Steuerung von Werkzeugmaschinen mit redundanten Achsen

Dissertation

zur Erlangung des akademischen Grades
„Doktor der Naturwissenschaften“
(Dr. rer. nat.)

am Fachbereich Mathematik und Informatik, Physik, Geographie
der Justus-Liebig-Universität Gießen

vorgelegt von

Dipl.-Math. Marco Bock

geb. in Weilburg

Gießen, August 2010

Datum der Disputation: 19. November 2010

Dekan: Prof. Dr. Christian Diller

Gutachter: Prof. Dr. Tomas Sauer (Gießen)
Prof. Dr. Ulrich Reif (Darmstadt)
Prof. Dr. Wolfgang Papiernik (Erlangen)

Vorwort

Diese Doktorarbeit geht aus einer Zusammenarbeit mit der Siemens AG hervor. Von Januar 2007 bis Juli 2010 habe ich in Erlangen als Doktorand in der Forschungs- und Entwicklungsabteilung der Business Unit *Motion Control Systems* der Division *Drive Technologies* des Sektors *Industry* (I DT MC R&D) daran gearbeitet.

Ein herzliches Dankeschön gilt allen, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Arbeit beigetragen haben.

Besonders möchte ich dabei zuerst meine beiden Betreuer, Herrn Professor Dr. Wolfgang Papiernik von der Siemens AG und Herrn Professor Dr. Tomas Sauer von der Justus–Liebig–Universität Gießen, hervorheben. Beide haben durch wichtige Anregungen und gute Ideen zu dieser Arbeit beigetragen.

Als zweites gilt mein besonderer Dank Herrn Andreas Kurniady und Herrn Dr. Klaus Geißdörfer von der Siemens AG. Sie standen mir bei technischen Fragen stets zur Verfügung.

Inhaltsverzeichnis

1. Einleitung **7**

2. Werkzeugmaschinen und ihre Steuerung **9**
2.1. Beschreibung der untersuchten Kinematiken und Werkzeugmaschinen . . . 9
2.1.1. Allgemeine Kinematiken 9
2.1.2. Bearbeitungsaufgaben der untersuchten Werkzeugmaschinen 10
2.1.3. Motivation einer Werkzeugmaschine mit redundanten Achsen 11
2.1.4. Kinematik A 13
2.1.5. Kinematik B 17
2.2. Bewegungsführung auf vorgegebenen Bahnen 19
2.2.1. Bewegungsführung als Optimalsteuerungsproblem 19
2.2.2. Prozesskette in CNC–Steuerungen 21
2.2.3. Algorithmus für die beschleunigungsbegrenzte Bewegungsplanung . . 23

3. Bekannte Verfahren zur Steuerung redundanter Kinematiken **31**
3.1. Bewegung durch Nachsetzen 31
3.2. Geometrische Zerlegung 32
3.3. Filterung im Zeitbereich 35
3.4. Regelungskonzept mit Schleppfehlerkompensation 38
3.4.1. Konturverzerrung durch Überschwingen der Summenachse 40
3.4.2. Hochdynamische Regelung durch vorgelagerter Sollwertfilterung . . 42

4. Präparative Bewegungsaufteilung im Zeitbereich **45**
4.1. Idee und Steuerungsschema 45
4.2. Approximation diskreter Daten durch Splinefunktionen 47
4.2.1. Grundlegende Notationen und Eigenschaften von Splinefunktionen . 47
4.2.2. Least–Squares–Approximation 50
4.2.3. Konkrete Randbedingungen für die Approximanten 53
4.2.4. Tschebyschow–Approximation 57
4.2.5. Approximation mit der Idee des Schoenbergschen Splineoperators . . 58
4.2.6. Approximation durch Smoothing–Splines 61
4.2.7. Glatte Splineapproximation durch Lösen quadratischer Programme . 65
4.2.8. Glatte Approximationen durch Lösen linearer Programme 67
4.3. Vollständiges Verfahren und Anwendungsbeispiele 68
4.3.1. Bewegungsaufteilung im Zeitbereich 68

4.3.2. Modifikation der Einstellungen in der Bewegungsplanung 73
4.3.3. Anwendung der Verfahren auf gesamte Konturen 78
4.3.4. Abschnittsweises Vorgehen 85
4.3.5. Ausdünnen der Daten 90
4.3.6. Ein Beispiel für ein Werkstück 91

5. Verkürzung der Bearbeitungszeit durch Umparametrisierung 93
5.1. Idee und Aufgabenstellung als Optimalsteuerungsproblem 93
5.2. Lösungsalgorithmus 97
5.2.1. Phase I: Die Grenzkurve 98
5.2.2. Phase II: Die optimale Bahntrajektorie 100
5.2.3. Sonderfall: Alle Achsen stehen 106
5.3. Ergebnisse und Beispiele 107

6. Zusammenfassung und Ausblick 115

Symbolverzeichnis 117

Literaturverzeichnis 120

A. Anhang 125
A.1. Optimalsteuerungsprobleme 125

1. Einleitung

Viele Bereiche der industriellen Fertigung setzen CNC–Werkzeugmaschinen zur automatisierten Herstellung von Werkstücken ein. Eine wesentliche Anforderung an derartige Maschinen ist eine hohe Produktivität. Das wiederum erfordert hohe Bearbeitungsgeschwindigkeiten und damit möglichst schnelle Bewegungen des Werkzeugs am Werkstück. Bei gewöhnlichen Maschinenkonstruktionen sinkt mit einem wachsenden Arbeitsbereich die Bewegungsgeschwindigkeit. Große Werkstücke können folglich nur langsam bearbeitet werden. Eine besondere Klasse von Werkzeugmaschinen versucht dieses Dilemma zu beheben. Sie nutzen redundante Achsen, um einen großen Verfahrbereich und eine hohe Bewegungsdynamik zu kombinieren.

Diese Arbeit betrachtet Maschinen mit redundanten Achsen und einem besonderen kinematischen Aufbau. Dabei ergänzt eine hochdynamische Kinematik mit kleinem Verfahrbereich, die Zusatzachsen, ein System mit großen, trägen Hauptachsen. Derartige Maschinen lassen sich nicht mit den Steuerungsmethoden für nichtredundante Werkzeugmaschinen betreiben. Zur Steuerung solcher redundanten Achsen gibt es bereits einige Ansätze, allerdings haben alle gewisse Schwachstellen in bestimmten Anwendungsbereichen. Diese Arbeit stellt einen neuen, alternativen Ansatz für den Fall der beschleunigungsbegrenzten Bewegungsführung vor, der die Schwachstellen nicht aufweist.

Der Ansatz nutzt den strukturellen Aufbau einer CNC–Steuerung. Darin gibt es einen präparativ arbeitenden Bereich, dort ist die Realisierung vorgesehen. Das Verfahren an sich besteht aus einer sich iterativ abwechselnden Erzeugung und Zerlegung von Sollwerttrajektorien im Zeitbereich. Für die Zerlegung werden verschiedene Approximationsverfahren aus der Splinetheorie vorgeschlagen. Einige von ihnen weisen bestimmte Glattheitseigenschaften auf. Außerdem ermöglicht der Ansatz ein abschnittsweises Vorgehen auf der zu bearbeitenden Kontur. Eine optionale Ergänzung erhält der Ansatz durch ein Verfahren zur Umparametrisierung im Sinne der Zeitoptimalität.

Die Arbeit beginnt mit einem Kapitel über Werkzeugmaschinen und deren Steuerung. Es beschreibt zuerst allgemeine und dann die hier betrachteten Kinematiken sowie deren Einsatzgebiete. Darauf folgt Abschnitt 2.2 über die Arbeitsweise der CNC–Steuerung und besonders der darin enthaltenen beschleunigungsbegrenzten Bewegungsführung. Für letztere wird in Unterabschnitt 2.2.3 ein konkreter Algorithmus vorgestellt.

Das dritte Kapitel fasst vier bekannten Ansätze zur Steuerung redundanter Achsen zusammen. Es beginnt mit der Bewegung durch Nachsetzen, bei der abwechselnd die redundanten Teilsysteme arbeiten. Der zweite Ansatz nutzt eine Zerlegung der Konturkurve, sodass die Bewegungsführung mit leichten Modifikationen genutzt werden kann. Die Filterung im Zeitbereich teilt mithilfe von Filtern zeitlich äquidistant abgetastete

Sollwertverläufe in die Anteile für die Einzelachsen auf. Der letzte Ansatz, die Schleppfehlerkompensation, ist ein spezielles Regelungskonzept, bei dem die Zusatzachsen den Schleppfehler der Hauptachsen kompensiert. Dieser Abschnitt enthält außerdem ein neuartiges Regelungskonzept womit eine auftretende Konturverzerrung vermieden wird und das eine hochdynamische Regelung redundanter Achsen ermöglicht.

Kapitel vier stellt den neuen Ansatz vor. Es startet mit der prinzipiellen Idee und der damit zusammenhängenden Realisierungsmöglichkeit in der Steuerung. Abschnitt 4.2 beschreibt die verwendeten Approximationsmethoden sowie die notwendigen Grundlagen aus der Splinetheorie. Der letzte Abschnitt dieses Kapitels vervollständigt zuerst das Vorgehen und präsentiert Ergebnisse aus Beispielen. Unterabschnitt 4.3.4 erläutert, wie das Vorgehen auch bei der in der Steuerung stattfindenden abschnittsweisen Bearbeitung der Kontur angewendet werden kann.

Das fünfte Kapitel führt eine zusätzliche Umparametrisierung aus. Es beginnt mit einem Abschnitt zur generellen Idee und der formalen mathematischen Formulierung als Optimalsteuerungsproblem. Der folgende Abschnitt 5.2 beschreibt den Lösungsalgorithmus und der letzte zeigt Ergebnisse aus dessen Anwendung.

Eine kurze Zusammenfassung der wesentlichen Ergebnisse und ein Ausblick auf weiterführende Frage- und Aufgabenstellungen schließen die Arbeit ab.

2. Werkzeugmaschinen und ihre Steuerung

2.1. Beschreibung der untersuchten Kinematiken und Werkzeugmaschinen

2.1.1. Allgemeine Kinematiken

Einer Werkzeugmaschine oder einem Roboter liegt immer eine bestimmte Kinematik zugrunde. Durch sie sind die Bewegungsmöglichkeiten des Werkzeugkopfes eindeutig festgelegt[1]. Für die Stellung des Kopfes stehen sechs Freiheitsgrade zur Verfügung, drei translatorische und drei rotatorische:

$$\mathbf{p} = [x, y, z]^T \qquad \text{und} \qquad \boldsymbol{\alpha} = [\alpha, \beta, \gamma]^T.$$

Der damit aufgespannte Raum ist natürlich der $\mathbb{R}^6$, in diesem Kontext spricht man vom *Basiskoordinatensystem* (BKS), vergleiche Abbildung 2.1. Der von den Komponenten x, y und z aufgespannte Raum ist der dreidimensionale Umgebungsraum, die Winkel in $\boldsymbol{\alpha}$ beschreiben die Orientierung des Werkzeugs. Sind eine oder mehrere Komponenten des Basiskoordinatensystems bei jeder Stellung der Maschine konstant, kann beziehungsweise können sie aus der Darstellung herausgenommen werden.

Der Vektor $\mathbf{q} = [q_1, \ldots, q_{n_\mathbf{q}}]^T$ enthält die Stellungen der $n_\mathbf{q}$ Achsgelenke der Maschine, der damit aufgespannte Raum ist das $n_\mathbf{q}$–dimensionale *Maschinenkoordinatensystem* (MKS). Zwei einfache Beispiele für kinematische Systeme sind in Abbildung 2.2 zu sehen. Das linke besteht aus zwei translatorischen Achsen, die senkrecht aufeinander stehen. Bis auf eine konstante Verschiebung stimmen MKS und BKS überein. Das rechte System hingegen besitzt zwei rotatorische Achsen, q_1 und q_2 sind deren Winkelstellungen. Im ersten Fall ist die Orientierung des Werkzeugs für jede Achsstellung konstant, im zweiten hingegen ändert sie sich.

Im Allgemeinen beschreibt die *kinematische (Vorwärts-)Transformation*

$$\Lambda : \mathcal{D}_{\text{MKS}} \ni \mathbf{q} \longmapsto \Lambda(\mathbf{q}) = \begin{bmatrix} \mathbf{p} \\ \boldsymbol{\alpha} \end{bmatrix} \in \mathbb{R}^6$$

das Achssystem der Maschine. Dabei ist $\mathcal{D}_{\text{MKS}}$ die Menge der erlaubten Achsstellungen. Sie kann sehr komplex sein, gerade wenn der Maschine irgendwelche Hindernisse im Weg stehen oder die Position einer Achse von anderen abhängt. Beispielsweise führt eine Achs-

[1] Etwaige Hindernisse in der Umgebung spielen in dieser Arbeit keine Rolle.

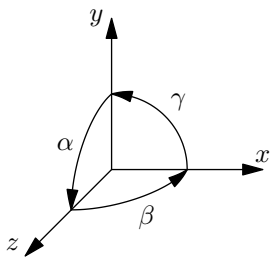

Abbildung 2.1.: Basiskoordinatensystem (BKS).

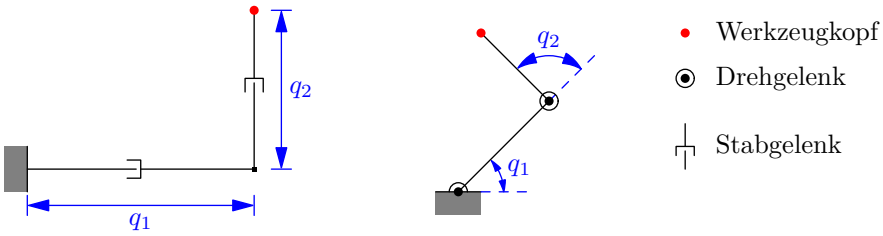

Abbildung 2.2.: Zwei Beispiele kinematischer Achssysteme.

stellung von $q_2 = \pi$ bei der rechten Kinematik in Abbildung 2.2 zu einer Kollision, wenn die Gelenkarme nicht versetzt sind. Die Menge $\mathcal{D}_{\text{BKS}} = \Lambda(\mathcal{D}_{\text{MKS}})$ enthält dann alle möglichen Stellungen des Werkzeugkopfes im BKS. Noch komplizierter sind Situationen, in denen sich $\mathcal{D}_{\text{MKS}}$ mit der Zeit verändert. Sie treten in der Realität normalerweise ein, zum Beispiel wenn bei der Bearbeitung Material vom Werkstück abgetragen wird.

2.1.2. Bearbeitungsaufgaben der untersuchten Werkzeugmaschinen

Für Werkzeugmaschinen und Roboter gibt es verschiedene Anforderungen an die Arbeitsweise. Das geht von geraden Punkt–zu–Punkt–Bewegungen über das Fahren entlang fest vorgegebener Konturen bis hin zum selbstständigen Finden der Wege. Diese Arbeit schränkt sich auf den zweiten Fall ein. Dabei ist eine zu bearbeitende Kontur in verschiedene Abschnitte aufgeteilt. Auf jedem davon soll sich der Werkzeugkopf höchstens mit einer Bahngeschwindigkeit $\hat{v}_b > 0$, dem *programmierten Vorschub*, bewegen. Der Wert von $\hat{v}_b$ kann sich von Abschnitt zu Abschnitt unterscheiden und hängt mit der Bearbeitungsart der Maschine zusammen. Insbesondere ist auch der unbegrenzte Fall $\hat{v}_b = \infty$ möglich.

Der in dieser Arbeit überwiegend betrachtete Anwendungsfall kommt aus dem 2–D–Laserschneiden. Die Bearbeitung kennt zwei Arten, das Schneiden und das Positionieren. Sie treten abwechselnd in der Kontur auf und an den Übergängen muss der Kopf stehen. Das Positionieren ist eine gerade Punkt–zu–Punkt–Bewegung, während der die Bahngeschwindigkeit nicht nach oben begrenzt ist. Ein Schneideblock besteht aus einer allgemei-

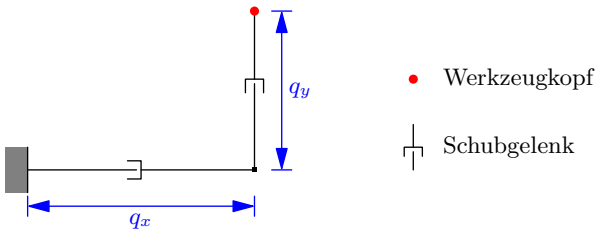

Abbildung 2.3.: Kinematik A.

neren Kurve, sie wird in Definition 2.1 präzisiert. Beim Schneiden ist der Vorschub des Kopfes auf der Kontur begrenzt. Abbildung 4.10 auf Seite 85 zeigt eine typische Kontur.

Die gesamte zu bearbeitende Kontur ist für allgemeine Kinematiken eine stetige Kurve h, die aus einer Raumkurve und einer Orientierungskurve besteht. Wir verwenden die Bezeichnung

$$h_{\mathbf{p}} : [s_0, s_f] \ni s \mapsto h_{\mathbf{p}}(s) = [x(s), y(s), z(s)]^T \in \mathbb{R}^3,$$

$s_0 < s_f$, für die Bogenlängenparametrisierung der Raumkurve. Ist die Orientierung des Kopfes für die Bearbeitung relevant, kommt ergänzend die über demselben Bahnparameter definierte Orientierungskurve

$$h_{\boldsymbol{\alpha}} : [s_0, s_f] \ni s \mapsto h_{\boldsymbol{\alpha}}(s) = [\alpha(s), \beta(s), \gamma(s)]^T \in \mathbb{R}^3$$

hinzu und die gesamte Kontur ist durch die Kurve

$$h = h_{\text{gesamt}} : [s_0, s_f] \ni s \mapsto h_{\text{gesamt}}(s) = \left[h_{\mathbf{p}}(s)^T, h_{\boldsymbol{\alpha}}(s)^T\right]^T \in \mathbb{R}^6$$

gegeben. Es muss natürlich $h([s_0, s_f]) \subset \mathcal{D}_{\text{BKS}}$ gelten. Die oben erwähnte maximale Bahngeschwindigkeit $\hat{v}_b$ bezieht sich immer auf die Raumkurve $h_{\mathbf{p}}(s)$.

2.1.3. Motivation einer Werkzeugmaschine mit redundanten Achsen

Viele Anwendungen in der industriellen Fertigung fordern Maschinen mit großem Verfahrbereich und kurzen Bearbeitungszeiten. Für gewisse Kinematiken widersprechen sich diese Anforderungen. Bei einer Kinematik aus zwei senkrecht zueinander stehenden Linearachsen wie in Abbildung 2.3, im Folgenden *Kinematik A* genannt, verringert sich mit zunehmendem Verfahrbereich die Dynamik der Achsen. Es müssen immer größere Massen bewegt werden und dadurch verkleinert sich das Beschleunigungsvermögen. Je größer der Verfahrbereich, desto geringer ist die Bearbeitungsgeschwindigkeit.

Eine besondere Art von Kinematiken versucht dieses „Dilemma“ zu beheben. Sie verwendet ein großes, träges Achssystem als Grundlage. Dafür nutzen wir die Bezeichnungen *Hauptachsen* oder *Grobachsen*. Darauf ist eine kleine hochdynamische Zusatzkinematik

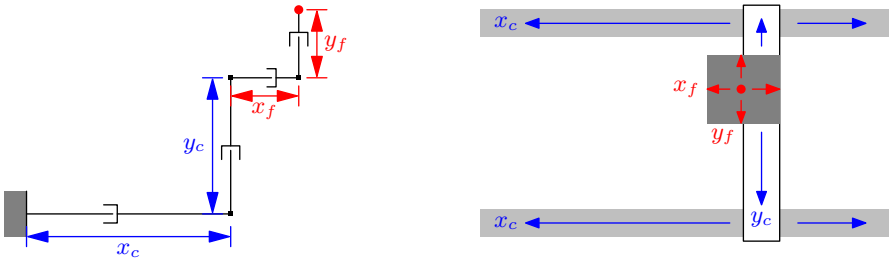

Abbildung 2.4.: Kinematik B.

montiert, die *Zusatz-* oder *Feinachsen*. Damit hat man die gewünschte Kombination aus großem Verfahrbereich und hochdynamischer Bearbeitungsmöglichkeit. Eine besondere Ausprägung einer solchen Kinematik zeigt Abbildung 2.4. Dabei sind die Grobachsen zwei große, senkrecht zueinander angeordnete Linearachsen[2] x_c und y_c, die durch zwei kleine, ebenfalls lineare Zusatzachsen x_f und y_f ergänzt werden. Dieser Aufbau trägt im Folgenden die Bezeichnung *Kinematik B*. Die Schwierigkeit hier liegt, verglichen mit Kinematik A, in seiner Steuerung, wie Kapitel 3 detailliert beschreibt. Diese Arbeit betrachtet einen derartigen Maschinenaufbau.

Kinematik B gehört zu den redundanten Kinematiken. Sie zeichnen sich im Allgemeinen — grob gesagt — dadurch aus, dass die Maschine mehr Freiheitsgrade aufweist, als für die zu erledigende Bewegung notwendig sind. Eine Maschine mit Kinematik B besitzt beispielsweise vier Freiheitsgrade, die Bewegung des Kopfes aber nur zwei. Die Zusatzachsen sind für die reine Realisierbarkeit der Bewegung entlang einer Kontur überflüssig.

Bisher gibt es nur wenige Werkzeugmaschinen mit redundanten Achsen. Zwei Beispiele für Kinematik B sind die Stanz–Laser–Maschine TruMatic 7000 ([36],[1]) und die Laserschneidmaschine TruLaser 3530 ([37]) der Firma Trumpf. Bei letzter gibt es nur in einer kartesischen Richtung eine Zusatzachse. Weitere Maschinen mit redundanten Achsen für die Laserbearbeitung sind die Syncrono ([31]) und die Optimo Vivida ([30]) der Firma Prima Industrie. Hier sind die Zusatzachsen allerdings nicht kartesisch. Auch im Bereich der Fräsbearbeitung großer Bauteile in der Luftfahrtindustrie findet man eine Maschine mit redundanter Kinematik. Die Firma Loxin hat eine Fünfachs–Parallelkinematik (Tricept) auf ein dreidimensionales kartesisches Portal montiert ([26]).

Diese Arbeit spezialisiert sich bewusst auf Kinematik B. Es gibt zwar, neben den Robotern mit redundanten Kinematiken, auch komplexere redundante Strukturen für Werkzeugmaschinen, allerdings sind die Ergebnisse dieser Arbeit nicht ohne weiteres auf sie anwendbar.

In der Robotik gibt es ebenfalls Systeme mit redundanten Achsen. Ein Beispiel ist etwa der Siebenachs–Roboterarm Mitsubishi PA–10 ([32, Kapitel 11]). Generell unterscheidet sich der kinematische Aufbau eines redundanten Roboters deutlich von Kinematik A oder B, meist besteht er aus einer Verkettung von Dreh- und Stabgelenken. Die kinematische

[2] Der Buchstabe c kommt vom englischen Wort „coarse“ für grob.

Transformation ist damit wesentlich komplexer als bei Kinematik B. Dafür bestehen die Bearbeitungsaufgaben eher aus der Bewegung entlang einfacher Bahnen mit einer geringen Genauigkeit. Bei Werkzeugmaschinen sind die Bahnen sehr komplex (vergleiche Abbildung 4.10 auf Seite 85) und eine hohe Genauigkeit ist erforderlich. Somit unterscheidet sich die Problemstellung bei redundanten Robotern deutlich von der in dieser Arbeit und die Forschungsergebnisse aus diesem Bereich können nicht verwendet werden. Eine Übersicht zu Redundanz in der Robotik bietet Kapitel 11 in [32].

2.1.4. Kinematik A

Die folgenden Abschnitte greifen häufig auf Kinematik A und die zu bearbeitenden Konturen zurück, daher sollen hier notwendige Notationen festgelegt werden. Für die Kinematiken A und B sind die Konturen im Fall der 2-D-Laserbearbeitung ebene Kurven, bei denen die Orientierung des Werkzeugkopfes keine Rolle spielt. Die Darstellung aus Abschnitt 2.1.1 vereinfachen wir daher, sodass h nur die x– und y–Komponente des BKS enthält.

Definition 2.1. *Sei* $h_u : [u_0, u_f] \ni u \mapsto h_u(u) \in \mathbb{R}^2$ *eine stetige Kurve und*

$$\{u_0, u_1, \ldots, u_{n_h}\}, \qquad u_0 < u_1 < \ldots < u_{n_h} = u_f, \tag{2.1}$$

eine Unterteilung des Intervalls $[u_0, u_f]$*, sodass* h_u *auf* (u_j, u_{j+1})*,* $j = 0, \ldots, n_h - 1$*, unendlich oft differenzierbar ist. Diese Arbeit behandelt nur Kurven, die weitere Eigenschaften erfüllen. Sie bestehen aus Polynomstücken oder Kreisbögen und die Bogenlänge jedes Stückes ist durch* $L_{\min} > 0$ *nach unten begrenzt. Außerdem soll die Krümmung nur an den Stellen* u_i *springen und auf den Intervallen* (u_i, u_{i+1}) *stetig und nicht größer als* $\kappa_{\max} < \infty$ *sein. Derartige Kurven heißen, unabhängig von ihrer Parametrisierung,* Konturkurven*. Ihre Bogenlängenparametrisierung ist*

$$h : [s_0, s_f] \ni s \longmapsto h(s) = [x(s), y(s)]^T \in \mathbb{R}^2$$

und die zu (2.1) *passende Unterteilung des Intervalls* $[s_0, s_f]$ *ist*

$$\mathbf{S}_h := \{s_{h,0}, s_{h,1}, \ldots, s_{h,n_h}\}.$$

Die Definition einer Konturkurve ist sinnvoll, denn die in der Anwendung bearbeiteten Konturen bestehen stückweise aus Polynomen oder Kreisbögen mit den erwähnten Eigenschaften. Die Bogenlängenparametrisierung der Polynomstücke ist auf den Intervallen $(s_{h,i}, s_{h,i+1})$, $i = 0, \ldots, n_h - 1$, unendlich oft differenzierbar ([3]). Sie kann aber im Allgemeinen nicht durch rationale Funktionen dargestellt werden ([14], [13]), eine numerische Näherung ist notwendig. Auf diesen Aspekt geht die Arbeit aber nicht weiter ein.

Für die Schreibweise verschiedener Parametrisierungen von Kurven und Funktionen gilt folgende Vereinbarung.

Bemerkung 2.2. *In dieser Arbeit werden Kurven und Funktionen häufig umparametrisiert. Im Interesse der Lesbarkeit soll die Umparametrisierung aber nicht immer dabeistehen, stattdessen wird eine Schreibweise mit dem Funktionsbezeichner und dem jeweiligen Parameter genutzt. Das Beispiel der Kontur soll das verdeutlicht, für anderen Funktionen gilt das analog. Sei h eine Konturkurve gemäß Definition 2.1. Bewegt sich der Kopf entlang dieser Kurve, so ist*

$$s_b : [t_0, t_f] \ni t \mapsto s_b(t) \in [s_0, s_f]$$

die Bahnbewegung über der Zeit t. Diese Funktion können wir als monoton steigend voraussetzen, da die Konturkurven nur einfach durchlaufen werden. Die Bewegungskurve in der Ebene über der Zeit ist damit $h \circ s_b$. Wir schreiben im Folgenden $h(s)$, wenn es um die Kurve geht und $h(t)$ anstelle von $(h \circ s_b)(t)$, wenn es um die Bewegung über der Zeit geht. Ebenso verhält es sich mit $x(s)$ und $x(t)$. Falls einmal nicht die Funktion, sondern der Funktionswert an einer bestimmten Stelle gemeint ist, geht das aus dem Kontext eindeutig hervor oder es wird besonders gekennzeichnet. Außerdem bezeichnet x den Wert der x-Komponente eines Punktes im $\mathbb{R}^2$.

Kommen wir zu den Bezeichnungen von Kinematik A. Mit dem Vektor der Maschinenachsen $\mathbf{q} = [q_x, q_y]^T$ ist die kinematische Transformation $\Lambda_A : \mathcal{D}_{\mathrm{MKS,A}} \to \mathbb{R}^2$ gleich der Identität, das heißt, es gilt

$$\begin{bmatrix} q_x(s) \\ q_y(s) \end{bmatrix} = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix}.$$

Da wir von einer Kontur immer annehmen, dass sie von der Kinematik beziehungsweise der Maschine bearbeitet werden kann, ist eine formale Verfahrbereichsbegrenzung der Achsen überflüssig. Daher gilt $\mathcal{D}_{\mathrm{MKS,A}} = \mathbb{R}^2$.

Bemerkung 2.3. *Bei den in dieser Arbeit verwendeten Funktionen kommt es häufig vor, dass sie an endlich vielen Stellen des Definitionsbereichs nicht differenzierbar sind, aber links- und rechsseitige Grenzwerte der Ableitung existieren. Ein Beispiel ist die Beziehung zwischen Geschwindigkeit $v_x(t)$ und Beschleunigung $a_x(t)$ der x-Achse. Hat letztere Sprünge, dann ist $v_x(t)$ nicht auf dem gesamten Intervall $[t_0, t_f]$ differenzierbar. Trotzdem schreiben wir*

$$\dot{v}_x(t) = a_x(t).$$

Es wird immer aus dem Kontext hervorgehen, ob diese Situation gerade besteht. Weitere Beispiele sind die erste und zweite Ableitung der Kontur, $x'(s)$ und $x''(s)$. Letztere springt an endlich vielen Stellen.

Zu einer Bewegung $x(t)$ sind

$$v_x(t) := \dot{x}(t),$$

$$a_x(t) := \dot{v}_x(t) = \ddot{x}(t),$$
$$j_x(t) := \dot{a}_x(t) = \ddot{v}_x(t) = \dddot{x}(t),$$

die Geschwindigkeit, die Beschleunigung und der Ruck der x–Achse. Das gilt analog für die y–Achse. Generell bezeichnet der Punkt in dieser Arbeit immer die Ableitung nach der Zeit, $\dot{x}(t) = \frac{\mathrm{d}x}{\mathrm{d}t}(t)$, der Strich, wenn nicht anders erwähnt, die Ableitung nach der Bogenlänge der Kontur, $x'(s) = \frac{\mathrm{d}x}{\mathrm{d}s}(s)$.

Die Absolutbeträge der physikalischen Größen der Maschinenachsen dürfen bestimmte Grenzwerte nicht überschreiten. Die Restriktionen für Geschwindigkeit, Beschleunigung und Ruck sind in den Vektoren

$$\widehat{\mathbf{V}} = \begin{bmatrix} \widehat{V}_x \\ \widehat{V}_y \end{bmatrix}, \qquad \widehat{\mathbf{A}} = \begin{bmatrix} \widehat{A}_x \\ \widehat{A}_y \end{bmatrix}, \qquad \widehat{\mathbf{J}} = \begin{bmatrix} \widehat{J}_x \\ \widehat{J}_y \end{bmatrix},$$

zusammengefasst, wobei die einzelnen Einträge endliche, positive Werte sind. Abschnitt 2.2 stellt ein Verfahren vor, das daraus und aus einer Konturkurve die Achsbewegungen berechnet. Zu einer Bahnbewegung $s_b(t)$ benutzen wir die Bezeichnungen

$$v_b(t) := \dot{s}_b(t), \qquad a_b(t) := \dot{v}_b(t), \qquad j_b(t) := \dot{a}_b(t). \tag{2.2}$$

für Bahngeschwindigkeit, -beschleunigung und -ruck. Es gilt hierbei wegen der Monotonie von $s_b(t)$ aus Bemerkung 2.2 die Beziehung

$$v_b(t) = \left|\left|[v_x(t), v_y(t)]^T\right|\right|_2$$

zwischen Achsgeschwindigkeiten und Bahngeschwindigkeit. Damit ist die Bahngeschwindigkeit immer nichtnegativ.

Bemerkung 2.4. *Wenn in dieser Arbeit von Trajektorien die Rede ist, sind damit meistens Verläufe über der Zeit gemeint. Im kontinuierlichen Fall sind das reelle Funktion, wobei der Parameter der Zeit entspricht. Im diskreten Fall ist eine Trajektorie eine diskrete Abbildung mit einer streng monoton steigenden Folge von Zeitpunkten und den zugeordneten Werten für Position, Geschwindigkeit oder Ähnlichem. Trajektorien können sowohl skalar als auch vektorwertig sein. Nur Abschnitt 2.2.3 benutzt den Begriff Trajektorie auch in einem anderen Zusammenhang.*

Die Bewegung jeder Achse einer Maschine mit Kinematik A muss die eingeführten Grenzen der Geschwindigkeit, der Beschleunigung und optional des Rucks einhalten. Die Ursachen hierfür sind verschieden.

Die maximale Geschwindigkeit der Achse hängt von der maximalen Drehzahl des Motors ab. Für die Grenze der Achsbeschleunigung ist das maximale Drehmoment des Motors, die bewegte Masse und die Übersetzung zwischen rotatorischer und translatorischer Bewegung verantwortlich. Bei höheren Achsbeschleunigungsvorgaben für die Antriebe kann die Achse den Sollwerten nicht folgen und es kommt zu Konturverzerrungen oder sogar zum Abbruch

der Bearbeitung. Eine *Bewegungsplanung* ist ein Verfahren, das zu einer Kontur passende Achstrajektorien berechnet, die die Grenzen einhalten. Arbeitet sie nur mit Limitierungen für Achsgeschwindigkeit und -beschleunigung, nennt man sie *beschleunigungsbegrenzt*. Berücksichtigt sie auch Restriktionen für den Achsruck, so heißt sie *ruckbegrenzt*. In der Praxis hilft eine Ruckbegrenzung, die Qualität des Werkstücks zu verbessern.

Man kann folglich zwei Arten von gültigen Trajektorien für Kinematik A definieren.

Definition 2.5. *Sei h eine Konturkurve gemäß Definition 2.1. Eine Trajektorie*

$$\mathbf{q} : [t_0, t_f] \ni t \mapsto \mathbf{q}(t) = [q_x(t), q_y(t)]^T = [x(t), y(t)]^T \in \mathbb{R}^2$$

für Kinematik A heißt gültig und beschleunigungsbegrenzt*, wenn sie die folgenden Bedingungen erfüllt:*

1. *(Bewegung auf der Kontur)*

$$\forall\, t \in [t_0, t_f] : \qquad \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = h(s_b(t)). \tag{2.3}$$

2. *(Einhaltung der Achsgrenzen) Für alle $t \in [t_0, t_f]$ gilt*

$$\begin{aligned} |v_x(t)| &\leq \widehat{V}_x, & |v_y(t)| &\leq \widehat{V}_y, \\ |a_x(t)| &\leq \widehat{A}_x, & |a_y(t)| &\leq \widehat{A}_y. \end{aligned}$$

3. *(Randbedingungen)*

$$\begin{aligned} v_x(t_0) &= 0, & v_x(t_f) &= 0, \\ v_y(t_0) &= 0, & v_y(t_f) &= 0. \end{aligned}$$

4. *(Begrenzte Bahngeschwindigkeit)*

$$\forall\, t \in [t_0, t_f] : \quad 0 \leq v_b(t) \leq \hat{v}_b.$$

Wegen der Identität als kinematische Transformation nutzen wir für Kinematik A auch die Schreibweise $\mathbf{q}(t) = h(t)$.

Definition 2.6. *Seien die Voraussetzungen aus Definition 2.5 erfüllt und $\mathbf{q}(t) = h(t)$ eine Trajektorie, die alle Bedingungen der Definition erfüllt. Gelten zusätzlich*

$$\begin{aligned} |j_x(t)| &\leq \widehat{J}_x, & |j_y(t)| &\leq \widehat{J}_y, \\ a_x(t_0) &= 0, & a_y(t_f) &= 0, \end{aligned}$$

so ist $\mathbf{q}(t)$ eine gültige und ruckbegrenzte *Trajektorie für Kinematik A.*

2.1.5. Kinematik B

Legen wir nun die Notation für Kinematik B aus Abbildung 2.4 fest. Mit dem Vektor der Maschinenachsen $q = [q_{xc}, q_{yc}, q_{xf}, q_{yf}]^T$ ist die Transformation durch

$$\Lambda_B : \mathcal{D}_{\text{MKS,B}} \ni \mathbf{q} = \begin{bmatrix} q_{xc} \\ q_{yc} \\ q_{xf} \\ q_{yf} \end{bmatrix} \longmapsto \begin{bmatrix} q_{xc} + q_{xf} \\ q_{yc} + q_{yf} \end{bmatrix} \in \mathbb{R}^2.$$

gegeben. Im Folgenden sei kurz $[x_c, y_c, x_f, y_f]^T := [q_{xc}, q_{yc}, q_{xf}, q_{yf}]^T$. Die Maschinenachsen haben hier natürlich keinen unbegrenzten Verfahrbereich. Wir nehmen an, dass nur solche Konturen $h(s)$ in Frage kommen, die alleine von den Hauptachsen x_c und y_c bearbeitet werden können. Die Begrenzung der Grobachsen spielt daher keine Rolle und wir definieren

$$\mathcal{D}_{\text{MKS,B}} = \mathbb{R}^2 \times [-\widehat{S}_{xf}, \widehat{S}_{xf}] \times [-\widehat{S}_{yf}, \widehat{S}_{yf}]$$

als Menge der gültigen Achsstellungen; $\widehat{S}_{xf} \in \mathbb{R}^+$ und $\widehat{S}_{yf} \in \mathbb{R}^+$ sind dabei die maximalen Auslenkungen der Feinachsen. Das Basiskoordinatensystem beschränkt sich im Zusammenhang mit Kinematik B auf die Ebene $\mathbb{R}^2$; die Auslenkung in z–Richtung und die Orientierungswinkel sind konstant und spielen daher keine Rolle.

Bemerkung 2.7. *Bei der Schreibweise der Parametrisierungen der einzelnen Achsen verfahren wir gemäß der Konvention in Bemerkung 2.2. Wichtig ist dabei, dass s immer der Bogenlängenparameter der Konturkurve $h(s)$ ist. Zum Beispiel ist $x_c(s)$ der Verlauf der x_c–Achse, parametrisiert nach s.*

Die Restriktionen für Geschwindigkeit, Beschleunigung und Ruck sind hier die Vektoren

$$\widehat{\mathbf{V}} = \begin{bmatrix} \widehat{V}_{xc} \\ \widehat{V}_{yc} \\ \widehat{V}_{xf} \\ \widehat{V}_{yf} \end{bmatrix}, \qquad \widehat{\mathbf{A}} = \begin{bmatrix} \widehat{A}_{xc} \\ \widehat{A}_{yc} \\ \widehat{A}_{xf} \\ \widehat{A}_{yf} \end{bmatrix}, \qquad \widehat{\mathbf{J}} = \begin{bmatrix} \widehat{J}_{xc} \\ \widehat{J}_{yc} \\ \widehat{J}_{xf} \\ \widehat{J}_{yf} \end{bmatrix}, \tag{2.4}$$

mit positiven, endlichen Einträgen. Die Bezeichnungen für die Bewegungen der Achsen sind analog zum vorherigen Abschnitt, zum Beispiel $x_c(t)$ oder $v_{xc}(t)$. Die Bahngeschwindigkeit $v_b(t) = \dot{s}_b(t)$ erfüllt in diesem Fall die Beziehung

$$v_b(t) = \left\| \begin{bmatrix} v_{xc}(t) + v_{xf}(t) \\ v_{yc}(t) + v_{yf}(t) \end{bmatrix} \right\|_2 .$$

Wie im vorherigen Abschnitt können zwei Arten gültiger Trajektorien definiert werden.

Definition 2.8. *Sei h eine Konturkurve gemäß Definition 2.1. Eine Trajektorie*

$$\mathbf{q} : [t_0, t_f] \ni t \mapsto \begin{bmatrix} q_{xc}(t) \\ q_{yc}(t) \\ q_{xf}(t) \\ q_{yf}(t) \end{bmatrix} = \begin{bmatrix} x_c(t) \\ y_c(t) \\ x_f(t) \\ y_f(t) \end{bmatrix} \in \mathbb{R}^4$$

für Kinematik B heißt gültig und beschleunigungsbegrenzt*, wenn sie die folgenden Bedingungen erfüllt:*

1. *(Bewegung auf der Kontur)*

$$\forall\, t \in [t_0, t_f] : \qquad \begin{bmatrix} x_c(t) + x_f(t) \\ y_c(t) + y_f(t) \end{bmatrix} = h(s_b(t)).$$

2. *(Einhaltung der Achsgrenzen) Für alle $t \in [t_0, t_f]$ gilt*

$$\begin{aligned} |v_{xc}(t)| &\le \widehat{V}_{xc}, & |v_{yc}(t)| &\le \widehat{V}_{yc}, \\ |v_{xf}(t)| &\le \widehat{V}_{xf}, & |v_{yf}(t)| &\le \widehat{V}_{yf}, \\ |a_{xc}(t)| &\le \widehat{A}_{xc}, & |a_{yc}(t)| &\le \widehat{A}_{yc}, \\ |a_{xf}(t)| &\le \widehat{A}_{xf}, & |a_{yf}(t)| &\le \widehat{A}_{yf}. \end{aligned}$$

3. *(Randbedingungen)*

$$\begin{aligned} v_{xc}(t_0) &= 0, & v_{xc}(t_f) &= 0, \\ v_{yc}(t_0) &= 0, & v_{yc}(t_f) &= 0, \\ v_{xf}(t_0) &= 0, & v_{xf}(t_f) &= 0, \\ v_{yf}(t_0) &= 0, & v_{yf}(t_f) &= 0. \end{aligned}$$

4. *(Begrenzte Bahngeschwindigkeit)*

$$\forall\, t \in [t_0, t_f] : \quad 0 \le v_b(t) \le \hat{v}_b.$$

Definition 2.9. *Seien die Voraussetzungen aus Definition 2.8 erfüllt und $\mathbf{q}(t)$ eine Trajektorie, die alle Bedingungen der Definition erfüllt. Gelten zusätzlich*

$$\begin{aligned} |j_{xc}(t)| &\le \widehat{J}_{xc}, & |j_{yc}(t)| &\le \widehat{J}_{yc}, \\ |j_{xf}(t)| &\le \widehat{J}_{xf}, & |j_{yf}(t)| &\le \widehat{J}_{yf}, \end{aligned}$$

so ist $\mathbf{q}(t)$ eine gültige und ruckbegrenzte *Trajektorie für Kinematik B.*

Tabelle 2.1 zeigt zwei typische Konfigurationen für Kinematik B, wie sie für Lasermaschinen vorkommen können. Ist später von Maschinendaten eins oder zwei die Rede, dann

		$\widehat{A}$ [m/sec^2]	$\widehat{V}$ [m/sec]	$\widehat{S}$ [cm]
MD1	x_c	8	1.6	∞
	y_c	8	1.6	∞
	x_f	60	2.5	4.5
	y_f	60	2.5	4.5
MD2	x_c	16	1.8	∞
	y_c	16	1.8	∞
	x_f	60	2.0	2.0
	y_f	60	2.0	2.0
MD1	$\hat{v}_b$	0.25 m/sec		
MD2	$\hat{v}_b$	0.40 m/sec		

Tabelle 2.1.: Konfigurationen, die für Kinematik B in dieser Arbeit betrachtet werden.

sind damit die Konfigurationen MD1 beziehungsweise MD2 gemeint. Der angegebene Wert für den Bahnvorschub $\hat{v}_b$ bezieht sich dabei auf den Schneidemodus der Maschine. Beim Positionieren ist der Vorschub durch die Grenzen der Achsgeschwindigkeit beschränkt. Die späteren Kapitel greifen hauptsächlich auf MD2 zurück, da mit ihnen die Parametrisierung der Verfahren schwieriger ist und die Beispiele damit eine größere Aussagekraft haben (siehe Abschnitt 4.3).

Alle Untersuchungen in dieser Arbeit im Zusammenhang mit Kinematik B können auf eine ähnliche dreidimensionale Kinematik übertragen werden. Auf den Haupt- und Zusatzachsen von Kinematik B sind jeweils eine weitere Linearachse in z–Richtung angeordnet.

2.2. Bewegungsführung auf vorgegebenen Bahnen

2.2.1. Bewegungsführung als Optimalsteuerungsproblem

Die Bedingungen aus Definition 2.5 und 2.6 sind typisch für Problemstellungen in der Optimalsteuerungstheorie. Dort werden sie von einer Zielfunktion ergänzt und die Lösung des sogenannten Optimalsteuerungsproblems ist eine gültige Trajektorie, die die Zielfunktion minimiert (siehe Anhang A.1). Die Steuergrößen sind dabei die Achsbeschleunigungen im beschleunigungsbegrenzten Fall beziehungsweise die Achsrucke im ruckbegrenzten Fall. Die Zeitdauer

$$Z[\mathbf{q}] = t_f - t_0 \tag{2.5}$$

ist beispielsweise eine solche Zielfunktion und eine Lösung des Problems entspricht der Trajektorie mit zeitoptimaler Bewegung.

Bei einer festen Kontur $h(s)$ gibt es für Kinematik A wegen

$$\mathbf{q}(t) = h(s_b(t))$$

zu jeder Bahntrajektorie $s_b(t)$ genau eine Achstrajektorie $\mathbf{q}(t)$. Aus diesem Grund ergeben sich aus den Bedingungen der Definition 2.5 äquivalente Forderungen an die Bahntrajektorie $s_b(t)$, wenn sie zu einer gültigen Achstrajektorie gehören soll. Bedingung eins fordert die Erfüllung der Randbedingungen

$$s_b(t_0) = s_0 \qquad \text{und} \qquad s_b(t_f) = s_f.$$

Aus der zweiten folgen mit $[x(s), y(s)]^T = h(s)$ und

$$v_x(t) = x'(s_b(t))\ v_b(t), \tag{2.6}$$
$$a_x(t) = x''(s_b(t))\ v_b(t)^2 + x'(s_b(t))\ a_b(t) \tag{2.7}$$

für $s_b(t)$ die Forderungen

$$\widehat{V}_x \geq |x'(s_b(t))\ v_b(t)|, \tag{2.8}$$
$$\widehat{A}_x \geq |x''(s_b(t))\ v_b(t)^2 + x'(s_b(t))\ a_b(t)|, \tag{2.9}$$

und natürlich die entsprechenden Ungleichungen der y–Achse. Mit

$$v_b(t_0) = 0 \qquad \text{und} \qquad v_b(t_f) = 0$$

ist Punkt drei der Definition erfüllt und die vierte Bedingung kann direkt übernommen werden.

Mit der Zielfunktion (2.5) hat man ein Optimalsteuerungsproblem, das zum obigen äquivalent ist. Die Zustands- beziehungsweise Steuergrößen sind hier $s_b(t)$ und $v_b(t)$ beziehungsweise $a_b(t)$, die Anzahl ist also geringer als bei der obigen Problemstellung. Dafür ist die Nebenbedingung (2.9) nichtlinear. Andererseits entfällt die nichtlineare Gleichungsnebenbedingung (2.3).

Für den ruckbegrenzten Fall mit Definition 2.6 kommt mit

$$j_x(t) = x'''(s_b(t))\ v_b(t)^3 + 3\ x''(s_b(t))\ a_b(t)\ v_b(t) + x'(s_b(t))\ j_b(t)$$

für $s_b(t)$ die Bedingung

$$\widehat{J}_x \geq |x'''(s_b(t))\ v_b(t)^3 + 3\ x''(s_b(t))\ a_b(t)\ v_b(t) + x'(s_b(t))\ j_b(t)|$$

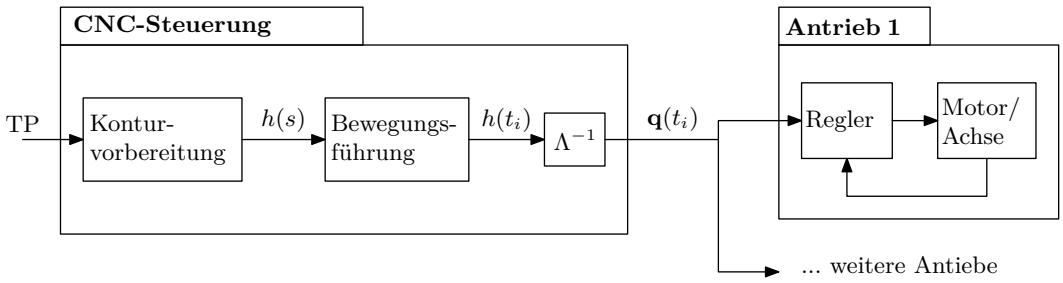

Abbildung 2.5.: Prozesskette in einer Werkzeugmaschine.

hinzu. Natürlich muss auch die analoge Bedingung für die y–Achse erfüllt werden. Außerdem müssen die zusätzlichen Randbedingungen

$$a_b(t_0) = 0 \qquad \text{und} \qquad a_b(t_f) = 0$$

gelten. Das Optimalsteuerungsproblem für den ruckbegrenzten Fall ist somit komplexer.

