Verallgemeinerte inverse Kinematik für Anwendungen in der Robotersimulation und der virtuellen Realität

Diplomarbeit
von
Wolfgang Smidt

www.worldwidewolf.de
smidt@worldwidewolf.de

1997 / 98

Betreuer : Prof. Dr. - Ing. E. Freund
Dr. - Ing. J. Rossmann

Institut für Roboterforschung
Universität Dortmund

Inhaltsverzeichnis

1 Einleitung

Im Rahmen der Entwicklung eines Programmes zur Echtzeitsimulations von Roboterarbeitszellen wird hier ein numerisches Verfahren erarbeitet, das es erlaubt, die durch simulierte äußere Krafteinwirkung hervorgerufenen Bewegungen beliebiger passiver Kinematiken zu berechnen. Ziel ist es, daß der gegriffene Mechanismus sich in der Simulation möglichst ebenso verhält wie in der Realität, auch wenn zwischen Greifer und Greifpunkt eine durch kinematische Bedingungen erzwungene Lageabweichung bestehen bleiben sollte.

Derartige greifbare Kinematiken sind durch Denavit-Hartenberg-Parameter modelliert. Sie bestehen im einfachsten Fall aus Klapp-, Dreh oder Schiebemechanismen wie sie zum Beispiel an Schränken oder Einlegevorrichtungen vorkommen, in komplizierteren Fällen aus vielachsigen kinematischen Ketten wie sie beispielsweise Industrieroboter darstellen (Teach-in Funktion). Achsspezifische Maximalbeschleunigungen und -geschwindigkeiten aktiver Teile werden dabei (optional) nie überschritten, jedoch optimal genutzt.

Anschließend wird das Verfahren erweitert auf die Simulation allgemeiner Plattformen (z.B. Steward-Plattformen) und geschlossener kinematischer Ketten.

Die Berechnung erfolgt unabhängig von der Art des Greifers, so daß sowohl ein Roboter, als auch eine mit einem Datenhandschuh versehene menschliche Hand in der virtuellen Realität hierfür in Frage kommen.

Besonders im zweiten Fall wird damit eine Beitrag zur intuitiven Mensch-Maschine-Kopplung geleistet, da die agierende Person keine spezifischen Kenntnisse der Simulationsumgebung benötigt, sondern sich fast ausschließlich in vertrauten realen Handlungsmustern bewegen kann.

Dies ist auch allgemein ein Ziel des bestehenden Simulationsprogrammes ‘COSIMIR’ in das das Verfahren schließlich modular in Form einer in C programmierte DLL implementiert wird und so neben anderen Modulen auswählbar zur Verfügung steht.

Diese Arbeit ist im Internet unter http://www.worldwidewolf.de/diplom.pdf abrufbar.

2 Grundlagen

Zunächst werden hier kurz die wichtigsten zum Verständniss notwendigen Grundlagen [1] vorgestellt. Dieses geschieht jedoch teilweise nicht vollständig und allgemeingültig, sondern lediglich im Hinblick auf die speziellen Erfordernisse dieser Arbeit.

2.1 Homogene Transformation

Im hier erweiterten Simulationsprogramm werden zusammenhängende Kinematiken als „Objekte“ und deren in sich starre Elemente als „Sektionen“ bezeichnet. Derartige Sektionen haben jeweils ein körperfestes Koordinatensystem, welches beispielsweise Ausgangspunkt für die Modellierung von Hüllkörpern zur graphischen Darstellung der jeweiligen Sektion ist. Diese Koordinatensysteme sind definiert durch homogene Transformationsmatrizen (Frames), welche ein einziges festes Welt-Koordinatensystem in die Systeme der jeweiligen Sektionen überführen.

Zur Berechnung dieser Matrizen ist es notwendig zunächst Rotations- und Translationsbewegungen der einzelnen Sektionen zueinander darzustellen, also die relative Lage jeweils zweier benachbarter Sektionen zueinander in Abhängigkeit der jeweiligen variablen Achskoordinate zu definieren.
Das körperfeste Koordinatensystem einer Sektion n-1 wird dabei durch Multiplikation mit einer von ihrer Achskoordinate abhängigen homogenen Transformationsmatrix in das körperfeste Koordinatensystem der ihr nachfolgenden Sektion n transformiert.

Bei einem reinen Translationsgelenk ist diese Matrix definiert durch

$$\underline{\underline{T}}_{n-1,\,n} = \left[\begin{array}{ccc:c} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right],$$

und beschreibt nur eine Verschiebung des Ursprungs.
Bei einem reinen Rotationsgelenk hat die homogene Transformationsmatrix folgende Form.

$$\underline{\underline{T}}_{n-1,\,n} = \left[\begin{array}{ccc:c} u_x & v_x & w_x & 0 \\ u_y & v_y & w_y & 0 \\ u_z & v_z & w_z & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right].$$

und beschreibt mit der 3x3 Orientierungsmatrix lediglich eine Raumdrehung.
Im folgenden gelten zwecks einfacher Notation die Abkürzungen

$$\underline{\mathrm{Pos}}\,(\underline{\underline{\mathbf{T}}}) = \left[p_x\,,\,p_y\,,\,p_z\right]^T$$

$$\underline{\mathbf{Rot}}\left(\underline{\mathbf{T}}\right) = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} = \left[\underline{\mathrm{Rot}}_0\left(\underline{\mathbf{T}}\right), \quad \underline{\mathrm{Rot}}_1\left(\underline{\mathbf{T}}\right), \quad \underline{\mathrm{Rot}}_2\left(\underline{\mathbf{T}}\right)\right]$$

1 Sämtliche sektionsfesten Koordinatensysteme entstehen aus dem Basis-Koordinatensystem des Objektes multipliziert mit sämtlichen homogenen Transformationsmatrizen der kinema-tischen Kette bis zu der betrachteten Sektion. Die ersten 3 Zeilen dieses Matrizenproduktes definieren also die aus Position und Orientierung bestehende Lage einer Sektion im Raum.Das Basis-Koordinatensystem eines Objekts ergibt sich analog durch Multiplikation des Welt-Koordinatensystem mit der homogenen Transformationsmatrix der Objektbasis.

$$\underline{\mathbf{T}}_{\text{Welt, 3}} = \underline{\mathbf{T}}_{\text{Welt, Basis}} \cdot \underline{\mathbf{T}}_{\text{Basis, 1}} \cdot \underline{\mathbf{T}}_{\text{1, 2}} \cdot \underline{\mathbf{T}}_{\text{2, 3}}$$

Siehe hierzu auch Abbildung 2.

2.2 Denavit-Hartenberg Parameter

Da die Beziehung zwischen 2 Sektions-Koordinatensystemen nur in einigen Fällen eine reine Translation oder eine reine Rotation darstellt, ist eine allgemeinere Transformationsvorschrift notwendig, die auch beliebig zueinander verlaufende Gelenkachsen berücksichtigt. Eine solche allgemeine Beschreibung liefert die Parametrisierung nach Denavit-Hartenberg.

Danach wird die relativ komplizierte Transformation des Koordinatensystems n-1 in das Koordinatensystem n durch folgende vier nacheinander auszuführende Elementartransformationen beschrieben :

$$\underline{\mathbf{T}}_{\text{n-1, n}} = \underline{\mathbf{Rot}}\left(\underline{z}_{n-1}, \Theta_n\right) \cdot \underline{\mathbf{Trans}}\left(0,0,d_n\right)^T \cdot \underline{\mathbf{Trans}}\left(a_n,0,0\right)^T \cdot \underline{\mathbf{Rot}}\left(\underline{x}_n, \alpha_n\right).$$

Hierbei ist $\underline{\mathbf{Rot}}\left(\underline{v}, \alpha\right)$ eine einfache Rotation um den Vektor $\underline{v}$ mit dem Winkel α und $\underline{\mathbf{Trans}}\left(x,y,z\right)^T$ eine translatorische Bewegung entlang des Vektors $\left[x,y,z\right]^T$.

Die Denavit-Hartenberg Parameter bezeichnen im einzelnen :

Θ_n = Winkel zwischen Achse x_{n-1} und Achse x_n.
a_n = Länge der einzigen gemeinsamen Normalen der Gelenkachsen.
d_n = Länge vom Ursprung des Koordinatensystems n-1 bis zum Schnittpunkt der gemeinsamen Normalen mit der Achse z_{n-1}.
α_n = Winkel zwischen Achse z_{n-1} und der Achse z_n nach Drehung um die x_n Achse.

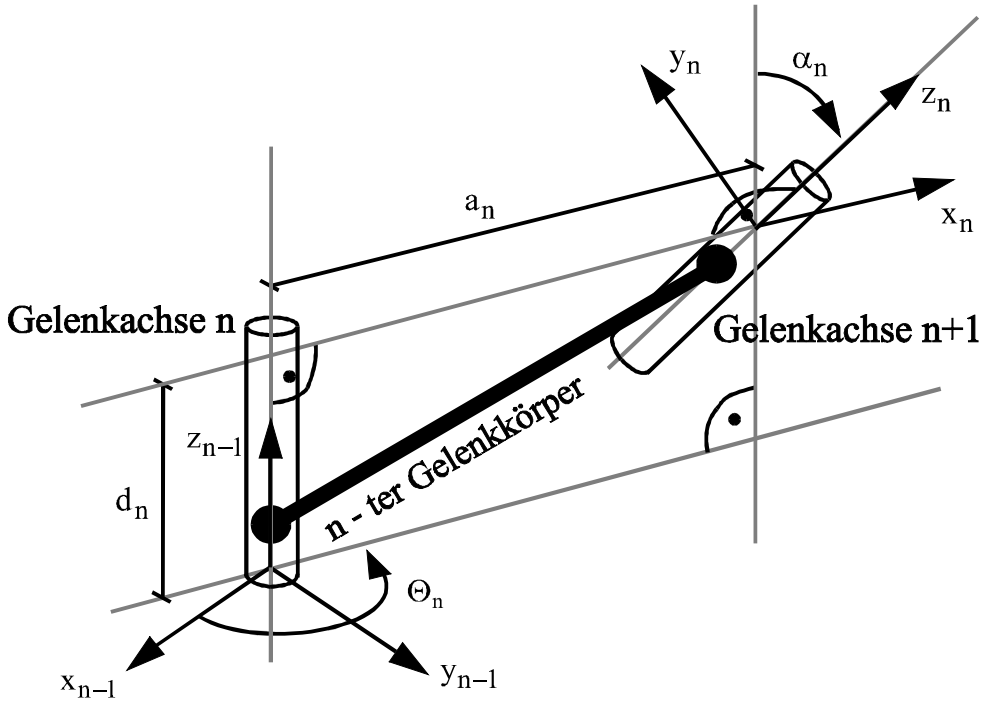

Abbildung 1 : Denavit-Hartenberg Parametrisierung eines Gelenks

2.3 Jakobi Matrix

Die Jakobi Matrix gibt die lineare Beziehung zwischen den karthesischen Raumkoordinaten-geschwindigkeiten $\dot{\underline{\ell}}$ (des Greifpunktes) und den Geschwindigkeiten der Achskoordinaten $\dot{\underline{q}}$ wieder und ist aufgrund ihrer numerischen Form nur für jeweils die aktuelle Gelenkstellung gültig.

Der karthesische Lagekoordinaten-Vektor wird hier mit $\underline{\ell}$ bezeichnet. In der Literatur findet sich häufig auch $\underline{x}$, was jedoch schnell zu Verwechslungen mit der Raumrichtung führen kann.

$$\dot{\underline{\ell}} = \underline{\mathbf{J}} \cdot \dot{\underline{q}}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{dx}{dq1} & \cdots & \frac{dx}{dqn} \\ \vdots & \ddots & \vdots \\ \frac{d\varphi z}{dq1} & \cdots & \frac{d\varphi z}{dqn} \end{bmatrix} \cdot \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

$\dot{\underline{\ell}}$ besteht in den ersten drei Elementen aus der Translationsgeschwindigkeit des Ursprungs. Die unteren drei Komponenten geben die Rotationsgeschwindigkeiten um die jeweiligen Hauptachsen des Koordinatensystems an.

Die Berechnung der n-ten Spalte der Jakobi-Matrix erfolgt, falls das Gelenk n ein Schubgelenk ist, nach der Formel

$$\underline{\underline{J}}_n = \begin{bmatrix} \underline{z}_{n-1} \\ \underline{0} \end{bmatrix} .$$

Falls das Gelenk n ein Drehgelenk ist gilt

$$\underline{\underline{J}}_n = \begin{bmatrix} \underline{z}_{n-1} \times \underline{r}_{n-1} \\ \underline{z}_{n-1} \end{bmatrix} .$$

Hierbei ist der Vektor $\underline{z}_{n-1}$ die in Normalform angegebene Bewegungsrichtung der n-ten Gelenkachse und $\underline{r}_{n-1}$ den Vektor vom n-1 ten Sektionskoordinatensystem bis zum Greifpunkt definiert, auf den sich die Jakobi-Matrix bezieht.

Die so erhaltene Matrix ist jedoch auf das Weltkoordinatensystem bezogen, da die Sektions-koordinatensysteme hierauf bezogen sind.

3 Allgemeines Lösungsverfahren

3.1 Überblick

Es muß prinzipiell zwischen zwei (in den Kapiteln 4 und 5 beschriebenen) Verfahren unterschieden werden, die gemeinsam eine verallgemeinerte numerische Lösung des inversen kinematischen Problems bilden. Beiden gemeinsam ist, daß sie schrittweise den Abstand zwischen Greifer und Greifpunkt durch Variation der zu ihm führenden Gelenkparameter minimieren.

Ist es grundsätzlich möglich den Greifpunkt exakt in die Lage des Greifers zu verfahren, so wird dieses durch unter Umständen mehrfache Wiederholung einer im nächsten Kapitel erläuterten Universaltransformation (oder auch Rückwärtstransformation) am schnellsten erreicht.

Sollte jedoch der Greifpunkt nicht exakt in die Greiferlage gebracht werden können, weil die hierzu notwendigen Freiheitsgrade aufgrund lokaler oder globaler Degenerationen nicht vorhanden sind, so würde das Verfahren der Rückwärtstransformation hier divergieren und es kommt ein Verfahren zur Anwendung, welches zwangsweise in Kauf zu nehmende bleibende Abweichungen so gut wie möglich minimiert.

Die Entscheidung, ob ein Greifpunkt einen Greifer vollständig erreichen kann, das heißt welches Verfahren in minimalem Zeitaufwand zu einer ausreichenden Lösung führen wird, stellt ein weiteres Problem dar, weil die betrachtete Abweichung zwischen Greifpunkt und Greifer im allgemeinen Fall aus translatorischen und rotorischen Anteilen besteht, deren jeweiliger Anteil an der „Güte“ einer Lösung nicht allgemeingültig angegeben werden kann.

Dabei wird jeweils eine ideale Achskonfiguration (wahlweise) nur innerhalb des zulässigen Wertebereiches (innerhalb der jeweiligen Anschlagswerte) gefunden. Wahlweise wird ferner die Konfiguration errechnet, welche unter Einhaltung gelenkspezifischer Maximalbeträge für Geschwindigkeit und Beschleunigung der idealen am nächsten kommt.

Das erarbeitete Gesamt-Verfahren wird in den Kapiteln 5 und 6 auf die Berechnung allgemeiner Steward-Plattformen und geschlossener kinematischer Ketten erweitert. Die Vorgaben sind dabei wieder die gleichen: Modellierte Greifpunkte sollen im Rahmen der Möglichkeiten dem Greifer angenähert werden.

Grundsätzlich sei noch erwähnt, daß das Verfahren als Teil eines Simulationsprogrammes echtzeitfähig sein muß, das heißt zur Berechnung einer Konfiguration steht nur jeweis ein Sekundenbruchteil zur Verfügung. Daher stößt es bei der Berechnung hochdimensionaler Problemen aufgrund seiner numerischen Vorgehensweise im allgemeinen an Grenzen und verlangt bei vielachsigen Kinematiken von dem jeweiligen Modellierer unter Umständen Geschick in der Wahl entsprechender Berechnungs-Parameter, welche möglichst schnell eine hinreichende Lösung liefern.

3.2 Greifer, Greifpunkte und Greifarten

Eine vom hier erweiterten Programm simulierte Roboterarbeitszelle besteht aus einem oder mehreren Objekten, die wiederum aus einer oder mehreren in sich starren Sektionen bestehen. Die räumliche Lage jeder Sektion im Modell wird jeweils durch eine im Kapitel 2 erläuterten homogene Transformationsmatrix definiert.

Ein Objekt ist an einer Stelle als greifbar modelliert, indem an der entsprechenden Sektion durch eine relative homogene Transformationsmatrix $\underline{\mathbf{T}}_{\text{Greifpunkt}}$ ein Greifpunkt (GP) definiert ist, welcher dann von dem Greifer (G) geführt werden kann, solange dieser geschlossen ist. In jedem Berechnungszyklus werden nacheinander alle Greifpunkte, welche tatsächlich zum jeweiligen Zeitpunkt gegriffen und in ihrer Lage verändert wurden neu berechnet, indem die variablen Achsparameter der ihnen jeweils zugrundeliegenden kinematischen Kette neu bestimmt und überschrieben wird. Diese Kette der abhängigen Gelenke führt immer von der Objektbasis bis zu derjenigen Sektion, an welcher der Greifpunkt definiert ist. Folgende Graphik veranschaulicht den Zusammenhang.

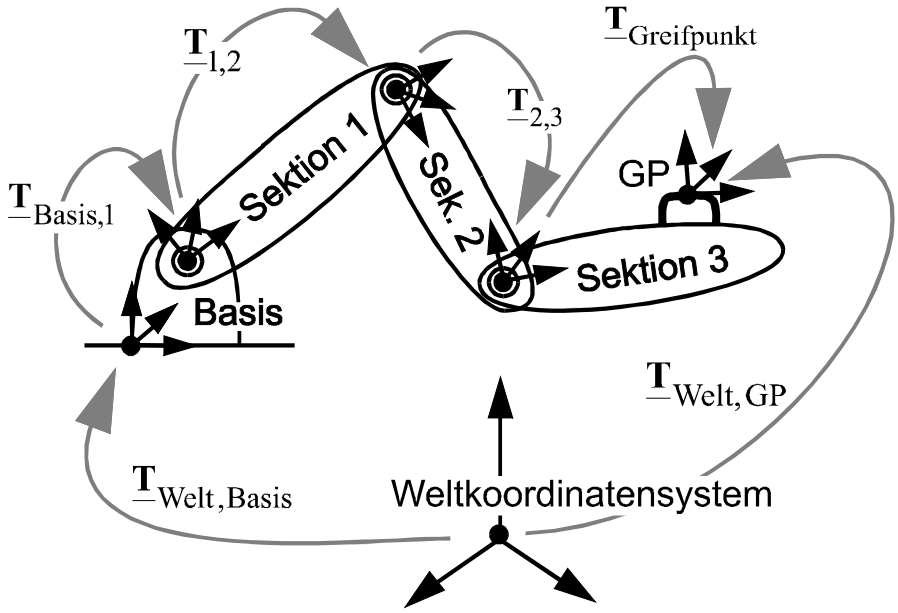

Abbildung 2 : Transformationsmatrizen eines Objektes

Jedem Greifpunkt ist eine Reichweite zugeordnet, welche die Maximalentfernung angibt, in welcher der Punkt gegriffen werden kann. Ein Greifvorgang wird dadurch ausgelöst, daß sich ein Greifer in einem Abstand

$$d = \left| \underline{\text{Pos}}\left(\underline{\mathbf{T}}_{G}\right) - \underline{\text{Pos}}\left(\underline{\mathbf{T}}_{GP}\right) \right|$$

unterhalb dieser Reichweite schließt und so der Punkt mit seiner Sektion als gegriffen gilt.

Es werden zwei unterschiedliche Greifarten unterschieden.

Im ‘SNAP’-Modus wird der reale Greifer selbst als die für den Greifpunkt in jedem Simulationsschritt zu erreichende Lage definiert. Es wird also eine Art magnetische Anziehung zwischen beiden simuliert. Als Beispiel sei hier eine Schublade angegeben. Diese würde sich, im ‘SNAP’-Modus gegriffen, einem Greifer unter Umständen direkt nach dem Zugriff ein Stück entgegenbewegen um den Abstand zu verringern.

Dieses wird vermieden im ‘NO_SNAP’-Modus, in dem im Augenblick des Zugriffs die relative Transformationsmatrix des Lageunteschiedes zwischen Greifer und Greifpunkt berechnet und während der Dauer des Greifvorganges dem Greifer zugeschlagen wird. Dieses ‘Intermediate Frame’ (Mittelstück-Matrix) sorgt dafür, daß die Simulation durch Einführung eines so entstandenen ‘virtuellen Greifers’ realistischer abläuft. So würde sich beispielsweise eine Schublade im Augenblick des Zugriffs nicht bewegen, ab da jedoch der Bahn des Greifers in möglichst gleicher Relativlage folgen und nicht seiner absoluten Lage wie im SNAP’-Modus.

3.3 Karthesische Lageabweichung und Jakobi-Matrix

In beiden Verfahren ist die Bestimmung der Lageabweichung $\underline{\Delta\ell}$ zwischen Greifer und Greifpunkt der erste Schritt, da diese in beiden Fällen möglichst gut ausgeregelt werden soll.

Hierzu wird zunächst die homogene Transformationsmatrix errechnet, welche das Greifpunkt-koordinatensystem in das des Greifers überführt. Es gilt

$$\underline{\mathbf{T}}_{\text{Welt,GP}} \cdot \underline{\mathbf{T}}_{\Delta} = \underline{\mathbf{T}}_{\text{Welt,G}} \ .$$

Also erhält man $\underline{\mathbf{T}}_{\Delta}$ nach

$$\underline{\mathbf{T}}_{\Delta} = \underline{\mathbf{T}}^{-1}_{\text{Welt,GP}} \cdot \underline{\mathbf{T}}_{\text{Welt,G}} \ .$$

Diese Matrix enthält (wie in Kapitel 2 erläutert) eine 3x3 Rotationsmatrix zur Darstellung der Orientierungsänderung und einen Translationsvektor, der die Ursprungsverschiebung angibt.

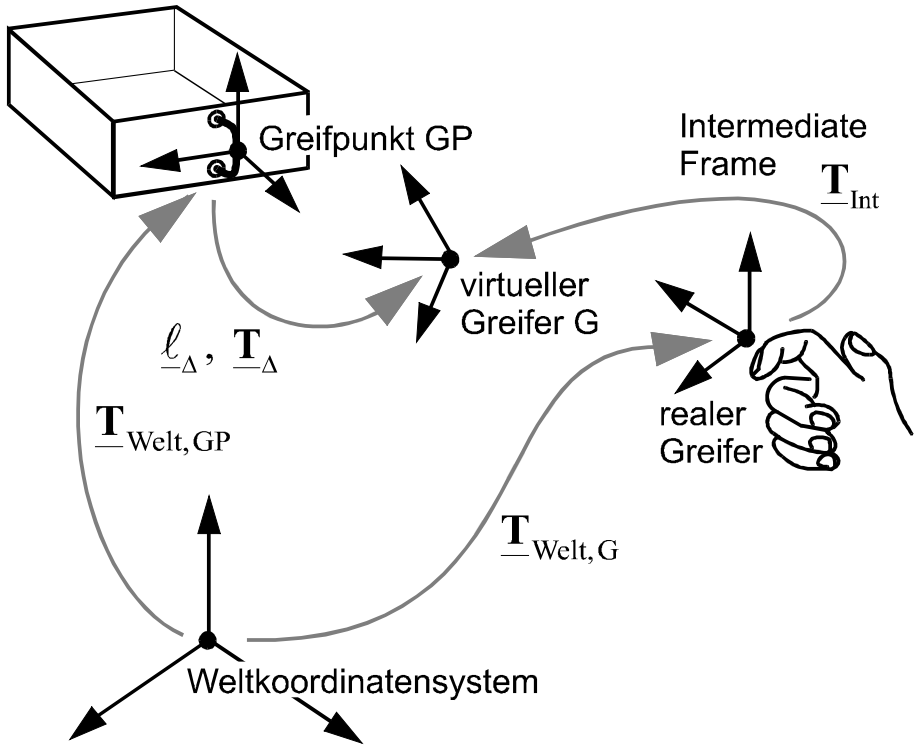

Abbildung 3 : Berechnung der Lagedifferenz zwischen Greifer und Greifpunkt

Die ersten drei Komponenten des Lageabweichungsvektors sind identisch mit dem Translationsvektor in der homogenen Transformationsmatrix, so daß also nur noch die Orientierungsmatrix in die drei Winkelkomponenten von $\underline{\Delta\ell}$ umgerechnet werden muß [3].

Zuerst wird die Matrix in einen normierten Drehvektor und einen zugehörigen Drehwinkel umgewandelt. Der Drehachsenvektor $\underline{v}$ ist ein Eigenvektor der Orientierungsmatrix zum Eigenwert 1. Der Winkel ergibt sich dann aus

$$\Theta = \begin{cases} \arccos\left(\dfrac{\mathrm{spur}(\underline{\mathbf{Rot}}(\underline{\mathbf{T}})) - 1}{2}\right) & , \text{falls } \det\left|\underline{v},\ \underline{n},\ \underline{\mathbf{Rot}}(\underline{\mathbf{T}})\cdot\underline{n}\right| > 0 \\ -\arccos\left(\dfrac{\mathrm{spur}(\underline{\mathbf{Rot}}(\underline{\mathbf{T}})) - 1}{2}\right) & , \text{falls } \det\left|\underline{v},\ \underline{n},\ \underline{\mathbf{Rot}}(\underline{\mathbf{T}})\cdot\underline{n}\right| < 0 \end{cases}$$

worin $\underline{n}$ ein beliebiger zu $\underline{v}$ senkrechter Vektor ist.

Der normierte Drehachsenvektor multipliziert mit dem Winkel ergibt dann gerade die Rotationsanteile um die Hauptachsen des Greifpunkt-Koordinatensystems, welche gesucht waren.

$$\underline{\Delta\ell} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta\varphi_x \\ \Delta\varphi_y \\ \Delta\varphi_z \end{pmatrix} = \begin{pmatrix} \underline{\Delta p} \\ \underline{v} \cdot \theta \end{pmatrix}$$

Dieser Lageabweichungvektor ist auf den Greifpunkt bezogen, das heißt er gibt die notwendige Translation und Rotation des Greifpunkt-Koordinatensystems an, um es in das des Greifers zu transformieren.

Die in Kapitel 2 erläuterte Jakobi-Matrix ist bezogen auf das Weltkoordinatensystem.
Beide müssen jedoch zwangsweise ein gleiches Bezugssystem aufweisen. Es besteht prinzipiell die Möglichkeit jedes beliebige System hierfür zu wählen, es ist jedoch für die hier betrachtete Aufgabenstellung (wie sich später zeigt) sehr viel günstiger als Bezugssystem das Greifpunkt-Koordinatensystem zu wählen. Folglich muß die bisher auf das Weltkoordinatensystem bezogene Jakobi-Matrix durch

$$\underline{\mathbf{J}}\big|_{GP} = \begin{pmatrix} \underline{\mathbf{Rot}}\left(\underline{\mathbf{T}}_{Welt,GP}\right)^{-1} & \underline{\mathbf{0}} \\ \underline{\mathbf{0}} & \underline{\mathbf{Rot}}\left(\underline{\mathbf{T}}_{Welt,GP}\right)^{-1} \end{pmatrix} \cdot \underline{\mathbf{J}}\big|_{Welt}$$

ins neue Bezugssystem gedreht werden [6].

3.4 Entscheidungskriterium für vollständige Erreichbarkeit

Es kann im allgemeinen solange keine scharfe Aussage darüber gemacht werden, ob ein Greifpunkt in allen gewünschten Lagekompomenten den Greifer erreichen kann, bis eine solche Lage gefunden ist. Von dieser Entscheidung hängt jedoch der rechenzeitoptimale Einsatz einer der beiden in den folgenden Kapiteln beschriebenen Verfahren ab.
Beide verringern schrittweise die Lageabweichung, indem entsprechende Konfigurationen der Gelenkparameter gesucht werden, wobei jedoch das erste Verfahren der Universaltransformation im Falle der Nichterreichbarkeit eine divergente Lösungsfolge liefert. Die Schwierigkeit besteht darin, daß die betrachtete Abweichung zwischen Greifer einen Greifpunkt im allgemeinen Fall aus einem translatorischen und einem rotorischen Anteil besteht, deren jeweiliger Anteil an der Güte einer Lösung nicht allgemeingültig angegeben werden kann. So ist beispielsweise unklar, ob eine Lösung mit großer translatorischer Abweichung und kleiner rotorischer ‘besser’ oder ‘schlechter’ als eine mit umgekehrten Verhältnissen ist.

Zur Lösung wird hier ein allgemeiner Gewichtungsvektor $\underline{g}$ eingeführt, welcher eine *ungefähre* Entscheidung zuläßt und die hiermit gewichteten Lageabweichungen der durch Universaltransformationen produzierten Lösungsfolgen untersucht.

Stellt sich eine auffällige Divergenz ein, so wird das stets stabile zweite Verfahren der Abstandsminimierung verwendet, welches dann unter Inkaufnahme einer etwas längeren Rechenzeit eine möglichst gute Annäherung herbeiführt.

Als Güte einer Lösung sei die Variable D eingeführt, mit

$$D^{(i)} = \sum_{k=1}^{6} g_k \cdot \Delta\ell_k^{(i)2}$$

Im günstigsten Fall wird dieses Abweichungsmaß schnell gegen Null konvergieren und die Berechnung beendet, sobald sämtliche relevanten Komponenten von $\underline{\Delta\ell}$ unterhalb festgesetzter (zur Greifpunktdefinition gehörender) Schranken liegen. Nichtbetrachtete Anteile der Abweichung wie etwa die Rotationsdifferenz von Schubladen werden dadurch ignoriert, daß entsprechende Komponenten des Gewichtungsvektors $\underline{g}$ auf Null gesetzt werden und somit diese Abweichungen nicht im Fehlermaß auftauchen.

Stellt sich jedoch eine bleibende Divergenz ein, so wird diese durch ein Verfahren erkannt, welche in etwa eine menschliche Entscheidung simuliert (Fuzzy Regler).
Zunächst wird vor der ersten Iteration ein Wert, welcher die 'Verbesserungsqualität' angibt, auf 10 gesetzt. Nach jeder Iteration wird dann geprüft, ob sich das Abweichungsmaß verringert oder erhöht hat. Im Falle einer Verringerung wird der Wert um 2 erhöht, sonst um 4 (bzw. während der ersten 2 iterationen nur um 1, da anfängliche Verschlechterungen weniger Gewicht bekommen sollen) verringert. Anschließend wird immer mit 0.8 multipliziert. Spätere Verschlechterungen werden somit nochmals grundsätzlich stärker gewichtet.
Sollte der Wert negativ werden oder die Anzahl der Iterationen die doppelte Gelenkanzahl übersteigen (bzw nach jeweils 10 Simulationsschritten der Nicht-Erreichbarkeit den Wert 30), so wird eine Nicht-Erreichbarkeit der Greiferlage angenommen und mit dem zweiten Verfahren gerechnet.
Auf diese Weise wird eine schnelle und einigermaßen effektive Erkennung der Erreichbarkeit und somit eine relativ kurze Rechenzeit möglich.

