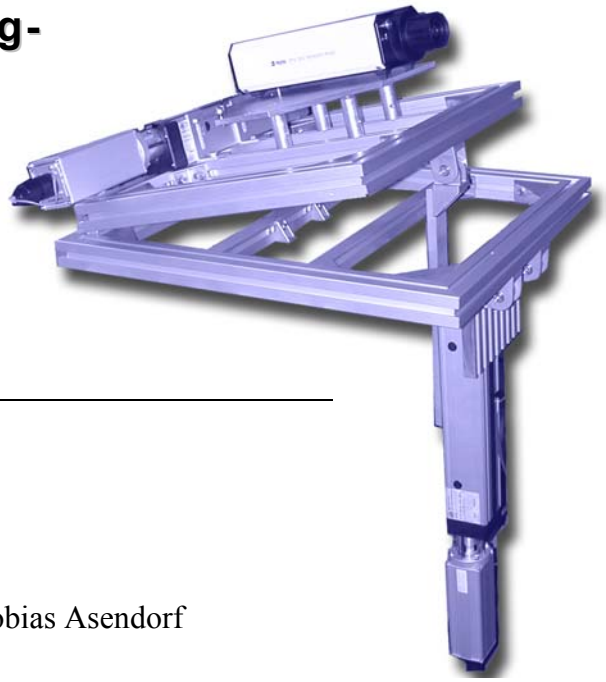
CARL VON OSSIETZKY universität OLDENBURG

Schriftliche Hausarbeit zur Prüfung für das Lehramt an Gymnasien

Thema der Arbeit:

Aufbau eines Scanning-Laservibrometers zur Visualisierung schwingender Oberflächen



vorgelegt von: Tobias Asendorf

Beurteilender Hochschullehrer: Prof. Dr. Volker Mellert

Zweitgutachter Dr. Reinhard Weber

Oldenburg, den 23. Juli 2004

Anlage: 1 CD-ROM

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung und Zielsetzung

Mechanische Schwingungen bzw. Vibrationen sind für das Auge nahezu unsichtbar. Trotzdem sind sie häufig die Ursache für unerwünschte Schallabstrahlungen, Störungen oder gar Beschädigungen. Diese können an Karosserie, Motor oder Getriebe von z.B. Autos oder Bussen auftreten, aber auch durch Lüfter und Laufwerke in einem PC verursacht werden. Um sie zu vermeiden, müssen sie zunächst sichtbar gemacht werden.

Das Visualisieren von schwingenden Oberflächen ist ein Ziel dieser Examensarbeit. Piezoelektrische Schwingungsaufnehmer oder Handmessgeräte bieten die Möglichkeit, Schwingungen auf Oberflächen in mehreren Punkten zu erfassen, ebenso wie berührungslos arbeitende Laser-Doppler-Vibrometer. Der Nachteil all dieser Messgeräte liegt auf der Hand: Es können immer nur einzelne Punkte gemessen werden; möchte man hingegen die vollflächigen Schwingungen einer Struktur erfassen, ist der zeitliche Aufwand für das manuelle Positionieren erheblich.

Hier bieten Scanning-Laser-Doppler-Vibrometer (*SLDV*) eine komfortable Möglichkeit. Sie arbeiten mit einer automatischen Positioniereinheit, die ein Abscannen vieler Messpunkte in kurzer Zeit gestattet. Ihr Nachteil ist der hohe Anschaffungspreis - im Rahmen dieser Examensarbeit soll daher ein solches Scanning-Laser-Doppler-Vibrometer mit einfachen Mitteln realisiert werden. Dazu wird ein bereits vorhandenes Ein-Punkt-Laservibrometer (*LDV*) der Firma

Polytec verwendet, das mit Hilfe eines noch zu konstruierenden Gerätes automatisch positioniert wird und so Oberflächen abscannen soll.

Die Steuerung und Auswertung soll über einen PC erfolgen; die Software Matlab bietet sich hier besonders an, da sie es erlaubt, über die serielle Schnittstelle des PCs externe Komponenten anzusteuern und zudem eine umfassende Programmbibliothek für die Messdatenauswertung zur Verfügung stellt.