2.2.2. Prozesskette in CNC–Steuerungen

Dieser Abschnitt erläutert, wie in einer Werkzeugmaschine aus der vom Benutzer eingegebenen Kontur derartige Achsbewegungen der Kinematik entstehen, dass der Werkzeugkopf die erwünschte Bahn abfährt. Die Darstellung in diesem und dem folgenden Abschnitt begrenzt sich auf die für diese Arbeit wesentlichen Punkte der Bewegungserzeugung und hält sich dabei an die Beschreibung in [28]. Einen umfassenderen Blick auf CNC–Steuerungen und Werkzeugmaschinen bieten [22] oder [39].

Es folgt eine Erläuterung der Prozesskette aus Abbildung 2.5. Der Maschinenbenutzer führt der Steuerung ein Teileprogramm (TP) zu. Es enthält eine Vielzahl von Informationen und Daten, die zur Abarbeitung einer Kontur notwendig sind. Für unsere Zwecke sind zum einen die geometrischen Informationen der abzufahrenden Kontur interessant. Sie bestehen aus einer Liste von Kreisbögen und Polynomstücken, den sogenannten NC–Sätzen. Zum anderen befinden sich darin Wechselbefehle für den Bearbeitungsmodus — zum Beispiel Ein- und Ausschalten des Lasers beim Laserschneiden — und Bahngeschwindigkeitsgrenzen für die Abschnitte der Kontur. Alles andere spielt für diese Arbeit keine Rolle.

Der erste Schritt in der Steuerung, die Konturvorbereitung, ist optional. Darin findet eine Näherung der Kontur durch eine Splinekurve statt. Dieser Schritt soll hier aber nicht weiter beachtet werden. Wir nehmen an, dass die Ausgabe eine Konturkurve $h(s)$ in ihrer Bogenlängenparametrisierung ist. Die endlich vielen Stellen, an denen die Kurve nur stetig und nicht differenzierbar ist, sind die Ecken der Kontur. Gemäß Definition 2.1 enthält eine Kontur n_h Stücke und in Anlehnung an den vorherigen Absatz nennen wir jedes Stück auch *Satz*.

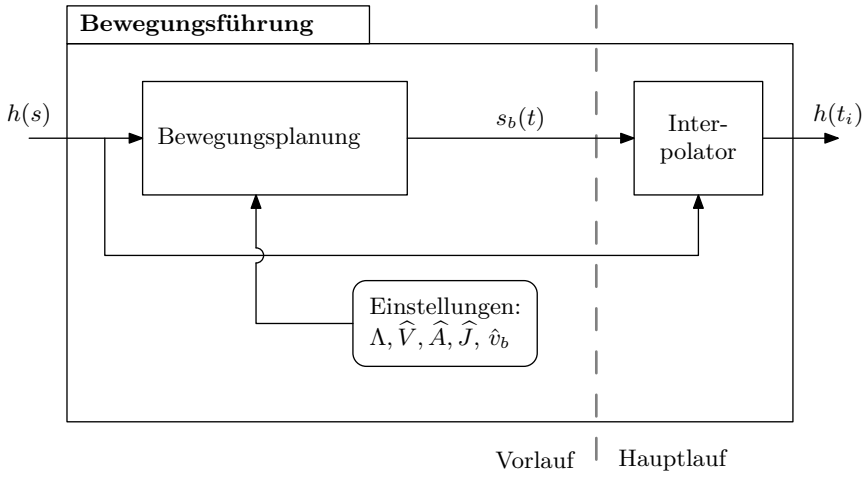

Abbildung 2.6.: Struktur Bewegungsführung.

Die Datenverarbeitung und -bearbeitung in der Steuerung geht abschnittsweise vor. Das heißt, jeder Block im Schaubild behandelt die Kontur immer nur auf einem Teilintervall $[s_{h,k_1}, s_{h,k_2}]$ von $[s_0, s_f]$ mit $k_1, k_2 \in \{0, \ldots, n_h\}$, $k_1 < k_2$, und gibt das Ergebnis an den nächsten Block weiter. Anschaulich kann man sich das wie ein begrenztes Fenster vorstellen, das über die Kontur wandert und sich immer um eine bestimmte Anzahl von Sätzen verschiebt.

Der zweite Schritt aus Abbildung 2.5 ist die Bewegungsführung. Sie berechnet den Verlauf des Werkzeugkopfes über der Zeit aus der Kurve $h(s)$, der kinematischen Transformation und den Achsrestriktionen. Die Ausgabe sind Konturwerte in einem äquidistanten Zeitraster, für Kinematik A beispielsweise[3] $h(t_i) = (x(t_i), y(t_i))^T$. Der konstante Zeitabstand t_{ipo} ist die Taktzeit der (Lage)Regelkreise der Antriebe und ist in dieser Arbeit durchgängig 0.001 Sekunden. Wie die $h(t_i)$ entstehen, sehen wir am Ende dieses Abschnittes. Zuletzt berechnet die inverse kinematische Transformation Λ^{-1} die Achsverläufe $\mathbf{q}(t_i)$. Sie verlassen die Steuerung und jede Komponente ist der Sollwerteingang eines Achsantriebs.

In den Antrieben sorgen komplexe Regelkreise dafür, dass die realen Achsen den Sollwerten aus der Steuerung mit hoher Präzision und Dynamik folgen. Über die Struktur und Auslegung der Regelkreise soll in dieser Arbeit nicht weiter eingegangen werden. Wir nehmen an, dass die Sollwerte akzeptabel geregelt werden können.

Wie gerade erwähnt, berechnet die Bewegungsführung (Abbildung 2.6) die Trajektorien im BKS für diskrete Zeitpunkte. In einem ersten Schritt, der Bewegungsplanung, ermittelt sie aus der kinematischen Transformation und den Achsrestriktionen eine gültige Bahntrajektorie $s_b(t)$. Sie und die Kontur $h(s)$ sind dann die Eingabe des Interpolators. Er bestimmt die diskreten Werte $h(t_i) = h(s_b(t_i))$.

[3] i ist hier einfach ein allgemeiner Laufindex.

Ein wichtiger Aspekt der Bewegungsführung ist der vor dem Interpolator stattfindende Wechsel vom Vorlauf in den Hauptlauf. Der Vorlauf arbeitet präparativ und ist mit der Bestimmung der Bahntrajektorie immer weiter als die reale Position des Kopfes, sozusagen „schneller als Echtzeit“. Diese Eigenschaft nennt man auch *Lookahead*. Der Hauptlauf arbeitet in dem festen Takt t_{ipo}, und in diesen Abständen verlässt immer ein Wert $h(t_i)$ den Interpolator, die Bewegungsführung und letztendlich auch die Steuerung. Dieser Aspekt wird weiter unten in Abschnitt 4.3 noch einmal aufgefasst und weiter ausgeführt.

2.2.3. Algorithmus für die beschleunigungsbegrenzte Bewegungsplanung

Diese Arbeit behandelt ausschließlich beschleunigungsbegrenzte Bewegungen der Maschinenachsen. Daher stellt dieser Abschnitt ein Verfahren für Kinematik A vor, das solche Trajektorien berechnet. Es kann für die Bewegungsplanung eingesetzt werden. Eine Methode für den ruckbegrenzten Fall findet sich in [12] oder [19]. Vor der Beschreibung des Verfahrens sind einige Vorbereitungen notwendig.

Wir betrachten hier eine Konturkurve $h(s)$ gemäß Definition 2.1. Wie erwähnt gibt es isolierte Stellen, an denen $h(s)$ nur stetig ist. Da dort die Bahngeschwindigkeit des Kopfes und damit auch die Geschwindigkeiten der x– und y–Achse null sein müssen, kann die Bewegungsplanung nacheinander die zwischen den Ecken liegenden Abschnitte abarbeiten. Wir können also in diesem Abschnitt annehmen, dass $h(s)$ überall C^1–stetig ist und natürlich weiter Definition 2.1 genügt.

Kommen wir zu den Bahntrajektorien $s_b(t)$, $v_b(t)$ und $a_b(t)$ aus (2.2). Ist $s_b(t)$ streng monoton wachsend, existiert ihre ebenfalls streng monoton wachsende, bijektive Inverse

$$t_b : [s_0, s_f] \ni s \longmapsto t_b(s) = s_b^{-1}(s) \in [t_0, t_f].$$

Sie ist auf (s_0, s_f) stetig differenzierbar mit

$$t_b'(s) = \frac{\mathrm{d}t_b}{\mathrm{d}s}(s) = \frac{1}{\dot{s}_b(t_b(s))} = \frac{1}{v_b(t_b(s))}.$$

Falls $v_b(t_0) = v_b(t_f) = 0$ gilt, folgt

$$\lim_{s \searrow s_0} t_b'(s) = \lim_{s \nearrow s_f} t_b'(s) = \infty.$$

Wir verwenden im Folgenden neben der Zeitparametrisierung auch die Parametrisierung nach der Bogenlänge s, $v_b(s) = v_b(t_b(s))$ und $a_b(s) = a_b(t_b(s))$. Zwischen beiden besteht der wichtige Zusammenhang

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}s}\left(v_b(s)^2\right) &= \frac{\mathrm{d}}{\mathrm{d}s}\left(v_b(t_b(s))^2\right)\\ &= 2v_b(t_b(s)) \cdot a_b(t_b(s)) \cdot t_b'(s)\\ &= 2v_b(t_b(s)) \cdot a_b(t_b(s)) \cdot \frac{1}{v_b(t_b(s))}\\ &= 2a_b(s),\end{aligned} \tag{2.10}$$

falls $v_b(s)^2$ differenzierbar ist, oder anders geschrieben

$$v_b(s)^2 - v_b(s_0)^2 = 2\int_{s_0}^{s} a_b(\sigma)d\sigma.$$

Das Verfahren für die beschleunigungsbegrenzte Bewegung wird unter einfacheren Bedingungen als denen in Abschnitt 2.2.1 eine Bewegung berechnen, allerdings dazu die zeitoptimale. Eine Vereinfachung ist wegen einer praktischen Anforderung notwendig. Eine Bewegung des Werkzeugkopfes muss derart sein, dass nach einem spontanen Eingriff des Maschinenbenutzers der Werkzeugkopf auf der Kontur bleibend abbremsen kann. Das ist mit der „echten“ zeitoptimalen Bewegung[4] nicht möglich. Realisiert ist das jederzeitige Abbremsen auf der Bahn durch die Bedingung

$$\left|x'(s)\ v_b^2(s)\right| \leq K_\mathrm{b}\widehat{A}_x \tag{2.11}$$

und dem Analogon für die y–Achse. Die Konstante K_b ist dabei ein Wert zwischen 0 und 1, den der Maschinenbenutzer festlegt. Mit (2.11) folgt für festes s, dass für ein $a_b(s)$ mit

$$a_b(s) \in \left[-\widehat{A}_x\frac{1-K_\mathrm{b}}{|x'(s)|}, \widehat{A}_x\frac{1-K_\mathrm{b}}{|x'(s)|}\right], \qquad K_\mathrm{b} \in (0,1),$$

die Ungleichung (2.9) sicher erfüllt ist. Damit lässt sich die Bewegung auf der Bahn jederzeit abbremsen. Je größer K_b ist, desto schneller kann nach einem Eingriff auf der Bahn angehalten werden, desto länger dauert aber auch die ursprüngliche Bewegung. Falls nicht anders erwähnt, wird in dieser Arbeit $K_\mathrm{b} = 0.7$ verwendet.

Phase I

Das Verfahren für die Bewegungsplanung besteht aus zwei Phasen. Die erste berechnet *Grenzkurven* $v_b^{\downarrow}(s)$ und $a_b^{\downarrow}(s)$ für Bahngeschwindigkeit und -beschleunigung. Die in der

[4] Ein entsprechender Algorithmus ist in [21] beschrieben.

zweiten Phase berechnete *Bahntrajektorie*[5] $v_b(s)^2$ und der dazugehörige Verlauf von $a_b(s)$ dürfen diese Werte nicht überschreiten. Die Grenzkurven ergeben sich mit den Überlegungen des vorherigen Abschnitts. Aus (2.8) folgt die Funktion

$$v_{bx1}(s) = \frac{\widehat{V}_x}{|x'(s)|} \tag{2.12}$$

und aus (2.11) leitet sich

$$v_{bx2}(s)^2 = K_{\mathrm{b}} \frac{\widehat{A}_x}{|x''(s)|} \tag{2.13}$$

ab. Die zweite Ableitung der Konturkurve muss an den isolierten Punkten $\mathbf{S}_h$ nicht stetig sein, hat dort aber links- und rechtsseitige Grenzwerte. An diesen Stellen soll

$$v_{bx2}(s)^2 = \min \left\{ \frac{K_{\mathrm{b}} \widehat{A}_x}{|x''(s^-)|}, \frac{K_{\mathrm{b}} \widehat{A}_x}{|x''(s^+)|} \right\} \tag{2.14}$$

gelten[6]. Angenommen, an einer Stelle $\tilde{s}$ ist $x''(\tilde{s}^-)$ für $v_{bx2}(\tilde{s})^2$ verantwortlich. Dann gibt es ein $\delta > 0$, sodass $v_{bx2}(s)^2$ auf $[\tilde{s}-\delta, \tilde{s})$ stetig ist. Umgekehrt verhält es sich, wenn $x''(\tilde{s}^+)$ verantwortlich ist. Analog zu (2.12), (2.13) und (2.14) ergeben sich $v_{by1}(s)$ und $v_{by2}(s)^2$ aus der y–Komponente der Kontur.

Zusammen definiert man die Grenzkurve der Bahngeschwindigkeit durch

$$v_b^{\downarrow}(s) := \min \left\{ \hat{v}_b, v_{bx1}(s), v_{by1}(s), v_{bx2}(s), v_{by2}(s) \right\} \tag{2.15}$$

für jedes $s \in [s_0, s_f]$. Sie ist begrenzt und höchstens an den Stellen $\mathbf{S}_h$ nicht stetig. Mit

$$a_{bx}(s) = \frac{\widehat{A}_x - |x''(s) v_b^{\downarrow}(s)^2|}{|x'(s)|}, \qquad s \in [s_0, s_f] \setminus \mathbf{S}_h, \tag{2.16}$$

und dem Analogon $a_{by}(s)$ aus der y–Komponente ist der Grenzverlauf der Bahnbeschleunigung

$$a_b^{\downarrow}(s) := \min \left\{ a_{bx}(s), a_{by}(s) \right\}, \qquad s \in [s_0, s_f] \setminus \mathbf{S}_h. \tag{2.17}$$

Die Werte für $a_b^{\downarrow}(s)$ an den diskreten Stellen $\mathbf{S}_h$ sind nicht eindeutig und benötigen Grenzwertbetrachtungen. Das folgt in der Beschreibung der zweiten Phase. Die Zuweisungen (2.12), (2.13) und (2.16) verzichten darauf, die Fälle $|x'(s)| = 0$ beziehungsweise $|x''(s)| = 0$ abzufangen. Es macht aber nichts, wenn einzelne Komponenten von (2.15) oder (2.17) nicht definiert sind, denn eine ist es immer. Das folgt, da wegen der Bogenlängenparametrisie-

[5] Hier findet der Begriff „Trajektorie“ auch bei einer Funktion Verwendung, die nicht über der Zeit parametrisiert ist.

[6] $x''(s^-) := \lim_{\sigma \nearrow s} x''(\sigma)$ und $x''(s^+) := \lim_{\sigma \searrow s} x''(\sigma)$

rung von $h(s) = [x(s), y(s)]^T$ die Beziehung $x'(s)^2 + y'(s)^2 = 1$ erfüllt ist und daher für festes s die Werte $x'(s)$ und $y'(s)$ nicht beide gleichzeitig null sein können.

Phase II

Die zweite Phase der Bewegungsplanung löst ein Optimalsteuerungsproblem mit der Bewegungszeit als zu minimierende Zielfunktion. Wir nehmen an, dass $v_b^{\downarrow}(s)^2$ bis auf eine endliche Anzahl von Punkten aus $[s_0, s_f]$ differenzierbar ist. Für einen beliebigen Verlauf der Bahngeschwindigkeit $v_b(s)$ ergibt sich die Bahntrajektorie $s_b(t)$ als Lösung der Differentialgleichung

$$v_b(s_b(t)) = \dot{s}_b(t)$$

mit einer Anfangsbedingung $s_b(t_0) = s_0$. Das führt zu der Beziehung

$$t - t_0 = \int_{s_0}^{s_b(t)} \frac{1}{v_b(\sigma)} d\sigma.$$

Insbesondere ist die Bearbeitungszeit

$$t_f - t_0 = \int_{s_0}^{s_f} \frac{1}{v_b(\sigma)} d\sigma$$

die oben erwähnte Zielfunktion.

Die Steuergröße $a_b(s)$ und die Zustandsgröße $v_b(s)$ sind durch (2.10) miteinander verknüpft. Mit den Grenzkurven aus der ersten Phase sind

$$0 \leq v_b(s) \leq v_b^{\downarrow}(s) \quad \text{und} \quad -a_b^{\downarrow}(s) \leq a_b(s) \leq a_b^{\downarrow}(s)$$

die Nebenbedingunen und

$$v_b(s_0) = v_{b0} \qquad \text{und} \qquad v_b(s_f) = v_{bf}$$

bilden die Randbedingungen. Bei einer Bewegung, die zwei Ecken der Kontur verbindet, gilt natürlich $v_{b0} = v_{bf} = 0$.

Es folgt die Beschreibung eines Algorithmus, der die Lösung dieses Optimalsteuerungsproblems berechnet. Dafür sind noch einige Vorbereitungen notwendig. Zuerst definieren wir Stellen und Bereiche, an denen die Grenzkurve lokal minimal ist.

Definition 2.10. *Sei $v_b^{\downarrow}(s)^2$ eine auf $[s_0, s_f]$ definierte Grenzkurve, die nur an endlich vielen Stellen nicht differenzierbar ist, die Ableitung dort aber links- und rechtsseitige Grenzwerte besitzt. Dann definieren wir die Menge*

$$\mathbf{S}_v = \left\{ \left(s'_{1,1}, s'_{2,1} \right), \ldots, \left(s'_{1,n_v}, s'_{2,n_v} \right) \right\},$$

sodass alle Paare $(s_1, s_2) \in \mathbf{S}_v$ *aus* $(s_0, s_f) \times (s_0, s_f)$ *sind und die folgenden Eigenschaften erfüllen:*

1. $s_1 \leq s_2$
2. $\exists\, \delta > 0 :\ v_b^{\downarrow}(s)^2 > v_b^{\downarrow}(s_1)^2, s \in (s_1 - \delta, s_1)$
3. $\exists\, \delta > 0 :\ v_b^{\downarrow}(s)^2 > v_b^{\downarrow}(s_2)^2, s \in (s_2, s_2 + \delta)$
4. $v_b^{\downarrow}(s)^2 = v_b^{\downarrow}(s_1)^2, s \in [s_1, s_2]$

Außerdem ist

$$v_{b,\min}^2 := \min \left\{\ v \in \mathbb{R}^+ \mid \exists (s_1, s_2) \in \mathbf{S}_v :\ v = v_b^{\downarrow}(s_1)^2\ \right\}. \tag{2.18}$$

Der Algorithmus berechnet ausgehend von den Randbedingungen Bahntrajektorienstücke und verbindet sie zu einer Trajektorie über ganz $[s_0, s_f]$. Dabei gibt es Abschnitte, auf denen die Bahntrajektorie $v_b(s)^2$ maximal wächst, und andere, wo sie steilstmöglich fällt. Soll eine Bahntrajektorie von einem s' bei gegebenem $v_b(s')^2$ nach rechts auf der s–Achse maximal wachsen, dann gilt

$$v_b(s)^2 = v_{b\rightarrow}(s)^2 := v_b(s')^2 + \frac{1}{2}\int_{s'}^{s} a_b^{\downarrow}(\sigma)\ \mathrm{d}\sigma, \qquad s > s',$$

solange $v_b(s)^2 \leq v_b^{\downarrow}(s)^2$ erfüllt ist. Die Integration kann durchgeführt werden, auch wenn $a_b^{\downarrow}(s)$ an den Stelle $\mathbf{S}_h$ nicht definiert ist. Dann läuft die Bahntrajektorie auf der Grenzkurve, solange

$$-2a_b^{\downarrow}(s^-) \leq \frac{\mathrm{d}}{\mathrm{d}s} v_b^{\downarrow}(s^-)^2 \leq 2a_b^{\downarrow}(s^-)$$

gilt. Sie verlässt die Grenzkurve wieder, falls $\frac{\mathrm{d}}{\mathrm{d}s} v_b^{\downarrow}(s^+)^2 > 2a_b^{\downarrow}(s^+)$ gilt und bricht darauf ab, falls $\frac{\mathrm{d}}{\mathrm{d}s} v_b^{\downarrow}(s^+)^2 < -2a_b^{\downarrow}(s^+)$ gilt oder $v_b^{\downarrow}(s)^2$ nach unten springt. Analog kann von s' nach links eine Bahntrajektorie mit steilstem Abstieg berechnet werden. Es gilt

$$v_b(s)^2 = v_{b\leftarrow}(s)^2 := v_b(s')^2 + \frac{1}{2}\int_{s'}^{s} -a_b^{\downarrow}(\sigma)\ \mathrm{d}\sigma, \qquad s < s',$$

und auch hier verläuft die Bahntrajektorie mit analogen Kriterien abwechselnd auf und unter der Grenzkurve und irgendwann bricht sie ab. Dieses Vorgehen nutzt Algorithmus 2.1 zur Lösung des Optimalsteuerungsproblems. Sein Funktionieren wird in [29] untersucht und bewiesen.

Es folgt eine grobe Erläuterung zur Entstehung der Bahntrajektorie aus Abbildung 2.7 mit Algorithmus 2.1. In der ersten Rekursionsstufe erzeugen die Schritte zwei und drei die Stücke der Bahntrajektorie bis zum Wert $v_{b,\min,1}^2$. Dann folgen zwei rekursive Aufrufe in Schritt sechs. Im ersten wird die Bahntrajektorie ohne eine weitere Rekursion ermittelt und nach Schritt vier zurückgegeben. Im zweiten bestimmen wieder die Schritte zwei und drei

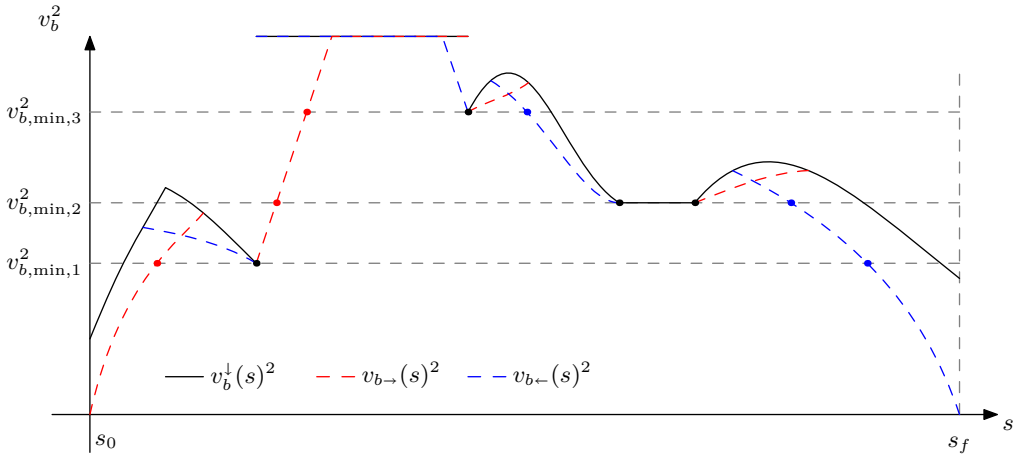

Abbildung 2.7.: Prinzipielle Entstehung einer Bahntrajektorie mit Algorithmus 2.1.

die Stücke der Bahntrajektorie, diesmal bis $v_{b,\min,2}^2$. Auch hier gibt es in Schritt sechs zwei rekursive Aufrufe. Beim zweiten entsteht die Bahntrajektorie ohne weitere Rekursionen. Beim ersten ist eine zusätzliche notwendig, bis der Rekursionsanker in Schritt vier greift.

Die zusammen mit den späteren Verfahren dieser Arbeit genutzte Implementierung der beschleunigungsbegrenzten Bewegungsplanung arbeitet mit einer Diskretisierung $\{s_j | j = 1, \ldots, n_b\}$ des Bahnparameters s. Die obigen Beziehungen werden durch ein einfaches Schema genähert. Die Bahnbeschleunigung zwischen zwei Abtastpunkten s_{j-1} und s_j ist konstant mit $a_b(s_{i-j})$. Die Werte t'_j bezeichnen die Zeitpunkte zu den Bahnpunkten s_j, also $s_b(t'_j) = s_j$. Insgesamt gelten für $t \in [t'_{j-1}, t'_j]$ die Gleichungen

$$\begin{aligned} v_b(t) &= v_b(t'_{j-1}) + a_b(s_{j-1})(t - t'_{j-1}), \\ s_b(t) &= s_b(t'_{j-1}) + v_b(t'_{j-1})(t - t'_{j-1}) + a_b(s_{j-1})(t - t'_{j-1})^2. \end{aligned}$$

Die Bahntrajektorie $s_b(t)$ ist also ein Spline vom Grad zwei mit den Bruchstellen t'_j (siehe auch Abschnitt 4.2.1). Eine derartige, für alle $t \in [t_0, t_f]$ definierte Darstellung von $s_b(t)$ ist auch notwendig. Sie kann an äquidistant verteilten Punkten t_i ausgewertet und in $h(s)$ eingesetzt werden. Das passiert im Interpolator mit der Abtastzeit t_{ipo} und somit hat man den Ausgang der Bewegungsführung. Es ist klar, dass diese Punkte nur eine Näherung der Abtastpunkte der „theoretisch exakten“ Lösung des Optimalsteuerungsproblems sind. Allerdings liegen die Positionswerte exakt auf der Kontur. Nur bei Geschwindigkeit und Beschleunigung kommt es zu Fehlern, insbesondere sind Überschreitungen der Achsgrenzen möglich. Mit der Abtastdichte über s hat man diese Fehler aber unter Kontrolle. Je feiner sie ist, desto kleiner ist auch der Fehler der Näherung. Dort, wo dieser Algorithmus später Verwendung findet, wurde so fein über s abgetastet, dass der Fehler keine Rolle mehr spielt.

Eingabe: $\mathbf{S}_v$, v_{b0}^2, v_{bf}^2, $v_b^{\downarrow}(s)^2$, $a_b^{\downarrow}(s)$, $[s_0, s_f]$

1. Bestimme $v_{b,\min}^2$ gemäß (2.18).
2. Aufbau der gültigen Bahntrajektorie $v_{b\rightarrow}(s)^2$ mit maximal möglicher Steigung von s_0 mit Randwert v_{b0}^2 nach rechts, bis in einem Punkt s_l eine der folgenden Eigenschaften erfüllt ist:
   - $v_{b\rightarrow}(s^+)^2 \geq v_{b,\min}^2$
   - $s = s_f$
   - $v_{b\rightarrow}(s^+)^2 < -2a_b^{\downarrow}(s)$ oder $v_b^{\downarrow}(s)^2$ macht einen Sprung nach unten
3. Aufbau der gültigen Bahntrajektorie $v_{b\leftarrow}(s)^2$ mit minimaler Steigung von s_f mit Randwert v_{bf}^2 nach links, analog wie in Schritt 2, bis zu einem Punkt s_r.
4. Gilt $s_l \geq s_r$, schneiden sich $v_{b\rightarrow}(s)^2$ und $v_{b\leftarrow}(s)^2$ und die Bahntrajektorie ist $v_b(s)^2 := \min\{v_{b\rightarrow}(s)^2, v_{b\leftarrow}(s)^2\}$, $s \in [s_0, s_f] \rightarrow$ Fertig!
   Andernfalls weiter mit Schritt 5.
5. Bestimme die geordnete Menge

$$\begin{aligned}\mathbf{S}'' &:= \{(s''_{1,1}, s''_{2,1}), \ldots, (s''_{1,n''}, s''_{2,n''})\} \\ &:= \{(s_l, s_l)\} \cup \left\{ (s_1, s_2) \in \mathbf{S}_v \mid s_1 > s_l \wedge s_2 < s_r \wedge v_b^{\downarrow}(s)^2 = v_{b,\min}^2 \right\} \cup \{(s_r, s_r)\}\end{aligned}$$

6. Für $i = 1, \ldots, n'' - 1$:
   - $\mathbf{S}_{v,i} := \{ (s_1, s_2) \in \mathbf{S}_v \mid s_1 > s''_{2,i} \wedge s_2 < s''_{1,i+1} \}$
   - Rekursiver Aufruf des Algorithmus mit $\mathbf{S}_{v,i}$, $v_{b0}^2 = v_{bf}^2 = v_{b,\min}^2$ und $[s''_{2,i}, s''_{1,i+1}]$.
   - Ergebnis ist $v_b(s)^2$ auf $[s''_{2,i}, s''_{1,i+1}]$.
7. Ergänze $v_b(s)^2$ auf $[s_0, s_l]$ durch $v_{b\rightarrow}(s)^2$.
   Ergänze $v_b(s)^2$ auf $[s_r, s_f]$ durch $v_{b\leftarrow}(s)^2$.
   Ergänze $v_b(s)^2$ durch $v_{b,\min}^2$ an den verbleibenden Stellen.

Ausgabe: $v_b(s)^2$ auf $[s_0, s_f]$

Algorithmus 2.1: Rekursives Verfahren zur Berechnung der Bahntrajektorie in der zweiten Phase der Bewegungsplanung.

Der hier beschriebene Algorithmus beschränkt sich nicht auf Kinematik A. Mit leichten Modifikationen durch die Hinzunahme der kinematischen Transformation in der ersten Phase kann er auf allgemeinere, nichtredundante Kinematiken übertragen werden (siehe [29]).

3. Bekannte Verfahren zur Steuerung redundanter Kinematiken

Mit der im vorherigen Abschnitt vorgestellten Methode lässt sich ein redundantes System, wie etwa Kinematik B, nicht steuern. Bei Kinematik A gibt es zu jedem Punkt auf einer Kontur eine eindeutige Stellung der Achsen x und y, da die kinematische Transformation Λ_A die Identität ist und damit trivialerweise überall eindeutig invertierbar. Das ist für Λ_B nicht der Fall. Es gibt für jede Position des Werkzeugkopfes unendlich viele Stellungen der Achsen mit

$$x_g + x_f = x \qquad \text{und} \qquad y_g + y_f = y,$$

selbst wenn man den begrenzten Verfahrbereich der Feinachsen berücksichtigt. Deswegen kann man bei Kinematik B im Gegensatz zur nichtredundanten Kinematik die Bewegung der Achsen nicht ohne weiteres über den Bahnparameter beziehungsweise die Bahngeschwindigkeit führen, wie es das Verfahren in Abschnitt 2.2.3 tut.

Aus diesem Grund sind andere Steuerungsverfahren für Maschinen mit redundanten Achsen notwendig. Diese Thematik ist schon seit längerem Gegenstand der Forschung. Die folgenden vier Abschnitte beschreiben bekannte Ansätze. Zu ihrer Realisierung sind, außer beim ersten Ansatz, Modifikationen im Steuerungs- beziehungsweise Antriebsschema notwendig. Die folgenden Darstellungen basieren auf den Beschreibungen von Abschnitt 2.2 und insbesondere auf Abbildung 2.5.

Die hier vorgestellten Ansätze sind in einigen Belangen nicht zufriedenstellend. Das nachfolgende Kapitel 4 beschreibt einen neuartigen Ansatz, der gewisse Vorteile gegenüber den bekannten mitbringt.

3.1. Bewegung durch Nachsetzen

Dieser Ansatz unterscheidet zwei verschiedene Bewegungsmodi. Im ersten muss der Werkzeugkopf eine einfache Punkt–zu–Punkt–Bewegung durchführen, an deren Anfang und Ende die Positionen der Einzelachsen fest vorgegeben sind. Die Bahngeschwindigkeit $\hat{v}_b$ des Werkzeugkopfes ist dabei nicht begrenzt. Derartige Bewegungen lassen sich sehr einfach berechnen, da jede Achse für sich betrachtet werden kann. Im zweiten Modus bewegt sich immer nur ein Teilsystem, entweder die Grobachsen x_c und y_c oder die Feinachsen x_f und y_f, über einen Abschnitt der Kontur aus mehreren Sätzen. Die herkömmliche Bewegungsführung kann für diese Abschnitte die Bewegung erzeugen.

Die Realisierung dieses Ansatzes kann durch eine Modifikation des Teileprogramms geschehen, bevor es in die CNC–Steuerung eintritt. Dabei müssen die Sätze der Kontur im Teileprogramm mit dem jeweiligen Bearbeitungsmodus und gegebenenfalls mit dem verwendeten Teilsystem gekennzeichnet werden. Dieser Ansatz ist Inhalt des Patentes [24].

Eine Besonderheit liegt vor, wenn die Maschine nur die beiden Bewegungsarten Schneiden[1] und Positionieren unterscheidet. Das ist etwa beim 2–D–Laserschneiden der Fall, wie schon in Abschnitt 2.1.2 erwähnt. Dabei ist das Positionieren eine einfache Punkt–zu–Punkt–Bewegung und nur diese Sätze werden, wenn es der Verfahrbereich der Zusatzachsen verlangt, mit dem ersten Bewegungsmodus im Teileprogramm gekennzeichnet. Die Bewegung auf einem Schneideblock geschieht mit dem zweiten Modus, ohne dass zwischenzeitliche Wechsel der Teilsysteme stattfinden. Um möglichst produktiv zu sein, schneiden die Feinachsen jeden Block, den ihr Verfahrbereich zulässt. Darin kann gerade der entscheidende Nachteil dieses Ansatzes liegen. Angenommen, es überwiegen im Teileprogramm Schneideblöcke, die wegen ihrer Größe alleine von den Grobachsen bearbeitet werden müssen. In diesem Fall nutzt die Maschine nicht die hohe Dynamik der Feinachsen aus und die Gesamtproduktivität ist kaum besser als die einer Maschine, die nur aus den Hauptachsen besteht.

Es ist natürlich möglich, jeden Schneideblock mit den Feinachsen zu beginnen. Wenn eine davon ihre Verfahrbereichsgrenze erreicht, muss der Werkzeugkopf zum Stehen gekommen sein und die Teilsysteme können sich neu ausrichten. Danach können sich wieder alleine die Feinachsen bewegen. Dieses Vorgehen ist allerdings in der Anwendung oft unerwünscht oder sogar verboten, da sich das längere Stillstehen des Kopfes während des Schneidens negativ auf die Schnittqualität auswirken kann.

3.2. Geometrische Zerlegung

Der folgende Ansatz erweitert Λ_B aus Abschnitt 2.1.5 zu einer invertierbaren kinematischen Transformation, sodass die Steuerung lediglich durch einen Aufteilungsalgorithmus ergänzt werden muss. Die Bewegungsplanung der Steuerung beziehungsweise das Vorgehen aus Abschnitt 2.2.3 funktioniert dabei nach einer leichten Anpassung wie gehabt. Abbildung 3.1 zeigt die modifizierte Struktur der Steuerung. Dieser Ansatz ist zur Patentierung angemeldet, [18] ist die Offenlegungsschrift.

Die Funktion

$$\Lambda_{\widetilde{B}} : \mathcal{D}_{\mathrm{MKS,B}} \ni \mathbf{q} = \begin{bmatrix} x_c \\ y_c \\ x_f \\ y_f \end{bmatrix} \longmapsto \begin{bmatrix} \Lambda_B(\mathbf{q}) \\ x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x_c + x_f \\ y_c + y_f \\ x_c \\ y_c \end{bmatrix}$$

[1] Beziehungsweise Fräsen, Wasserstrahlschneiden oder eine andere Bearbeitungsart.

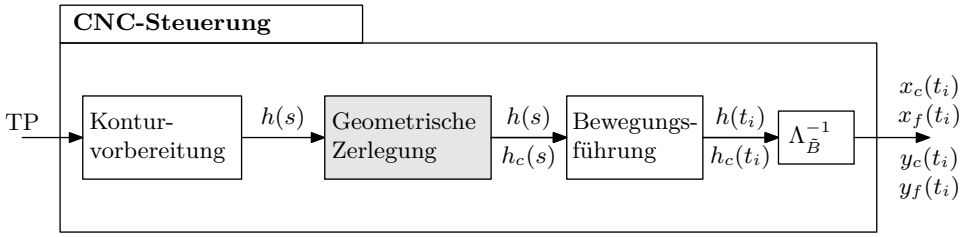

Abbildung 3.1.: Datenfluss bei geometrischer Aufteilung.

ist eindeutig invertierbar mit

$$\Lambda_{\widetilde{B}}^{-1} : \mathbb{R}^4 \ni \begin{bmatrix} x \\ y \\ x_c \\ y_c \end{bmatrix} \longmapsto \begin{bmatrix} x_c \\ y_c \\ x - x_c \\ y - y_c \end{bmatrix} \in \mathbb{R}^4$$

und soll in diesem Ansatz als kinematische Transformation dienen. Mit ihr berechnet die Bewegungsführung eine eindeutige Bahntrajektorie $s_b(t)$. Allerdings unterscheidet sich die Eingabe in die Bewegungsführung vom herkömmlichen, oben beschriebenen Vorgehen bei Kinematik A. Dort war die Eingabe die Kontur als ebene Kurve $h(s)$. Hier benötigt die Bewegungsführung eine vierdimensionale Kurve, das heißt $h(s) = [x(s), y(s)]^T$ wird durch eine weitere Kurve $h_c(s) = [x_c(s), y_c(s)]^T$ ergänzt, die ebenfalls nach der Bogenlänge s der Kontur parametrisiert ist[2]. Erfüllt $h_c(s)$ für jedes s die Bedingungen

$$|x(s) - x_c(s)| \leq \widehat{S}_x \qquad \text{und} \qquad |y(s) - y_c(s)| \leq \widehat{S}_y$$

so bleiben die Verläufe der Feinachsen

$$x_f(s) = x(s) - x_c(s),$$
$$y_f(s) = y(s) - y_c(s)$$

innerhalb ihrer Verfahrbereiche.

Das Verfahren aus Abschnitt 2.2.3 zur Berechnung der Bewegung von Kinematik A kann nach einer Modifikation auch für diesen Ansatz genutzt werden. Phase eins nimmt zur Berechnung der Grenzkurven $v_b^{\downarrow}(s)$ und $a_b^{\downarrow}(s)$ für die Bedingungen (2.12), (2.13) und (2.16) nicht die Ableitungen von $x(s)$ und $y(s)$, sondern die von $x_c(s)$, $y_c(s)$, $x_f(s)$ und $y_f(s)$, sowie die Achsrestriktionen (2.4). Phase zwei berechnet unverändert die Bahntrajektorie $s_b(t)$. Damit entstehen die Achstrajektorien durch

$$x_c(t) = x_c(s_b(t)), \qquad x_f(t) = x_f(s_b(t)),$$

[2] $h_c(s)$ ist nicht die Parametrisierung nach ihrer eigenen Bogenlänge!

$$y_c(t) = y_c(s_b(t)), \qquad y_f(t) = y_f(s_b(t)),$$

und sie erfüllen die Achsrestriktionen.

Da dieser Ansatz gewissermaßen eine Zerlegung der Kontur $h(s)$ in $h_c(s)$ und $h_f(s) = h(s) - h_c(s)$ durchführt, die vor der Bewegungsführung stattfindet, das heißt, rein auf dem geometrischen Verlauf der Kontur, nennen wir diesen Ansatz *geometrische Zerlegung*.

Für die Berechnung der Grobbahn $h_c(s)$ bieten sich Spline–Approximationstechniken an, [5] fasst zwei konkrete Verfahren zusammen, [4] erläutert sie genauer. Das erste Verfahren nutzt den Schoenberg–Approximanten, das zweite löst ein quadratisches Programm mit einem Glättefunktional für $h_c(s)$ als Zielfunktion und mit linearen Nebenbedingungen durch die begrenzten Verfahrbereiche der Feinachsen.

Bei diesem Ansatz ist es natürlich auch so, dass der Werkzeugkopf an einer Ecke der Kontur zum Stillstand kommt. Ist s^* der Bahnparameter einer Ecke, dann gilt $v_b(s^*) = 0$. Wegen der Gleichung für die Achsgeschwindigkeit,

$$v_{xc}(s) = x_c'(s) \cdot v_b(s), \tag{3.1}$$

stehen die Grobachsen (und damit auch die Feinachsen) an einer Ecke, wenn $|x_c'(s^*)|$ beziehungsweise $|y_c'(s^*)|$ endlich ist. Das trifft für Splines als Grobbahnen zu. Generell ist das Anhalten aller redundanten Achsen an einer Ecke nicht notwendig, wie die weiteren Ansätze zeigen. Auch an Stellen der Konur mit starker Krümmung und damit zusammenhängender geringer Bahngeschwindigkeit führt (3.1) zu niedrigen Achsgeschwindigkeiten.

Dieser Effekt hat zwei entscheidende Auswirkungen. Zum einen ist die Bearbeitungszeit langsam. Da die Grobachsen an Ecken anhalten, wegen des begrenzten Verfahrbereichs aber eine gewisse Distanz zurücklegen müssen, dominiert die träge Dynamik der Hauptachsen die Gesamtbewegung. Die Feinachsen können ihr großes Beschleunigungsvermögen nicht ausschöpfen.

Die zweite mögliche Auswirkung sieht man an der Qualität des Werkstücks. Das vor und nach Ecken oft maximale Beschleunigen mit einer Grobachse regt die Mechanik der Maschine und damit auch den Werkzeugkopf zu Schwingungen an. So kann ein nicht tolerierbarer Konturfehler entstehen. Abbildung 3.2 zeigt ein Beispiel für einen solchen Fall aus dem Laserschneiden dünner Bleche. Nach der Spitze mit dem kleinen Kreisbogen rechts oben soll eigentlich ein gerader Schnitt folgen. Am Werkstück ist das allerdings eine gewellte Linie.

Zuletzt soll ein Beispiel den gerade beschriebenen Effekt verdeutlichen. Abbildung 3.3 zeigt ausgewählte Achsverläufe. Sie stammen aus einem Testlauf mit der zweiten Konfiguration (MD2) aus Tabelle 2.1 auf Seite 19. Der Verlauf der Grobachse x_c über dem Bahnparameter s ist eine Gerade und damit gewissermaßen glatt. Im zeitlichen Verlauf dieser Achse sieht man allerdings das ständige Abbremsen beziehungsweise Anhalten. Die drei „echten“ Ecken werden bei den Zeiten 0.06, 0.35 und 0.43 Sekunden durchlaufen. Dort ist die Geschwindigkeit aller Achsen null. Bei 0.31 Sekunden liegt eine abgerundete Ecke, das heißt, dort ist ein kleiner Kreisbogen und die Bahngeschwindigkeit sowie die

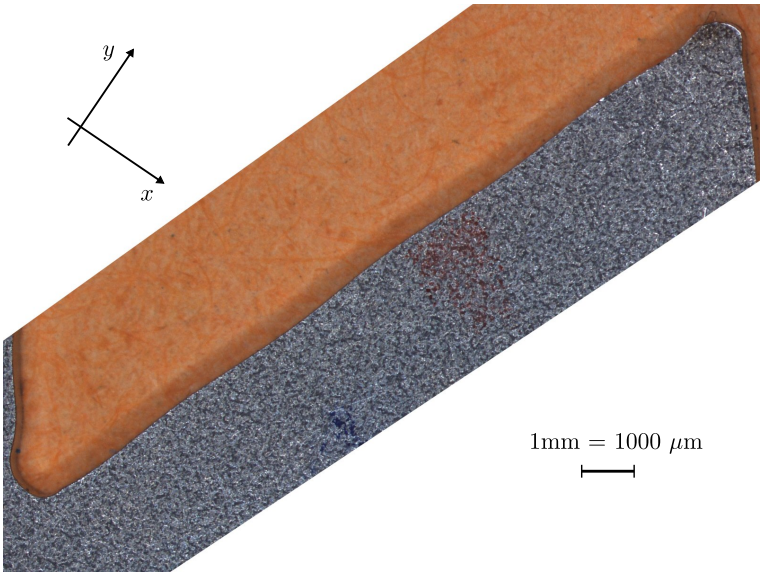

Abbildung 3.2.: Werkstück, erstellt mit dem Verfahren der geometrischen Zerlegung.

x_c–Geschwindigkeit sind etwas größer als null. Die Beschleunigungsverläufe der x_c– und x_f–Achse zeigen einerseits, dass x_c in den Beschleunigungsphasen um eine Ecke herum ihre maximale Beschleunigung nutzt, was zu Schwingungsanregungen führen kann. Andererseits beschleunigt die Feinachse bei weitem nicht mit ihrem Maximalwert von 60 m/sec^2. Das heißt, die Gesamtbewegung könnte schneller sein.

In der Kontur sind zwei Kreisbögen enthalten. Den großen sieht man deutlich, der andere ist die abgerundete Ecke bei etwa $[x, y]^T = [6.2, -2]^T$. Der erste Kreisbogen hat eine so schwache Krümmung, dass die Geschwindigkeit auf der Bahn nicht reduziert werden muss. Die Glattheit des Verlaufs von $x_c(s)$ überträgt sich wegen (3.1) auf $x_c(t)$. Beim zweiten Kreisbogen (bei etwa $t = 0.31$ Sekunden) ist die Krümmung sehr stark und die Grobachse muss abbremsen, allerdings nicht ganz bis zum Stillstand. Diese Beobachtung zeigt, dass der Ansatz der geometrischen Zerlegung bei Konturen ohne Ecken und mit geringen Krümmungen nicht zu Qualitätsproblemen führt.

3.3. Filterung im Zeitbereich

Dieser Ansatz nutzt ein Filter hinter der Bewegungsführung, um eine Referenzbewegung des Kopfes in die Anteile für die Grob- und Feinachsen zu zerlegen. Der modifizierte Datenfluss ist in Abbildung 3.4 skizziert. Darin wurde auf den Block der kinematischen Transformation verzichtet, da Λ_A der Identität entspricht.

Bis einschließlich zur Bewegungsführung weiß die Steuerung nicht, dass es sich um eine Maschine mit redundanten Achsen handelt. Die Ausgabe der Bewegungsführung sind die (diskreten) Trajektorien $x(t_i)$ und $y(t_i)$. Danach findet für jede davon separat eine

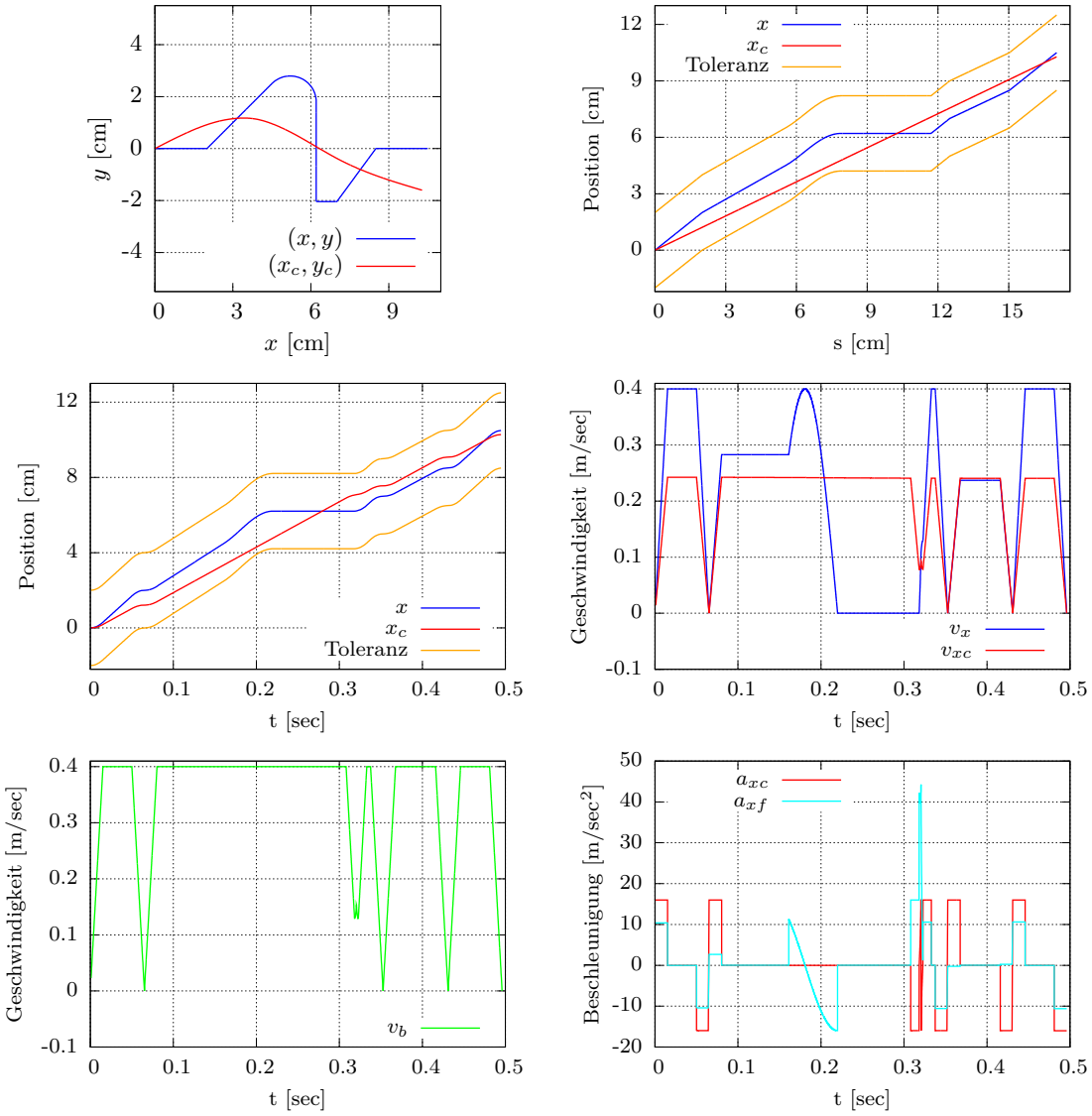

Abbildung 3.3.: Achsverläufe zum Beispiel der geometrischen Zerlegung in Abschnitt 3.2.

