ","
14307         << psistar << ","
14308         << psistar_rad << ","
14309         << x_Bikstar << ","
14310         << y_Bikstar << ","
14311         << x_Bik_strichstar << ","
14312         << y_Bik_strichstar << ","
14313         << x_i_versetztstar << ","
14314         << y_i_versetztstar << ","
14315         << x_B_versetztstar << ","
14316         << y_B_versetztstar << ","
14317         << x_a_versetztstar << ","
14318         << y_a_versetztstar << ","
14319         << Abstand << ","
14320         << mue << ","
14321         << r_K << ","
14322         << x_s << ","
14323         << y_s << ","
14324         << x_W << ","
14325         << y_W << endl;    // endl : Manipulator, erzeugt Zeilenvorschub
14326 ofl.close();
14327
14328 ofl.open(filename_innen, ios::app);
14329 ofl << x_i_versetzt << ","
14330     << y_i_versetzt << endl;
14331 ofl.close();
14332
14333 ofl.open(filename_mitte, ios::app);
14334 ofl << x_B_versetzt << ","
14335     << y_B_versetzt << endl;
14336 ofl.close();
14337
14338 ofl.open(filename_aussen, ios::app);
14339 ofl << x_a_versetzt << ","
14340     << y_a_versetzt << endl;
14341 ofl.close();
14342

```

```

14343     ofl.open(filename_s_phi, ios::app);
14344     ofl << phi << ", "
14345         << s << endl;
14346     ofl.close();
14347 }
14348
14349 }
14350
14351
14352 void Opticurv::weiseZu_konvertiereTyp()
14353 {
14354     /* Zuweisung an die Vektorfelder fuer die grafische Ausgabe */
14355
14356     index_zuweisung++;
14357
14358     phigrafisch[index_zuweisung] = phi;
14359     x_B_verstzt[index_zuweisung] = x_B_versetzt;
14360     y_B_verstzt[index_zuweisung] = y_B_versetzt;
14361     x_i_verstzt[index_zuweisung] = x_i_versetzt;
14362     y_i_verstzt[index_zuweisung] = y_i_versetzt;
14363     x_a_verstzt[index_zuweisung] = x_a_versetzt;
14364     y_a_verstzt[index_zuweisung] = y_a_versetzt;
14365     sgrafisch[index_zuweisung] = s;
14366     s_strichgrafisch[index_zuweisung] = s_strich;
14367     psigrafisch[index_zuweisung] = psi_rad;
14368     psi_strichgrafisch[index_zuweisung] = psi_strich;
14369     psi_zweistrichgrafisch[index_zuweisung] = psi_zweistrich;
14370     x_Bikgrafisch[index_zuweisung] = x_Bik;
14371     y_Bikgrafisch[index_zuweisung] = y_Bik;
14372     x_Bik_strichgrafisch[index_zuweisung] = x_Bik_strich;
14373     y_Bik_strichgrafisch[index_zuweisung] = y_Bik_strich;
14374     x_sgrafisch[index_zuweisung] = x_s;
14375     y_sgrafisch[index_zuweisung] = y_s;
14376     x_Wgrafisch[index_zuweisung] = x_W;
14377     y_Wgrafisch[index_zuweisung] = y_W;
14378     muegrafisch[index_zuweisung] = mue;
14379     r_Kgrafisch[index_zuweisung] = r_K;
14380
14381     if(indexKurve==1)        // Zuweisung nur bei Doppelkurve
14382     {
14383         psistargrafisch[index_zuweisung] = psistar_rad;
14384         x_Bikstargrafisch[index_zuweisung] = x_Bikstar;
14385         y_Bikstargrafisch[index_zuweisung] = y_Bikstar;
14386         x_Bik_strichstargrafisch[index_zuweisung] = x_Bik_strichstar;

```



```

14387     y_Bik_strichstargrafisch[index_zuweisung] = y_Bik_strichstar;
14388     x_B_versetztstargrafisch[index_zuweisung] = x_B_versetztstar;
14389     y_B_versetztstargrafisch[index_zuweisung] = y_B_versetztstar;
14390     x_i_versetztstargrafisch[index_zuweisung] = x_i_versetztstar;
14391     y_i_versetztstargrafisch[index_zuweisung] = y_i_versetztstar;
14392     x_a_versetztstargrafisch[index_zuweisung] = x_a_versetztstar;
14393     y_a_versetztstargrafisch[index_zuweisung] = y_a_versetztstar;
14394 }
14395
14396
14397 /* implizite Typkonvertierungen für Ausgabe notw. */
14398
14399 dphi = phi;
14400 ds = s;
14401 ds_strich = s_strich;
14402 dpsi = psi;
14403 dpsi_rad = psi_rad;
14404 dpsi_strich_Grad = psi_strich;
14405 dpsi_zweistrich_Grad = psi_zweistrich;
14406 dx_Bik = x_Bik;
14407 dy_Bik = y_Bik;
14408 dx_Bik_strich = x_Bik_strich;
14409 dy_Bik_strich = y_Bik_strich;
14410 dx_i_versetzt = x_i_versetzt;
14411 dy_i_versetzt = y_i_versetzt;
14412 dx_B_versetzt = x_B_versetzt;
14413 dy_B_versetzt = y_B_versetzt;
14414 dx_a_versetzt = x_a_versetzt;
14415 dy_a_versetzt = y_a_versetzt;
14416 dr_Bik = r_Bik;
14417 dmue = mue;
14418 dsmue = smue;
14419 dr_K = r_K;
14420 dabstand = Abstand;
14421
14422 if(indexKurve==1)      // Konvertierung nur bei Doppelkurve
14423 {
14424     dpsistar = psistar;
14425     dpsistar_rad = psistar_rad;
14426     dx_Bikstar = x_Bikstar;
14427     dy_Bikstar = y_Bikstar;
14428     dx_Bik_strichstar = x_Bik_strichstar;
14429     dy_Bik_strichstar = y_Bik_strichstar;
14430     dx_i_versetztstar = x_i_versetztstar;

```