3.5 Begrenzungen in Achsbewegungen

Eine Notwendigkeit zur Begrenzung der Achsbewegungen ergibt sich aus der Vorgabe von eventuellen Betrags-Maximalwerten für Geschwindigkeit und Beschleunigung jeder Achse. Gibt N die Gesamtzahl der zu variierenden Achsen an, so muß stets gelten

$$|\dot{q}_n| \leq v_{max,\,n} \quad \text{und} \quad |\ddot{q}_n| \leq a_{max,\,n} \quad , \forall n = 1..N$$

Dieses scheint zwar für eine Schublade oder eine Klapptür nicht viel Sinn zu machen, ist jedoch beinahe unerlässlich, falls massive Kinematiken realitätsnahe simuliert werden sollen. Außerdem fordert die Führungssimulation greifbarer aktiver Anordnungen, wie etwa von Roboterarmen, aufgrund physikalischer Trägheiten und der endlichen Drehmomente der Motoren derartige Einschränkungen in den Bewegungsmöglichkeiten. Allgemein gilt

$$q_n^{(s)}\left(\Delta t^{(s)}\right) = q_n^{(s)}(0) + \dot{q}_n^{(s)}(0)\Delta t^{(s)} + \int_0^{\Delta t^{(s)}} \frac{\ddot{q}_n^{(s)}\left(t^{(s)}\right)}{2}\, dt^2$$

$\Delta t^{(s)}$ ist dabei die Zeitdauer des gesamten Simulationsschrittes s. $t^{(s)}$ bezeichnet die Zeit *während* Schritt s. Gesucht ist ausgehend von der Sollposition $\tilde{q}_n^{(s)}$ und der Sollgeschwindigkeit

$$\tilde{\dot{q}}_n^{(s)} = \frac{\tilde{q}_n^{(s)} - q_n^{(s)}(0)}{\Delta t^{(s)}} ,$$

die beide (wie alle Sollwerte im folgenden) durch einen geschweiften Überstrich gekennzeichnet sind, für jeden Achsparameter ein neuer Wert und eine neue Geschwindigkeit mit

$$\left| q_n^{(s)}\left(t^{(s)} = \Delta t^{(s)}\right) - \tilde{q}_n^{(s)} \right| \rightarrow \text{Minimum} \qquad \text{(Primäres Ziel)}$$

$$\left| \dot{q}_n^{(s)}\left(t^{(s)} = \Delta t^{(s)}\right) - \tilde{\dot{q}}_n^{(s)} \right| \rightarrow \text{Minimum} \quad \text{(Sekundäres Ziel)}$$

die sowohl die beiden oben genannten Grenzbedingungen im aktuellen Simulationsschritt nicht verletzen, als auch in der Modell-Zukunft eine Einhaltung ermöglichen.
Das bedeutet, daß es nicht erlaubt sein darf eine Achse in einen zulässigen Zustand (bestehend aus zulässigen Werten für DH-Parameter und Geschwindigkeit) zu verfahren, der in einem nachfolgenden Simulationsschritt jedoch aufgrund endlicher Achsanschlagswerte mindestens eine Zwangsnebenbedingung verletzen *müßte*. Wird etwa ein schwerer Roboterarm mit maximaler Geschindigkeit bewegt, so ist es notwendig ihn rechtzeitig vor Erreichen seines Anschlages abzubremsen, um die zulässigen maximalen Verzögerungswerte später nie überschreiten zu müssen. Es muß gelten

$$\left| \dot{q}_n^{(s)}(t) \right| \leq \begin{cases} \sqrt{2\left(q_{max,\,n} - q_n^{(s)}(t)\right) \cdot a_{max,\,n}} & , \text{falls } \dot{q}_n^{(s)}(t) \geq 0 \\ \sqrt{2\left(q_n^{(s)}(t) - q_{min,\,n}\right) \cdot a_{max,\,n}} & , \text{falls } \dot{q}_n^{(s)}(t) < 0 \end{cases} \qquad , \forall n = 1..N ,$$

damit in der Systemzukunft keine ‘Zwangsverletzung’ einer Nebenbedingung auftritt. Die Ungleichung folgt aus der Überlegung, daß die Geschwindiglkeit stets klein genug sein muß, damit eine maximale Verzögerung die Achse noch vor dem (in Bewegungsrichtung liegenden) Anschlag zum Stillstand bringen kann. Ist $t_{min,n}$ die minimale für eine Zwangsbremsung notwendige Zeit, so muß gelten

$$\left| \dot{q}_n^{(s)}\left(t_x\right) \right| = a_{max,\,n} \cdot t_{min,\,n}$$

und

$$\left| q_{\text{Anschlag in Bewegungsrichtung, n}} - q_n^{(s)}\left(t_x\right) \right| = \frac{a_{max,\,n} \cdot t_{min,\,n}^2}{2} .$$

Nach Kürzung von $t_{min,n}$ folgt

$$\frac{\dot{q}_n^{(s)}(t_x)^2}{2\, a_{max,\, n}} = \left| q_{\text{Anschlag in Bewegungsrichtung., n}} - q_n^{(s)}(t_x) \right|$$

und es ergibt sich so die obige Formel. Die maximal zulässige Geschwindigkeit einer Achse ist damit eine Funktion der Bewegungsrichtung und der noch in Bewegungsrichtung verbleibenden ‘Reserve’ bis zum Anschlag. Ist die Geschwindigkeit höher, so wird sie infolge einer ‘Zwangsbremsung’ entsprechend verringert. Die sich dabei ergebende Achsparameterdifferenz

$$\Delta q_n^{(s)}\left(t^{(s)}\right) = \int_0^{\Delta t^{(s)}} \dot{q}_n^{(s)}\left(t^{(s)}\right)\, dt$$

errechnet sich dann aus dem Zeitintegral der Geschwindigkeit, welche ja innerhalb des Zeitintervalls des Simulationsschritt s veränderbar und somit zeitabhängig ist. Es gilt also

$$q_n^{(s)}\left(\Delta t^{(s)}\right) = q_n^{(s)}(0) + \Delta q_n^{(s)}\left(\Delta t^{(s)}\right) \quad .$$

Die Achsparameterdifferenz nach einer Zwangsbremsung muß vom Betrag her nicht stets kleiner als die ursprünglich errechnete sein, da die Bremsung unter Umständen erst nach einer gewissen Zeit beginnen muß und die Achse in dieser Zeit noch beschleunigt werden kann, sofern ihre Maximalgeschwindigkeit noch nicht erreicht ist.

Der gesamte Zusammenhang sei zunächst übersichtshalber für den Fall einer positiven Achsbewegungsrichtung graphisch dargestellt.

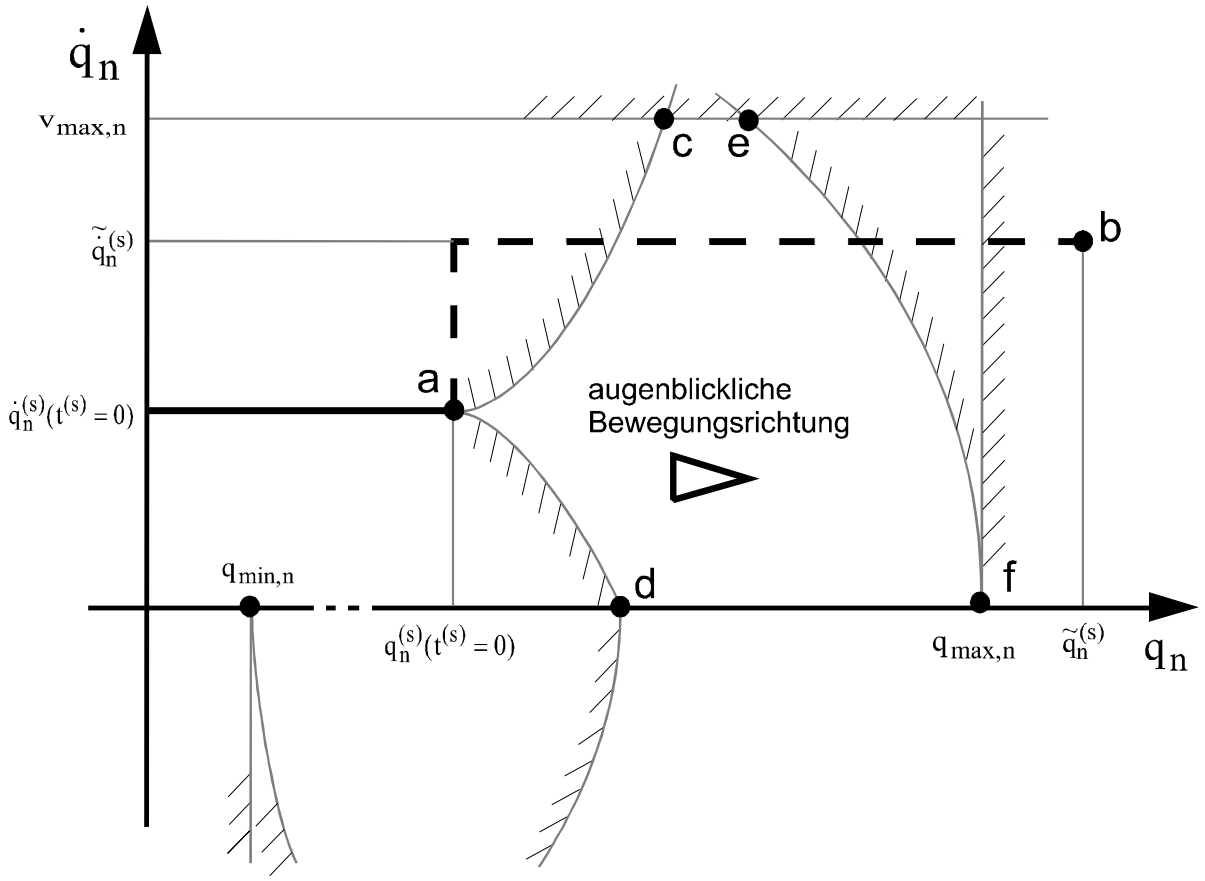

Abbildung 4 : Einzuhaltende Nebenbedingungen eines Simulationsschrittes s

Die gestrichelte Linie (a-b) zeigt den ursprünglich errechneten, idealisierten Geschwindigkeitsverlauf über dem Achsparameter an. Dieser ergibt sich aus der Dauer $\Delta t^{(s)}$ des Simulationsschrittes und der Differenz zwischen gewünschtem Soll-Achswert $\tilde{q}_n^{(s)}$ und dem anfänglichen Istwert $q_n^{(s)}\left(t^{(s)}=0\right)$. Nach Definition eines Richtungsindex

$$r_n^{(s)} = \begin{cases} 1 \text{ , falls } \tilde{\dot{q}}_n^{(s)} \geq \dot{q}_n^{(s)}(0) \\ -1 \text{ , sonst} \end{cases}$$

beträgt die begrenzte Geschwindigkeit im Sollzustand somit

$$\tilde{\dot{q}}_n^{(s)} = \begin{cases} \dfrac{\tilde{q}_n^{(s)} - q_n^{(s)}(0)}{\Delta t^{(s)}} \text{ , falls } \left| \dfrac{\tilde{q}_n^{(s)} - q_n^{(s)}(0)}{\Delta t^{(s)}} \right| < v_{max,\,n} \\ v_{max,\,n} \cdot r_n^{(s)} \text{ , sonst} \end{cases}.$$

Der Zustand a soll nun innerhalb $\Delta t^{(s)}$ möglichst gut in den Zustand b überführt werden, ohne dabei die beschriebenen Nebenbedingungen zu verletzten. Es zeigen im einzelnen die Kurven/Geraden

(a-c) und (a-d)	die Begrenzung der maximalen Beschleunigung/Verzögerung (hier nur für den Startzustand eingezeichnet)
(c-e)	die positive Maximalgeschwindigkeit der Achse
(e-f)	die Grenze zur Zwangsbremsung in Bewegungsrichtung.

Entsprechende Kurven ergeben sich auch im Bereich negativer Geschwindigkeit und/oder Beschleunigung. Zur besseren Übersicht wird in Abbildung 5 die Achsgeschwindigkeit mit gleichen Einschränkungen über der *Zeit* aufgetragen. Die Wurzelfunktionen und Parabeln geht dabei in leicht zu berechnende Linearfunktionen über.

Zunächst sei das Vorgehen ohne Achsanschlagbehandlung erläutert. Anschließend wird diese mit einbezogen.

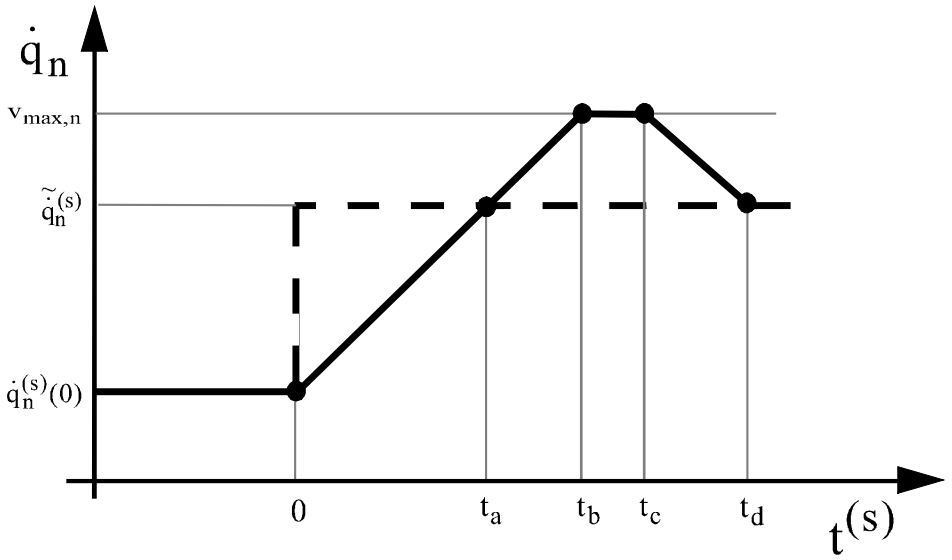

Abbildung 5 : Geschwindigkeits-Zeit-Diagramm eines Simulationsschrittes

Die gestrichelte Kurve zeigt wieder den von der Achse so niemals aufzubringenden Idealverlauf, die durchgezogene eine exemplarische reale Lösung.
Der Idealverlauf zeigt zur Startzeit einen sprunghaften Anstieg, weshalb die Achse zunächst maximal beschleunigt wird, bis die Sollgeschwindigkeit nach

$$t_a = \frac{\left| \tilde{\dot{q}}_n^{(s)} - \dot{q}_n^{(s)}(0) \right|}{a_{max,n}}$$

erreicht ist. (Der Übersicht halber entfällt die Indizierung der Zeiten t_a bis t_d mit dem s des Simulationsschrittes.) Der Betrag der Sollgeschwindigkeit ist ja durch die oben gemachte

Begrenzung stets kleiner oder gleich der Maximalgeschwindigkeit. Ist er gleich, so gilt $t_a = t_b = t_c = t_d$. Um im anderen Fall die zwischen beiden Kurven von $t^{(s)}=0$ bis $t^{(s)}=t_a$ eingeschlossene Fläche zu kompensieren, muß eine möglichst gleichgroße Fläche zwischen t_a und t_d eingeschlossen werden. Eine Fläche entspricht einem Weg oder Winkel, den die Achse zurücklegt, bzw. nicht zurücklegt und 'aufgeholt' werden muß. Es muß folglich gelten

$$\frac{t_a^2 \cdot a_{max,n}}{2} = (t_b - t_a)^2 \cdot a_{max,n} + (t_c - t_b) \cdot (t_b - t_a) \cdot a_{max,n}$$

$$\Leftrightarrow \frac{t_a^2}{2(t_b - t_a)} = (t_b - t_a) + (t_c - t_b)$$

$$\Leftrightarrow \frac{t_a^2}{2} = (t_c - t_a)(t_b - t_a)$$

Wird v_{max} nicht erreicht, so ist $t_c = t_b$. Es gilt somit allgemein mit einer anfänglichen maximal zulässigen Beschleunigungs- bzw. Verzögerungsdauer

$$t_{max} = \frac{\left| v_{max,\,n} \cdot r_n^{(s)} - \dot{q}_n^{(s)}(0) \right|}{a_{max,n}}$$

$$t_b = \begin{cases} t_a\left(1+\frac{1}{\sqrt{2}}\right) & \text{, falls } t_a\left(1+\frac{1}{\sqrt{2}}\right) \le t_{max,\,n} \\ t_{max} & \text{, sonst} \end{cases}$$

$$t_c = \begin{cases} t_a\left(1+\frac{1}{\sqrt{2}}\right) & \text{, falls } t_a\left(1+\frac{1}{\sqrt{2}}\right) \le t_{max,\,n} \\ \frac{t_a^2}{2(t_{max} - t_a)} + t_a & \text{, sonst} \end{cases}$$

$$t_d = t_c + t_b - t_a$$

Der nach einer Zeit t erreichte Achsparameter $q_n^{(s)}(t) = q_n^{(s)}(0) + \Delta q_n^{(s)}(t)$ berechnet sich aus

$$\Delta q_n^{(s)}(t) = \begin{cases} \dot{q}_n^{(s)}(0)\cdot t + \dfrac{a_{max,\,n}\cdot t^2}{2}\cdot r_n^{(s)} & , \text{falls} \quad t \le t_b \\ \dot{q}_n^{(s)}(0)\cdot t_b + \left(\dfrac{a_{max,\,n}\cdot {t_b}^2}{2} + v_{max,\,n}\cdot\left(t - t_b\right)\right)\cdot r_n^{(s)} & , \text{falls } t_b < t < t_c \\ \Delta q_n^{(s)}\left(t_c\right) + \dot{q}_n^{(s)}\left(t_c\right)\cdot\left(t - t_c\right) - \left[\dfrac{a_{max,\,n}\cdot\left(t - t_c\right)^2}{2}\right]\cdot r_n^{(s)} & , \text{falls } t_c \le t < t_d \\ \tilde{\dot{q}}_n^{(s)}\cdot t & , \text{falls } t_d \le t \end{cases}$$

und für die Achsgeschwindigkeit an dieser Stelle gilt entsprechend

$$\dot{q}_n^{(s)}(t) = \begin{cases} \dot{q}_n^{(s)}(0) + a_{max,\,n}\cdot t\cdot r_n^{(s)} & , \text{falls} \quad t \le t_b \\ v_{max,\,n}\cdot r_n^{(s)} & , \text{falls } t_b < t < t_c \\ \dot{q}_n^{(s)}(0) + a_{max,\,n}\cdot\left(t_b - \left(t - t_c\right)\right)\cdot r_n^{(s)} & , \text{falls } t_c \le t < t_d \\ \tilde{\dot{q}}_n^{(s)} & , \text{falls } t_d \le t \end{cases}$$

Falls $\Delta t^{(s)} \ge t_d$ wird also der Sollzustand exakt erreicht und mit $\Delta t^{(s)} \le t_c$ entspricht der erreichte Zustand den optimal möglichen Werten.

3.5.1 Weitere Optimierungsmöglichkeit

Ist jedoch $t_c < \Delta t^{(s)} < t_d$, so ist noch eine Optimierung möglich, indem man versucht die durch die Bedingung $t^{(s)} \le \Delta t^{(s)}$ abgeschnittene Fläche noch zwischen t_b und t_c auszugleichen. Es werden drei neue Zeitpunkte t_b' , t_c' und t_d' errechnet, welche dann ein optimales Ergebnis bewirken.

Falls mit $\Delta t^{(s)} \le t_{max,\,n}$ gilt

$$\left(q_n^{(s)}(0) + \dot{q}_n^{(s)}(0)\cdot\Delta t^{(s)} + \frac{a_{max,n}\cdot r_n^{(s)}\cdot{\Delta t^{(s)}}^2}{2}\right)\cdot r_n^{(s)} < \tilde{q}_n^{(s)}\cdot r_n^{(s)} \quad ,$$

oder mit $\Delta t^{(s)} > t_{max,\,n}$ gilt

$$\left(q_n^{(s)}(0) + \dot{q}_n^{(s)}(0)\cdot t_{max} + \left(\frac{a_{max,n}\cdot {t_{max}}^2}{2} + v_{max,n}\cdot\left(\Delta t^{(s)} - t_{max}\right)\right)\cdot r_n^{(s)}\right)\cdot r_n^{(s)} < \tilde{q}_n^{(s)}\cdot r_n^{(s)} \quad ,$$

so ist die Sollposition innerhalb $\Delta t^{(s)}$ nicht erreichbar und es wird mit

$$t_b' = t_{max}$$
$$t_c' = t_d' = \infty$$

das optimales Ergebnis erzielt.

Andernfalls ist der Sollwert des Achsparameters (nicht mit Sollgeschwindigkeit) erreichbar, da die 'abgeschnittene' Fläche vollständig kompensierbar ist.

Zur Lösung wird zunächst ein Sollwert für t_b' unter Vernachlässigung der Maximalgeschwindigkeit berechnet. Die untenstehende Formel hat dann eine eindeutige Lösung, da vollständige Kompensation erreicht werden kann.
(Achtung bei der Unterscheidung zwischen t_b und t_b').

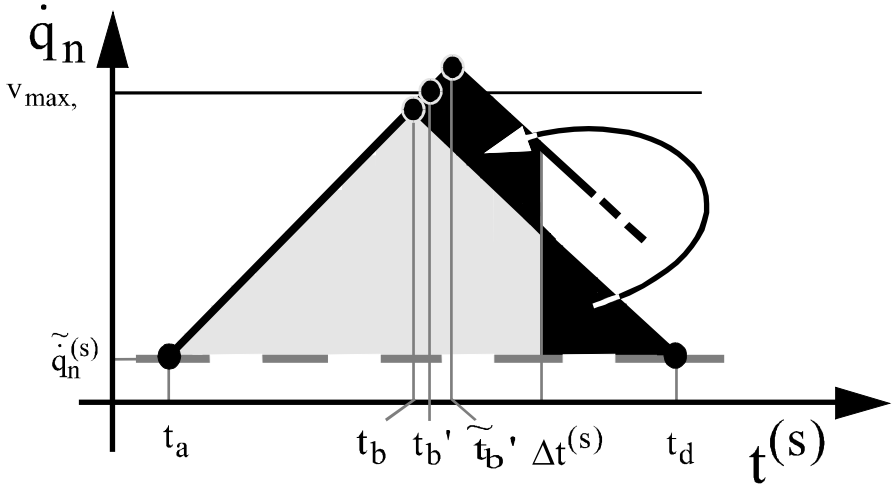

Abbildung 6 : Berechnung einer Kompensationsfläche im Falle der Erreichbarkeit der Sollposition

$$\left(\tilde{t}_b'^2 - t_b^2 - \frac{\left(2\,\tilde{t}_b' - \Delta t^{(s)}\right)^2 - \left(2\,t_b - \Delta t^{(s)}\right)^2}{2} \right) \cdot a_{max,\,n} = \frac{\left(\dot{q}\left(\Delta t^{(s)}\right) - \tilde{\dot{q}}_n^{(s)}\right)^2}{2\,a_{max,\,n}} + \left(t_c - t_b\right) \cdot v_{max,\,n}$$

$$\Leftrightarrow -\tilde{t}_b'^2 + t_b^2 + 2\tilde{t}_b'\Delta t^{(s)} - 2t_b\Delta t^{(s)} - \frac{\left(\dot{q}\left(\Delta t^{(s)}\right) - \tilde{\dot{q}}_n^{(s)}\right)^2}{2a_{max,\,n}^2} - \left(t_c - t_b\right) \cdot \frac{v_{max,\,n}}{a_{max,\,n}} = 0$$

$$\Leftrightarrow \tilde{t}_b' = \Delta t^{(s)} \pm \sqrt{\Delta t^{(s)2} - 2\,t_b\,\Delta t^{(s)} + t_b^2 - \frac{\left(\dot{q}\left(\Delta t^{(s)}\right) - \tilde{\dot{q}}_n^{(s)}\right)^2}{2\,a_{max,\,n}^2} - \left(t_b - t_c\right) \cdot \frac{v_{max,\,n}}{a_{max,\,n}}}$$

Da hier immer $\tilde{t}_b' \le \Delta t^{(s)}$ gelten muß folgt

$$\Rightarrow \tilde{t}_b' = \Delta t^{(s)} - \sqrt{\left(\Delta t^{(s)} - t_b\right)^2 - \frac{\left(\dot{q}\left(\Delta t^{(s)}\right) - \tilde{\dot{q}}_n^{(s)}\right)^2}{2\, a_{max,\,n}{}^2}} - \left(t_c - t_b\right)\cdot\frac{v_{max,\,n}}{a_{max,\,n}}$$

Anschließend wird der Zeitpunkt auf seinen maximal zulässigen Wert begrenzt.

$$t_b' = \mathrm{Min}\left(\tilde{t}_b', t_{max}\right)$$

Ist $t_b' < t_{max}$, so ist $t_c' = t_b'$. Andernfalls berechnet sich t_c' (das immer kleiner als $\Delta t^{(s)}$ sein muß) folgendermaßen

$$t_c' = t_{max} + 2\left(\tilde{t}_b' - t_{max}\right) + \frac{\left(\tilde{t}_b' - t_{max}\right)^2 \cdot a_{max,\,n}}{v_{max,\,n}} .$$

Es wird also der durch die Begrenzung auf Maximalgeschwindigkeit abgeschnittene Teil der Kurve wieder einfach zwischen t_b' und t_c' kompensiert (Flächengleichheit). Ferner gilt

$$t_d' = t_c' + t_b' - t_a .$$

3.5.2 Berücksichtigung von Zwangsbremsungen

Wird nun ebenfalls die Zwangsbremsbedingung mitberücksichtigt, liegt also der Zielzustand jenseits der ‘Zwangsbremsungs’-Geraden und gelangt die bisherige Lösung ebenfalls in den verbotenen Bereich (was nicht immer der Fall sein muß), so erweitet sich die Rechnung um einen Term, welcher die in folgender Abbildung exemplarisch dargestellte ‘abgeschnittene’ Fläche zwischen Grenze und Lösungsverlauf mitberücksichtigt.

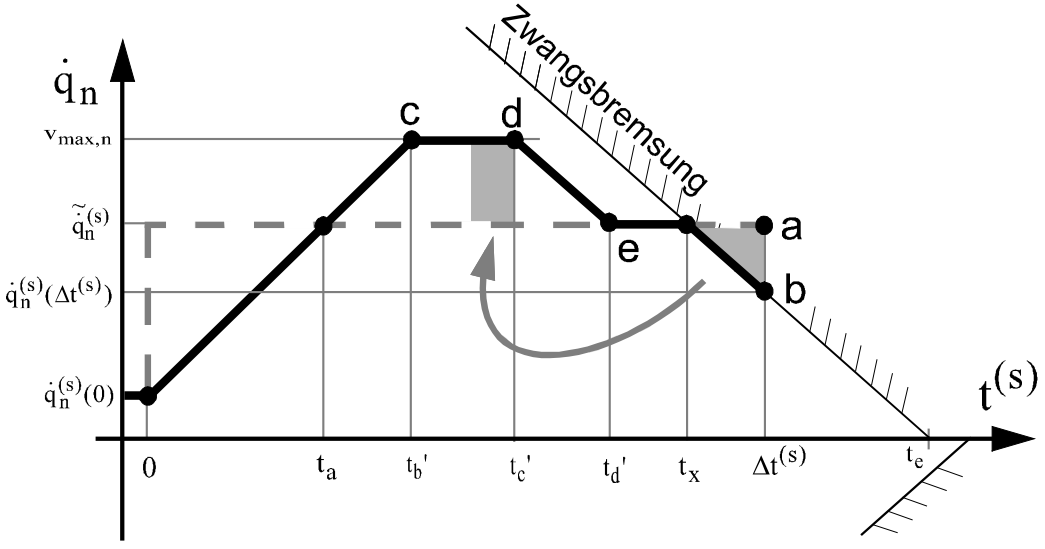

Abbildung 7 : Kompensation der Abweichung aufgrund einer Zwangsbremsung

Da dieser (der Flächengröße entsprechende) Teil des Verfahrweges der Achse unter Umständen ebenfalls durch entsprechende vorherige Geschwindigkeitsänderungen aufzubringen ist, wird erneut versucht ihn auf der gegenüberliegenden Seite der Sollgeschwindigkeits-Geraden auszugleichen, indem die Achse entsprechend länger beschleunigt bzw erst später gebremst wird.

In den Fällen, in denen die neu zu kompensierende Fläche auf der gleichen Seite der Sollgeschwindigkeits-Geraden liegt wie die zwischen t_a und t_d liegenden Fläche, wird zunächst versucht letztere entsprechend zu verkleinern. Reicht dieses nicht aus (t_d=t_a), vergrößert man die Fläche zwischen t=0 und t_a, indem man die Beschleunigungsrichtung am Anfangspunkt invertiert und so eine entsprechend möglichst gute Kompensationsfläche erzielt.

Ist insgesamt eine vollständige Kompensation möglich, ohne daß Punkt e hierbei hinter die Grenze gerät, so kann der Sollwert $\tilde{q}_n^{(s)}$ exakt erreicht werden (die Geschwindigkeiten bis zu Punkt a bzw. b haben dann den gleichen Mittelwert), wenngleich mit anderer Geschwindigkeit als ursprünglich gewünscht (Punkt b statt Punkt a). Gerät Punkt e durch den Ausgleichsversuch hinter die Grenze, so erreicht man die Sollposition $\tilde{q}_n^{(s)}$ nicht ganz, sondern einen Wert, der sich wieder aus der Integration des Geschwindigkeitsverlaufes, der sich nach der Kompensation ergibt, errechnen läßt.

Die Reihenfolge der Lösungsschritte im Überblick :

1) Bestimmung des 'Kollisionszeitpunktes' t_x der Kurve mit einer Grenzbedingung zur Zwangsverzögerung der Achsbewegung.
1) Berechnung eines Geschwindigkeitsverlaufs durch Neuberechnung der Zeiten t_a bis t_d, der die zu kompensierenden Fläche zwischen der Grenzgeraden und dem bisherigen Kurvenverlauf im Geschwindigkeits - Zeit - Diagramm möglichst kompensiert.
1) Integration der neu errechneten Kurve bis $\Delta t^{(s)}$ zur Berechnung von idealem Achszustand am Ende von Simulationsschritt s.

Um zunächst also die Zeit t_x zu berechnen wird die Gleichung

$$\dot{q}_n^{(s)}(t_x) = \begin{cases} \sqrt{2\left(q_{max,\,n} - q_n^{(s)}(t_x)\right)\cdot a_{max,\,n}} & ,\text{falls } \dot{q}_n^{(s)}(t_x) > 0 \\ -\sqrt{2\left(q_n^{(s)}(t_x) - q_{min,\,n}\right)\cdot a_{max,\,n}} & ,\text{falls } \dot{q}_n^{(s)}(t_x) < 0 \end{cases}$$

unter Zuhilfenahme von Fallunterscheidungen gelöst. Dabei wird der Geschwindigkeitsverlauf in seine linearen Anteile zerlegt und getrennt (in zeitlicher Reihefolge) auf Kollision mit der Grenzgeraden untersucht.
Liegt die Kollisionszeit innerhalb des jeweils untersuchten Intervalls, so ergibt sich hieraus t_x und die Suche bricht ab. Andernfalls ist ein Intervall nicht grenzverletzend. In beschleunigten und verzögerten Abschnitten wird zwischen den beiden Werten des Richtungsindex unterschieden, da bei $r_n^{(s)}$ = 1 die Grenze nur oberhalb, bei $r_n^{(s)}$ = -1 nur unterhalb der Zeitachse verletzt werden kann, was die Fallunterscheidung in obiger Formel auflöst.

Zu beachten ist, daß die beiden Grenzgeraden im allgemeinen die Zeitachse nicht in einem Punkt berühren und prinzipiell immer offen ist, welche Grenze die Zeitachse später schneidet, da der 'Kollisionszeitpunkt' t_x vom Geschwindigkeitsprofil abhängt.
Es gilt jedoch immer, daß in den Diagrammen der Schnittpunkt der Abszisse mit der Grenze, welche dem gegenwärtigen Zustand bezüglich der Abzisse gegenüberliegt, rechts vom augenblicklichen Zustand liegt. Nur die augenblickliche Bewegungsrichtung der Achse entscheidet darüber, welche Gerade überhaupt zum betrachteten Zeitpunkt überschritten werden kann. Streng genommen handelt es sich also nicht um Grenzgeraden, sondern um 'Grenz-Strahlen'.

Die Lösung wird in diesem Kapitel übersichtlichkeitshalber nicht ausführlich hergeleitet. Die gesamte Rechnung befindet sich jedoch in Anhang C. Hier sei nur das Endergebnis angegeben. Mit

$$q_{lim1,\,n} = \begin{cases} q_{max,\,n} & , \text{ falls } r_n^{(s)} = 1 \\ q_{min,\,n} & , \text{ falls } r_n^{(s)} = -1 \end{cases}$$

$$q_{lim2,\,n} = \begin{cases} q_{min,\,n} & , \text{ falls } r_n^{(s)} = 1 \\ q_{max,\,n} & , \text{ falls } r_n^{(s)} = -1 \end{cases}$$

$$q_{lim3,\,n} = \begin{cases} q_{max,\,n} & , \text{ falls } \tilde{\dot{q}}_n^{(s)} \geq 0 \\ q_{min,\,n} & , \text{ falls } \tilde{\dot{q}}_n^{(s)} < 0 \end{cases}$$

gilt

$$t_x = \begin{cases} -\dfrac{\dot{q}_n^{(s)}(0)\cdot r_n^{(s)}}{a_{max,\,n}} \pm \sqrt{\dfrac{\dot{q}_n^{(s)}(0)^2}{2\,a_{max,\,n}^2} + \dfrac{\left(q_{lim1,\,n} - q_n^{(s)}(0)\right)\cdot r_n^{(s)}}{a_{max,\,n}}} & , \text{falls} \quad t_x \leq t_b' \\ t_b' - \dfrac{v_{max,\,n}}{2\,a_{max,\,n}} + \dfrac{\left(q_{lim1,\,n} - q_n^{(s)}\left(t_b'\right)\right)\cdot r_n^{(s)}}{v_{max,\,n}} & , \text{falls } t_b' < t_x < t_c' \\ t_c + \dfrac{\dot{q}_n^{(s)}\left(t_b'\right)\cdot r_n^{(s)}}{a_{max,\,n}} \pm \sqrt{\dfrac{\dot{q}_n^{(s)}\left(t_b'\right)^2}{2\,a_{max,\,n}^2} - \dfrac{\left(q_{lim2,\,n} - q_n^{(s)}\left(t_c'\right)\right)\cdot r_n^{(s)}}{a_{max,\,n}}} & , \text{falls } t_c' \leq t_x < t_d \\ t_d - \dfrac{\tilde{\dot{q}}_n^{(s)2}}{2\,a_{max,\,n}\cdot\left|\tilde{\dot{q}}_n^{(s)}\right|} + \dfrac{q_{lim3,\,n} - q_n^{(s)}\left(t_d'\right)}{\tilde{\dot{q}}_n^{(s)}} & , \text{falls } t_d \leq t_x \end{cases}$$

Ist nun $t_x \geq \Delta t^{(s)}$, so tritt im Berechnungszeitraum keine Grenzverletzung auf und mit den bisherigen Zeitwerten läßt sich wie oben ein Lösungszustand ausrechnen.

Andernfalls berechnet man zunächst die Zeit

$$t_e = t_x + \frac{\left| \dot{q}_n^{(s)}(t_x) \right|}{a_{max,\, n}},$$

nach der die Achse aufgrund der Zwangsbremsung in Bewegungsrichtung zum Sillstand kommen würde.

Das Zeitintegral des Lösungsgeschwindigkeitsverlaufes unter Einbeziehung der Zwangsbremsung ergibt dann wieder den von Achse n im Schritt s zurückgelegten Weg bzw. Winkel

$$\Delta q_n^{(s)}(t) = \begin{cases} \text{wie bisher} & \text{, falls } t \leq t_x \\ \Delta q_n^{(s)}(t_x) + \dot{q}_n^{(s)}(t_x) \cdot (t - t_x) - \frac{a_{max,\, n} \cdot (t - t_x)^2}{2} \cdot \frac{\dot{q}_n^{(s)}(t_x)}{\left| \dot{q}_n^{(s)}(t_x) \right|} & \text{, falls } t_x < t < t_e \\ \Delta q_n^{(s)}(t_x) + \frac{a_{max,\, n} \cdot (t_e - t_x)^2}{2} \cdot \frac{\dot{q}_n^{(s)}(t_x)}{\left| \dot{q}_n^{(s)}(t_x) \right|} & \text{, falls } t_e \leq t \end{cases}$$

Für die Geschwindigkeit gilt entsprechend

$$\dot{q}_n^{(s)}(t) = \begin{cases} \text{wie bisher} & \text{, falls } t \leq t_x \\ \dot{q}_n^{(s)}(t_x) - a_{max,\, n} \cdot (t - t_x) \cdot \frac{\dot{q}_n^{(s)}(t_x)}{\left| \dot{q}_n^{(s)}(t_x) \right|} & \text{, falls } t_x < t < t_e \\ 0 & \text{, falls } t_e \leq t \end{cases}$$

Die Größe der möglicherweise vor t_x zu kompensierenden Fläche erhält man dann durch

$$A_{Komp,\, n} = \Delta q_n^{(s)}\left(\Delta t^{(s)}\right)_{\text{ohne ZB Berücksichtigung}} - \Delta q_n^{(s)}\left(\Delta t^{(s)}\right)_{\text{mit ZB Berücksichtigung}}$$

Falls $t_x \leq t_c'$, so ist keine Verbesserung der bisherigen Lösung mehr möglich.
Andernfalls gibt es zwei oben bereits erwähnte Möglichkeiten.