Filterung, sinnvollerweise mit einem Tiefpass, statt und die Ausgabe ist im Fall der x–Komponente die Bewegung der Hauptachse $x_c(t_i)$. Für die Feinachse $x_f(t_i)$ nutzt der Ansatz den Eingang $x(t_i)$, führt optional eine zeitliche Verzögerung durch, und zieht davon die Bewegung der Grobachse ab.

Die Idee der Filterung findet man in einigen Patenten, zum Beispiel [11], [10] und [9], sowie in Veröffentlichungen, etwa [27] oder [15]. Inhalt von [27] und dem Patent [17] ist ein Vorgehen mit einem FIR–Filter und einer Verzögerung um dessen Gruppenlaufzeit. Die Verzögerung ist sinnvoll, denn ohne sie benötigt die Feinachse einen größeren Verfahrbereich. Bei diesem Vorgehen verlängert sich die Bearbeitungszeit der Gesamtbewegung (von Anhalten zu Anhalten) um das Zweifache der Gruppenlaufzeit. Das führt besonders dann zu einem deutlichen Anstieg der Gesamtbearbeitungszeit, wenn die Anwendung ein Stehen aller Achsen beim Wechsel zwischen Positionieren und Schneiden erfordert und in der Kontur viele derartige Wechsel vorkommen.

Natürlich müssen die mit diesem Ansatz gewonnenen Trajektorien $x_c(t_i)$ und $x_f(t_i)$ den Achsrestriktionen genügen. Es bieten sich zwei Stellen an, um die Ausgabe diesbezüglich zu beeinflussen. Zum einen ergeben verschiedene Achsrestriktionen in der Bewegungsführung unterschiedliche Referenztrajektorien, zum anderen können verschiedene Filter und Filterparametrisierungen gewählt werden. In den oben zitierten Quellen finden sich weder theoretische Überlegungen, noch konkrete Empfehlungen zur Konfiguration und Optimierung von Bewegungsführung oder Filtern sowie deren Zusammenspiel. In der Praxis kann man diese Aufgabe durch Simulation und „Ausprobieren“ erledigen.

Auch diesen Abschnitt soll ein Beispiel abschließen. Abbildung 3.5 zeigt einige der Achsverläufe. Wie im vorherigen Abschnitt sollen die Ergebnisse für eine Maschine mit Konfiguration zwei aus Tabelle 2.1 gültig sein. Die Einstellungen für Bewegungsführung und Filter wurden durch Probieren herausgefunden. Sie sind weder optimal, noch ist ein Funktionieren bei beliebigen Konturen sicher. Die Beschleunigungsgrenze ist in dem Beispiel 55 m/sec^2 für x- und y-Achse, die Grenzen der Geschwindigkeit je 2.5 m/sec. Der Vorschub $\hat{v}_b$ bleibt unverändert bei 0.4 m/sec. Das FIR–Filter besteht exemplarisch aus 61 Koeffizienten, die mit der Matlab–Funktion[3] `fir1.m` berechnet werden. Darin ist ein Tiefpassfilter mit Hamming–Fenster und normalisierter Grenzfrequenz implementiert. Die Grenzfrequenz beträgt hier 1 Hz. Die Abtastzeit der diskreten Trajektorien ist $t_{\text{ipo}} = 0.001$ Sekunden.

Die Achsverläufe zeigen deutlich die oben schon erwähnte, zusätzliche Bearbeitungszeit. Am Anfang und am Ende der Bewegung bleibt die Position des Kopfes jeweils 0.03 Sekunden unverändert, die Bewegungen der Einzelachsen kompensieren sich. Im Vergleich zum vorherigen Ansatz der geometrischen Zerlegung stehen nicht alle Achsen an den Ecken der Kontur, obwohl der Werkzeugkopf anhält. Die Beschleunigung verläuft zudem anschaulich glatter. Die Feinachsen nutzen allerdings nur einen kleinen Teil ihres Verfahrbereichs.

[3] http://www.mathworks.com/products/matlab/, letzter Aufruf am 8. Juli 2010.

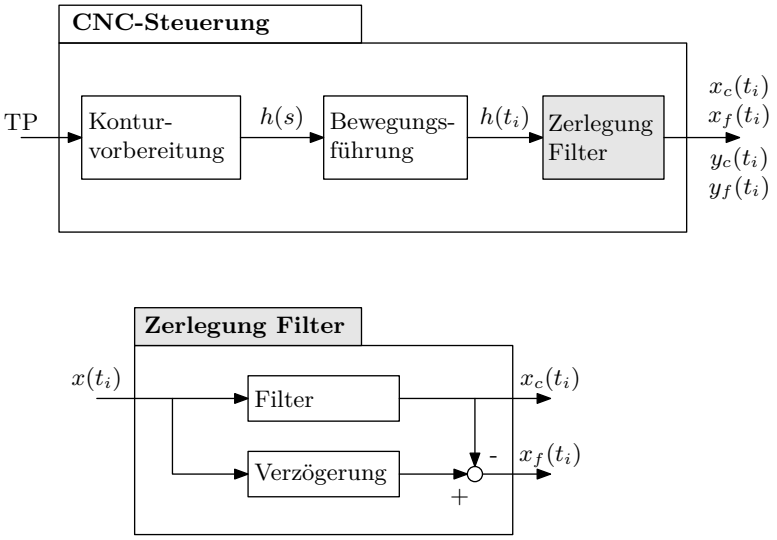

Abbildung 3.4.: Modifizierter Datenfluss für die Filterung im Zeitbereich.

3.4. Regelungskonzept mit Schleppfehlerkompensation

Bei den bisherigen Ansätzen fand die Behandlung der Redundanz ausschließlich in der Steuerung oder davor statt. Die Achsantriebe bekommen ihre Sollwerte und regeln die Bewegung völlig unabhängig voneinander. Das ändert sich in diesem Ansatz (Abbildung 3.6). Die Steuerung arbeitet für eine Kinematik vom Typ A und weiß nichts von den redundanten Achsen. Quellen dieses Vorgehens sind zum Beispiel [33] oder das Patent [16].

Wie beim Ansatz mit der Filterung im vorherigen Abschnitt erstellt die Bewegungsführung eine Referenztrajektorie $x(t_i)$ beziehungsweise $y(t_i)$. Sie verlässt die Steuerung allerdings ungefiltert und ist jeweils die Eingabe des Antriebs der x_c– beziehungsweise der y_c–Achse. Jeder der beiden Achsantriebe bestimmt im Betrieb seinen Schleppfehler. Das ist die Differenz aus dem Wert der Referenz und der gemessenen Istposition der Achse. Dieser Schleppfehler wird dann zum Eingang des Antriebs der entsprechenden Feinachse.

In diesem Ansatz muss die Eingabe des Antriebs der Grobachse keine im Sinne der vorherigen Abschnitte gültige Trajektorie sein. Andernfalls wäre die Gesamtbewegung nicht schneller als mit den Grobachsen alleine. Hier können die Achsrestriktionen bei der Erstellung der Referenz in der Steuerung größer sein als die der Hauptachse. Ihr Antrieb kann dann dem Eingang nicht folgen und es entsteht ein verhältnismäßig großer Schleppfehler[4], $x(t_i) - x_{c,\mathrm{ist}}(t_i)$. Er muss als Eingang der Feinachse für ihre Restriktionen gültig sein, damit die Feinachse folgen kann und der Werkzeugkopf die Kontur abfährt.

Das Funktionieren dieses Ansatzes hängt im Wesentlichen von der Wahl der Restriktionen in der Steuerung beziehungsweise der Bewegungsführung ab. In den oben erwähnten

[4] Bei jeder Achsregelung in einem Antrieb entsteht ein Schleppfehler. Allerdings ist dieser bei einer gültigen Eingangstrajektorie klein und tolerierbar.

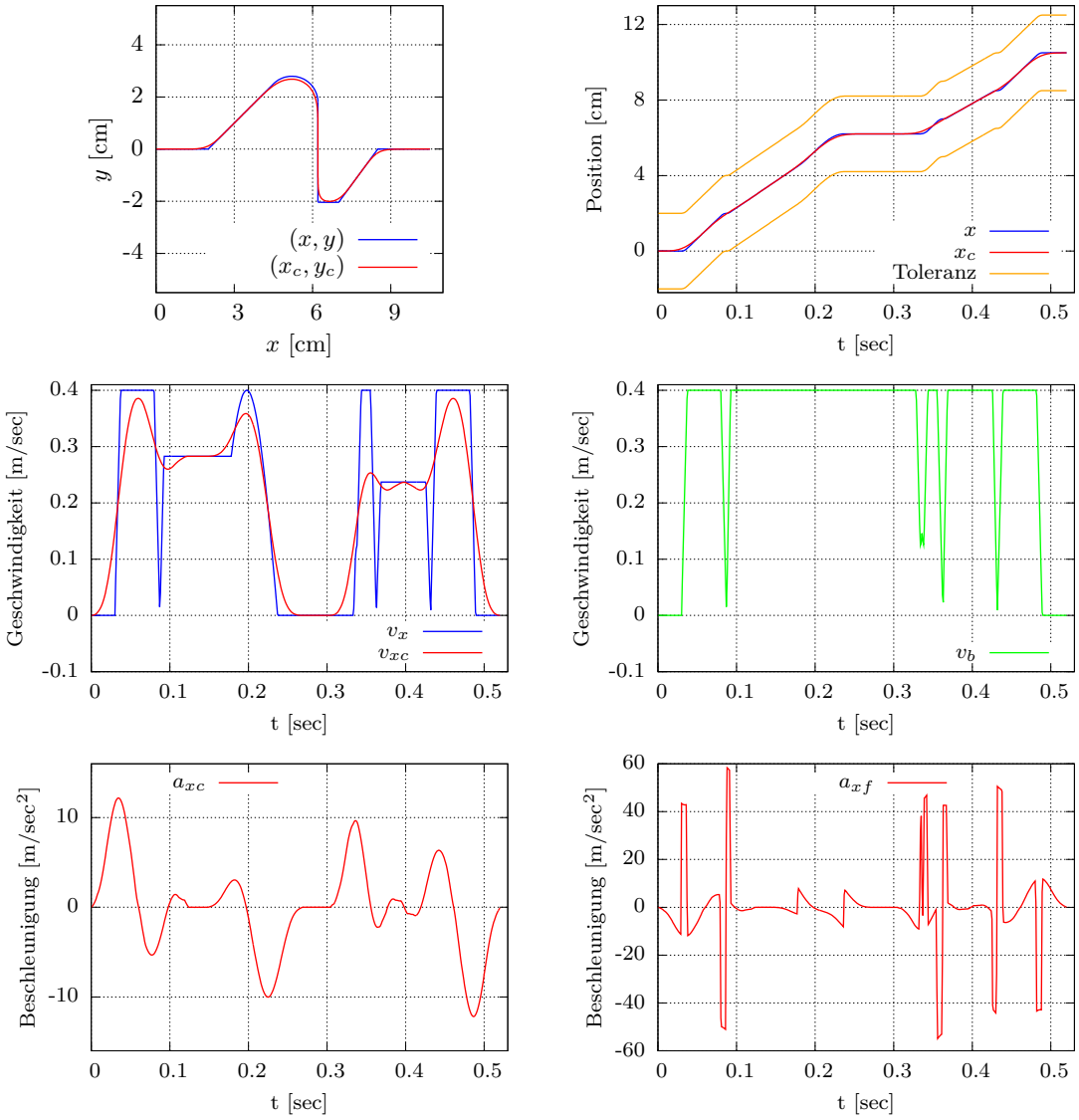

Abbildung 3.5.: Achsverläufe des Beispiels zur Filterung in Abschnitt 3.3.

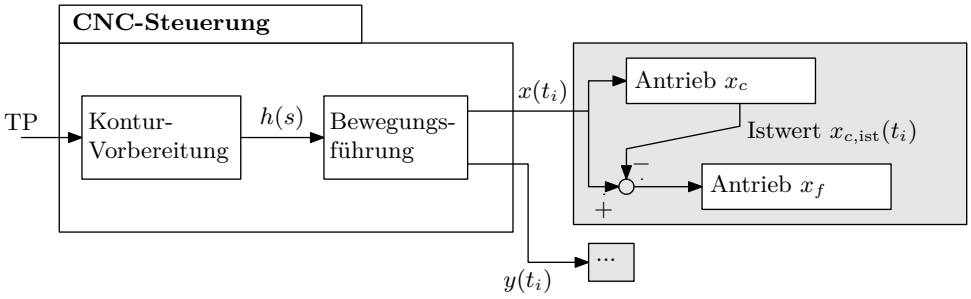

Abbildung 3.6.: Schema Schleppfehler-Ansatz.

Quellen wird dafür kein allgemeines Vorgehen vorgestellt. Letztendlich kann man sich wieder durch Simulation und „Ausprobieren“ helfen. Da dies auch ein komplexes Simulationsmodell der Antriebe erfordert, verzichten wir auf ein ähnliches Beispiel wie in den beiden vorherigen Abschnitten. Einen Effekt dieses Ansatzes lässt sich schon mit einfachsten Modellen für die Achsen feststellen. Das betrachtet der folgende Abschnitt.

3.4.1. Konturverzerrung durch Überschwingen der Summenachse

Die Erläuterung des angedeuteten Effektes benötigt einige Grundlagen über Regelsysteme, die dieser Abschnitt zu Anfang angibt. Die Darstellung beschränkt sich dabei auf das Wesentliche und außerdem auf den Fall kontinuierlicher Systeme. Für Details sei auf Literatur zur Regelungstechnik verwiesen, zum Beispiel [38].

Bei der Betrachtung von Regelsystemen ist die Laplace–Transformation ein wichtiges Hilfsmittel. Zu einer Funktion

$$f : \mathbb{R}^+ \ni t \longmapsto f(t) \in \mathbb{R} \tag{3.2}$$

ist die Laplace–Transformierte von $f(t)$ die komplexe Funktion[5]

$$F(\sigma) := \mathcal{L}[f](\sigma) := \int_0^\infty f(t)\, e^{-\sigma t} \mathrm{d}t, \qquad \sigma \in \mathbb{C}. \tag{3.3}$$

Die Verwendung von Zeitfunktionen wie in (3.2) nennt man Zeitbereichsdarstellung, mit Funktionen wie in (3.3) spricht man von der Darstellung im Frequenzbereich.

[5] In der Literatur zur Regelungstechnik wird für die komplexe Variable überwiegend s verwendet. Da diese Variable schon von dem Parameter der Bogenlängenparametrisierung belegt ist, verwenden wir die Variable $\sigma \in \mathbb{C}$.

Ein einfaches System besteht aus einer Eingangsgröße $f_e(t)$ und einer Ausgangsgröße $f_a(t)$ mit den Laplace–Transformierten $F_e(\sigma)$ und $F_a(\sigma)$. Darin findet eine Modifikation der Eingangsgröße statt, dargestellt durch eine Faltung mit einer weiteren Funktion $g(t)$:

$$f_a(t) = f_e(t) * g(t) := \int_0^t g(t) f_e(t-\tau) \mathrm{d}\tau, \qquad t \in \mathbb{R}^+.$$

Im Frequenzbereich ist die Darstellung

$$F_a(\sigma) = \mathcal{L}[f_e * g](\sigma) = F_e(\sigma) \ \cdot \ G(\sigma)$$

und $G(\sigma)$ als Laplace–Transformierte von $g(t)$ heißt *Übertragungsfunktion* des Systems.

Eine einfache Modellierung des Regelsystems einer Maschinenachse geschieht durch ein sogenanntes Verzögerungsglied erster Ordnung oder kurz PT_1–Glied. Die Übertragungsfunktion ist im Fall der x_c–Achse

$$G_{xc}(\sigma) = \frac{1}{T_{xc}\sigma + 1}. \tag{3.4}$$

Die Konstante $T_{xc} \in \mathbb{R}^+$ repräsentiert das dynamische Verhalten des Systems. Zu dessen Untersuchung nutzt man als Eingangsgröße den Einheitssprung

$$f_e(t) := 1, \qquad t \in \mathbb{R}^+,$$

mit

$$F_e(\sigma) = \frac{1}{\sigma}$$

als Darstellung im Frequenzbereich. Damit ergibt sich für die Ausgangsgröße

$$F_a(\sigma) = \frac{1}{\sigma} \ \cdot \ \frac{1}{T_{xc}\sigma + 1}.$$

Deren Zeitbereichsdarstellung ist

$$f_a(t) = 1 - e^{-t/T_{xc}}, \qquad t \in \mathbb{R}^+.$$

Für die Feinachse gibt es eine Übertragungsfunktion wie in (3.4),

$$G_{xf}(\sigma) = \frac{1}{T_{xf}\sigma + 1},$$

nur die Konstanten T_{xf} und T_{xc} unterscheiden sich. Je kleiner die Konstante bei einer Übertragungsfunktion der Form (3.4) ist, desto schneller nähern sich die Istwerte den

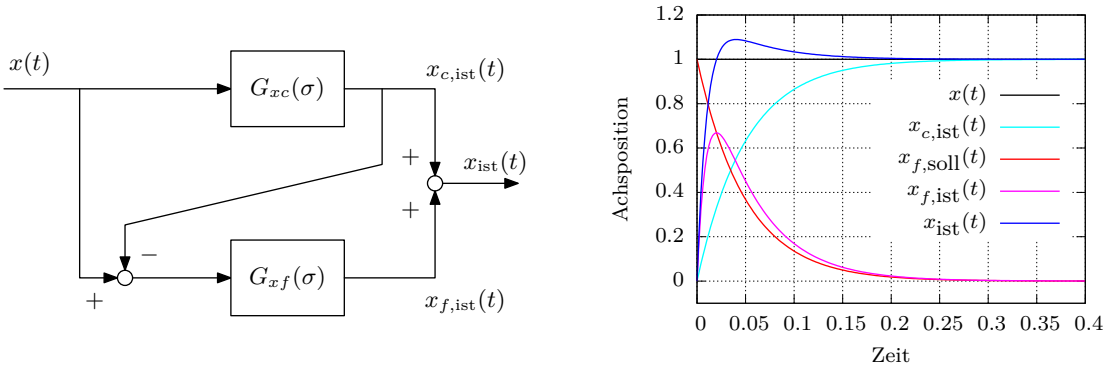

Abbildung 3.7.: Blockschaubild des Regelsystems (links) und exemplarisches Verhalten bei Eingabe des Einheitssprung (rechts).

Sollwerten. Da wir in unserer Anwendung eine träge Grobachse und eine hochdynamische Feinachse haben, können wir

$$T_{xc} > T_{xf} \tag{3.5}$$

annehmen.

Mit diesen Modellen für die Regelsysteme der Achsen ergibt sich im Ansatz der Schleppfehlerkompensation (Abbildung 3.7, links) mit $x(t)$ als Eingangsgröße und $x_{\text{ist}}(t)$ als Ausgangsgröße die Übertragungsfunktion des Gesamtsystems:

$$\begin{aligned} G_1(\sigma) &:= \left[\left(1 - G_{xc}(\sigma)\right) G_{xf}(\sigma)\right] + G_{xc}(\sigma) \\ &= \left[\left(1 - \frac{1}{T_{xc}\sigma + 1}\right) \frac{1}{T_{xf}\sigma + 1}\right] + \frac{1}{T_{xc}\sigma + 1}. \end{aligned}$$

Das zeitliche Verhalten bei Eingabe des Einheitssprungs ist

$$x_{\text{ist}}(t) = 1 + \frac{1}{T_{xc} - T_{xf}} \left[-T_{xf} e^{-t/T_{xc}} + T_{xc} e^{-t/T_{xf}}\right]$$

und der Verlauf (Abbildung 3.7, rechts) zeigt ein Überschwingen (dunkelblaue Linie) bezüglich der Eingabe. Das ist ein typischer Effekt, wenn (3.5) gilt. Er führt in der Praxis zu Konturverzerrungen, bei einem Kreis und Kinematik B beispielsweise zu einer Vergrößerung des Radius.

3.4.2. Hochdynamische Regelung durch vorgelagerter Sollwertfilterung

Im Rahmen dieser Doktorarbeit wurde ein Regelungskonzept entwickelt, mit dem der im vorherigen Abschnitt beschriebene Effekt des Überschwingens nicht auftritt. Eine zusätzliche Anwendung besteht darin, schon aufgeteilte Sollwerte hochdynamisch zu regeln.

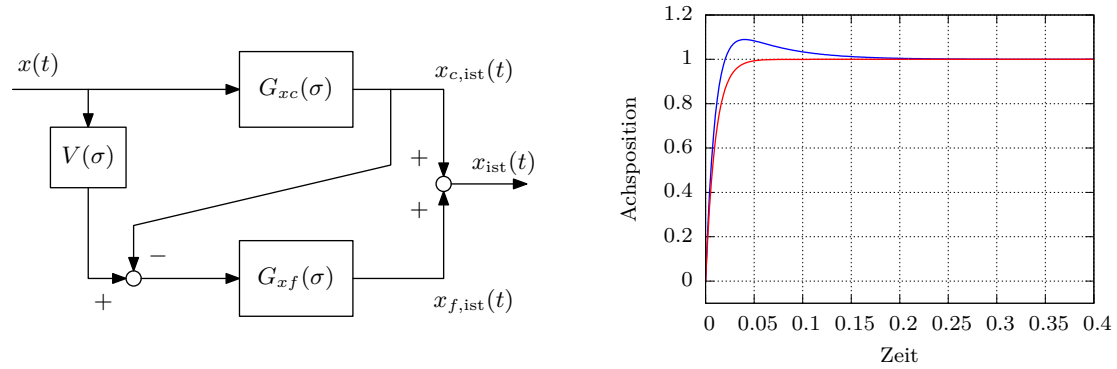

Abbildung 3.8.: Blockschaubild des neuen Regelungskonzeptes mit vorgelagerter Sollwertfilterung (links) und Vergleich (rechts) der Istpositionen mit herkömmlicher (blau) und modifizierter (rot) Schleppfehlerkompensation.

Das Konzept erweitert das Schema aus Abbildung 3.7. Der linke Teil von Abbildung 3.8 zeigt das. Ein zusätzlicher Block mit der Übertragungsfunktion

$$V(\sigma) := 1 + G_{xc}(\sigma) - \frac{G_{xc}(\sigma)}{G_{xf}(\sigma)}$$

modifiziert das Eingangssignal, bevor davon der Istwert der Grobachse abgezogen wird. Damit ergibt sich für das Gesamtsystem die Übertragungsfunktion

$$G_2(\sigma) := (V(\sigma) - G_{xc}(\sigma))\, G_{xf}(\sigma) + G_{xc}(\sigma) = G_{xf}(\sigma).$$

Das Gesamtsystem verhält sich also genauso wie die Feinachse alleine und kann deren hohe Regeldynamik ausnutzen. Mit den einfachen Näherungen durch PT_1–Glieder sieht man das sehr deutlich.

Die Übertragungsfunktionen der realen Regelungssysteme der Achsen sind natürlich sehr komplex und nur näherungsweise durch Formeln zu beschreiben. Mit diesen Näherungen kann der neue Block parametrisiert werden. Ergeben die rationalen Funktionen für die Frequenzbereichsdarstellung von $G_{xc}(\sigma)$ und $G_{xf}(\sigma)$ ein $V(\sigma)$, dessen Zählergrad kleiner oder gleich dem Nennergrad ist, so ist eine Realisierung des neuen Blocks in der Anwendung möglich.

Dieses Konzept verbessert auch die Regelung der redundanten Achsen, wenn die Sollwerte schon in $x_c(t)$ und $x_f(t)$ aufgeteilt sind. Abbildung 3.9 zeigt das entsprechende Blockdiagramm. Arbeiten die Regelungen von Grob- und Feinachse getrennt voneinander, weiß man in der Regelungstechnik, dass die Regelparameter in dem jeweiligen System in bestimmter Art und Weise aufeinander abgestimmt werden müssen, damit es zu keinen Konturverzerrungen kommt. Dabei passt sich die Feinachse allerdings in ihrer Regeldyna-

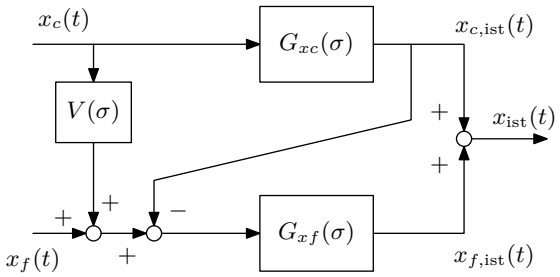

Abbildung 3.9.: Schema Schleppfehler-Ansatz.

mik an die Grobachse an. Das Gesamtsystem nutzt also die hohe Dynamik der Feinachse nicht aus.

Mit dem neuen Konzept tritt diese Problematik nicht mehr auf. Die Regelsysteme beider Achsen können unabhängig voneinander parametrisiert werden. Die Istwerte der Summenachse verhalten sich dynamisch, wie wenn sie aus dem Regelsystem der Feinachse alleine kommen. Die Anwendung dieses Ansatzes ist unabhängig von der verwendeten Sollwertaufteilung möglich. Er kann mit den Verfahren aus den Abschnitten 3.2 und 3.3, sowie mit den Ansätzen aus den folgenden Kapiteln 4 und 5 kombiniert werden.

4. Präparative Bewegungsaufteilung im Zeitbereich

Jedes der im vorherigen Kapitel vorgestellten Verfahren zur Steuerung redundanter Kinematiken hat unerwünschte Eigenschaften. Besonders bei Maschinendaten, ähnlich zu denen aus Tabelle 2.1, treten sie auf. Das Nachsetzen ist in vielen Anwendungsfällen kaum produktiver als die Hauptachsen alleine. Die geometrische Zerlegung nutzt die hohe Dynamik der Feinachsen selten aus und kann außerdem zu schlechter Qualität am Werkstück führen. Der Ansatz mit der Filterung hinter der Bewegungsführung ist schwer zu parametrisieren und auch hier ist ein Produktivitätsverlust durch die Verzögerung des Filters möglich. Bei der Schleppfehlerkompensation ist ebenfalls die Parametrisierung der Bewegungsführung und der Antriebe schwierig.

Dieses Kapitel behandelt einen alternativen, neuartigen Ansatz zur Steuerung einer Maschine mit redundanten Achsen, der die erwähnten, unerwünschten Eigenschaften nicht besitzt. Der folgende Abschnitt 4.1 stellt die Idee vor und beschreibt die notwendigen Modifikationen im Steuerungsschema. Der darauffolgende Abschnitt erläutert Methoden zur Splineapproximation, die in dem Ansatz für die Zerlegung benutzt werden. Der letzte Abschnitt dieses Kapitels beschäftigt sich mit der Parametrisierung des Vorgehens und zeigt Ergebnisse aus der Anwendung.

4.1. Idee und Steuerungsschema

Der neue Ansatz erweitert die Bewegungsführung aus Abbildung 2.6 auf Seite 22. Abbildung 4.1 stellt die neue Struktur des Datenflusses dar. Wir gehen davon aus, dass mit den diskreten Positionen, wie etwa $x_c(t_i)$, auch die Werte für Geschwindigkeit und Beschleunigung von den Blöcken weitergegeben werden. Um Platz zu sparen, fehlen die Bezeichner in der Abbildung. Das Vorgehen arbeitet iterativ im Vorlauf. Wir nehmen für die Beschreibung dieser Iterationen zuerst an, dass die redundante Bewegung für die gesamte Kontur auf einmal berechnet wird. Das entspricht natürlich nicht einem tatsächlich möglichen Ablauf in der Steuerung, denn dort wird die Kontur abschnittsweise abgearbeitet. Diesen Aspekt behandeln wir weiter unten in Abschnitt 4.3.4.

Die Bewegungsplanung berechnet hier für die Kontur nicht nur die Bahntrajektorie, sondern auch die diskreten Achswerte $x(t_i)$ und $y(t_i)$, $i \in \{1, \ldots, n_t\}$. Diese wandern alle zusammen zur Zerlegung. Dort teilt ein Verfahren die Werte für jede Achse separat in Anteile für die entsprechende Grob- und Feinachse auf. Diese Zerlegung im Zeitbereich arbeitet im Gegensatz zur Filterung aus Abschnitt 3.3 mit Splineapproximation zur

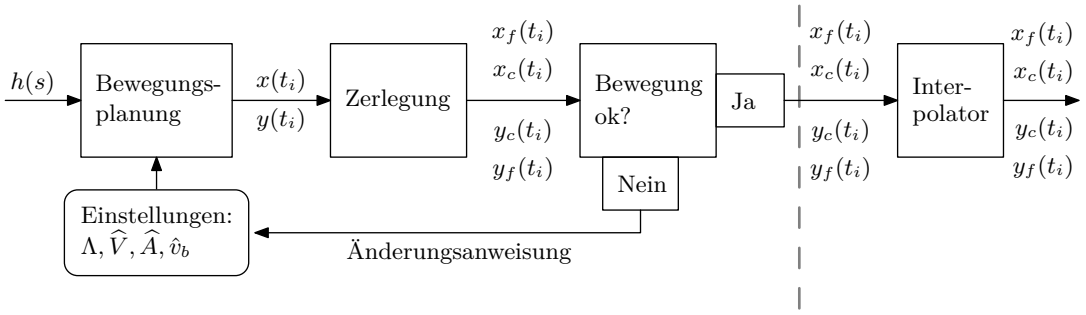

Abbildung 4.1.: Struktur der präparativen Zerlegung im Zeitbereich.

Berechnung der Bewegung der Grobachse. Die Details der verwendeten Approximationsmethoden folgen in Abschnitt 4.2. Die Feinachse ergibt sich dann natürlich direkt durch Differenzbildung. Die zerlegten Achswerte wandern dann weiter an einen Block, der an den diskreten Stellen t_i ihre Gültigkeit überprüft. Nur wenn alle Achsen ihre Grenzen für Verfahrbereich, Geschwindigkeit und Beschleunigung einhalten, verlassen sie den Vorlauf und wandern durch den Interpolator aus der Bewegungsführung und der Steuerung heraus zu den Achsantrieben. Die Positions-, Geschwindigkeits- und Beschleunigungswerte der x_c–Achse ergeben sich durch Auswertung des Splineapproximanten und dessen Ableitungen an den Stellen t_i. Damit und mit den entsprechenden Werten der Summenachse x aus der Bewegungsplanung berechnen sich die Werte der Feinachse durch Differenzbildung.

Sind die Achswerte allerdings nicht gültig, beginnt die Berechnung noch einmal von vorne. Damit aber nicht wieder dasselbe Ergebnis herauskommt, müssen die Einstellungen in der Bewegungsplanung verändert werden. Diese Iterationen finden so lange statt, bis die Achswerte für den Interpolator gültig sind.

Da die Steuerung abschnittsweise arbeitet (vergleiche 2.2.2), ist der Bewegungsplanung die Kontur nur bis zu einem bestimmten Punkt (beziehungsweise Satz) bekannt. Für diesen Abschnitt bestimmt das Verfahren eine Bewegung, sodass alle Achsen am Ende stehen. Der nächste Abschnitt wird dann allerdings nicht an das Ende des vorherigen angehängt. Das würde dazu führen, dass der Kopf und alle Einzelachsen an Stellen anhalten, wo dies gar nicht notwendig ist. Deswegen wird die Bewegung für eine feste Anzahl von Sätzen am Ende jedes Abschnitts verworfen. Das iterative Vorgehen nimmt dieses Stück der Kontur im nächsten Abschnitt noch einmal hinzu. Dort müssen die Achsbewegungen natürlich am Anfangszeitpunkt Randbedingungen erfüllen, damit sie stetig in Position und Geschwindigkeit (und im Fall der ruckbegrenzten Bewegungsfühung auch in der Beschleunigung) sind. Anschaulich wandert ein Fenster über die Kontur, das sich aber um weniger als die volle Fensterbreite verschiebt. Abschnitt 4.3.4 beschreibt dieses Vorgehen noch einmal detaillierter.

Die Motivation für diesen alternativen Ansatz ist naheliegend. Die Zerlegung im Zeitbereich verhindert das Abbremsen an Ecken, wie es mit der geometrischen Zerlegung auftritt. Die Aufteilung mithilfe von Splineapproximation braucht keine zusätzliche Verzögerung

vor und nach der Bewegung, wie das bei der Filterung der Fall ist. Die Realisierung im Vorlauf ermöglicht das iterative Vorgehen zusammen mit der Bewegungsplanung und damit das Finden gültiger Zerlegungen.

4.2. Approximation diskreter Daten durch Splinefunktionen

Dieser Abschnitt stellt die Verfahren für die Zerlegung der Referenztrajektorien vor. Er beginnt mit einer Festlegung der benutzten Notation und den notwendigen Ergebnissen aus der Splinetheorie. Darauf folgen die verschiedenen Approximationsverfahren für diskrete Daten, die letztendlich für die Zerlegung verwendet werden können.

4.2.1. Grundlegende Notationen und Eigenschaften von Splinefunktionen

Dieser Abschnitt listet lediglich die notwendigen Ergebnisse aus der Splinetheorie auf. Eine ausführlichere Darstellung, auch mit Beweisen, bietet [6].

Definition 4.1. *Seien $n_b \in \mathbb{N}$, $n_b > 1$, $m \in \mathbb{N}$, $m \geq 1$ und $I = [b_1, b_{n_b}]$ ein kompaktes Intervall und sei $B = (b_1, \ldots, b_{n_b}) \in \mathbb{R}^{1\times n_b}$ eine streng monton steigende Folge. Die Menge*

$$\mathbb{S}_m(B) = \{\ f \in C(I) \mid f|_{(b_i,b_{i+1})} \in \mathbb{P}_m((b_i, b_{i+1})), i = 1, \ldots, n_b - 1\ \}$$

ist der Splineraum *vom Grad m zur* Bruchstellenfolge *B. Jedes Element aus $\mathbb{S}_m(B)$ ist eine stetige* Splinefunktion *vom Grad m.*

Der Splineraum $\mathbb{S}_m(B)$ ist ein Vektorraum der Dimension $m(n_b - 1) + 1$.

Definition 4.2. *Seien $n, m \in \mathbb{N}_0$ mit $n > m$ und seien $\zeta_a, \zeta_b \in \mathbb{R}$, mit $\zeta_a < \zeta_b$. Die Folge*

$$\begin{aligned} T = T_{m,n} = (\zeta_1, \zeta_2, \ldots, \zeta_{n+m}, \zeta_{n+m+1}) \quad & \textit{mit} \\ \zeta_1 = \zeta_a, \qquad \zeta_{n+m+1} &= \zeta_b, \\ \zeta_i \leq \zeta_{i+1}, \qquad i &= 1, \ldots, n+m, \\ \zeta_i < \zeta_{i+m}, \qquad i &= 2, \ldots, n, \end{aligned}$$

heißt Knotenvektor *oder* Knotenfolge *zu $[\zeta_a, \zeta_b]$. Ein Knoten ζ_i mit*

$$\zeta_{i-1} < \zeta_i = \ldots = \zeta_{i+k-1} < \zeta_{i+k}$$

hat die Vielfachheit[1] *k. Die Knoten ζ_1 bis ζ_{m+1} sind die linken* Randknoten*, ζ_{n+1} bis ζ_{n+m+1} entsprechend die rechten. Die* inneren Knoten *bestehen aus ζ_{m+2} bis ζ_n. Man spricht von einer* einfachen Knotenfolge*, wenn die inneren Knoten Vielfachheit eins besitzen. Die maximale Vielfachheit eines Randknoten ist $m + 1$, die eines inneren Knotens m.*

[1] Für $k = 1$ auch *einfacher* Knoten genannt.

Definition 4.3. *Sei $T = T_{m,n}$ ein Knotenvektor. Der j-te* B–Spline*, $j = 1, \ldots, n$,*

$$N_j^m(\cdot|T) : \mathbb{R} \ni \zeta \longmapsto N_j^m(\zeta|T) \in \mathbb{R}$$

vom Grad m zum Knotenvektor T ist rekursiv definiert durch:

- $$N_j^0(\zeta|T) = \begin{cases} 1, & \zeta \in [\zeta_j, \zeta_{j+1}), \\ 0, & \textit{sonst.} \end{cases} \qquad j = 1, \ldots, n+m.$$

- $1 \leq k \leq m$*:*

$$u_0 = \frac{\zeta - \zeta_j}{\zeta_{j+k} - \zeta_j}, \qquad \textit{falls} \quad \zeta_{j+k} - \zeta_j > 0, \qquad \textit{andernfalls} \quad u_0 = 0,$$
$$u_1 = \frac{\zeta - \zeta_{j+1}}{\zeta_{j+k+1} - \zeta_{j+1}}, \qquad \textit{falls} \quad \zeta_{j+k+1} - \zeta_{j+1} > 0, \qquad \textit{andernfalls} \quad u_1 = 0,$$

$$N_j^k(\zeta|T) = u_0 \, N_j^{k-1}(\zeta|T) + (1 - u_1) \, N_{j+1}^{k-1}(\zeta|T), \qquad j = 1, \ldots, n+m-k.$$

Der von den B–Splines zur Knotenfolge T aufgespannte Vektorraum heißt $\mathbb{S}_{m,T}$ und hat die Dimension n. Mit dem Vektor der Kontrollpunkte $d = (d_1, \ldots, d_n) \in \mathbb{R}^{1 \times n}$ ist die Linearkombination

$$N_{m,T} d(\cdot) := \sum_{j=1}^{n} d_i N_j^m(\cdot|T)$$

eine Splinefunktion aus $\mathbb{S}_{m,T}$.

Passend dazu nutzen wir die Notation

$$N^m(\zeta) := N^m(\zeta|T) := \left[N_1^m(\zeta|T) \ \ldots \ N_n^m(\zeta|T) \right]^T \in \mathbb{R}^n, \qquad \zeta \in [\zeta_a, \zeta_b],$$

und können damit

$$N_{m,T} d(\zeta) = d \cdot N^m(\zeta|T)$$

schreiben.

Proposition 4.4. *Seien die Voraussetzungen aus Definition 4.3 erfüllt. Dann gilt für jedes $j \in \{1, \ldots, n\}$ mit $(\zeta_j, \zeta_{j+m+1}) \neq \emptyset$:*

$$N_j^m(\zeta|T) > 0, \qquad \zeta \in (\zeta_j, \zeta_{j+m+1}),$$
$$N_j^m(\zeta|T) = 0, \qquad \zeta \notin (\zeta_j, \zeta_{j+m+1}).$$

Sind B eine Bruchstellenfolge und T ein Knotenvektor und ist jeder Knoten aus T in B enthalten und umgekehrt jede Bruchstelle auch ein Knoten, dann ist $\mathbb{S}_{m,T}$ ein Unterraum von $\mathbb{S}_m(B)$. Haben die äußeren Knoten von T außerdem die Vielfachheit $m+1$ und die inneren m, so sind die beiden Vektorräume identisch.

Mit der Vielfachheit der Knoten hängt die Differenzierbarkeit der Splinefunktionen an den Knoten zusammen. Ein Knoten ζ_i, $\zeta_i > \zeta_{i-1}$, mit Vielfachheit $k \in \{1,\dots,m\}$ garantiert, dass alle $f \in \mathbb{S}_{m,T}$ an dieser Stelle $(m-k)$–mal stetig differenzierbar sind[2], das heißt

$$f|_{(\zeta_{i-1},\zeta_{i+k})} \in C^{m-k}((\zeta_{i-1},\zeta_{i+k})).$$

Mit einfachen inneren Knoten hat der Spline die maximale Differenzierbarkeit.

Proposition 4.5. *Sei T eine Knotenfolge und d ein Vektor mit Kontrollpunkten, dann gilt für alle $j \in \{m+1,\dots,n\}$ und für $\zeta \in [\zeta_j,\zeta_{j+1}]$:*

$$N_{m,T}d(\zeta) \in [\,\min\{d_{j-m},\dots,d_j\},\max\{d_{j-m},\dots,d_j\}\,].$$

Der Verlauf des Splines auf dem Intervall $[\zeta_j,\zeta_{j+1}]$ liegt also in der konvexen Hülle der Punkte d_{j-m} bis d_j.

Aus den letzten beiden Propositionen folgt, dass für die Auswertung eines Splines auf dem Intervall (ζ_i,ζ_{i+1}) die Knoten ζ_{i-m+1} bis ζ_{i+m} und die Kontrollpunkte d_{i-m} bis d_i relevant sind. Liegen diese Indizes teilweise außerhalb $\{1,\dots,n\}$, so existieren die entsprechenden Kontrollpunkte und B–Splines nicht und können daher für die Berechnung des Splines keine Rolle spielen. Betrachten wir Splinefunktionen nur auf dem Intervall $[\zeta_{m+1},\zeta_{n+1}]$, so findet keine Überschreitung der Indizes statt. Daher ist folgende Konvention sinnvoll.

Bemerkung 4.6. *Wenn im Folgenden nicht anders angemerkt, ist $T_{m,n}$ eine Knotenfolge mit $(m+1)$–fachen Randknoten und wir betrachten die Splinefunktion nur auf dem Intervall $[\zeta_1,\zeta_{n+m+1}] = [\zeta_{m+1},\zeta_{n+1}]$.*

Satz 4.7. *Sei $T = T_{m,n}$ eine Knotenfolge, d ein Vektor mit Kontrollpunkten und ζ_j mit $\zeta_{j-1} < \zeta_j = \ldots = \zeta_{j+r-1} < \zeta_{j+r}$ ein Knoten der Vielfachheit $r < m$. Mit $d_0 := d_{n+1} := 0$ gilt für $\zeta \in (\zeta_{j-1},\zeta_{j+r})$*

$$\left(\frac{d}{d\zeta}N_{m,T}d\right)(\zeta) = m\sum_{k=1}^{n+1}\frac{d_k-d_{k-1}}{\zeta_{k+m}-\zeta_k}\,N_k^{m-1}(\zeta|T). \tag{4.1}$$

Die Formel in diesem Satz kann auch auf die Randstellen ζ_1 und ζ_{n+m+1} übernommen werden. In diesen Fällen gilt die Formel für $[\zeta_1,\zeta_{m+2})$ beziehungsweise $(\zeta_{n-1},\zeta_{n+m+1}]$.

[2] Auf den offenen Intervallen zwischen zwei Knoten ist der Spline natürlich unendlich oft stetig differenzierbar. Außerdem kann der Spline bei ζ_i auch mehr als $(m-k)$–mal differenzierbar sein.

Definition 4.8. *Sei $T = T_{m,n}$ eine Knotenfolge und $\mathbf{t} = (t_1, \ldots, t_{n_t}) \in \mathbb{R}^{1\times n_t}$, $t_i \in [\zeta_1, \zeta_{n+m+1}]$, eine streng monoton steigende Folge. Dann heißt*

$$\begin{aligned}\mathbf{N}_m(\mathbf{t}) = \mathbf{N}_{m,T}(\mathbf{t}) &= \left[\, N_i^m(t_j|T) \mid i = 1, \ldots, n,\ j = 1, \ldots, n_t \,\right] \\ &= \left[\, N^m(t_1|T),\ \ldots,\ N^m(t_{n_t}|T) \,\right] \in \mathbb{R}^{n\times n_t}\end{aligned}$$

Vandermondsche Matrix *oder* Kollokationsmatrix *zur Knotenfolge T und zur Punktefolge $\mathbf{t}$. Außerdem verwenden wir die Schreibweise*

$$N_i^m(\mathbf{t}|T) = \left[\, N_i^m(t_1|T)\ \ldots\ N_i^m(t_{n_t}|T) \,\right] \in \mathbb{R}^{1\times n_t}, \quad i = 1, \ldots, n.$$

Sind die Voraussetzungen aus Definition 4.8 mit $n = n_t$ erfüllt und ist $\mathbf{y} = (y_1, \ldots, y_{n_t})^T \in \mathbb{R}^{n_t}$, dann gibt es genau dann einen eindeutigen (interpolierenden) Spline $f \in \mathbb{S}_{m,T}$ mit

$$f(t_j) = y_j, \qquad j = 1, \ldots, n_t,$$

wenn die in diesem Fall quadratische Matrix $\mathbf{N}_m(\mathbf{t})$ invertierbar ist. Das wiederum ist genau dann der Fall, wenn die Bedingung von Schoenberg und Whitney erfüllt ist.

Satz 4.9 (Schoenberg–Whitney)**.** *Seien die Voraussetzungen aus Definition 4.8 gegeben und $n = n_t$. Die Matrix $\mathbf{N}_m(\mathbf{t})$ ist genau dann invertierbar, wenn*

$$\zeta_j < t_j < \zeta_{j+m+1}, \quad j = 1, \ldots, n, \tag{4.2}$$

gilt.

Besteht die Knotenfolge T aus $(m+1)$–fachen Randknoten, so ist die Bedingung nicht erfüllt, wenn $\zeta_1 = t_1$ gilt. Allerdings ist die Berechnung eines Splinewertes (generell) völlig unabhängig vom Wert für ζ_1 und ζ_{n+m+1}. Das wirkt sich natürlich auf die Matrix $\mathbf{N}_m(\mathbf{t})$ aus. Deswegen müssen in diesem Fall weder für $j = 1$ die linke und noch für $j = n$ die rechte Ungleichung aus (4.2) erfüllt sein.

4.2.2. Least–Squares–Approximation

Dieser und die folgenden Unterabschnitte von 4.2 behandeln Approximationsverfahren, die zu gegebenen Daten

$$\mathbf{t} = (t_1, \ldots, t_{n_t}) \in \mathbb{R}^{1\times n_t}, \qquad \mathbf{x} = (x_1, \ldots, x_{n_t}) \in \mathbb{R}^{1\times n_t}, \tag{4.3}$$

und einer Toleranz $\varepsilon > 0$ eine Splinefunktion $f : [t_1, t_{n_t}] \to \mathbb{R}$ suchen, die

$$\max_{i=1,\ldots,n_t} \{|f(t_i) - x_i|\} \le \varepsilon \tag{4.4}$$

erfüllt und zudem verschiedene Randbedingungen einhält. Wenn es um die Approximation dieser Daten geht, soll die Knotenfolge eines Splineapproximanten immer $(m+1)$–fache Randknoten besitzen, wobei der linke mit t_1 und der rechte mit t_{n_t} übereinstimmt.

Das erste Vorgehen nutzt zur Lösung der Aufgabe eine iterative Berechnung des Least–Squares–Approximanten.

Definition 4.10. *Seien zu approximierende Daten wie in* (4.3) *gegeben und sei $T_{m,n}$ eine Knotenfolge mit $\zeta_{m+1} = t_1$ und $\zeta_{n+1} = t_{n_t}$. Dann ist ein* Least–Squares–Approximant *ein Spline $f^* \in \mathbb{S}_{m,T}$, der*

$$E(f^*) = \sum_{i=1}^{n_t} (f^*(t_i) - x_i)^2 \leq \sum_{i=1}^{n_t} (f(t_i) - x_i)^2 = E(f), \qquad f \in \mathbb{S}_{m,T}, \tag{4.5}$$

erfüllt.

Da ein Spline bei fester Knotenfolge eindeutig über die Kontrollpunkte definiert ist, ergibt sich mit der Kollokationsmatrix aus Definition 4.8 für (4.5) das Minimierungsproblem

$$\left|\left| d\mathbf{N}_{m,T}(\mathbf{t}) - \mathbf{x} \right|\right|_2^2 \longrightarrow \min! \tag{4.6}$$

Es hat genau dann eine eindeutige Lösung, wenn die Matrix $\mathbf{N}_{m,T}(\mathbf{t})$ vollen Rang besitzt ([20]). Das wiederum ist erfüllt, wenn eine n–elementige Teilmenge von $\mathbf{t}$ zusammen mit $T_{m,n}$ die Schoenberg–Whitney–Bedingung aus Satz 4.9 erfüllt.