Nachdem im nächsten Kapitel weitere Messmethoden zur Schwingungsmessung vorgestellt worden sind, werden zur Erläuterung der Funktionsweise eines Laservibrometers die theoretischen Grundlagen erörtert. Im vierten Kapitel erfolgt eine Beschreibung der entwickelten Hard- und Software des Scanning-Laservibrometers.

Die Ergebnisse von drei verschiedenen Messungen werden im anschließenden fünften Kapitel dargestellt - hier wurden die Oberflächen eines mit einem Shaker angeregten Panels, eines Breitbandlautsprechers und eines PC-Gehäuses abgescannt.

Die Ergebnisse dieser Arbeit, ein Ausblick auf die Möglichkeiten des Einsatzes des Scanning-Laservibrometers und seiner Weiterentwicklung werden im anschließenden sechsten Kapitel erörtert.

Kapitel 2

Methoden zur Messung von Schwingungen

Schwingungen von Oberflächen können auf unterschiedliche Art gemessen werden. Man unterscheidet hier zunächst berührungslose Messverfahren, etwa mit einem Laser-Doppler-Vibrometer, und Kontaktmessverfahren, die z.B. mit Schwingungsaufnehmern (sog. „vibration transducers“) oder Handmessgeräten arbeiten.

Im Allgemeinen haben diese den Nachteil, dass Schwingungen immer nur in einem einzigen Punkt gemessen werden können, bzw. dass die Zahl der gleichzeitigen Messpunkte auf die Anzahl der Schwingungsaufnehmer begrenzt ist.

Scanning-Laser-Doppler-Vibrometer hingegen können eine Vielzahl von Punkten in kurzer Zeit und damit die Schwingungen ganzer Flächen erfassen.

In diesem Kapitel werden zunächst diese Messmethoden beschrieben und anschließend die Motivation der Entwicklung des dieser Arbeit zugrunde liegenden Scanning-Laser-Doppler-Vibrometers erläutert.

2.1 Schwingungsaufnehmer

Das Messen von Schwingungen mit Hilfe von Schwingungsaufnehmern (genauer: Beschleunigungsaufnehmern) basiert auf dem piezoelektrischen Wirkprinzip. Wird auf ein piezoelektrisches Material Druck ausgeübt oder wird es

mechanischen Spannungen ausgesetzt, produziert es elektrische Ladungen. Kombiniert man dieses piezoelektrische Material nun mit einer Masse, erhält man ein zur Schwingungsbeschleunigung proportionales Ladungssignal. Der schematische Aufbau eines typischen Beschleunigungsaufnehmers ist in Abb. 1 dargestellt (vgl. [MMF]).