Falls gilt (erster Fall)

$$t_x \geq t_d' \quad \text{und} \quad r_n^{(s)} \cdot \tilde{\dot{q}}_n^{(s)} > 0 \ ,$$

so wird versucht so gut wie möglich durch Vergrößerung der zwischen t_a und t_d liegenden Fläche zu kompensieren, da diese auf der gegenüberliegenden Seite der Sollgeraden liegt (siehe Beispiel in Abbildung 7). An t_a ändert sich nichts.

Zunächst ist wieder ein theoretisches (unbegrenztes) t_b gesucht, welches die vollständige Kompensation herbeiführen würde. Das Vorgehen vereinfacht sich hier sehr, da falls $t_d \geq t_x$ immer auch $t_c \geq t_x$ gilt. Mit

$$A_{ist,\,n} = \left(t_b' - t_a\right)^2 \cdot a_{max,\,n} + \left(t_b' - t_a\right)\left(t_c' - t_b'\right) \cdot a_{max,\,n}$$

$$\Leftrightarrow A_{ist,\,n} = \left(t_b' - t_a\right)\left(t_c' - t_a\right) \cdot a_{max,\,n}$$

gilt allgemein

$$A_{ist,\,n} + \left|A_{Komp,\,n}\right| = \left(\tilde{t}_b'' - t_a\right)^2$$

$$\Leftrightarrow \tilde{t}_b''^2 - 2\,t_a \tilde{t}_b'' + t_a^2 - \frac{A_{ist,\,n} + \left|A_{Komp,\,n}\right|}{a_{max,\,n}} = 0$$

$$\Rightarrow \tilde{t}_b'' = t_a + \sqrt{\frac{A_{ist,\,n} + \left|A_{Komp,\,n}\right|}{a_{max,\,n}}}$$

Anschließend wird $\tilde{t}_b''$ wieder auf einen Maximalwert begrenzt und anschließend ein hierzu optimales t_c'' bestimmt.

$$t_b'' = \begin{cases} \tilde{t}_b'' & \text{, falls } \tilde{t}_b'' \leq t_{max,\,n} \\ t_{max} & \text{, sonst} \end{cases}$$

Wenn nach t_b'' die Maximalgeschwindigkeit erreicht sein sollte, so gilt

$$\left(\tilde{t}_b'' - t_a\right)^2 \cdot a_{max,\,n} = \left(t_{max} - t_a\right)^2 \cdot a_{max,\,n} + \left(t_c'' - t_{max}\right) \cdot \left(t_{max} - t_a\right) \cdot a_{max,\,n}$$

und somit allgemein nach Auflösung

$$t_c'' = \begin{cases} t_b'' & \text{, falls } t_b'' \leq t_{max,} \\ t_a + \dfrac{\left(\tilde{t}_b'' - t_a\right)^2}{t_{max} - t_a} & \text{, sonst} \end{cases}$$

$$t_d'' = t_c'' + t_b'' - t_a$$

Diese Zeiten führen dann in obige Formeln eingesetzt immer zum optimalen Ergebnis.

Im zweiten möglichen betrachteten Fall ist

$$t_c' < t_x < t_d' \quad \text{oder} \quad r_n^{(s)} \cdot \tilde{\dot{q}}_n^{(s)} < 0 \ .$$

Die zu kompensierende Fläche liegt also mit der zwischen t_a und t_d auf einer Seite der Sollgeschwindigkeits-Geraden. Jetzt kommt prinzipiell sowohl eine Verkleinerung der Fläche zwischen t_a und t_d, als auch eine Vergrößerung der Fläche vor t_a zur Kompensation in Frage.

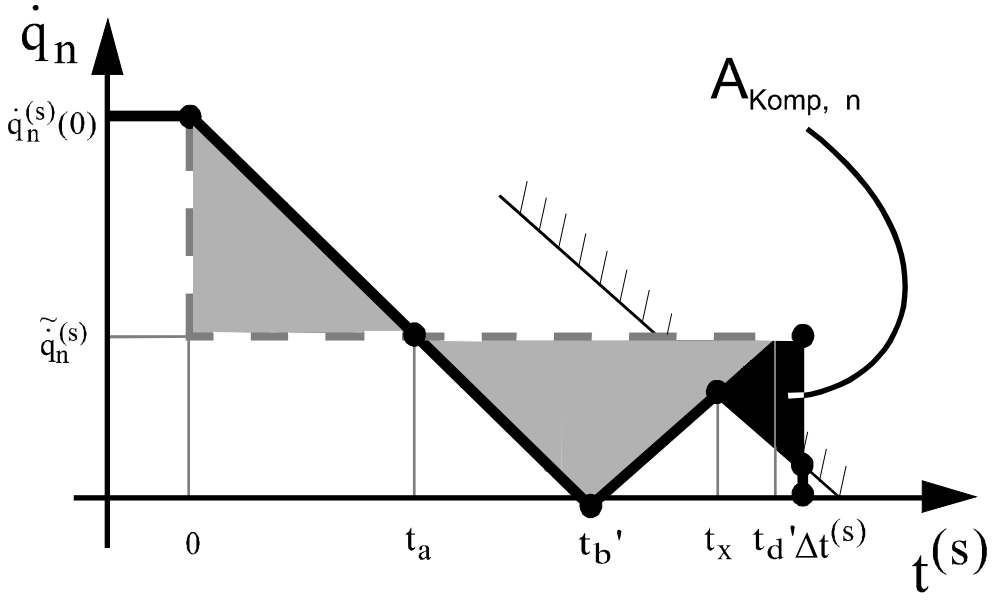

Abbildung 8 : Kompensationsfläche

Eine Kompensation ist vollständig mit der zwischen t_a und t_d liegenden Fläche möglich, wenn sich für $t_d' = t_a$ das Vorzeichen der neuberechnete Kompensationsfläche umkehrt.

Achtung: In der Formel zur Berechnung von t_x ändert sich dann für den Fall $t_x \geq t_d$ die Gleichung in

$$t_x = t_b' - \frac{\left|\tilde{\dot{q}}_n^{(s)}\right|}{2\, a_{max,\ n}} + \frac{\left(q_{lim3,\ n} - q_n^{(s)}\left(t_b'\right)\right)}{\left|\tilde{\dot{q}}_n^{(s)}\right|} \cdot \frac{\tilde{\dot{q}}_n^{(s)}}{\left|\tilde{\dot{q}}_n^{(s)}\right|}$$

Im Falle eines Vorzeichenwechsels existiert immer ein Zeitpunkt $0 < t_b'' < t_b'$, mit dem sich die Sollposition der Achse erreichen läßt. Ferner gilt im hier betrachteten Fall immer

$$\left| \dot{q}_n^{(s)}\left(t_b''\right) \right| < v_{max,\,n}$$

und somit ist stets $t_c'' = t_b''$. Eine analytische Berechnung von t_b'' führt jedoch selbst unter diesen starken Einschränkungen auf ein Polynom vierten Gerades, welches dadurch entsteht, daß die Wurzel in der Berechnung von $t_x = f(t_b'')$ durch Quadieren aufgelöst werden muß.
Eine Faktorzerlegung dieses Polynoms kann jedoch nicht durchgeführt werden, da keine Nullstellen geraten werden können. Es kommt daher ein einfacher Näherungsalgorithmus zum Einsatz, der schnell und problemlos eine Lösung liefert und im folgenden dargestellt ist.

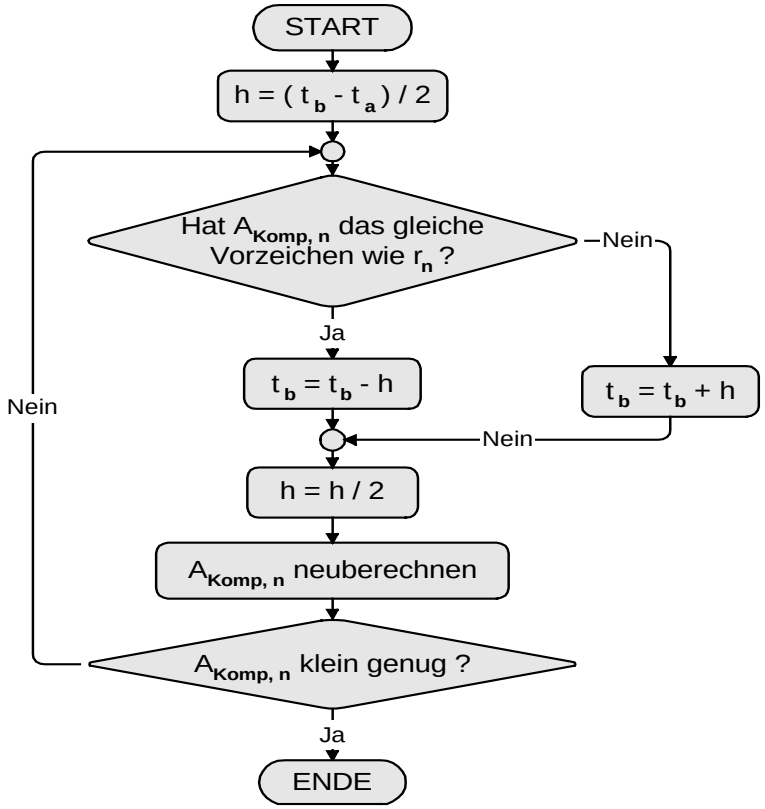

Abbildung 9 : Flußdiagramm zur näherungsweisen Berechnung von t_b

Für den anderen Fall, daß sich das Vorzeichen für $t_d' = t_a$ nicht umgekehrt hat, berechnen sich die Zeitpunkte und damit der optimal erreichbare Zustand folgendermaßen. t_x bezeichnet ab jetzt die Kollisionszeit der Sollgeschwindigkeit mit der Grenze.

Um mit den bisherigen Formeln weiterhin rechnen zu können, wird zunächst *der Richtungsindex negiert*, um die Fläche vor t_a vergrößern zu können. Ferner gilt ab jetzt

$$t_a'' = 0 \ .$$

Anschließend wird ein Zeitpunkt $t_{b,\,max}$'' bestimmt, welcher entweder der Kollisionszeit t_x' bei Nichtbeachtung der Maximalgeschwindigkeit oder der Zeit zur Erreichung der Maximalgeschwindigkeit $t_{max,\,n}$ entspricht, je nach dem welcher der Werte kleiner ist.

Dann wird analytisch aus einem nachträglich begrenzten Sollwert ein t_c'' errechnet.

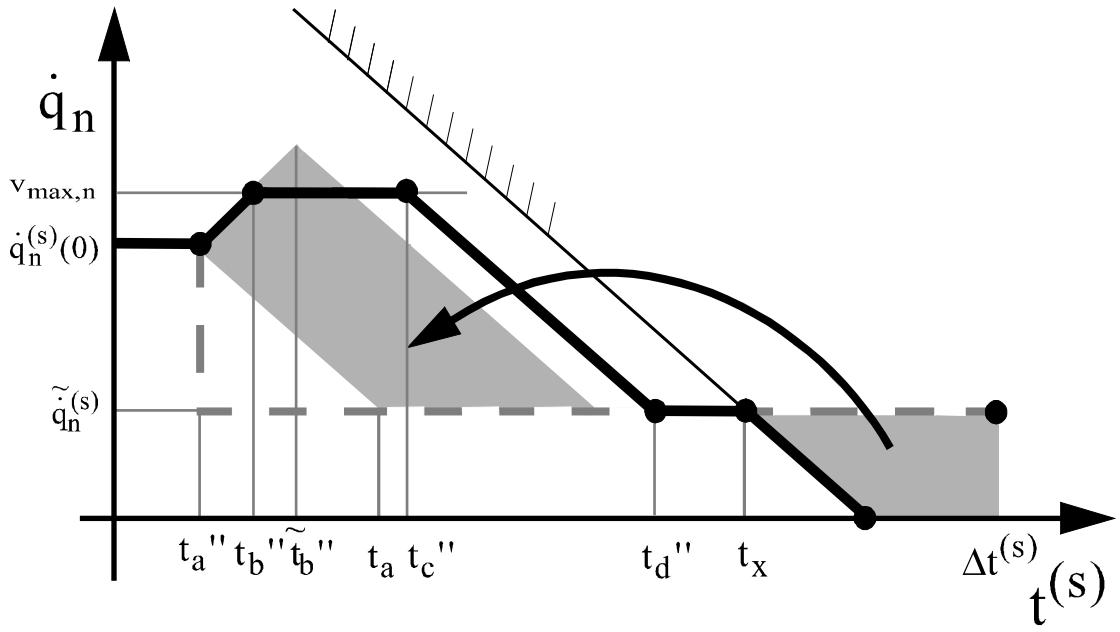

Abbildung 10 : Kompensationsfläche

$$\tilde{t}_b''^{\,2} \cdot a_{max,\,n} + \left| \dot{q}_n^{(s)}(0) - \tilde{\dot{q}}_n^{(s)} \right| \cdot 2\,\tilde{t}_b'' = \left| A_{Komp,\,n} \right|$$

$$\Leftrightarrow \tilde{t}_b''^{\,2} + \frac{2\left| \dot{q}_n^{(s)}(0) - \tilde{\dot{q}}_n^{(s)} \right|}{a_{max,\,n}} \cdot \tilde{t}_b'' - \frac{\left| A_{Komp,\,n} \right|}{a_{max,\,n}} = 0$$

$$\Rightarrow \tilde{t}_b'' = -\frac{\left| \dot{q}_n^{(s)}(0) - \tilde{\dot{q}}_n^{(s)} \right|}{a_{max,\,n}} + \sqrt{\frac{\left(\dot{q}_n^{(s)}(0) - \tilde{\dot{q}}_n^{(s)} \right)^2}{a_{max,\,n}^{\,2}} + \frac{\left| A_{Komp,\,n} \right|}{a_{max,\,n}}}$$

$$t_b'' = Min\left(\tilde{t}_b'' ,\, t_{b,\,max}'' \right)$$

Falls nun $t_b'' = \tilde{t}_b''$, so ist $t_c'' = t_b''$. Andernfalls berechnet sich t_c'' wie folgt.

$$t_c'' = t_{b,\,max}'' + 2\left(\tilde{t}_b'' - t_{b,\,max}'' \right) + \frac{\left(\tilde{t}_b'' - t_{b,\,max}'' \right)^2 \cdot a_{max,\,n}}{v_{max,\,n}}$$

Und für t_d'' gilt hier immer

$$t_d'' = t_c'' + t_b'' + \frac{\left| \dot{q}_n^{(s)}(0) - \tilde{\dot{q}}_n^{(s)} \right|}{a_{max,\,n}} .$$

Diese Zeiten führen dann in obige Formeln eingesetzt wieder zu einem optimalen Endergebnis, obwohl t_c'' oder t_d'' in einigen Fällen möglicherweise nie erreicht werden.

3.5.3 Vermeidung von Überschwingungen

Das bisherige Hauptziel den Sollwert eines Achsparameters möglichst gut mit der errechneten theoretischen Durchschnittsgeschwindigkeit (=Sollgeschwindigkeit) zu erreichen führt unter Umständen zu Problemen. Tritt nämlich in einem Simulationsschritt eine starke Änderung dieser Geschwindigkeit auf, so beginnt die Kinematik zu schwingen. Besonders deutlich tritt dieser Effekt beispielsweise auf, wenn ein Greifpunkt im 'SNAP-Modus' von einem relativ weit entfernten ruhenden Greifer gegriffen wird und die Simulationsintervalle relativ groß sind .

Die ausgeführte Bewegung ist zwar prinzipiell optimal, da in der Echtzeit-Berechnung einer Kinematik im allgemeinen nichts über den nachfolgenden Sollwert des Achsparameters bekannt ist, jedoch ist es teilweise notwendig diesen Effekt zu unterdrücken.

Das zugrundeliegende Prinzip ist es, hier die Anschlagswerte einer Achse im aktuellen Schritt derart zu verändern, daß aufgrund einer einsetzenden Zwangsbremsung die Sollposition möglichst mit *Null*-Geschwindigkeit erreicht wird. Man kann also ehr von einer Zielbremsung sprechen. Die bisher geltenden Vorgaben ändern sich in

$$\left| \dot{q}_n^{(s)}\left(q_n^{(s)} = \tilde{q}_n^{(s)}\right) \right| \rightarrow \text{Minimum} \qquad \text{(Primäres Ziel)}$$

$$\left| q_n^{(s)}\left(t^{(s)} = \Delta t^{(s)}\right) - \tilde{q}_n^{(s)} \right| \rightarrow \text{Minimum} \qquad \text{(Sekundäres Ziel)}$$

Falls $\tilde{q}_n^{(s)} > q_n^{(s)}(0)$ ergibt sich mit

$$t' = \frac{\left| \dot{q}_n^{(s)}(0) \right|}{a_{max,\,n}}$$

und

$$q'_{max,\,n} = \begin{cases} \text{Max}\left(\tilde{q}_n^{(s)} ,\ \dot{q}_n^{(s)}(0) + \dot{q}_n^{(s)}(0) \cdot t' - \frac{a_{max,\,n} \cdot t'^{\,2}}{2} \right) & , \text{falls } \dot{q}_n^{(s)}(0) > 0 \\ \tilde{q}_n^{(s)} & , \text{sonst} \end{cases}$$

für die temporären Extremwerte

$$q^{(s)}_{max,\,n} = \mathrm{Min}\left(q_{max,\,n}\,,\; q_{max,\,n}{}'\right)$$
$$q^{(s)}_{min,\,n} = q_{min,\,n}$$

Und falls gilt $\tilde{q}^{(s)}_n < q^{(s)}_n(0)$, ergibt sich mit

$$q'_{min,\,n} = \begin{cases} \mathrm{Min}\left(\tilde{q}^{(s)}_n\,,\; \dot{q}^{(s)}_n(0) + \dot{q}^{(s)}_n(0)\cdot t' + \dfrac{a_{max,\,n}\cdot t'^{\,2}}{2}\right) & \text{, falls } \dot{q}^{(s)}_n(0) < 0 \\ \tilde{q}^{(s)}_n & \text{, sonst} \end{cases}$$

für die Extremwerte des Schrittes s

$$q^{(s)}_{min,\,n} = \mathrm{Max}\left(q_{min,\,n}\,,\; q_{min,\,n}{}'\right)\,.$$
$$q^{(s)}_{max,\,n} = q_{max,\,n}$$

Die Lösung erfolgt dann mit den neuen Werten nach dem oben beschriebenen Verfahren.

4 Universaltransformation bei vollständiger Erreichbarkeit

In diesem Kapitel wird das Verfahren zur Berechnung der Achskoordinaten einer zu einem Greifpunkt führenden Kinematik unter der Annahme behandelt, daß es prinzipiell möglich ist den Greifpunkt in sämtlichen betrachteten Raumkoordinaten exakt in die durch den Greifer vorgegebene Lage zu verfahren. Die Vorgehensweise entspricht zunächst einer üblichen Universaltransformation [1], wird jedoch etwas erweitert, um die in der Nähe von Singularitäten auftretenden Instabilitäten möglichst zu stabilisieren.

4.1 Rückwärtstransformationen mittels inverser Jakobi-Matrix

Wie im Kapitel 2 bereits erläutert, gibt die Jakobi-Matrix den linearen Zusammenhang zwischen den karthesischen Koordinatengeschwindigkeiten und den Geschwindigkeiten der Achskoordinaten an.

$$\dot{\underline{\ell}} = \underline{\mathbf{J}} \cdot \dot{\underline{q}}$$

Durch Invertierung der Matrix lassen sich umgekehrt die Achskoordinatengeschwindigkeiten als Funktion der karthesischen Koordinatengeschwindigkeiten errechnen.

$$\dot{\underline{q}} = \underline{\mathbf{J}}^{-1} \cdot \dot{\underline{\ell}}$$

Multipliziert man nun beide Geschwindigkeiten mit dem Zeitinkrement dt, so erhält man die Beziehung

$$\underline{dq} = \underline{\mathbf{J}}^{-1} \cdot \underline{d\ell}$$

Als Näherung läßt sich diese Gleichung, welche nur für infinitisimal kleine Abweichungen gilt, umschreiben in

$$\underline{\Delta q} = \underline{\mathbf{J}}^{-1} \cdot \underline{\Delta \ell} \ .$$

Die Abweichungen $\underline{\Delta q}$ und $\underline{\Delta \ell}$ geben dabei nur noch die Differenz zwischen Soll- und Istwert an, da beim hier betrachteten Lösungsverfahren Beschleunigungen vernachlässigt werden und somit die Geschwindigkeit linearisiert wird.

$$\underline{\Delta q}^{(i)} = \underline{q}^{(i+1)} - \underline{q}^{(i)}$$

$$\begin{aligned} \underline{\Delta \ell}^{(i)} &= \underline{\ell}_{soll} - \underline{\ell}^{(i)}_{ist}(\underline{q}) \\ &= \underline{\ell}_{G} - \underline{\ell}^{(i)}_{GP}(\underline{q}) \end{aligned}$$

Dabei gibt die hochgestellte umklammerte Variable die Anzahl der Iterationen an. Durch mehrmaliges Ausrechnen einer derartigen Achskoordinatenänderung $\underline{\Delta q}$ und entsprechender Aufsummierung

$$\underline{q}^{(s)} = \underline{q}^{(0)} + \sum_{i=0}^{i=s-1} \underline{\Delta q}^{(i)}$$

läßt sich unter oben genannter Einschränkung schnell eine neue Gelenkkonfiguration finden, welche den Greifpunkt der Greiferlage in allen betrachteten Raumkoordinaten annähert, bis alle gewählten Schranken unterschritten sind. s bezeichnet dabei die Nummer des übergeordneten Simulationsschrittes.
Was die Rechengeschwindigkeit betrifft, so ist zu beachten, daß nach jeder Iteration eine Vorwärtstransformation durchgeführt werden muß, die Jakobi-Matrix nach den aktualisierten Sektionskoordinaten neu erstellt und invertiert werden muß.

Die folgende Graphik veranschaulicht das Prinzip des Verfahrens.

Zu beachten ist jedoch, daß $\underline{\ell}$ maximal die Dimension 6 hat, $\underline{q}$ ein Vektor höherer Dimension sein kann und die Graphik somit den Zusammenhang nur symbolisch darstellen kann. Die Tangentensteigungen entsprechen hier den jeweiligen Jakobi-Matrizen.

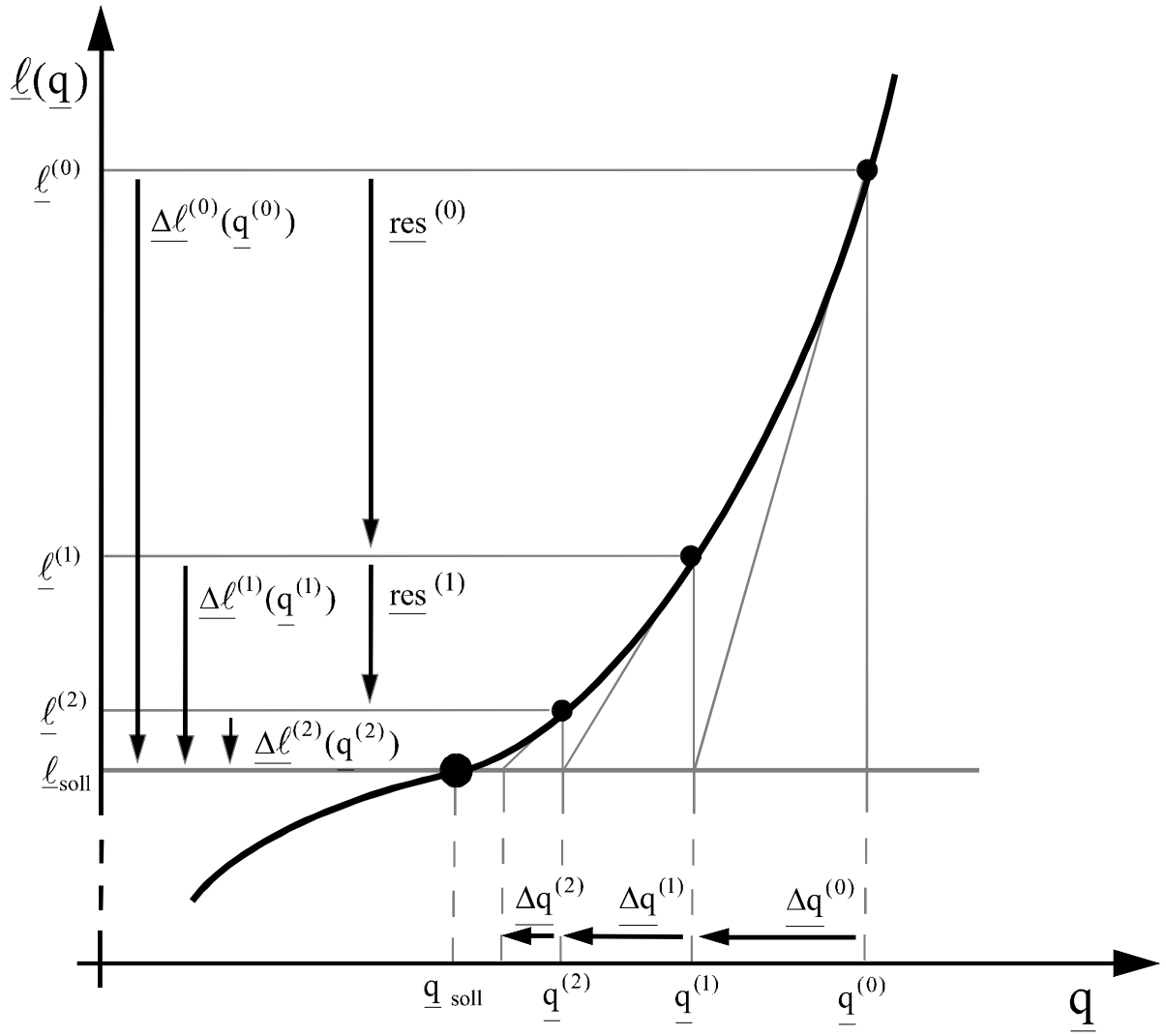

Abbildung 11 : Mehrmalige Rückwärtstransformationen mittels inverser Jakobi-Matrix

Der Abstand vom Greifpunkt zum Greifer verringert sich also im günstigsten Fall jedesmal um $\underline{res}^{(i)}$ und das Verfahren bricht ab, sobald der dem Greifpunkt zugrundeliegende Schrankenvektor in sämtlichen Komponenten kleiner als der Lageabweichungsvektor ist.

Die mit jeder Iteration resultierende karthesische Lageverbesserung ergibt sich durch

$$\underline{res}^{(i)} = \underline{\mathbf{J}} \cdot \underline{\Delta q}^{(i)} \quad .$$

Dieser Zusammenhang sei durch nachfolgende Abbildung verdeutlicht.

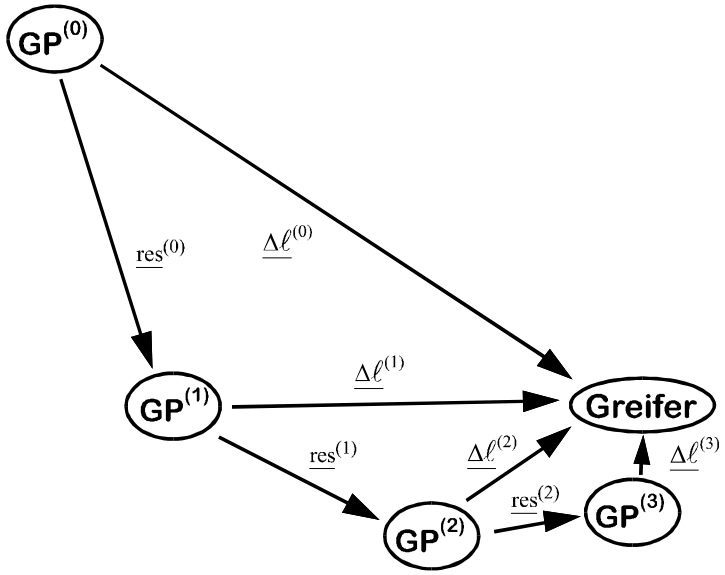

Abbildung 12 : Schrittweise Näherung des Greifpunktes GP an den Greifer

4.2 Reduktion der Jakobi-Matrix und Begrenzung der Schrittweiten

Unter Umständen ist es notwendig Zeilen oder Spalten der Jakobi-Matrix zu kürzen.
Dieses kann notwendig werden um die Zwangsbedingung

$$\det \left| \underline{\mathbf{J}} \right| \neq 0$$

zur Berechnung der Inversen zu erfüllen. Wird die Jakobi-Matrix nach dem im Kapitel 2 geschilderten Verfahren erstellt, so enthält sie 6 Zeilen und ihre Spaltenzahl entspricht der Anzahl der Achsen, welche von der Objektbasis zur Greifpunktsektion führen.

Weisen mehrere Achsen den gleichen Einfluß auf die Lage des Greifpunktes auf, so befindet sich die Kinematik in einer Singularität. Diese wird aufgehoben, indem man diese Achsen zu *einer* transformierten zusammenfaßt und die realen als Anteile dieser definiert. Die entstandene transformierte Achse ist dann immer unabhängig von den verbleibenden.

Die Information über die realen Achsen werden in zwei Vektoren abgelegt. Der erste Vektor gibt für jede Achse die zugehörige Spaltennummer der gekürzten Jakobi-Matrix an, in der der Einfluß der Achse auf die Lage des Greifpunktes nun steht. Der zweite Vektor gibt den Faktor an, mit dem die Achse an der sich ergebenden transformierten Parameteränderung beteiligt ist. Auf diese Art geht keinerlei Information über das Gesamtsystem verloren.

Ist eine gegriffene Kinematik degeneriert, also die Anzahl der unabhängigen Freiheitsgrade des Greifpunktes in seiner aktuellen Lage kleiner als 6, so werden jene Zeilen, welche somit nur Nullen enthalten, ebenfalls aus der bereits spaltengekürzten Jakobi-Matrix gestrichen,.
Der Lageänderungsvektor wird entspechend um die herausgefallenen Zeilen gekürzt und die neue Position der Komponenten ebenfalls durch einen Vektor festgehalten.

Auf diese Art ergibt sich ein transformiertes Modell mit potentiell invertierbarer Jakobi-Matrix, welches nun durch

$$\underline{\dot{\tilde{\ell}}} = \underline{\tilde{\mathbf{J}}} \cdot \underline{\dot{\tilde{q}}}$$

beziehungsweise durch

$$\underline{\dot{\tilde{q}}} = \underline{\tilde{\mathbf{J}}}^{-1} \cdot \underline{\dot{\tilde{\ell}}}$$

definiert ist.

$\underline{\dot{\tilde{\ell}}}$ enthält nur noch Lageänderungen, welche von der Kinematik in ihrer aktuellen Gelenkkonfiguration auch tatsächlich aufgebracht werden können und die transformierten Achsgeschwindigkeiten $\underline{\dot{\tilde{q}}}$ beziehen sich auf Achsen, welche keine singulären Stellungen mehr haben können.

Nach Invertierung und Berechnung entsprechender transformierter Achsgeschwindigkeiten lassen sich beide Vektoren zurücktransformieren und sich so der resultierende Lageunterschied res sowie die realen Achskoordinatenänderungen $\underline{\Delta q}$ berechnen.

Liegt eine vollkommene Singularität vor, so wird die Jakobi-Matrix wie oben erläutert gekürzt um diese aufzuheben. Befindet sich die Kinematik jedoch nur in großer Nähe einer solchen, so lassen sich die entsprechenden Achsen nicht zusammenfassen, da sich ihre Jakobispalten zwar stark ähneln, jedoch nicht identisch sind. Obiges Verfahren greift daher ausschließlich bei reinen Singularitäten wie sie unter Umständen bei singulären Grundstellungen eines Objekts auftreten können.

Da die Jakobi-Matrix mit steigender Singularitätsnähe der Kinematik in mindestens 2 Spalten zunehmend ähnliche Werte aufweist oder weil die Jakobi-Matrix einer Kinematik nahe einer lokalen Degeneration in mindestens einer Zeile sehr kleine Werte enthält, erhöhen sich die Werte der Inversen entsprechend. Das heißt es gilt die Umkehrung, daß Lageänderungen zunehmend größere Achskoordinatenänderungen verursachen, welche dann ihrerseits nach einer Vorwärtstransformation möglicherweise große Änderungen der Lage bewirken.

Daher werden alle neu errechneten Achskoordinatenänderungen von Rotationsachsen um ganzzahlige Vielfache von 2π gekürzt, was auf die resultierende Greifpunktlage keinen Einfluß hat. Die Koordinatenänderungen solcher Achsen liegen also danach zwischen $-\pi$ und π .

Anschließend wird der Achskoordinatenänderungsvektor $\underline{\Delta q}$ um einen Faktor f gekürzt, so daß gilt

$$\left| \underline{\mathbf{J}}_k \cdot \frac{\underline{\Delta q}}{f} \right| \leq \left| \underline{\Delta \ell}_k \right| \quad \forall \; k = 1..6 \quad .$$

$\underline{\mathbf{J}}_k$ bezeichnet hier die k-te Zeile der Jakobi-Matrix und $\underline{\Delta \ell}_k$ die k-te Komponente des karthesischen Lageabweichungsvektors. Der Kürzungfaktor f berechnet sich also nach

$$f = \mathrm{Max}\left(\frac{\left| \underline{\mathrm{Res}}_k \right|}{\left| \underline{\Delta \ell}_k \right|} \;,\; k = 1..6 \right) \quad .$$

So ist gewährleistet, daß sich der Greifpunkt in jeder Raumrichtung maximal um den Betrag der verbleibenden Lageabweichung bewegt, was das Verfahren insgesamt sehr stabilisiert.

Ist der so errechnete Faktor für eine oder mehrere Komponenten der Lageabweichung jedoch extrem groß, so wird $\underline{\Delta q}$ nicht mehr gekürzt, sondern limitiert auf eine sehr kleine Achsbewegung, die die Kinematik aus der extremen Singularitätsnähe oder Degeneration herausführt.