In dieser Arbeit entspricht der Approximant einer Bewegung der Grobachse. Sie muss zum Anfangszeitpunkt t_1 und zum Endzeitpunkt t_{n_t} Vorgaben erfüllen. Da in (4.6) alle Kontrollpunkte des Splines Variablen sind, erfüllt die Lösung nicht unbedingt die Vorgaben. Soll der Spline an den Rändern t_1 und t_{n_t} vorgegebene Positionen oder Werte für Ableitungen einhalten, muss (4.6) modifiziert werden. Ein Beispiel ist, wenn die Bewegung am Anfang und am Ende anhält, das heißt, die erste Ableitung des Approximanten muss bei t_1 und bei t_{n_t} gleich null sein. Damit ein Spline solche Bedingungen erfüllt, müssen die Kontrollpunkte nahe am Rand gewisse Abhängigkeiten zueinander aufweisen, wie im Folgenden genauer ausgeführt ist. Die Konstanten P_1, V_1 und A_1 sind die Vorgaben für Position, Geschwindigkeit und Beschleunigung bei t_1 und P_{n_t}, V_{n_t} und A_{n_t} die entsprechenden bei t_{n_t}. Da ein Spline mit $m+1$–fachen Randknoten den ersten Kontrollpunkt interpoliert, muss

$$d_1 = P_1$$

gelten. Für die Ableitungen folgen aus (4.1) die Abhängigkeiten

$$\begin{aligned} d_1 &= d_2 - \frac{V_1}{m}(\zeta_{2+m} - \zeta_2) =: d_2 - M_1, \\ d_1 &= d_3 - \left(\frac{A_1(\zeta_{2+m} - \zeta_3)(\zeta_{3+m} - \zeta_3)}{m(m-1)} + \frac{V_1[(\zeta_{3+m} - \zeta_3) + (\zeta_{2+m} - \zeta_2)]}{m} \right) =: d_3 - K_1, \end{aligned}$$

$$d_2 = d_3 - \left(\frac{A_1(\zeta_{2+m} - \zeta_3)(\zeta_{3+m} - \zeta_3)}{m(m-1)} + \frac{V_1(\zeta_{3+m} - \zeta_3)}{m} \right) =: d_3 - L_1.$$

Analog gelten für den rechten Rand:

$$\begin{aligned}
d_n &= P_{n_t}, \\
d_n &= d_{n-1} - \frac{V_{n_t}}{m}(\zeta_{n+m} - \zeta_n) =: d_{n-1} - M_{n_t}, \\
d_n &= d_{n-2} - \left(\frac{A_{n_t}(\zeta_n - \zeta_{n+m-1})(\zeta_{n-1} - \zeta_{n+m-1})}{m(m-1)} \right) \\
&\qquad - \left(\frac{V_{n_t}[(\zeta_{n-1} - \zeta_{n+m-1}) + (\zeta_n - \zeta_{n+m})]}{m} \right) \\
&=: d_{n-2} - K_{n_t}, \\
d_{n-1} &= d_{n-2} - \left(\frac{A_{n_t}(\zeta_n - \zeta_{n+m-1})(\zeta_{n-1} - \zeta_{n+m-1})}{m(m-1)} + \frac{V_{n_t}(\zeta_{n-1} - \zeta_{n+m-1})}{m} \right) \\
&=: d_{n-2} - L_{n_t}.
\end{aligned}$$

Auf diese Beziehungen wird in der Arbeit immer wieder zurückgegriffen.

Randbedingungen führen also dazu, dass nicht mehr alle Kontrollpunkte d_1 bis d_n variabel sind und die Dimension des Lösungsraums für die Approximation sinkt. Generell kann man $d \in \mathbb{R}^{1\times n}$ schreiben als

$$\begin{gathered}
d = \tilde{d}I + v, \\
I \in \mathbb{R}^{\tilde{n}\times n}, \quad v \in \mathbb{R}^{1\times n}, \quad \tilde{d} \in \mathbb{R}^{1\times \tilde{n}},
\end{gathered} \tag{4.7}$$

wobei $\tilde{n} \leq n$ gilt und $\tilde{d}$ der Variablenvektor ist. Die Matrix I und der Vektor v sind dabei so gewählt, dass d unabhängig von $\tilde{d}$ die Randbedingungen erfüllt. Details über die Gestalt von I und v folgen im kommenden Abschnitt. Aus dem Minimierungsproblem (4.6) wird dann

$$\min_{\tilde{d}\in\mathbb{R}^{1\times\tilde{n}}} \left\{ \left|\left| \left[\tilde{d}I + v\right] \mathbf{N}_m(\mathbf{t}) - \mathbf{x} \right|\right|_2^2 \right\} = \min_{\tilde{d}\in\mathbb{R}^{1\times\tilde{n}}} \left\{ \left|\left| \tilde{d}I\mathbf{N}_m(\mathbf{t}) + (v\mathbf{N}_m(\mathbf{t}) - \mathbf{x}) \right|\right|_2^2 \right\}$$

und das ist genau dann eindeutig lösbar, wenn $I\mathbf{N}_n(\mathbf{t})$ vollen Rang besitzt.

Um mit dem Least–Squares–Approximanten die Toleranzbedingung (4.4) einzuhalten, ist die Wahl der Knotenfolge entscheidend. Damit die Anzahl der Knoten möglichst gering bleibt, werden die Verfahren ausgehend von einer sehr groben Knotenfolge iterativ Knoten einfügen oder die Knotenfolge schrittweise verfeinern, bis der Approximant (4.4) erfüllt. Eine sinnvolle Strategie für unsere Anwendung ist weiter unten in Abschnitt 4.3.1 beschrieben. Generell endet die Einfüge- oder Verfeinerungsstrategie hier im schlimmsten Fall nach einer endlichen Anzahl von Schritten in einer Knotenfolge $T_{\max}$. Der dazugehörige Approximant interpoliert die Datenpunkte und damit ist natürlich auch (4.4) erfüllt. Das genaue Aussehen von $T_{\max}$ folgt im nächsten Abschnitt.

4.2.3. Konkrete Randbedingungen für die Approximanten

Der in dieser Arbeit relevante Fall für die Randbedingungen hat eine vorgegebene Position P_1 bei t_1 und feste erste Ableitungen V_1 und V_{n_t} bei t_1 beziehungsweise t_{n_t}.

Die Matrizen und Vektoren aus (4.7) für eine Knotenfolge $T = T_{m,n}$ sind hier

$$\tilde{d} = (d_3, \ldots, d_{n-1}) \in \mathbb{R}^{1\times\tilde{n}}, \qquad \tilde{n} = n - 3, \tag{4.8}$$

$$I = \left[\begin{array}{cc|c|c} 0 & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ & & E_{n-3} & \vdots \\ \vdots & \vdots & & 0 \\ 0 & 0 & & 1 \end{array}\right] \in \mathbb{R}^{\tilde{n}\times n}, \tag{4.9}$$

$$v = (P_1, P_1 + M_1, 0, \ldots, 0, -M_{n_t}) \in \mathbb{R}^{1\times n}.$$

Die Matrix E_{n-3} ist die $(n-3)\times(n-3)$–Einheitsmatrix. Die beiden ersten Kontrollpunkte aus $d = \tilde{d}I + v$ sind durch die linken Randbedingungen definiert und der letzte ist abhänging vom vorletzten. Für jedes $\tilde{d} \in \mathbb{R}^{1\times\tilde{n}}$ erfüllt der Spline aus T und d also die Randbedingungen. Für die Approximation betrachten wir zudem die um t_1 und x_1 reduzierten Datenfolgen

$$\tilde{\mathbf{t}} = (t_2, \ldots, t_{n_t}), \qquad \text{und} \qquad \tilde{\mathbf{x}} = (x_2, \ldots, x_{n_t}), \tag{4.10}$$

da der Abstand $N_{m,T}d(t_1) - x_1$ unabhängig von $\tilde{d}$ konstant ist.

Wie im vorherigen Abschnitt schon erwähnt, liegt dem iterativen Approximationsverfahren eine maximale Knotenfolge

$$T_{\max} = (\zeta_1^{\max}, \ldots, \zeta_{n_{\max}+m+1}^{\max})$$

zugrunde, sodass der Least–Squares–Approximant dazu die Datenpunkte (4.4) interpoliert. Das iterative Vorgehen baut daraus eine Knotenfolge auf. Es beginnt mit einer Knotenfolge, die nur aus den Randknoten besteht, und fügt darin solange Knoten aus $T_{\max}$ ein, bis der Fehler des Approximanten die Toleranz ε unterschreitet. Die so entstehende Knotenfolge ist immer eine Teilmenge von $T_{\max}$ und im extremen Fall sind beide identisch. Dann interpoliert der Approximant die Daten. Diesen Fall betrachten wir im Folgenden etwas genauer.

Wir haben hier $n_t - 1$ Interpolationsbedingungen und es muss daher $\tilde{n}_{\max} = n_t - 1$ gelten, wobei $\tilde{n}_{\max}$ gemäß (4.8) passend zu $n_{\max}$ ist. Daraus folgt

$$n_{\max} = \tilde{n}_{\max} + 3 = n_t - 1 + 3 = n_t + 2$$

als Anzahl der Kontrollpunkte des interpolierenden Splines, der auch die Randbedingungen erfüllt. Da die Randknoten von $T_{\max}$ Vielfachheit $m+1$ besitzen, bleiben

$$n_{\max} + m + 1 - 2(m+1) = n_t - m + 1$$

innere Knoten übrig, die sich auf (t_1, t_{n_t}) verteilen müssen. Dabei spielt die Vielfachheit eine Rolle. Wir gehen im Folgenden immer so vor, dass die inneren Knoten von $T_{\max}$ alle dieselbe Vielfachheit k_{vf} besitzen. Das bedeutet natürlich, dass $n_t - m + 1$ ein Vielfaches von k_{vf} sein muss. Ist das nicht der Fall, können die Daten $\tilde{\mathbf{t}}$ und $\tilde{\mathbf{x}}$ am Ende mit x_{n_t} und entsprechenden weiterlaufenden Zeitpunkten aufgefüllt werden. In den betrachteten Anwendungsfällen der Approximation hat das keine wesentlichen Auswirkungen, da sich die Anzahl der Daten um weniger als k_{vf} vergrößert. Wir können also von

$$n_t - m + 1 = k_{\mathrm{vf}} q$$

für ein $q \in \mathbb{N}$ ausgehen. Es sei an dieser Stelle erwähnt, dass für unsere Anwendung nicht alle möglichen Kombinationen von Splinegrad m und Vielfachheit k_{vf} möglich sind, da die Ableitungen der Bewegung der Grobachse gewisse Stetigkeitseigenschaften haben müssen. Betrachten wir bescheunigungsbegrenzte Bewegungen, muss $k_{\mathrm{vf}} \leq m-1$ gelten, damit der Approximant C^1–stetig ist. Im ruckbegrenzten Fall ist für die C^2–Stetigkeit $k_{\mathrm{vf}} \leq m-2$ notwendig.

Zunächst betrachten wir $T_{\max}$ für den Spezialfall

$$\mathbf{t} = (t_1^*, \ldots, t_{n_t}^*) = (1, \ldots, n_t).$$

Die inneren Knoten sollen sich hier gleichmäßig verteilen, das heißt, ein Intervall der Länge $n_t - 1$ muss in $q+1$ gleichlange Abschnitte aufgeteilt werden. Das ergibt einen Abstand von

$$\Delta_{\mathrm{Tmax}} = \frac{n_t - 1}{q+1}.$$

Für die Knotenfolge $T_{\max,1} := (\theta_1^{\max}, \ldots, \theta_{n_{\max}+m+1}^{\max})$ folgt daher

$$\begin{aligned} \theta_1^{\max} = \ldots = \theta_{m+1}^{\max} = t_1^* = 1, \qquad \theta_{n_{\max}+1}^{\max} = \ldots = \theta_{n_{\max}+m+1}^{\max} = t_{n_t}^* = n_t, \\ \theta_{m+1+(i-1)k_{\mathrm{vf}}+1}^{\max} = \ldots = \theta_{m+1+(i-1)k_{\mathrm{vf}}+k_{\mathrm{vf}}}^{\max} = 1 + i\Delta_{\mathrm{Tmax}}, \quad i = 1, \ldots, q. \end{aligned} \tag{4.11}$$

Mit der Kollokationsmatrix $\mathbf{N}_{m,T_{\max,1}}(\tilde{\mathbf{t}}) \in \mathbb{R}^{n_{\max} \times \tilde{n}}$ muss nun die Invertierbarkeit der Matrix

$$G_1 := I\mathbf{N}_{m,T_{\max,1}}(\tilde{\mathbf{t}}) = \begin{bmatrix} N_3^m(\tilde{\mathbf{t}}|T_{\max,1}) \\ N_4^m(\tilde{\mathbf{t}}|T_{\max,1}) \\ \vdots \\ N_{n-2}^m(\tilde{\mathbf{t}}|T_{\max,1}) \\ N_{n-1}^m(\tilde{\mathbf{t}}|T_{\max,1}) + N_n^m(\tilde{\mathbf{t}}|T_{\max,1}) \end{bmatrix} \in \mathbb{R}^{\tilde{n} \times \tilde{n}} \tag{4.12}$$

gezeigt werden, wobei hier $\tilde{\mathbf{t}} = (t_2^*, \ldots, t_{n_t}^*) = (2, \ldots, n_t)$ ist.

Proposition 4.11. *Sei $T_{\max,1}$ die Knotenfolge aus* (4.11)*. Dann ist die Matrix G_1 aus* (4.12) *invertierbar.*

Beweis. Nimmt man von der Matrix G_1 die letzte Zeile und die letzte Spalte weg, so ergibt sich die quadratische Kollokationsmatrix $G_2 := \mathbf{N}_{m,T_\mu}(\hat{\mathbf{t}})$ der Knotenfolge

$$\begin{aligned} T_\mu &= (\mu_1, \ldots, \mu_{\hat{n}+m+1}), \qquad \hat{n} = n_t - 2, \\ \mu_1 &= \theta_3^{\max},\ \ldots\ ,\ \mu_{\hat{n}+m+1} = \theta_{n_{\max}+m-1}^{\max}, \end{aligned}$$

und der Punkte

$$\begin{aligned} \hat{\mathbf{t}} &= (\tau_1, \ldots, \tau_{\hat{n}}), \\ \tau_1 = t_2^*,\ \ldots,\ \tau_{\hat{n}} &= t_{n_t-1}^*. \end{aligned}$$

Sie ist genau dann invertierbar, wenn die Bedingung von Schoenberg und Whitney erfüllt ist, das heißt, es muss

$$\mu_j < \tau_j < \mu_{j+m+1}, \qquad j = 1, \ldots, \hat{n},$$

gelten.

Wir beginnen mit der linken Ungleichung und setzen $k := k_{\mathrm{vf}}$. Für $j = 1, \ldots, m-1$ ist sie erfüllt, da in diesem Fall $\mu_j = t_1^* < t_2^* = \tau_1$ gilt. Sei jetzt $j \in \{m, \ldots, \hat{n}\}$. Aus $\mu_j = \theta_{j+2}^{\max}$ folgt die Beziehung

$$\begin{aligned} 1 + i\Delta_{\mathrm{Tmax}} &= \theta_{m+1+(i-1)k+1}^{\max} = \ldots = \theta_{m+1+(i-1)k+k}^{\max} \\ &= \mu_{m-1+(i-1)k+1} = \ldots = \mu_{m-1+(i-1)k+k}, \qquad i = 1, \ldots, q. \end{aligned} \tag{4.13}$$

Sei nun

$$i := \max\left\{\iota \in \mathbb{N} \mid j \geq m - 1 + (\iota - 1)k + 1\right\}.$$

Daraus folgt $i \leq (j - m + k)/k$ und es gilt

$$\begin{aligned} \mu_j &= 1 + i\Delta_{\mathrm{Tmax}} \\ &\leq 1 + \frac{j-m+k}{k}\,\frac{n_t - 1}{q+1} = 1 + \frac{j-m+k}{k}\,\frac{(n_t-1)k}{n_t - m + 1 + k} \\ &= 1 + \frac{jn_t - mn_t + kn_t - j + m - k}{n_t - m + 1 + k} \\ &= 1 + \frac{jn_t - jm + j + jk + jm - 2j - jk - mn_t + kn_t + m - k}{n_t - m + 1 + k} \\ &= 1 + j + \underbrace{\frac{jm - 2j - jk - mn_t + kn_t + m - k}{n_t - m + 1 + k}}_{=:c_1}. \end{aligned}$$

Da

$$jm - 2j - jk - mn_t + kn_t + m - k = (n_t - 1 - j)(k - m)$$

und

$$k - m < 0, \qquad n_t - 1 - j \geq 0,$$

gelten, folgt $c_1 < 0$. Das wiederum impliziert $\mu_j < j + 1 = \tau_j$, also genau das, was wir zeigen wollten. Ähnlich verläuft die Argumentation für $\tau_j < \mu_{j+m+1}$.

Die letzte Spalte der Matrix G_1 besteht, bis auf das letzte Element, ausschließlich aus Nullen. Das liegt an $N_j^m(t_{n_t}|T_{\max,1}) = 0$, $j = 3, \ldots, n_{\max} - 2$. Für das letzte Element gilt

$$N_{n-1}^m(t_{n_t}|T_{\max,1}) + N_n^m(t_{n_t}|T_{\max,1}) = 0 + 1 = 1.$$

Zusammen mit der Invertierbarkeit von G_2 folgt sie auch für G_1. □

Die maximale Knotenfolge $T_{\max}$ für eine beliebige, streng monoton steigende Folge $\mathbf{t}$ leiten wir aus dem gerade betrachteten $T_{\max,1}$ ab. Die Knoten sind

$$\begin{gathered}\zeta_i^{\max} = t_p + (\theta_i^{\max} - p)\ (t_{p+1} - t_p)\,, \qquad p = \lfloor \theta_i^{\max} \rfloor, \qquad i = 1, \ldots, n_{\max}. \\ \zeta_{n_{\max}+1}^{\max} = \ldots = \zeta_{n_{\max}+m+1}^{\max} = t_{n_t}.\end{gathered} \tag{4.14}$$

Durch diese Wahl der Knoten folgt aus $\theta_i^{\max} \in [j, j+1)$ für ein beliebiges $j \in \{1, \ldots, n_t - 1\}$, dass $\zeta_i^{\max} \in [t_j, t_{j+1}]$ gilt. Deswegen überträgt sich Proposition 4.11 direkt auf den allgemeinen Fall, und $I\mathbf{N}_{m,T_{\max}}(\tilde{\mathbf{t}})$ ist invertierbar.

Nimmt man als Knotenfolge für die Least–Squares–Approximation eine Teilfolge von $T_{\max}$ mit denselben Randknoten, dann ist der Least–Squares–Approximant immer eindeutig bestimmt, wie die folgende Proposition zeigt.

Proposition 4.12. *Seien $T_{\max}$ die Knotenfolge aus* (4.14)*, $\tilde{\mathbf{t}}$ die Folge aus* (4.10) *und I die Matrix aus* (4.9)*. Sei außerdem T eine Teilfolge von $T_{\max}$ mit*

$$\zeta_1 = \zeta_1^{\max}, \ \ldots\ , \ \zeta_{m+1} = \zeta_{m+1}^{\max}, \tag{4.15}$$

$$\zeta_{n+1} = \zeta_{n_{\max}+1}, \ \ldots\ , \ \zeta_{n+m+1} = \zeta_{n_{\max}+m+1}^{\max}. \tag{4.16}$$

Dann hat die Matrix

$$G_3 := I\mathbf{N}_{m,T}(\tilde{\mathbf{t}})$$

vollen Rang $n - 3$.

Beweis. Zuerst schreiben wir

$$G_3 = \left[\begin{array}{c|c} G_4 & \mathbf{0}_{n-4} \\ \hline * & 1 \end{array}\right] \in \mathbb{R}^{n-3\times n_t-1},$$

wobei $G_4 = \mathbf{N}_{m,T_1}(\hat{\mathbf{t}}) \in \mathbb{R}^{n-4\times n_t-2}$ mit $T_1 = (\zeta_3, \ldots, \zeta_{n+m-1})$ ist. Das Aussehen der letzten Zeile und letzten Spalte von G_3 folgt ähnlich wie im Beweis von Proposition 4.11. Die Matrix G_3 hat genau dann vollen Rang, wenn G_4 diesen hat.

Das ist hier auch der Fall. Die Folge T_1 ist eine Teilfolge von $(\zeta_3^{\max}, \ldots, \zeta_{n_{\max}+m-1}^{\max})$, die bekanntermaßen zusammen mit $(t_2, \ldots, t_{n_t})$ die Bedingung von Schoenberg und Whitney erfüllt. Daher gibt es eine Teilfolge von $(t_2, \ldots, t_{n_t})$ die mit T_1 ebenfalls diese Bedingung erfüllt. Daraus folgt der volle Rang von G_4. □

Es gibt noch weitere Arten von Randbedingungen, die im Zusammenhang mit Bewegungen denkbar sind. Zum Beispiel können nur die Geschwindigkeiten am Rand feststehen. Da diese Fälle in der Arbeit nicht vorkommen, findet hier keine Untersuchung statt.

4.2.4. Tschebyschow–Approximation

Eine Alternative zur Least–Squares–Approximation ist die Tschebyschow–Approximation. Sie liefert zur Knotenfolge T direkt den Spline mit dem minimalen Maximalabstand.

Definition 4.13. *Seien die Voraussetzungen aus Definition 4.10 efüllt. Eine* Tschebyschow–Approximation *ist ein Spline $f^* \in \mathbb{S}_{m,T}$ mit*

$$\max_{i\in\{1,\ldots,n_t\}} |f^*(t_i) - x_i| = ||f^*(\mathbf{t}) - \mathbf{x}||_\infty \leq ||f(\mathbf{t}) - \mathbf{x}||_\infty, \qquad f \in \mathbb{S}_{m,T}.$$

Ein solcher Approximant ist die Lösung eines linearen Programms, also einem Optimierungsproblem mit linearer Zielfunktion und linearen Nebenbedingungen. Wir verwenden wieder die Schreibweise (4.7) für die Einhaltung der Randbedingungen. Für die Variablen des linearen Programms erweitert man die Kontrollpunkte $\tilde{d} = (\tilde{d}_1, \ldots, \tilde{d}_{\tilde{n}})$ um eine weitere reelle Variable d_ε. Aus

$$\left|\left[\tilde{d}I + v\right]\mathbf{N}_m(t_i) - x_i\right| \leq d_\varepsilon \quad \Leftrightarrow \quad -d_\varepsilon \leq \left[\tilde{d}I + v\right]\mathbf{N}_m(t_i) - x_i \leq d_\varepsilon, \quad i = 1, \ldots, n_t,$$

ergeben sich die linearen Nebenbedingungen

$$\begin{aligned} \left[\tilde{d}\ \ d_\varepsilon\right]\left[\begin{array}{c} -\mathbf{N}_m(t_i) \\ -1 \end{array}\right] &\leq +v\mathbf{N}_m(t_i) - x_i, \\ \left[\tilde{d}\ \ d_\varepsilon\right]\left[\begin{array}{c} \mathbf{N}_m(t_i) \\ -1 \end{array}\right] &\leq -v\mathbf{N}_m(t_i) + x_i, \qquad i = 1, \ldots, n_t. \end{aligned}$$

Die Zielfunktion ist

$$F([\tilde{d}, d_\varepsilon]) = d_\varepsilon = \min!$$

Wie bei der Least–Squares–Approximation ist der Fehler abhängig von der Knotenfolge. Deswegen wird auch hier eine iterative Einfüge- oder Verfeinerungsstrategie gewählt, die im schlimmsten Fall zum Interpolanten führt. Weitere Details folgen in Abschnitt 4.3.1.

4.2.5. Approximation mit der Idee des Schoenbergschen Splineoperators

Eine weitere Alternative zur Näherung diskreter Daten besteht aus einem Vorgehen, dass auf der Idee des Schoenbergschen Splineoperators basiert. Er bietet die Möglichkeit, reelle Funktionen durch Splines zu approximieren. Für Funktionen mit beschränkter erster Ableitung lässt sich der Approximationsfehler abschätzen. Sind diskrete Daten die Abtastung einer reellen Funktion, gelten ähnliche Abschätzungen.

Definition 4.14. *Seien $[a, b]$ ein Intervall und $T_{m,n}$ eine Knotenfolge mit $\zeta_1 = a$ und $\zeta_{n+m+1} = b$, dann nennt man*

$$S_{m,T} : C^1([a, b]) \ni x \longmapsto S_{m,T}x \in \mathbb{S}_{m,T}$$

mit

$$S_{m,T}x(\zeta) = \sum_{i=1}^{n} x(\xi_i)\ N_i^m(\zeta|T), \quad x \in C^1([a, b]), \quad \zeta \in [a, b],$$

$$\xi_i = \frac{\zeta_{i+1} + \ldots + \zeta_{i+m}}{m},$$

den Schoenbergschen Splineoperator[3] *und $S_{m,T}x$ ist der* Schoenberg–Approximant *vom Grad m mit der Knotenfolge T zur Funktion $x \in C^1([a, b])$. Die Punkte ξ_i heißen* Greville–Abszissen.

Aus [4] stammt die folgende Abschätzung für den Approximationsfehler in Abhängigkeit von der betragsmäßig maximalen Ableitung und von der Knotenfolge.

Satz 4.15. *Seien $x \in C^1([a, b])$ und $R = \max\{|\dot{x}(t)| \mid t \in [a, b]\}$ und außerdem $T_{m,n}$ eine Knotenfolge mit $\zeta_1 = a$ und $\zeta_{n+m+1} = b$. Mit $\Delta\zeta = \max_j |\zeta_{j+1} - \zeta_j|$ gilt*

$$|S_{m,T}x(t) - x(t)| \leq R\sqrt{\frac{m+1}{12}}\ \Delta\zeta, \qquad t \in [a, b]. \tag{4.17}$$

Dieser Satz gilt auch, wenn die Funktion $x(t)$ nur stetig und bis auf endlich viele Stellen stetig differenzierbar ist und ein $R > 0$ den Betrag der Ableitung auf den differenzierbaren

[3] Oder einfach kurz *Schoenberg–Operator*.

Bereichen begrenzt. Zu einem erlaubten Fehler $\varepsilon > 0$ ergibt sich mit dem Satz für den Abstand der Knoten die Bedingung

$$\Delta\zeta \leq \frac{\varepsilon}{R}\sqrt{\frac{12}{m+1}}.$$

Eine weitere Abschätzung für den Approximationsfehler im Fall $x \in C^2([a,b])$ findet sich in [6]:

$$\begin{aligned} |S_{m,T}x(t) - x(t)| &\leq \max\left\{(\zeta_{j+1} - \xi_{j-m})^2, (\xi_j - \zeta_j)^2\right\} \frac{||\ddot{x}||_\infty}{2} \\ &\leq \frac{(m+1)^2}{8}\, ||\ddot{x}||_\infty\, \Delta\zeta^2. \end{aligned}$$

Im Allgemeinen ist diese Abschätzung irgendwann die genauere, wenn man ε gegen 0 laufen lässt, da der Fehler quadratisch kleiner wird.

Sind die zu approximierenden Daten wie in (4.3) diskret gegeben und entstehen durch Abtastung $x_i = x(t_i)$ einer Funktion $x \in C^1([a,b])$, so ist mit einer gegebenen Knotenfolge T nicht garantiert, dass die Folge $\mathbf{t}$ jede Greville-Abszisse ξ_i enthält. Daher sind die Kontrollpunkte $x(\xi_i)$ des Schoenberg–Approximanten nicht bekannt. Trotzdem gilt eine ähnliche Abschätzung wie in Satz 4.15, sodass ein ähnlicher Approximant wie $S_{m,T}x(t)$ bestimmt werden kann.

Satz 4.16. *Seien die Voraussetzungen aus Satz 4.15 erfüllt, sei* $\mathbf{t} = (t_1, \ldots, t_{n_t})$ *eine streng monoton steigende Abtastung des Intervalls* $[a,b]$ *mit* $t_1 = a$, $t_{n_t} = b$ *und* $\tilde{t} = \max_j\{t_{j+1} - t_j\}$ *und sei* $\mathbf{x} = (x_1, \ldots, x_{n_t})$ *mit* $x_i = x(t_i)$. *Dazu sei*

$$k_i = \arg\min_j\{|t_j - \xi_i| \mid j = 1, \ldots, n_t\}, \quad i = 1, \ldots, n.$$

Dann gilt

$$\left|\sum_{i=1}^{n} x(t_{k_i}) N_i^m(t|T) - x(t)\right| \leq R\left(\sqrt{\frac{m+1}{12}}\Delta\zeta + \frac{1}{2}\tilde{t}\right). \tag{4.18}$$

Beweis. Wir greifen hier auf dieselben Argumente wie im Beweis von Satz 4.15 zurück, siehe [4]:

$$\begin{aligned} \left|\sum_{i=1}^{n} x(t_{k_i}) N_i^m(t|T) - x(t)\right| &\leq \sum_{i=1}^{n} |x(t_{k_i}) - x(t)| N_i^m(t|T) \\ &\leq R\sum_{i=1}^{n} |t_{k_i} - t| N_i^m(t|T) \\ &\leq R\sum_{i=1}^{n} \left(|\xi_i - t| + \frac{1}{2}\tilde{t}\right) N_i^m(t|T) \end{aligned}$$

$$
\begin{aligned}
&= R \sum_{i=1}^{n} |\xi_i - t| N_i^m(t|T) + \frac{1}{2} R\tilde{t} \\
&\leq R \left(\sqrt{\frac{m+1}{12}} \Delta\zeta + \frac{1}{2}\tilde{t} \right)
\end{aligned}
$$

□

Der Approximant entsteht in (4.18) nicht exakt durch Auswertung der Funktion $x(t)$ an den Greville–Abszissen. Stattdessen ist der i–te Kontrollpunkt ein x_j, $j \in \{1, \ldots, n_t\}$, sodass $|\xi_i - t_j|$ minimal ist. Diese Abweichung macht den Unterschied zwischen den Sätzen 4.15 und 4.16 aus.

Auch aus Satz 4.16 ergibt sich bei einem fest vorgegebenen $\varepsilon > 0$ eine Regel für den maximalen Abstand der Knoten:

$$
\Delta\zeta \leq \frac{\varepsilon - \frac{1}{2} R\tilde{t}}{R} \sqrt{\frac{12}{m+1}}. \tag{4.19}
$$

Je feiner die Abtastung der Funktion $x(t)$ ist, desto näher ist die Abschätzung an der kontinuierlichen Variante aus Satz 4.15. Allerdings müssen ε, $\tilde{t}$ und R in einem passenden Verhältnis stehen, sodass die rechte Seite von (4.19) größer null ist.

Diese Abschätzungen ermöglichen die Bestimmung des gesuchten Approximanten ohne ein iteratives Vorgehen. Allerdings betrachten sie immer einen extremen Fall. Das führt zu einer Knotenfolge des Approximanten, die oftmals viel mehr Knoten enthält, als eigentlich notwendig sind. Mehr Knoten bedeuten in unseren Anwendungsfällen meistens, dass der Approximant höhere zweite Ableitungen besitzt. Das soll aber gerade bei den Bewegungen der groben Achse vermieden werden (siehe Abschnitt 4.3). Deswegen ist auch mit dem Schoenberg–Operator ein iteratives Vorgehen in der Anwendung sinnvoll. Es funktioniert wie dasjenige der Least–Squares–Approximation, nur dass die Knotenfolge $T_{\max}$ die Abschätzung (4.19) nutzt. Beim iterativen Verfeinern führt das im extremen Fall nicht zu einem Interpolanten wie bei der Least–Squares–Approximation, sondern nur zu einem Approximanten, der die Toleranzbedingung erfüllt. Das genügt aber für die Anwendung. Die Details zum Verfeinern der Knoten folgen in Abschnitt 4.3.1.

Der Approximant aus Satz 4.16 erfüllt allerdings nicht die Randbedingungen aus Abschnitt 4.2.3. Dazu müssen drei Kontrollpunkte verändert werden und insgesamt erhält man die n Kontrollpunkte

$$
\begin{gathered}
d_i = x(t_{k_i}), \qquad i = 3, \ldots, n-1, \\
d_1 = P_1, \quad d_2 = d_1 + M_1, \quad d_n = d_{n-1} - M_{n_t}.
\end{gathered}
$$

Die Konstanten kommen aus Abschnitt 4.2.2. Diese Änderung der Kontrollpunkte heben am Rand des Approximanten die Abschätzung aus Satz 4.16 auf. Das kann im ungünstigen Fall zu einer Verletzung der Toleranzbedingung führen. In den Anwendungsfällen aus

Abschnitt 4.3 trat diese Situation nicht ein und deswegen wird dieser mögliche Effekt im Folgenden nicht weiter betrachtet.

4.2.6. Approximation durch Smoothing–Splines

Die folgenden drei Approximationsmethoden beachten nicht nur die Toleranzbedingung (4.4), sondern haben auch das Ziel, möglichst glatte Näherungen zu erzeugen. Wir beginnen mit den Smoothing–Splines.

Definition 4.17. *Seien die zu approximierenden Daten aus* (4.3) *gegeben. Sei* $T_{m,n}$ *eine Knotenfolge mit* $\zeta_{m+1} = t_1, \zeta_{n+1} = t_{n_t}$ *und seien* $\lambda \in [0,1]$ *und* $r \in \mathbb{N}, 0 < r < m$. *Dann ist ein* Smoothing–Spline *eine Funktion* $f^* \in \mathbb{S}_{m,T}$, *die das Funktional*[4]

$$F(f) := (1-\lambda)E(f) + \lambda G(f) \tag{4.20}$$

$$:= (1-\lambda)\sum_{i=1}^{n_t}(f(t_i)-x_i)^2 + \lambda\int_{t_1}^{t_{n_t}}\left(f^{(r)}(t)\right)^2 dt, \quad f \in \mathbb{S}_{m,T} \tag{4.21}$$

minimiert, das heißt

$$F(f^*) \leq F(f), \qquad \forall\ f \in \mathbb{S}_{m,T}.$$

Das Funktional $F(f)$ erweitert das Fehlerfunktional $E(f)$ aus dem vorherigen Abschnitt um das Glättefunktional $G(f)$. Der Parameter $\lambda \in [0,1]$ gewichtet beide Terme gegeneinander. Die Idee dahinter ist, dass der Smoothing–Spline zu einem λ nahe bei eins eher glatt verläuft, ein λ nahe null führt zu einer eng bei den Daten liegenden Approximation. Im Fall $\lambda = 0$ entspricht der Smoothing–Spline dem Least–Squares–Approximanten aus Abschnitt 4.2.2.

Die Berechnung eines Smoothing–Splines reduziert sich bei einer festen Wahl von T, λ und r auf das Lösen eines linearen Gleichungssystems, wie im Folgenden gezeigt wird. In diesem Fall repräsentiert der Vektor der Kontrollpunkte $d \in \mathbb{R}^{1\times n}$ den Spline f, es gilt $f = \sum_{i=1}^{n} d_i N_i^m(\cdot|T) = d\ N^m(\cdot)$. Ist nichts anderes erwähnt, gilt in dieser Arbeit $r = 2$. Dann ist in der Anwendung von Abschnitt 4.3 die Funktion $f^{(r)}$ die Beschleunigung der Grobachse und $G(f)$ ein Mittelwert der Gesamtbeschleunigung.

Für den linken Summanden aus (4.21) folgt

$$\begin{aligned}
(1-\lambda)E(f) &= (1-\lambda)\sum_{j=1}^{n_t}(f(t_j)-x_j)^2 \\
&= (1-\lambda)\,||d\ \mathbf{N}_m(\mathbf{t}) - \mathbf{x}||_2^2 \\
&= (1-\lambda)\,(d\ \mathbf{N}_m(\mathbf{t}) - \mathbf{x})\ \ (d\ \mathbf{N}_m(\mathbf{t}) - \mathbf{x})^T \\
&= (1-\lambda)\left(d\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T d^T - 2x\mathbf{N}_m(\mathbf{t})^T d^T + xx^T\right)
\end{aligned}$$

[4] In der Literatur finden sich auch andere, äquivalente Darstellungen für (4.20), wie beispielsweise $E(f) + \rho G(f), \rho \in \mathbb{R}^+$.

$$=: (1-\lambda)E_d(d).$$

Die Matrix $\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T$ ist symmetrisch, positiv semidefinit und genau dann invertierbar, wenn $\mathbf{N}_m(\mathbf{t})$ vollen Rang besitzt. Hier kommt dasselbe Argument wie bei der Least–Squares–Approximation zum Tragen.

Für den rechten Summanden aus (4.21) gilt

$$\lambda G(f) = \lambda \int_{t_1}^{t_{n_t}} \left(f^{(r)}(t)\right)^2 dt = \lambda\, d\, H d^T =: \lambda\, G_d(d), \tag{4.22}$$

wobei H eine symmetrische Matrix ist. Sie setzt sich aus zwei Bestandteilen zusammen. Die Ableitung eines Splines vom Grad m ist wieder ein Spline, allerdings vom Grad $m-1$. Mit (4.1) und einer Knotenfolge $T = \{\zeta_1, \ldots, \zeta_{n+m+1}\}$ mit $m+1$–fachen Randknoten folgt für die Kontrollpunkte $d^{(1)}$ der ersten Ableitung

$$d^{(1)} = d \cdot \begin{bmatrix} \frac{-m}{\delta\zeta_2} & 0 & & & & & & \\ \frac{m}{\delta\zeta_2} & \frac{-m}{\delta\zeta_3} & 0 & & & & & \\ 0 & \frac{m}{\delta\zeta_3} & \frac{-m}{\delta\zeta_4} & 0 & & & & \\ & \ddots & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & 0 & \frac{m}{\delta\zeta_{n-2}} & \frac{-m}{\delta\zeta_{n-1}} & 0 \\ & & & & & 0 & \frac{m}{\delta\zeta_{n-1}} & \frac{-m}{\delta\zeta_n} \\ & & & & & & 0 & \frac{m}{\delta\zeta_n} \end{bmatrix} =: d\, \underbrace{D(T,m)}_{\in\mathbb{R}^{n\times n-1}},$$

wobei $\delta\zeta_i = \zeta_{i+m} - \zeta_i$. Für die r–te Ableitung bildet man r solcher Matrizen $D(T^0, m)$ bis $D(T^{r-1}, m-r+1)$ und damit das Produkt

$$D^{(r)} := D(T^0, m) \cdot \ldots \cdot D(T^{r-1}, m-r+1) \ \in \mathbb{R}^{n\times n-r}, \tag{4.23}$$

wobei $T^0 = T$ und T^{k+1} den zweiten bis den vorletzten Knoten aus T^k enthält. Damit berechnen sich die Kontrollpunkte der Ableitung durch

$$d^{(r)} = dD^{(r)} \in \mathbb{R}^{1\times n-r}.$$

Die Knotenfolge dieser Ableitung ist $T^{r-1} = (\zeta_{r+1}, \ldots, \zeta_{n+m+1-r})$. Insgesamt ist daher

$$f^{(r)} = d \cdot D^{(r)} \cdot N^{m-r}(\cdot|T^{r-1}).$$

Bei $(m+1)$–fachen Randknoten des Ursprungssplines $N_{m,T}d(\cdot)$ sind dieser und seine Ableitungen auf demselben Intervall definiert.

Für das Integral (4.22) gilt daher

$$\begin{aligned}\lambda \int_{t_1}^{t_{n_t}} \left(f^{(r)}(t)\right)^2 dt &= \lambda \int_{t_1}^{t_{n_t}} \left(dD^{(r)}N^{m-r}(t)\right)\left(dD^{(r)}N^{m-r}(t)\right)^T dt \\ &= \lambda d \underbrace{D^{(r)}H_1\left(D^{(r)}\right)^T}_{=:H} d^T\end{aligned}$$

mit

$$H_1 = \left[\int_{t_1}^{t_{n_t}} N_i^{m-r}(t)N_j^{m-r}(t)\,dt,\ i,j = 1,\ldots,n-r\right] \in \mathbb{R}^{n-r\times n-r}.$$

Die Matrix H_1 nennt man auch *Gram–Matrix*. Da die Integranden Produkte von B–Splines sind, können die Integrale stückweise mit Gausschen Quadraturformeln exakt ausgerechnet werden. Da das Integral in (4.22) für keinen Spline f einen negativen Wert besitzt, ist H positiv semidefinit. Mit einem gleichen Argument folgt auch die positive Semidefinitheit der Gram–Matrix H_1. Denn jeder Spline $f \in \mathbb{S}_{m,T}$ mit $f^{(r)}(t) = 0$, $t \in [t_1, t_{n_t}]$, minimiert $G(f)$. Im Fall $m = 3$ und $r = 2$ sind beispielsweise alle Polynome vom Grad eins die minimierenden Splines. Insbesondere ist H nicht invertierbar.

Formel (4.21) entspricht einer Funktion

$$\begin{aligned}F(d,T,\lambda,r) &= (1-\lambda)\sum_{i=1}^{n_t}(f(t_i)-x_i)^2 + \lambda\int_{t_1}^{t_{n_t}}\left(f^{(r)}\right)^2 dt \\ &= (1-\lambda)\left(d\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T d^T - 2\mathbf{x}\mathbf{N}_m(\mathbf{t})^T d^T + \mathbf{x}\mathbf{x}^T\right) + \lambda\, d\, H d^T \\ &= d\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right) d^T - 2(1-\lambda)\mathbf{x}\mathbf{N}_m(\mathbf{t})^T d^T + (1-\lambda)\mathbf{x}\mathbf{x}^T.\end{aligned}$$

Sind T, $\lambda \in [0,1)$ und r fest, wird daraus eine quadratische Funktion

$$F_d(d) = d\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right) d^T - (1-\lambda)\left(2\mathbf{x}\mathbf{N}_m(\mathbf{t})^T d^T - \mathbf{x}\mathbf{x}^T\right) \tag{4.24}$$

über den Kontrollpunkten d des Splines. Für eine universellere Handhabung der Smoothing–Splines, die auch die Einhaltung von Randbedingungen ermöglicht, soll auch hier die Darstellung $d = \tilde{d}I + v$ aus (4.7) genutzt werden. Aus (4.24) ergibt sich mit

$$\begin{aligned}\widetilde{F}(\tilde{d}) =& (1-\lambda)\underbrace{\left(\left[\tilde{d}I+v\right]\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})\left[\tilde{d}I+v\right]^T - 2\mathbf{x}\mathbf{N}_m(\mathbf{t})^T\left[\tilde{d}I+v\right]^T - \mathbf{x}\mathbf{x}^T\right)}_{=:\widetilde{E}(\tilde{d})} \\ &+ \lambda\underbrace{\left[\tilde{d}I+v\right]H\left[\tilde{d}I+v\right]^T}_{=:\widetilde{G}(\tilde{d})}\end{aligned} \tag{4.25}$$

eine quadratische Funktion mit $\tilde{d}$ als Variablenvektor. Mit weiteren Umformungen folgt

$$\begin{aligned}
\widetilde{F}(\tilde{d}) &= \left[\tilde{d}I + v\right]\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right)\left[\tilde{d}I + v\right]^T \\
&\quad - (1-\lambda)\left(2x\mathbf{N}_m(\mathbf{t})^T\left[\tilde{d}I + v\right]^T + \mathbf{x}\mathbf{x}^T\right) \\
&= \tilde{d}I\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right)I^T\tilde{d}^T + 2v\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right)I^T\tilde{d}^T \\
&\quad + v\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right)v^T \\
&\quad - (1-\lambda)\left(2\mathbf{x}\mathbf{N}_m(\mathbf{t})^T I^T\tilde{d}^T - 2\mathbf{x}\mathbf{N}_m(\mathbf{t})^T v^T + \mathbf{x}\mathbf{x}^T\right) \\
&= \tilde{d}\underbrace{\left[I\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right)I^T\right]}_{=:B_1}\tilde{d}^T \\
&\quad + \underbrace{\left[\left(2v\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right) - 2(1-\lambda)\mathbf{x}\mathbf{N}_m(\mathbf{t})^T\right)I^T\right]}_{=:b_1}\tilde{d} \\
&\quad + v\left((1-\lambda)\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T + \lambda H\right)v^T - 2(1-\lambda)\mathbf{x}\mathbf{N}_m(\mathbf{t})^T v^T + (1-\lambda)\mathbf{x}\mathbf{x}^T.
\end{aligned}$$

Eine notwendige Bedingung für ein Minimum dieser Funktion ist das Verschwinden des Gradienten,

$$2\tilde{d}B_1 + b_1 = 0. \tag{4.26}$$

Das ist sogar hinreichend, falls B_1 positiv definit ist. Da in unserem Fall mit den Randbedingungen aus Abschnitt 4.2.3 sowohl die Matrix $I\mathbf{N}_m(\mathbf{t})\mathbf{N}_m(\mathbf{t})^T I^T$ als auch IHI^T positv definit sind, ist (4.26) immer eindeutig invertierbar. Im Fall $\lambda = 1$ ist das Ergebnis das eindeutige Polynom vom Grad zwei, das die Randbedingungen erfüllt, auch wenn der Splinegrad m für die Approximation größer ist[5]. Somit lässt sich eine stetige Funktion

$$[0,1] \ni \lambda \longmapsto \tilde{d}(\lambda) \in \mathbb{R}^{1\times\tilde{n}}$$

definieren, die jedem $\lambda \in [0,1]$ die (eindeutige) Lösung von (4.26) zuweist und damit ergibt sich

$$d(\lambda) = \tilde{d}(\lambda)I + v, \qquad \lambda \in [0,1].$$

Der maximale Fehler des Approximanten zu den Daten,

$$||d(\lambda)\mathbf{N}_m(\mathbf{t}) - \mathbf{x}||_\infty,$$

ist eine stetige Funktion über λ. Sie muss im Gegensatz zu $E(\tilde{d}(\lambda))$ aus (4.25) aber nicht monoton steigend sein. Damit der Smoothing–Spline zu Daten der Form (4.3) die To-

[5] Bei anderen Randbedingungen kann IHI^T singulär sein.

leranzbedingung (4.4) erfüllt und gleichzeitig möglichst glatt ist, müssen die Knotenfolge T und der Glätteparameter λ passend gewählt werden. Die Knotenfolge liefert ein Least–Squares–Approximant, der die Toleranzbedingung und die Randbedingungen erfüllt. Außerdem soll die dazugehörige Matrix $I\mathbf{N}_m$ vollen Rang besitzen. Das garantiert, wie oben schon erwähnt, die Eindeutigkeit des Smoothing–Splines in Abhängigkeit von λ. Da $G_d(d(\lambda))$ monoton fallend ist ([7]), benötigen wir für die größtmögliche Glattheit noch das unter der Einhaltung der Toleranzbedingung maximale $\lambda \in [0,1]$. Es ist die größte Nullstelle der stetigen, aber nicht streng monoton steigenden Funktion

$$E_\infty : [0,1] \ni \lambda \longmapsto ||d(\lambda)\mathbf{N}_m(\mathbf{t}) - \mathbf{x}||_\infty - \varepsilon \in \mathbb{R}.$$

Im Fall eines negativen Funktionswertes für $\lambda = 1$ ist eins auch der gesuchte Wert für den Glätteparameter. Andernfalls liefert zum Beispiel die Regula Falsi mit den Startwerten $\lambda_0 = 0$ und $\lambda_1 = 1$ eine Nullstelle, sie muss aber nicht die größte sein.

4.2.7. Glatte Splineapproximation durch Lösen quadratischer Programme

Neben dem Verfahren aus dem vorherigen Abschnitt gibt es zwei weitere Methoden, um glatte Splineapproximationen zu berechnen. Sie lösen bestimmte Optimierungsaufgaben, in diesem Abschnitt handelt es sich um quadratische, im folgenden um lineare Programme.

Wie im vorherigen Abschnitt 4.2.6 betrachten wir bei festen Knoten für den Approximanten die quadratische Funktion über den Kontrollpunkten $\tilde{d}$,

$$\begin{aligned} \widetilde{G}(\tilde{d}) &= \left[\tilde{d}I + v\right] H \left[\tilde{d}I + v\right]^T \\ &= \tilde{d}IHI^T\tilde{d}^T + 2vHI^T\tilde{d}^T + vHv^T. \end{aligned} \tag{4.27}$$

Die Darstellung umfasst wieder die Randbedingungen aus Abschnitt 4.2.3. Aus (4.4) ergeben sich mit

$$\left|\left[\tilde{d}I + v\right]\mathbf{N}_m(t_i) - x_i\right| \le \varepsilon \quad \Leftrightarrow \quad -\varepsilon \le \tilde{d}I\mathbf{N}_m(t_i) + v\mathbf{N}_m(t_i) - x_i \le \varepsilon, \quad i = 1, \ldots, n_t,$$

die linearen Nebenbedingungen

$$\begin{aligned} \tilde{d}I\mathbf{N}_m(t_i) &\le \varepsilon - v\mathbf{N}_m(t_i) + x_i, \\ -\tilde{d}I\mathbf{N}_m(t_i) &\le \varepsilon + v\mathbf{N}_m(t_i) - x_i, \qquad i = 1, \ldots n_t. \end{aligned} \tag{4.28}$$

Zusammmen mit der zu minimierenden Zielfunktion $\widetilde{G}(\tilde{d})$ aus (4.27) entsteht ein quadratisches Minimierungsproblem mit linearen Nebenbedingungen, kurz quadratisches Programm. Die Lösung sind die Kontrollpunkte $\tilde{d}$ und durch $d = \tilde{d}I + v$ erhält man die Kontrollpunkte des gesuchten Splineapproximanten.