```

14431     dy_i_versetztstar = y_i_versetztstar;
14432     dx_B_versetztstar = x_B_versetztstar;
14433     dy_B_versetztstar = y_B_versetztstar;
14434     dx_a_versetztstar = x_a_versetztstar;
14435     dy_a_versetztstar = y_a_versetztstar;
14436 }
14437 }
14438
14439 void Opticurv::ergebnisseMainWindow()
14440 {
14441     if(indexKurve==0)
14442     {
14443         QString erg_ausgabe_phi=anzeige->text();
14444         erg_ausgabe_phi.setNum( dphi );
14445         static const char* text_phi_a[] = {"<tr>"
14446                                             "<td>" ,0};
14447         static const char* text_phi_b[] = {"</td>" ,0};
14448         text+=text_phi_a[0]+erg_ausgabe_phi+text_phi_b[0];
14449
14450         QString erg_ausgabe_s=anzeige->text();
14451         erg_ausgabe_s.setNum( ds );
14452         static const char* text_a[] = {"<td>" ,0};
14453         static const char* text_b[] = {"</td>" ,0};
14454         text+=text_a[0]+erg_ausgabe_s+text_b[0];
14455
14456         QString erg_ausgabe_s_strich=anzeige->text();
14457         erg_ausgabe_s_strich.setNum( ds_strich );
14458         text+=text_a[0]+erg_ausgabe_s_strich+text_b[0];
14459
14460         QString erg_ausgabe_psi=anzeige->text();
14461         erg_ausgabe_psi.setNum( dpsi );
14462         text+=text_a[0]+erg_ausgabe_psi+text_b[0];
14463
14464         QString erg_ausgabe_psi_rad=anzeige->text();
14465         erg_ausgabe_psi_rad.setNum( dpsi_rad );
14466         text+=text_a[0]+erg_ausgabe_psi_rad+text_b[0];
14467
14468         QString erg_ausgabe_psi_strich_Grad=anzeige->text();
14469         erg_ausgabe_psi_strich_Grad.setNum( dpsi_strich_Grad );
14470         text+=text_a[0]+erg_ausgabe_psi_strich_Grad+text_b[0];
14471
14472         QString erg_ausgabe_psi_zweistrich_Grad=anzeige->text();
14473         erg_ausgabe_psi_zweistrich_Grad.setNum( dpsi_zweistrich_Grad );
14474         text+=text_a[0]+erg_ausgabe_psi_zweistrich_Grad+text_b[0];

```

```

14475
14476     QString erg_ausgabe_x_Bik=anzeige->text();
14477     erg_ausgabe_x_Bik.setNum( dx_Bik );
14478     text+=text_a[0]+erg_ausgabe_x_Bik+text_b[0];
14479
14480     QString erg_ausgabe_y_Bik=anzeige->text();
14481     erg_ausgabe_y_Bik.setNum( dy_Bik );
14482     text+=text_a[0]+erg_ausgabe_y_Bik+text_b[0];
14483
14484     QString erg_ausgabe_x_Bik_strich=anzeige->text();
14485     erg_ausgabe_x_Bik_strich.setNum( dx_Bik_strich );
14486     text+=text_a[0]+erg_ausgabe_x_Bik_strich+text_b[0];
14487
14488     QString erg_ausgabe_y_Bik_strich=anzeige->text();
14489     erg_ausgabe_y_Bik_strich.setNum( dy_Bik_strich );
14490     text+=text_a[0]+erg_ausgabe_y_Bik_strich+text_b[0];
14491
14492     QString erg_ausgabe_x_i_versetzt=anzeige->text();
14493     erg_ausgabe_x_i_versetzt.setNum( dx_i_versetzt );
14494     text+=text_a[0]+erg_ausgabe_x_i_versetzt+text_b[0];
14495
14496     QString erg_ausgabe_y_i_versetzt=anzeige->text();
14497     erg_ausgabe_y_i_versetzt.setNum( dy_i_versetzt );
14498     text+=text_a[0]+erg_ausgabe_y_i_versetzt+text_b[0];
14499
14500     QString erg_ausgabe_x_B_versetzt=anzeige->text();
14501     erg_ausgabe_x_B_versetzt.setNum( dx_B_versetzt );
14502     text+=text_a[0]+erg_ausgabe_x_B_versetzt+text_b[0];
14503
14504     QString erg_ausgabe_y_B_versetzt=anzeige->text();
14505     erg_ausgabe_y_B_versetzt.setNum( dy_B_versetzt );
14506     text+=text_a[0]+erg_ausgabe_y_B_versetzt+text_b[0];
14507
14508     QString erg_ausgabe_x_a_versetzt=anzeige->text();
14509     erg_ausgabe_x_a_versetzt.setNum( dx_a_versetzt );
14510     text+=text_a[0]+erg_ausgabe_x_a_versetzt+text_b[0];
14511
14512     QString erg_ausgabe_y_a_versetzt=anzeige->text();
14513     erg_ausgabe_y_a_versetzt.setNum( dy_a_versetzt );
14514     text+=text_a[0]+erg_ausgabe_y_a_versetzt+text_b[0];
14515
14516     QString erg_abstand=anzeige->text();
14517     erg_abstand.setNum( dabstand );
14518     text+=text_a[0]+erg_abstand+text_b[0];

```

```

14519
14520     QString erg_ausgabe_mue=anzeige->text();
14521     erg_ausgabe_mue.setNum( dmue );
14522     text+=text_a[0]+erg_ausgabe_mue+text_b[0];
14523
14524     QString erg_ausgabe_r_K=anzeige->text();
14525     erg_ausgabe_r_K.setNum( dr_K );
14526     static const char* text_r_K_a[] = {"<td>" ,0};
14527     static const char* text_r_K_b[] = {"</td>"
14528                                     "</tr>" ,0};
14529     text+=text_r_K_a[0]+erg_ausgabe_r_K+
14530         text_r_K_b[0];
14531 }
14532 if(indexKurve==1)      // Konvertierung nur bei Doppelkurve
14533 {
14534     QString erg_ausgabe_phi=anzeige->text();
14535     erg_ausgabe_phi.setNum( dphi );
14536     static const char* text_phi_a[] = {"<tr>"
14537                                     "<td>" ,0};
14538     static const char* text_phi_b[] = {"</td>" ,0};
14539     text+=text_phi_a[0]+erg_ausgabe_phi+text_phi_b[0];
14540
14541     QString erg_ausgabe_s=anzeige->text();
14542     erg_ausgabe_s.setNum( ds );
14543     static const char* text_a[] = {"<td>" ,0};
14544     static const char* text_b[] = {"</td>" ,0};
14545     text+=text_a[0]+erg_ausgabe_s+text_b[0];
14546
14547     QString erg_ausgabe_s_strich=anzeige->text();
14548     erg_ausgabe_s_strich.setNum( ds_strich );
14549     text+=text_a[0]+erg_ausgabe_s_strich+text_b[0];
14550
14551     QString erg_ausgabe_psi=anzeige->text();
14552     erg_ausgabe_psi.setNum( dpsi );
14553     text+=text_a[0]+erg_ausgabe_psi+text_b[0];
14554
14555     QString erg_ausgabe_psi_rad=anzeige->text();
14556     erg_ausgabe_psi_rad.setNum( dpsi_rad );
14557     text+=text_a[0]+erg_ausgabe_psi_rad+text_b[0];
14558
14559     QString erg_ausgabe_psi_strich_Grad=anzeige->text();
14560     erg_ausgabe_psi_strich_Grad.setNum( dpsi_strich_Grad );
14561     text+=text_a[0]+erg_ausgabe_psi_strich_Grad+text_b[0];
14562

```