4.3 Begrenzung auf zulässige Achsparameter

Da jede Achse nur über einen begrenzten Bewegungsfreiraum von q_{min} bis q_{max} verfügt, ist es notwendig den Wert einer Achskoordinate auf eben diesen Bereich zu limitieren. Es muß für jeden neu errechneten Lösungsvektor $\underline{q}$ gelten

$$q_{\min,\,n} \le q_n \le q_{\max,\,n} \qquad , \forall n = 1..N \; .$$

N bezeichnet die Gesamtzahl der unabhängigen Achsen. Würde man nun nach jeder Iteration i lediglich den ‘überschüssigen’ Anteil jedes Achswertes abschneiden, so würde in der darauffolgenden Iteration die Achse vermutlich erneut über ihre Grenze verfahren, erneut ein großer Teil abgeschnitten und somit ebenfalls ein Anteil der Lageverbesserung dieser Iteration wegfallen.
Wesentlich effektiver ist es alle Achskoordinaten, welche in der aktuellen Iteration einen ihrer Anschlagswerte überschritten haben und auf den entsprechenden Wert begrenzt wurden, in der darauffolgenden Recheniteration als Festwerte anzusehen, die natürlich berücksichtigt werden, jedoch invariabel sind. Dadurch erhalten alle noch verbleibenden, sich also noch innerhalb ihrer Schranken bewegenden Achsen die Möglichkeit den entsprechenden Anteil der Lagedifferenz aufzubringen, der zuvor durch den abgeschittenen Anteil der angeschlagenen Achsen aufgebracht wurde.

4.4 Die Pseudoinverse

Ist die gekürzte Jakobi-Matrix eine quadratische Matrix, so läßt sie sich einfach invertieren. Ist die Matrix jedoch aufgrund der Anzahl der realen Áchsen oder aufgrund der Transformation in das gekürzte Modell entweder über- oder unterbestimmt, so muß eine quadratische Pseudoinverse gebildet werden, welche die Inverse ersetzt.

Bei unterbestimmten Gleichungssystemen hat die Jakobi-Matrix mehr Spalten als Zeilen. Ihr Rang ist also kleiner als die Anzahl der Achsen. Die betrachtete Kinematik besitzt in diesem Fall einen sogenannten Nullraum. Ein und dieselbe Greifpunktlage ist durch mehrere verschiedene Achsstellungen erreichbar und die Gesamtheit der Lösungen spannen diesen Raum auf. Jede

Bewegung innerhalb des Nullraumes hat folglich keinen Einfluß auf die Greifpunktlage. In diesen Fällen wird statt der (aufgrund ihrer nicht-quadratischen Form) nicht invertierbaren Jakobi-Matrix die sogenannte Rechtsinverse

$$\underline{\mathbf{J}}^{+} = \underline{\mathbf{G}}^{-1}\underline{\mathbf{J}}^{T}\left(\underline{\mathbf{J}}\ \underline{\mathbf{G}}^{-1}\underline{\mathbf{J}}^{T}\right)^{-1}$$

verwendet, welche das Gleichungssystem unter zusätzlicher Minimierung einer Nebenbedingung löst [5]. $\underline{\mathbf{G}}$ ist hierbei eine hauptdiagonalbesetzte Gewichtungsmatrix

$$\underline{\mathbf{G}} = \begin{bmatrix} g_1 & & \underline{0} \\ & \ddots & \\ \underline{0} & & g_n \end{bmatrix}$$

zur Gewichtung der N Achsgeschwindigkeiten in der zu minimierenden Fehlerfunktion

$$e = \sum_{n=1}^{N} g_n \dot{q}_n^2 \Rightarrow \text{Minimum}\,.$$

Liegt hingegen ein überbestimmtes Gleichungssystem vor, so gibt es in der aktuellen Achskonfiguration weniger unabhängige Gelenke als notwendige Freiheitsgrade. Die die inverse Jakobi-Matrix in diesem Fall ersetzende Linksinverse berechnet sich nach

$$\underline{\mathbf{J}}^{+} = \left(\underline{\mathbf{J}}^{T}\ \underline{\mathbf{G}}^{-1}\underline{\mathbf{J}}\right)^{-1}\ \underline{\mathbf{G}}^{-1}\underline{\mathbf{J}}^{T}\,.$$

In dieser Arbeit wird in beiden Fällen als Gewichtungsmatrix die Einheitsmatrix verwendet.

5 Minimierung der Lageabweichung bei unvollständiger Erreichbarkeit

5.1 Prinzipieller Lösungsweg

Die Grundidee hinter dem Verfahren ist es die zwischen Greifpunkt und Greifer verbleibende Raumabweichung zu quadrieren, mit einer Gewichtungsmatrix zu gewichten und die so erhaltene skalare Funktion

$$\left\| \underline{\Delta \ell} \right\|_{\underline{\mathbf{G}}}^{2} = \underline{\Delta \ell}(\underline{q})^{T} \cdot \underline{\mathbf{G}} \cdot \underline{\Delta \ell}(\underline{q}),$$

die eine positiv definite quadratische Form als Funktion der Achskoordinaten $\underline{q}$ darstellt, zu minimieren. Die nur in der hauptdiagonalen besetzte Gewichtungsmatrix habe die allgemeine Form

$$\underline{\mathbf{G}} = \begin{bmatrix} g_{11} & & \underline{0} \\ & \ddots & \\ \underline{0} & & g_{66} \end{bmatrix},$$

so daß also g_{11} die Gewichtung der quadrierten Abweichung in x-Richtung bezeichnet, g_{22} die der quadrierten Abweichung in y-Richtung und so weiter. In Kapitel 5.6 wird später genauer auf die Bedeutung dieser Matrix eingegangen. Es läßt sich auch schreiben

$$\left\| \underline{\Delta \ell} \right\|_{\underline{\mathbf{G}}}^{2} = \sum_{k=1}^{6} g_{kk} \cdot \Delta \ell(\underline{q})_{k}^{2} \quad .$$

Durch Einführung dieser Funktion ist das in seiner ursprünglichen Form unlösbare Problem ein mehrdimensionales Minimum als Funktion eines mehrdimensionalen Achskoordinatenvektors zu finden, reduziert worden auf mehrdimensionales Minimierungsproblems eines Skalars, nämlich dieser Fehlermaßfunktion. Dieses geschieht jedoch nur unter der Einschränkung der ausschließlichen Betrachtung der durch die Gewichtungsmatrix gefilterten Lösungen, was unter Umständen mehrere übergeordnete Iterationen erforderlich macht.

In einem ersten Schritt wird an dem aktuellen Punkt $\underline{q}_0$ eine Suchrichtung $\underline{s}(\underline{q}_0)$ ermittelt, also ein Richtungsvektor der Achsbewegungen in welcher das betrachtete Skalar (Fehlermaß) abnimmt. In dieser Richtung sucht man ein Minimum, was nur noch ein eindimensionales und leicht zu lösendes Optimierungsproblem darstellt. Das so gefundene Minimum bildet dann in der nächsten Iteration den Ausgangspunkt für eine weiterführende Suche und immer so fort, bis ein relatives Minimum des mehrdimensionalen Problems gefunden ist.

Da man sich ständig auf einem ‘Kurs’ durch die Achskoordinatenkonfiguration befindet auf dem der gefilterte Abstand zwischen Greifpunkt und Greifer nur kleiner werden kann, ist das Verfahren vollkommen stabil und konvergiert gegen ein lokales Minimum. Als Nachteil ist jedoch

zu sehen, daß mehr Rechenzeit als bei der Universaltransformation notwendig ist, da die Minimierung aufgrund ihrer Prazision in all ihren Schritten aufwendiger ist.

Die folgenden Abbildungen 13 und 14 verdeutlichen zunächst nochmals den Lösungsweg [3], bevor dann auf den genauen Ablauf der Einzelschritte eingegangen wird.

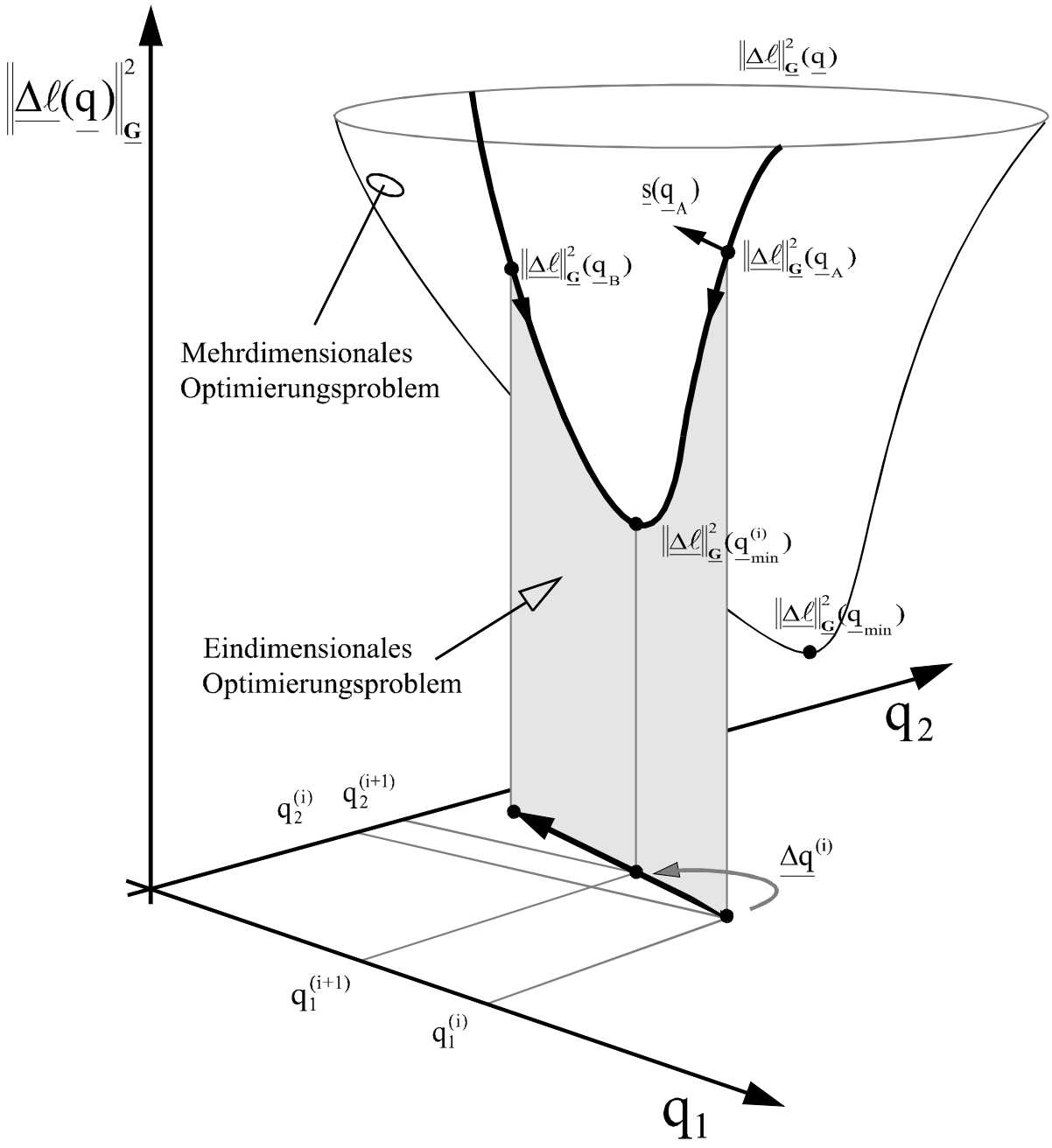

Abbildung 13 : Reduktion der Problemdimension und Lösung einer Iteration i

Das Problem ist stets auf eine 1- oder 2- achsige Kimematik bezogen dargestellt, da sich höherdimensionale Probleme wie etwa ein 6-achsiger greifbarer Roboterarm jeder Anschaulichkeit entziehen.

In der Konturansicht der durch Höhenlinien dargestellten Funktionswerte läßt sich dann ein ‘Kurs’ Richtung Minimum, den eine Lösung mit jeder neuen Iteration nimmt verfolgen. Der Weg hängt dabei jedesmal vom Suchrichtungsverfahren und vom durch die anfänglichen Achskoordinaten bestimmten Startpunkt ab. Es muß erreicht werden, daß einerseits mit möglichst wenigen Schritten, andererseits mit möglicht wenig Rechenaufwand pro Schritt ein Minimum gefunden wird. Diese sich widersprechenden Anforderungen gilt es gegeneinander abzuwägen, um die Effizienz des Verfahrens im Echtzeitbetrieb der Simulation zu optimieren.

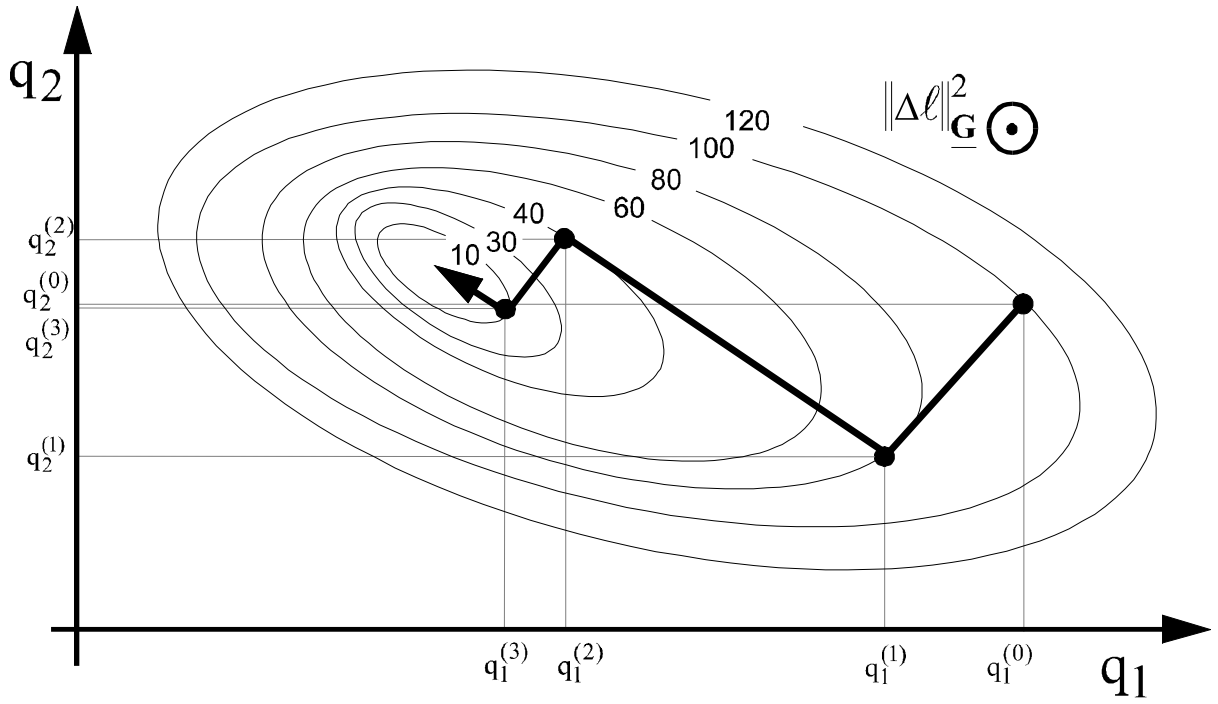

Abbildung 14 : Lösung des mehrdimensionalen Problems durch mehrfache Lösung des eindimensionalen (Hier unter Verwendung des Gradientenverfahrens).

5.2 Suchrichtungsbestimmung

5.2.1 Abstieg entlang der Achsrichtungen

Als einfachste Suchrichtungen bieten sich die in Richtungen der Achsen an. Hierbei wird jeweils nur der Einfluß einer einzelnen Achse untersucht und diese dann auf einen das betrachtete Fehlermaß minimierenden Wert verfahren.
Es ist jedoch für nur eine Iteration notwendig nacheinander in Richtung sämtlicher Achsen ‘abzusteigen’, was verdeutlicht, daß dieses Verfahren höchstens bei eindimensionalen Problemen wie etwa Schubladen, Klappen oder Schiebern geeignet ist. Der Rechenaufwand zur Bestimmung der Suchrichtung ist jedoch minimal, denn er erfordert nur eine einfache Zählschleife und unter Umständen eine Invertierung der Suchrichtungen, falls sie im (unten beschriebenen) anschließenden Verfahren der Linienoptimierung im Startpunkt einen Anstieg liefern.

5.2.2 Gradientenverfahren

Effizienter und eigentlich am naheliegensten ist das Gradientenverfahren, wobei sich die Suchrichtung aus dem negativen Gradienten der Fehlerfunktion ergibt. Man steigt also jeweils in Richtung des steilsten Abstieges ab. Es gilt also für die Suchrichtung

$$\underline{s}(\underline{q}^{(i)}) \;=\; -\,\underline{\text{grad}}\left\|\Delta\ell^{(i)}\right\|_{\underline{\mathbf{G}}}^{2}$$

Der Gradient berechnet sich folgendermaßen.

$$\underline{\text{grad}}\left\|\underline{\Delta\ell}\right\|_{\underline{\mathbf{G}}}^{2} \;=\; \underline{\text{grad}}\,(\,\underline{\Delta\ell}^{\mathrm{T}}\cdot\underline{\mathbf{G}}\cdot\underline{\Delta\ell}\,) = \begin{pmatrix} \dfrac{d\left\|\Delta\ell\right\|_{\underline{\mathbf{G}}}^{2}}{dq_1} \\ \vdots \\ \dfrac{d\left\|\Delta\ell\right\|_{\underline{\mathbf{G}}}^{2}}{dq_n} \end{pmatrix}$$

$$= \frac{d}{d\underline{q}}\Big[g_{11}(x_{soll}-x_{ist})^2+g_{22}(y_{soll}-y_{ist})^2+g_{33}(z_{soll}-z_{ist})^2+$$
$$+\,g_{33}(\varphi_{x,soll}-\varphi_{x,ist})^2+g_{55}(\varphi_{y,soll}-\varphi_{y,ist})^2+g_{66}(\varphi_{z,soll}-\varphi_{z,ist})^2\Big]$$

$$= g_{11}\frac{d}{d\underline{q}}\left(x(\underline{q})^2-2x(\underline{q})x_{soll}+x_{soll}{}^2\right)+\;\ldots$$

$$= g_{11}\left(2x(\underline{q})\cdot\frac{dx(\underline{q})}{d\underline{q}}\;-\;2x_{soll}\cdot\frac{dx(\underline{q})}{d\underline{q}}\right)+\;\ldots$$

$$= -\left[2g_{11}\left(\Delta x(\underline{q})\cdot\frac{dx(\underline{q})}{d\underline{q}}\right)+2g_{22}\left(\Delta y(\underline{q})\cdot\frac{dy(\underline{q})}{d\underline{q}}\right)+\;\ldots\right]$$

$$= -2\left[g_{11}\cdot\Delta x(\underline{q})\cdot\begin{pmatrix}\dfrac{dx}{dq_1}\\ \vdots \\ \dfrac{dx}{dq_n}\end{pmatrix}+g_{22}\cdot\Delta y(\underline{q})\cdot\begin{pmatrix}\dfrac{dy}{dq_1}\\ \vdots \\ \dfrac{dy}{dq_n}\end{pmatrix}+\;\ldots\right]$$

Man erhält somit eine Formel, deren Vektorkomponenten gerade den Zeilen der Jakobi-Matrix entsprechen und man kann die Schreibweise vereinfachen zu

$$\underline{\text{grad}} \left\| \underline{\Delta\ell} \right\|_{\underline{\mathbf{G}}}^{2} = -2\, \underline{\mathbf{J}}^{T} \cdot \underline{\mathbf{G}} \cdot \underline{\Delta\ell} \ .$$

Die z - te Zeile des Gradientenvektors berechnet sich folglich nach

$$\underline{\text{grad}} \left(\left\| \underline{\Delta\ell} \right\|_{\underline{\mathbf{G}}}^{2} \right)_{z} = -2 \sum_{k=1}^{6} \underline{\mathbf{J}}_{k,z} \cdot g_{kk} \cdot \underline{\Delta\ell}_{k} \ .$$

Für die gesuchte Suchrichtung ergibt sich

$$\underline{s}(\underline{q}^{(i)}) = \underline{\mathbf{J}}^{T} \cdot \underline{\mathbf{G}} \cdot \underline{\Delta\ell}^{(i)} \ .$$

Das Gradientenverfahren hat bei hier betrachteten Problemen jedoch einen gravierenden Nachteil. Falls nämlich die Fehlerfunktion nicht annähernd die Form einer sogenannten Hyperkugel hat, sondern die Höhenlinien etwa stark elliptische Form aufweisen, sind unter Umständen enorm viele Iterationen notwendig um das Minimum zu finden. Der ‘Suchkurs’ in den Achskonfigurationen nimmt dann sägezahnförmige Gestalt an, wobei die Zackenform vom Startpunkt abhängt. Leider überwiegen elliptische Formen in den betrachteten Funktionen und das Verfahren erweist sich bereits bei einer einfachen 2-achsigen Kinematik und einfacher reiner Gewichtung des Abstandes als vollkommen unbrauchbar (Es werden hierbei in Singularitätsnähe weit über 100 Iterationen benötigt, um eine nur einigermaßen akzeptable Lösung zu finden). Abbildung 15 verdeutlicht, daß aufgrund der stets rechtwinklig zueinander verlaufenden Suchrichtungen das Minimums erst sehr spät erreicht wird. Die dem Verfahren zugrundeliegende Rechtwinkligkeit der Suchrichtungen resultiert aus der Tatsache, daß der Gradient stets senkrecht auf den Höhenlinien der Fehlermaßfunktion steht.

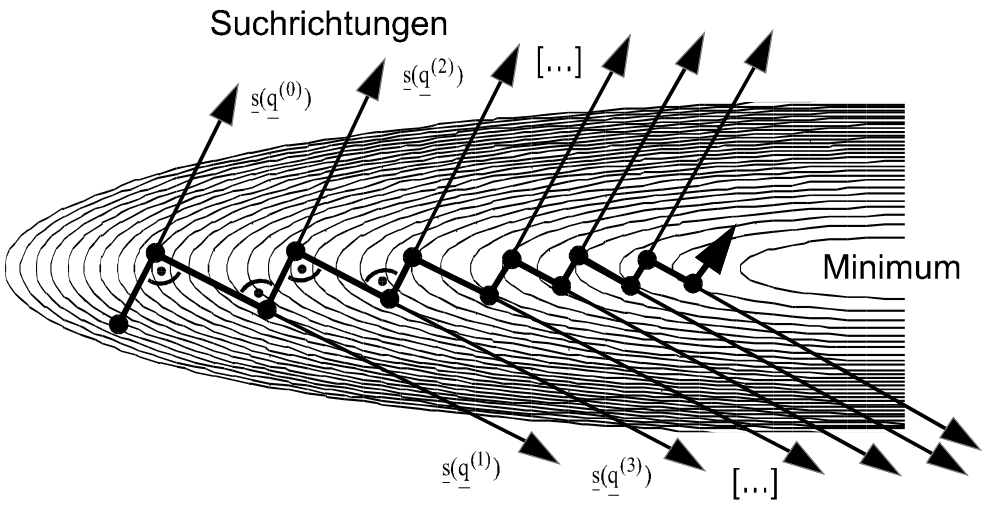

Abbildung 15 : Suchpfad des Gradientenverfahrens durch schlecht konditioniertes Problem

Dennoch wird die Richtung des Gradienten in den nachfolgenden Verfahren benötigt, welche diesen Nachteil glücklicherweise ausgleichen.

5.2.3 Newton Verfahren

Das Newton Verfahren erreicht mit einem Minimum an Schritten das relative Minimum, stellt aber leider auch die größten Anforderungen an die Berechnung der Suchrichtung

$$\underline{s}(\underline{q}^{(i)}) = -\underline{\underline{\mathbf{H}}}(\underline{q}^{(i)})^{-1} \cdot \underline{\mathrm{grad}} \left\| \underline{\Delta \ell}^{(i)} \right\|^2_{\underline{\underline{\mathbf{G}}}} \quad ,$$

denn es ist die Berechnung der Hesse-Matrix notwendig. Diese Matrix gibt hier die zweite Ableitung der Fehlermaßfunktion nach den Achskoordinaten wieder, ist also die Ableitung des Gradienten und sei mit $\underline{\underline{\mathbf{H}}}$ bezeichnet. Es gilt

$$\underline{\underline{\mathbf{H}}}(\underline{q}) = \frac{\mathrm{d}}{\mathrm{d}\underline{q}} \left(\underline{\mathrm{grad}} \left(\left\| \underline{\Delta \ell} \right\|^2_{\underline{\underline{\mathbf{G}}}} \right) \right) = \begin{bmatrix} \frac{\mathrm{d}^2 \left\| \Delta \ell \right\|^2_{\underline{\underline{\mathbf{G}}}}}{\mathrm{d}q_1{}^2} & \cdots & \frac{\mathrm{d}^2 \left\| \Delta \ell \right\|^2_{\underline{\underline{\mathbf{G}}}}}{\mathrm{d}q_1 \, \mathrm{d}q_n} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}^2 \left\| \Delta \ell \right\|^2_{\underline{\underline{\mathbf{G}}}}}{\mathrm{d}q_n \, \mathrm{d}q_1} & \cdots & \frac{\mathrm{d}^2 \left\| \Delta \ell \right\|^2_{\underline{\underline{\mathbf{G}}}}}{\mathrm{d}q_n{}^2} \end{bmatrix}$$

Die Elemente der Hesse-Matrix (Zeile z, Spalte s) berechnen sich mit den bisherigen Ergebnissen wie folgt. $\underline{\mathrm{grad}} \left(\left\| \underline{\Delta \ell} \right\|^2_{\underline{\underline{\mathbf{G}}}} \right)_z$ gibt dabei die z - te Zeile des Gradientenvektors an.

$$\underline{\underline{\mathbf{H}}}(\underline{q})_{z,s} = \frac{\mathrm{d}}{\mathrm{d}q_s} \cdot \underline{\mathrm{grad}} \left(\left\| \underline{\Delta \ell} \right\|^2_{\underline{\underline{\mathbf{G}}}} \right)_z$$

$$= \frac{\mathrm{d}}{\mathrm{d}q_s} \left(-2 \sum_{k=1}^{6} \underline{\mathbf{J}}_{k,z} \cdot g_{kk} \cdot \Delta \ell_k \right)$$

$$= -2 \sum_{k=1}^{6} g_{kk} \cdot \left(\left(\frac{\mathrm{d}\underline{\mathbf{J}}_{k,z}}{\mathrm{d}q_s} \cdot \Delta \ell_k \right) + \left(\underline{\mathbf{J}}_{k,z} \cdot \frac{\mathrm{d}\Delta \ell_k}{\mathrm{d}q_s} \right) \right)$$

$$= -2 \sum_{k=1}^{6} g_{kk} \cdot \left(\left(\frac{\mathrm{d}\underline{\mathbf{J}}_{k,z}}{\mathrm{d}q_s} \cdot \Delta \ell_k \right) + \left(\underline{\mathbf{J}}_{k,z} \cdot \left(\frac{\mathrm{d}\ell_{\mathrm{soll},k} - \mathrm{d}\ell_k(\underline{q})}{\mathrm{d}q_s} \right) \right) \right)$$

$$= 2 \sum_{k=1}^{6} g_{kk} \cdot \left(\underline{\mathbf{J}}_{k,s} \cdot \underline{\mathbf{J}}_{k,z} - \underline{\underline{\Delta \ell}}_{k} \cdot \frac{d\underline{\mathbf{J}}_{k,z}}{dq_s} \right)$$

Man sieht an der letzten Formel, das $\underline{\underline{\mathbf{H}}}(\underline{q})_{z,s} = \underline{\underline{\mathbf{H}}}(\underline{q})_{s,z}$, die Matrix also symetrisch ist.

Zur Berechnung sind jedoch die Ableitungen der Jakobi-Matrix nach allen Achsparametern notwendig. Es werden im folgenden die Ableitungen kompletter Spalten der Jakobi-Matrix $\frac{d}{dq_s}\underline{\mathbf{J}}_n$ hergeleitet, deren Komponenten dann in obiger Summe benötigt werden. Gesucht ist also der Einfluß jeder Achse s auf den Einfluß, den jede Achse n auf die Greifpunktlage hat.
Hierzu werden alle 10 möglichen Kombinationen folgender Fallunterscheidungen untersucht. (Ist s = n, so können die Achsen s und n nicht verschiedene Achstypen sein)

1) Achse n kann eine Rotations- oder eine Translationsachse sein.
1) Achse s kann eine Rotations- oder eine Translationsachse sein.
1) Achse s kann sich in der kinematischen Kette vor oder hinter Achse n befinden, oder identisch mit ihr sein.

Wie in den Grundlagen beschrieben berechnet sich die Jakobi-Matrix wie folgt.

$$\underline{J}_n = \begin{cases} \begin{bmatrix} \underline{z}_{n-1} \\ \underline{0} \end{bmatrix}, \text{ falls Achse n Translationsachse ist} \\ \\ \begin{bmatrix} \underline{z}_{n-1} \times \underline{r}_{n-1} \\ \underline{z}_{n-1} \end{bmatrix}, \text{ falls Achse n Rotationsachse ist} \end{cases} .$$

Dabei ist der Vektor $\underline{z}_{n-1}$ die in Normalform angebebenen Bewegungsrichtung der n-ten Gelenkachse und $\underline{r}_{n-1}$ definiert den Vektor vom n-1 ten Sektionskoordinatensystem bis zum Greifpunkt auf den sich die Jakobi-Matrix bezieht.
Ferner sei hier der Vektor $\underline{r}_{s-1,n-1}$ eingeführt, der vom Ursprung des Sektionskoordinaten-systems, in dem Achse s definiert ist, zum Ursprung des Koordinatensystems, in dem Achse n definiert ist, zeigt.

Zur Ableitungsberechnung der n ten Spalte der Jakobi-Matrix nach q_s ist jeweils der Einfluß von q_s auf $\underline{z}_{n-1}$ oder auf $\underline{r}_{n-1}$, beziehungsweise auf beide Vektoren von Interesse. Die so errechneten Wirkrichtungsvektoren lassen dann eine relativ einfache Berechnung des Ableitungsvektors zu.

$$\frac{d}{dq_s}\left(\underline{J}_n\right) = \begin{cases} \begin{bmatrix} \frac{d}{dq_s}\left(\underline{z}_{n-1}\right) \\ \underline{0} \end{bmatrix} & , \text{ falls Achse n Translationsachse ist} \\ \begin{bmatrix} \left(z_2' \cdot r_3 + z_2 \cdot r_3'\right) - \left(z_3' \cdot r_2 + z_3 \cdot r_2'\right) \\ \left(z_3' \cdot r_1 + z_3 \cdot r_1'\right) - \left(z_1' \cdot r_3 + z_1 \cdot r_3'\right) \\ \left(z_1' \cdot r_2 + z_1 \cdot r_2'\right) - \left(z_2' \cdot r_1 + z_2 \cdot r_1'\right) \\ \frac{d}{dq_s}\left(\underline{z}_{n-1}\right) \end{bmatrix} & , \text{ falls Achse n Rotationsachse ist} \end{cases}$$

Es bedeuten $\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \underline{z}_{n-1}$, $\begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} = \underline{r}_{n-1}$, $\begin{pmatrix} z_1' \\ z_2' \\ z_3' \end{pmatrix} = \frac{d}{dq_s}\left(\underline{z}_{n-1}\right)$ und $\begin{pmatrix} r_1' \\ r_2' \\ r_3' \end{pmatrix} = \frac{d}{dq_s}\left(\underline{r}_{n-1}\right)$.

Für die oben beschriebenen 10 Fälle laßt sich folgendes zusammenfassen:

Es gilt, falls sich Achse s in der kinematischen Kette vor Achse n befindet

$$\frac{d}{dq_s}\left(\underline{z}_{n-1}\right) = \begin{cases} \underline{0} & , \text{ falls s Translationsachse ist} \\ \underline{z}_{s-1} \times \underline{z}_{n-1} & , \text{ falls s Rotationsachse ist} \end{cases}$$

und

$$\frac{d}{dq_s}\left(\underline{r}_{n-1}\right) = \begin{cases} \underline{0} & , \text{ falls s Translationsachse ist} \\ \left(\underline{z}_{s-1} \times \underline{r}_{s-1}\right) - \left(\underline{z}_{s-1} \times \underline{r}_{s-1,n-1}\right) & , \text{ falls s Rotationsachse ist} \end{cases}.$$

Falls sich Achse s in der kinematischen Kette hinter Achse n befindet, oder Achse s mit Achse n identisch ist gilt

$$\frac{d}{dq_s}\left(\underline{z}_{n-1}\right) = \underline{0}$$

und

$$\frac{d}{dq_s}\left(\underline{r}_{n-1}\right) = \begin{cases} \underline{z}_{s-1} & , \text{ falls s Translationsachse ist} \\ \underline{z}_{s-1} \times \underline{r}_{s-1} & , \text{ falls s Rotationsachse ist} \end{cases}$$

5.2.4 Quasi-Newton Verfahren

Das auch Variable-Metrik Verfahren genannte Verfahren liefert eine Näherungslösung

$$\underline{s}(\underline{q}^{(i)}) = -\underline{\mathbf{N}}^{(i)} \cdot \underline{\text{grad}} \left\| \underline{\Delta \ell}^{(i)} \right\|_{\underline{\mathbf{G}}}^2$$

unter Umgehung der rechenzeitintensiven Bildung der inversen Hesse-Matrix mit jeder Iteration. Diese wird ersetzt durch

$$\underline{\mathbf{N}}^{(0)} = \underline{\mathbf{E}}, \qquad \underline{\mathbf{N}}^{(i)} = \left[\underline{\mathbf{N}} + \frac{\underline{\delta}\underline{\delta}^T}{\underline{\delta}^T \underline{y}} + \frac{\underline{\mathbf{N}}\underline{y}\underline{y}^T\underline{\mathbf{N}}}{\underline{y}^T\underline{\mathbf{N}}\underline{y}} \right]^{(i-1)} .$$

Eine Näherung mittels der sogenannten DFP-Formel, welche jeweils nur zwei Terme addiert. Die Vektoren

$$\underline{\delta}^{(i-1)} = \underline{q}^{(i)} - \underline{q}^{(i-1)}$$

und

$$\underline{y}^{(i-1)} = \underline{\mathrm{grad}} \left\| \underline{\Delta \ell} \right\|^{2\,(i)}_{\underline{\mathbf{G}}} - \underline{\mathrm{grad}} \left\| \underline{\Delta \ell} \right\|^{2\,(i-1)}_{\underline{\mathbf{G}}} ,$$

geben dabei die Abweichungen der Achsparameter und des Gradienten bezüglich der vorherigen Iteration an.