Im Vergleich zum Verfahren aus dem vorherigen Abschnitt ist hier das Glattheitsfunktional (4.27) das einzige Gütekriterium für den Approximanten. Die Lösung des Optimierungsproblems entspricht demjenigen Spline aus $\mathbb{S}_{m,T}$, der das Funktional unter Einhaltung der Toleranzbedingung minimiert. Die Ergebnisse sind bei derselben zugrundeliegenden Knotenfolge bezüglich der Glattheit mindestens genauso gut wie die mit dem Verfahren des vorherigen Abschnitts, meistens sogar besser.

Zur Lösung quadratischer Programme existieren leistungsfähige Methoden und Implementierungen, auf die in dieser Arbeit aber nicht weiter eingegangen werden soll. Details liefert beispielsweise [25]. Die Algorithmen erkennen insbesondere den Fall, dass die Nebenbedingungen keinen gültigen Punkt erlauben. Letzteres kann vorkommen, wenn die Knotenfolge T unpassend gewählt ist. Deswegen ist es sinnvoll, sie von einem Least–Squares–, einem Tschebyschow– oder einem Schoenberg–Approximanten, der jeweils (4.4) erfüllt, zu nehmen. Damit ist die Existenz eines (4.28) erfüllenden Vektors garantiert und die Lösungsmethode kann diesen sogar als Startwert nutzen.

Mit diesem Ansatz ist es außerdem möglich, Restriktionen für Ableitungen des Approximanten als lineare Nebenbedingungen zu formulieren. Wir gehen davon aus, dass der Betrag der k–ten Ableitung einen Wert von $\widehat{W}_k$ nicht überschreiten darf. In dieser Arbeit sind natürlich nur die Fälle $k = 1, 2, 3$ für Geschwindigkeit, Beschleunigung und Ruck relevant. Jede Ableitungsrestriktion kann auf zwei Arten als Satz linearer Nebenbedingungen formuliert werden. Die erste enthält für jeden Punkt t_i zwei lineare Ungleichungsnebenbedingungen. Mit der Matrix $D^{(k)}$ aus (4.23) ergibt sich[6]

$$\left|\left[\tilde{d}I + v\right]\ D^{(k)}\ \mathbf{N}_{m-k,T^{k-1}}(\mathbf{t})\right| \leq \widehat{W}_k\ \mathbf{1}_{n_t}^T$$

und daraus

$$\begin{aligned} \tilde{d}I\ D^{(k)}\ \mathbf{N}_{m-k,T^{k-1}}(\mathbf{t}) &\leq \widehat{W}_k\ \mathbf{1}_{n_t}^T - vD^{(k)}\mathbf{N}_{m-k,T^{k-1}}(\mathbf{t}), \\ -\tilde{d}I\ D^{(k)}\ \mathbf{N}_{m-k,T^{k-1}}(\mathbf{t}) &\leq \widehat{W}_k\ \mathbf{1}_{n_t}^T + vD^{(k)}\mathbf{N}_{m-k,T^{k-1}}(\mathbf{t}), \end{aligned} \tag{4.29}$$

als Nebenbedingungen für den Variablenvektor $\tilde{d}$. Die zweite Variante nutzt die Eigenschaft der konvexen Hülle aus Proposition 4.5 und die Ungleichung wird für die Kontrollpunkte der Ableitung verlangt. Das ergibt die Bedingungen

$$\left|\left[\tilde{d}I + v\right] D^{(k)}\right| \leq \widehat{W}_k\ \mathbf{1}_{n-k}^T$$

oder in anderer Schreibweise

$$\begin{aligned} \tilde{d}I\ D^{(k)} &\leq \widehat{W}_k\ \mathbf{1}_{n-k}^T - vD^{(k)}, \\ -\tilde{d}I\ D^{(k)} &\leq \widehat{W}_k\ \mathbf{1}_{n-k}^T + vD^{(k)}. \end{aligned} \tag{4.30}$$

[6] Das "$\leq$"–Zeichen bedeutet hier den komponentenweisen Vergleich.

Erfüllt ein $\tilde{d}$ die Bedingung (4.30), so auch die Ungleichungen (4.29). Die Umkehrung gilt im Allgemeinen nicht; darin besteht der Unterschied der beiden Varianten. Im ungünstigen Fall erklärt (4.30) einen Spline für ungültig, obwohl er (4.29) erfüllt. Andererseits enthält (4.30) deutlich weniger Nebenbedingungen ($2n$ zu $2n_t$), da (zumindest in unseren Anwendungsfällen) $n \ll n_t$ gilt. Besonderheiten stellen die Fälle $m-k=1$ und $m-k=0$ dar. Im ersten ist die Ableitung eine stückweise lineare Funktion und die jeweils (4.29) und (4.30) einhaltenden (Spline)Mengen sind gleich. Gleiches gilt für den zweiten Fall, wohingegen die k–te Ableitung stückweise konstant ist. Der erste Fall tritt beispielsweise bei $m=3$ und Beschleunigungsrestriktionen auf.

Verwendet die Zerlegung dieses Verfahren mit den Nebenbedingungen für die Ableitungen, kann es natürlich sein, dass die durch die Nebenbedingungen definierte Menge leer ist und das Optimierungsproblem keine Lösung besitzt. In diesen Fällen führt die Zerlegung die Optimierung noch einmal durch, diesmal aber lediglich mit den Nebenbedingungen für die Positionen. Dann hat das quadratische Programm immer eine Lösung.

4.2.8. Glatte Approximationen durch Lösen linearer Programme

Das letzte Verfahren, das neben der reinen Approximationsgüte auch ein Gütemaß für die Glattheit minimiert, löst ein lineares Programm. Das Maß für die Glattheit unterscheidet sich von denen der beiden vorherigen Abschnitte. Hier soll das Maximum des Absolutbetrags der r–ten Ableitung so klein wie möglich sein und natürlich die Bedingungen (4.28) erfüllt werden. Gesucht ist also

$$\begin{aligned} \underset{\tilde{d}\in G_1}{\operatorname{argmin}} \left\{ \max_i \left| \left[\tilde{d}I + v\right] D^{(r)} \mathbf{N}_{m-r,T^{r-1}}(t_i) \right| \right\}, \\ G_1 = \left\{ \tilde{d} \in \mathbb{R}^{\tilde{n}} \mid \tilde{d} \text{ erfüllt (4.28)} \right\}. \end{aligned}$$

Diese Aufgabenstellung lässt sich als Optimierungsproblem mit einer linearen Zielfunktion und linearen Nebenbedingungen, kurz lineares Programm, formulieren. Mit einer zusätzlichen Variablen $w \in \mathbb{R}$ ist die lineare Zielfunktion

$$F : [\tilde{d}, w] \mapsto F([\tilde{d}, w]) = w. \tag{4.31}$$

Die linearen Nebenbedingungen sind zum einen

$$\left| \left[\tilde{d}I + v\right] \; D^{(r)} \; \mathbf{N}_{m-r,T^{r-1}} \right| \leq w \cdot \mathbf{1}_{n_t}$$

oder anders geschrieben

$$\begin{aligned} \tilde{d}I \; D^{(r)} \; \mathbf{N}_{m-r,T^{r-1}} &\leq w \cdot \mathbf{1}_{n_t} - vD^{(r)} \; \mathbf{N}_{m-r,T^{r-1}}, \\ -\tilde{d}I \; D^{(r)} \; \mathbf{N}_{m-r,T^{r-1}} &\leq w \cdot \mathbf{1}_{n_t} + vD^{(r)} \; \mathbf{N}_{m-r,T^{r-1}} \end{aligned} \tag{4.32}$$

und zum anderen (4.28). Die Variable w entspricht dabei einer oberen Grenze des Betrags der r–ten Ableitung. Die Minimierung von (4.31) bewirkt dann gerade, dass eine Lösung dieses linearen Programms einer Approximation mit der betragsmäßig kleinsten maximalen Ableitung entspricht.

Ebenso wie im vorherigen Abschnitt bietet sich eine Vereinfachung von (4.32) an, indem der betragsmäßig größte Kontrollpunkt der r–ten Ableitung des Approximanten minimiert wird. Die Bedingungen

$$\begin{aligned} \tilde{d}I\ D^{(r)} &\leq w \cdot \mathbf{1}_{\tilde{n}} - vD^{(r)}, \\ -\tilde{d}I\ D^{(r)} &\leq w \cdot \mathbf{1}_{\tilde{n}} + vD^{(r)} \end{aligned}$$

ersetzen in diesem Fall (4.32). Die Unterschiede der beiden Varianten aus dem vorherigen Abschnitt gelten hier ebenfalls.

Natürlich kann man auch hier wie im vorherigen Abschnitt Nebenbedingunen $\widehat{W}_k$ für die k–te Ableitung integrieren. Eine Übersicht zu Lösungsverfahren für lineare Programme liefert auch hier [25].

4.3. Vollständiges Verfahren und Anwendungsbeispiele

Abschnitt 4.1 und Abbildung 4.1 stellen das Vorgehen in dem neuen Ansatz vor. Im Wesentlichen sind es zwei Stellen, bei denen passende Methoden das Funktionieren des gesamten Vorgehens sichern müssen. Zum einen braucht man Zerlegungsverfahren. Sie verwenden hier Splineapproximation auf Grundlage von Abschnitt 4.2 und werden in 4.3.1 genauer erläutert. Zum anderen benötigt die Anpassung der Einstellungen in der Bewegungsplanung Regeln, die aus der Art und Größe der Verletzung im Überprüfungsblock die Modifikation bestimmen. Diesen Punkt behandelt Abschnitt 4.3.2.

Die Anwendung des Verfahrens auf ganze Konturen beschreibt Abschnitt 4.3.3. Die in der realen Steuerung mögliche, abschnittsweise Variante des Ansatzes folgt im darauffolgenden Abschnitt 4.3.4. Danach wird in Abschnitt 4.3.5 diskutiert, wie sich eine Ausdünnung der Daten in der Zerlegung auswirkt. Den Abschluss bildet ein Beispiel eines fertigen Werkstücks.

4.3.1. Bewegungsaufteilung im Zeitbereich

Für die Aufteilung der Bewegung kann bei Kinematik A jede der Richtungskomponenten x und y separat betrachtet werden. Wir beschränken uns daher in diesem Abschnitt auf die Darstellung der x–Komponente. Der Zerlegungsblock aus Abbildung 4.1 hat als Eingang die Zeiten und Positionen der Referenztrajektorie,

$$(t_1, \ldots, t_{n_t}) \qquad \text{und} \qquad (x_1, \ldots, x_{n_t}) = (x(t_1), \ldots, x(t_{n_t})). \tag{4.33}$$

Die Ausgangswerte sind die Achspositionen der Fein- und Grobachse,

$$x_c(t_i) \quad \text{und} \quad x_f(t_i), \qquad i = 1, \ldots, n_t. \tag{4.34}$$

Die Bewegung der Grobachse entsteht, indem ein Spline die Referenzwerte (4.33) approximiert, die Feinachse ergibt sich dann direkt durch Differenzbildung. Die wesentliche Arbeit bei der Zerlegung ist also die Splineapproximation. Dafür sollen im Folgenden Verfahren vorgestellt werden, die auf den Approximationsmethoden aus Abschnitt 4.2 basieren und deren Ergebnis ein auf dem Zeitintervall $[t_1, t_{n_t}]$ definierter Spline $x_c(t)$ ist, der

$$|x(t_i) - x_c(t_i)| \leq \widehat{S}_{xf}, \qquad i = 1, \ldots, n_t, \tag{4.35}$$

erfüllt und Randbedingungen genügt. Letztere sind die in Abschnitt 4.2.3 erwähnten, mit der Positionsvorgabe P_1 bei t_1 und den Geschwindigkeitsvorgaben V_1 beziehungsweise V_{n_t} bei t_1 und t_{n_t}. Die Feinachse bewegt sich dann wegen (4.35) sicher in ihrem Verfahrbereich $[-\widehat{S}_{xf}, \widehat{S}_{xf}]$.

Der Zerlegungsblock besteht aus zwei Phasen (Abbildung 4.2). Für jede kann ein Approximationsverfahren gewählt werden. In der ersten Phase sind das die Least–Squares–Approximation aus Abschnitt 4.2.2, die Tschebyschow–Approximation aus Abschnitt 4.2.4 oder die Variante der Approximation mit dem Schoenbergschen Splineoperator aus Abschnitt 4.2.5. In allen drei Fällen bestimmt ein iteratives Vorgehen eine Knotenfolge, zu der der jeweilige Approximant (4.35) erfüllt[7]. Die Details dazu folgen etwas weiter unten. Die zweite Phase ist optional und nutzt die Knotenfolge des Ergebnisses der ersten. Mit ihr kann eines der drei Verfahren aus den Abschnitten 4.2.6, 4.2.7 oder 4.2.8 einen (4.35) erfüllenden Approximanten berechnen, der meistens glatter als das Ergebnis der ersten Phase verläuft. Lediglich die Kombination Tschebyschow– oder Schoenberg–Approximation und Smoothing–Splines funktioniert nicht. Es können dabei Situationen auftreten, in denen der Smoothing–Spline mit der Knotenfolge der ersten Phase nicht die Toleranzbedingung erfüllt, unabhängig von der Wahl des Glätteparameters λ.

An dieser Stelle kommt die Frage auf, warum der Approximant als Bewegung für die Grobachse möglichst glatt sein soll, worauf ja besonders die zweite Phase abzielt. Das hängt zum einen damit zusammen, dass starke und häufig vorkommende Beschleunigungswechsel der Hauptachse zu einem Qualitätsverlust am Werkstück führen können, wie das bei dem Ansatz mit der geometrischen Zerlegung vorkommen kann (vergleiche Abschnitt 3.2). Zum anderen soll das Ergebnis einer Zerlegung den Überprüfungsblock passieren. Eine glatte Bewegung der Grobachse bedeutet in gewissem Sinne, dass die Beschleunigungswerte niedrig sind und damit die Grobbewegung eher ihre Restriktionen erfüllt. Besonders die Nebenbedingungen in dem zweiten und dritten Verfahren aus Abbildung 4.2 zielen darauf ab.

[7] Man beachte dabei die Möglichkeit des Effektes, der am Ende von Abschnitt 4.2.5 beschrieben ist.

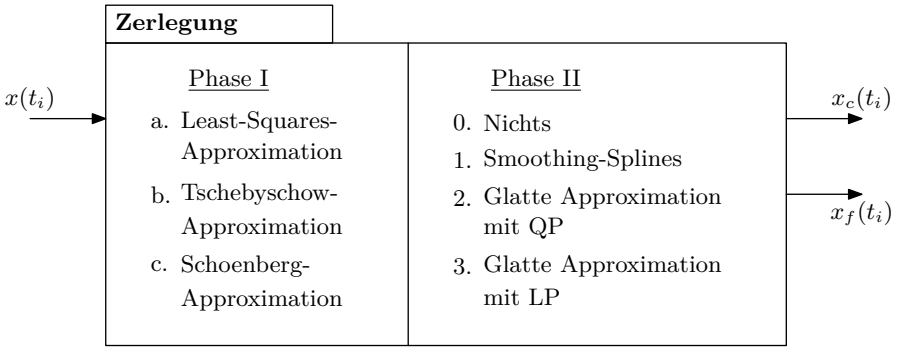

Abbildung 4.2.: Aufbau und Möglichkeiten der Zerlegung.

Knoteneinfügen bei Least–Squares–Approximation

Kommen wir nun zu der Strategie für das Einfügen neuer Knoten in der iterativen Least–Squares–Approximation. Sie wird solange angewendet, bis der Least–Squares–Approximant die Bedingung (4.35) erfüllt. Wir gehen davon aus, dass in einer Iteration der Approximant mit der Knotenfolge T_{akt} berechnet wurde und der maximale Fehler zwischen Daten und Approximant an der Stelle $t_{i_{\text{err}}}$, $i_{\text{err}} \in \{1, \ldots, n_t\}$, liegt. Außerdem bezeichnet

$$\widetilde{T}_{\max} := (\alpha_1, \ldots, \alpha_{n_\alpha}) \tag{4.36}$$

die größtmögliche Teilfolge von $T_{\max}$, deren Elemente (als reelle Zahlen) paarweise verschieden sind. Weiter ist $\{\kappa_1, \ldots, \kappa_{n_\alpha}\}$ eine Menge von Werten, mit $\kappa_k = 1$, falls α_k ein Knoten von T_{akt} ist, und $\kappa_k = 0$ andernfalls. Es gibt einen Index $j \in \{1, \ldots, n_\alpha\}$, sodass $|t_{i_{\text{err}}} - \alpha_j|$ minimal ist[8]. Dieser Index j gehört zu dem Knoten aus $\widetilde{T}_{\max}$, der am dichtesten an der Stelle mit dem maximalen Fehler liegt. Außerdem nutzen wir für einen beliebigen Index $k \in \{1, \ldots, n_\alpha\}$ mit $\kappa_k = 0$ die folgende Bezeichnung für eine Indexmenge:

$$I_k := \{\, \eta \in \{1, \ldots, n_\alpha\} \mid \forall i \text{ mit } \min\{k, \eta\} \leq i \leq \max\{k, \eta\} \text{ gilt: } \kappa_i = 0 \,\}\,.$$

Sie enthält die Indices der noch nicht in T_{akt} enthaltenen Knoten aus $\widetilde{T}_{\max}$, die zwischen zwei benachbarten Knoten aus T_{akt} um α_k liegen.

Es folgt eine Fallunterscheidung für das Einfügen eines neuen Knotens. Ist $\kappa_j = 0$, so betrachten wir $\#I_j$. Ist dieser Wert größer als eine vorab zu definierende Konstante $K^* \in \mathbb{N}$, so nehmen wir das mittlere[9] Element j^* aus I_j und α_{j^*} ist der neue Knoten. Andernfalls ergeben sich aus

$$\begin{aligned} j_l &= \max\{k \in \{1, \ldots, j\} \mid (\kappa_k = 0) \wedge (\forall i \in I_j : k < i)\}\,, \\ j_r &= \min\{k \in \{j, \ldots, n_\alpha\} \mid (\kappa_k = 0) \wedge (\forall i \in I_j : i < k)\}\,, \end{aligned}$$

[8] Dieser Index kann zweideutig sein, in diesem Fall muss man einen wählen.
[9] Ist $\#I_j$ gerade, gibt es zwei mittlere Elemente und das Vorgehen muss sich für eines entscheiden.

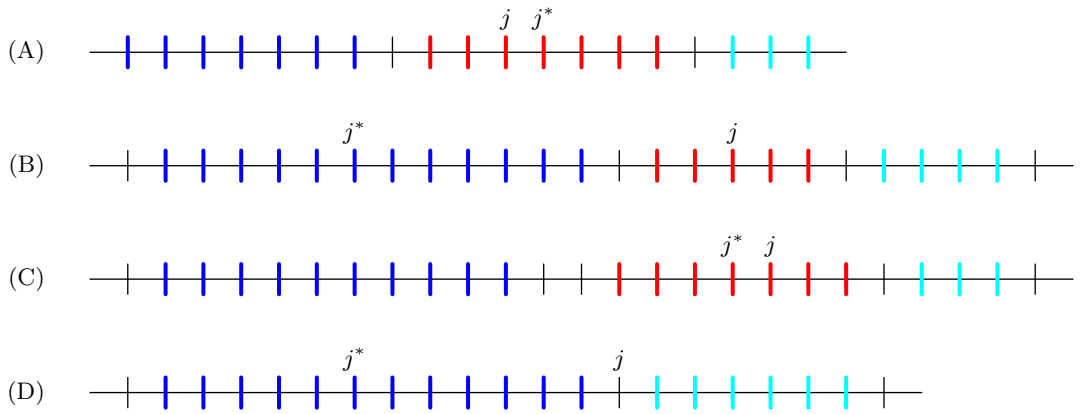

Abbildung 4.3.: Vier Beispiele für das Einfügen eines neuen Knotens mit $K^* = 6$. Die Striche kennzeichnen die Knoten aus $T_{\max}$ und die Indexmengen I_j (rot), I_{j_l} (blau) und I_{j_r} (cyan).

die Indexmengen I_{j_l} und I_{j_r}. Daraus definieren wir

$$\widetilde{I}_{j_l} := \begin{cases} I_{j_l}, & j_l + 2 = \min\{I_j\}, \\ \emptyset, & \text{sonst.}, \end{cases} \tag{4.37}$$

und

$$\widetilde{I}_{j_r} := \begin{cases} I_{j_r}, & j_r - 2 = \max\{I_j\}, \\ \emptyset, & \text{sonst.} \end{cases} \tag{4.38}$$

Der Index j^* des neuen Knotens ist die Mitte der größten der drei Indexmengen I_j, $\widetilde{I}_{j_l}$ und $\widetilde{I}_{j_r}$. Die Fallunterscheidung (4.37) bewirkt, dass die Menge I_{j_l} nur dann für die Wahl des neuen Knotens hinzugezogen wird, wenn sie nur durch einen schon aktiven Knoten von I_j getrennt ist. Andernfalls hätte die Mitte von I_{j_l} wegen der Lokalität der Splines wenig Einfluss auf die Reduzierung des Fehlers bei $t_{i_{\text{err}}}$. Gleichermaßen verhält es sich mit (4.38). Abbildung 4.3 zeigt drei Beispiele ((A) bis (C)) für die Wahl von j^* in den gerade beschriebenen Situationen.

Kommen wir zum Fall $\kappa_j = 1$, das heißt, α_j ist schon in T_{akt} enthalten. Zuerst definieren wir die Werte

$$j_l = \max\{k \in \{1, \ldots, j\} \mid \kappa_k = 0\},$$
$$j_r = \min\{k \in \{j, \ldots, n_\alpha\} \mid \kappa_k = 0\}.$$

Daraus ergeben sich Indexmenge I_{j_l} und I_{j_r}, von denen eine leer sein kann. Sie enthalten die Indices, der noch nicht belegten, zusammenhängenden Knoten, die am nähesten links

beziehungsweise rechts von α_j liegen. Die Mitte der größeren der beiden Indexmengen ist der Index j^* des neuen Knotens. Das Beispiel (D) in Abbildung 4.3 verdeutlicht das.

In Abschnitt 4.2 wurde festgelegt, dass die inneren Knoten des Approximanten immer dieselbe Vielfachheit besitzen sollen. Deswegen wird in T_{akt} der neue Knoten α_{j^*} mit entsprechender Vielfachheit eingefügt. Außerdem braucht die erste Iteration eine Knotenfolge T_{akt}. Falls nicht anders erwähnt, besteht sie nur aus den $m+1$–fachen Randknoten.

In den typischen Anwendungsfällen dieser Arbeit braucht ein Approximant, der die Toleranzbedingung erfüllt, deutlich weniger Knoten, als ihm in $T_{\max}$ zur Verfügung stehen. Das liegt an dem verhältnismäßig großen Verfahrbereich und der dicht abgetasteten Referenztrajektorie. Die gerade beschriebene Strategie zum Erstellen der Knotenfolge führt dazu, dass es nur in Extremfällen zu vielen dicht beieinanderliegenden Knoten kommt. Das liegt hauptsächlich daran, dass ein neuer Knoten immer mittig eingefügt wird und dass es den Sonderfall mit K^* gibt. Dieses Vorgehen führt auch zu glatteren Verläufen des Approximanten als wenn sich Knoten konzentrieren. Wie oben schon erwähnt ist das in unserer Anwendung gewollt. Die weiter unten beschriebenen Beispiele zeigen, dass das Vorgehen zufriedenstellende Ergebnisse liefert.

Das hier beschriebene iterative Verfahren endet, wenn der Approximant die Toleranzbedingung erfüllt. Das ist nach einer endlichen Anzahl von Iterationen der Fall, spätestens wenn der Approximant die Daten interpoliert. Die zweite Phase nutzt die so gewonnene Knotenfolge. Bei der glatten Approximation mit quadratischen oder linearen Programmen erhöht eine erweiterte Knotenfolge die Chance, dass der Approximant eine gültige Bewegung der Grobachse ist. Deswegen soll es die Möglichkeit eines „Nachbrenners“ in Phase eins geben. Dabei werden, nachdem die Toleranzbedingung schon erfüllt ist, weitere Iterationen durchgeführt. Nicht jede Wiederholung verkleinert den maximalen Fehler. Deswegen laufen die Iterationen so lange, bis der Fehler zum N^*–ten Mal unter der Grenze $\widehat{S}_{xf}$ liegt.

Verfeinerung der Knotenfolge bei der Schoenberg–Approximation

Das iterative Einfügen der Knoten geht hier genauso wie bei der Least–Squares–Approximation. Lediglich $T_{\max}$ unterscheidet sich. Es ist eine Knotenfolge, deren Knotenabstand im Fall $k_{\text{vf}} = 1$ konstant[10]

$$\frac{t_{n_t} - t_1}{\lceil (t_{n_t} - t_1)/\Delta\zeta \rceil}$$

ist, wobei das $\Delta\zeta$ aus (4.19) kommt. Bei $k_{\text{vf}} > 1$ ändern sich die inneren Knoten nicht, nur ihre Vielfachheit wird zu k_{vf}.

Mit dieser Knotenfolge erfüllt der Approximant die geforderte Toleranzbedingung. Es kommt allerdings nicht zur Interpolation wie im Fall der Least–Squares–Approximation. Trotzdem endet das Einfügen nach endlich vielen Iterationen. Möglicherweise kann das

[10] Die Randknoten sind natürlich weiterhin $(m+1)$–fach.

Austauschen der Kontrollpunkte am Rand Probeme mit der Toleranzbedingung machen (vergleiche mit Ende von Abschnitt 4.2.5), in den Anwendungsfällen traten aber keine auf.

Verfeinerung der Knotenfolge bei Tschebyschow–Approximation

Im Fall der Tschebyschow–Approximation hat sich die gerade beschriebene Strategie zum Finden einer Knotenfolge nicht als sinnvoll erwiesen. Das liegt daran, dass der Approximant an mehreren Stellen den gleichen, maximalen Fehler zu den Daten mit unterschiedlichem Vorzeichen hat, eine typische Eigenschaft der Tschebyschow–Approximation[11]. Es dauert deswegen zu viele Iterationen und die Knotenfolge besitzt dann reichlich Knoten, bis der Approximant die Bedingung (4.35) erfüllt. Deswegen soll hier ein anderes, einfaches Vorgehen genutzt werden.

Die Grundlage ist wieder die Knotenfolge $\widetilde{T}_{\max}$ aus (4.36). In der k-ten Iteration ergibt sich mit

$$\mu(j) := \left\lfloor 1 + \frac{n_\alpha - 1}{k}(j-1) + \frac{1}{2} \right\rfloor, \qquad j = 1, \ldots, k+1,$$

eine Teilfolge von $\widetilde{T}_{\max}$ aus (4.36) durch

$$\left(\alpha_{\mu(1)}, \ldots, \alpha_{\mu(k+1)}\right). \tag{4.39}$$

Diese Knoten sind relativ gleichmäßig verteilt, wenn $\mathbf{t}$ äquidistant ist. Mit ihnen setzt sich T_{akt} folgendermaßen zusammen. Die Randknoten sind $\alpha_{\mu(1)} = \alpha_1$ und $\alpha_{\mu(k+1)} = \alpha_{n_\alpha}$, jeweils mit Vielfachheit $m+1$. Die anderen bilden mit der gewählten Vielfachheit k_{vf} die inneren Knoten. In jeder Iteration wächst so die Knotenfolge für den Approximanten um k_{vf} Knoten. In der $(n_\alpha - 1)$-ten Iteration interpoliert der Approximant die Daten, da $T_{\text{akt}} = T_{\max}$. Das heißt insbesondere, dass das Vorgehen spätestens dann die Toleranzbedingung erfüllt. Da für alle vorherigen Iterationen $\frac{n_\alpha - 1}{k} > 1$ gilt, ist die Folge (4.39) streng monoton steigend und T_{akt} hat keine inneren Knoten mit unerwünscht hoher Vielfachheit.

Auch bei diesem Vorgehen ist ein Nachbrenner möglich. Allerdings brauchen die Iterationen nicht durchgeführt werden. Die feinere Knotenfolge ergibt sich direkt gemäß dem Vorgehen im vorherigen Absatz.

4.3.2. Modifikation der Einstellungen in der Bewegungsplanung

Dieser Abschnitt beschreibt die Regeln zur Anpassung der Einstellungen in der Bewegungsplanung. Sie greifen, wenn der Ausgang der Zerlegung den Überprüfungsblock nicht passiert, also Geschwindigkeits- oder Beschleunigungsrestriktionen der Achsen verletzt sind. Eine Regel für überschrittene Verfahrbereiche ist überflüssig, da dieser Fall mit den beschriebenen Zerlegungsmethoden nicht eintritt.

[11] Siehe Alternantensatz der Approximationstheorie.

Um in der Bewegungsplanung beim iterativen Vorgehen mehr Flexibilität zu haben, soll die Bewegung nicht für die gesamte Kontur mit konstanten Werten für die Beschleunigungs- und Geschwindigkeitsgrenze erzeugt werden. Vielmehr sollen im j–ten der n_h Sätze die Grenzen

$$\widehat{A}_x(j), \quad \widehat{V}_x(j), \quad \widehat{A}_y(j), \quad \widehat{V}_y(j), \quad \hat{v}_b(j)$$

gelten. Das Verfahren ist so ausgelegt, dass sich diese Werte in jeder Iterationen verkleinern oder gleich bleiben.

Ein Ausgang der Zerlegung besteht aus (4.34) für die x–Achse und den Analoga $y_c(t_i)$ und $y_f(t_i)$, $i = 1, \ldots, n_t$. Da man aus der Bewegungsplanung auch die Geschwindigkeits- und die Beschleunigungswerte der Referenz kennt und da man die Splines $x_c(t)$ und $y_c(t)$ ableiten und auswerten kann, stehen im Überprüfungsblock die Ableitungswerte der Einzelachsen zur Verfügung,

$$\begin{array}{llll} a_{xc}(t_i), & a_{xf}(t_i), & a_{yc}(t_i), & a_{yf}(t_i), \\ v_{xc}(t_i), & v_{xf}(t_i), & v_{yc}(t_i), & v_{yf}(t_i), \qquad i = 1, \ldots, n_t. \end{array}$$

Außerdem existiert eine Abbildung

$$\beta : \{0, \ldots, n_h\} \ni j \mapsto \beta(j) \in \mathbb{N},$$

sodass $t_{\beta(j)}$ der letzte Zeitpunkt im j–ten Satz ist. Mit dem Algorithmus aus Abschnitt 2.2.3 sind diese Werte bekannt. Liegt ein solcher Zeitpunkt genau auf einem Satzwechsel, soll er dem linken Satz zugeordnet werden. Zudem soll $\beta(0) = 0$ gelten. Damit beschreibt die Menge

$$U(j) = \{\beta(j-1) + 1, \ldots, \beta(j)\}$$

für jedes $j \in \{1, \ldots, n_h\}$ die Indices der Zeitpunkte des j-ten Satzes.

In der N–ten Iteration benötigen wir für die Regeln die folgenden Notationen:

$$\begin{aligned} \hat{a}_{xc}(j) &:= \max\{|a_{xc}(t_i)| \mid i \in U(j)\}, \\ \hat{a}_{xf}(j) &:= \max\{|a_{xf}(t_i)| \mid i \in U(j)\}, \\ \hat{v}_{xc}(j) &:= \max\{|v_{xc}(t_i)| \mid i \in U(j)\}, \\ \hat{v}_{xf}(j) &:= \max\{|v_{xf}(t_i)| \mid i \in U(j)\}. \end{aligned}$$

Die Regel zur Verkleinerung der Beschleunigungsrestriktion in der Bewegungsplanung braucht außerdem eine Konstante $k^* \in \mathbb{N}$ und dazu die Abbildung

$$\lambda_k : \{-k^*, \ldots, k^*\} \ni \iota \mapsto \lambda_k(\iota) \in [0, 1]$$

(1) $\widehat{A}_{x,\mathrm{alt}}(j) := \widehat{A}_x(j), \quad j = 1, \ldots, n_h$

(2) Für $j = 1, \ldots, n_h$:

(3) Falls $\hat{a}_{xc}(j) > \widehat{A}_{xc}$:

(4) $\Delta\hat{a}_{xc}(j) := \hat{a}_{xc}(j) - \widehat{A}_{xc}$

(5) Für $i = \max\{1, j - k^*\}, \ldots, \min\{j + k^*, n_h\}$:

(6) $\widehat{A}_{x,2}(i) := \min\left\{\widehat{A}_x(i), \widehat{A}_{x,\mathrm{alt}}(i) - \lambda_k(i-j)\ \Delta\hat{a}_{xc}(j)\right\}$

(7) $\widehat{A}_x(i) := \max\left\{\widehat{A}_{x,1}, \widehat{A}_{x,2}(i)\right\}$

(8) Ansonsten, falls $\hat{a}_{xf}(j) > \widehat{A}_{xf}$:

(9) $\Delta\hat{a}_{xf}(j) := \hat{a}_{xf}(j) - \widehat{A}_{xf}$

(10) $\widehat{A}_x(j) := \min\left\{\widehat{A}_x(j), \widehat{A}_{x,\mathrm{alt}}(j) - K_{xf}\ \Delta\hat{a}_{xf}(j)\right\}$

Algorithmus 4.1: Regel für das Anpassen der Beschleunigungsgrenze in der Bewegungsplanung

sowie die Zahl

$$\widehat{A}_{x,1} := \max\left\{\widehat{A}_{xc} - (N-1)\frac{\widehat{A}_{xc}}{20}, \frac{\widehat{A}_{xc}}{20}\right\}. \tag{4.40}$$

Algorithmus 4.1 zeigt die Modifikation von $\widehat{A}_x(j)$, $j = 1, \ldots, n_h$, in der N–ten Iteration. Derselbe muss natürlich auch für die y–Achse durchlaufen werden.

Dieses Vorgehen soll genauer erläutert werden. Die äußere Schleife durchläuft jeden Satz der Kontur. Verletzt die Grobachse ihre Beschleunigungsgrenze $\widehat{A}_{xc}$ im j–ten Satz, dann legen die Zeilen vier bis sieben die Veränderungen für $\widehat{A}_x(j - k^*)$ bis $\widehat{A}_x(j + k^*)$ fest[12]. Dabei soll der alte Wert für die Grenze um die maximale Verletzung multipliziert mit einem Faktor verkleinert werden. Dieser Faktor ist $\lambda_k(i - j)$. Die Idee dabei ist, dass er für $i = j$ den Wert eins hat und nach außen hin abnimmt. Für $k^* = 2$ könnte das Bild von λ_k beispielsweise $(0.4, 0.7, 1, 0.7, 0.4)$ sein, die kommenden Beispiele verwenden diese Werte. Damit wird die Beschleunigung an dem Satz der Verletzung stärker reduziert als an den benachbarten Sätzen. Die Minimumwahl in Zeile sechs stellt sicher, dass eine stärkere Verkleinerung in einer vorherigen Satziteration nicht zurückgenommen wird. Die Wahl in Zeile sieben ist gewissermaßen ein „Rettungsanker“, damit die Beschleunigungsgrenze für die Referenz nicht zu klein wird. Mit der Zahl 20 in (4.40) ist der Anker festgelegt. Dieser Wert 20 sorgt im extremen Fall für eine verhältnismäßig kleine Beschleunigung $\frac{\widehat{A}_{xc}}{20}$.

[12]Laufen diese Indices aus den Sätzen heraus, passiert dort natürlich nichts.

Verletzt nur die Bewegung der Feinachse ihre Restriktion im j–ten Satz, dann legt Zeile zehn die Reduzierung fest. Sie ist die Beschleunigungsverletzung der Feinachse multipliziert mit einem Faktor K_{xf}. (In den folgenden Beispielen gilt $K_{xf} = 1.15$.) In der Praxis erweisen sich Werte als sinnvoll, die etwas größer als eins sind. Auch hier bewirken die zwei Möglichkeiten in Zeile zehn, dass eventuell schon in einer vorherigen Satziteration vorgenommene, stärkere Verkleinerungen nicht überschrieben werden.

Kommen wir jetzt zur Modifikation von $\hat{v}_b(j)$. Mit

$$\begin{aligned}
\Delta\hat{v}_{xc}(j) &:= \max\left\{0,\ \hat{v}_{xc}(j) - \widehat{V}_{xc}\right\},\\
\Delta\hat{v}_{xf}(j) &:= \max\left\{0,\ \hat{v}_{xf}(j) - \widehat{V}_{xf}\right\},\\
\Delta\hat{v}_{yc}(j) &:= \max\left\{0,\ \hat{v}_{yc}(j) - \widehat{V}_{yc}\right\},\\
\Delta\hat{v}_{yf}(j) &:= \max\left\{0,\ \hat{v}_{yf}(j) - \widehat{V}_{yf}\right\}
\end{aligned} \tag{4.41}$$

und

$$\Delta v_b(j) := \max\left\{\Delta\hat{v}_{xc}(j),\ \Delta\hat{v}_{xf}(j),\ \Delta\hat{v}_{yc}(j)\ \Delta\hat{v}_{yf}(j)\right\}$$

folgt die Regel

$$\hat{v}_b(j) := \max\left\{\hat{v}_b(j) - 1.1\Delta v_b(j),\ \hat{v}_{b,\min}\right\}. \tag{4.42}$$

Da die Werte aus (4.41) (theoretisch) größer als $\hat{v}_b$ sein können, ist mit $\hat{v}_{b,\min}$ in (4.42) auch hier ein Anker eingebaut. Dieser Wert muss natürlich sinnvoll belegt werden. In den Anwendungsfällen dieser Arbeit kommen sehr selten Verletzungen der Achsgeschwindigkeiten nach der Zerlegung vor. Für die durch Tabelle 2.1 auf Seite 19 gegebene Situation, werden nur in den Beispielen manchmal die Achsgeschwindigkeiten verletzt, in denen Positioniersätze mit sehr hoher Geschwindigkeit enthalten sind.

Einstellungen der Bewegungsplanung in der ersten Iteration

Bisher hat dieser Abschnitt nur die Regeln zur Änderung der Einstellungen in der Bewegungsplanung am Ende einer Iteration beschrieben. Dieser Unterabschnitt erläutert die Wahl der Grenzen für die erste Iteration des Vorgehens.

Wie schon in Abschnitt 4.3.1 beschrieben, erzeugt die Zerlegung einen glatten Spline als Bewegung der Grobachse. Im „Idealfall“, wenn es der Verfahrbereich der Feinachse zulässt, bewegt sich die Grobachse gar nicht. Dann sind die Bewegung von x und x_f identisch. Aus diesem Grund ist für die Beschleunigungsgrenze der ersten Iteration der Wert $\widehat{A}_{xf}$ sinnvoll. Auch wenn die Grobachse nicht steht, ist diese Wahl vernünftig. In diesem Fall werden dann mehrere Iterationen durchlaufen, bis die Bewegung gültig ist. Falls nicht anders erwähnt, findet in den noch folgenden Beispielen eine derartige Wahl statt.

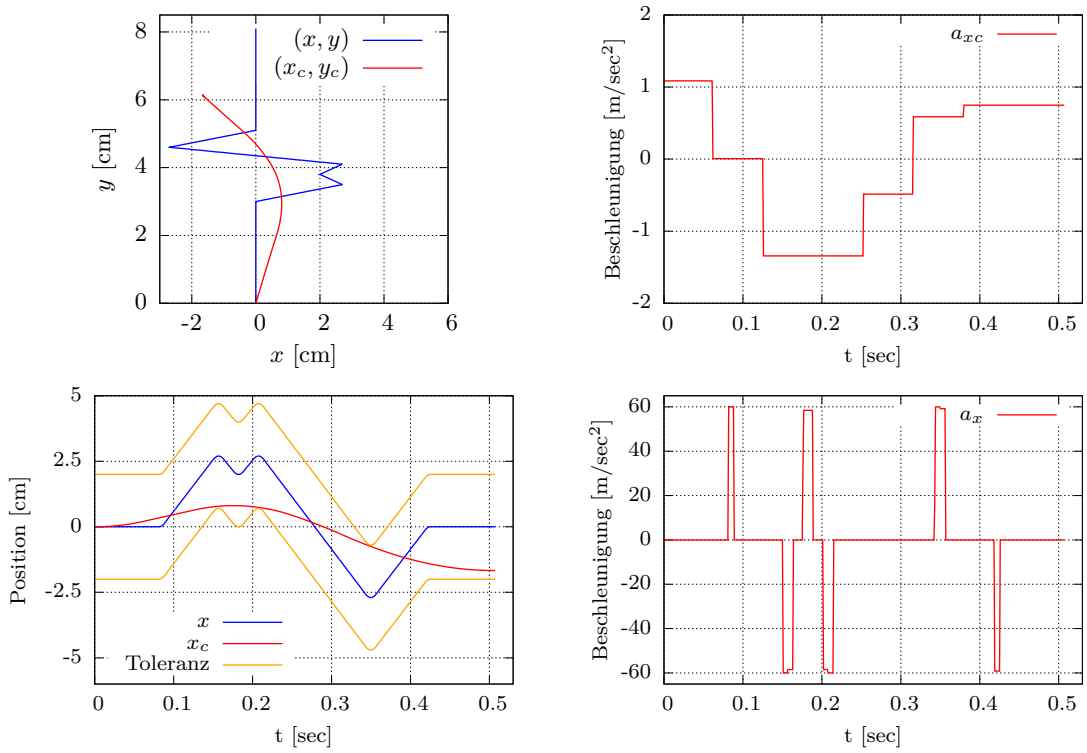

Abbildung 4.4.: Beispiel für die Argumentation der Wahl der Beschleunigungsgrenzen in der Bewegungsplanung.

Damit ist die Gesamtbewegung höchstens genauso schnell wie die Feinachse alleine. Es gibt zwar auch schnellere Bewegungen (siehe Kapitel 5), sie lassen sich aber mit diesem Ansatz nicht berechnen. Das soll anhand eines kleinen Beispiels mit MD2 der Tabelle 2.1 verdeutlicht werden. Der Grad bei der Splineapproximation in der Zerlegung ist $m = 2$ und die Variante gemäß Abbildung 4.2 ist a.2. Abbildung 4.4 zeigt die relevanten Achsverläufe. Die Beschleunigung der Grobachse hat sehr niedrige Absolutwerte und die Positionen nutzen den zulässigen Bereich voll aus. Die Phasen mit konstanter Beschleunigung sind deutlich länger als die Beschleunigungsphasen der Referenz im Bild rechts unten. Wäre die Beschleunigung der Referenz größer als $\widehat{A}_{xf}$, dann müsste die Beschleunigung der Grobachse ebenfalls diese dicht beieinanderliegendenden Wechsel haben, damit sie die Feinachse beim Beschleunigen unterstützte. Gerade das widerspräche aber der Idee der glatten Grobbewegung.

Der Startwert für den Bahnvorschub ist der entsprechende Wert für $\hat{v}_b$ aus Tabelle 2.1. Die Grenzen für die Achsgeschwindigkeit $\widehat{V}_x(j)$ und $\widehat{V}_y(j)$ müssen jeweils größer als $\hat{v}_b$ sein, dann haben sie in keiner Iteration Einfluss auf das Ergebnis der Bewegungsplanung (vergleiche Abschnitt 2.2). Wie oben schon erwähnt, ist kein Beispiel einer Kontur oh-

ne Positioniersätze bekannt, in dem es in einer Iteration zu einer Überschreitung einer Achsgeschwindigkeitsrestriktion kommt.

Anders verhält es sich, wenn die Kontur Positioniersätze enthält. Für sie gibt es keine technisch bedingte Grenze für die Bahngeschwindigkeit, sondern es zählen nur die der Achsgeschwindigkeit. Die oben beschriebene Regel benötigt aber einen endlichen Wert für den Vorschub, da dieser bei Geschwindigkeitsverletzungen am Ende einer Iteration verringert wird.

Die Grenze der Bahngeschwindigkeit soll die maximal mögliche Bahngeschwindigkeit sein, die die Grobachsen alleine auf dem aktuellen Positioniersatz erreichen kann. Sei hier Δx die Länge der x–Komponente des Positioniersatzes und Δy entsprechend die der y–Komponente. Dann sollen die Geschwindigkeiten von x_c und y_c dasselbe Verhältnis haben wie Δx und Δy. Die maximale Bahngeschwindigkeit ergibt sich dann, wenn mindestens eine der beiden Geschwindigkeitskomponenten an ihrer Grenze ist. Angenommen das ist die Komponente der x–Achse. Dann ist die gesuchte Bahngeschwindigkeitsgrenze für den betrachteten Positioniersatz

$$\hat{v}_b = \sqrt{\widehat{V}_{xc}^2 + \left(\widehat{V}_{xc}\frac{\Delta y}{\Delta x}\right)^2}.$$

Wenn die y_c-Achse an ihre Grenze kommt, dann ergibt sich entprechend

$$\hat{v}_b = \sqrt{\left(\widehat{V}_{yc}\frac{\Delta x}{\Delta y}\right)^2 + \widehat{V}_{yc}^2}.$$

4.3.3. Anwendung der Verfahren auf gesamte Konturen

Durch die Untersuchung von Testfällen konnten Erfahrungen über Ablauf und Funktionieren des Gesamtverfahrens erlangt werden. Dieser Abschnitt erläutert sie, auch anhand ausgewählter Beispiele.

In vielen Fällen sind die Grobbewegungen aus der Zerlegung in allen Iterationen gültig. In den Wiederholungen erfolgt dann die Anpassung der Feinachse. Die Abbildungen 4.6 und 4.7 zeigen ein Beispiel mit der Konfiguration MD2 aus Tabelle 2.1 und der Least–Squares–Approximation für die Zerlegung. Dabei ist $m = 3$ der Splinegrad und die inneren Knoten sind einfach. In der ersten Iteration verletzt die Feinachse ihre Grenze von 60 m/sec^2 noch an drei Stellen. Nach der Reduzierung von $\widehat{A}_x(j)$ in den entsprechenden Sätzen erzeugt schon die zweite Iteration eine gültige Gesamtbewegung. Es genügt aber nicht immer nur eine weitere Iteration, um die Feinbewegung anzupassen, denn die Trajektorie der Grobachse verändert sich durch die immer anderen Referenztrajektorien im Laufe der Wiederholungen. So kann eine Verletzung der Feinachse auf einmal an Stellen auftreten, wo in der vorangegangenen Iteration keine war. Meistens sind sich die Grobbewegungen aber sehr ähnlich (wie im Beispiel) und nach wenigen Iterationen ist die Gesamtbewegung gültig.

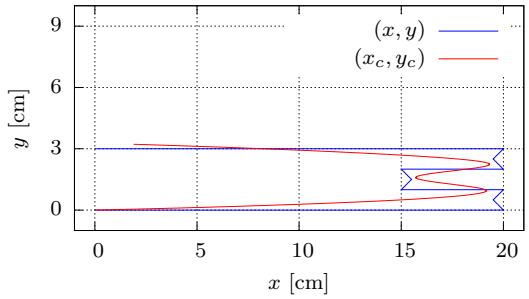

Abbildung 4.5.: Kontur zu Abbildungen 4.6 und 4.7 mit Grobbahn des Endergebnisses.

Für die Tschebyschow–Approximation und die glatte Approximation mit linearen Programmen trifft diese Beobachtung nicht zu. Hier ändern sich die Beschleunigungsverläufe der Grobachse mit den Iterationen deutlich. Das führt zu vielen Iterationen und damit zusammenhängend auch zu vielen (im Laufe der Iterationen unterschiedlichen) Sätzen, bei denen $\widehat{A}_x(j)$ reduziert wird. Das wirkt natürlich verlängernd auf die Gesamtzeit. Zudem gibt es ein weiteres Argument, das gegen diese beiden Varianten spricht. Die Lösungen derartiger Approximationsprobleme haben einen Verlauf des Fehlers (also der Feinachse), der ständig das Vorzeichen wechselt. Zusammen mit den äquidistant verteilten Knoten kann das im ungünstigen Fall zu einer Schwingungsanregung in der Mechanik der Maschine führen. Damit sind diese Varianten für die Zerlegung gegenüber den anderen im Nachteil und werden im Folgenden nicht weiter betrachtet.