```

14563   QString erg_ausgabe_psi_zweistrich_Grad=anzeige->text();
14564   erg_ausgabe_psi_zweistrich_Grad.setNum( dpsi_zweistrich_Grad );
14565   text+=text_a[0]+erg_ausgabe_psi_zweistrich_Grad+text_b[0];
14566
14567   QString erg_ausgabe_x_Bik=anzeige->text();
14568   erg_ausgabe_x_Bik.setNum( dx_Bik );
14569   text+=text_a[0]+erg_ausgabe_x_Bik+text_b[0];
14570
14571   QString erg_ausgabe_y_Bik=anzeige->text();
14572   erg_ausgabe_y_Bik.setNum( dy_Bik );
14573   text+=text_a[0]+erg_ausgabe_y_Bik+text_b[0];
14574
14575   QString erg_ausgabe_x_Bik_strich=anzeige->text();
14576   erg_ausgabe_x_Bik_strich.setNum( dx_Bik_strich );
14577   text+=text_a[0]+erg_ausgabe_x_Bik_strich+text_b[0];
14578
14579   QString erg_ausgabe_y_Bik_strich=anzeige->text();
14580   erg_ausgabe_y_Bik_strich.setNum( dy_Bik_strich );
14581   text+=text_a[0]+erg_ausgabe_y_Bik_strich+text_b[0];
14582
14583   QString erg_ausgabe_x_i_versetzt=anzeige->text();
14584   erg_ausgabe_x_i_versetzt.setNum( dx_i_versetzt );
14585   text+=text_a[0]+erg_ausgabe_x_i_versetzt+text_b[0];
14586
14587   QString erg_ausgabe_y_i_versetzt=anzeige->text();
14588   erg_ausgabe_y_i_versetzt.setNum( dy_i_versetzt );
14589   text+=text_a[0]+erg_ausgabe_y_i_versetzt+text_b[0];
14590
14591   QString erg_ausgabe_x_B_versetzt=anzeige->text();
14592   erg_ausgabe_x_B_versetzt.setNum( dx_B_versetzt );
14593   text+=text_a[0]+erg_ausgabe_x_B_versetzt+text_b[0];
14594
14595   QString erg_ausgabe_y_B_versetzt=anzeige->text();
14596   erg_ausgabe_y_B_versetzt.setNum( dy_B_versetzt );
14597   text+=text_a[0]+erg_ausgabe_y_B_versetzt+text_b[0];
14598
14599   QString erg_ausgabe_x_a_versetzt=anzeige->text();
14600   erg_ausgabe_x_a_versetzt.setNum( dx_a_versetzt );
14601   text+=text_a[0]+erg_ausgabe_x_a_versetzt+text_b[0];
14602
14603   QString erg_ausgabe_y_a_versetzt=anzeige->text();
14604   erg_ausgabe_y_a_versetzt.setNum( dy_a_versetzt );
14605   text+=text_a[0]+erg_ausgabe_y_a_versetzt+text_b[0];
14606

```

```

14607   QString erg_ausgabe_psistar=anzeige->text();
14608   erg_ausgabe_psistar.setNum( dpsistar );
14609   text+=text_a[0]+erg_ausgabe_psistar+text_b[0];
14610
14611   QString erg_ausgabe_psistar_rad=anzeige->text();
14612   erg_ausgabe_psistar_rad.setNum( dpsistar_rad );
14613   text+=text_a[0]+erg_ausgabe_psistar_rad+text_b[0];
14614
14615   QString erg_ausgabe_x_Bikstar=anzeige->text();
14616   erg_ausgabe_x_Bikstar.setNum( dx_Bikstar );
14617   text+=text_a[0]+erg_ausgabe_x_Bikstar+text_b[0];
14618
14619   QString erg_ausgabe_y_Bikstar=anzeige->text();
14620   erg_ausgabe_y_Bikstar.setNum( dy_Bikstar );
14621   text+=text_a[0]+erg_ausgabe_y_Bikstar+text_b[0];
14622
14623   QString erg_ausgabe_x_Bik_strichstar=anzeige->text();
14624   erg_ausgabe_x_Bik_strichstar.setNum( dx_Bik_strichstar );
14625   text+=text_a[0]+erg_ausgabe_x_Bik_strichstar+text_b[0];
14626
14627   QString erg_ausgabe_y_Bik_strichstar=anzeige->text();
14628   erg_ausgabe_y_Bik_strichstar.setNum( dy_Bik_strichstar );
14629   text+=text_a[0]+erg_ausgabe_y_Bik_strichstar+text_b[0];
14630
14631   QString erg_ausgabe_x_i_versetztstar=anzeige->text();
14632   erg_ausgabe_x_i_versetztstar.setNum( dx_i_versetztstar );
14633   text+=text_a[0]+erg_ausgabe_x_i_versetztstar+text_b[0];
14634
14635   QString erg_ausgabe_y_i_versetztstar=anzeige->text();
14636   erg_ausgabe_y_i_versetztstar.setNum( dy_i_versetztstar );
14637   text+=text_a[0]+erg_ausgabe_y_i_versetztstar+text_b[0];
14638
14639   QString erg_ausgabe_x_B_versetztstar=anzeige->text();
14640   erg_ausgabe_x_B_versetztstar.setNum( dx_B_versetztstar );
14641   text+=text_a[0]+erg_ausgabe_x_B_versetztstar+text_b[0];
14642
14643   QString erg_ausgabe_y_B_versetztstar=anzeige->text();
14644   erg_ausgabe_y_B_versetztstar.setNum( dy_B_versetztstar );
14645   text+=text_a[0]+erg_ausgabe_y_B_versetztstar+text_b[0];
14646
14647   QString erg_ausgabe_x_a_versetztstar=anzeige->text();
14648   erg_ausgabe_x_a_versetztstar.setNum( dx_a_versetztstar );
14649   text+=text_a[0]+erg_ausgabe_x_a_versetztstar+text_b[0];
14650

```