5.3 Linienoptimierung

Ziel der Linienoptimierung ist es, ein Minimum α_{min} des eindimensionalen Problemes

$$\left\| \underline{\Delta \ell} \right\|^{2}_{\underline{\mathbf{G}}} = f\left(\underline{q}_0 + \alpha \cdot \underline{s}(\underline{q}_0) \right)$$

zu finden, wobei α die hier einzige Variable der Abszisse ist.
Der Funktionswert entlang der sich ergebenden Kurve ist jeweils nach einer Vorwärtstransformation unter Zuhilfenahme der Gewichtungsmatrix leicht zu bestimmen. Die hier ebenfalls benötigte erste Richtungsableitung der Funktion in Suchrichtung $\underline{s}$ wird nach

$$f'\left(\underline{q}_0 + \alpha \cdot \underline{s}(\underline{q}_0) \right) = \underline{\mathrm{grad}} \left(\left\| \underline{\Delta \ell} \right\|^{2}_{\underline{\mathbf{G}}} \left(\underline{q} \right) \right) \cdot \frac{\underline{s}(\underline{q}_0)}{|\underline{s}\,(\underline{q}_0)|}$$

berechnet. Dieses ist notwendig, da lediglich die Gradientenkomponente interessiert, die parallel zur Suchrichtung liegt.

Man führt zunächst 2 Schranken a und b ein, die beide den Startwert 0 haben. Der Funktionswert $f(\underline{q}_a)$ ergibt sich dann für α=a, $f(\underline{q}_b)$ für α=b. Die Ableitungen der beiden Funktionswerte seien hier übersichtshalber mit $f'(\underline{q}_a)$ und $f'(\underline{q}_b)$ abgekürzt.
Die Richtungsableitung des Startwertes $\underline{q}_0$ ist in jedem Falle negativ, da die Suchrichtung ja eine Abstiegsrichtung sein muß. Also sind auch $f'(\underline{q}_a)$ und $f'(\underline{q}_b)$ vorerst negativ.

Dann setzt man b auf einen sehr kleinen Startwert und bildet $f'(\underline{q}_b)$. Ist diese Ableitung positiv so hat α=b das Minimum bereits überschritten und die Suche nach b bricht ab. Sollte $f'(\underline{q}_b)$ jedoch noch negativ sein, so liegt das Minimum noch weiter rechts und der Wert von b wird verdoppelt, während a auf den bisherigen Wert von b erhöht wird.

Es wird jedoch b jedesmal maximal soviel erhöht, daß die dadurch resultierenden Änderungen der Achsparameter festgesetzte rotorische und translatorische Maximalwerte nicht überschreiten.

Nach einigen Durchläufen liegt die Schranke a garantiert links des Minimums α_{min} und die Schranke b garantiert rechts davon, womit die Eingrenzunsphase beendet ist. Nun wird mit der quadratischen Interpolation

$$\tilde{\alpha}_{min} = \frac{b \cdot f'(a) - a \cdot f'(b)}{f'(a) - f'(b)}$$

ein Schätzwert für α_{min} errechnet und die Richtungsableitung an dieser Stelle gebildet. Ist sie negativ so liegt α_{min} noch weiter rechts und $\tilde{\alpha}_{min}$ wird zur neuen linken Schranke a. Andernfalls wird b dieser Wert zugewiesen.
Sollte der Betrag der Ableitung bei $\tilde{\alpha}_{min}$ jedoch einen Grenzwert unterschreiten ist das Minimum gefunden und die Linienoptimierung beendet.

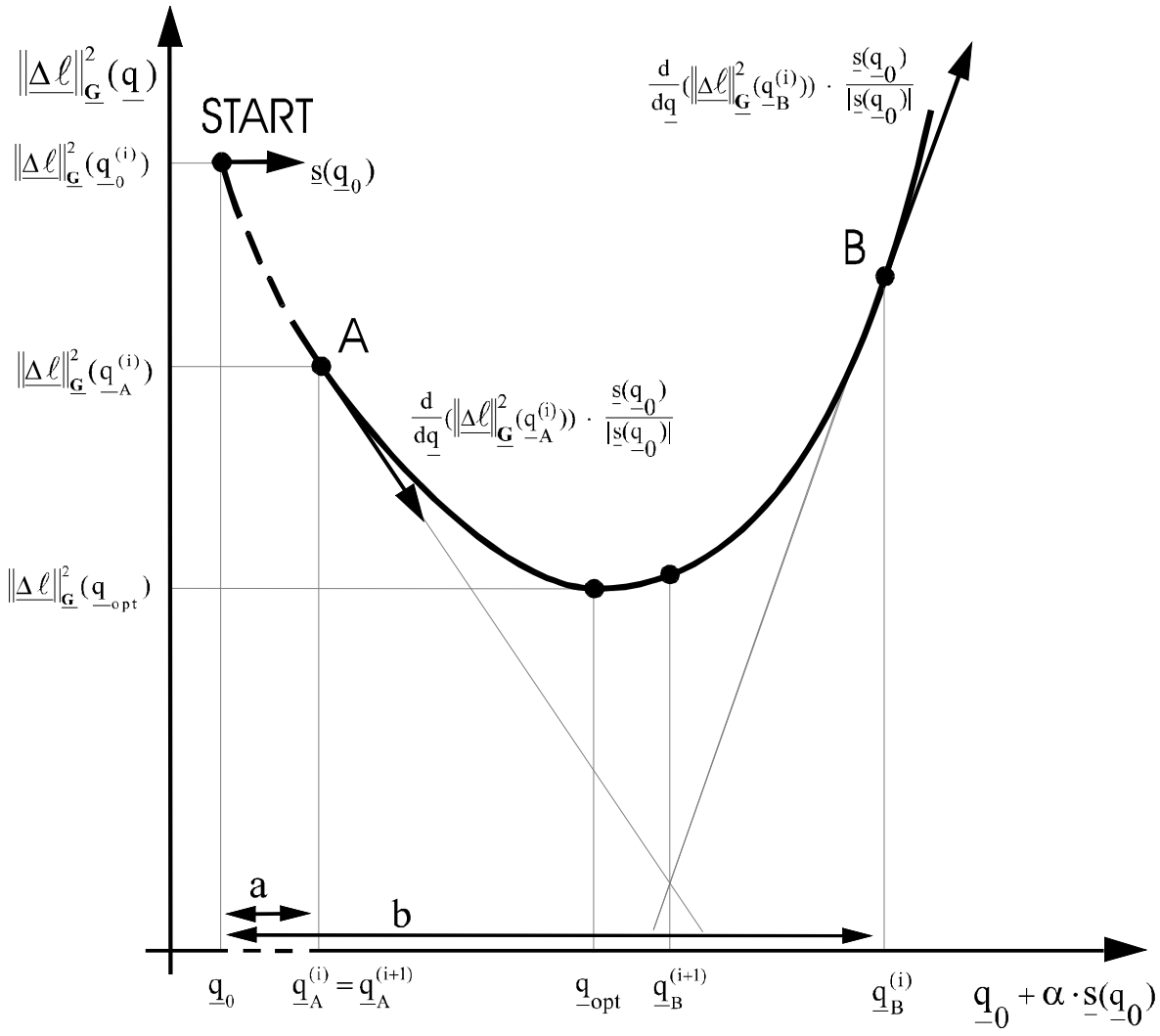

Abbildung 16 : Linienoptimierung entlang der Suchrichtung

Zu beachten ist noch, daß als Startwert für b nur bei der *ersten* Linienoptimierung einer Gesamtberechnung ein sehr kleiner Festwert verwendet wird. Anschließend erweist es sich als wesentlich günstiger einen Bruchteil des b-Endwertes der jeweils letzten Berechnung zu wählen, da so viele unnötige Vorwärtstransformationen in der Eingrenzungphase eingespart werden können, welche ja relativ rechenintensiv sind.

5.4 Abbruchbedingungen und Restart

Das Ziel, also die Achskonfiguration des nächstliegenden relativen Minimums der gewichteten Fehlerfunktion, gilt als gefunden, sobald der Gradient eines Punktes einen für den speziellen Greifpunkt festgesetzten Schrankenvektor unterschreitet. Abgebrochen wird eine Minimierung auch wenn Start- und Zielvektor sich nur noch minimal unterscheiden, also keine Suchbewegung mehr stattgefunden hat.

Ein sogenannter Restart wird durchgeführt, sobald eine Suchrichtung des Quasi-Newton Verfahrens am Startpunkt einer Iteration eine Anstiegsrichtung liefert. In diesem Fall verwendet man den negativen Gradienten als erste Suchrichtung in der neuen Iterationsfolge.
Rechnet man mit dem Quasi-Newton Verfahren, so wird prinzipiell nach einer festgesetzten Anzahl von Iterationen ein Restart durchgeführt und damit die Näherungsmatrix erneut initialisiert um Fehlerfortpflanzungen zu vermeiden.

5.5 Suchrichtungseinschränkungen zur Achskoordinatenbegrenzung

Um gegebenenfalls die Forderung nach Einhaltung von Achsparameterbeschränkungen zu erfüllen, wird eine Begrenzung der erlaubten Suchrichtungen notwendig.

Grundidee hierbei ist es, den Suchpfad immer innerhalb eines zulässigen Raumes zu belassen, dessen Dimension der Achsanzahl entspricht und dessen Grenzen durch $\underline{q}_{min}$ und $\underline{q}_{max}$ beschrieben wird. Dieser Raum ist ersichtlich grundsätzlich rechtwinklig, was die Lösung der hier beschriebenen 2n Ungleichungsnebenbedingungen

$$q_{min,\,n} \le q_n \le q_{max,\,n} \qquad \forall\, n = 1..\text{Anzahl der Achsen}\ .$$

erheblich vereinfacht.

Zunächst wird wie bislang zum jeweiligen Startpunkt eine Suchrichtung errechnet.
Zeigt diese für eine bereits angeschlagene Achse in die entsprechend verbotene Richtung, so wird die Komponente auf Null gesetzt, so daß bei der nachfolgenden Linienoptimierung in dieser Richtung keine Grenzverletztung auftreten kann. Ein Restart ist nicht erforderlich. Dieser ist ebenfalls nicht notwendig falls eine Achse, welche bislang angeschlagen war, sich mit einer neuen Suchrichtung vom Anschlag löst.
Anders verhält es sich, wenn eine bislang freie Achse in einen ihrer Anschläge verfährt. Hier ist in der nachfolgenden Iteration ein Restart erforderlich.

Es ist durchaus möglich, daß eine Achse auf ihrem Weg zum Minimum einer einzigen Fehlermaßfunktion anschlägt und sich wieder vom Anschlag löst. Ein Neuanschlag verlangsamt das Gesamtverfahren zwar eventuell ein wenig, da ja ein Restart erforderlich wird, andererseits verringert sich mit jeder dauerhaft angeschlagenen Achse die Dimension des Suchrichtungproblems um Eins, so daß ein Gesamtminimum im Durchschnitt schneller erreicht werden wird.

Als Beispiel sei hier eine Fehlerfunktion einer 2-achsige Kinematik symbolisch in der Konturansicht dargestellt.

Die zu Anfang gefundene Suchrichtung (hellgrau dargestellt) verfährt zunächst Achse q_2 in ihren Maximalwert. Die in Punkt 1 im Restart errechnete zweite Suchfunktion würde ungekürzt für Achse 2 in einen verbotenen Bereich führen und wird daher in ihrer q_2 - Komponente auf Null gesetzt. Die nun folgende Linienoptimierung sucht also entlang der Grenze und findet schließlich in Punkt 2 das Ziel, da eine neue Suchrichtung nach Kürzung Null ergibt und somit keine weitere Bewegung ausgeführt würde.

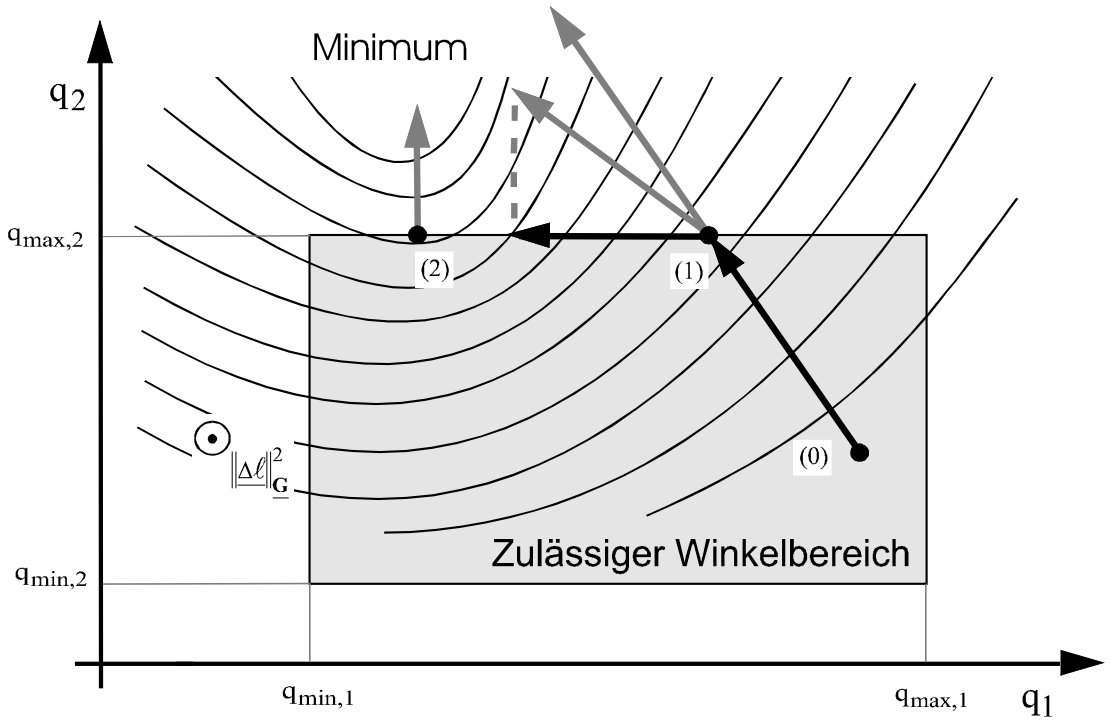

Abbildung 17 : Eingrenzung des Suchbereichs auf zulässige Achswerte

Bei jeder Linienoptimierung muß darauf geachtet werden, daß kein Wert außerhalb der Grenzen gefunden wird. Das heißt, sobald die Eingrenzungsphase eine rechte Schranke liefert, die für eine oder mehrere Achsen eine Nebenbedingung verletzt, wird die Schrankensuche beendet und der Punkt bestimmt, ab dem die Grenzverletzung auftrat.

Dieser ‘letzte gültige’ Punkt muß exakt auf der Grenze liegen. Anschließend wird wie üblich die Steigung ermittelt. Ist sie größer als Null, wird der Grenzpunkt als neue rechte Schranke definiert und die Linienoptimierung fortgesetzt. Ist die Richtungsableitung hingegen kleiner oder gleich Null, so ist der Grenzpunkt bereits das gesuchte Linienoptimum und die Linienoptimierung bricht ab.

Das Newton Verfahren erweist sich als ungünstig, sobald Suchrichtungskomponenten aus den hier beschriebenen Gründen zu Null werden. Wesentlich schneller führt dann das Quasi-Newton Verfahren zum Ziel.

5.6 Gewichtung der Lageabweichung

Die jeweilige hauptdiagonalbesetzte Gewichtungsmatrix **G** filtert die karthesische Lageabweichung jeder Achskonfiguration und erzeugt so die Fehlermaßfunktion, nach der eine Lösung gefunden werden kann. Das Problem besteht, wie oben erwähnt darin, sich widersprechende Güteanforderungen an eine 'optimale' Greifpunktlage möglichst gut zu erfüllen.
Die Matrix wird, falls eine Lagegüte eines meist niederdimensionalen Problemes nicht mit einer einzigen eindeutig beschrieben werden kann, *mehrmals* gebildet und *jedesmal* das lokale Minimum gesucht, in dessen Einzugsgebiet der Startpunkt liegt. Damit stellt die Bildung der Gewichtungsmatrix eine übergeordnete Schleife zum bisher betrachteten Minimierungsprozeß dar.
Das prinzipielle Vorgehen wird hier zunächst anhand einer einfachen 2-achsigen Kinematik verdeutlicht. Diese habe folgende 2-dimensionale Form.

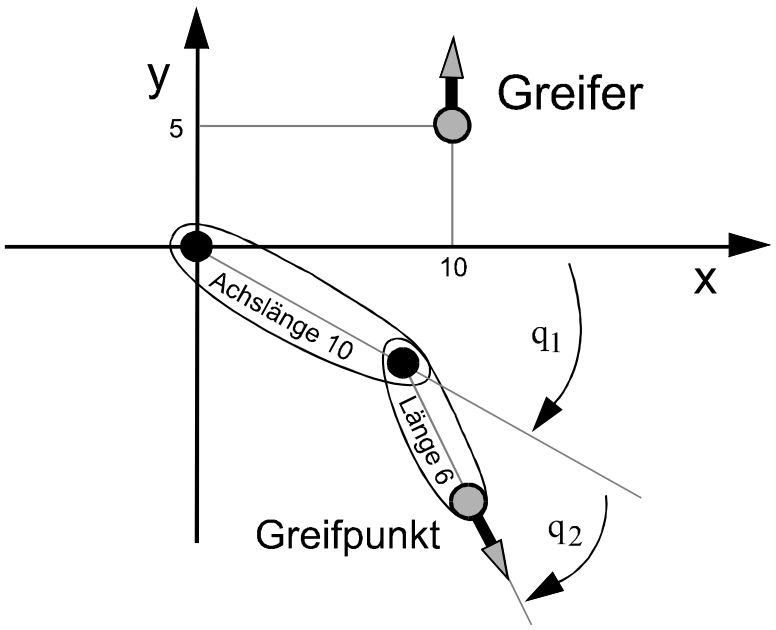

Abbildung 18 : Beispielkinematik

Die Pfeile zeigen dabei in Richtung der Orientierung von Greifer und Greifpunkt. Bei reiner Betrachtung des Abstandes zwischen beiden Punkten, also einer Gewichtungs-Matrix, in der lediglich $g_{11}=1$ und $g_{22}=1$ gesetzt und alle übrigen Komponenten Nullen sind, ergibt sich als Fehlerfunktion folgendes.

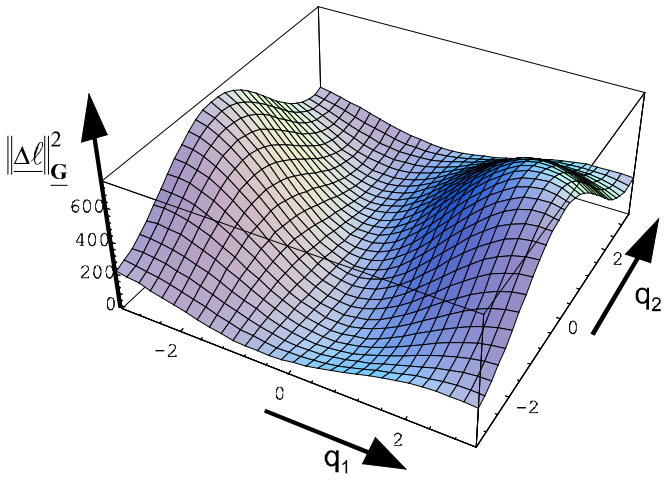

Abbildung 19 : Positionsabweichung der Beispielkinematik als Funktion der Gelenkparameter

In der Konturansicht, also der Darstellung von ‘Höhenlinien’ der Funktion erhält man nach einer Begrenzung des Wertebereichs auf den hier wichtigen Teil der Umgebung der Minima.

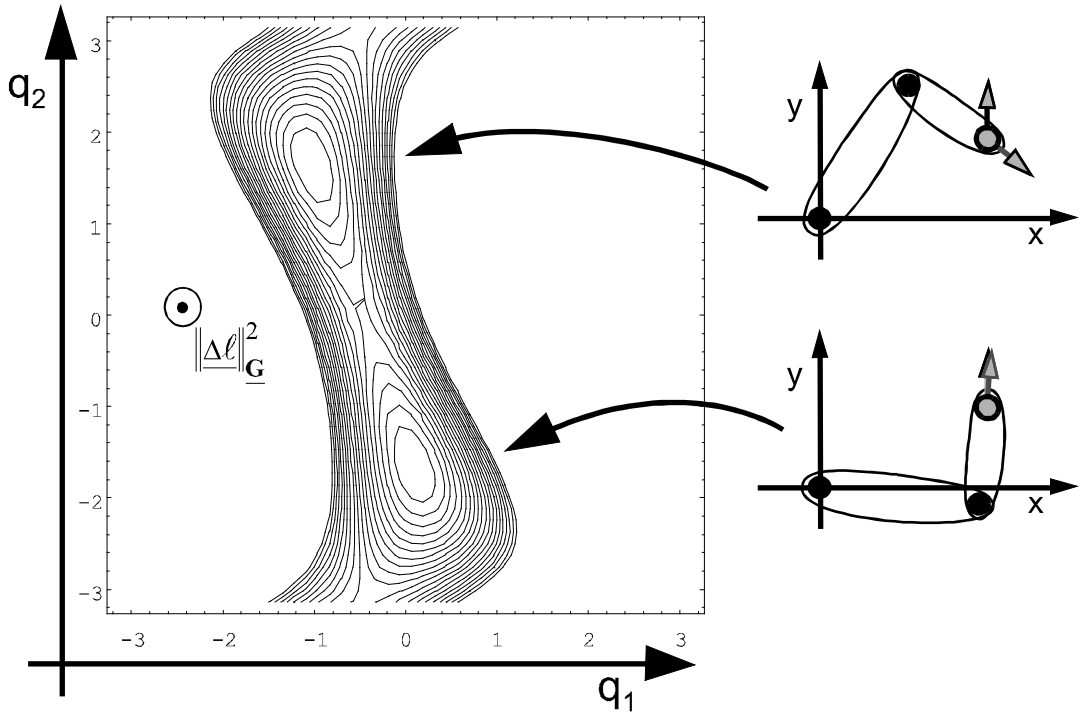

Abbildung 20 : Positionsabweichung in Konturansicht

Man sieht deutlich, daß hier zwei verschiedene Achsstellungen ein Minimum der Positionsabweichung bewirken. Mit der im Augenblick betrachteten Gewichtungsmatrix wird die Fehlerfunktion hier sogar exakt Null. Gut erkennbar ist auch die in etwa elliptische Form der Fehlermaßfunktion im Bereich der Minima.

Allgemein entstehen so Minima, welche durch Singularitäten voneinander getrennt werden. Im hier betrachteten Beispiel liegen Singularitäten bei $q_2 = 0$ und $q_2 = \pi$ vor und wie man sieht werden beide Minima von der horizontal in der Bildmitte liegenden Gerade getrennt.

Gewichtet man nun mit einer Matrix **G**, in welcher $g_{66} = 1$ und alle übrigen Komponenten Null sind, so wird nur die aus jede Gelenkstellung resultierende quadrierte Rotationsabweichung $\Delta\varphi_z$ zwischen Greifer und Greifpunkt als Fehlermaßfunktion gewählt. Es wird also ausschließlich der Orientierungsfehler betrachtet. φ_z ist der einzige rotorische Freiheitsgrad der betrachteten Kinematik und es ergibt sich in der 3D - und in der Konturansicht der Fehlerfunktion folgendes.

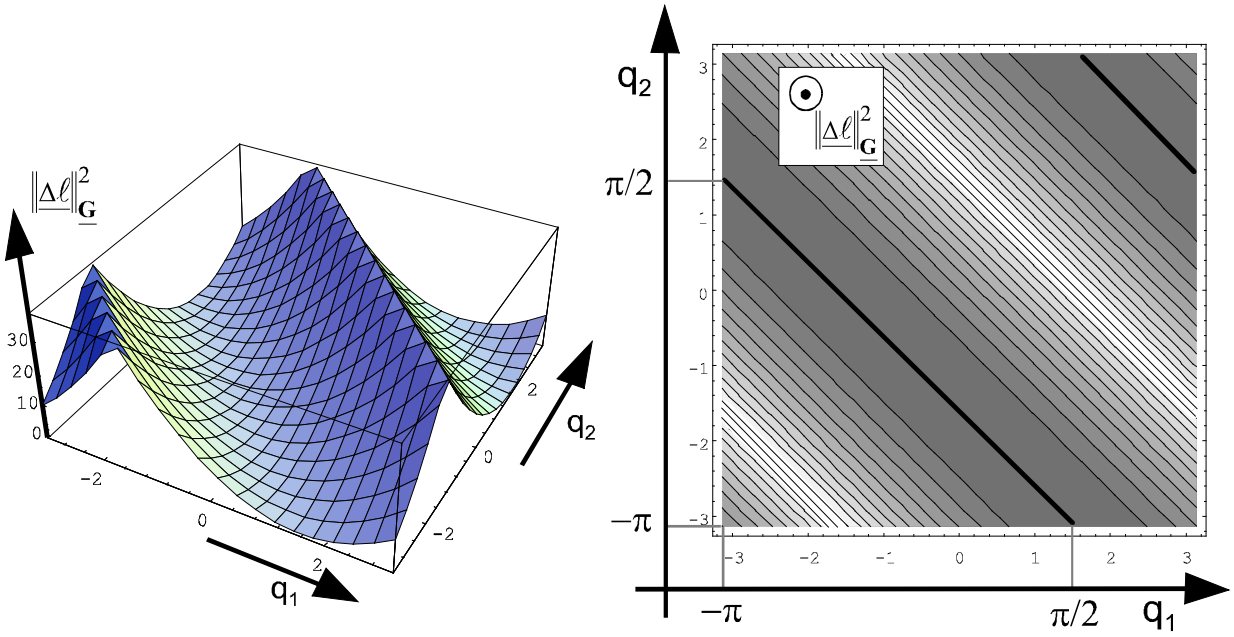

Abbildung 21 : Orientierungsabweichung der Beispielkinematik als Funktion der Gelenkparameter

Hier erfüllen also unendlich viele Achsstellungen die Bedingung einer minimalen rotorischen Differenz zwischen Greifer und Greifpunkt und alle liegen entlang einer Linie im Achswinkel-Koordinatensystem. Alle diese Punkte bilden den Nullraum der Kinematik bezüglich der durch diese Gewichtungsmatrix gefilterte Lösung. Würde man den Definitionsbereich obiger Funktion vergrößern, so wären die erhaltenen Werte in Richtung sämtlicher Drehachsen jeder Kinematik periodisch mit 2π.
Transformiert man die Lösung in das Koordinatensystem der karthesischen Lage, so ergibt sich ein Lösungskreis.

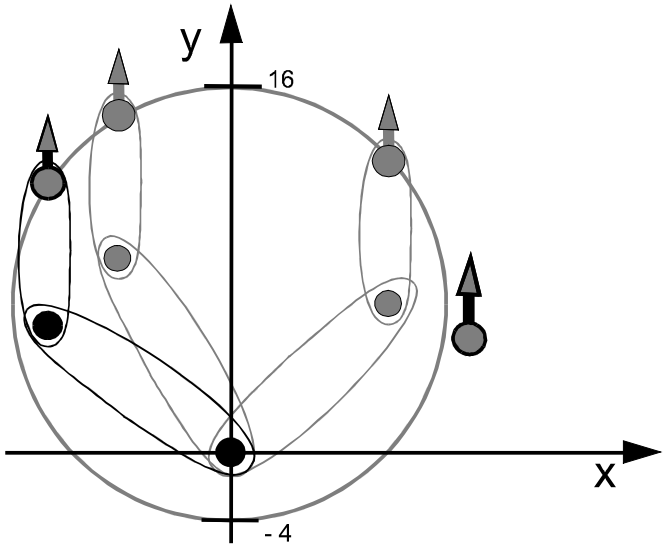

Abbildung 22 : Darstellung des Nullraumes der Beispielkinematik bei reiner Gewichtung der Orientierung

Eine Lösung mit reiner Gewichtung der Orientierung verfährt die Beispielkinematik also in eine nächstliegende Achskonfiguration, die mit obiger Graphik exemplarisch dargestellt ist. Die Position des Greifpunktes liegt danach immer auf einem Punkt des eingezeichneten Kreises.

Das Problem besteht in diesem Beipiel nun darin, diesen Lösungskreis mit den beiden möglichen Lösungspunkten der idealen Abstandsminimierung zu kombinieren und eine oder mehrere zum Gesamtziel führende Gewichtungsmatrizen zu erstellen.

Zunächst sei angemerkt, daß es hier keine Universallösung geben *kann*, da sich die Anforderung an eine greifbare Kinematik nicht einmal bei einfachen Anordnungen mit wenigen Freihheitsgraden *pauschal* angeben lassen. Eine grundsätzliche Bestimmungsmethode einer Gewichtungsmatrix, die eine Kinematik nach den jeweiligen Vorstellungen des Modellprogrammierers simuliert, läßt sich jedoch erkennen. Sie sei wieder an obigem Modell erläutert.

In einer ersten Näherung sei es hauptsächlich erwünscht, die Orientierungsabweichung zu minimieren und dabei noch, quasi als untergeordnete Nebenbedingung, den Abstand zwischen Greifpunkt und Greifer möglichst klein zu halten. Es ergibt sich hieraus eine Matrix etwa folgender Form,

$$\underline{\mathbf{G}} = \begin{bmatrix} 1 & & & & & & \underline{0} \\ & 1 & & & & & \\ & & 0 & & & & \\ & & & 0 & & & \\ & & & & 0 & & \\ \underline{0} & & & & & & 100 \end{bmatrix}$$

welche dann folgende Diagramme der Fehlerfunktion erzeugt. Das Konturdiagramm zeigt dabei wieder nur den interessierende Bereich in der Nähe des Minimums.

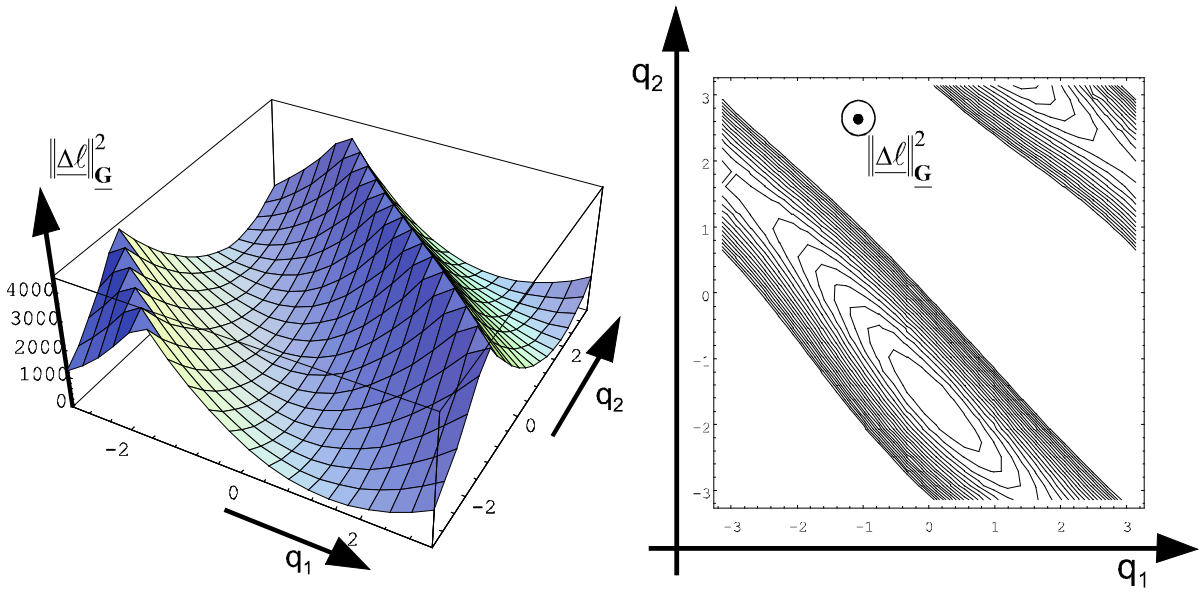

Abbildung 23 : Lageabweichungung der Beispielkinematik bei Gewichtung von Positions- und Orientierungsabweichung

Wie man sieht ergibt sich hier nur eine einzige Lösung. (Da beide Achsen Rotationsachsen sind ist die Lösung in Richtung beider Achsen periodisch mit 2π.) Diese liegt beinahe auf der Stelle des Lösungskreises, der der Stelle optimalen Abstands am nächsten liegt und hat somit beinahe ideale Orientierung. Die Tatsache, daß der Lösungspunkt nicht exakt auf dem Kreis liegt folgt daraus, daß ja die weit niedriger gewichtete Abstandsabweichung mit berücksichtigt wurde und sich der Greifpunkt folglich ein kleines Stück in Richtung Greifer bewegt hat.

Ein Vergleich mit Abbildung 20 zeigt, daß so eindeutig eine Lösung im Einzugsgebiet des günstigeren lokalen Minimums ermittelt wurde.
Der so erreichte Punkt kann nun beispielsweise den Ausgangspunkt für eine zweite Näherung bilden, in der wie zuvor beschrieben nur der Abstand gewichtet wird.

Allgemein läßt sich sagen, daß bei gewünschter Minimierung der Translationsabweichung zwischen Greifer und Greifpunkt die Rotationsabweichung anfänglich stark und zunehmend schwächer gewichtet werden muß, bis sie in der letzten Minimierung nicht mehr mit ins Fehlermaß eingehen sollte.
Im umgekehrten Fall muß bei gewünschter Minimierung der Rotationsabweichung die Translationsabweichung anfänglich stark und zunehmend schwächer gewichtet werden, bis sie im letzten Schritt ignoriert oder vernachlässigt wird.

Grundsätzlich ist bei mehreren übergeordneten Iterationen (Gewichtungsmatrizen, Abbruchschranken) nach Möglichkeit darauf zu achten, daß die jeweils erhaltene Lösung eindeutig im Einzugsgebiet des günstigsten lokalen Minimums der darauffolgenden Fehlermaßfunktion liegt.