Es gibt auch Anwendungsfälle, in denen die Bewegung von x_c in den ersten Iterationen nicht gültig ist. Dann dauert es häufig vom Start mehrere Wiederholungen, bis die Grobbewegung alle ihre Grenzen einhält. Das ist vor allem der Fall, wenn die Kontur lange Positioniersätze enthält. In solchen Situationen sind deutlich weniger Iterationen notwendig, wenn die Zerlegung die zweite Phase nutzt. Die glatte Approximation mit quadratischen Programmen, Variante a.2. aus Abbildung 4.2, braucht in der Regel die wenigsten Iterationen. Das Beispiel in Abbildung 4.9 verdeutlicht diesen Aspekt. Das obere Bild zeigt die Grobbewegung aus der Least–Squares–Approximation, auch hier mit $m = 3$ und einfachen Knoten. Vor und nach $t = 0.6$ Sekunden treten deutliche Verletzungen von $\widehat{A}_{xc}$ auf. Es dauert insgesamt drei Iterationen bis zu einer gültigen Gesamtbewegung (Bild in der Mitte). Das untere Bild präsentiert das Ergebnis nach der ersten Iteration bei Zerlegung mit Variante a.2., der Nachbrenner war in diesem Fall $N^* = 5$. Die Grobbewegung ist gültig und durch den (zufällig) günstigen Verlauf auch die Feinbewegung.

Das Beispiel verdeutlicht ebenfalls den Einfluss der Zerlegung auf die Zeitdauer der Bewegung. Ist die Grobbewegung anfangs ungültig, verkleinert das Vorgehen $\widehat{A}_x(j)$ auf vielen Sätzen deutlich, wie das Bild des Endergebnisses in der Mitte von Abbildung 4.9 zeigt. Das führt natürlich zu einer längeren Zeitdauer der Gesamtbewegung. Aber auch

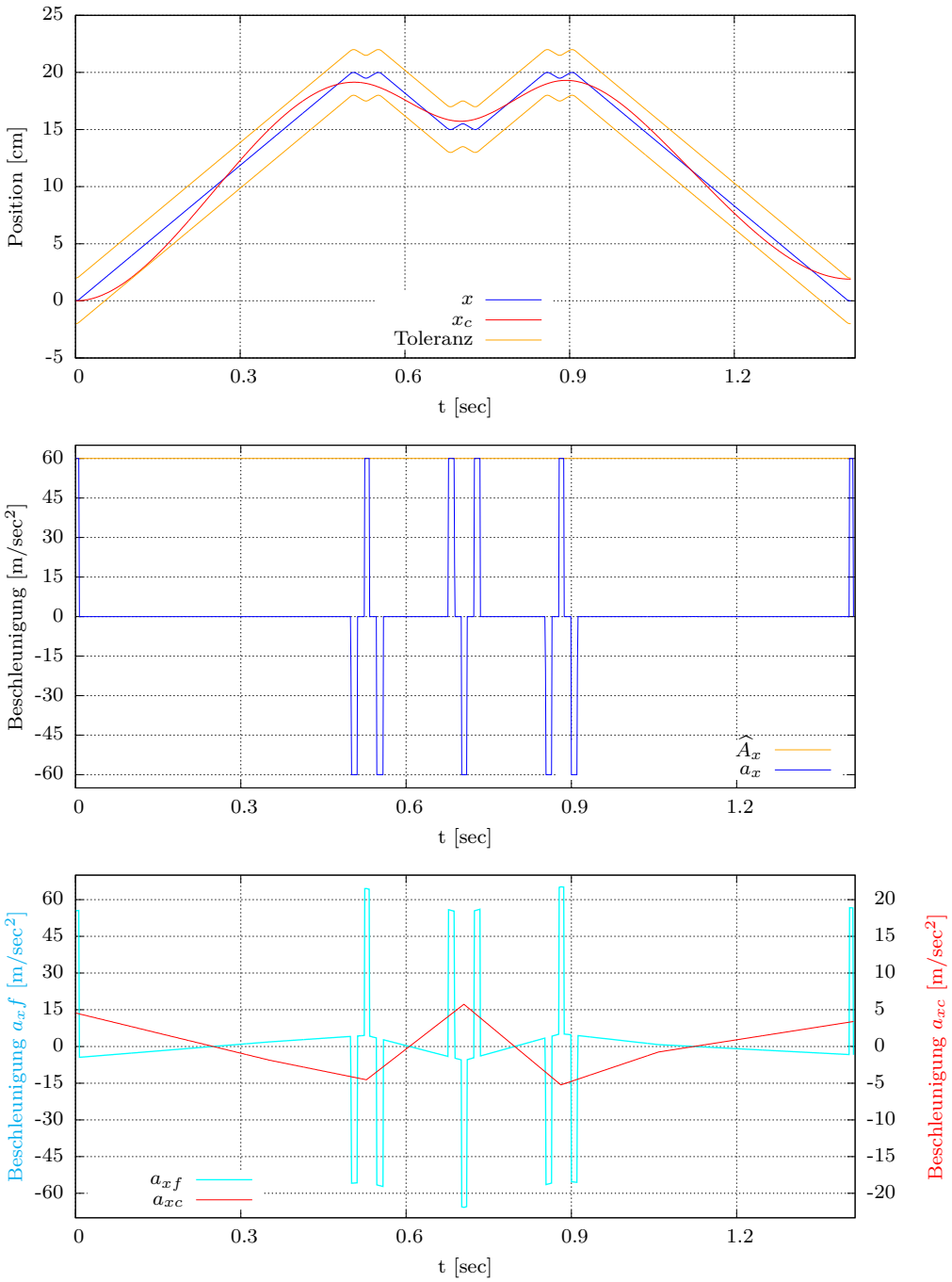

Abbildung 4.6.: Ergebnis der ersten Iteration aus dem Beispiel der iterativen Zerlegung mit Least–Squares–Approximation. Abbildung 4.5 zeigt die dazugehörige Kontur.

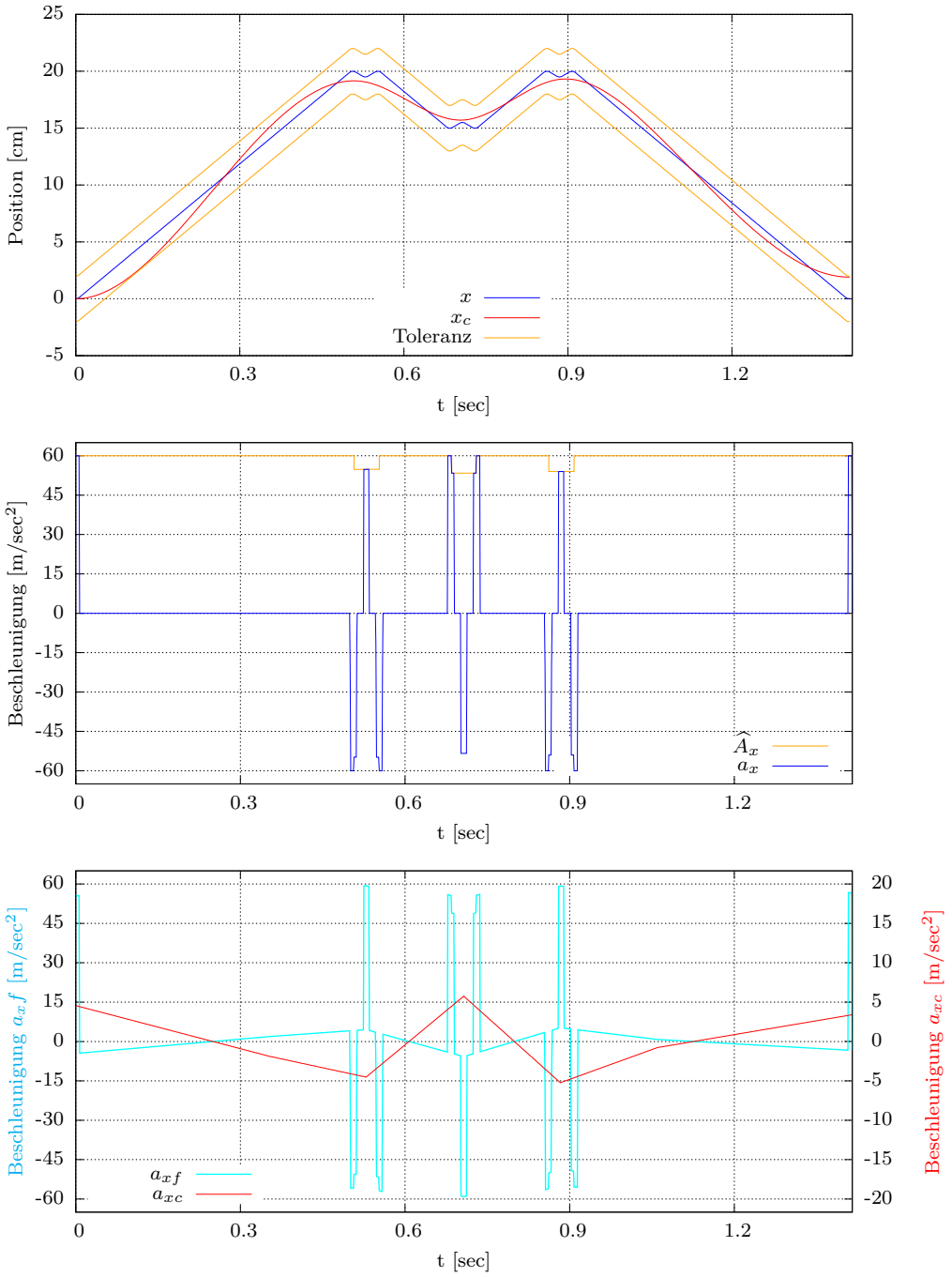

Abbildung 4.7.: Endergebnis aus dem Beispiel der iterativen Zerlegung mit Least–Squares–Approximation. Abbildung 4.5 zeigt die dazugehörige Kontur, Abbildung 4.6 die erste Iteration.

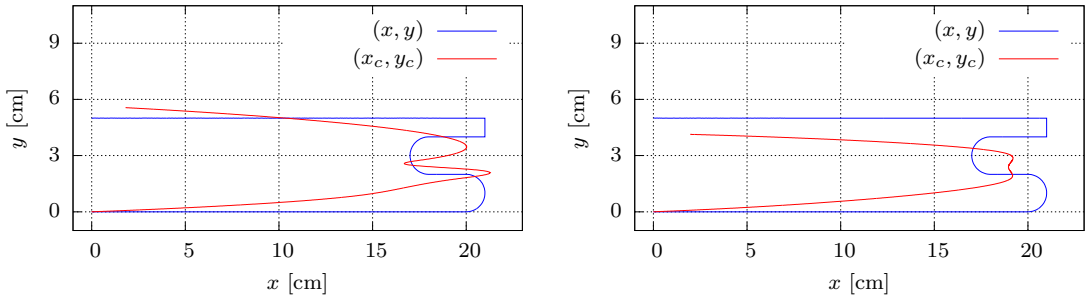

Abbildung 4.8.: Konturen und Grobbewegung zum Beispiel aus Abbildung 4.9, links Least–Squares–Approximation, rechts glatte Approximation mit quadratischen Programmen.

wenn die Zerlegung immer eine gültige Bewegung der Grobachse ergibt, sind niedrige Absolutwerte der Beschleunigung besser. Je niedriger sie sind, desto weniger müssen die Werte für $\widehat{A}_x(j)$ reduziert werden. Deswegen ist es auch im Sinne der Produktivität umso besser, je glatter die Grobbewegung ist. Für die Anwendungsfälle ist dieser Aspekt aber nicht immer wesentlich. Tabelle 4.1 zeigt einige Bearbeitungszeiten von Testläufen an dem Teileprogramm aus Abbildung 4.10. Die Unterschiede sind verglichen mit der Gesamtdauer der Bewegung gering. Das liegt hauptsächlich daran, dass sich der Werkzeugkopf auf einem Großteil der Kontur mit dem programmierten Vorschub bewegt. Ebenso ist der Unterschied zur Zeitdauer der Bewegung der ersten Iteration, die eine Untergrenze für die Bearbeitungszeit darstellt, gering. Lediglich die Anzahl der Iterationen ist im Fall der Zerlegung mit der Least–Squares– oder der Schoenberg–Approximation größer als bei den anderen Zerlegungsmethoden. Das liegt daran, dass es dort mehr Wiederholungen braucht, bis die Grobbewegung gültig ist. Genauer betrachtet führt der Positioniersatz von circa $[x, y]^T = [0.4\text{cm}, 7\text{cm}]^T$ nach $[1.5\text{cm}, -0.2\text{cm}]^T$ einige Iterationen lang zu einer Beschleunigungsverletzung der x_c–Achse. In diesem Bereich kommt es zu deutlichen Reduzierungen von $\widehat{A}_x(j)$.

Auf eine entsprechende Tabelle zur Konfiguration MD1 wurde hier verzichtet. Es ergeben sich fast identische Bearbeitungszeiten für alle Zerlegungsmethoden von etwa 10.038 Sekunden. Außerdem treten in diesem Fall wesentlich seltener ungültige Grobbewegungen nach der Zerlegung auf. Das liegt hauptsächlich an dem kleineren Vorschub beim Schneiden und dem größeren Verfahrbereich der feinen Achsen.

Für die Bewegung der Grobachse gibt es neben der Wahl der Zerlegungsmethode auch die Möglichkeit, den Splinegrad m und die Vielfachheit k_{vf} der inneren Knoten zu variieren. Die Auswirkung auf die Bearbeitungszeit ist eher unbedeutend, wie Tabelle 4.1 zeigt. Allerdings sind zwei andere Aspekte nicht unwesentlich. Zum einen steigt mit dem Splinegrad die Anzahl der Rechenoperationen in jeder Iteration. Zum anderen haben hö-

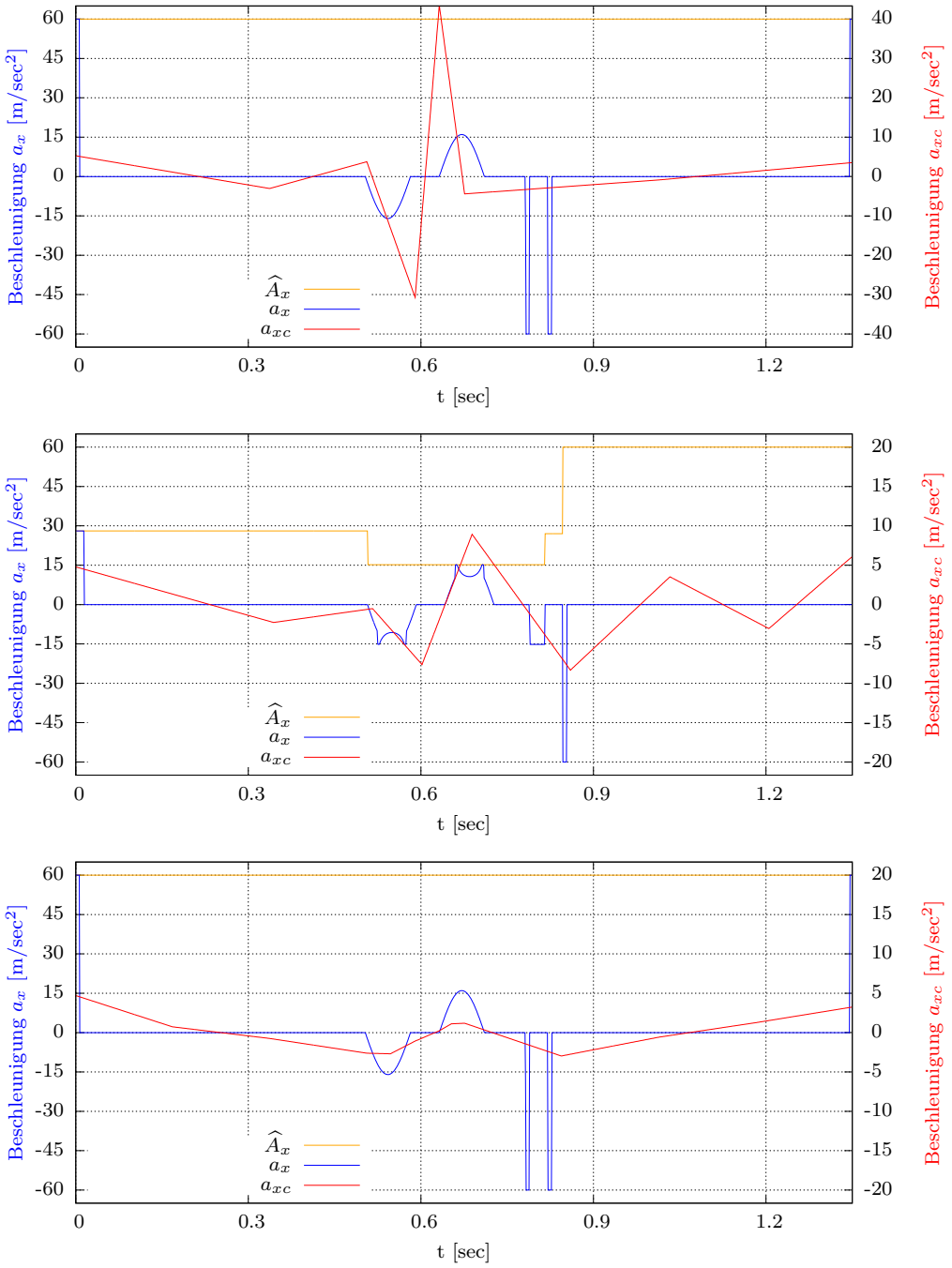

Abbildung 4.9.: Beispiel für das iterative Vorgehen mit Konfiguration MD2. Oben: 1. Iteration Least–Squares. Mitte: Endergebnis Least–Squares. Unten: Erste Iteration Approximation mit QP.

Zerlegung		$m = 2$, $k_{\mathrm{vf}} = 1$	$m = 3$, $k_{\mathrm{vf}} = 1$	$m = 3$, $k_{\mathrm{vf}} = 2$	$m = 4$, $k_{\mathrm{vf}} = 1$	$m = 4$, $k_{\mathrm{vf}} = 2$
Least–	(s)	6.835	6.847	6.848	6.832	6.833
Squares	(p)	0.612	0.679	0.627	0.649	0.637
(a.0)	(g)	7.447	7.526	7.475	7.481	7.470
	(i)	7	10	4	4	5
Schoenberg	(s)	6.848	6.822	6.828	6.816	6.828
(c.0)	(p)	0.656	0.632	0.594	0.588	0.624
	(g)	7.504	7.454	7.422	7.405	7.452
	(i)	7	7	4	5	8
Smoothing–	(s)	6.819	6.820	6.819	6.818	6.821
Spline	(p)	0.590	0.586	0.583	0.584	0.584
(a.1)	(g)	7.409	7.406	7.402	7.402	7.405
	(i)	4	6	5	4	8
Glatt mit	(s)	6.814	6.813	6.814	6.814	6.813
QP	(p)	0.582	0.580	0.580	0.580	0.580
(a.2)	(g)	7.396	7.393	7.394	7.394	7.394
	(i)	3	2	4	5	4
Erste	(s)	6.809				
Iteration	(p)	0.564				
	(g)	7.373				

Tabelle 4.1.: Ergebnisse mit dem Ansatz der iterativen Zerlegung zur Kontur aus Abbildung 4.10 bei Konfiguration MD2. Die Zeilen mit den Abkürzungen (s), (p) und (g) sind die Zeiten für das Schneiden und Positionieren sowie die Gesamtzeit, die Zeilen mit (i) sind die Anzahlen der Iterationen. Bei der Addition treten Rundungsfehler auf.

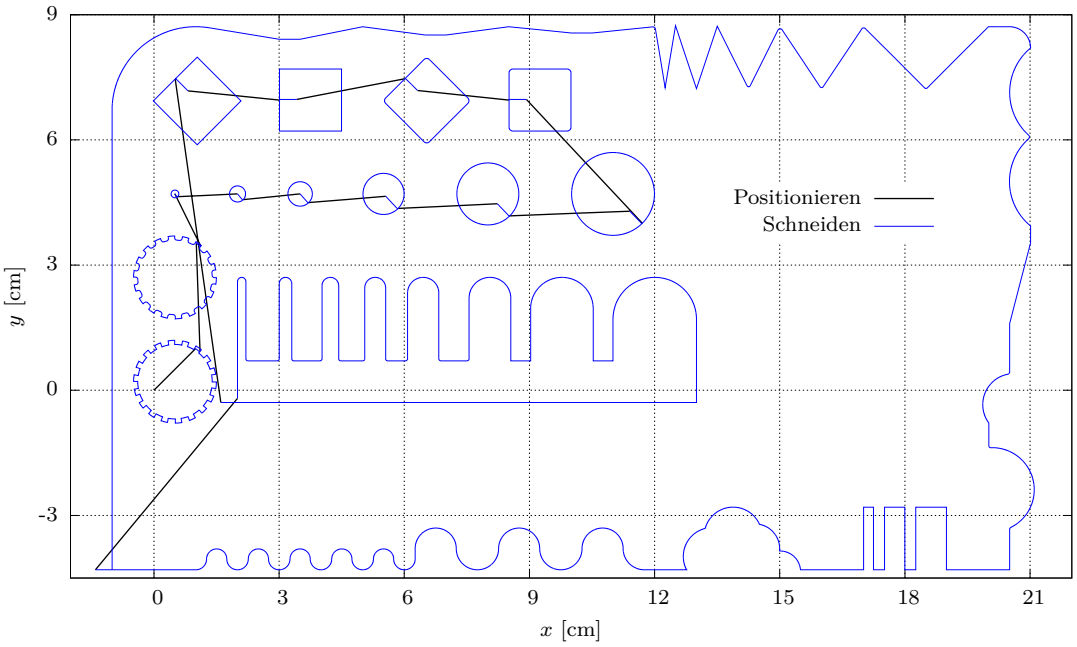

Abbildung 4.10.: Größere Kontur

hergradige Splines ($m > 2$) mit einfachen inneren Knoten dort glattere Übergänge. Im Fall $m = 2$ und $k_{\mathrm{vf}} = 1$ sind das beispielsweise noch Sprünge in der Beschleunigung. Im Fall $m = 4$ und $k_{\mathrm{vf}} = 1$ sind die Übergänge der zweiten Ableitung C^2–stetig, die Grobbewegung ist damit sogar ruckbegrenzt. Mit derartigen Sollwerten für die Bewegung der Grobachse ist eine Anregung der Mechanik weniger wahrscheinlich.

Der letzte Aspekt ist auch in Hinblick auf die Anwendung des Ansatzes bei ruckbegrenzter Bewegungsführung interessant. Es ist sehr wahrscheinlich, dass in den Fällen von oben, wo die Bewegung der groben Achse in allen Iterationen gültig ist, auch bei Eingabe ruckbegrenzter Trajektorien in den Zerlegungsblock gültige, ruckbegrenzte Trajektorien der groben Achse entstehen. Erste Tests haben dies bestätigt. Für weitreichendere Aussagen, auch im Zusammenhang mit der modifizierten Steuerungsstruktur aus Abbildung 4.1, muss dieser Fall aber noch ausführlich getestet werden. Das ist aber nicht mehr Inhalt dieser Arbeit.

4.3.4. Abschnittsweises Vorgehen

Bis hierher betrachtet der Ansatz immer die gesamte Kontur auf einmal. Wie in Kapitel 2 schon erwähnt, kann das in einer Steuerung nicht realisiert werden. Sie geht abschnittsweise vor und kennt immer nur einen Teil der Sätze. Dieser Abschnitt beschreibt ein Vorgehen, das eine Umsetzung der iterativen Zerlegung im Zeitbereich in der Steuerung ermöglicht.

1. $i := 0, \quad v_{xc,w} = v_{yc,w} = v_{b,w} = 0,$
   $x_{c,w} = x(s_0),\ y_{c,w} = y(s_0)$

2. Berechne mit der iterativen Zerlegung im Zeitbereich
   die Achsbewegungen für die Sätze $i+1$ bis $i+n_S$
   mit dem Zeitintervall $[t_a, t_e]$, sodass
   $v_{xc}(t_a) = v_{xc,w}, \quad v_{yc}(t_a) = v_{yc,w}, \quad v_b(t_a) = v_{b,w}$ und
   $v_{xc}(t_e) = 0, \quad v_{yc}(t_e) = 0, \quad v_b(t_e) = 0$ und
   $x_c(t_a) = x_{c,w} \quad y_c(t_a) = y_{c,w}$ gelten.

3. Bestimme den Zeitpunkt t_w, zu dem die Bewegung
   von Satz $i + n_S - n_w$ auf $i + n_S - n_w + 1$ wechselt.

4. Setze $v_{xc,w} := \dot{x}_c(t_w)$, $v_{yc,w} := \dot{y}_c(t_w)$, $v_{b,w} := v_b(t_w)$ und
   $x_{c,w} = x_c(t_w),\ y_{c,w} = y_c(t_w)$.

5. Gebe die Bewegung des Zeitintervalls $[t_a, t_w]$
   in Richtung Interpolator weiter.

6. Falls $i + n_S - 1 < n_h - 1$,
   dann $i := i + n_S - n_w$ und gehe zu Schritt 2.
   Andernfalls ist man fertig.

Algorithmus 4.2: Phase II

Wie gehabt besteht die Kontur aus n_h Sätzen. Das Vorgehen durchläuft sie, sodass die iterative Zerlegung immer auf eine Teilkontur aus $n_S \in \mathbb{N}$ Sätzen angewendet wird. Etwas formaler stellt es Algorithmus 4.2 dar. Dabei sind zwei Punkte sehr wichtig und bewusst gewählt. Zum einen stehen alle Achsen am Ende des aktuellen Abschnitts. Die Ergebnisse mit anderen Bedingungen für den rechten Rand hängen stark von dem weiteren Verlauf der Kontur ab, der aber zu diesem Zeitpunkt noch nicht bekannt ist. Die Vorgabe des Anhaltens ist sozusagen ein guter Kompromiss. Zum anderen wandert nach Beendigung der Iterationen die gültige Bewegung nicht komplett an den Interpolator weiter, sondern der Anteil der letzten $n_w \in \mathbb{N}$ Sätze, $n_w < n_S$, wird verworfen. Somit ergeben sich für den neuen Abschnitt Positions- und Geschwindigkeitsrandbedingungen für jede Achse und damit auch für die Bahnbewegung $s_b(t)$. Die Bewegungsplanung und die Zerlegung müssen sie bei ihren Berechnungen berücksichtigen, damit letztendlich C^1–stetige Bewegungen aller Achsen entstehen.

Anschaulich entspricht dieses Vorgehen einem Fenster, das sich schrittweise über die Kontur bewegt. Es verschiebt sich allerdings in jedem Schritt nicht um seine Fensterbreite, sondern um weniger. Der wesentliche Vorteil von diesem Vorgehen ist, dass die redundanten Achsen nicht ständig anhalten, sondern vor allem die Grobachsen mit gleichmäßigerer Geschwindigkeit verlaufen. Abbildung 4.11 zeigt die Entstehung einer Bewegung mit diesem abschnittsweisen Vorgehen. Die Konfiguration ist wieder MD2, der Splinegrad $m = 2$ und die Vielfachheit $k_{\mathrm{vf}} = 1$. Die Parameter des abschnittsweisen Vorgehens sind $n_S = 10$ und $n_w = 3$. Die orangen, vertikalen Linien markieren die Wechsel. Die Kontur in diesem Beispiel ist der äußere Rand aus Abbildung 4.10, derart verschoben, dass der Startpunkt bei $[0, 0]^T$ liegt.

Bei diesem Vorgehen kann ein Problem auftreten. Angenommen, am linken Rand eines Abschnitts ist $v_{b,w}$ die Bedingung für die Bahngeschwindigkeit. Dieser Wert stammt aus den Berechnungen des vorherigen Abschnitts und basiert unter anderem auf den Beschleunigungsgrenzen der letzten n_w Sätze. Diese Grenzen haben am Ende der Iterationen eine gewisse Größe. Jetzt kann im ungünstigen Fall das iterative Vorgehen im neuen Abschnitt für diese Sätze kleinere Beschleunigungsgrenzen bestimmen, sodass die Randbedingung $v_{b,w}$ nicht erfüllt werden kann. An solchen Stellen versagt das abschnittsweise Vorgehen.

Allerdings können passende Modifikationen diese Problematik entschärfen. Eine Maßnahme ist, in jedem Abschnitt für die letzten n_w Sätze in der ersten Iteration kleinere Werte als $\widehat{A}_{xf}$ und $\widehat{A}_{yf}$ für $\widehat{A}_x(j)$ und $\widehat{A}_y(j)$ zu wählen. Im darauffolgenden Abschnitt beginnt die Bewegungsplanung auf diesen Sätzen wieder mit $\widehat{A}_{xf}$ und $\widehat{A}_{yf}$ und kommt in den Iterationen (hoffentlich) nicht unter die Werte der Endsätze des vorherigen Abschnitts. Die zweite Maßnahme ist, die Beschleunigungsgrenzen auf den ersten n_w Sätzen durch die Werte der letzten Iteration des vorherigen Abschnitts nach unten zu begrenzen. Damit ist garantiert, dass die Anschlussbedingung der Bahngeschwindigkeit erfüllt werden kann. Das heißt aber nicht, dass das iterative Vorgehen eine gültige Zerlegung findet. Es gibt aber einen „Rettungsanker“, der dann genutzt wird, wenn nach einer bestimmten Anzahl von Iterationen keine gültige Bewegung vorliegt. Dabei verzichtet man auf den Überlappungsbereich der beiden aufeinanderfolgenden Abschnitte und verschiebt das Fenster um n_S Sätze. Das führt zu einem Stillstand aller Achsen auf der Kontur, wenn auch nur für einen Zeitpunkt. Im neuen Abschnitt gibt es nur Geschwindigkeiten von null als Randbedingungen und die Bewegungsplanung findet eine Referenzbewegung, egal wie groß die Beschleunigungsgrenzen sind. Dieses Anhalten auf der Kontur kann dabei auch an Stellen auftreten, wo es eigentlich nicht notwendig ist, das heißt, wo die Kontur keine Ecke hat.

Bei der Verwendung der zweiten Phase in der Zerlegung ist eine Anpassung des Nachbrenners sinnvoll. Die ursprüngliche Variante führt noch eine bestimmte Anzahl von Iterationen mit der iterativen Least–Squares–Approximation durch, nachdem die Toleranzbedingung schon eingehalten wird. Der neue Knoten wird immer in der Nähe der Stelle mit dem größten Fehler zwischen Referenz und Approximation eingefügt. Im abschnittsweisen Vorgehen kann die linke Positionsrandbedingung überall innerhalb des Toleranzbereichs liegen und damit auch auf dessen Rand. In diesem Fall führt die obige Einfügestrategie zu einer Konzentration von Knoten am linken Rand. Das ist nicht erwünscht. Der Nach-

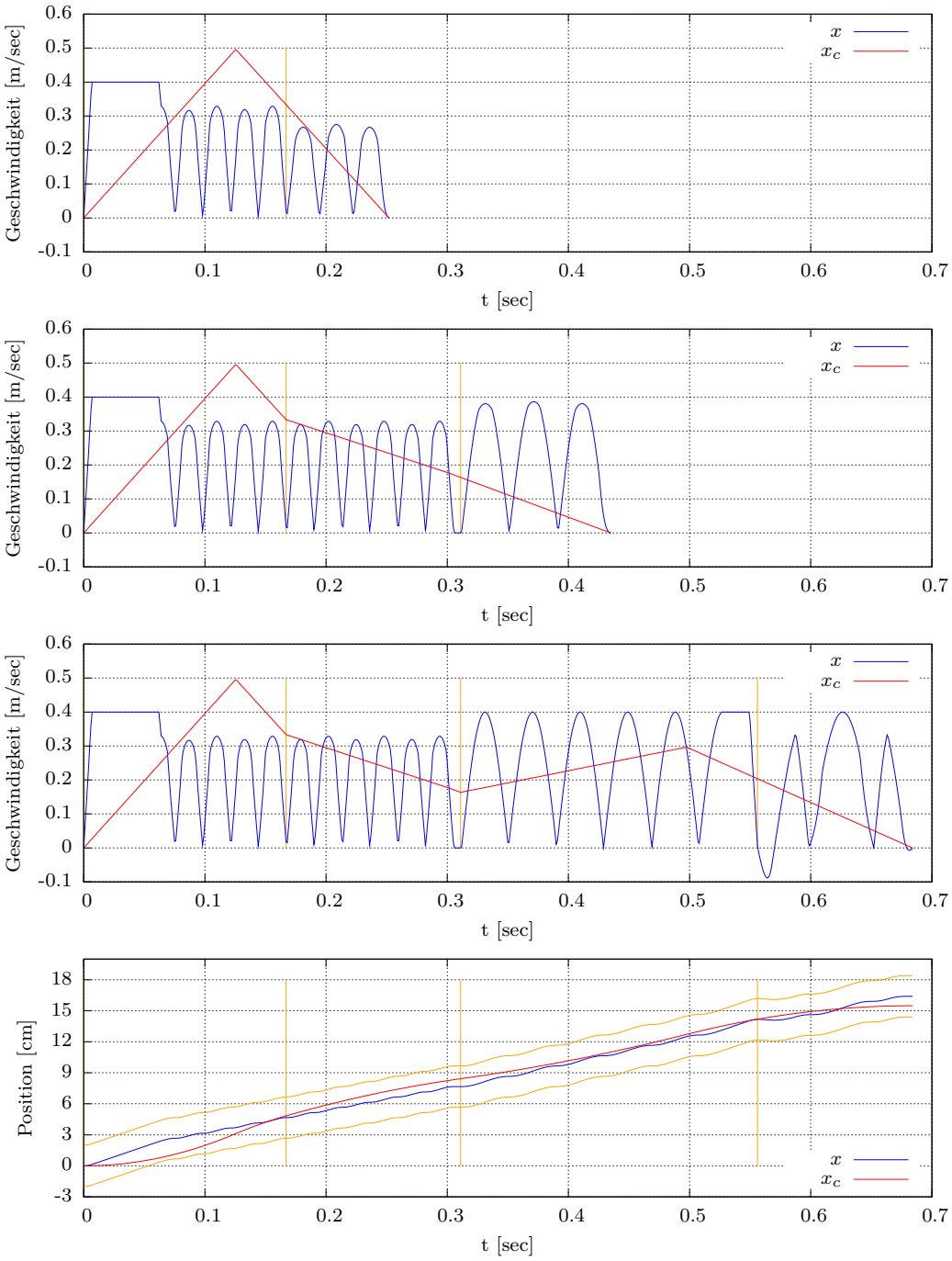

Abbildung 4.11.: Beispiel für die ersten drei Abschnitte mit dem abschnittsweisen Vorgehen.

brenner beim abschnittsweisen Vorgehen fügt Knoten auf eine andere Art und Weise ein. In der Mitte des jeweils längsten „Intervalls“ zweier benachbarter Knoten der aktuellen Knotenfolge wird ein Knoten mit Vielfachheit k_{vf} eingefügt. Nach dem N^*-ten Mal, in dem der maximale Fehler kleiner als $\widehat{S}_{xf}$ ist, bricht der Nachbrenner ab.

In der Anwendung verhält sich das abschnittsweise Vorgehen je nach gewählter Zerlegungsmethode und je nach Parametrisierung unterschiedlich. Das soll anhand der Ergebnisse mit dem Teileprogramm aus Abbildung 4.10 verdeutlicht werden. Der erste Testlauf nutzt $n_S = 10$ und $n_w = 3$ sowie $m = 3$ und $k_{\mathrm{vf}} = 1$. Mit der Least–Squares–Approximation funktioniert das Vorgehen zwar in vielen Abschnitten, allerdings gibt es einen, in dem es den Rettungsanker benötigt und selbst dieser findet in 100 Iterationen keine gültige Bewegung. Es gelingt nicht, eine Verletzung der x_c–Beschleunigung zu beheben. Solche Fälle führen dann zum Abbruch[13] und es kann nicht die gesamte Kontur bearbeitet werden. Es sei noch angemerkt, dass mit $m = 2$ und $k_{\mathrm{vf}} = 1$ das Vorgehen mit dem gesamten Teileprogramm funktioniert, mit $m = 3$ und $k_{\mathrm{vf}} = 2$ wiederum nicht. Bei der Verwendung der Schoenberg–Approximation gibt es in allen drei Fällen Ergebnisse ohne Hilfe des Rettungsankers.

Mit der Approximation durch Smoothing–Splines und $N^* = 5$ in der Least–Squares–Approximation für Phase eins funktioniert das Vorgehen auf allen Abschnitten ohne Rettungsanker. Allerdings verwenden einige Abschnitte die zweite Phase der Zerlegung nicht. Dort liegt die linke Randbedingung der x_c–Position genau auf oder sehr nahe am Rand des Toleranzbereichs. In diesen Fällen erfüllt schon der Least–Squares–Approximant die Abbruchbedingung der Regula Falsi (vergleiche mit Ende von Abschnitt 4.2.6). Das ergibt einen Glätteparameter $\lambda = 0$ und birgt die Gefahr, dass die Grobbewegung in den Iterationen wie oben nicht gültig wird. In diesem konkreten Beispiel tritt diese Situation aber nicht ein. Im Fall der Zerlegung mit der glatten Approximation durch quadratische Programme funktioniert das Vorgehen ebenfalls. Die Situation mit der nicht durchgeführten zweiten Phase kann hier nicht eintreten. Die Optimierung wird durchgeführt.

Im zweiten Testlauf mit einer Konfiguration $n_S = 20$ und $n_w = 5$ benötigt das Vorgehen mit keiner Zerlegungsmethode den Rettungsanker. Die Zeiten sind mit beiden Konfigurationen in derselben Größenordnung wie die aus Tabelle 4.1.

Dieses Beispiel zeigt, dass das abschnittsweise Vorgehen bei Teileprogrammen mit relativ kurzen Positioniersätzen und mit Variante a.2 für die Zerlegung zuverlässige Ergebnisse liefert. Mit den anderen Varianten gibt es eher Abschnitte, die Probleme machen, auch wenn das in diesem Beispiel mit der Schoenberg–Approximation nicht der Fall ist. In einer Realisierung dieses Vorgehens ist es daher vernünftig, besonders im Hinblick auf die benötigte Rechenzeit, die zweite Phase und besonders Variante zwei nur in den Abschnitten zu verwenden, in denen die erste Phase keine zufriedenstellenden Ergebnisse liefert.

Bei Teileprogrammen mit langen Positioniersätzen treten problematische Abschnitte mit allen Varianten für die Zerlegung auf. In diesen Fällen hilft es nur, die jeweiligen Positioniersätze gesondert zu behandeln. Dabei endet das abschnittsweise Vorgehen vor dem

[13] In einer Implementierung des Vorgehens muss man natürlich die Anzahl der Iterationen begrenzen.

langen Positioniersatz. Dann werden alle Achsen zum Ende des Positioniersatzes bewegt, sodass die Feinachse wieder zentriert ist und zuletzt beginnt das Vorgehen wieder von neuem. Ab welcher Länge der Sätze eine derartige Handhabung notwendig ist, ermittelt man durch Testen mit repräsentativen Teileprogrammen.

Dieses Vorgehen kann auch aus einem anderen Grund bei langen Positioniersätzen sinnvoll sein. In den Beispielen stellt man fest, dass bei Konturen mit lediglich kurzen Positioniersätzen die Beschleunigungen der Grobachse betragsmäßig deutlich unter dem Grenzwert $\widehat{A}_{xc}$ liegt. Mit der Hinzunahme langer Positioniersätze ändert sich das. Besonders bei der glatten Approximation mit quadratischen Programme gibt es wegen den Nebenbedingungen Bereiche mit $a_{xc}(t) = \pm\widehat{A}_{xc}$, die auch in Schneideblöcke hineinreichen. Das kann gegebenenfalls wieder zu Qualitätsproblemen führen.

Unter der Berücksichtigung aller dieser Punkte und der Ergebnisse aus den Beispielen, lässt sich zusammenfassen, dass sich das abschnittsweise Vorgehen auf typische Konturen bei Verwendung der Maschinendaten MD1 und MD2 aus Tabelle 2.1 anwenden lässt. Wie man das Vorgehen letztendlich genau parametrisiert, muss man im Zusammenhang mit dem speziellen Anwendungsfall beziehungsweise mit der konkreten Maschine untersuchen.

4.3.5. Ausdünnen der Daten

Für die in diesem Ansatz durchgeführten numerischen Berechnungen ist eine gewisse Rechenleistung notwendig. Sie hängt wesentlich von der Anzahl n_t der Punkte ab. Weniger Punkte bedeuten eine kürzere Rechenzeit. Ist sie notwendig, kann die Referenztrajektorie ausgedünnt werden oder die Bewegungsplanung tastet sie grober ab. Bisher haben wir angenommen, dass die Eingabedaten des Zerlegungsblocks in dem Echtzeittakt t_{ipo} des Hauptlaufs vorliegen. Dieser Abschnitt untersucht die Möglichkeit, in der präparativen Phase oder zumindest in Teilen davon mit gröberen Daten zu arbeiten.

Die erste Idee besteht darin, nur in der Zerlegung die Daten gröber als mit t_{ipo} abzutasten. Die darin entstehende Splinefunktion $x_c(t)$ lässt sich überall auswerten, also auch in einem t_{ipo}–Raster. Der Überprüfungsblock funktioniert daher wie gehabt. Eine Überlegung ist aber notwendig. Bisher waren die Ergebnisse der Zerlegung Verläufe der Grobachse, die an den diskreten Stellen (des t_{ipo}–Rasters) die Toleranzbedingung erfüllen. Das ist hier nicht mehr garantiert. Der Fehler lässt sich aber abschätzen, da in unserem Fall die Absolutwerte der Geschwindigkeiten von x_c und x begrenzt sind, $\widehat{V}_{xc,1}$ und $\widehat{V}_{x,1}$ sind diese Grenzen. Für zwei aufeinanderfolgende Referenzwerte $x(t_j)$ und $x(t_{j+1})$ ist $\frac{1}{2}\widehat{V}_{x,1}(t_{j-1}-t_j)$ eine obere Grenze für die Entfernung der Referenz von diesen beiden Punkten innerhalb des Intervalls (t_j, t_{j+1}). Gleichermaßen verhält es sich mit $\frac{1}{2}\widehat{V}_{xc,1}(t_{j+1}-t_j)$ und den Punkten $x_c(t_j)$ und $x_c(t_{j+1})$. Gilt nun

$$|x(t_j) - x_c(t_j)| \leq \varepsilon \quad \text{und} \quad |x(t_{j+1}) - x_c(t_{j+1})| \leq \varepsilon,$$

so folgt

$$|x(t) - x_c(t)| \leq \varepsilon + \frac{1}{2}\widehat{V}_{xc,1}(t_{j+1} - t_j) + \frac{1}{2}\widehat{V}_{x,1}(t_{j+1} - t_j), \qquad t \in (t_j, t_{j+1}).$$

Wendet man die Approximationsverfahren mit dem maximal erlaubten Fehler

$$\widehat{S}_{xf} - \frac{1}{2}\widehat{V}_{xc,1}\Delta t - \frac{1}{2}\widehat{V}_{x,1}\Delta t, \qquad \Delta t = \max_j\{t_{j+1} - t_j\},$$

an, dann erfüllt jede feinere Abtastung die Toleranzbedingung mit $\widehat{S}_{xf}$. An dieser Stelle muss man natürlich beachten, dass $\widehat{S}_{xf} > \frac{1}{2}\widehat{V}_{xc,1}\Delta t + \frac{1}{2}\widehat{V}_{x,1}\Delta t$ erfüllt ist. Die Abschätzung ist außerdem nicht scharf und daher reicht in der Praxis meist eine geringere Reduzierung von $\widehat{S}_{xf}$. Noch besser ist eine adaptive Abtastung der Referenz. Dafür braucht man aber Regeln, die klären, wo mit welcher Dichte abgetastet wird.

Eine weitere Möglichkeit ist es, in der gesamten präparativen Phase mit gröberen Daten zu arbeiten. In diesem Fall muss der Interpolator noch zwei Dinge erledigen. Zum einen interpoliert er die Referenz wie im nichtredundanten Fall und berechnet die Werte der Summenachse. Zum anderen wertet er den Spline der Grobbewegung im t_{ipo}–Abtastraster aus und ermittelt die Werte der Feinachse. Die Abschätzung für den Verfahrbereich der Feinachse aus dem vorherigen Abschnitt ist hier relevant. Zusätzlich kann es zu Verletzungen der Geschwindigkeits- und Beschleunigungsgrenzen kommen, da der präparative Teil mit einer gröberen Abtastung arbeitet. Die Details dieser Verletzungen sollen hier aber nicht weiter untersucht werden.

4.3.6. Ein Beispiel für ein Werkstück

Abschnitt 3.2 zeigt mit Abbildung 3.2 ein Beispiel eines mit der geometrischen Zerlegung gefertigten Werkstücks mit der problematischen Schnittqualität. An derselben Maschine konnte auch der neue Ansatz getestet werden, Abbildung 4.12 zeigt das Ergebnis. Die Qualität der Schnittkante ist deutlich besser, es treten kaum sichtbare Wellen auf.

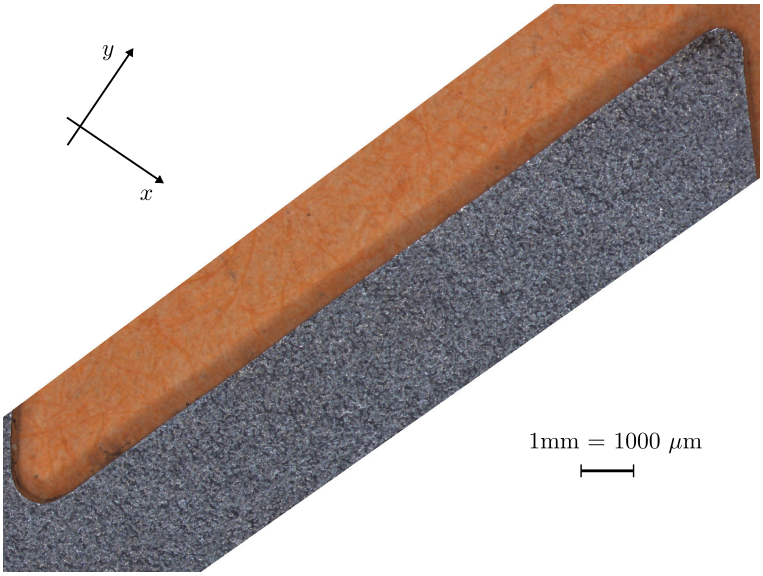

Abbildung 4.12.: Werkstück ohne Qualitätsprobleme an der Schnittkante.

5. Verkürzung der Bearbeitungszeit durch Umparametrisierung

Das in diesem Kapitel vorgestellte Verfahren kann als Ergänzung zum Ansatz des vorherigen Kapitels aufgefasst werden. Es verkürzt mithilfe einer Umparametrisierung der Achstrajektorien die Bearbeitungszeit. Die entstehende Bewegung ist sogar in einem gewissen Sinne zeitoptimal.

Der erste Abschnitt enthält die Idee des Verfahrens und die konkrete Aufgabenstellung. Der zweite führt den verwendeten Algorithmus aus und der letzte zeigt beispielhaft Ergebnisse.

5.1. Idee und Aufgabenstellung als Optimalsteuerungsproblem

In den Verfahren der Kapitel 3 und 4 ist eine möglichst kurze Bearbeitungszeit nie ein direktes Gütekriterium, das minimiert wird. Prinzipiell ist es natürlich möglich, die Aufgabe des Findens einer zeitoptimalen Bewegung der redundanten Kinematik B als Optimalsteuerungsproblem zu formulieren. Es besteht aus den Bedingungen in Definition 2.8 und der Zielfunktion (2.5). Das sieht dann sehr ähnlich wie dasjenige für Kinematik A in Abschnitt 2.2.1 aus. Allerdings ist hier keine äquivalente Umformung auf ein Problem mit nur einer Steuergröße möglich. Wegen der redundanten Achsen sind es mindestens drei. Daher ist auch kein so eleganter Algorithmus wie für Kinematik A bekannt.

Für die Lösung von Optimalsteuerungsproblemen gibt es eine Vielzahl numerischer Verfahren, einen Überblick liefert beispielsweise [35]. Allerdings sind bei deren Anwendung zur zeitoptimalen Bewegung von Kinematik B die Aussichten auf Erfolg wegen der hohen Komplexität der Nebenbedingungen gering. Zum einen finden derartige Verfahren nicht verlässlich ein globales Minimum, zum anderen steigt die benötigte Rechenzeit mit zunehmender Größe und Detailiertheit der Konturen sehr stark. An eine performante Nutzung in der Steuerung ist zur Zeit nicht zu denken. Die Doktorarbeit [8] untersucht einen derartigen Ansatz für einen Roboter. Die kinematische Struktur ist zwar komplexer als unser Fall B, die Anwendungsfälle beschränken sich aber auf einzelne Polynome vom Grad fünf als Konturen.