```

14651     QString erg_ausgabe_y_a_versetztstar=anzeige->text();
14652     erg_ausgabe_y_a_versetztstar.setNum( dy_a_versetztstar );
14653     text+=text_a[0]+erg_ausgabe_y_a_versetztstar+text_b[0];
14654
14655     QString erg_abstand=anzeige->text();
14656     erg_abstand.setNum( dabstand );
14657     text+=text_a[0]+erg_abstand+text_b[0];
14658
14659     QString erg_ausgabe_mue=anzeige->text();
14660     erg_ausgabe_mue.setNum( dmue );
14661     text+=text_a[0]+erg_ausgabe_mue+text_b[0];
14662
14663     QString erg_ausgabe_r_K=anzeige->text();
14664     erg_ausgabe_r_K.setNum( dr_K );
14665     static const char* text_r_K_a[] = {"<td>" ,0};
14666     static const char* text_r_K_b[] = {"</td>"
14667                                         "</tr>" ,0};
14668     text+=text_r_K_a[0]+erg_ausgabe_r_K+
14669           text_r_K_b[0];
14670 }
14671 }
14672 void Opticurv::plotArbeitsundGegenkurve()
14673 {
14674     valuePair v, u, r, s;
14675     int      i=0, z=0;
14676     int index;
14677     elemente = (360/n_1);
14678
14679     /* Zaehlen der Werte */
14680
14681     for ( index=0; index<=elemente; index++ )
14682         z++;
14683
14684     plotWindow = new FunctionPlot( z, dPlotSkalierFaktor );
14685     plotWindow->setBackgroundColor(Qt::white);
14686     plotWindow->setGeometry( 150, 150, dPlotSkalierFaktor*500,
14687                             dPlotSkalierFaktor*500 );
14688     plotWindow->setCaption("Grafikfenster");
14689
14690     if( konturAussenAussen->isChecked() )
14691     {
14692         for ( index=0; index<=elemente; index++ )
14693         {
14694             v.x=x_B_verstzt[index];

```

```

14695         v.y=y_B_verstzt[index];
14696
14697         plotWindow->setValueKurve5( i, v );
14698
14699         u.x=x_B_versetztstargrafisch[index];
14700         u.y=y_B_versetztstargrafisch[index];
14701
14702         plotWindow->setValueKurve6( i, u );
14703
14704         r.x=x_i_verstzt[index];
14705         r.y=y_i_verstzt[index];
14706
14707         plotWindow->setValueKurve9( i, r );
14708
14709         s.x=x_i_versetztstargrafisch[index];
14710         s.y=y_i_versetztstargrafisch[index];
14711
14712         plotWindow->setValueKurve10( i, s );
14713
14714         i++;
14715     }
14716     if( dmanuelleArbeitskurven == 1 )
14717     {
14718         plotWindow->plotIt_manuell( dx_minArbkurven, dx_maxArbkurven,
14719             ddxArbkurven, dy_minArbkurven, dy_maxArbkurven, ddyArbkurven );
14720     }
14721     else if( dmanuelleArbeitskurven == 2 )
14722     {
14723         plotWindow->plotIt_DK_aa();
14724     }
14725 }
14726 if( konturAussenInnen->isChecked() )
14727 {
14728     for ( index=0; index<=elemente; index++ )
14729     {
14730         v.x=x_B_verstzt[index];
14731         v.y=y_B_verstzt[index];
14732
14733         plotWindow->setValueKurve5( i, v );
14734
14735         u.x=x_B_versetztstargrafisch[index];
14736         u.y=y_B_versetztstargrafisch[index];
14737
14738         plotWindow->setValueKurve6( i, u );

```



```

14739
14740     r.x=x_i_verstzt[index];
14741     r.y=y_i_verstzt[index];
14742
14743     plotWindow->setValueKurve9( i, r );
14744
14745     s.x=x_a_versetztstargrafisch[index];
14746     s.y=y_a_versetztstargrafisch[index];
14747
14748     plotWindow->setValueKurve10( i, s );
14749
14750     i++;
14751 }
14752 if( dmanuelleArbeitskurven == 1 )
14753 {
14754     plotWindow->plotIt_manuell( dx_minArbkurven, dx_maxArbkurven,
14755         ddxArbkurven, dy_minArbkurven, dy_maxArbkurven, ddyArbkurven );
14756 }
14757 else if( dmanuelleArbeitskurven == 2 )
14758 {
14759     plotWindow->plotIt_DK_ai();
14760 }
14761 }
14762 if( konturInnenAussen->isChecked() )
14763 {
14764     for ( index=0; index<=elemente; index++ )
14765     {
14766         v.x=x_B_verstzt[index];
14767         v.y=y_B_verstzt[index];
14768
14769         plotWindow->setValueKurve5( i, v );
14770
14771         u.x=x_B_versetztstargrafisch[index];
14772         u.y=y_B_versetztstargrafisch[index];
14773
14774         plotWindow->setValueKurve6( i, u );
14775
14776         r.x=x_a_verstzt[index];
14777         r.y=y_a_verstzt[index];
14778
14779         plotWindow->setValueKurve9( i, r );
14780
14781         s.x=x_i_versetztstargrafisch[index];
14782         s.y=y_i_versetztstargrafisch[index];

```

```

14783
14784     plotWindow->setValueKurve10( i, s );
14785
14786     i++;
14787 }
14788 if( dmanuelleArbeitskurven == 1 )
14789 {
14790     plotWindow->plotIt_manuell( dx_minArbkurven, dx_maxArbkurven,
14791         ddxArbkurven, dy_minArbkurven, dy_maxArbkurven, ddyArbkurven );
14792 }
14793 else if( dmanuelleArbeitskurven == 2 )
14794 {
14795     plotWindow->plotIt_DK_ia();
14796 }
14797 }
14798 plotWindow->show();
14799 }
14800
14801 void Opticurv::plot0()    // bei Nutkurve
14802 {
14803     valuePair v, u, w, m, o;
14804     int      i=0, z=0;
14805     int index;
14806     elemente = (360/n_1);
14807
14808     /* Zaehlen der Werte */
14809
14810     for ( index=0; index<=elemente; index++ )
14811         z++;
14812
14813     plotWindow = new FunctionPlot( z, dPlotSkalierFaktor );
14814     plotWindow->setBackgroundColor(Qt::white);
14815     plotWindow->setGeometry( 150, 150, dPlotSkalierFaktor*500,
14816         dPlotSkalierFaktor*500 );
14817     plotWindow->setCaption("Grafikfenster");
14818
14819     if ( radiobuttonArbeitskurven->isChecked() )
14820     {
14821         for ( index=0; index<=elemente; index++ )
14822         {
14823             v.x=x_B_verstzt[index];
14824             v.y=y_B_verstzt[index];
14825
14826             plotWindow->setValueKurve5( i, v );

```