5.7 Nullraumnutzung

Für den Fall, daß zu der *letzten* betrachteten Gewichtungsmatrix möglicherweise ein Nullraum existiert, wird zur letzten Fehlermaßfunktion zusätzlich ein Term addiert, welcher einen solchen dazu nutzt, die Achsen möglichst nicht in Anschlagsnähe zu verfahren. Das Vorgehen ähnelt dem der Bildung einer Rechtsinversen bei der Universaltransformation.

Die allein durch die Gewichtungsmatrix bestimmte Fehlermaßfunktion liefert im Falle eines vorhandenen Nullraumes für verschiedene Achskonfigurationen den gleichen (minimalen) Wert. Durch Addition einer Funktion

$$f_{NR} = \sum_{n=1}^{N} g_n = \sum_{n=1}^{N} \left(\frac{q_n - \frac{q_{max,\,n} + q_{min,\,n}}{2}}{\frac{q_{max,\,n} - q_{min,\,n}}{2}} \right)^2 = \sum_{n=1}^{N} \frac{\left(2\,q_n - \left(q_{max,\,n} + q_{min,\,n}\right)\right)^2}{\left(q_{max,\,n} - q_{min,\,n}\right)^2}$$

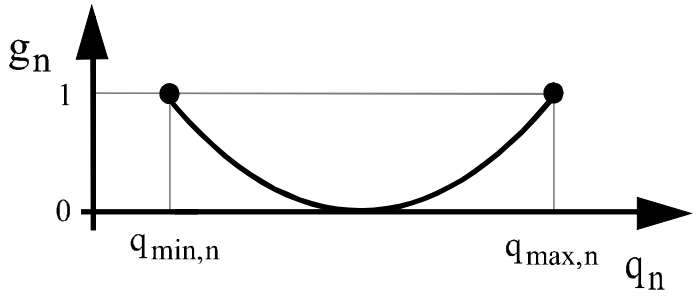

Abbildung 24 : Gewichtungsfaktor als Funktion der Achskoordinate

wird in diesem Lösungsraum dann die optimale Konfiguration gefunden, da eine (unerwünschte) Nähe der Achsen zu ihren Anschlagswerten stärker gewichtet und so möglichst vermieden wird .

Zu beachten ist jedoch, daß die bisherige Fehlermaßfunktion Priorität haben muß. Ihr Wertebereich muß also um mehrere Zehnerpotenzen größer sein. Der Wertebetreich obiger Funktion ist ersichtlich [0,N], da jeder Summand bei Anschlag der Achse maximal 1 werden kann.

Der im Minimierungsverfahren benötigte Gradientenvektor und die Hessematrix der zusätzlichen Fehlerfunktion lassen sich hier einfach berechnen, da die Anschlagswerte der Achsen invariabel und die Achswerte selbst voneinander unabhängig sind.
Es gilt für den Gradienten

$$\underline{\mathrm{grad}}\left(f_{NR}\right) = \frac{d}{d\underline{q}} f_{NR} = \frac{d}{d\underline{q}} \left(\sum_{n=1}^{N} \frac{4q_n^2 - 4q_n \cdot \left(q_{max,\,n} + q_{min,\,n}\right) + \left(q_{max,\,n} - q_{min,\,n}\right)^2}{\left(q_{max,\,n} - q_{min,\,n}\right)^2} \right)$$

und somit für seine z-te Zeile

$$\underline{\text{grad}}\left(f_{NR}\right)_z = \frac{8\,q_z - 4\left(q_{max,\,z} + q_{min,\,z}\right)}{\left(q_{max,\,z} - q_{min,\,z}\right)^2} \,.$$

Für die Komponenten der Hessematrix (Zeile z, Spalte s) gilt

$$\underline{\mathbf{H}}\left(f_{NR}\right)_{z,s} = \frac{d}{dq_s} \cdot \underline{\text{grad}}\left(f_{NR}\right)_z = \begin{cases} \dfrac{8}{\left(q_{max,\,z} - q_{max,\,z}\right)^2} & \text{, falls } z = s \\ 0 & \text{, sonst} \end{cases} \,.$$

6 Simulation allgemeiner Steward-Plattformen

6.1 Struktur und universelle Modellierung

Eine allgemeine Steward-Plattform besteht aus einer raumfesten Basis und einer theoretisch frei beweglichen Plattform, welche durch 6 Schubachsen (Beine) miteinander verbunden sind. Die Verbindungspunkte bestehen aus perfekten Kugelgelenken. Eine solche Plattform kann jede denkbare Anordnung der Gelenke auf den Ebenen haben, die Gesamtkonstruktion muß jedoch stabil sein und die kinematischen Zwangsbedingungen (jede Achse muß anfangs die Länge der zwischen ihren beiden Befestigungspunkten liegenden Strecke haben und darf keine andere Achse berühren) müssen eingehalten werden.

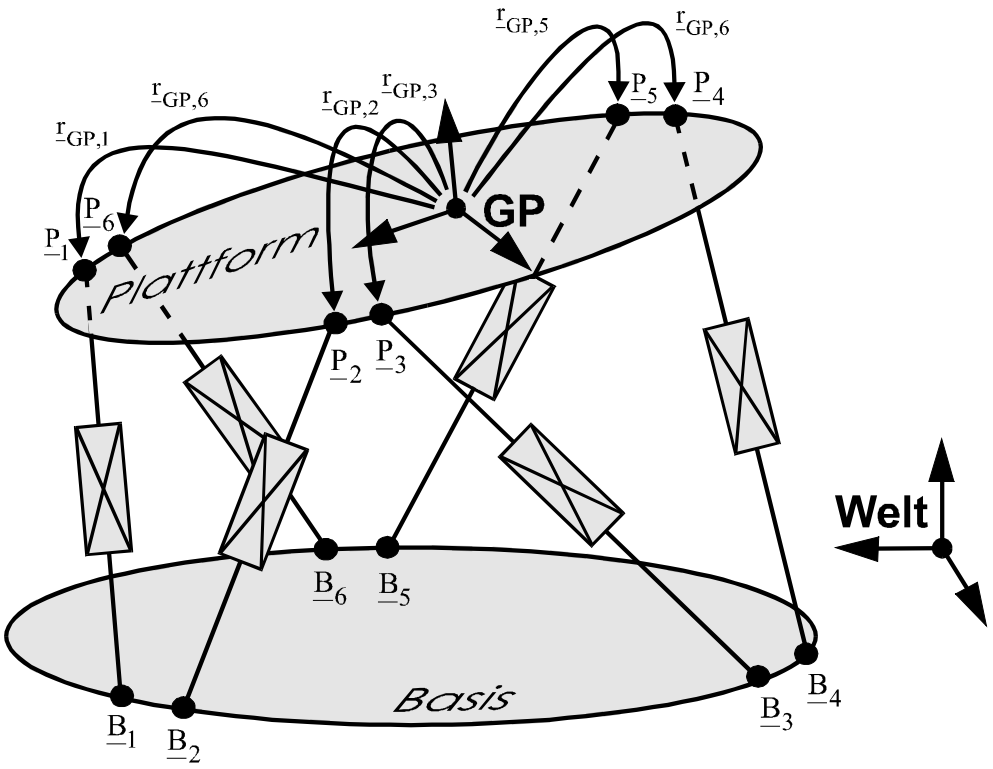

Abbildung 25 : Modell einer Steward-Plattform

Modelliert wird die Anordnung zunächst durch ein (redundantes) raumfestes Basis-Objekt sowie ein frei bewegliches Plattform-Objekt, an welchem der Greifpunkt definiert wird.
Alle Bein-Objekte bestehen zunächst aus einer raumfesten Basis B und zwei Drehgelenken ohne räumliche Ausdehnung zur Simulation eines unteren Kugelgelenkes. Diese Gelenke besitzen senkrecht zueinander stehende Achsvektoren.
Da sich die Achsparameter von Plattformbeinen leicht *direkt* berechnen lassen wird keinerlei Minimierung durchgeführt. Um diese direkte Berechnung möglichst einfach (=schnell) zu halten wird vereinbart, daß die Orientierung der Basissektion der des Weltkoordinatensystems entsprechen muß und die erste Drehachse des unteren Kugelgelenkes die z-Achse, die zweite die so entstandene x-Achse ist. Beide Gelenke haben keinen Offset und positives Vorzeichen.

Dann folgt das Schubgelenk, welches in der Realität aus einer einzigen Druckkammer besteht, die durch hydraulische Kräfte ein oder mehrere Segmente abgestuften Durchmessers teleskopartig ausfahren läßt. Eine derartige Konstruktion läßt sich durch eine Kette von Translationsgelenken (in z-Richtung, Offset und Vorzeichen frei wählbar) simulieren. Das dabei entstehende Problem der Gesamt- Maximalbeschleunigung und -geschwindigkeit der Anordung läßt sich durch eine übergeordnete Definition lösen. Es wird also jeweils explizit angegeben mit welchen (Translations-) Maximalwerten sich das *gesamte* Bein bewegen kann.
Das Schubgelenk endet im Drehpunkt eines oberen Kugelgelenkes $\underline{P}$. Dieses obere Gelenk wird jedoch *nicht* mehr durch Drehgelenke simuliert. Bezüglich der letzte Schubgelenk-Sektion wird dann jeweils ein Greifpunkt definiert, welcher im Drehpunkt des oberen Kugelgelenkes liegen muß und dessen Orientierung unwichtig ist.

Mit dem hier vorgestellten Verfahren können nicht nur Steward-Plattformen simuliert werden, denn es ist prinzipiell völlig unerheblich, wo im Raum die Endpunkte der Achsen liegen, solange die Basen raumfest sind und die Drehpunkte der oberen Kugelgelenke plattformfest. Auch die Anzahl der Achsen ist hier beliebig und strukturelle Stabilität ist im Grunde nicht erforderlich. Bei Simulation einer instabilen Plattform wird sich dieses Verhalten jedoch nur bedingt zeigen, da eine Gravitationswirkung hier nur durch eine entsprechende Lage des Greifers simuliert werden kann. Allgemeine Plattformen können also auch (mit diesen Einschränkungen) auf Stabilität getestet werden. Instabile Kinematiken zeigen gegebenenfalls Kollisionen einzelner Sektionen, strukturell stabile Plattformen werden sich auch in der Simulation immer stabil verhalten.

6.2 Berechnung bei bekannter Plattformlage

Die Position der Punkte $\underline{P}_1$ bis $\underline{P}_6$ bezeichnen die Drehpunkte der oberen Kugelgelenke und sind invariant bezüglich der Plattform. Die Position dieser Punkte relativ zum Greifpunkt-Koordinatensystem der Plattform wird zu Beginn einer Simulation vektoriell gespeichert ($\underline{r}_{GP,1}$ bis $\underline{r}_{GP,6}$). Die so jeweils von der Plattformlage abhängigen Positionen der plattformfesten Kugelgelenke $\underline{P}$ stellen jeweils die Soll-Positionen für die Greifpunkte der Bein-Objekte dar. Die Orientierung der Greifpunkte jedes Beines spielt aufgrund der simulierten Kugelgelenke keinerlei Rolle.

Der Drehwinkel α um die z-Achse und der Winkel β und der x-Achse lassen sich dann auf direktem Wege berechnen. Mit.

$$\underline{d} = \underline{P} - \underline{B}$$

gilt

$$\alpha = \begin{cases} \arctan\left(-\frac{d_x}{d_y}\right) & ,\text{falls } d_x \geq 0 \text{ und } d_y < 0 \\ \frac{\pi}{2} & ,\text{falls } d_x \geq 0 \text{ und } d_y = 0 \\ \pi - \arctan\left(-\frac{d_x}{d_y}\right) & ,\text{falls } d_x \geq 0 \text{ und } d_y > 0 \\ -\arctan\left(-\frac{d_x}{d_y}\right) & ,\text{falls } d_x < 0 \text{ und } d_y < 0 \\ -\frac{\pi}{2} & ,\text{falls } d_x < 0 \text{ und } d_y = 0 \\ -\pi + \arctan\left(-\frac{d_x}{d_y}\right) & ,\text{falls } d_x < 0 \text{ und } d_y > 0 \end{cases} = \begin{cases} 0 & ,\text{ falls } d_x = d_y = 0 \\ \text{atan2}\left(d_x, -d_y\right) & ,\text{ sonst} \end{cases}$$

und

$$\beta = \begin{cases} 0 & ,\text{ falls } d_x = 0 \text{ und } d_y = 0 \\ \frac{\pi}{2} - \arctan\left(\frac{d_z}{\sqrt{d_x^2 + d_y^2}}\right) & ,\text{ sonst} \end{cases} .$$

α hat damit einen Wertebereich $]-\pi, \pi]$ und β einen Wertebereich $[0, \pi/2]$.

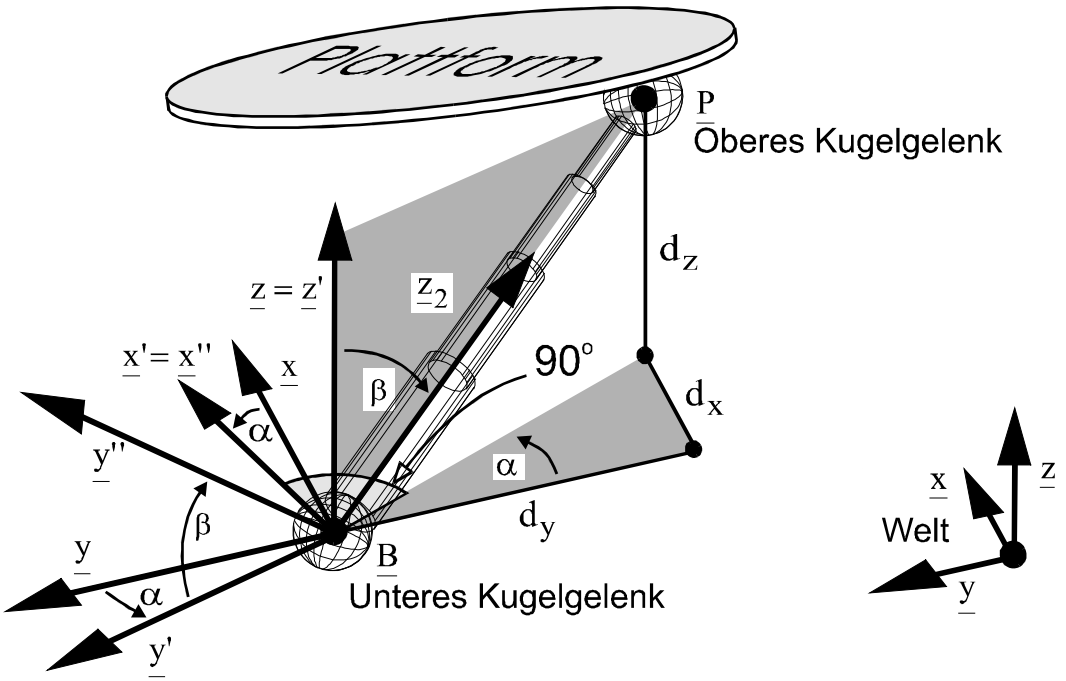

Abbildung 26 : Berechnung der beiden Drehwinkel im unteren Kugelgelenk eines Plattform-Beines

Die Achsparameter der einzelnen Schubsegment-Sektionen ergeben sich durch

$$q = q_{min} + \frac{L_{soll} - L_{min}}{L_{max} - L_{min}} \cdot (q_{max} - q_{min}),$$

wobei L_{max} die Maximallänge und L_{soll} die aktuelle Solllänge des Beines bezeichnet.

Bein-Greifpunkte werden nur dann neu berechnet, wenn die zugehörige Plattform gegriffen ist und somit aktuelle Soll-Positionen der oberen Kugelgelenke vorliegen. Sie lassen sich *nicht direkt* von einem Greifer greifen (dann keine Berechnung), da wohl niemand ein Kugelgelenk aus seiner Halterung herausbewegen soll (und will...).

Das Plattform-Objekt selbst ist wie erwähnt theoretisch frei beweglich. Ist seine im aktuellen Simulationsschritt erreichbare Lage bekannt, so kann eine Verschiebungsmatrix (move_frame) direkt exakt definiert werden. Dennoch ist der auf der Plattform definierte Greifpunkt eingeschränkt (restricted_grippoint), da sich die Plattform ja nur in Raumlagen befinden kann, die keinerlei kinematische Zwangsbedingungen verletzten.

6.3 Inverse Kinematik bei vollständiger Erreichbarkeit

Die einzige Einschränkung in der Berechnung einer das Gesamtproblem lösenden Gelenkkonfiguration ist die Notwendigjkeit der Einhaltung von Intervallen, in denen die Gesamtlängen der Beine (Formelzeichen L) im aktuellen Simulationsschritt s liegen dürfen. Ein solches Intervall von optimaler Größe läßt sich mit der in Kapitel 3.5 beschriebenen Methode errechnen, indem für jedes Schubgelenk jeweils ein Sollwert von $L_{b,\,min}$ und von $L_{b,\,max}$ vorgegeben wird. Die begrenzten Lösungen bilden dann die Grenzen des gesuchten Wertebereich-Intervalls. Es gilt somit für die Schubachse des b-ten Beines

$$L_{b\,min}^{(s)} = L_b^{(s)} \left(\tilde{L}_b^{(s)} = L_{b,\,min} \right)$$

und

$$L_{b\,max}^{(s)} = L_b^{(s)} \left(\tilde{L}_{b,\,n}^{(s)} = L_{b,\,max} \right)$$

Eine Soll-Lösung für die Länge jedes Beines ist bei bekannter Lage des Plattform-Greifpunktes relativ einfach berechenbar. Mit dem Vektor

$$\begin{aligned} \underline{v}_b^{(s)} &= \underline{P}_b - \underline{B}_b \\ &= \underline{Pos}\left(\underline{\underline{T}}_{Welt,GP}^{(s)} \right) + \underline{Rot}\left(\underline{\underline{T}}_{Welt,GP}^{(s)} \right)^{-1} \cdot \underline{Pos}\left(\underline{r}_{GP,b} \right) - \underline{Pos}\left(\underline{\underline{T}}_{Welt,\,Basis,\,b}^{(s)} \right) \end{aligned}$$

ist

$$\tilde{L}_b^{(s)} = \left| v_b^{(s)} \right| .$$

Vorraussetzung ist jedoch, daß die Soll-Position für die Greifpunkte aller Bein-Kinematiken erreichbar ist, das heißt falls gilt

$$L_{b\,min}^{(s)} \leq \tilde{L}_b^{(s)} \leq L_{b\,max}^{(s)} \qquad \forall\, b = 1..6 \;,$$

In diesem Fall ist eine Lösung gefunden und es lassen sich schnell entsprechende Denavit-Hartenberg-Parameter für alle zu berechnenden Sektionen finden. Im Anschluß ist unter Umständen (falls erforderlich) noch eine Berechnung aller Bein-Geschwindigkeiten durch erneute Begrenzung auf Maximalbescheunigung und Maximalgeschwindigkeit notwendig.

6.4 Abweichungsminimierung bei unvollständiger Erreichbarkeit

Ist mindestens eine Solllänge von der Kinematik nicht aufzubringen, so kommt ein Näherungsverfahren zum Einsatz, das die zwangsweise in Kauf zu nehmende Abweichung minimiert, ähnlich wie dies bereits in Kapitel 5 beschrieben wurde.

Die kinematischen Zwangsbedingungen einer Steward-Plattform lassen sich so veranschaulichen, daß um jeden Basispunkt $\underline{B}_b$ zwei Kugel liegen, von denen eine einen Radius $L_{b\,min}^{(s)}$, die andere einen Radius $L_{b\,max}^{(s)}$ hat. Die plattformfesten Punkte $\underline{P}_b$ der jeweiligen Beine müssen nun stets in dem Raum liegen, den die beiden Kugeln (im betrachteten Halbraum) einschliessen. Ein dabei entstehender Nullraum kann genutzt werden um wie bisher Nebenbedingungen der Plattform-Lage (Greifpunkt-Lage) bezüglich der Greifer-Lage zu minimieren.

Zunächst wird ein Fehlermaß bezüglich der Längen aller Beine eingeführt. Bezeichnet i wieder die Iterationen der Minimierung, so gilt mit

$$d_b^{(s,i)} = \begin{cases} L_b^{(s,i)} - L_{b\,min}^{(s)} & , \text{ falls } L_b^{(s,i)} < L_{b\,min}^{(s)} \\ L_b^{(s,i)} - L_{b\,max}^{(s)} & , \text{ falls } L_b^{(s,i)} > L_{b\,max}^{(s)} \\ 0 & , \text{ sonst} \end{cases}$$

für das Fehlermaß

$$F_L = \sum_{b=1}^{N_b} d_b^{(s,i)\,2}$$

N_b bezeichnet hier die Gesamtzahl der Beine. In jedem Simulationsschritt ist nun (jedesmal ausgehend vom anfänglichen, *nicht* vom Nullvektor) ein Lageabweichungs-Vektor $\underline{\ell}_\Delta^{(s,i)}$ zwischen Greifpunkt und Greifer zu berechnen, der F_L zu Null werden läßt und damit die kinematischen Zwangsbedingungen erfüllt. Da zusätzlich im Nullraum dieser Funktion nach der Lageabweichung gesucht wird, die das bekannte Fehlermaß

$$F_\Delta = \sum_{k=1}^{6} g_k \cdot \underline{\ell}_{\Delta}^{(s,i)2} = \left\| \underline{\ell}_{\Delta}^{(s,i)} \right\|_{\underline{\underline{G}}}^2$$

minimiert, (über den Gewichtungsvektor $\underline{g}$ bzw die entsprechend hauptdiagonalbesetzte Gewichtungsmatrix $\underline{\underline{\mathbf{G}}}$ lassen hier wieder Raumkoordinaten der in Kauf zu nehmenden Abweichung frei gewichten) ergibt sich als Gesamt-Fehlermaß einer ersten Minimierung

$$F_{ges} = F_\Delta + F_L$$

Eine so gefundene Plattformlage bildet dann den Ausgangspunkt für eine zweiten Minimierung, in welcher die Funktion F_Δ entfällt, so daß eine Lösung gefunden wird, welche die kinematischen Zwangsbedingungen *exakt* erfüllt.

Diese beiden Minimierungen unterscheiden sich von bisherigen dadurch, daß *jeweils* nur eine *einzige* Iteration (Suchrichtungsbestimmung und Linienoptimierung) durchgeführt wird. Beide werden jedoch in einer übergeordneten Schleife so oft durchlaufen, bis der Unterschied der Lageabweichung bezüglich der vorhergehenden Iteration einen Schrankenvektor (Bewegungslimit) unterschreitet.

$$\underline{\ell}_{\Delta k}^{(s,i)} - \underline{\ell}_{\Delta k}^{(s,i-1)} \leq \underline{\ell}_{\Delta \text{ limit } k} \quad \forall\, k = 1..6$$

Auf diese Weise befindet sich ein ‘Suchkurs’ stets in unmittelbarer Nähe einer kinematisch möglichen Lösung. Die Gewichtungsfaktoren sind klein genug zu wählen, daß dieser Kurs nicht zu weit in kinematisch verbotene Bereiche führt (Instabilitäten). Andererseits bewirken höhere Faktoren eine bessere Lösung. Die Wahl hängt also nicht nur von den relativen Gewichtungen der Lagekoordinatenabweichungen ab, sondern auch vom Aufbau und der Größe der Plattform und muß jeweils individuell angepaßt werden.

Eine hierdurch gefundene optimale Greifpunktlage

$$\underline{\underline{\mathbf{T}}}_{\text{Welt,GP}}^{(s)} = \underline{\underline{\mathbf{T}}}_{\text{Welt,G}}^{(s)} \cdot \underline{\underline{\mathbf{T}}}\left(\underline{\ell}_{\Delta}^{(s)}\right)^{-1}$$

ist dann Ausgangspunkt zur schnellen und einfachen Berechnung sämtlicher Sektionen der Plattform.

Bei der notwendigen Umrechnung der karthesischen Lageabweichung $\underline{\ell}_\Delta$ in eine homogene Tansformationsmatrix $\underline{\underline{\mathbf{T}}}$ sind die ersten drei Komponenten identisch mit dem Translationsvektor der Matrix, so daß also nur noch die drei Winkelkomponenten von $\underline{\ell}_\Delta$ in eine Orientierungsmatrix umgerechnet werden müssen [3].

Zuerst werden die drei Winkelkomponenten in einen normierten Drehvektor $\underline{v}$ und einen zugehörigen Drehwinkel θ umgewandelt. Anschließend wird ein zu $\underline{v}$ senkrecht stehender Vektor $\underline{b}$ bestimmt. Die gesuchte Rotationsmatrix ist dann

$$\underline{\mathbf{Rot}}\left(\underline{\mathbf{T}}\left(\underline{\ell}_{\Delta}\right)\right) = \left[\underline{v},\ \underline{b},\ \underline{v}\times\underline{b}\right]\cdot\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\Theta & -\sin\Theta\\ 0 & \sin\Theta & \cos\Theta\end{bmatrix}\cdot\left[\underline{v},\ \underline{b},\ \underline{v}\times\underline{b}\right]^{T}$$

und die Gesamt-Matrix somit

$$\underline{\mathbf{T}}\left(\underline{\ell}_{\Delta}\right) = \left[\begin{array}{ccc|c} & \underline{\mathbf{Rot}}\left(\underline{\mathbf{T}}\left(\underline{\ell}_{\Delta}\right)\right) & & \begin{matrix}\ell_{\Delta x}\\ \ell_{\Delta y}\\ \ell_{\Delta z}\end{matrix}\\ \hline 0 & 0 & 0 & 1\end{array}\right].$$

Die Mimierung ist dann nicht allzu rechenintensiv, da die Greifpunkt-Greifer Abweichung direkt betrachtet wird und somit keine Vorwärtstransformationen notwendig sind, welche ja bisher jeweils zur Berechnung der Abweichungen aus den Achskoordinaten-Vektoren notwendig waren. Der Gradient von F_{Δ} ist leicht sehr zu bestimmen. Es gilt für seine z - te Zeile

$$\underline{\mathrm{grad}}\left(F_{\Delta}\right)_{z} = \frac{d}{d\ell_{\Delta z}}\left(\left\|\underline{\ell}_{\Delta}^{(s,i)}\right\|_{\underline{\mathbf{G}}}^{2}\right) = \sum_{k=1}^{6}\frac{d}{d\ell_{\Delta z}}\left(g_{k}\cdot\ell_{\Delta k}^{(s,i)2}\right) = 2g_{z}\cdot\underline{\ell}_{\Delta z}^{(s,i)}.$$

Die Berechnung des Gradienten von F_L erfordert jedoch eine Jakobi-Matrix **J**'. Zunächst sei hierfür der gesamte kinematische Zusammenhang bezüglich eines Beines b graphisch dargestellt.

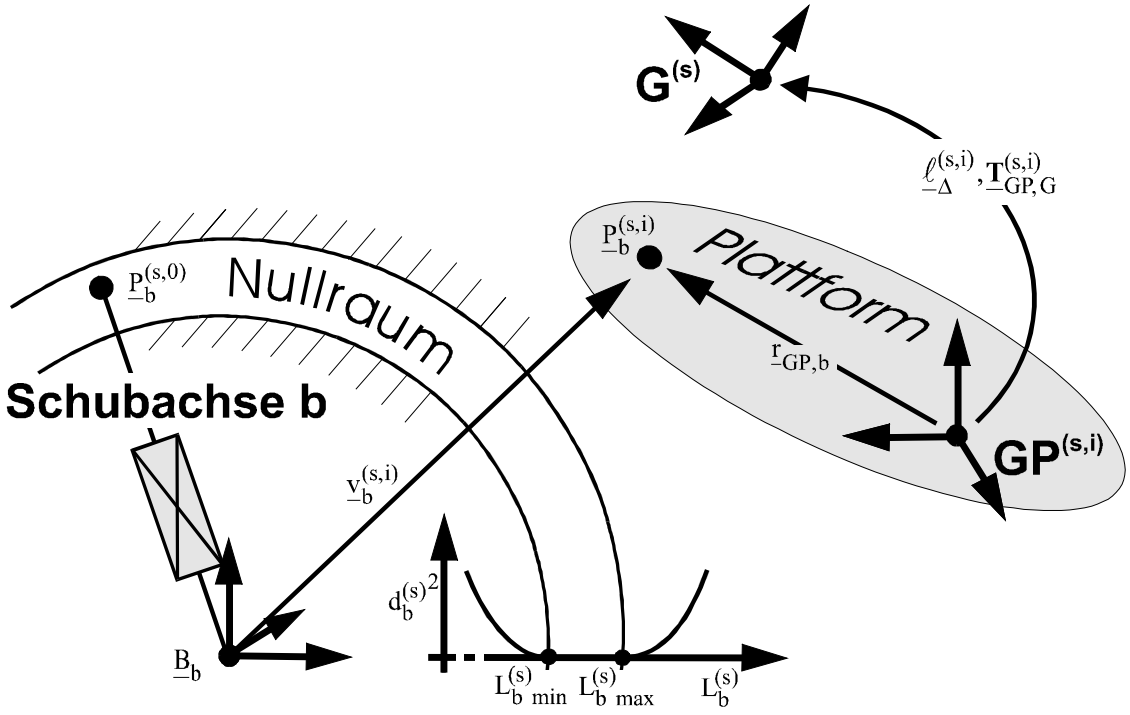

Abbildung 27 : Einfluß von $\underline{\ell}_{\Delta}^{(s,i)}$ auf die Länge einer Schubachse b

Es läßt sich eine Jakobi-Matrix berechnen, die jeweils den Zusammenhang zwischen der zeitlichen Änderung der Greifpunkt-Lage und der zeitlichen Änderung der Längen der Schubachsen einer Plattform angibt.

$$\underline{\dot{L}}^{(s,i)} = \underline{\mathbf{J}}'^{(s,i)^T} \cdot \underline{\dot{\ell}}_{\Delta}^{(s,i)} \quad .$$

Um Verwechslungen mit der bisherigen Jakobi-Matrix zu vermeiden, wird sie mit einem Strich indiziert. Da das Bezugssystem für $\underline{L}$ keine Rolle spielt (Der Abstand zwischen den Basispunkten $\underline{B}$ und den plattformfesten Punkten $\underline{P}$ ist in allen Koordinatensystemen gleich), ist das Gesamtbezugssystem der Matrix das Welt-Koordinatensystem. Stellt man die Formel um und kürzt durch das Zeitinkrement, so ergibt sich

$$\underline{\Delta L}^{(s,i)} = \underline{\mathbf{J}}'^{(s,i)^T} \cdot \underline{\Delta \ell}_{\Delta}^{(s,i)} \quad .$$

$\underline{\mathbf{J}}'^{(s,i)}_{k,b}$ bezeichnet die k-te Zeile und b-te Spalte der Jakobi-Matrix und gibt die Wirkungsrichtung der k-ten Komponente von $\underline{\ell}_{\Delta}^{(s,i)}$ auf die Länge des b-ten Beines an.

$$\underline{\mathbf{J}}' = \begin{bmatrix} \frac{dL_1}{d\ell_{\Delta 1}} & \cdots & \frac{dL_{Nb}}{d\ell_{\Delta 1}} \\ \vdots & \ddots & \vdots \\ \frac{dL_1}{d\ell_{\Delta 6}} & \cdots & \frac{dL_{Nb}}{d\ell_{\Delta 6}} \end{bmatrix}$$

Mit

$$\underline{\mathbf{T}}^{(s,i)}_{Welt,GP} = \underline{\mathbf{T}}^{(s)}_{Welt,G} \cdot \underline{\mathbf{T}}\left(\underline{\ell}_{\Delta}^{(s,i)}\right)^{-1}$$

gilt

$$\underline{\mathbf{J}}'^{(s,i)}_{k,b} = \begin{cases} -\frac{\underline{v}_b^{(s,i)^T}}{\left|\underline{v}_b^{(s,i)}\right|} \cdot \underline{\mathrm{Rot}}_k\left(\underline{\mathbf{T}}^{(s,i)}_{Welt,GP}\right) & , \text{falls } k \le 2 \\ -\frac{\underline{v}_b^{(s,i)^T}}{\left|\underline{v}_b^{(s,i)}\right|} \cdot \left[\underline{\mathrm{Rot}}_{(k-3)}\left(\underline{\mathbf{T}}^{(s,i)}_{Welt,GP}\right) \times \left[\underline{\mathrm{Pos}}\left(\underline{\mathbf{T}}^{(s,i)}_{Welt,GP}\right) - \underline{\mathrm{Pos}}\left(\underline{P}_b^{(s,i)}\right)\right]\right] & , \text{sonst} \end{cases}$$

$\underline{\mathrm{Rot}}_k\left(\underline{\mathbf{T}}\right)$ gibt hier, wie im Kapitel ‘Grundlagen’ vereinbart, die k-te Hauptachse des Koordinatensystems an, welches aus dem Weltkoordinatensystem durch Multiplikation mit einer homogenen Transformationsmatrix $\underline{\mathbf{T}}$ entsteht.

Ersichtlich interessieren allein die Anteile der Bewegungen in Richtung der jeweiligen Vektoren $\underline{v}_b^{(s,i)}$. Das negative Vorzeichen resultiert aus der Richtungsdefinition von $\underline{\ell}_{\Delta}^{(s,i)}$ und der Tatsache, daß die Greiferlage im jeweiligen Simulationsschritt unverändert bleibt.

Es gilt nun für die k-te Komponente des 6 zeiligen Gradientenvektors von F_L

$$\operatorname{grad}\left(F_L\right)_k = \frac{d}{d\ell_{\Delta k}^{(s,i)}}\left(\sum_{b=1}^{N_b} d_b^{(s,i)2}\right)$$

$$= \sum_{b=1}^{N_b}\left(\frac{d}{d\ell_{\Delta k}^{(s,i)}} d_b^{(s,i)} \cdot 2 d_b^{(s,i)}\right) = 2\sum_{b=1}^{N_b}\left(d_b^{(s,i)} \cdot \underline{J}'^{(s,i)}_{k,b}\right)$$

Da in besagtem Nullraum eine Gradientenkomponente zu Null wird, läßt sich auch etwas anschaulicher schreiben

$$\operatorname{grad}\left(F_L\right)_k = 2\sum_{b=1}^{N_b}\begin{cases} d_b^{(s,i)} \cdot \underline{J}'^{(s,i)}_{k,b} & \text{, falls } L_b^{(s,i)} < L_{b\,min}^{(s)} \text{ oder } L_b^{(s,i)} > L_{b\,max}^{(s)} \\ 0 & \text{, sonst} \end{cases}$$

Dieser Gradientenvektor wird also im Laufe eines Simulationsschrittes *immer* zum Nullvektor, da die zweite Minimierung (mit $F_\Delta = 0$) jeweils genau dann abbricht.