Der Ansatz in diesem Kapitel löst auch ein Optimalsteuerungsproblem mit der Bearbeitungszeit als Zielfunktion. Allerdings hat es, wie auch die beschleunigungsbeschränkte Bewegungsplanung, nur eine Steuergröße. Die Lösung ist dann auch nicht das globale Minimum im Sinne des vorangegangenen Absatzes, sondern die zeitoptimale Bewegung für

bestimmte Bahnen der Grob- und Feinachsen, die über einen gemeinsamen Parameter verknüpft sind.

Es folgt die konkrete Formulierung des Problems für den zeitkontinuierlichen Fall. Als Grundlage dient eine gültige, beschleunigungsbegrenzte Achsbewegung gemäß Definition 2.8. Für den Algorithmus im folgenden Abschnitt 5.2 soll sie außerdem folgender Definition genügen.

Definition 5.1. *Sei* $\mathbf{q}(t) = [x_c(t), y_c(t), x_f(t), y_f(t)]^T$ *eine Bewegung von Kinematik B, die Definition 2.8 erfüllt. Wir nennen sie* stückweise glatt, *wenn es eine endliche Menge*

$$\mathbf{t}' = \{t'_1, \ldots, t'_{n'}\} \subset [t_a, t_b], \qquad t_0 = t'_1 < \ldots < t'_{n'} = t_f,$$

gibt, sodass auf den Intervallen (t'_i, t'_{i+1}), $i = 1, \ldots, n'-1$, *die Beschleunigungen aller Achsen* C^∞ *sind und sodass an jedem* t'_i *wenigstens eine Achsbewegung* $w(t)$ *eine der folgenden Bedingungen erfüllt*[1]*:*

1. *Achsbewegung hat Sprung in der Beschleunigung:*

$$\lim_{t \nearrow t'_i} a_w(t) \neq \lim_{t \searrow t'_i} a_w(t).$$

2. *Achsbewegung hat isolierte Nullstelle in Geschwindigkeit:*

$$\left(v_w(t'_i) = 0\right) \wedge \left(\exists\, \delta > 0 :\ \forall\, \delta' \in (0, \delta) :\ \exists t \in (t'_i - \delta', t'_i + \delta') :\ v_w(t) \neq 0\right). \tag{5.1}$$

3. *Bahngeschwindigkeit hat eine Nullstelle,* $v_b(t'_i) = 0$.

Die Folge $\mathbf{t}'$ *enthält die* besonderen Punkte *der Bewegung.*

Bedingung (5.1) bedeutet anschaulich, dass sich von der Nullstelle mindestens in eine Richtung über t die Achsgeschwindigkeit verändert. Die Nullstellen in einem offenen Intervall, in dem die Achsgeschwindigkeit null ist, deckt die Bedingung bewusst nicht ab, denn ansonsten hätte man keine Garantie, dass die Menge $\mathbf{t}'$ endlich ist. Bei den Nullstellen der Bahngeschwindigkeit gehen wir ohnehin davon aus, dass sie nur an diskreten Punkten auftreten.

Die Lösung des nachfolgenden Optimalsteuerungsproblems ist eine bijektive, stetige, monoton wachsende Funktion

$$\varphi : [\tau_0, \tau_f] \ni \tau \longmapsto \varphi(\tau) \in [t_0, t_f],$$

sodass $\tau_f - \tau_0$ kleinstmöglich ist. Sie ist die schon erwähnte Umparametrisierung im Zeitbereich und die Komposition von Achstrajektorie und $\varphi(\tau)$, wie etwa

$$x_c \circ \varphi : [\tau_0, \tau_f] \ni \tau \longmapsto (x_c \circ \varphi)(\tau) = x_c(\varphi(\tau)) \in \mathbb{R}$$

[1] w ist dabei die Bezeichnung für eine der Achsen x_c, y_c, x_f und y_f und deren Achsgrenzen sind entsprechend $\widehat{A}_w$ und $\widehat{V}_w$.

für die x_c–Achse, ist die neue, optimierte Trajektorie der Achse. Die Variable τ ist hier sozusagen der „neue“ Zeitparameter.

Natürlich müssen die neuen Achstrajektorien den Restriktionen (2.4) genügen. Das ergibt mit der neuen Achsgeschwindigkeit und -beschleunigung,

$$v_{xc}(\tau) = \frac{\mathrm{d}(x_c \circ \varphi)}{\mathrm{d}\tau}(\tau) \quad \text{und} \quad a_{xc}(\tau) = \frac{\mathrm{d}^2(x_c \circ \varphi)}{\mathrm{d}\tau^2}(\tau),$$

am Beispiel der x_c–Achse, für jedes $\tau \in [\tau_0, \tau_f]$ an $\varphi(\tau)$ die Bedingungen

$$\widehat{V}_{xc} \geq |v_{xc}(\tau)| = \left| \frac{\mathrm{d}x_c}{\mathrm{d}t}(\varphi(\tau)) \cdot \frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau) \right|, \tag{5.2}$$

$$\widehat{A}_{xc} \geq |a_{xc}(\tau)| = \left| \frac{\mathrm{d}^2x_c}{\mathrm{d}t^2}(\varphi(\tau)) \cdot \left(\frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau) \right)^2 + \frac{\mathrm{d}x_c}{\mathrm{d}t}(\varphi(\tau)) \cdot \frac{\mathrm{d}^2\varphi}{\mathrm{d}\tau^2}(\tau) \right|. \tag{5.3}$$

Hier muss man wieder die Aspekte in Bemerkung 2.3 berücksichtigen. Die Achsgeschwindigkeit $v_{xc}(t) = \frac{\mathrm{d}}{\mathrm{d}t} x_c(t)$ muss an den besonderen Punkten nicht differenzierbar sein, allerdings existieren dort links- und rechtsseitige Grenzwerte der Ableitung. Deswegen muss (5.3) für beide Grenzwerte erfüllt sein. Damit muss $a_{xc}(\tau)$ nicht stetig sein und daher sind auch Sprünge in $\frac{\mathrm{d}^2}{\mathrm{d}\tau^2}\varphi(\tau)$ erlaubt. Insbesondere ist $\frac{\mathrm{d}}{\mathrm{d}\tau}\varphi(\tau)$ nicht überall differenzierbar. Damit die neue Bahngeschwindigkeit

$$v_b(\tau) = \frac{\mathrm{d}(s_b \circ \varphi)}{\mathrm{d}\tau}(\tau) = v_b(\varphi(\tau)) \cdot \frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau) \tag{5.4}$$

die Grenze $\hat{v}_b$ einhält, muss

$$\frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau) \leq \frac{\hat{v}_b}{|v_b(\varphi(\tau))|} \tag{5.5}$$

gelten. Ist die Umparametrisierung $\varphi(\tau)$ streng monoton wachsend[2], gilt zudem $0 \geq \frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau)$. Außerdem existiert dann die Inverse

$$\tau_b : [t_0, t_f] \ni t \longmapsto \tau_b(t) = \varphi^{-1}(t) \in [\tau_0, \tau_f]$$

und wir schreiben

$$\lambda_b(t) := \frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau_b(t)), \tag{5.6}$$

$$\mu_b(t) := \frac{\mathrm{d}^2\varphi}{\mathrm{d}\tau^2}(\tau_b(t)). \tag{5.7}$$

Da $\frac{\mathrm{d}}{\mathrm{d}\tau}\varphi(\tau)$ nicht überall differenzierbar ist, sondern an bestimmten Stellen nur links- und rechtsseitige Grenzwerte der Ableitung besitzt, müssen an diesen Stellen für (5.7) zwei Fälle

[2]Das ist bei einer zeitoptimalen Umparametrisierung, die wir später betrachten, der Fall.

unterschieden werden (siehe unten). Für die rechten Seiten von (5.2) und (5.3) ergibt sich die t–Parametrisierung[3],

$$\widehat{V}_{xc} \geq |\dot{x}_c(t)\ \lambda_b(t)|\,, \tag{5.8}$$

$$\widehat{A}_{xc} \geq \left|\ddot{x}_c(t)\ \lambda_b(t)^2 + \dot{x}_c(t)\ \mu_b(t)\right|. \tag{5.9}$$

Zwischen (5.6) und (5.7) besteht an den Stellen t, wo $\lambda_b(t)^2$ differenzierbar ist, der Zusammenhang[4]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda_b(t)^2\right) = 2\mu_b(t). \tag{5.10}$$

Das nutzt der Algorithmus im folgenden Abschnitt aus.

Auch hier erfordert die Möglichkeit zum Abbremsen auf der Kontur zusätzliche Bedingungen an $\varphi(\tau)$. Die t–Parametrisierung der Summenbeschleunigung in x–Richtung nach der Umparametrisierung,

$$\ddot{x}(t)\ \lambda_b(t)^2 + \dot{x}(t)\ \mu_b(t), \tag{5.11}$$

hat nach Einsetzen von (2.6) für $\dot{x}(t)$ und (2.7) für $\ddot{x}(t)$ die Form

$$\left[x''(s_b(t))\ v_b(t)^2 + x'(s_b(t))\ a_b(t)\right]\ \lambda_b(t)^2 + \left[x'(s_b(t))\ v_b(t)\right]\ \mu_b(t).$$

Die Bedingung an $\lambda_b(t)^2$ ist dann (vergleiche (2.11))

$$\left|x''(s_b(t))\ v_b(t)^2\ \lambda_b(t)^2\right| \leq K_\mathrm{b}\left(\widehat{A}_{xc} + \widehat{A}_{xf}\right). \tag{5.12}$$

Sie lässt der Summenbewegung immer genügend Reserve, um auf der Bahn abzubremsen.

Auf bestimmten Abschnitten der Bewegung ist sofort klar, wie die Umparametrisierung aussehen muss. Das sind die Bereiche über t, wo sich der Werkzeugkopf schon mit dem programmierten Vorschub $\hat{v}_b$ bewegt, wo eine Achse mit ihrer vollen Beschleunigung fährt oder wo

$$\left|x''(s_b(t))\ v_b(t)^2\right| = K_\mathrm{b}\left(\widehat{A}_{xc} + \widehat{A}_{xf}\right)$$

gilt. Dort kann durch eine Umparametrisierung die Bearbeitungszeit nicht weiter reduziert werden und $\varphi(\tau)$ ist die Identität mit konstanter Ableitung eins. Für das Verfahren sind also nur Bereiche relevant, bei denen keiner dieser Fälle auftritt. Sie sollen *B–Bereiche* heißen. Auf alle diese Abschnitte kann der Algorithmus separat angewendet werden. Wir nehmen daher im Folgenden an, dass das Intervall $[t_0, t_f]$ der „alten“ Zeit gerade zu einem

[3] Das ist analog zu der Schreibweise der Bahngeschwindigkeit $v_b(s)$ und -beschleunigung $a_b(s)$ über dem Bahnparameter s in dem Algorithmus der Bewegungsplanung in Abschnitt 2.2.3.

[4] Das folgt wie in (2.10) auf Seite 24.

B–Bereich gehört und wir nur diesen behandeln. An deren Rändern muss $\varphi(\tau)$ wegen ihrer Bijektivität

$$\varphi(\tau_0) = t_0 \qquad \text{und} \qquad \varphi(\tau_f) = t_f \tag{5.13}$$

erfüllen, sowie

$$\frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau_0) = \lambda_b(t_0) = \lambda_{b,0} = 1 \qquad \text{und} \qquad \frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau_f) = \lambda_b(t_f) = \lambda_{b,f} = 1. \tag{5.14}$$

Letzteres folgt, da φ für die gesamte Kontur stetig sein muss und dort, wo kein B–Bereich ist, die Umparametrisierung gleich der Identität ist. Lediglich an Stellen der Gesamtbewegung, an denen alle Achsen stehen und die in einem B–Bereich liegen, gelten andere Bedingungen. Genaueres dazu folgt weiter unten in Abschnitt 5.2.3.

Insgesamt ergibt sich daraus ein Optimalsteuerungsproblem über t, ähnlich wie in Abschnitt 2.2.3. Die Steuergröße ist $\mu_b(t)$ und die Zustandsgröße $\lambda_b(t)^2$. Beide sind über die dynamische Gleichung (5.10) miteinander verknüpft. Die Ungleichungen (5.8) und (5.9) für jede der vier Achsen, sowie (5.5) und (5.12) für x– und y–Komponente in der t–Parametrisierung ergeben die Nebenbedingungen. Die Gleichungen (5.13) und (5.14) sind die Randbedingungen. Aus der Differentialgleichung

$$\lambda_b(\varphi(\tau)) = \frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau)$$

mit der Anfangsbedingung $\varphi(\tau_0) = t_0$ folgt

$$\tau_f - \tau_0 = \int_{t_0}^{t_f} \frac{1}{\lambda_b(t)}\mathrm{d}t = \int_{\varphi(\tau_0)}^{\varphi(\tau_f)} \frac{1}{\lambda_b(t)}\mathrm{d}t$$

und das ist das zu minimierende Zielfunktional. Das gesuchte $\varphi(t)$ ist die Lösung der Differentialgleichung.

5.2. Lösungsalgorithmus

Das Verfahren zur Lösung des Optimalsteuerungsproblems setzt sich aus zwei Phasen zusammen und basiert auf dem Algorithmus zur zeitoptimalen beschleunigungsbegrenzten Bewegungsplanung in [21]. Die erste ermittelt eine Funktion $\lambda_b^{\downarrow}(t)^2$, in Folgenden mit *Grenzkurve* bezeichnet, als obere Begrenzung für alle sogenannten *Bahntrajektorien* $\lambda_b(t)^2$. Die zweite Phase berechnet daraus das zum optimalen $\varphi(t)$ gehörende $\lambda_b(t)^2$.

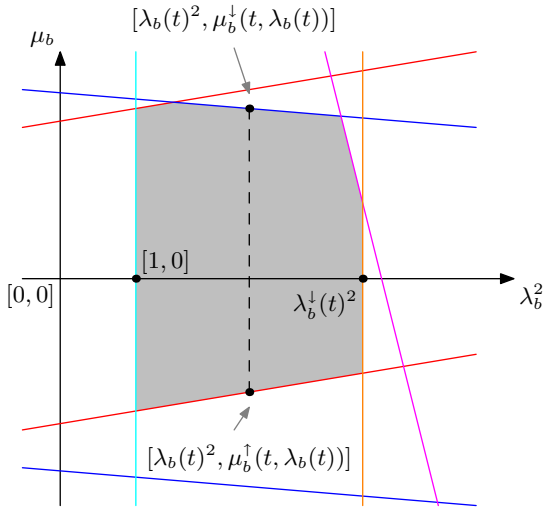

Abbildung 5.1.: Beispiel für ein Geradendiagramm.

5.2.1. Phase I: Die Grenzkurve

Aus den Nebenbedingungen ergibt sich in Phase eins für jedes $t \in [t_0, t_f] \setminus \mathbf{t}'$ ein Polygon[5] in der λ_b^2–μ_b–Ebene, das die erlaubten Kombinationen von Steuer- und Zustandsgrößen enthält.

Ungleichung (5.9) sowie die Analoga der anderen Achsen ergeben vier Paare jeweils paralleler Geraden. Zwei weitere vertikale Parallelen sind durch $\lambda_b(t)^2 = 1$ und das Minimum für $\lambda_b(t)^2$ definiert, das aus den Ungleichungen (5.2), (5.5) und (5.12) hervorgeht. Der Schnitt der Zwischenräume dieser fünf Parallelenpaare ist das erwähnte Polygon. Die λ_b^2–Komponente eines am weitesten rechts liegenden Punktes ist der Wert für $\lambda_b^{\downarrow}(t)^2$. Abbildung 5.1 zeigt ein Beispiel für ein solches *Geradendiagramm*, der Ausschnitt enthält aber nicht alle Geraden. Es müssen außerdem nicht immer alle Geradengleichungen existieren. Sind die Geschwindigkeit und die Beschleunigung einer Achse null, entfallen die Parallelen aus (5.8) und (5.9). Bei verschwindendem $x''(s_b(t))$ gibt es die Gerade aus (5.12) nicht. Da wir die besonderen Punkte aus $\mathbf{t}'$ hier noch nicht betrachten, ist die Bahngeschwindigkeit nicht null und deswegen $\lambda_b^{\downarrow}(t)^2$ ein endlicher Wert.

Lemma 5.2. *Die Grenzkurve $\lambda_b^{\downarrow}(t)^2$ ist auf allen offenen Intervallen (t_i', t_{i+1}') stetig.*

Beweis. Auf dem Intervall (t_i', t_{i+1}') sind alle Achsbeschleunigungen und -geschwindigkeiten stetig. Daher ist der aus (5.5) hervorgehende maximale Verlauf von $\lambda_b(t)^2$ stetig und divergiert höchstens an den Rändern. Der Verlauf aus (5.12) ist stetig, bis auf die Stellen oder Bereiche, wo $x''(s_b(t)) = 0$ gilt. Dort ist er nicht definiert, an den Rändern dieser Bereiche geht der Verlauf von $\lambda_b(t)^2$ gegen unendlich. Ähnliches gilt für den Verlauf aus (5.2), der dort nicht definiert ist, wo die Achsgeschwindigkeit verschwindet.

[5] Hier ist mit Polygon der Inhalt des Polygonzuges sowie dessen Rand gemeint.

Seien nun w_a und w_b zwei beliebige der vier Achsen, deren Geschwindigkeit nicht auf ganz (t'_i, t'_{i+1}) verschwindet. Dann folgt aus

$$\begin{aligned} \widehat{A}_{wa} &= \ddot{w}_a(t)\ \lambda_b(t)^2 + \dot{w}_a(t)\ \mu_b(t), \\ \pm\widehat{A}_{wb} &= \ddot{w}_b(t)\ \lambda_b(t)^2 + \dot{w}_b(t)\ \mu_b(t) \end{aligned}$$

für die erste Komponente des Schnittpunktes die Beziehung

$$\lambda_b(t)^2 = \frac{\widehat{A}_{wa}\dot{w}_b(t) \mp \widehat{A}_{wb}\dot{w}_a(t)}{\ddot{w}_a(t)\dot{w}_b(t) - \ddot{w}_b(t)\dot{w}_a(t)}. \tag{5.15}$$

Dieser Verlauf ist auf offenen Teilintervallen von (t'_i, t'_{i+1}) definiert und dort stetig. An deren Rändern divergiert er gegen unendlich, nur bei t'_i und t'_{i+1} muss das nicht sein. Beim Zusammenstellen der einzelnen Verläufe zu $\lambda_b^{\downarrow}(t)^2$ ensteht eine auf ganz (t'_i, t'_{i+1}) definierte, stetige Grenzkurve. □

Lemma 5.3. *Gibt es mindestens eine Achse w_a mit $v_{wa}(t'_i) \neq 0$, dann konvergiert die Grenzkurve $\lambda_b^{\downarrow}(t)^2$ für $t \to t'_i$ gegen einen endlichen, positiven Wert.*

Diese Aussage ergibt sich direkt aus der rechten Seite von Ungleichung (5.8). Bewegt sich in der Referenzbewegung $\mathbf{q}(t)$ an den Stellen t'_i und t'_{i+1} also mindestens eine Achse, dann ist die Grenzkurve auf $[t'_i, t'_{i+1}]$ beschränkt.

Bemerkung 5.4. *Für den Rest dieses Kapitels nehmen wir an, dass es für jedes (t'_i, t'_{i+1}) eine (endliche) Menge*

$$\Theta(i) = \left\{\theta_{i,1}, \ldots, \theta_{i,n(i)}\right\}, \qquad t'_i = \theta_{i,1} < \ldots < \theta_{i,n(i)} = t'_{i+1},$$

gibt, sodass die Grenzkurve $\lambda_b^{\downarrow}(t)^2$ auf jedem $(\theta_{i,j}, \theta_{i,j+1})$ nur aus einer der obigen Gleichungen und einer Achse hervorgeht. Auf diesen offenen Intervallen ist die Grenzkurve differenzierbar.

Obwohl die Grenzkurve an den besonderen Punkte aus Definition 5.1 begrenzt ist, müssen für die Werte genauere Untersuchungen gemacht werden. Ist bei t'_i eine Nullstelle in der Geschwindigkeit der w_a–Achse, ohne dass die Beschleunigung springt oder verschwindet, ergeben sich daraus mit (5.3) ein Paar vertikaler Geraden und deren Zwischenraum durch

$$-\frac{\widehat{A}_{wa}}{|\ddot{w}_a(t'_i)|} \leq \lambda_b(t'_i)^2 \leq \frac{\widehat{A}_{wa}}{|\ddot{w}_a(t'_i)|},$$

wobei natürlich nur die rechte Ungleichung für das Polygon relevant ist. Betrachtet man den Schnittpunkt der Geraden dieser Achse mit einer anderen, w_b, die dort keine Nullstelle hat, in einem genügend kleinen Intervall $(t'_i - \delta, t'_i)$, dann konvergiert dieser Schnittpunkt

gegen $\widehat{A}_{wa}/|\ddot{w}_a(t'_i)|$. Das folgt mit der Formel für den Schnittpunkt aus dem Beweis von Lemma 5.2 mit

$$\lim_{t \nearrow t'_i} \left(\frac{\widehat{A}_{wa}\dot{w}_b(t) \mp \widehat{A}_{wb}\dot{w}_a(t)}{\ddot{w}_a(t)\dot{w}_b(t) - \ddot{w}_b(t)\dot{w}_a(t)} \right) = \frac{\widehat{A}_{wa}}{\ddot{w}_a(t'_i)}.$$

Dieser Schnittpunkt hängt vom Vorzeichen von $\ddot{w}_a(t'_i)$ ab. Im Fall $\ddot{w}_a(t'_i) < 0$ muss man den Grenzwert des Schnittpunktes mit der Geraden $-\widehat{A}_{wa} = \ddot{w}_a(t)\ \lambda_b(t)^2 + \dot{w}_a(t)\ \mu_b(t)$ betrachten. Das ergibt $-\widehat{A}_{wa}/\ddot{w}_a(t'_i)$.

Bei einem Beschleunigungssprung bei t'_i ist das Parallelenpaar nicht eindeutig, es gibt ein linkes und ein rechtes, das jeweils mit dem links- beziehungsweise rechtsseitigen Grenzwert der Achsbeschleunigung entsteht. Beide haben einen größten gültigen Wert für $\lambda_b(t'_i)^2$ und der kleinere davon ist $\lambda_b^{\downarrow}(t'_i)^2$. Ist ein linksseitiger Grenzwert für $\lambda_b^{\downarrow}(t'_i)^2$ verantwortlich, dann ist die Grenzkurve auf $(t'_{i-1}, t'_i]$ stetig. Entsprechend ist sie im anderen Fall auf $[t'_i, t'_{i+1})$ stetig. Ergeben beide Seiten denselben Wert für die Grenzkurve, ist sie sogar auf (t'_{i-1}, t'_{i+1}) stetig.

Das Ergebnis der ersten Phase ist also eine Funktion

$$\lambda_b^{\downarrow} : [t_0, t_f] \ni t \longmapsto \lambda_b^{\downarrow}(t) \in \mathbb{R}.$$

Sie ist nicht stetig. Die Sprünge können nur an den Stellen $\mathbf{t}'$ liegen. Gibt es in $[t_0, t_f]$ keine Stellen, an denen alle Achsen stehen, so ist die Grenzkurve beschränkt. Das setzen wir im Folgenden voraus. Den anderen Fall betrachtet Abschnitt 5.2.3.

5.2.2. Phase II: Die optimale Bahntrajektorie

Zur Vorbereitung der zweiten Phase müssen zwei Überlegungen durchgeführt werden. Zuerst geht es um die möglichen Verläufe von $\lambda_b(t)^2$ unterhalb der Grenzkurve $\lambda_b^{\downarrow}(t)^2$. Phase zwei soll für jedes $t \in [t_0, t_f]$ einen größtmöglichen Wert erzeugen, denn das führt wegen (5.4) zu einer maximalen Bahngeschwindigkeit und damit auch zur gesuchten zeitoptimalen Umparametrisierung. Die Funktion $\lambda_b(t)^2$ wächst über t umso schneller, je größer $\mu_b(t)$ ist. Das folgt aus (5.10). Für festes t und festes $\lambda_b(t)^2$ sind die möglichen Werte für $\mu_b(t)$ ein Intervall. Dessen obere Grenze ist die maximale zweite Komponente des Punktes im Polygon des Geradendiagramms für t, deren erste Komponente $\lambda_b(t)^2$ ist (siehe Abbildung 5.1). Wir bezeichnen sie mit $\mu_b^{\downarrow}(t, \lambda_b(t))$, da sie von den Werten für t und $\lambda_b(t)^2$ abhängt. Analog ergibt sich $\mu_b^{\uparrow}(t, \lambda_b(t))$ für die untere Grenze. Für beide Grenzen ist jeweils mindestens eine Gerade relevant und damit auch eine Achse.

Sei nun $\tilde{t} \in (t'_i, t'_{i+1})$. Gehört $[\lambda_b(\tilde{t})^2, \mu_b^{\downarrow}(\tilde{t}, \lambda_b(\tilde{t}))]$ zu einem gültigen $\varphi(\tau)$, dann ist bei $\tilde{\tau}$ mit $\tilde{t} = \varphi(\tilde{\tau})$ nach der Umparametrisierung mindestens eine Achse an ihrer Beschleunigungsgrenze. Diese Situation muss dann nicht auftreten, wenn $\lambda_b(t)^2$ in einer Umgebung von $\tilde{t}$ auf der Grenzkurve verläuft und diese dort aus den Ungleichungen (5.2), (5.5) oder (5.12) hervorgeht. Nehmen wir daher an, dass sich $\lambda_b(\tilde{t})^2$ unterhalb der Grenzkurve aufhält. Da sich auf (t'_i, t'_{i+1}) die Polygone aus den Geradendiagrammen stetig ändern, gibt

es ein $\delta > 0$, sodass das optimale $\lambda_b(t)^2$ auf $[\tilde{t}, \tilde{t} + \delta]$ mit einer Achse w an ihrer Grenze entsteht[6]. Auf diesem Intervall ist $\lambda_b(t)^2$ die Lösung einer Differentialgleichung. Zunächst gilt

$$\begin{aligned} & \pm\widehat{A}_w = \ddot{w}(t)\ \lambda_b(t)^2 + \dot{w}(t)\ \mu_b(t) \\ \iff \quad & \pm\widehat{A}_w\ \dot{w}(t) = \dot{w}(t)\ \ddot{w}(t)\ \lambda_b(t)^2 + \dot{w}(t)^2\ \mu_b(t) \\ & = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(\dot{w}(t)^2\ \lambda_b(t)^2\right). \end{aligned}$$

Durch Integration der letzten Gleichung folgt

$$\begin{aligned} & \pm\widehat{A}_w \int_{\tilde{t}}^{t} \dot{w}(\theta)\ \mathrm{d}\theta = \frac{1}{2}\dot{w}(t)^2\ \lambda_b^2(t) \\ \iff \quad & \pm\widehat{A}_w w(t) + K_\lambda = \frac{1}{2}\dot{w}(t)^2\ \lambda_b^2(t), \qquad t \in (\tilde{t}, \tilde{t} + \delta]. \end{aligned}$$

Damit ergibt sich letztendlich

$$\lambda_b(t)^2 = 2\ \frac{\pm\widehat{A}_w w(t) + K_\lambda}{\dot{w}(t)^2}, \qquad t \in (\tilde{t}, \tilde{t} + \delta], \tag{5.16}$$

wobei die Konstante K_λ wegen der Anfangsbedingung bei $\tilde{t}$ den Wert

$$K_\lambda = \frac{1}{2}\lambda_b(\tilde{t})^2\ \dot{w}(\tilde{t})^2 \mp \widehat{A}_w\ w(\tilde{t}) \tag{5.17}$$

hat. Die Bahntrajektorie $\lambda_b(t)^2$ aus der vorletzten Gleichung erfüllt

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda_b(t)^2\right) = 2\ \mu_b^{\downarrow}(t, \lambda_b(t)), \qquad t \in (\tilde{t}, \tilde{t} + \delta].$$

Ein optimales $\lambda_b(t)^2$ ergibt sich stückweise mit Formel (5.16), solange es unterhalb der Grenzkurve $\lambda_b^{\downarrow}(t)^2$ verläuft. Stößt es auf die Grenzkurve, dann sind mehrere Möglichkeiten für den weiteren Verlauf möglich, mehr dazu folgt weiter unten. Natürlich kann die relevante Achse wechseln, auf jedem Stück ist eine andere in (5.16) aktiv. Genauso ist es mit dem Vorzeichen der Beschleunigungsgrenze in (5.16) und (5.17). Eines davon führt zum schnellstmöglichen Aufbau von $\lambda_b(t)^2$, das andere zum steilsten Abstieg. Gelten für eine Phase des Aufbaus beispielsweise $\ddot{w}(t) > 0$ und $\dot{w}(t) > 0$, ist das $+\widehat{A}_w$, im Fall $\ddot{w}(t) > 0$ und $\dot{w}(t) < 0$ aber $-\widehat{A}_w$.

Auch die besonderen Stellen t_i' kann man mit diesem Vorgehen behandeln. Dabei ist nur der Fall gesondert zu betrachten, in dem die Geschwindigkeit der relevanten Achse null

[6] w ist wie in Definition 5.1 eine der vier Achsen von Kinematik B.

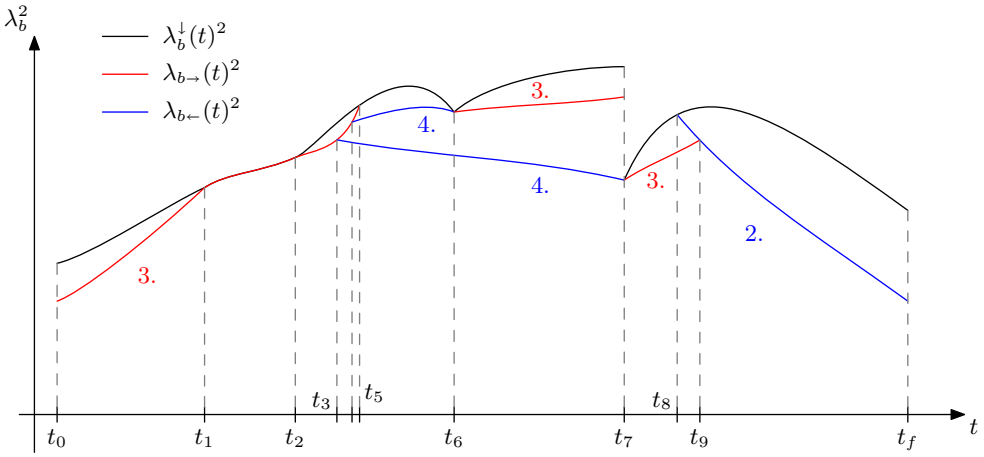

Abbildung 5.2.: Ein Ergebnis aus Algorithmus 5.1 hat eine derartige Struktur. Die Nummern sind die Schritte des Algorithmus, die jeweils für die Berechnung von $\lambda_b(t)^2$ benutzt wurden. Die roten Linien sind die von links nach rechts entstehenden Stücke der Bahntrajektorie, die blauen entsprechend die von rechts nach links entstehenden.

ist. Gilt $\dot{w}(\tilde{t}) = 0$, ist $\lambda_b(\tilde{t})^2$ aus (5.16) nicht definiert. Allerdings folgt für den Grenzwert mit der Regel von L'Hospital ([2, Seite 346])

$$\lim_{t \searrow \tilde{t}} \lambda_b(t)^2 = \lim_{t \searrow \tilde{t}} \left(\pm 2 \widehat{A}_w \frac{\dot{w}(t)}{2\dot{w}(t)\ddot{w}(t)} \right) = \pm \widehat{A}_w \frac{1}{\ddot{w}(\tilde{t})},$$

falls $\ddot{w}(\tilde{t}) \neq 0$ gilt.

Dieses Vorgehen, basierend auf $\mu_b^{\downarrow}(t)$, erzeugt das schnellstmöglichste Wachstum von $\lambda_b(t)^2$ von links nach rechts über der t–Achse. Analoge Überlegungen gibt es für das größtmögliche Wachstum mit $\mu_b^{\uparrow}(t)$ von rechts nach links. Diese Verläufe sinken natürlich schnellstmöglich, wenn man sie von links nach rechts betrachtet. In diesem Fall gilt für differenzierbare Stellen der Bahntrajektorie die Beziehung

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\lambda_b(t)^2 \right) = 2 \, \mu_b^{\uparrow}(t, \lambda_b(t)).$$

Im Folgenden bezeichnen wir ein Stück der Bahntrajektorie, das von links nach rechts größtmögliche Steigung hat, mit $\lambda_{b\rightarrow}(t)^2$ und entsprechend eines, das kleinstmögliche Steigung hat mit $\lambda_{b\leftarrow}(t)^2$. Insgesamt sind gerade diese beiden Arten von Verläufen gesucht, denn $\lambda_b(t)^2$ verhält sich optimal, wenn es — grob gesagt — schnellstmöglich steigt, lange auf einem hohen Niveau bleibt und zuletzt schnellstmöglich sinkt. Es sei hier noch angemerkt, dass weder $\mu_b^{\downarrow}(t) \geq 0$ noch $\mu_b^{\uparrow}(t) \leq 0$ gelten muss und dass beide Funktionen nicht stetig sind.

Bemerkung 5.5. *Im Folgenden nehmen wir an, dass es eine endliche Menge*

$$\left\{\theta'_1, \ldots, \theta'_{n(\mu)}\right\}, \quad t_0 = \theta'_1, \quad t_f = \theta'_{n(\mu)},$$
$$\theta'_i < \theta'_{i+1}, \qquad i = 1, \ldots, n(\mu) - 1,$$

gibt, sodass die Funktionen $\mu_b^{\downarrow}(t, \lambda_b^{\downarrow}(t)^2)$ *und* $\mu_b^{\uparrow}(t, \lambda_b^{\downarrow}(t)^2)$ *auf den offenen Intervallen* $(\theta'_i, \theta'_{i+1})$ *aus jeweils einer Achse hervorgehen.*

Bei der zweiten Überlegung spielen Eigenschaften der Grenzkurve $\lambda_b^{\downarrow}(t)^2$ eine entscheidende Rolle. Jeder Punkt darauf kann eine *Trajektoriensenke* und/oder eine *Trajektorienquelle* sein, wie weiter unten begründet wird. Trajektorienquelle bei $t \in [t_0, t_f]$ bedeutet, dass vom Punkt $[t, \lambda_b^{\downarrow}(t)^2]^T$ nach rechts ein zulässiger Verlauf $\lambda_b(t)^2$ bis zu einem gewissen $t + \delta$, $\delta > 0$, existiert. Zulässig bedeutet hier, dass mit dieser Bahntrajektorie keine der Nebenbedingungen verletzt werden. Entsprechend ist bei t eine Senke, wenn es von dort nach links einen zulässigen Verlauf gibt. Oder anders gesagt: Es gibt ein von links kommendes, zulässiges $\lambda_b(t)^2$, das von unten in die Grenzkurve $\lambda_b^{\downarrow}(t)^2$ stößt oder auf ihr verläuft.

An Stellen $t \in [t_0, t_f]$, an denen $\lambda_b^{\downarrow}(t)^2$ stetig ist, existieren links- und rechtsseitige Grenzwerte für die Ableitung. Gilt

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda_b^{\downarrow}(t^-)^2\right) \leq 2\mu_b^{\downarrow}(t^-, \lambda_b^{\downarrow}(t)), \tag{5.18}$$

dann ist bei t eine Senke, im Fall

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda_b^{\downarrow}(t^+)^2\right) \geq 2\mu_b^{\uparrow}(t^+, \lambda_b^{\downarrow}(t)), \tag{5.19}$$

liegt eine Quelle vor[7]. Die Konventionen in den Bemerkungen 5.4 und 5.5 ermöglichen diese Charakterisierung. Wir schließen hier auch den Fall ein, in dem einer der Grenzwerte unbeschränkt ist. Wie in der obigen Bemerkung 5.4 vereinbart, ist die Anzahl der Punkte, an denen

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda_b^{\downarrow}(t^-)^2\right) \neq \frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda_b^{\downarrow}(t^+)^2\right)$$

gilt, endlich. Von den ebenfalls nur endlich vorhandenen Sprungstellen der Grenzkurve $\lambda_b^{\downarrow}(t)^2$ kann in die Richtung des Sprungs (Das ist die Seite, die nicht für den Wert der Grenzkurve verantwortlich war, siehe oben.) immer ein gültiger Verlauf $\lambda_b(t)^2$ ausgehen; diese Stelle hat entsprechend die Quellen- beziehungsweise Senkeneigenschaft. Für eine Richtung mit stetiger Anbindung kann die Eigenschaft gemäß (5.18) oder (5.19) zugewiesen werden. Somit ist jeder Punkt aus $[t_0, t_f]$ Quelle, Senke oder beides.

Die bisherigen Überlegungen nutzt das Verfahren in der zweiten Phase. Es bestimmt den optimalen Verlauf von $\lambda_b(t)^2$. Die einzelnen Schritte zeigt Algorithmus 5.1. Anhand

[7] Die Definition der Begriffe Quelle und Senke unterscheidet sich hier von der in [21].

1. Wähle $\lambda_{b,l}^2 := \lambda_{b,0}^2$ und $t_l := t_0$.
   Wähle $\lambda_{b,r}^2 := \lambda_{b,f}^2$ und $t_r := t_f$.

2. Bilde von t_r aus mit dem Randwert $\lambda_{b,r}^2$ nach links die schnellstmöglich steigende Funktion $\lambda_{b\leftarrow}(t)^2$. Sie entsteht durch wechselnde Formeln (5.16), wenn $\lambda_{b\leftarrow}(t)^2$ unterhalb von $\lambda_b^{\downarrow}(t)^2$ verläuft, oder durch $\lambda_b^{\downarrow}(t)^2$. Dieser Prozess geht bis zu einem Punkt t^*, an dem einer der folgenden Fälle eintritt:
   - Bei einem Sprung in der Grenzkurve gilt $\lambda_{b\leftarrow}(t^*)^2 > \lambda_b^{\downarrow}(t^*)^2$.
   - $t^* = t_l$
   - Mit $\lambda_{b\leftarrow}(t^*)^2 = \lambda_b^{\downarrow}(t^*)^2$ ist die Grenzkurve keine Senke mehr.

3. Bilde von t_l aus mit dem Randwert $\lambda_{b,l}^2$ nach rechts die schnellstmöglich steigende Funktion $\lambda_{b\rightarrow}(t)^2$. Sie entsteht durch wechselnde Formeln (5.16), wenn $\lambda_{b\rightarrow}(t)^2$ unterhalb von $\lambda_b^{\downarrow}(t)^2$ verläuft, oder durch $\lambda_b^{\downarrow}(t)^2$. Dieser Prozess geht solange, bis bei einem t_m einer der folgenden Fälle eintritt:
   - Bei einem Sprung in der Grenzkurve gilt $\lambda_{b\rightarrow}(t_m)^2 > \lambda_b^{\downarrow}(t_m)^2$.
   - Mit $\lambda_{b\rightarrow}(t_m)^2 = \lambda_b^{\downarrow}(t_m)^2$ ist die Grenzkurve keine Quelle mehr.
   - $\lambda_{b\rightarrow}(t_m)^2$ stößt in die von rechts kommende Funktion $\lambda_{b\leftarrow}(t)^2$ aus Schritt 2. In diesem Fall endet das Verfahren, da $\lambda_b(t)^2$ auf ganz $[t_0, t_f]$ definiert ist.

4. Suche auf der Grenzkurve $\lambda_b^{\downarrow}(t)^2$ den nächstgrößeren Punkt t' nach t_m, der sowohl Senke als auch Quelle ist.

5. Bilde von t' wie in Schritt 2. nach links die schnellstmöglich steigende Funktion $\lambda_{b\leftarrow}(t)^2$, bis sie sich mit der von links kommenden Funktion schneidet.

6. Definiere $\lambda_{b,l} := \lambda_b^{\downarrow}(t')^2$ und $t_l := t'$ und gehe zu Schritt 3.

Algorithmus 5.1: Phase II.

von Abbildung 5.2 soll der Ablauf des Algorithmus erläutert werden. Nach der Festlegung der Randbedingungen entsteht zuerst mit Schritt zwei ein bei t_f startendes und nach links steilstmögliches $\lambda_{b\leftarrow}(t)^2$, das bei t_8 in die Grenzkurve stößt und nicht unterhalb von ihr oder auf ihr nach links weiterlaufen kann, da keine Senke vorliegt. Danach wird zunächst mit Schritt drei von t_0 nach rechts eine optimale Bahntrajektorie $\lambda_{b\rightarrow}(t)^2$ erzeugt. Sie verläuft sogar ein Stück auf $\lambda_b^{\downarrow}(t)^2$, verlässt diese dann bei t_2 und endet schließlich bei t_5. Die nächste Stelle auf der Grenzkurve, die sowohl Senke als auch Quelle ist, liegt bei t_6. Von dort entstehen Bahntrajektorien in beide Richtungen. Die nach rechts gehende bricht ab, weil die Grenzkurve einen Sprung nach unten macht. Da von t_7 nach links und rechts gültige Verläufe für $\lambda_b(t)^2$ ausgehen können, enstehen auch entsprechende Bahntrajektorienstücke. Dasjenige nach links schneidet sich mit dem Stück, das bei t_0 startet. Dasjenige nach rechts schneidet das von rechts kommende und somit ist das optimale $\lambda_b(t)^2$ komplett. Die vom Punkt $[t_6, \lambda_b^{\downarrow}(t_6)^2]^T$ ausgehenden Stücke spielen im Endergebnis keine Rolle.

Existenz und Zeitoptimalität

Es folgt eine Begründung, warum Algorithmus 5.1 eine über ganz $[t_0, t_f]$ definierte Bahntrajektorie $\lambda_b(t)^2$ erzeugt. Angenommen, Schritt zwei und der erste Durchlauf von Schritt drei wurden durchgeführt und es gibt keinen gemeinsamen Schnittpunkt der Bahntrajektorien. Dann sucht der Algorithmus in Schritt vier die nächstgrößere Stelle t', an der von der Grenzkurve nach links und rechts gültige Bahntrajektorien verlaufen können. Sie muss es geben, denn die Bahntrajektorie von links endet bei einem t_a in einer Senke auf der Grenzkurve, die keine Quelle ist. Auf der anderen Seite endet die Bahntrajektorie aus Schritt zwei in einem t_b in einer Quelle, die keine Senke ist. Dazwischen muss ein Punkt liegen, der sowohl Senke als auch Quelle ist, da jeder Punkt auf der Grenzkurve mindestens eine der beiden Eigenschaften besitzt. In Abbildung 5.2 gibt es beispielsweise mindestens zwei dieser Punkte, die zwischen den Stellen t_5 und t_8 liegen. Sie befinden sich bei t_6 und t_7.

Das zweite Argument in der Begründung ist die Tatsache, dass die von t' nach links laufende Bahntrajektorie $\lambda_{b\leftarrow}(t)^2$ in das von links kommende, schon berechnete Bahntrajektorienstück $\lambda_{b\rightarrow}(t)^2$ stößt. Zum einen kann $\lambda_{b\leftarrow}(t)^2$ nicht vor dem Zusammenstoßen in der Grenzkurve versinken, denn dann wäre dort eine Quelle und Schritt vier hätte nicht den nächstgrößten Punkt t' gefunden, der Senke und Quelle ist. Zum anderen kann $\lambda_{b\leftarrow}(t)^2$ nicht vor t_0 einfach aufhören, ohne in ein links von t' liegendes Bahntrajektorienstück zu stoßen, denn unterhalb der Grenzkurve kann $\lambda_{b\leftarrow}(t)^2$ immer weiter nach links gehen, ist dabei aber durch eins nach unten begrenzt. Da $\lambda_{b,l}^2 = 1$ gilt, muss es einen Schnittpunkt der beiden Bahntrajektorien geben. In Abbildung 5.2 liegen solche Schnittpunkte bei t_3 und t_4. Diese Argumentation kann man auf jedes Durchlaufen von Schritt vier im Algorithmus anwenden. So wie am Anfang dieses Absatzes begründet sich auch, dass irgendwann in Schritt drei der Algorithmus endet und er eine auf ganz $[t_0, t_f]$ definierte Bahntrajektorie $\lambda_b(t)^2$ ermittelt hat.

Die Bahntrajektorie aus Algorithmus 5.1 ergibt die zeitoptimale Umparametrisierung. Das folgt durch die Art und Weise, wie der Algorithmus arbeitet. Angenommen, eine

Umparametrisierung $\varphi^*(\tau)$ hat eine kürzere Zeitdauer. Dann gilt mit dem dazugehörigen $\lambda_b^*(t)^2$ die Ungleichung

$$\int_{t_0}^{t_f} \frac{1}{\lambda_b(t)} \mathrm{d}t \geq \int_{t_0}^{t_f} \frac{1}{\lambda_b^*(t)} \mathrm{d}t.$$

Das wiederum bedeutet, dass es ein $t^* \in [t_0, t_f]$ mit $\lambda_b^*(t^*)^2 > \lambda_b(t^*)^2$ gibt. Das kann aber nicht sein, denn das vom Algorithmus berechnete $\lambda_b(t)^2$ verläuft schon auf den höchstmöglichen Werten, die unter den Nebenbedingungen möglich sind.

5.2.3. Sonderfall: Alle Achsen stehen

Einen besondere Situation liegt an Stellen $t' \in \mathbf{t}'$ vor, an denen alle Achstrajektorien der Eingangsbewegung stehen. Zum einen divergiert dort die Grenzkurve, zum anderen ergeben sich für den ersten und den letzten B–Bereich andere Randbedingungen als (5.14).

Wir betrachten die Divergenz am Beispiel der Stelle t_f des letzten B–Bereichs. Dort haben alle Achsen eine Geschwindigkeit von null. Die Werte für die Grenzkurve aus den Ungleichungen (5.2), (5.5) und (5.12) divergieren für $t \nearrow t_f$. Der Schnittpunkt der Geraden zweier Achsen w_a und w_b aus Bedingung (5.3) ergibt sich mit Gleichung (5.15). Für $t \nearrow t_f$ gehen sowohl Zähler als auch Nenner gegen null. Den Grenzwert des gesamten Ausdrucks erhält man mit der Regel von L'Hospital. Es gilt

$$\lim_{t \nearrow t_f} \lambda_b(t)^2 = \lim_{t \nearrow t_f} \frac{\widehat{A}_{wa}\dot{w}_b(t) \mp \widehat{A}_{wb}\dot{w}_a(t)}{\ddot{w}_a(t)\dot{w}_b(t) - \ddot{w}_b(t)\dot{w}_a(t)} = \lim_{t \nearrow t_f} \frac{\widehat{A}_{wa}\ddot{w}_b(t) \mp \widehat{A}_{wb}\ddot{w}_a(t)}{\dddot{w}_a(t)\dot{w}_b(t) - \dddot{w}_b(t)\dot{w}_a(t)} = \infty,$$

falls $\widehat{A}_{wa}\ddot{w}_b(t_f) \mp \widehat{A}_{wb}\ddot{w}_a(t_f) \neq 0$. Tritt diese Divergenz für jeden Schnittpunkt auf, dann divergiert die Grenzkurve für $t \nearrow t_f$.

Für den Randwert $\lambda_{b,f}^2$ betrachtet man für jede Achse w den Wert $\widehat{A}_w / |\ddot{w}(t_f)|$. Der minimale von den definierten ergibt den Randwert. Er ist nur dann gleich eins, wenn bei den Eingangstrajektorien schon eine Achse bei t_f an ihrer Beschleunigungsgrenze ist.

Es folgt eine Begründung, warum der Algorithmus auch in diesem Fall eine vollständige Bahntrajektorie erzeugt. Der einzige Unterschied zur Argumentation des vorherigen Unterabschnittes ist, dass die Randbedingungen größer als eins sein können. Wir betrachten den Fall des rechten Randes t_f. Es ist hier nicht offensichtlich, dass die von links kommende Bahntrajektorie $\lambda_{b\rightarrow}(t)^2$ die von rechts kommende und bei $\lambda_{b,f}^2$ startende $\lambda_{b\leftarrow}(t)^2$ schneidet.