```

14827
14828     u.x=x_i_verstzt[index];
14829     u.y=y_i_verstzt[index];
14830
14831     plotWindow->setValueKurve6( i, u );
14832
14833     w.x=x_a_verstzt[index];
14834     w.y=y_a_verstzt[index];
14835
14836     plotWindow->setValueKurve7( i, w );
14837     i++;
14838 }
14839 if( dmanuelleArbeitskurven == 1 )
14840 {
14841     plotWindow->plotIt_manuell( dx_minArbkurven, dx_maxArbkurven,
14842         ddxArbkurven, dy_minArbkurven, dy_maxArbkurven, ddyArbkurven );
14843 }
14844 else if( dmanuelleArbeitskurven == 2 )
14845 {
14846     plotWindow->plotIt_NK();
14847 }
14848 }
14849 if ( radiobuttonHub_geschw->isChecked() )
14850 {
14851     for ( index=1; index<=elemente; index++ )
14852     {
14853         v.x=phigrafisch[index];
14854         v.y=sgrafisch[index];
14855
14856         plotWindow->setValueKurve1( i, v );
14857
14858         w.x=phigrafisch[index];
14859         w.y=s_strichgrafisch[index];
14860
14861         plotWindow->setValueKurve3( i, w );
14862
14863         i++;
14864     }
14865     if( dmanuelles_phi == 1 )
14866     {
14867         plotWindow->plotIt_manuell( dx_mins_phi, dx_maxs_phi,
14868             ddxs_phi, dy_mins_phi, dy_maxs_phi, ddys_phi );
14869     }
14870     else if( dmanuelles_phi == 2 )

```

```

14871     {
14872         plotWindow->plotIt_Hub_geschw();
14873     }
14874 }
14875 if ( radiobuttonpsi_geschw_beschl->isChecked() )
14876 {
14877     for ( index=1; index<=elemente; index++ )
14878     {
14879         v.x=phigrafisch[index];
14880         v.y=psigrafisch[index];
14881
14882         plotWindow->setValueKurve1( i, v );
14883
14884         w.x=phigrafisch[index];
14885         w.y=psi_strichgrafisch[index];
14886
14887         plotWindow->setValueKurve3( i, w );
14888
14889         m.x=phigrafisch[index];
14890         m.y=psi_zweistrichgrafisch[index];
14891
14892         plotWindow->setValueKurve4( i, m );
14893
14894         i++;
14895     }
14896     if( dmanuellepsi_phi == 1 )
14897     {
14898         plotWindow->plotIt_manuell( dx_minpsi_phi, dx_maxpsi_phi,
14899             ddxpsi_phi, dy_minpsi_phi, dy_maxpsi_phi, ddypsi_phi );
14900     }
14901     else if( dmanuellepsi_phi == 2 )
14902     {
14903         plotWindow->plotIt_psi_geschw_beschl();
14904     }
14905 }
14906 if ( radiobuttonx_Bik_x_Bikstrich->isChecked() )
14907 {
14908     for ( index=1; index<=elemente; index++ )
14909     {
14910         v.x=phigrafisch[index];
14911         v.y=x_Bikgrafisch[index];
14912
14913         plotWindow->setValueKurve1( i, v );
14914

```

```

14915         w.x=phigrafisch[index];
14916         w.y=x_Bik_strichgrafisch[index];
14917
14918         plotWindow->setValueKurve3( i, w );
14919
14920         i++;
14921     }
14922     if( dmanuellex_Bik == 1 )
14923     {
14924         plotWindow->plotIt_manuell( dx_minx_Bik, dx_maxx_Bik,
14925             ddxx_Bik, dy_minx_Bik, dy_maxx_Bik, ddyx_Bik );
14926     }
14927     else if( dmanuellex_Bik == 2 )
14928     {
14929         plotWindow->plotIt_xBik();
14930     }
14931 }
14932 if ( radiobuttony_Bik_y_Bikstrich->isChecked() )
14933 {
14934     for ( index=1; index<=elemente; index++ )
14935     {
14936         v.x=phigrafisch[index];
14937         v.y=y_Bikgrafisch[index];
14938
14939         plotWindow->setValueKurve1( i, v );
14940
14941         w.x=phigrafisch[index];
14942         w.y=y_Bik_strichgrafisch[index];
14943
14944         plotWindow->setValueKurve3( i, w );
14945
14946         i++;
14947     }
14948     if( dmanuelley_Bik == 1 )
14949     {
14950         plotWindow->plotIt_manuell( dx_miny_Bik, dx_maxy_Bik,
14951             ddxy_Bik, dy_miny_Bik, dy_maxy_Bik, ddy_Bik );
14952     }
14953     else if( dmanuelley_Bik == 2 )
14954     {
14955         plotWindow->plotIt_yBik();
14956     }
14957 }
14958 if ( radiobuttondurchmesser->isChecked() )

```

```

14959     {
14960         for ( index=0; index<=elemente; index++ )
14961         {
14962             v.x=x_B_verstzt[index];
14963             v.y=y_B_verstzt[index];
14964
14965             plotWindow->setValueKurve5( i, v );
14966
14967             u.x=x_i_verstzt[index];
14968             u.y=y_i_verstzt[index];
14969
14970             plotWindow->setValueKurve6( i, u );
14971
14972             w.x=x_a_verstzt[index];
14973             w.y=y_a_verstzt[index];
14974
14975             plotWindow->setValueKurve7( i, w );
14976
14977             m.x=x_sgrafisch[index];
14978             m.y=y_sgrafisch[index];
14979
14980             plotWindow->setValueKurve8( i, m );
14981
14982             o.x=x_Wgrafisch[index];
14983             o.y=y_Wgrafisch[index];
14984
14985             plotWindow->setValueKurve11( i, o );
14986             i++;
14987         }
14988         if( dmanuelleKurvenscheibe == 1 )
14989         {
14990             plotWindow->plotIt_manuell( dx_minKurvenscheibe, dx_maxKurvenscheibe,
14991                 ddxKurvenscheibe, dy_minKurvenscheibe, dy_maxKurvenscheibe,
14992                 ddyKurvenscheibe );
14993         }
14994         else if( dmanuelleKurvenscheibe == 2 )
14995         {
14996             plotWindow->plotIt_durchmesser();
14997         }
14998     }
14999     if ( radiobuttonmue->isChecked() )
15000     {
15001         for ( index=1; index<=elemente; index++ )
15002         {

```