Grenzen in der Bewegungsfreiheit einer so berechneten Plattform ergeben sich ausschließlich durch die Beschaffenheit (Maximale und minimale Länge, Maximalwerte für Beschleunigung und Geschwindigkeit) der Schubgelenke und nicht etwa durch Anschläge der Gelenke.

7 Simulation geschlossener kinematischer Ketten

Eine geschlossene kinematische Kette muß im Gegensatz zu einer offenen stets die zusätzliche Bedingung erfüllen, daß *beide* Endpunkte raumfest bleiben. Unter dieser Nebenbedingung soll sich dann wieder ein an einer beliebigen Sektion definierter Greifpunkt dem Greifer möglichst ideal annähern.

7.1 Modellierung und grundsätzliche Vorgehensweise

Die Modellierung einer solchen geschlossene Kette unterscheidet sich vom Modell einer offenen im Grunde allein durch Definition (Wahl eines speziellen Objektnamens),
Hierdurch lassen sich aus der Gesamtstruktur dann programmintern zwei Ketten unterscheiden, welche die Grundlage zur Berechnung sämtlicher Achsparameter bilden. Die geschlossene Kette führt unverändert von einer Basis zur anderen, während eine 'Greifpunkt-Kette' von der Basis bis zu der Sektion mit dem aktuellen Greifpunkt führt.

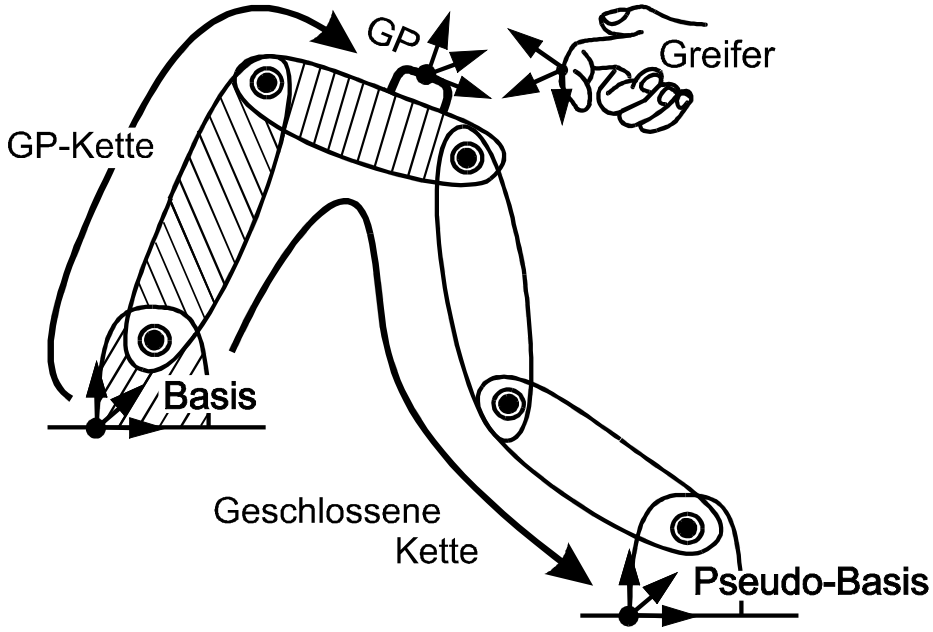

Abbildung 28 : Modell einer geschlossenen kinematischen Kette

Es ergeben sich so folgende Berechnungs-Anforderungen:

1.) Die 'Pseudo-Basis' *muß* nach einer gesamten Berechnung ihre ursprüngliche Lage einnehmen. Dieses ist *immer* möglich und ersichtlich von höchster Priorität.
2.) Die GP-Kette muß unter der Bedingung 1 den Greifpunkt möglichst optimal dem Greifer annähern. Sollte die geschlossene Kette einen Nullraum haben (und das ist bei Definition eines Greifpunktes eigentlich zu erwarten ... ansonsten bewegt sich

nichts), so gibt es hier auch grundsätzlich eine Optimierungsmöglichkeit.
3.) Ein eventueller anschließend noch *verbleibender* Nullraum kann wie bisher genutzt werden.

Alle 3 Fehlermaßfunktionen (Beträge entsprechen wieder Prioritäten) überlagern (addieren) sich einfach, da alle den gleichen Raum nutzen, sprich die gleichen Achsen berechnet werden. Lediglich die Fehlerfunktion zu Punkt 2 nutzt nicht die Achsparameter zwischen der Greifpunkt-Sektion und der Pseudo-Basis.
Anforderung 1 wird erfüllt, indem die Ausgangslage der Pseudo-Basis einfach wie ein Greifer behandelt wird, der einen (ebenfalls nicht existierenden) Greifpunkt gegriffen hält, dessen Lage mit der Ist-Lage der Pseudo-Basis identisch ist. Fuer einen solchen Pseudo-Greifvorgang lassen sich ebenfalls Gewichtungs-Matrizen und Schranken vorgeben, welche dann parallel zur eigentlichen Mimierung des Greifpunkt-Greifer Abstandes bearbeitet werden. Die Berechnung entspricht also im wesentlichen der einer offenen Kette.
Zu beachten ist jedoch, daß in der *letzen* Minimierungsiteration sehr exakt minimiert wird, das heißt die Abbruchschranke für den Pseudo-Greifvorgang sehr klein ist und der reale Greifpunkt-Greifer Abstand mit Null gewichtet wird.

7.2 Koordinierte Mehrrobotersysteme

Greifen 2 Roboter ein Werkstück, so läßt sich im Augenblick des Zugriffs eine Transformationmatrix erstellen, welche eine Greiferlage in die andere überführt. Modelliert man diese Matrix als eine entsprechende (unsichtbare) Sektion, so entsteht wieder eine geschlossene kinematische Kette von einer Roboterbasis über das Werkstück zur anderen.
Durch entsprechend gleiche Vorgehensweise lassen sich so stets Achskonfigurationen finden, welche das Werkstück nicht belasten, aber dennoch greifbar machen.
Das gilt natürlich besonders für den nicht-trivialen Fall, daß beide Roboter unterschiedliche Struktur haben, unterschiedliche Achskonfigurationen beim Zugriff oder beispielsweise maximal erreichbare Geschwindigkeiten oder Beschleunigungen der Achsen vorliegen.
Ein Beispiel sind etwa 2 Industrieroboter, welche eine Windschutzscheibe mit 2 Sauggreifern greifen und ohne jede Relativbewegung, die das Glas platzen lassen würde, in eine Karosserie setzten müssen.

8 Programmbeschreibung

Im folgenden werden kurz die wichtigsten Bestandteile der erstellten Erweiterungs-DLL zum bestehenden, am IRF Dortmund entstandenen Simulationsprogramm COSIMIR erläutert.
Nähere Einzelheiten befinden sich als Kommentarzeilen direkt im Quelltext. Hierdurch sollte eine Einarbeitung im Bezug auf Erweiterungen oder Änderungen in relativ kurzer Zeit möglich werden. Die Kenntniss der grundsätzlichen COSIMIR-Datenstrukur wird dabei vorrausgesetzt.

8.1 Spezifisches Umweltmodell

Es wird zunächst mit Initialisierung jeder neuen Arbeitszelle eine Listenstruktur erzeugt, welche alle zur Berechnung notwendigen Parameter enthalten. Die Gründe hierfür liegen in einer schnelleren Zugriffsmöglichkeit und einer wesentlich erhöhten Übersichtlichkeit. Außerdem lassen sich so einige ansonsten mit jedem Simulationsschritt durchzuführenden Berechnungen in die Initialisierungsphase verlegen, was die Rechengeschwindigkeit der Simulation selbst (ein wenig) steigert.

Das Sub-Umweltmodell besteht aus einer Zeiger-Liste sämtlicher Greifpunkte des aktuellen Modells, wobei jedes Element den Kopf einer Liste aller abhängigen Sektionen des Greifpunktes bildet. Als erste abhängige Sektion gilt die in der kinematischen Kette vom Greifer am weitesten entfernte, als letzte abhängige gilt die dem Greifer am nächsten liegende.
Ein Zeiger svar.first_rgp_specs_ptr zeigt in dieser Struktur auf sämtliche Informationen, ähnlich wie der Environment-Zeiger in COSIMIR auf die gesamte dortige Struktur zeigt.

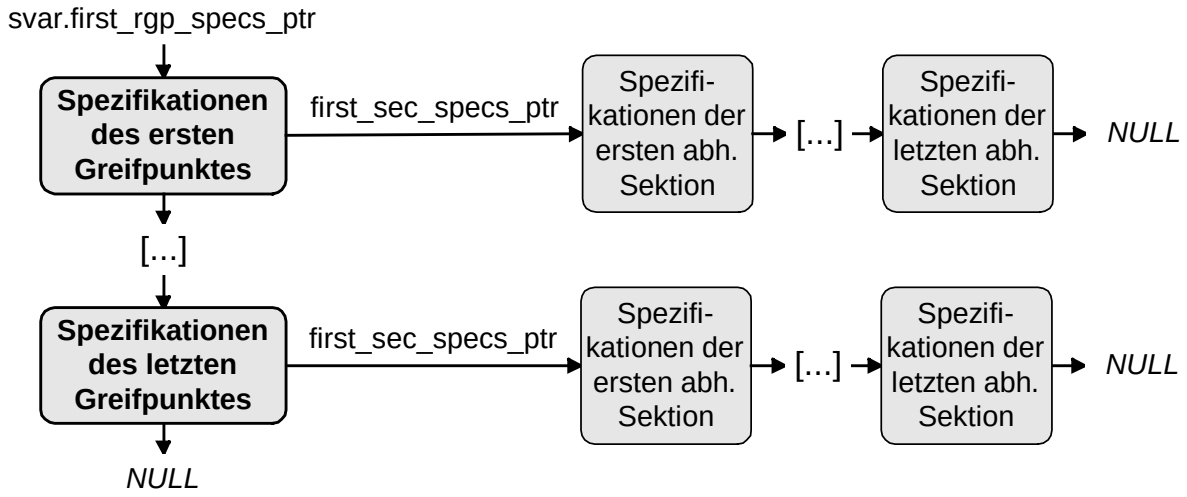

Abbildung 29 : Listenstruktur des spezifischen Umweltmodelles

Jedes Greifpunkt-Element ist dabei von der Struktur GP_SPECS, welche folgende Variablen enthält.

Typ	Name	Beschreibung
SECTION _SPECS	*first_sec_specs_ptr	Pointer auf Liste aller abhängigen Sektionen
RESTRICTED _GP_LIST	*rgp_ptr	Zeiger auf aktuellen eingeschränkten Greifpunkt im Cosimir Environment
BOOL	calculation_required	Wahr, falls Greifpunkt gegriffen ist oder losgelassen wurde und Geschwindigkeit mind. einer Sektion noch ungleich Null ist.
int	num_to_vary	Anz. der variablen Achsparam. des GP's
double	gripper [4][4]	Aktuelle Greiferlage. Bei Plattformen optimale mögliche Greiferlage. Bei Plattform-Beinen nur zu erreichende Positionsvektor wichtig.
int	num_of_unrivivable_calculations	Anzahl der Simulationsschritte, vor der Greifpunkt noch vollständig erreichbar war
double	last_final_ldelta [6]	Greiferabstand *nach* letztem Simulationsschritt
double	rivivable_delta_penalty [6]	Gewichtungsvektor im Falle der vollständigen Erreichbarkeit (nur zur Konvergenz-Entscheidung benötigt)
double	rivivable_delta_limit [6]	Abbruchschranken für Lagedifferenz im Falle der vollständigen Erreichbarkeit
int	num_of_approximations	Anzahl der Näherungen im Falle der unvollständigen Erreichbarkeit (1..9)
BOOL	use_nullspace	Wahr, falls Fehlermaßfunktion zur Nullraumnutzung addiert werden soll
int	search_mode	Suchmodus i.F.d.u.E. 0 = Gradientenverfahren 1 = Quasi-Newtonverfahren 2 = Newtonverfahren +5 = Zuerst Lösungsversuch mit Universaltransformation
double	unrivivable_delta_penalty [9][6]	Hauptdiagonalen der Gewichtungsmatrizen im Falle der unvollständigen Erreichbarkeit
double	unrivivable_delta_limit [9][6]	Abbruchschranken für Lagedifferenz i.F.d.u.E.
double	diversion_limit [9]	Abbruchschranke für Gradientensteigung im Fehlermaß i.F.d.u.E.
int	loops_until_warning	Anzahl der Minimierungs-iterationen, ab der eine Warnmeldung erscheinen soll
int	notice_joint_limits [9]	Grenzen für Achsparameter, die *während* jeweiliger Minimierung berücksichtigt werden sollen 0 = $\pm\infty$ 1 = Absolutwerte (q_{min}, q_{max}) 2 = In aktuellem Schritt erreichbare Werte (Bei eventueller Universaltransformation wird letzter *gültiger* Wert verwendet)

int	limit_mode	Begrenzungsart *nach* Gesamtminimierung 0 = ohne Begrenzung 1 = Sollpostion optimal erreichen ohne Zwangsbremsung 2 = wie 1, mit Zwangsbremsung 3 = Vermeidung von Überschwingungen
BOOL	closed_chain [1])	Wahr, falls Greifpunkt-Objekt eine geschlossene kinematische Kette ist
SECTION _SPECS	*whole_sec_specs_ptr [1])	Pointer auf Liste aller abhängigen Sektionen bis zur Pseudobasis
int	whole_num_to_vary [1])	Anz. der Elemente von pWhole_chain_section_specs
SECTION	*last_section_ptr [1])	Zeiger auf Pseudobasis-Sektion (Ist-Wert)
double	pseudo_base [4][4] [1])	Anfangslage der Pseudobasis (Soll-Wert) zur Vermeidung von Fehlerfortpflanzung
double	whole_delta_penalty [9][6] [1])	Hauptdiagonalen der Gewichtungsmatrizen der Pseudobasis-Lageabweichung
double	whole_delta_limit [9][6] [1])	Abbruchschranken für Lagedifferenz bei Pseudobasis-Lageabweichung
double	whole_diversion_limit [9] [1])	Abbruchschranke für Gradientensteigung
BOOL	leg [2])	Wahr, falls Greifpunkt-Objekt ein Bein einer Steward-Plattform ist
char	*my_platforms_name [2])	Name d. zugehörigen Plattform-Objektes
double	leg_length [2])	Länge eines gesamten Beines (Istwert)
double	leg_start_length [2])	Länge eines gesamten Beines zu Anfang eines Simulationsschrittes
double	leg_maximal_length [2])	Maximale Länge eines gesamten Beines
double	leg_minmal_length [2])	Minimale Länge eines gesamten Beines
double	leg_base [3] [2])	Basisfester Punkt (Zentrum des unteren Kugelgelenkes) in Weltkoordinaten
double	leg_amax [2])	Maximalbeschl. eines Beines $(\infty = 0)$
double	leg_vmax [2])	Maximalgeschw. eines Beines $(\infty = 0)$
double	leg_rivivable_max [2])	Maximal erreichbare Beinlänge in aktuellem Simulationsschritt
double	leg_rivivable_min [2])	Minimal erreichbare Beinlänge ...
double	leg_speed [2])	Änderungsgeschwindigkeit der Beinlänge
double	relative_vector [3] [2])	Vektor von Greifpunkt der Plattform bis Greifpunkt des Beines (bzg auf Koordinatensystem des Greifpunkt der Plattform)
BOOL	platform [3])	Wahr, falls Greifpunkt-Objekt eine Plattform ist
int	num_of_dependent_legs [3])	Anzahl der Beine einer Plattform
GP_SPECS	*leg_list [10] [3])	Liste mit Zeigern auf alle zur Plattform gehörenden Bein-GP_SPECS
GP_SPECS	*next	Zeiger auf nächstes Element

[1]) = Nur benötigt, falls Greifpunkt-Objekt eine geschlossene kinematische Kette ist
[2]) = Nur benötigt, falls Greifpunkt-Objekt ein Bein einer Plattform ist

[3]) = Nur benötigt, falls Greifpunkt-Objekt eine Plattform ist

Alle Sektions-Elemente sind von der Struktur SECTION_SPECS, mit folgenden Variablen.

Typ	Name	Beschreibung
int	number	Nummer der Sektion unter den veränderlichen in der kinematischen Kette (Basisnächste = 1)
double	DH_param [4]	Soll-Werte der Denavit-Hartenberg Parameter
int	DH_var	Stelle des variablen Denavit-Hartenberg Paramerters im Quatrupel (0..3)
double	DH_start_value	Startwert für DH_var in aktullem Sim.Schritt
double	DH_real	Als Gesamtösung errechneter Ist-Wert für variablen DH-Parameter
int	DH_index	Stelle des variablen DH- Paramerters im Objektwinkelvektor
double	DH_offset	Offset-Wert für DH_var
int	DH_sign	Zu multiplizierendes Vorzeichen
double	DH_abs_max	Absoluter Maximalwert für DH_var (nicht verwechseln mit max_pos im COSIMIR Umweltmodell)
double	DH_abs_min	Absoluter Minimalwert für DH_var (nicht verwechseln mit min_pos im COSIMIR Umweltmodell)
double	DH_rivivable_max	Maximal erreichbarer Wert für DH_var im aktuellen Simulationsschritt
double	DH_rivivable_min	Minimal erreichbarer Wert für DH_var im aktuellen Simulationsschritt
double	DH_valid_max	Maximalwert für DH_var in aktuellem Näherungsschritt
double	DH_valid_min	Minimalwert für DH_var in aktuellem Näherungsschritt
double	DH_vmax	Maximalgeschwinigkeit für DH_var $(\infty = 0)$
double	DH_amax	Maximalbeschleunigung für DH_var $(\infty = 0)$
double	DH_speed	Geschwindigkeit von DH_var
BOOL	joint_crossed_limit	DH_var hat in der aktuellen Iteration einen Grenzwert überschritten
BOOL	joint_at_limit	DH_var befindet sich auf einem Grenzwert
SECTION	*sec_ptr	Zeiger auf Sektion im Cosimir-Environment
SECTION	*predecessing_sec_ptr	Zeiger auf die Sektion im Cosimir-Environment, welche der beschriebenen Sektion in kin. Kette vorhergeht
SECTION_SPECS	*next	Zeiger auf nächstes Element

Es gilt hier : ... ->Angles[sec_specs_ptr->DH_index] =
(sec_specs_ptr ->DH_param [sec_specs_ptr ->DH_var]
- sec_specs_ptr ->DH_offset) * sec_specs_ptr ->DH_sign

Alle Variablen im spezifischen Umweltmodell sind statisch, da der Hauptzeiger ein Strukturelement der (einzigen) statischen Variable ‘svar’ ist. Die Gesamtstruktur besteht aus folgenden Anteilen. (Also enthält etwa svar.t in jeder Unterroutine die Simulationszeit.)

Typ	Name	Beschreibung

ENVIRONMENT	*mod_ptr	Zeiger auf aktuelles Cosimir-Environment
double	n [20][20]	Näherung der Hesse-Matrix im Quasi-Newton Verfahren
double	t	Zeit seit Simulationsbeginn
double	oldt	Zeit von Simulationsbeginn bis zum vorhergehenden Simulationsschritt
double	deltat	t - oldt (also die Dauer des akt. Simulationschrittes)
BOOL	platform_included	Wahr, falls Arbeitszelle min. eine Steward-Plattform enthält
BOOL	newton_impossible	Mit dem Newton-Verfahren kann in aktueller Achskonfiguration keine Suchrichtung bestimmt werden, da die Hessematrix nicht invertierbar ist.
BOOL	no_update	Zur Vermeidung von Update_workcell nach Update_SEDD Aufruf durch DLL
GP_SPECS	*first_rgp_specs_ptr	Hauptzeiger auf spezifisches Umweltmodell

Sämtliche anderen Variablen der DLL haben nur lokalen Gültigkeitsbereich.

8.2 Überblick über den Programmablauf der DLL

Die folgenden Abbildungen geben einen groben Überblick über das Zusammenspiel der wichtigsten Routinen und den prinzipiellen Aufbau der DLL. Ausgangspunkt ist zunächst die Initialisierung einer neuen Arbeitszelle (In diesem Fall erzeugt COSIMIR einen Aufruf der Routine 'InitWorkcellDll'). Einen zweiten Ausgangspunkt bildet ein Aufruf von 'UpdateWorkcellDll', mit dem COSIMIR der DLL jeweils vor dem Aufbau eines Bildschirminhaltes Gelegenheit gibt Änderungen am Environment vorzunehmen. Bei der hier vorgestellten DLL beschränkt sich dieser 'Eingriff' fast ausschließlich auf Objektwinkel (object_ptr->Angles). Einzige Ausnahme bildet das 'moveframe' eines Plattform-Objektes, welches immer direkt überschrieben wird, da ja keine zu variierenden Gelenkwinkel existieren.
Ferner wird mit jedem Schließen einer Arbeitszelle (COSIMIR ruft Routine 'FreeWorkcellDll' auf) der zuvor zur Erstellung des spezifischen Umweltmodelles reservierte Speicherplatz wieder freigegeben.
Da das spezifische Umweltmodell nach Erzeugung beziehungsweise Löschung von Template-Objekten (z.B. in Fließbandsimulationen) aktualisiert werden muß, werden diese Objekte auf eingeschränkte Greifpunkte untersucht. Sollten sie mindestens einen besitzen, so werden dessen Spezifikationen an die Gesamtliste angehängt beziehungsweise hieraus wieder entfernt. Reservierter Speicherplatz wird im zweiten Fall wieder freigegeben.

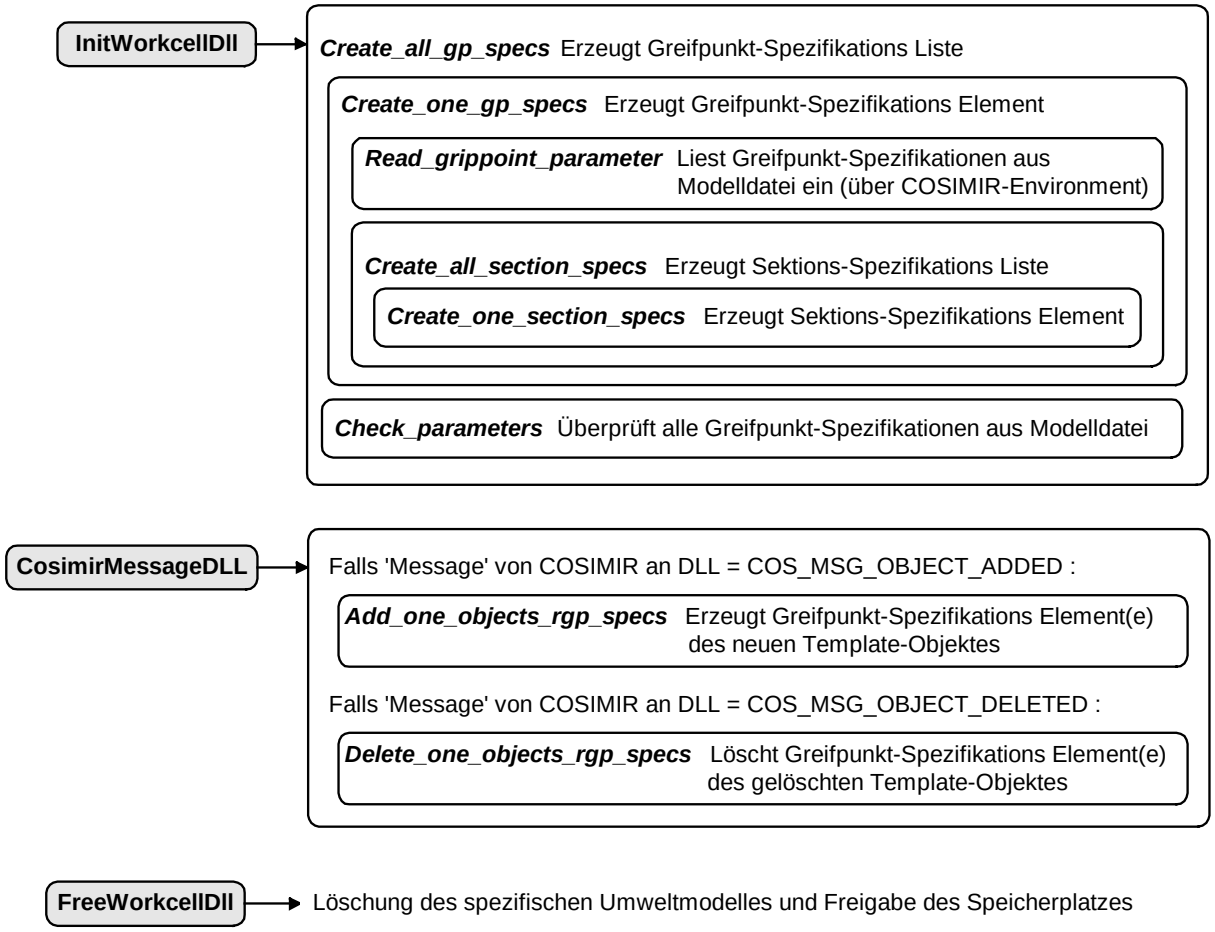

Abbildung 30 : Erzeugung, Aktualisierung und Löschung des spezifischen Umweltmodelles

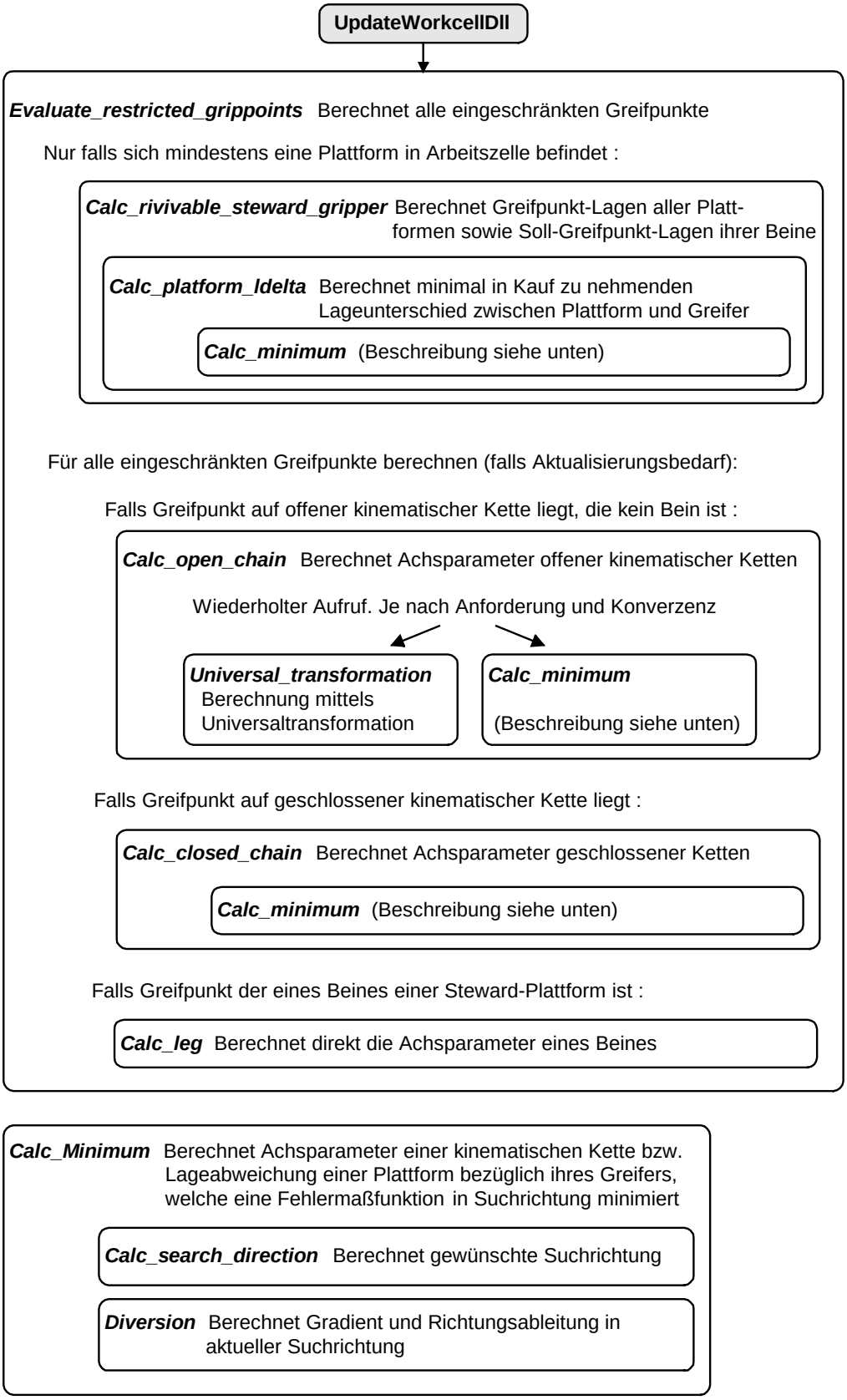

Abbildung 31 : Die grobe Programmstruktur mit den wichtigsten Routinen

8.3 Zu ergänzende Parameter in EGEMOL-Modelldateien

Einige der Parameter des spezifisches Umweltmodelles werden indirekt (über das Umweltmodell von COSIMIR) aus der entspechenden Arbeitszellen-Modelldatei eingelesen. Es handelt sich dabei fast ausschließlich um um Werte, welche speziell für jedes einzelne Modell gefunden werden müssen, damit dieses mit minimalem Rechenaufwand den spezifischen Anforderungen in der Simulation gerecht wird.
Alle nachfolgenden Angaben werden auf Greifpunktebene gemacht, das heißt sie sind für jeden eingeschränkten Greifpunkt in der Modelldatei anzugeben. Dabei wird zwischen den vier möglichen Objekttypen, an dem ein Greifpunkt modelliert sein kann unterschieden und nur jeweils die notwendigen Angaben eingelesen. Dieser Objekttyp ist im einzelnen

(object_type : "OPEN_CHAIN") , falls Objekt offene kinematische Kette ist
(object_type : "CLOSED_CHAIN") , falls Objekt geschlossene kinematische Kette ist
(object_type : "LEG") , falls Objekt ein Bein einer Plattform ist
(object_type : "PLATFORM") , falls Objekt die Plattform selbst ist

Redundante Informationen wie etwa den Suchmodus bei Plattform-Objekten ergänzt das Programm in der Initialisierungsphase.
Die Reihenfolge der Eingaben ist beliebig. So können beispielsweise die Hauptdiagonalen der Gewichtungsmatrizen mit ihren zugehörenden Abbruchschranken-Vektoren untereinander stehen, oder jeweils in Blöcken getrennt. Nicht benötigte Werte und Vektoren können (müssen aber nicht) weggelassen werden. Das Fließkomma-Zahlenformat ist frei wählbar (zB. 8, 2.3e-5, -31.233).

<vektor>	bezeichnet nachfolgend eine Array aus sechs durch Komma getrennten postive Fließkomma-Zahlen.
<#numbers>	bezeichnet eine Array aus mind. ebensovielen durch Komma getrennten postiven Fließkomma-Zahlen, wie Näherungen durchgeführt werden sollen (num_of_approximations).
#	steht für eine Zahl von 1 bis 9. (Nummer des Minimierungsschrittes)
[1])	Anzahl entspricht min. Zahl der Näherungen, die durchgeführt werden sollen

Alle Angaben in <> werden in der Modelldatei in " " gefaßt.
Das allen spezifischen Parametern vorrangestellte ‘IK’ bedeutet ‘Inverse Kinematik’. Der Vektor ‘IK_move_limit’ wird programmintern als unrivivable_delta_penalty[0][1..6] gespeichert. Alle anderen unter dem entsprechenden gleichen Namen.

<u>Falls das Greifpunkt-Objekt eine offene kinematische Kette ist (keine Plattform, kein Bein)</u>

```
(restricted_gp = <name>
   (grippointfame = <frame>)                { Wie bisher }
   (grip_range = <number>)                  { Wie bisher }
   (properties :
      (IK_search_mode = <number>)
      (IK_num_of_approximations = <number>)
      (IK_ loops_until_warning = <number>)
```

```
        (IK_use_nullspace = <number>)                { 1 = Nullraum nutzen, 0 = nicht nutzen }
        (IK_rivivable_delta_penalty = <vektor>)      { Nur erforderlich falls search_mode > 4 }
        (IK_rivivable_delta_limit = <vektor>)        { Nur erforderlich falls search_mode > 4 }
        (IK_unrivivable_delta_penalty_# = <vektor>) { [1]) }
        (IK_unrivivable_delta_limit_# = <vektor>)    { [1]) }
        (IK_diversion_limit = <#numbers)
        (IK_notice_joint_limits = <#numbers >)
        (IK_limit_mode = <number>)
    )
)
```

Falls das Greifpunkt-Objekt eine geschlossenen kinematische Kette ist

```
(restricted_gp = <name>
    (grippointfame = <frame>)                    { Wie bisher }
    (grip_range = <number>)                      { Wie bisher }
    (properties :
        (IK_search_mode = <number>)              { search_mode < 5 }
        (IK_num_of_approximations = <number>)
        (IK_ loops_until_warning = <number>)
        (IK_use_nullspace = <number>)            { 1 = Nullraum nutzen, 0 = nicht nutzen }
        (IK_unrivivable_delta_penalty_# = <vektor>) { [1]) }
        (IK_unrivivable_delta_limit_# = <vektor>)    { [1]) }
        (IK_diversion_limit = <#numbers)
        (IK_whole_delta_penalty_# = <vektor>)    { [1]) }
        (IK_whole _delta_limit_# = <vektor>)     { [1]) }
        (IK_whole _diversion_limit = <#numbers>)
        (IK_notice_joint_limits = <#numbers >)
        (IK_limit_mode = <number>)
    )
)
```

Falls das Greifpunkt-Objekt ein Bein einer Plattform ist

```
 (restricted_gp = <name>
    (grippointfame = <frame>)                    { Wie bisher, aber Einheitsmatrix }
    (grip_range = <number>)                      { Wie bisher, aber ohne Bedeutung }
    (properties :
        (IK_my_platforms_name = <char[30]>)
        (IK_leg_amax = <number>)
        (IK_leg_vmax = <number>)
    )
)
```

Falls das Greifpunkt-Objekt eine Plattform ist

```
 (restricted_gp = <name>
    (grippointfame = <frame>)                    { Wie bisher }
    (grip_range = <number>)                      { Wie bisher }
    (properties :
        (IK_ loops_until_warning = <number>)
```

```
        (IK_unrivivable_delta_penalty_1 = < vektor >) { Nur zur ℓΔ Berechnung }
        (IK_move_limit = < vektor >)                  { Nur zur ℓΔ Berechnung }
        (IK_diversion_limit = <number>)               { Nur zur ℓΔ Berechnung }
        (IK_notice_joint_limits = <number>)           { Wert gilt für alle abh. Bein-Objekte !!! }
        (IK_limit_mode = <number>)                    { Wert gilt für alle abh. Bein-Objekte !!! }
    )
)
```

Allgemein ist noch zu erwähnen, daß pro Sektion nur *ein* DH_Parameter als variabel modelliert sein darf. Sollten jedoch zwei oder mehr benötigt werden, so ist dieses problemlos durch Einfügung zusätzlicher (unsichtbarer) Sektionen erreichbar.