Mit $\lambda_{b,f}^2$ als Randwert bei t_f berechnet der Algorithmus nach links einen Verlauf $\lambda_{b\leftarrow}(t)^2$ mit möglichst großer Steigung, das heißt, $\frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda_{b\leftarrow}(t)^2\right)$ soll so klein wie möglich sein. Das geht bis zu einer bestimmten Stelle mit Formel (5.16) für die entsprechende Achse w_a und das passende Vorzeichen von $\widehat{A}_{wa}$. Dieser Bereich soll das Intervall $[t_f - \delta, t_f]$, $\delta > 0$, enthalten. Dann können wir annehmen, dass $\lambda_{b\leftarrow}(t)^2$ auf diesem Intervall begrenzt ist, dass darin kein besonderer Punkt liegt und dass die Achsbeschleunigung $\ddot{w}_a(t)$ ihr Vorzeichen nicht ändert und bei t_f nicht verschwindet. Angenommen, die von links kommende Bahntrajek-

torie $\lambda_{b\rightarrow}(t)^2$ existiert bei $t_f - \delta$ noch und ist nicht zuvor mit $\lambda_{b\leftarrow}(t)^2$ zusammengestoßen, denn dann wäre die Bahntrajektorie schon vollständig. Es gilt also

$$\lambda_{b\rightarrow}(t_f - \delta)^2 < \lambda_{b\leftarrow}(t_f - \delta)^2.$$

Wir betrachten nun den Verlauf von $\lambda_{b\rightarrow}(t)^2$ auf $[t_f - \delta, t_f)$ wenn er mit Formel (5.16) für w_b als eine der vier Achsen entsteht. Die Achsgeschwindigkeit $\dot{w}_b(t)$ hat auf $(t_f - \delta, t_f)$ keinen Vorzeichenwechsel. Im Fall $\dot{w}_b(t) > 0$ folgt $\ddot{w}_b(t) < 0$ und für die maximale Steigung von $\lambda_{b\rightarrow}(t)^2$ ist Formel (5.16) mit $+\widehat{A}_{wb}$ verantwortlich. Das ergibt

$$\lambda_{b\rightarrow}(t)^2 = 2\frac{\widehat{A}_{wb}w_b(t) + \frac{1}{2}\lambda_{b\rightarrow}(t_f - \delta)^2\dot{w}_b(t_f - \delta)^2 - \widehat{A}_{wb}w_b(t_f - \delta)}{\dot{w}_b(t)}, \qquad t \in (t_f - \delta, t_f).$$

Der Nenner dieses Ausdrucks geht gegen null. Der Zähler

$$\widehat{A}_{wb}\left(w_b(t) - w_b(t_f - \delta)\right) + \frac{1}{2}\lambda_{b\rightarrow}(t_f - \delta)^2\dot{w}_b(t_f - \delta)^2$$

besteht aus zwei Summanden. Da auf $(t_f - \delta, t_f)$ die Ungleichung $w_b(t) - w_b(t_f - \delta) > 0$ wegen $\dot{w}_b(t) > 0$ gilt, ist der Zähler nicht null und konvergiert auch nicht dagegen, da $w_b(t_f) \neq w_b(t_f - \delta)$ gilt. Analog ergibt sich die Argumentation im Fall $w_b(t) < 0$. Insgesamt folgt, dass $\lambda_{b\rightarrow}(t)^2$ für $t \nearrow t_f$ gegen unendlich divergiert und deswegen die Kurve von $\lambda_{b\leftarrow}(t)^2$ treffen muss.

Es kann natürlich auch die Situation eintreten, in der alle Achsen an einer Stelle $t^* \in (t_0, t_f)$ Geschwindigkeitsnullstellen haben, also im Inneren eines B–Bereichs. Springen an dieser Stelle auch noch die Achsbeschleunigungen, können die Werte für die Grenzkurve von der linken und rechten Seite verschieden sein. Nach der Beschreibung der erste Phase wird der kleinere der beiden Werte verwendet. Das ist in diesem Fall aber nicht optimal. Hier darf die Bahntrajektorie $\lambda_b(t)^2$ einen Sprung machen. Der B–Bereich kann bei t' geteilt werden und die Umparametrisierung nacheinander für die Intervalle $[t_0, t']$ und $[t', t_f]$ bestimmt werden.

5.3. Ergebnisse und Beispiele

Der Abschluss des Kapitels veranschaulicht die Ergebnisse dieses Ansatzes anhand von Beispielen und zieht daraus Schlussfolgerungen für dessen Nutzen. Die Beispiele sind die ersten drei B–Bereiche einer redundanten Referenzbewegung, die mit Konfiguration MD2 (siehe Tabelle 2.1) und der iterativen Zerlegung des vorherigen Kapitels erstellt wird. Abbildung 5.3 zeigt die Kontur und die Bahn der Grobachsen mit den rot eingefärbten B–Bereichen.

Beginnen wir mit dem ersten B–Bereich. Die Grenzkurve $\lambda_b^{\downarrow}(t)^2$ und die Bahntrajektorie $\lambda_b(t)^2$ sind in Abbildung 5.4 (oben) zu sehen. Die Beschleunigungsverläufe der Achsen vor und nach der Umparametrisierung zeigt Abbildung 5.5. Dabei ist es wichtig, dass die roten

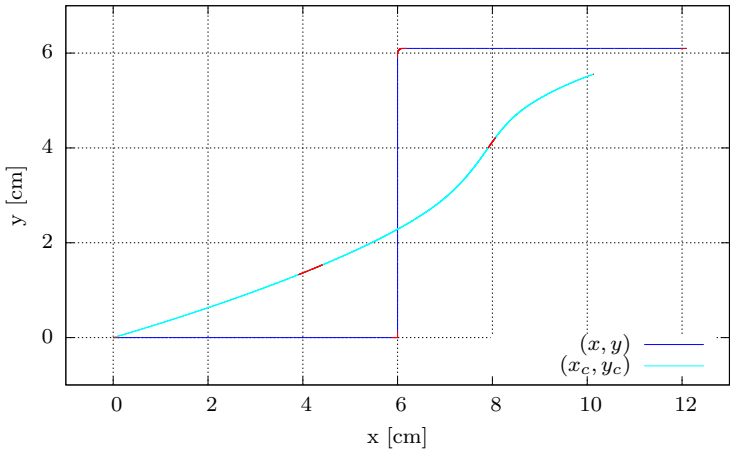

Abbildung 5.3.: Kontur und Bahn der Grobachsen mit rot eingefärbten B–Bereichen. Die Bewegung startet bei $[0, 0]$. Die rote Einfärbung am Anfang und ab Ende der cyanfarbenen Linie kann man kaum erkennen.

Linien die neuen Beschleunigungen sind, allerdings parametrisiert über der „alten“ Zeit, zum Beispiel

$$\frac{\mathrm{d}^2 (x_c \circ \varphi)}{\mathrm{d}\tau^2}(\tau_b(t))$$

für die x_c–Achse. So lassen sich die Unterschiede besser vergleichen. Natürlich ist das zur neuen Bewegung gehörende Zeitintervall $[\tau_0, \tau_f]$ kürzer als $[t_0, t_f]$. Für diesen ersten B–Bereich sieht man, dass die Umparametrisierung ein Ausnutzen der maximalen Beschleunigung der x_f–Achse bewirkt. Nur am Ende des Abschnitts ist keine Achse an ihrer Grenze. Das ist verständlich, denn die neue Bewegung erreicht schon eher auf der Bahn den Bahnvorschub $\hat{v}_b$. Die Zeit verkürzt sich hier um 0.29 Millisekunden von 6.66 auf 6.37.

Der zweite B–Bereich befindet sich an einer scharfen Ecke, an der die Kontur einen rechten Winkel hat. Hier führt die Umparametrisierung zu einer maximal negativen Beschleunigung mit x_f vor und einer maximal positiven Beschleunigung mit y_f nach der Ecke. Ganz am Anfang und ganz am Ende beschleunigt die grobe Achse für eine kurze Zeit maximal. Der kleine, aus einem Punkt bestehende Zacken, jeweils in der Mitte der Verläufe von x_c und y_c, ist ein Effekt der Implementierung des Verfahrens und tritt bei den (exakten) kontinuierlichen Verläufen nicht auf. Die Bewegungszeit sinkt hier von 13.32 Millisekunden auf 13.28, das ist eine Verkleinerung von 0.04 Millisekunden.

Im dritten B–Bereich hat die Kontur eine Ecke, die ein Viertelkreis mit einem Millimeter Radius abgerundet. Hier wechseln die mit maximaler Beschleunigung bewegenden Achsen häufig (siehe Abbildung 5.7). Außerdem gibt es zwei Teilstücke, auf denen keine Achse maximal beschleunigt. Sie liegen bei etwa 312 und 317 Millisekunden. Auf diesen Stücken bewegt sich die Bahntrajektorie auf der Grenzkurve, wie man im unteren Teilbild von

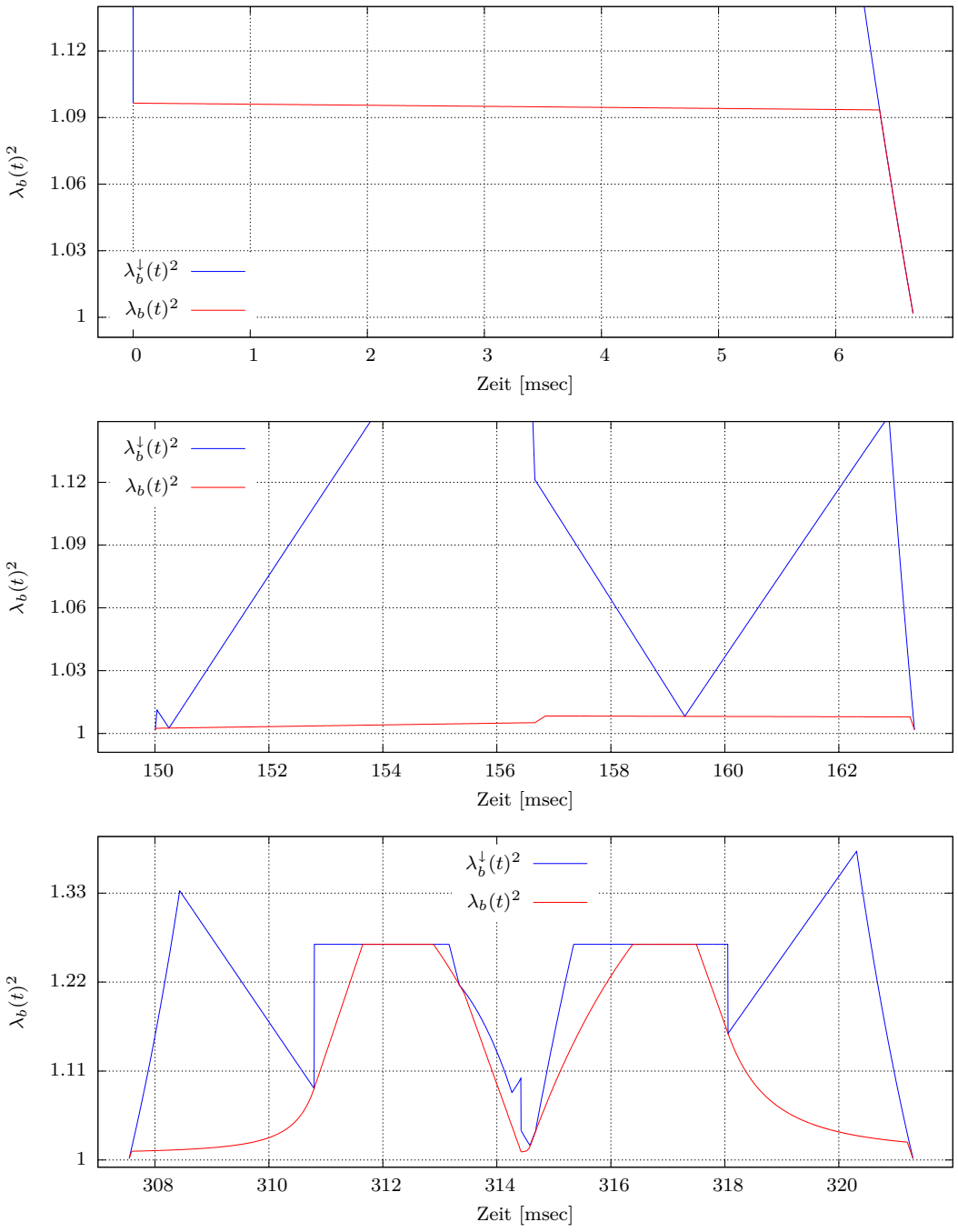

Abbildung 5.4.: Grenzkurve und Bahntrajektorie der ersten drei B–Bereiche.

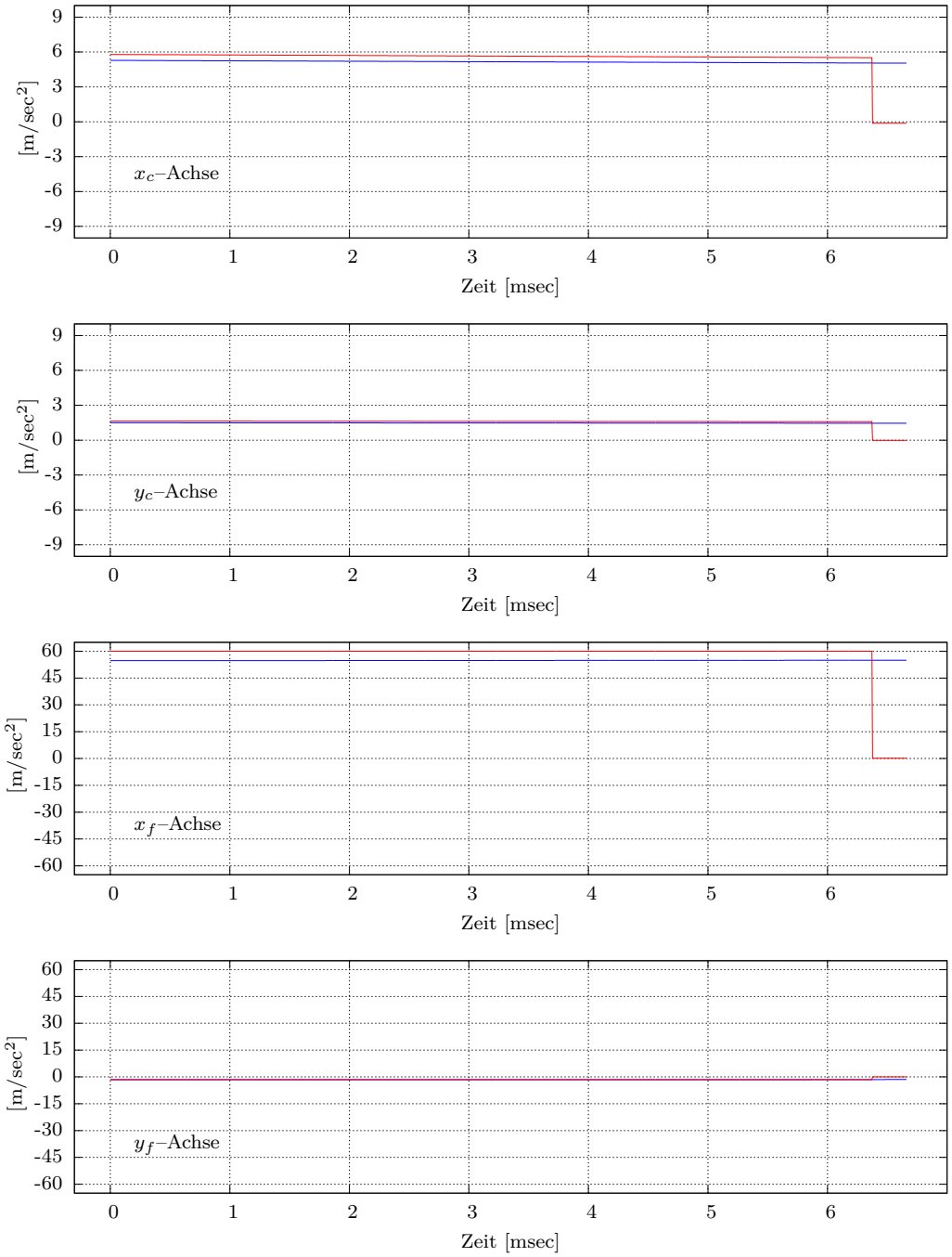

Abbildung 5.5.: Beschleunigungsverläufe des ersten B–Bereichs vor (blau) und nach (rot) Umparametrisierung.

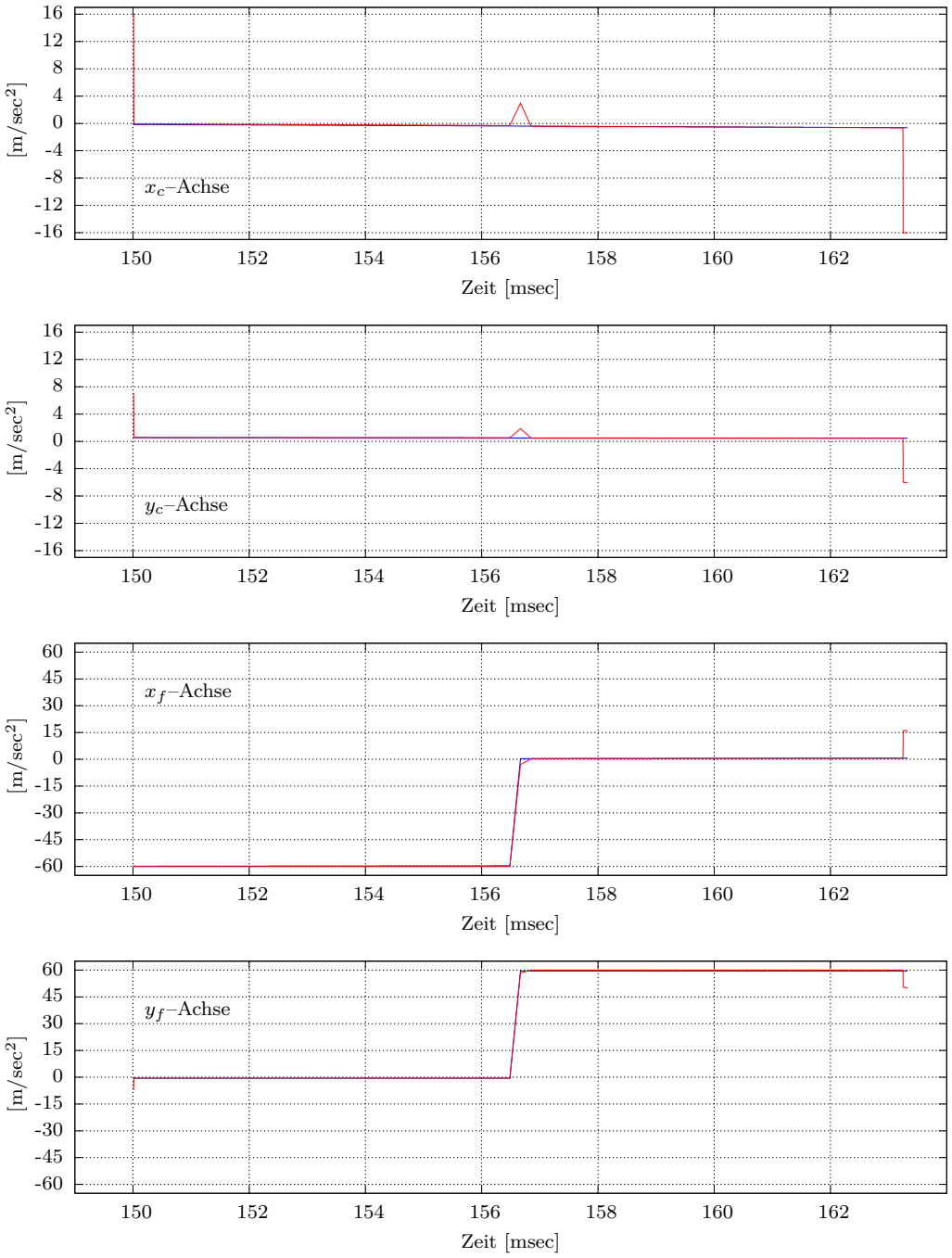

Abbildung 5.6.: Beschleunigungsverläufe des zweiten B–Bereichs vor (blau) und nach (rot) Umparametrisierung.

Abbildung 5.4 deutlich sieht. In beiden Fällen ist dafür Ungleichung (5.12) verantwortlich, zuerst für die x–, dann für die y–Komponente. Auch hier zeigen die Bilder unerwünschte Effekte der Implementierung. Die Ausschläge der Grobachsen bei etwa 313 Millisekunden sind beispielsweise solche. Die Zeitdauer für diesen B–Bereich vor der Umparametrisierung beträgt 13.75 Millisekunden, danach sind es nur noch 13.03. Das entspricht einer Reduzierung von 0.72 Millisekunden.

Betrachtet man die Gesamtbearbeitungszeit dieser Kontur von 472 Millisekunden, dann sieht die Zeitersparnis von etwa einer Millisekunde durch die Umparametrisierung unwesentlich aus. Das liegt vor allem daran, dass die Referenzbewegung auf einem großen Teil der Kontur nicht verkürzt werden kann, da sich der Werkzeugkopf dort schon mit Bahngeschwindigkeit $\hat{v}_b$ bewegt. Nur auf den B–Bereichen, die hier eine Gesamtzeit von 39 Millisekunden ausmachen, erfolgt die Reduzierung.

Um die Bearbeitungszeiten vor und nach der Umparametrisierung besser einordnen zu können, ist eine untere Grenze für die Zeitdauer interessant. Sie ergibt sich aus einer Bewegung mit der nichtredundanten Kinematik A, deren Grenzen für Beschleunigung und Geschwindigkeit die Summen der Werte für Grob- und Feinachsen sind,

$$\widehat{A}_x = \widehat{A}_{xc} + \widehat{A}_{xf} \qquad \text{und} \qquad \widehat{A}_y = \widehat{A}_{yc} + \widehat{A}_{yf}.$$

Das ist offensichtlich, denn das redundante System kann sich in eine Richtung nicht schneller bewegen. Für die Beispielkontur macht das eine Zeit von 467 Millisekunden. Schneller kann Kinematik B die Kontur nicht abarbeiten.

Bemerkung 5.6. *Diese Verhältnisse der Bearbeitungszeiten mit den verschiedenen Ansätzen zeigen, dass die redundante Bewegung aus der iterativen Zerlegung im Zeitbereich in den hier betrachteten Anwendungsfällen schon eine Produktivität nahe am Maximum hat. Im Allgemeinen hängt das natürlich von dem Anteil ab, den die B–Bereiche ausmachen. Darauf wiederum haben die Maschinendaten, der Vorschub $\hat{v}_b$ und die betrachteten Konturen wesentlichen Einfluss. In Situationen, in denen der Werkzeugkopf nur selten den programmierten Vorschub $\hat{v}_b$ erreicht, sind prozentual höhere Zeitverkürzungen zu erwarten.*

Ein weiterer Aspekt der Umparametrisierung ist der mögliche Verlust der Bewegungsglattheit der Grobachsen. Der Ansatz aus dem vorherigen Kapitel zeichnet sich dadurch aus, dass die Trajektorien der Grobachsen besonders glatte Beschleunigungsverläufe haben. Die Umparametrisierung führt aber wegen der Optimierung der Bearbeitungszeit zu Phasen, wo die Grobachsen maximal beschleunigen und sich nicht mehr so glatt bewegen. Die ersten beiden Bilder in Abbildung 5.7 zeigen diesen Effekt sehr schön. Im ungünstigen Fall kann eine Bewegung nach der Umparametrisierung zu schlechter Schnittqualität führen, so wie das bei dem Ansatz der geometrischen Zerlegung der Fall ist.

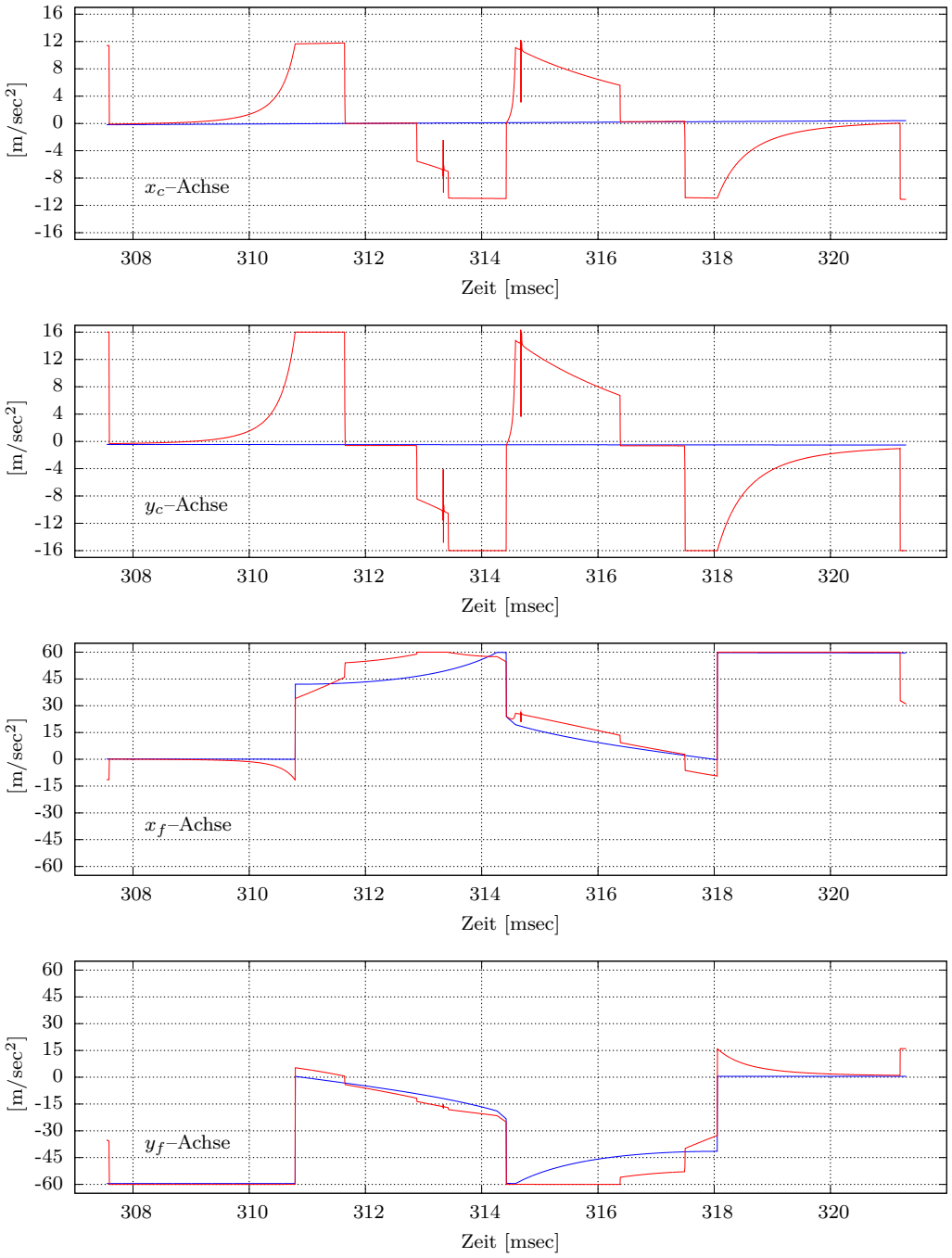

Abbildung 5.7.: Beschleunigungsverläufe des dritten B–Bereichs vor (blau) und nach (rot) Umparametrisierung.

6. Zusammenfassung und Ausblick

Die vorangegangenen Kapitel behandeln die Problematik der Steuerung von Werkzeugmaschinen mit redundanten Achsen. Es sei an dieser Stelle auch erwähnt, dass aus den im Rahmen dieser Arbeit entwickelten Verfahren zwei Patentanmeldungen hervorgegangen sind.

Kapitel zwei stellt zuerst die konkret untersuchte kinematische Struktur vor, ein redundantes Achssystem mit großen, linearen Hauptachsen und hochdynamischen, linearen Zusatzachsen. Maschinen mit derartigem kinematischen Aufbau werden beim 2-D-Laserschneiden von detailreichen Konturen aus Blechen eingesetzt. Mit diesem Anwendungsgebiet beschäftigt sich die Arbeit hauptsächlich. Die darauffolgende kurze Beschreibung der Arbeitsweise einer CNC–Steuerung und der angegebene Algorithmus für die beschleunigungsbegrenzte Bewegungsplanung dienen den weiteren Kapiteln als Grundlage.

In Kapitel drei folgt eine Übersicht über vier bekannte Ansätze zur Steuerung derartiger Maschinen. Jeder ist für bestimmte Anwendungsfälle ungeeignet oder erfüllt besondere Anforderungen nicht. Es bleiben Situationen, in denen keiner dieser Ansätze zufriedenstellend funktioniert. Das ist die Hauptmotivation für die in den beiden nachfolgenden Kapiteln entwickelten und untersuchten neuartigen Ansätze.

Die Abschnitte 3.4.1 und 3.4.2 beschreiben eine Verfahren für die hochdynamische Regelung redundanter Achsen. Es entstand ebenfalls im Rahmen dieser Arbeit, obwohl es sich von den anderen entwickelten Ansätzen unterscheidet, da es in den Regelkreisen der Antriebe Verwendung findet. Trotzdem ist es erwähnenswert, da es eine bisherige Schwachstelle der Regelung von redundanten Achsen behebt.

Der Ansatz in Kapitel vier erweitert das gewöhnliche Steuerungsschema und ermöglicht ein iteratives Verfahren. Darin wechseln sich die Bewegungsplanung nichtredundanter Referenztrajektorien und deren Zerlegung in die Verläufe der Einzelachsen solange ab, bis eine gültige Bewegung entsteht. Die Darstellung beschränkt sich auf den Fall beschleunigungsbeschränkter Bewegungen. Die Ergebnisse aus typischen Testbeispielen zeigen, dass dieser Ansatz in den untersuchten Anwendungsfällen zuverlässig funktioniert. Nachteilige Effekte der anderen Ansätze treten nicht auf.

Der in Kapitel fünf beschriebene Ansatz ist eine optionale Erweiterung des vorherigen. Er führt eine nachträgliche Umparametrisierung der Bahnbewegung durch, um die Bearbeitungszeit zu verkleinern. Die Ergebnisse zeigen, dass sich diese Zusatzphase nur bei ganz besonderen Konturen lohnt. In den anderen Fällen gewinnt man kaum noch Zeit. Vergleicht man die Ergebnisse des Ansatzes aus Kapitel vier mit einer theoretischen Un-

tergrenze für die Bearbeitungszeit, stellt sich heraus, dass er schon nahe am zeitlichen Optimum liegende Bewegungen erzeugt.

Natürlich gibt es noch einige offene Fragestellungen in diesem Themengebiet, die weitere Forschungsaktivitäten motivieren. Die Verfahren der Kapitel vier und fünf beschränken sich auf beschleunigungsbegrenzte Bewegungen. In der Praxis gibt es aber auch Anwendungsfälle, in denen sich die Achsen einer Maschine ruckbegrenzt bewegen sollen. Aus diesem Grund ist es eine interessante Frage, inwiefern sich die entwickelten Ansätze auf diesen Fall übertragen lassen.

Eine weitere mögliche Forschungsaktivität liegt in der Aufgabe der Steuerung komplexerer Kinematiken. Wie in Abschnitt 2.1.3 erwähnt, gibt es schon Maschinen mit redundanten Achsen, deren kinematischer Aufbau komplexer ist als der in dieser Arbeit betrachtete. Es ist noch offen, ob und wie man die hier entwickelten Verfahren für derartige Maschinen nutzen kann.

Symbolverzeichnis

Allgemeines

$\mathbb{N}$, $\mathbb{N}_0$	Menge der natürlichen Zahlen: 1,2,3,..., $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
$\mathbb{R}$, $\mathbb{R}^+$	Menge der reellen Zahlen, Menge der nichtnegativen reellen Zahlen
$C(I)$	Menge der auf einem Intervall I stetigen Funktionen
$C^k(I)$	Menge der Funktionen, die auf dem Intervall I definiert und k–mal stetig differenzierbar sind
$\mathbb{P}_m(I)$	Menge der Polynome vom Grad m auf einem Intervall I
$\mathbf{1}_n$	$(1, \ldots, 1)^T \in \mathbb{R}^n$
E_n	Einheitsmatrix der Größe $n \times n$
$\|\|\mathbf{p}\|\|_2$	L_2-Norm des Vektors $\mathbf{p}$
$\|\|\mathbf{p}\|\|_\infty$	Supremumsnorm des Vektors $\mathbf{p}$
$\circ$	Kompositionszeichen für Funktionen
$\lfloor\ \rfloor$	Untere Gauß-Klammern
$\#M$	Anzahl der Elemente der endlichen Menge M
$t \nearrow t^*$, $t \searrow t^*$	$t \to t^*$, $t < t^*$ beziehungsweise $t \to t^*$, $t > t^*$
$f(t^+), f(t^-)$	Rechtsseitiger Grenzwert: $f(t^+) = \lim_{\tau \nearrow t} f(\tau)$, linksseitiger Grenzwert: $f(t^-) = \lim_{\tau \searrow t} f(\tau)$

Kinematiken allgemein

$\mathbf{q}$	Vektor der $n_\mathbf{q}$ Maschinenachsen
$\mathbf{p} = (x, y, z)^T$	Punkt im dreidimensionalen Umgebungsraum
$\boldsymbol{\alpha} = (\alpha, \beta, \gamma)^T$	Vektor der Orientierungswinkel
Λ	Kinematische (Vorwärts)Transformation
$\mathcal{D}_{\text{MKS}}$	Menge der erlaubten Achsstellungen einer Maschine
$\mathcal{D}_{\text{BKS}}$	Menge aller möglichen Stellungen des Werkzeugkopfes
$h_\mathbf{p}(s) = [x(s), y(s), z(s)]^T$	Verlauf der Kontur im Raum
$h_{\boldsymbol{\alpha}}(s) = [\alpha(s), \beta(s), \gamma(s)]^T$	Verlauf der Orientierung der Kontur, parametrisiert nach der Bogenlänge von $h_\mathbf{p}$
$h(s) = [h_\mathbf{p}(s)^T, h_\alpha(s)^T]^T$	Abzufahrende Werkstückbahn mit Orientierungskurve
s	Bogenlängenparameter der Kurve $h_\mathbf{p}$
$[s_0, s_f]$	Bogenlängenintervall der Konturkurve mit Länge $s_f - s_0$
t	Zeitparameter
$[t_0, t_f]$	Zeitintervall einer Bewegung auf der Kontur
$\hat{v}_b$	Maximal erlaubte Bahngeschwindigkeit entlang $h_\mathbf{p}$ (Seiten 10, 16)

t_{ipo}	Taktzeit des Lagereglers und gleichzeitig Abtastzeit der Achstrajektorien am Ende der Bewegungsführung (Seite 22)
$L_{\min}$	Minimale Bogenlänge jedes Stückes einer Kontur (Seite 13)
$\kappa_{\max}$	Maximale Krümmung einer Konturkurve (Seite 13)

Kinematik A

Λ_A	Kinematische Transformation (hier Identität)
$h(s) = [x(s), y(s)]^T$	Bogenlängenparametrisierung der Kontur
$s_b(t)$	Bahntrajektorie über Zeit
$v_b(t)$	Bahngeschwindigkeit über Zeit
$a_b(t)$	Bahnbeschleunigung über Zeit
$j_b(t)$	Bahnruck über Zeit
$\mathbf{q} = [q_x, q_y]^T$	Maschinenachsen
$[v_x(t), v_y(t)]^T$	Vektor der Achsgeschwindigkeiten
$\widehat{\mathbf{V}} = [\widehat{V}_x, \widehat{V}_y]^T$	Vektor der maximalen Achsgeschwindigkeiten
$[a_x(t), a_y(t)]^T$	Vektor der Achsbeschleunigungen
$\widehat{\mathbf{A}} = [\widehat{A}_x, \widehat{A}_y]^T$	Vektor der maximalen Achsbeschleunigungen
$[j_x(t), j_y(t)]^T$	Vektor der Achsrucke
$\widehat{\mathbf{J}} = [\widehat{J}_x, \widehat{J}_y]^T$	Vektor der maximalen Achsrucke
$\mathbf{S}_h$	Satzwechsel der Kontur
K_{b}	Konstante aus $(0, 1)$ für den Algorithmus der Bewegungsplanung (Seite 24)

Kinematik B (ergänzend)

Λ_B	Kinematische Transformation
$\mathbf{q} = [x_c, y_c, x_f, y_f]^T$	Maschinenachsen
$[\widehat{S}_{xf}, \widehat{S}_{yf}]^T$	Verfahrbereichsgrenzen der feinen Achsen
$[v_{xc}(t), v_{yc}(t), v_{xf}(t), v_{yf}(t)]^T$	Vektor der Achsgeschwindigkeiten
$\widehat{\mathbf{V}} = [\widehat{V}_{xc}, \widehat{V}_{yc}, \widehat{V}_{xf}, \widehat{V}_{yf}]^T$	Vektor der maximalen Achsgeschwindigkeiten
$[a_{xc}(t), a_{yc}(t), a_{xf}(t), a_{yf}(t)]^T$	Vektor der Achsgeschwindigkeiten
$\widehat{\mathbf{A}} = [\widehat{A}_{xc}, \widehat{A}_{yc}, \widehat{A}_{xf}, \widehat{A}_{yf}]^T$	Vektor der maximalen Achsgeschwindigkeiten
$[j_{xc}(t), j_{yc}(t), j_{xf}(t), j_{yf}(t)]^T$	Vektor der Achsgeschwindigkeiten
$\widehat{\mathbf{J}} = [\widehat{J}_{xc}, \widehat{J}_{yc}, \widehat{J}_{xf}, \widehat{J}_{yf}]^T$	Vektor der maximalen Achsgeschwindigkeiten
$\Lambda_{\widetilde{B}}$	Kinematische Transformation im Ansatz der geometrischen Zerlegung

Splines

B, n_b	Folge und Anzahl der Bruchstellen
m	Splinegrad
$\mathbb{S}_m(B)$	Splineraum zur Bruchstellenfolge B
n	Dimension des Splineraums
$T = T_{n,m} = (\zeta_1, \ldots, \zeta_{n+m+1})$	Knotenfolge
d	Folge der Kontrollpunkte
$\mathbb{S}_{m,T}$	Vektorraum der Splines vom Grad m zur Knotenfolge T
$N_j^m(\cdot\|T)$	j-ter B-Spline
$N_{m,T}d(\cdot)$	Splinefunktion
$\mathbf{t}$, $\mathbf{x}$	Zu approximierende Daten t_i, x_i
n_t	Anzahl der zu approximierenden Datenpunkte
P_1, V_1, L_1	Linke Randbedingungen für Position, erste und zweite Ableitung (Seite 51)
P_{n_t}, V_{n_t}, L_{n_t}	Rechte Randbedingungen für Position, erste und zweite Ableitung (Seite 51)
M_1, K_1, L_1	Konstanten aus linken Randbedingungen für Approximation (Seite 51)
M_{n_t}, K_{n_t}, L_{n_t}	Konstanten aus rechten Randbedingungen für Approximation (Seite 51)
$\mathbf{N}_{m,T}(\mathbf{t})$	Spline-Kollokationsmatrix (Seite 50)
λ	Glätteparameter bei Smoothing-Splines (Seite 61)
k_{vf}	Vielfachheit der inneren Knoten bei Approximationsverfahren (Seite 54)

Literaturverzeichnis

[1] V. Albrecht. „Doppeltes Flottchen — Stanz–Laser–Kombimaschinen: Im Trend zur Komplettbearbeitung“. In: *Industrieanzeiger* 37 (2009), S. 52.

[2] H. Amann und J. Escher. *Analysis I*. Basel (Schweiz): Birkhäuser Verlag, 2006.

[3] B. A. Barsky und A. D. DeRose. *Geometric Continuity of Parametric Curves*. Techn. Ber. CSD-84-205. Berkeley, CA, USA, 1984.

[4] M. Bock. „Bahnplanung von redundanten CNC–Achsen — Ansätze für die geometrische Bahnaufteilung“. Diplomarbeit. Justus–Liebig–Universität Gießen, 2006.

[5] M. Bock, W. Papiernik und T. Sauer. „Methods for Path Decomposition of Redundant CNC–Axes“. In: *Proceedings of PCIM Europe Conference 2008, May 27-29*. 2008, S. 620–625.

[6] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 2001.

[7] C. de Boor. „Calculation of the smoothing spline with weighted roughness measure“. In: *Mathematical Models and Methods in Applied Sciences* 11.1 (2001), S. 33–41.

[8] S. Breun. „Optimale Steuerung redundanter Roboter auf Mannigfaltigkeiten — Strukturanalyse und numerische Realisierung“. Dissertation. Technische Universität München, 2007.

[9] M. Cardinale u. a. „Method for Controlling Systems Provided With Redundant Actuators“. Pat. EP1758003A1.

[10] D. R. Cutler und R. M. Pailthorp. „Apparatus and Method for Coordinating the Movements of Stages in a Multi-Stage Multi-Rate Positioner System“. Pat. US5798927. 1998.

[11] A. Ehlerding. „CNC-gesteuerte mehrachsige Werkzeugträger“. Pat. EP594699B1. 1993.

[12] R. Ekachaiworasin. „Design and Evaluation of an Algorithm for Jerk–Limited Motion Control“. Masterarbeit. Technische Universität Hamburg-Harburg, 2005.

[13] R. T. Farouki und T. Sakkalis. „Rational Space Curves are not “Unit Speed”“. In: *Computer Aided Geometric Design* 24.4 (2007), S. 238–240.

[14] R. T. Farouki und T. Sakkalis. „Real Rational Curves are not “Unit Speed”“. In: *Computer Aided Geometric Design* 8.2 (1991), S. 151–157.

[15] T. Franitza, A. Dietmair und A. Verl. „Werkzeugmaschinen mit dynamischem Bearbeitungskopf“. In: *wt Werkstattstechnik online* 96 (2006), S. 246–251.

[16] M. Fujita und T. Bamba. „Positioning Device for a Machining Apparatus“. Pat. US5109148. 1992.

[17] K. Geißdörfer, C. Hamm und W. Papiernik. „Einrichtung und Verfahren zur Bewegungsaufteilung einer Bewegung eines Maschinenteils entlang einer Antriebsachse einer Werkzeug- oder Produktionsmaschine“. Pat. DE10355614B4. 2006.

[18] W. Hoffmann, W. Papiernik und T. Sauer. „Ermittlungsverfahren für eine lagegeführte abzufahrende Grobbahn“. Pat. DE102005061570A1. (Patent noch nicht erteilt).

[19] W. Hoffmann, W. Papiernik und T. Sauer. „Verfahren und Einrichtung zur Bewegungsführung eines bewegbaren Maschinenelements einer numerisch gesteuerten Maschine“. Pat. DE102004059966B3 (München). 2006.

[20] G. Hämmerlin und K.-H. Hoffmann. *Numerische Mathematik*. Springer, 1991.

[21] R. Johanni. *Optimale Bahnplanung bei Industrierobotern*. VDI-Verlag, 1988.

[22] H. B. Kief und H. A. Roschiwal, Hrsg. *NC/CNC Handbuch 2007/2008*. München: Carl Hanser Verlag, 2007.

[23] D. E. Kirk. *Optimal Control Theory*. New York: Dover Publications, 2004.

[24] P. Leibinger, T. Rauser und L. Zeygerman. „Laser Cutting Machine with Two y-Axis“. Pat. EP1366846B1. 2007.

[25] J. Nocedal und S. J. Wright. *Numerical Optimization*. New York: Springer, 2006.

[26] J. L. Olazagoitia und S. Wyatt. „New PKM Tricept T9000 and Its Application to Flexible Manufacturing at Aerospace Industry“. In: *Automated Fastening and Assembly & Tooling in Aerospace*. SAE International, 2007.

[27] T. Ostermann, W. Herfs und C. Brecher. „Steigerung der Dynamik durch redundante Zusatzachsen“. In: *Proceedings of SPS/IPC/DRIVES*. Nürnberg, 2007.

[28] W. Papiernik. „Architecture and Design of Modern CNC/Drive Systems“. In: *Proceedings on PCIM-Conference*. Bd. 29. 1996, S. 271–280.

[29] G. Pardo-Castellote und R. H. Cannon Jr. „Proximate Time–Optimal Algorithm for On–Line Path Parameterization and Modification“. In: *IEEE International Conference on Robotics and Automation*. Bd. 2. 1996, S. 1539–1546.

[30] Prima Industrie S.p.A. *Optimo Vivida — The "Full–of–Life" Optimo Head*. Collegno (Torino), Italien, 2008. URL: `http://www.primaindustrie.com`.

[31] Prima Industrie S.p.A. *Syncrono — The New Laser Era*. Collegno (Torino), Italien, 2006. URL: `http://www.primaindustrie.com`.

[32] B. Siciliano und O. Khatib, Hrsg. *Handbook of Robotics*. Springer, 2008.

[33] S. Staroselsky und K. A. Stelson. „Two–Stage Actuation for Improved Accuracy of Contouring“. In: *Proceedings of the American Control Conference*. 1988, S. 127–132.

[34] R. F. Stengel. *Optimal Control and Estimation, Dover Edition*. New York: Dover Publications, 1994.

[35] O. von Stryk. „Numerische Lösung optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung der adjungierten Variablen“. Diss. Technische Universität München, 1994.

[36] TRUMPF Werkzeugmaschinen Deutschland Vertrieb + Service GmbH + Co. KG. *TruMatic 7000: Kombination in Perfektion*. Ditzingen, 2008. URL: `http://www.trumpf.com`.

[37] TRUMPF Werkzeugmaschinen GmbH + Co. KG. *TruLaser: Flexible Cutting Through Thick and Thin*. Ditzingen, 2008. URL: `http://www.trumpf.com`.

[38] H. Unbehauen. *Regelungstechnik I*. Vieweg Verlag, 1997.

[39] M. Weck und C. Brecher. *Werkzeugmaschinen — Automatisierung von Maschinen und Anlagen*. Bd. 4. Springer-Verlag, 2006.

A. Anhang

A.1. Optimalsteuerungsprobleme

Ein allgemeines Optimalsteuerungsproblem besteht aus einem Funktional als Zielfunktion, das unter Differentialgleichungs- und Differentialungleichungsnebenbedingungen minimiert werden soll und dazu noch vorgegebene Randbedingungen erfüllen muss. Als allgemeine Literaturquellen zu diesem Thema bieten sich neben vielen anderen gleichermaßen die Bücher von Kirk [23] und Stengel [34] an. Die hier genutzt Notation ist nicht auf die des Rests der Arbeit abgestimmt, sondern eigenständig.

Die variablen Größen in einem Optimalsteuerungsproblem bestehen aus dem Vektor[1] $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))$ der *Zustandsgrößen* und dem Vektor $\mathbf{u}(t) = (u_1(t), \ldots, u_l(t))$ der *Steuergrößen*. Beides sind Kurven über dem Intervall $[t_0, t_f]$, wobei t_f fest oder variabel in die Problemstellung eingehen kann. Die x_k und u_j sind durch die *dynamischen Gleichungen*

$$\dot{x}_k(t) = f_k(\mathbf{x}(t), \mathbf{u}(t), t), \qquad k = 1, \ldots, n, \quad t \in [t_0, t_f], \tag{A.1}$$

sowie die *Anfangs- und Endbedingungen*

$$r_j(\mathbf{x}(t_0), \mathbf{x}(t_f)) = 0, \qquad j = 1, \ldots, n + n_f \leq 2n, \tag{A.2}$$

miteinander verknüpft. Zusätzlich wird die Erfüllung der *Gleichungs- und Ungleichungsbedingungen*,

$$h_j(\mathbf{x}(t), \mathbf{u}(t), t) = 0, \quad t \in [t_0, t_f], \quad j = 1, \ldots, m_h, \quad \text{und} \tag{A.3}$$

$$g_j(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0, \quad t \in [t_0, t_f], \quad j = 1, \ldots, m_g, \tag{A.4}$$

von den Funktionen x_i und u_j verlangt. Auf die Eigenschaften der Funktionen f_k, r_j, h_j und g_j gehen wir hier nicht weiter ein. Unter allen diesen Restriktionen werden nun diejenigen Größen $\mathbf{x}$ und $\mathbf{u}$ gesucht, die ein Funktional

$$J[\mathbf{u}, t_f] = \Phi(\mathbf{x}(t_f), t_f) \tag{A.5}$$

minimieren. (Wir gehen davon aus, dass aus einem vorgegebenen $\mathbf{u}$ durch die dynamischen Gleichungen und die Anfangs- und Endbedingungen der Zustandsvektor $\mathbf{x}$ eindeutig folgt und dass es überhaupt gültige, das heißt, die Bedingungen (A.1) bis (A.4) erfüllende, Größen gibt)

[1] Die Einträge des Vektors sind Funktionen.

Erkärung

Ich erkläre: Ich habe die vorgelegte Dissertation selbständig und ohne unerlaubte fremde Hilfe und nur mit den Hilfen angefertigt, die ich in der Dissertation angegeben habe.
Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen sind, und alle Angaben, die auf mündlichen Auskünften beruhen, sind als solche kenntlich gemacht.
Bei den von mir durchgeführten und in der Dissertation erwähnten Untersuchungen habe ich die Grundsätze guter wissenschaftlicher Praxis, wie sie in der „Satzung der Justus-Liebig-Universität Gießen zur Sicherung guter wissenschaftlicher Praxis“ niedergelegt sind, eingehalten.

Gießen, August 2010