```

15003         v.x=phigrafisch[index];
15004         v.y=muegrafisch[index];
15005
15006         plotWindow->setValueKurve1( i, v );
15007         i++;
15008     }
15009     if( dmanuelleUebertragungswinkel == 1 )
15010     {
15011         plotWindow->plotIt_manuell( dx_minUebertragungswinkel,
15012             dx_maxUebertragungswinkel, ddxUebertragungswinkel,
15013             dy_minUebertragungswinkel, dy_maxUebertragungswinkel,
15014             ddyUebertragungswinkel );
15015     }
15016     else if( dmanuelleUebertragungswinkel == 2 )
15017     {
15018         plotWindow->plotIt_mue();
15019     }
15020 }
15021 if ( radiobuttonr_K->isChecked() )
15022 {
15023     for ( index=1; index<=elemente; index++ )
15024     {
15025         v.x=phigrafisch[index];
15026         v.y=r_Kgrafisch[index];
15027
15028         plotWindow->setValueKurve1( i, v );
15029         i++;
15030     }
15031     if( dmanuelleKruemmungsradius == 1 )
15032     {
15033         plotWindow->plotIt_manuell( dx_minKruemmungsradius,
15034             dx_maxKruemmungsradius, ddxKruemmungsradius,
15035             dy_minKruemmungsradius, dy_maxKruemmungsradius,
15036             ddyKruemmungsradius );
15037     }
15038     else if( dmanuelleKruemmungsradius == 2 )
15039     {
15040         plotWindow->plotIt_rK();
15041     }
15042 }
15043 plotWindow->show();
15044 }
15045 void Opticurv::plot1() // bei Doppelkurve
15046 {

```

```

15047     valuePair v, u, w, m, r, s, o;
15048     int      i=0, z=0;
15049     int index;
15050     elemente = (360/n_1);
15051
15052     /* Zaehlen der Werte*/
15053
15054     for ( index=0; index<=elemente; index++ )
15055         z++;
15056
15057     plotWindow = new FunctionPlot( z, dPlotSkalierFaktor );
15058     plotWindow->setBackgroundColor(Qt::white);
15059     plotWindow->setGeometry( 150, 150, dPlotSkalierFaktor*500,
15060         dPlotSkalierFaktor*500 );
15061     plotWindow->setCaption("Grafikfenster");
15062
15063     if ( radiobuttonArbeitskurven->isChecked() )
15064     {
15065         if( konturAussenAussen->isChecked() )
15066         {
15067             for ( index=0; index<=elemente; index++ )
15068             {
15069                 v.x=x_B_verstzt[index];
15070                 v.y=y_B_verstzt[index];
15071
15072                 plotWindow->setValueKurve5( i, v );
15073
15074                 u.x=x_B_versetztstargrafisch[index];
15075                 u.y=y_B_versetztstargrafisch[index];
15076
15077                 plotWindow->setValueKurve6( i, u );
15078
15079                 r.x=x_i_verstzt[index];
15080                 r.y=y_i_verstzt[index];
15081
15082                 plotWindow->setValueKurve9( i, r );
15083
15084                 s.x=x_i_versetztstargrafisch[index];
15085                 s.y=y_i_versetztstargrafisch[index];
15086
15087                 plotWindow->setValueKurve10( i, s );
15088
15089                 i++;
15090             }

```



```

15091     if( dmanuelleArbeitskurven == 1 )
15092     {
15093         plotWindow->plotIt_manuell( dx_minArbkurven, dx_maxArbkurven,
15094             ddxArbkurven, dy_minArbkurven, dy_maxArbkurven, ddyArbkurven );
15095     }
15096     else if( dmanuelleArbeitskurven == 2 )
15097     {
15098         plotWindow->plotIt_DK_aa();
15099     }
15100 }
15101 if( konturAussenInnen->isChecked() )
15102 {
15103     for ( index=0; index<=elemente; index++ )
15104     {
15105         v.x=x_B_verstzt[index];
15106         v.y=y_B_verstzt[index];
15107
15108         plotWindow->setValueKurve5( i, v );
15109
15110         u.x=x_B_versetztstargrafisch[index];
15111         u.y=y_B_versetztstargrafisch[index];
15112
15113         plotWindow->setValueKurve6( i, u );
15114
15115         r.x=x_i_verstzt[index];
15116         r.y=y_i_verstzt[index];
15117
15118         plotWindow->setValueKurve9( i, r );
15119
15120         s.x=x_a_versetztstargrafisch[index];
15121         s.y=y_a_versetztstargrafisch[index];
15122
15123         plotWindow->setValueKurve10( i, s );
15124
15125         i++;
15126     }
15127     if( dmanuelleArbeitskurven == 1 )
15128     {
15129         plotWindow->plotIt_manuell( dx_minArbkurven, dx_maxArbkurven,
15130             ddxArbkurven, dy_minArbkurven, dy_maxArbkurven, ddyArbkurven );
15131     }
15132     else if( dmanuelleArbeitskurven == 2 )
15133     {
15134         plotWindow->plotIt_DK_ai();

```

```

15135     }
15136 }
15137 if( konturInnenAussen->isChecked() )
15138 {
15139     for ( index=0; index<=elemente; index++ )
15140     {
15141         v.x=x_B_verstzt[index];
15142         v.y=y_B_verstzt[index];
15143
15144         plotWindow->setValueKurve5( i, v );
15145
15146         u.x=x_B_versetztstargrafisch[index];
15147         u.y=y_B_versetztstargrafisch[index];
15148
15149         plotWindow->setValueKurve6( i, u );
15150
15151         r.x=x_a_verstzt[index];
15152         r.y=y_a_verstzt[index];
15153
15154         plotWindow->setValueKurve9( i, r );
15155
15156         s.x=x_i_versetztstargrafisch[index];
15157         s.y=y_i_versetztstargrafisch[index];
15158
15159         plotWindow->setValueKurve10( i, s );
15160
15161         i++;
15162     }
15163     if( dmanuelleArbeitskurven == 1 )
15164     {
15165         plotWindow->plotIt_manuell( dx_minArbkurven, dx_maxArbkurven,
15166             ddxArbkurven, dy_minArbkurven, dy_maxArbkurven, ddyArbkurven );
15167     }
15168     else if( dmanuelleArbeitskurven == 2 )
15169     {
15170         plotWindow->plotIt_DK_ia();
15171     }
15172 }
15173 }
15174 if ( radiobuttonHub_geschw->isChecked() )
15175 {
15176     for ( index=1; index<=elemente; index++ )
15177     {
15178         v.x=phigrafisch[index];

```