8.4 Parameterbeispiele

Im folgenden werden unterschiedliche Beispiele kurz erläutert, welche einen Einblick in die Auswahlkriterien der spezifischen Parameter geben. Die Beispiele sind nach Komplexität gestaffelt

Beispiel 1

Eine Schublade, welche sich nur in x-Richtung (im Greifpunkt-Koordinatensystem) bewegen kann, soll innerhalb ihrer Anschläge einer greifenden Hand (Datenhandschuh) folgen.

```
(restricted_gp : "drawer_grippoint"
        (grippointframe :
                1.0000000000,   0.0000000000,   0.0000000000,  500.0000000000,
                0.0000000000,   1.0000000000,   0.0000000000,  250.0000000000,
                0.0000000000,   0.0000000000,   1.0000000000,  100.0000000000,
                0.0000000000,   0.0000000000,   0.0000000000,    1.0000000000
        )
        (grip_range :   100.000)
        (properties:
                (IK_search_mode =                   "5")
                (IK_num_of_approximations =         "1")
                (IK_loops_until_warning =           "10")
                (IK_use_nullspace =                 "0")
                (IK_rivivable_delta_penalty =       "  1,  0,  0,  0,  0,  0")
                (IK_rivivable_delta_limit =         "  1,  0,  0,  0,  0,  0")
                (IK_unrivivable_delta_penalty_1 = "  1,  0,  0,  0,  0,  0")
                (IK_unrivivable_delta_limit_1 =     "  1,  0,  0,  0,  0,  0")
                (IK_diversion_limit =               "1")
                (IK_notice_joint_limits =           "1")
                (IK_limit_mode =                    "0")
        )
)
```

Der Suchmodus ist hier 5. Es wird also zunächst versucht mittels Universaltransformation eine Lösung zu errechnen und erst bei einer divergenten Lösungsfolge wird das hier ausreichende Gradientenverfahren verwendet.

Die ausschließlich Gewichtung der x-Abweichung zwischen Greifpunkt und Greifer im Lösungspunkt mit 1 (IK_rivivable_delta_penalty) sorgt für eine konvergente Lösungsfolge. Diese besteht bei dem hier betracheten Problem aus nur einem Schritt, sofern sich die Schublade innerhalb der Anschläge bewegt. Dann ist bereits die Abbruchschranke IK_rivivable_delta_limit in sämtlichen von Null verschiedenen und damit interessanten Werten unterschritten (hier nur max. x-Abweichung von 1mm) und die Lösung ist gefunden.

IK_notice_joint_limits ist im ersten und einzigen Element gleich 1. Die Anschlagswerte der Schublade werden daher berücksichtigt. Entfernt sich der Greifer in x-Richtung weiter von der Schublade, als diese folgen kann, so liefert die Universaltransformation keine ausreichende Lösung, da der betrachtete Lageunterschied in x-Richtung nicht mehr kleiner als 1mm werden kann. In diesem Fall errechnet das Gradientenverfahren die für die Schublage günstigste Bewegungsrichtung, stellt fest, daß keine Bewegung in dieser Richtung mehr eine Verbesserung der durch IK_unrivivable_delta_penalty_1 gewichteten Lösung bewirkt und bricht die Suche ab.

IK_use_nullspace ist 0 (=FALSE), da zu der Kinematik nie ein Nullraum existiert und ein solcher daher auch nicht berücksichtigt wird.

Wäre IK_notice_joint_limits gleich 0, so würden die Anschläge der Schublade ignoriert und die Universaltransformation würde stets in einem Schritt die exakte Lösung liefern.
Wäre der Suchmodus hier 0, so würde das Gradientenverfahren die für die Schublade günstige Bewegungsrichtung errechnen und sie in dieser Richtung soweit bewegen, bis keine Minimierung des Lageunterschiedes in x-Richtung mehr möglich wäre. Das ist in einem Anschlag der Fall, oder sobald der Gradient der durch IK_unrivivable_delta_penalty_1 gewichteten Lösung in seiner x-Komponente das IK_diversion_limit von 1 unterschreitet. Dort würde es in der nächsten Iterationen feststellen, daß das bestmögliche Minimum gefunden ist und abbrechen. 10 IK_loops_until_warning werden also nie erreicht, sondern maximal 2.

Beispiel 2

Eine Klappenschranktür, welche einen einzigen rotorischen Freiheitsgrad in z-Richtung (Greifpunkt-Koordinatensystem) besitzt, soll innerhalb ihrer Anschläge der Position einer greifenden Hand möglichst gut folgen. Der Greifpunkt bewegt sich also stets in der x-y Ebene (Greifpunkt-Koordinatensystem).

```
(restricted_gp : "drawer_grippoint"
    (grippointframe :
        1.0000000000,  0.0000000000,  0.0000000000,  100.0000000000,
        0.0000000000,  1.0000000000,  0.0000000000,  200.0000000000,
        0.0000000000,  0.0000000000,  1.0000000000, 1000.0000000000,
        0.0000000000,  0.0000000000,  0.0000000000,    1.0000000000
    )
    (grip_range :  300.000)
    (properties:
        (IK_search_mode =                "1")
        (IK_num_of_approximations =      "1")
        (IK_loops_until_warning =        "50")
        (IK_use_nullspace =              "0")
        (IK_unrivivable_delta_penalty_1 = "  1e2,  1e2,  0,  0,  0,  0")
```

```
            (IK_unrivivable_delta_limit_1 =    "     1,    1,  0,  0,  0,  0")
            (IK_diversion_limit =              "1")
            (IK_notice_joint_limits =          "1")
            (IK_limit_mode =                   "0")
        )
)
```

Da bei diesem Problem beinahe ausgeschlossen ist, daß der Greifpunkt exakt in die gleiche Position wie der Greifer bewegt werden kann, wird gar nicht erst versucht mittels Universaltransformation eine Lösung zu errechnen, sondern es wird hier mit dem Quasi-Newton Verfahren (Suchmodus = 1) gearbeitet um den Abstand zu minimieren. Es gibt jeweils nur eine Idealstellung der Klappe, also auch nur ein globales Minimum.

Der interessierende Greifpunkt-Greifer Abstand berechnet sich aus den beiden Komponenten in x und y Richtung. Alle anderen Komponenten werden mit Null gewichtet, sind also unwichtig (IK_unrivivable_delta_penalty_1). Da die erste Näherung auch die einzige (IK_num_of_approximations = 1) und somit letzte ist, werden beide Anteile mit 100 gewichtet, um die zusätzliche Fehlermaßfunktion, die die Klappe in einer Mittelstellung zwischen den Anschlägen zu halten versucht, zu überlagern. (Diese zusätzliche programminterne Fehlermaßfunktion hat immer maximal den Wert der der Anzahl der abhängigen Achsen entspricht.)
IK_unrivivable_delta_limit_1 gibt an, daß die Suche abgebrochen wird, sobald der Abstand in x und y Richtung jeweils 1 mm unterschreitet. Da das, wie erwähnt, sehr unwahrscheinlich ist, bricht die Suche vermutlich dadurch ab, daß der Gradient der Fehlermaßfunktion in seiner x und y Komponente das IK_diversion_limit von 1 unterschreitet.
Die Achsanschläge werden berücksichtigt (IK_notice_joint_limits = 0) und Maximalwerte für Beschleunigung und Geschwindigkeit des Drehgelenkes werden ignoriert (IK_imit_mode = 0).

Beispiel 3

Ein 6 achsiger Industierroboter mit 6 Achsen und günstigstenfalls 6 Freiheitsgraden, soll mit dem TCP der Lage einer greifenden Hand möglichst gut folgen. Dabei sollen sämtliche Anschläge sowie die Maximalwerte für Beschleunigung und Geschwindigkeit der Achsen eingehalten werden. Eine unter Umständen unvermeidbare Orientierungsabweichung soll dabei weniger bedeutsam sein, als eine unvermeidbare Positionsabweichung.

```
(restricted_gp : "R15_rest_gp"
    (grippointframe :
        1.0000000000,   0.0000000000,   0.0000000000,   0.0000000000,
        0.0000000000,   1.0000000000,   0.0000000000,   0.0000000000,
        0.0000000000,   0.0000000000,   1.0000000000,  99.5000000000,
        0.0000000000,   0.0000000000,   0.0000000000,   1.0000000000
    )
    (grip_range :   1000.000)
    (properties:
        (IK_search_mode = "6")
         (IK_num_of_approximations =        "4")
        (IK_loops_until_warning =           "1000")
        (IK_use_nullspace =                 "0")
        (IK_rivivable_delta_penalty =       "   1,   1,   1,   1e2,  1e2,  1e2")
        (IK_rivivable_delta_limit =         "   1,   1,   1,  1e-4, 1e-4, 1e-4")
```

```
            (IK_unrivivable_delta_penalty_1 =     "  1,   1,   1,  1e4,  1e4,  1e4")
            IK_unrivivable_delta_limit_1 =        "  1,   1,   1, 1e-3,1e-3,  1e-3")
            (IK_unrivivable_delta_penalty_2 =     "  1,   1,   1,  1e2,  1e2,  1e2")
            (IK_unrivivable_delta_limit_2 =       "  1,   1,   1, 1e-3, 1e-3,  1e-3")
            (IK_unrivivable_delta_penalty_3 =     "  1,   1,   1,    1,    1,    1")
            (IK_unrivivable_delta_limit_3 =       "  1,   1,   1, 1e-3, 1e-3,  1e-3")
            (IK_unrivivable_delta_penalty_4 =     "  1,   1,   1,    0,    0,    0")
            (IK_unrivivable_delta_limit_4 =       "  1,   1,   1,    0,    0,    0")
            (IK_diversion_limit =                 "1, 1, 1, 1")
            (IK_notice_joint_limits =             "2, 2, 2, 2")
            (IK_limit_mode =                      "3")
        )
)
```

Da der TCP unter Umständen die gewünschte Lage exakt erreichen kann, wird zunächst versucht mittels Universaltransformation eine Lösung zu errechnen. Im anderen Falle wird das Quasi-Newton Verfahren mit 4 Minimierungsschritten verwendet um eine möglichst gute TCP-Lage auszurechnen.
Dabei wird im ersten Schritt (IK_unrivivable_delta_penalty_1) eine Orientierungsabweichung sehr stark ‘bestraft’ und eine Positionsabweichung wenig. Mit jedem weiteren Schritt kehren sich dann die Verhältnisse schrittweise um, bis in der vierten Minimierung nur noch die Positionsabweichung minimiert wird.
Der TCP bewegt sich dadurch in jeder Berechnung aus einer guten Orientierung eine gute Position ohne dabei jedoch seine *möglichst* gute Orientierung zu verlieren.

Da IK_notice_joint_limits in allen vier Werten 2 ist, wird in jedem Schritt nur innerhalb der (durch die aktuellen Achsgeschwindigkeiten, -positionen und -maximalwerte festgelegten) erreichbaren Grenzen nach einem Minimum gesucht. Diese Grenzen sind so berechnet (IK_imit_mode = 3), daß Überschwingungen der Achsen nach Möglichkeit verhindert werden. Die Universaltransformation sucht ebenfalls nur in diesem Bereich nach einer Lösung, da der vierte (letzter gültiger) Wert von IK_notice_joint_limits eine 2 ist.

Jede der vier Minimierungen endet wieder, indem die karthesische Lageabweichung den jeweiligen Schrankenvektor IK_unrivivable_delta_limit unterschreitet oder der Gradient in allen Komponenten (nicht angeschlagener Achsen) das jeweilige IK_diversion_limit unterschreitet.

Beispiel 4

Wie Beispiel 3, jedoch mit Nullraumnutzung

```
(restricted_gp : "R15_rest_gp"

                        [...]

        (IK_use_nullspace =             "1)

                        [...]

        (IK_unrivivable_delta_penalty_4 =   " 1e3, 1e3, 1e3, 10, 10, 10")
        (IK_unrivivable_delta_limit_4 =     " 1e-6, 1e-6, 1e-6, 1e-6, 1e-6, 1e-6")
        (IK_diversion_limit =               "1, 1, 1, 0.1")

                        [...]
    )
)
```

Jede der ersten drei Minimierungen endet, indem die karthesische Lageabweichung den jeweiligen Schrankenvektor IK_unrivivable_delta_limit unterschreitet oder der Gradient in allen Komponenten (nicht angeschlagener Achsen) das jeweilige IK_diversion_limit unterschreitet.

IK_use_nullspace ist hier 1 (=TRUE), da zu der Kinematik unter Umständen ein Nullraum existiert und ein solcher in der vierten Minimierung dazu genutzt werden soll, die Achsen möglichst in Mittelstellungen zu belassen (Vermeidung von Anschlagsnähe). Der maximale Wert der hierfür addierten Fehlermaßfunktion ist 6 (Achsanzahl), der minimale Null. Die vierte Minimierungen endet daher, indem der Gradient in allen Komponenten (nicht angeschlagener Achsen) das vierte IK_diversion_limit unterschreitet. Dieses ist so niedrig angesetzt (0.1), daß die zusätzlich addierte Fehlermaßfunktion mitberücksichtigt wird.

9 *Anlagen*

10 Anlage A - Literaturverzeichniss, Web-Referenz

Literaturverzeichniss

[1] Freund, E. / Weber, H. : „Robotertechnologie I/II“, Skriptum zur Vorlesung, Institut für Roboterforschung und Lehrstuhl für Automatisierung und Robotertechnologie, Universität Dortmund (1996).

[2] Rossmann, J. : „Einführung in die Modellierung von Arbeitszellen mit EGEMOL“, Praktikum Automatisierungstechnik und Robotik zur Vorlesung, Versuch 1, Kapitel 3, Institut für Roboterforschung und Lehrstuhl für Automatisierung und Robotertechnologie, Universität Dortmund (1993/94).

[3] Papageorgiou, M. : „Optimierung“, 2. Auflage, Technische Universität Kreta (1996).

[4] Merziger, G. / Wirth, T. : „Repititorium der höheren Mathematik“ (1991).

[5] Novitzke, Udo : „Untersuchung eines Verfahrens zur Kollisionsvermeidung für redundante Roboter“, Institut für Roboterforschung und Lehrstuhl für Automatisierung und Robotertechnologie, Universität Dortmund (1995).

[6] Kovács, Peter : „Rechnergestützte symbolische Roboterkinematik“, Technische Universität Berlin (1991).

Web-Referenz

[1] http://www.mech.nwu.edu/MFG/AML/SPAEA/Model.html
Patcl, Amit : „SPAEA, Modeling Stewart Platform Errors“, Ingersoll, Northwestern University (1997).

11 Anlage B - Erklärung, Einwilligung

Erklärung

Ich versichere, daß ich diese wissenschaftliche Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen sind, wurden in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.
Das gleiche gilt auch für beigegebene Skizzen und Darstellungen.
Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Dortmund, den 3.2.1998

Einwilligung

Hiermit erkläre ich mich damit einverstanden, daß diese wissenschaftliche Arbeit nach den Bestimmungen des § 6 Abs. 1 des Gesetzes über Urheberrecht vom 9.9.1965 in der Bereichsbibliothek aufgenommen und damit für Leser der Bibliothek öffentlich zugänglich gemacht wird.
Ferner bin ich damit einverstanden, daß gemäß § 54 Abs. 1 Satz 1 dieses Gesetzes Leser zu persönlichen wissenschaftlichen Zwecken Kopien aus der Arbeit anfertigen dürfen.

Dortmund, den 3.2.1998

12 Anlage C - Herleitung der Zeit t_x bis zur Grenzverletzung

Falls $t_x < t_b$ und $r_n^{(s)}=1$ ($\dot{q}_n^{(s)}(t_x)$ ist daher positiv) ergibt sich

$$\dot{q}_n^{(s)}(t_x) = \sqrt{2\left(q_{max,\,n} - q_n^{(s)}(t_x)\right)\cdot a_{max,\,n}}$$

$$\Leftrightarrow \left(\dot{q}_n^{(s)}(0) + a_{max,\,n}\cdot t_x\right)^2 = 2\,q_{max,\,n}\cdot a_{max,\,n} - 2\,q_n^{(s)}(t_x)\cdot a_{max,\,n}$$

$$\Leftrightarrow \dot{q}_n^{(s)}(0)^2 + 2\,\dot{q}_n^{(s)}(0)\cdot a_{max,\,n}\cdot t_x + a_{max,\,n}^2\cdot t_x^{\,2} - 2\,q_{max,\,n}\cdot a_{max,\,n} + 2\left(q_n^{(s)}(0) + \dot{q}_n^{(s)}(0)\cdot t_x + \frac{a_{max,\,n}\cdot t_x^{\,2}}{2}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow 2\,a_{max,\,n}^2\cdot t_x^{\,2} + 4\,\dot{q}_n^{(s)}(0)\cdot a_{max,\,n}\cdot t_x - 2\,q_{max,\,n}\cdot a_{max,\,n} + \dot{q}_n^{(s)}(0)^2 + 2\,q_n^{(s)}(0)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow t_x^{\,2} + \frac{4\,\dot{q}_n^{(s)}(0)\cdot a_{max,\,n}}{2\,a_{max,\,n}^2}\,t_x - \frac{\left(2\,q_{max,\,n}\cdot a_{max,\,n} - \dot{q}_n^{(s)}(0)^2 - 2\,q_n^{(s)}(0)\cdot a_{max,\,n}\right)}{2\,a_{max,\,n}^2} = 0$$

$$\Leftrightarrow t_x = -\frac{\dot{q}_n^{(s)}(0)}{a_{max,\,n}} \pm \sqrt{\left(\frac{\dot{q}_n^{(s)}(0)}{a_{max,\,n}}\right)^2 + \frac{q_{max,\,n} - q_n^{(s)}(0)}{a_{max,\,n}} - \frac{\dot{q}_n^{(s)}(0)^2}{2\,a_{max,\,n}^2}}$$

$$\boxed{\Leftrightarrow t_x = -\frac{\dot{q}_n^{(s)}(0)}{a_{max,\,n}} \pm \sqrt{\frac{\dot{q}_n^{(s)}(0)^2}{2\,a_{max,\,n}^2} + \frac{q_{max,\,n} - q_n^{(s)}(0)}{a_{max,\,n}}}}$$

Falls $t_x < t_b$ und $r_n^{(s)}= -1$ ($\dot{q}_n^{(s)}(t_x)$ muß folglich negativ sein) ergibt sich

$$\dot{q}_n^{(s)}(t_x) = -\sqrt{\left(q_n^{(s)}(t_x) - q_{min,\,n}\right)\cdot a_{max,\,n}}$$

$$\Leftrightarrow \left(\dot{q}_n^{(s)}(0) - a_{max,\,n}\cdot t_x\right)^2 = -\,q_{min,\,n}\cdot a_{max,\,n} + q_n^{(s)}(t_x)\cdot a_{max,\,n}$$

$$\Leftrightarrow \dot{q}_n^{(s)}(0)^2 - 2\,\dot{q}_n^{(s)}(0)\cdot a_{max,\,n}\cdot t_x + a_{max,\,n}^2\cdot t_x^2 + 2\,q_{min,\,n}\cdot a_{max,\,n} - 2\left(q_n^{(s)}(0) + \dot{q}_n^{(s)}(0)\cdot t_x - \frac{a_{max,\,n}\cdot t_x^2}{2}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow 2\,a_{max,\,n}^2\cdot t_x^2 - 4\,\dot{q}_n^{(s)}(0)\cdot a_{max,\,n}\cdot t_x + 2\,q_{min,\,n}\cdot a_{max,\,n} + \dot{q}_n^{(s)}(0)^2 - 2\,q_n^{(s)}(0)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow t_x^2 - \frac{4\,\dot{q}_n^{(s)}(0)\cdot a_{max,\,n}}{2\,a_{max,\,n}^2}\,t_x + \frac{\left(2\,q_{min,\,n}\cdot a_{max,\,n} + \dot{q}_n^{(s)}(0)^2 - 2\,q_n^{(s)}(0)\cdot a_{max,\,n}\right)}{2\,a_{max,\,n}^2} = 0$$

$$\Leftrightarrow t_x = \frac{\dot{q}_n^{(s)}(0)}{a_{max,\,n}} \pm \sqrt{\left(\frac{\dot{q}_n^{(s)}(0)}{a_{max,\,n}}\right)^2 - \frac{q_{min,\,n} - q_n^{(s)}(0)}{a_{max,\,n}} - \frac{\dot{q}_n^{(s)}(0)^2}{2\,a_{max,\,n}^2}}$$

$$\boxed{\Leftrightarrow t_x = \frac{\dot{q}_n^{(s)}(0)}{a_{max,\,n}} \pm \sqrt{\frac{\dot{q}_n^{(s)}(0)^2}{2\,a_{max,\,n}^2} - \frac{q_{min,\,n} - q_n^{(s)}(0)}{a_{max,\,n}}}}$$

Falls $t_b < t_x < t_c$ und $r_n^{(s)}=1$ ergibt sich

$$\dot{q}_n^{(s)}(t_x) = \sqrt{2\left(q_{max,\,n} - q_n^{(s)}(t_x)\right)\cdot a_{max,\,n}}$$

$$\Leftrightarrow v_{max,\,n}^2 = 2\,q_{max,\,n}\cdot a_{max,\,n} - 2\,q_n^{(s)}(t_x)\cdot a_{max,\,n}$$

$$\Leftrightarrow v_{max,\,n}^2 - 2\,q_{max,\,n}\cdot a_{max,\,n} + 2\left(q_n^{(s)}(t_b) + v_{max,\,n}\cdot(t_x - t_b)\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow 2\,v_{max,\,n}\cdot t_x\cdot a_{max,\,n} + v_{max,\,n}^2 + 2\left(q_n^{(s)}(t_b) - v_{max,\,n}\cdot t_b - q_{max,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\boxed{\Leftrightarrow t_x = t_b - \frac{v_{max,\,n}}{2\,a_{max,\,n}} + \frac{q_{max,\,n} - q_n^{(s)}(t_b)}{v_{max,\,n}}}$$

Falls $t_b < t_x < t_c$ und $r_n^{(s)} = -1$ ergibt sich

$$\dot{q}_n^{(s)}(t_x) = -\sqrt{2\left(q_n^{(s)}(t_x) - q_{min,\,n}\right)\cdot a_{max,\,n}}$$

$$\Leftrightarrow\; v_{max,\,n}^2 = 2\,q_n^{(s)}(t_x)\cdot a_{max,\,n} - 2\,q_{min,\,n}\cdot a_{max,\,n}$$

$$\Leftrightarrow\; v_{max,\,n}^2 + 2\,q_{min,\,n}\cdot a_{max,\,n} - 2\left(q_n^{(s)}(t_b) - v_{max,\,n}\cdot(t_x - t_b)\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow\; 2\,v_{max,\,n}\cdot t_x\cdot a_{max,\,n} + v_{max,\,n}^2 - 2\left(q_n^{(s)}(t_b) + v_{max,\,n}\cdot t_b - q_{min,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\boxed{\Leftrightarrow\; t_x = t_b - \frac{v_{max,\,n}}{2\,a_{max,\,n}} + \frac{q_n^{(s)}(t_b) - q_{min,\,n}}{v_{max,\,n}}}$$

Falls $t_c < t_x < t_d$ und $r_n^{(s)} = 1$ ($\dot{q}_n^{(s)}(t_x)$ muß jetzt negativ sein) ergibt sich

$$\dot{q}_n^{(s)}(t_x) = -\sqrt{2\left(q_n^{(s)}(t_x) - q_{min,\,n}\right)\cdot a_{max,\,n}}$$

$$\Leftrightarrow\; \left(\dot{q}_n^{(s)}(t_b) - a_{max,\,n}\cdot(t_x - t_c)\right)^2 = -2\,q_{min,\,n}\cdot a_{max,\,n} + 2\,q_n^{(s)}(t_x)\cdot a_{max,\,n}$$

$$\Leftrightarrow\; \dot{q}_n^{(s)}(t_b)^2 - 2\,\dot{q}_n^{(s)}(t_b)\cdot a_{max,\,n}\cdot(t_x - t_c) + a_{max,\,n}^2\cdot(t_x - t_c)^2$$
$$- 2\left(q_n^{(s)}(t_c) + \dot{q}_n^{(s)}(t_b)\cdot(t_x - t_c) - \frac{a_{max,\,n}\cdot(t_x - t_c)^2}{2} - q_{min,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow\; \dot{q}_n^{(s)}(t_b)^2 - 2\,\dot{q}_n^{(s)}(t_b)\,a_{max,\,n}\cdot(t_x - t_c) + a_{max,\,n}^2\cdot\left(t_x^2 - 2\,t_c\cdot t_x + t_c^2\right)$$
$$- 2\left(q_n^{(s)}(t_c) + \dot{q}_n^{(s)}(t_b)\cdot(t_x - t_c) - \frac{a_{max,\,n}\cdot\left(t_x^2 - 2\,t_c\cdot t_x + t_c^2\right)}{2} - q_{min,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow\; 2\,a_{max,\,n}^2\cdot t_x^2 - \left(4\,\dot{q}_n^{(s)}(t_b) + 4\,a_{max,\,n}\cdot t_c\right)\cdot a_{max,\,n}\cdot t_x$$
$$+ \dot{q}_n^{(s)}(t_b)^2 - \left(2\,q_n^{(s)}(t_c) - 4\,\dot{q}_n^{(s)}(t_b)\cdot t_c - 2\,t_c^2\cdot a_{max,\,n} - 2\,q_{min,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow\ t_x^2 - \left(\frac{2\,\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} + 2\,t_c\right)\cdot t_x - \frac{q_n^{(s)}(t_c) - 2\,\dot{q}_n^{(s)}(t_b)\cdot t_c - q_{min,\,n}}{a_{max,\,n}} + t_c^2 + \frac{\dot{q}_n^{(s)}(t_b)^2}{2\,a_{max,\,n}^2} = 0$$

$$\Leftrightarrow\ t_x = \frac{\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} + t_c \pm \sqrt{\left(\frac{\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} + t_c\right)^2 + \frac{q_n^{(s)}(t_c) - 2\,\dot{q}_n^{(s)}(t_b)\cdot t_c - q_{min,\,n}}{a_{max,\,n}} - t_c^2 - \frac{\dot{q}_n^{(s)}(t_b)^2}{2\,a_{max,\,n}^2}}$$

$$\boxed{\Leftrightarrow\ t_x = t_c + \frac{\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} \pm \sqrt{\frac{\dot{q}_n^{(s)}(t_b)^2}{2\,a_{max,\,n}^2} - \frac{q_{min,\,n} - q_n^{(s)}(t_c)}{a_{max,\,n}}}}$$

Falls $t_c < t_x < t_d$ und $r_n^{(s)} = -1$ ($\dot{q}_n^{(s)}(t_x)$ muß jetzt positiv sein) ergibt sich

$$\dot{q}_n^{(s)}(t_x) = \sqrt{2\left(q_{max,\,n} - q_n^{(s)}(t_x)\right)\cdot a_{max,\,n}}$$

$$\Leftrightarrow\ \left(\dot{q}_n^{(s)}(t_b) + a_{max,\,n}\cdot(t_x - t_c)\right)^2 = 2\,q_{max,\,n}\cdot a_{max,\,n} - 2\,q_n^{(s)}(t_x)\cdot a_{max,\,n}$$

$$\Leftrightarrow\ \dot{q}_n^{(s)}(t_b)^2 + 2\,\dot{q}_n^{(s)}(t_b)\cdot a_{max,\,n}\cdot(t_x - t_c) + a_{max,\,n}^2\cdot(t_x - t_c)^2 + 2\left(q_n^{(s)}(t_c) + \dot{q}_n^{(s)}(t_b)\cdot(t_x - t_c) + \frac{a_{max,\,n}\cdot(t_x - t_c)^2}{2} - q_{max,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow\ \dot{q}_n^{(s)}(t_b)^2 + 2\,\dot{q}_n^{(s)}(t_b)\,a_{max,\,n}\cdot(t_x - t_c) + a_{max,\,n}^2\cdot\left(t_x^2 - 2\,t_c\cdot t_x + t_c^2\right) + 2\left(q_n^{(s)}(t_c) + \dot{q}_n^{(s)}(t_b)\cdot(t_x - t_c) + \frac{a_{max,\,n}\cdot\left(t_x^2 - 2\,t_c\cdot t_x + t_c^2\right)}{2} - q_{max,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow\ 2\,a_{max,\,n}^2\cdot t_x^2 + \left(4\,\dot{q}_n^{(s)}(t_b) - 4\,a_{max,\,n}\cdot t_c\right)\cdot a_{max,\,n}\cdot t_x + \dot{q}_n^{(s)}(t_b)^2 + 2\left(q_n^{(s)}(t_c) - 2\,\dot{q}_n^{(s)}(t_b)\cdot t_c + t_c^2\cdot a_{max,\,n} - q_{max,\,n}\right)\cdot a_{max,\,n} = 0$$

$$\Leftrightarrow t_x^2 + \left(\frac{2\,\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} - 2\,t_c \right) \cdot t_x + \frac{q_n^{(s)}(t_c) - 2\,\dot{q}_n^{(s)}(t_b) \cdot t_c - q_{max,\,n}}{a_{max,\,n}} + t_c^2 + \frac{\dot{q}_n^{(s)}(t_b)^2}{2\,a_{max,\,n}^2} = 0$$

$$\Leftrightarrow t_x = -\left(\frac{\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} - t_c \right) \pm \sqrt{\left(\frac{\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} - t_c \right)^2 - \frac{q_n^{(s)}(t_c) - 2\,\dot{q}_n^{(s)}(t_b) \cdot t_c - q_{max,\,n}}{a_{max,\,n}} - t_c^2 - \frac{\dot{q}_n^{(s)}(t_b)^2}{2\,a_{max,\,n}^2}}$$

$$\boxed{\Leftrightarrow t_x = t_c - \frac{\dot{q}_n^{(s)}(t_b)}{a_{max,\,n}} \pm \sqrt{\frac{\dot{q}_n^{(s)}(t_b)^2}{2\,a_{max,\,n}^2} + \frac{q_{max,\,n} - q_n^{(s)}(t_c)}{a_{max,\,n}}}}$$

Falls $t_x > t_d$ und $\tilde{\dot{q}}_n^{(s)} \geq 0 > 0$ ergibt sich

$$\dot{q}_n^{(s)}(t_x) = \sqrt{2\left(q_{max,\,n} - q_n^{(s)}(t_x)\right) \cdot a_{max,\,n}}$$

$$\Leftrightarrow \tilde{\dot{q}}_n^{(s)2} = 2\,q_{max,\,n} \cdot a_{max,\,n} - 2\,q_n^{(s)}(t_x) \cdot a_{max,\,n}$$

$$\Leftrightarrow \tilde{\dot{q}}_n^{(s)2} - 2\,q_{max,\,n} \cdot a_{max,\,n} + 2\left(q_n^{(s)}(t_d) + \tilde{\dot{q}}_n^{(s)} \cdot (t_x - t_d)\right) \cdot a_{max,\,n} = 0$$

$$\Leftrightarrow 2\,\tilde{\dot{q}}_n^{(s)} \cdot t_x \cdot a_{max,\,n} + \tilde{\dot{q}}_n^{(s)2} + 2\left(q_n^{(s)}(t_d) - \tilde{\dot{q}}_n^{(s)} \cdot t_d - q_{max,\,n}\right) \cdot a_{max,\,n} = 0$$

$$\boxed{\Leftrightarrow t_x = t_d - \frac{\tilde{\dot{q}}_n^{(s)}}{2\,a_{max,\,n}} + \frac{q_{max,\,n} - q_n^{(s)}(t_d)}{\tilde{\dot{q}}_n^{(s)}}}$$

Falls $t_x > t_d$ und $\tilde{\dot{q}}_n^{(s)} < 0$ ergibt sich

$$\dot{q}_n^{(s)}(t_x) = -\sqrt{2\left(q_n^{(s)}(t_x) - q_{min,\,n}\right) \cdot a_{max,\,n}}$$

$$\Leftrightarrow \tilde{\dot{q}}_n^{(s)2} = 2\, q_n^{(s)}(t_x) \cdot a_{max,\,n} - 2\, q_{min,\,n} \cdot a_{max,\,n}$$

$$\Leftrightarrow \tilde{\dot{q}}_n^{(s)2} + 2\, q_{min,\,n} \cdot a_{max,\,n} - 2\left(q_n^{(s)}(t_d) + \tilde{\dot{q}}_n^{(s)} \cdot (t_x - t_d)\right) \cdot a_{max,\,n} = 0$$

$$\Leftrightarrow -2\, \tilde{\dot{q}}_n^{(s)} \cdot t_x \cdot a_{max,\,n} + \tilde{\dot{q}}_n^{(s)2} - 2\left(q_n^{(s)}(t_d) - \tilde{\dot{q}}_n^{(s)} \cdot t_d - q_{min,\,n}\right) \cdot a_{max,\,n} = 0$$

$$\Leftrightarrow 2\, \tilde{\dot{q}}_n^{(s)} \cdot t_x \cdot a_{max,\,n} = \tilde{\dot{q}}_n^{(s)2} + 2\left(\tilde{\dot{q}}_n^{(s)} \cdot t_d - q_n^{(s)}(t_d) + q_{min,\,n}\right) \cdot a_{max,\,n}$$

$$\boxed{\Leftrightarrow t_x = t_d + \frac{\tilde{\dot{q}}_n^{(s)}}{2\, a_{max,\,n}} + \frac{q_{min,\,n} - q_n^{(s)}(t_d)}{\tilde{\dot{q}}_n^{(s)}}}$$