```

15179         v.y=sgrafisch[index];
15180
15181         plotWindow->setValueKurve1( i, v );
15182
15183         w.x=phigrafisch[index];
15184         w.y=s_strichgrafisch[index];
15185
15186         plotWindow->setValueKurve3( i, w );
15187
15188         i++;
15189     }
15190     if( dmanuelles_phi == 1 )
15191     {
15192         plotWindow->plotIt_manuell( dx_mins_phi, dx_maxs_phi,
15193             ddxs_phi, dy_mins_phi, dy_maxs_phi, ddys_phi );
15194     }
15195     else if( dmanuelles_phi == 2 )
15196     {
15197         plotWindow->plotIt_Hub_geschw();
15198     }
15199 }
15200 if ( radiobuttonpsi_geschw_beschl->isChecked() )
15201 {
15202     for ( index=1; index<=elemente; index++ )
15203     {
15204         v.x=phigrafisch[index];
15205         v.y=psigrafisch[index];
15206
15207         plotWindow->setValueKurve1( i, v );
15208
15209         w.x=phigrafisch[index];
15210         w.y=psi_strichgrafisch[index];
15211
15212         plotWindow->setValueKurve3( i, w );
15213
15214         m.x=phigrafisch[index];
15215         m.y=psi_zweistrichgrafisch[index];
15216
15217         plotWindow->setValueKurve4( i, m );
15218
15219         i++;
15220     }
15221     if( dmanuellepsi_phi == 1 )
15222     {

```

```

15223         plotWindow->plotIt_manuell( dx_minpsi_phi, dx_maxpsi_phi,
15224             ddxpsi_phi, dy_minpsi_phi, dy_maxpsi_phi, ddypsi_phi );
15225     }
15226     else if( dmanuellepsi_phi == 2 )
15227     {
15228         plotWindow->plotIt_psi_geschw_beschl();
15229     }
15230 }
15231 if ( radiobuttonx_Bik_x_Bikstrich->isChecked() )
15232 {
15233     for ( index=1; index<=elemente; index++ )
15234     {
15235         v.x=phigrafisch[index];
15236         v.y=x_Bikgrafisch[index];
15237
15238         plotWindow->setValueKurve1( i, v );
15239
15240         w.x=phigrafisch[index];
15241         w.y=x_Bik_strichgrafisch[index];
15242
15243         plotWindow->setValueKurve3( i, w );
15244
15245         i++;
15246     }
15247     if( dmanuellex_Bik == 1 )
15248     {
15249         plotWindow->plotIt_manuell( dx_minx_Bik, dx_maxx_Bik,
15250             ddx_Bik, dy_minx_Bik, dy_maxx_Bik, ddyx_Bik );
15251     }
15252     else if( dmanuellex_Bik == 2 )
15253     {
15254         plotWindow->plotIt_xBik();
15255     }
15256 }
15257 if ( radiobuttony_Bik_y_Bikstrich->isChecked() )
15258 {
15259     for ( index=1; index<=elemente; index++ )
15260     {
15261         v.x=phigrafisch[index];
15262         v.y=y_Bikgrafisch[index];
15263
15264         plotWindow->setValueKurve1( i, v );
15265
15266         w.x=phigrafisch[index];

```

```

15267         w.y=y_Bik_strichgrafisch[index];
15268
15269         plotWindow->setValueKurve3( i, w );
15270
15271         i++;
15272     }
15273     if( dmanuelley_Bik == 1 )
15274     {
15275         plotWindow->plotIt_manuell( dx_miny_Bik, dx_maxy_Bik,
15276             ddx_Bik, dy_miny_Bik, dy_maxy_Bik, ddy_Bik );
15277     }
15278     else if( dmanuelley_Bik == 2 )
15279     {
15280         plotWindow->plotIt_yBik();
15281     }
15282 }
15283 if ( radiobuttondurchmesser->isChecked() )
15284 {
15285     if( konturAussenAussen->isChecked() )
15286     {
15287         for ( index=0; index<=elemente; index++ )
15288         {
15289             v.x=x_B_verstzt[index];
15290             v.y=y_B_verstzt[index];
15291
15292             plotWindow->setValueKurve5( i, v );
15293
15294             u.x=x_B_versetztstargrafisch[index];
15295             u.y=y_B_versetztstargrafisch[index];
15296
15297             plotWindow->setValueKurve6( i, u );
15298
15299             r.x=x_i_verstzt[index];
15300             r.y=y_i_verstzt[index];
15301
15302             plotWindow->setValueKurve9( i, r );
15303
15304             s.x=x_i_versetztstargrafisch[index];
15305             s.y=y_i_versetztstargrafisch[index];
15306
15307             plotWindow->setValueKurve10( i, s );
15308
15309             o.x=x_Wgrafisch[index];
15310             o.y=y_Wgrafisch[index];

```

```

15311
15312         plotWindow->setValueKurve11( i, o );
15313
15314         i++;
15315     }
15316     if( dmanuelleArbeitskurven == 1 )
15317     {
15318         plotWindow->plotIt_manuell( dx_minKurvenscheibe, dx_maxKurvenscheibe,
15319             ddxKurvenscheibe, dy_minKurvenscheibe, dy_maxKurvenscheibe,
15320             ddyKurvenscheibe );
15321     }
15322     else if( dmanuelleArbeitskurven == 2 )
15323     {
15324         plotWindow->plotIt_DK_aa_durchmesser();
15325     }
15326 }
15327 if( konturAussenInnen->isChecked() )
15328 {
15329     for ( index=0; index<=elemente; index++ )
15330     {
15331         v.x=x_B_verstzt[index];
15332         v.y=y_B_verstzt[index];
15333
15334         plotWindow->setValueKurve5( i, v );
15335
15336         u.x=x_B_versetztstargrafisch[index];
15337         u.y=y_B_versetztstargrafisch[index];
15338
15339         plotWindow->setValueKurve6( i, u );
15340
15341         r.x=x_i_verstzt[index];
15342         r.y=y_i_verstzt[index];
15343
15344         plotWindow->setValueKurve9( i, r );
15345
15346         s.x=x_a_versetztstargrafisch[index];
15347         s.y=y_a_versetztstargrafisch[index];
15348
15349         plotWindow->setValueKurve10( i, s );
15350
15351         o.x=x_Wgrafisch[index];
15352         o.y=y_Wgrafisch[index];
15353
15354         plotWindow->setValueKurve11( i, o );

```

```

15355
15356         i++;
15357     }
15358     if( dmanuelleArbeitskurven == 1 )
15359     {
15360         plotWindow->plotIt_manuell( dx_minKurvenscheibe, dx_maxKurvenscheibe,
15361             ddxKurvenscheibe, dy_minKurvenscheibe, dy_maxKurvenscheibe,
15362             ddyKurvenscheibe );
15363     }
15364     else if( dmanuelleArbeitskurven == 2 )
15365     {
15366         plotWindow->plotIt_DK_ai_durchmesser();
15367     }
15368 }
15369 if( konturInnenAussen->isChecked() )
15370 {
15371     for ( index=0; index<=elemente; index++ )
15372     {
15373         v.x=x_B_verstzt[index];
15374         v.y=y_B_verstzt[index];
15375
15376         plotWindow->setValueKurve5( i, v );
15377
15378         u.x=x_B_versetztstargrafisch[index];
15379         u.y=y_B_versetztstargrafisch[index];
15380
15381         plotWindow->setValueKurve6( i, u );
15382
15383         r.x=x_a_verstzt[index];
15384         r.y=y_a_verstzt[index];
15385
15386         plotWindow->setValueKurve9( i, r );
15387
15388         s.x=x_i_versetztstargrafisch[index];
15389         s.y=y_i_versetztstargrafisch[index];
15390
15391         plotWindow->setValueKurve10( i, s );
15392
15393         o.x=x_Wgrafisch[index];
15394         o.y=y_Wgrafisch[index];
15395
15396         plotWindow->setValueKurve11( i, o );
15397         i++;
15398     }

```

```

15399         if( dmanuelleArbeitskurven == 1 )
15400         {
15401             plotWindow->plotIt_manuell( dx_minKurvenscheibe, dx_maxKurvenscheibe,
15402                 ddxKurvenscheibe, dy_minKurvenscheibe, dy_maxKurvenscheibe,
15403                 ddyKurvenscheibe );
15404         }
15405         else if( dmanuelleArbeitskurven == 2 )
15406         {
15407             plotWindow->plotIt_DK_ia_durchmesser();
15408         }
15409     }
15410 }
15411 if ( radiobuttonmue->isChecked() )
15412 {
15413     for ( index=1; index<=elemente; index++ )
15414     {
15415         v.x=phigrafisch[index];
15416         v.y=muegrafisch[index];
15417
15418         plotWindow->setValueKurve1( i, v );
15419         i++;
15420     }
15421     if( dmanuelleUebertragungswinkel == 1 )
15422     {
15423         plotWindow->plotIt_manuell( dx_minUebertragungswinkel,
15424             dx_maxUebertragungswinkel, ddxUebertragungswinkel,
15425             dy_minUebertragungswinkel, dy_maxUebertragungswinkel,
15426             ddyUebertragungswinkel );
15427     }
15428     else if( dmanuelleUebertragungswinkel == 2 )
15429     {
15430         plotWindow->plotIt_mue();
15431     }
15432 }
15433 if ( radiobuttonr_K->isChecked() )
15434 {
15435     for ( index=1; index<=elemente; index++ )
15436     {
15437         v.x=phigrafisch[index];
15438         v.y=r_Kgrafisch[index];
15439
15440         plotWindow->setValueKurve1( i, v );
15441         i++;
15442     }

```



```

15443     if( dmanuelleKruemmungsradius == 1 )
15444     {
15445         plotWindow->plotIt_manuell( dx_minKruemmungsradius,
15446             dx_maxKruemmungsradius, ddxKruemmungsradius,
15447             dy_minKruemmungsradius, dy_maxKruemmungsradius,
15448             ddyKruemmungsradius );
15449     }
15450     else if( dmanuelleKruemmungsradius == 2 )
15451     {
15452         plotWindow->plotIt_rK();
15453     }
15454 }
15455 if ( radiobuttonGegenkurven->isChecked() )
15456 {
15457     for ( index=0; index<=elemente; index++ )
15458     {
15459         v.x=x_B_verstzt[index];
15460         v.y=y_B_verstzt[index];
15461
15462         plotWindow->setValueKurve5( i, v );
15463
15464         u.x=x_B_versetztstargrafisch[index];
15465         u.y=y_B_versetztstargrafisch[index];
15466
15467         plotWindow->setValueKurve6( i, u );
15468
15469         i++;
15470     }
15471     if( dmanuelleMittelpunktskurven == 1 )
15472     {
15473         plotWindow->plotIt_manuell( dx_minMittelpunktskurven,
15474             dx_maxMittelpunktskurven, ddxMittelpunktskurven,
15475             dy_minMittelpunktskurven, dy_maxMittelpunktskurven,
15476             ddyMittelpunktskurven );
15477     }
15478     else if( dmanuelleMittelpunktskurven == 2 )
15479     {
15480         plotWindow->plotIt_MK();
15481     }
15482 }
15483 plotWindow->show();
15484 }
15485
15486 void Opticurv::print()

```

```

15487 {
15488 #ifndef QT_NO_PRINTER
15489     QPainter printer;
15490     printer.setFullPage(TRUE);
15491     if ( printer.setup( this ) )
15492     {
15493         statusBar()->message( "Drucke gerade...", 2000 );
15494         QPainter p( &printer );
15495         QPaintDeviceMetrics metrics(p.device());
15496         int dpix = metrics.logicalDpiX();
15497         int dpiy = metrics.logicalDpiY();
15498         const int margin = 72;
15499         QRect body(margin*dpix/72, margin*dpiy/72,
15500                   metrics.width()-margin*dpix/72*2,
15501                   metrics.height()-margin*dpiy/72*2 );
15502         QFont font( "times", 10 );
15503         QSimpleRichText richText( anzeige->text(), font,
15504                                   anzeige->context(),
15505                                   anzeige->styleSheet(),
15506                                   anzeige->mimeTypeSourceFactory(),
15507                                   body.height() );
15508         richText.setWidth( &p, body.width() );
15509         QRect view( body );
15510         int page = 1;
15511         do {
15512             richText.draw( &p, body.left(), body.top(), view, colorGroup() );
15513             view.moveBy( 0, body.height() );
15514             p.translate( 0 , -body.height() );
15515             p.setFont( font );
15516             p.drawText( view.right() - p.fontMetrics().width( QString::number(page) ),
15517                       view.bottom() + p.fontMetrics().ascent() + 5,
15518                       QString::number(page) );
15519             if ( view.top() >= body.top() + richText.height() )
15520                 break;
15521             printer.newPage();
15522             page++;
15523         }
15524         while (TRUE);
15525         statusBar()->message( "Drucken beendet", 2000 );
15526     }
15527 #endif
15528 }
15529
15530 #include "opticurv273.moc"

```

```

15531
15532 //----- main -----
15533 int main( int argc, char* argv[] )
15534 {
15535     QApplication myapp(argc,argv);
15536
15537
15538     #ifdef QT_DLL
15539         QBaseApplication myapp(argc,argv);
15540     #else
15541         QApplication myapp(argc,argv);
15542     #endif
15543
15544     Opticurv* mywidget = new Opticurv();
15545     mywidget->setGeometry( 50, 50, 400, 400 );
15546
15547     myapp.setMainWidget( mywidget );
15548     mywidget->setCaption("OPTICURV");
15549     mywidget->show();
15550     return myapp.exec();
15551 }

```