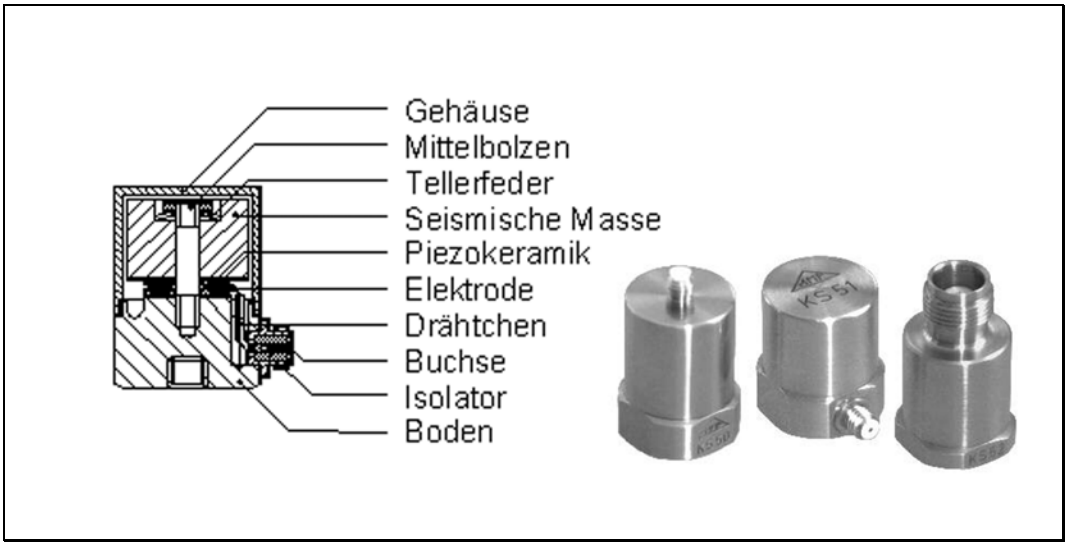


Abbildung 1: Schematische Darstellung eines Beschleunigungsaufnehmers (*Quelle: [MMF]*)

Das Messen von Schwingungen mit Beschleunigungsaufnehmern hat viele Vorteile. Ein hoher Dynamikumfang und geringes Rauschen ermöglicht das Erfassen sehr schwacher und sehr starker Schwingungen gleichermaßen. Die große Auswahl der auf dem Markt erhältlichen Sensoren lässt zudem eine optimale Abstimmung der Sensoren auf das Messobjekt zu. Das Signal des Sensors, also i.d.R. die Ladungen - bei anderen Typen auch eine Gleichspannung - wird an einen Ladungsverstärker bzw. Messverstärker geleitet. Dieser hat oftmals mehrere (typischerweise 8, 16, 32 oder 48) voneinander getrennte Signaleingänge, an denen die Sensoren angeschlossen werden können. Die weitere Signalanalyse erfolgt anschließend mit einem PC.

Die Empfindlichkeit von Beschleunigungsaufnehmern variiert je nach Wunsch von wenigen Piko-Coulomb (pC) pro g (g = 9,81 m/s²) bis zu mehreren Tausend pC/g . Der Frequenzbereich gestaltet sich ebenso flexibel und liegt je nach gewünschter Auflösung im Intervall von ca. 0,1 Hz bis 25 kHz.
Ein entscheidender Nachteil bei diesem System ist die Tatsache, dass die Beschleunigungsaufnehmer mit der zu vermessenden Oberfläche fest verbunden sein müssen. Dies ist bei einer festen Installation, etwa zur Überwachung von Maschinen, weniger von Bedeutung. Bei der Messung von Schwingungen an Oberflächen von Objekten mit geringer Masse darf die Eigenmasse der Sensoren hingegen nicht so groß sein, dass sie das Messergebnis entscheidend beeinflusst.
Ein weiterer Nachteil ist die Beschränkung der Anzahl der Sensoren. Sollen mehrere Punkte auf einem Objekt gemessen werden, müssen die Sensoren entweder in ausreichender Zahl zur Verfügung stehen oder jedes Mal neu positioniert und fixiert werden.

2.2 Handmessgeräte

Tragbare Vibrationsmessgeräte arbeiten nach dem gleichen Prinzip wie ein piezoelektronischer Beschleunigungsaufnehmer. Dieser ist in einem solchen Gerät entweder in einem Handgerät eingebaut, dessen Nagelfühler direkt auf die schwingende Oberfläche gehalten wird, oder er ist in einer externen Sonde untergebracht, die z.B. mit Hilfe eines Haftmagneten auf der Oberfläche befestigt wird. Ein Beispiel eines solchen Handschwingungsmessers ist in Abb. 2 dargestellt und soll kurz exemplarisch vorgestellt werden: Das Gerät der Firma PCE Group (vgl. [PCE]) misst Beschleunigungen im Bereich bis 200g mit einer

Abbildung 2: Schwingungsmesser der Firma PCE Group (*Quelle: [PCE]*)

Auflösung von bis zu 2 mg, Geschwindigkeiten im Bereich bis 2000 mm/s mit einer Auflösung von 0,1 m/s und Auslenkungen bis zu 20 µm mit einer Auflösung von 0,2 µm. Der messbare Frequenzbereich liegt ungefähr zwischen 10 Hz und 20 kHz.
Der Vorteil dieser Geräte liegt sicherlich in ihrer schnellen Einsatzbereitschaft und Flexibilität. Sie sind sehr gut dafür geeignet, die charakterisierenden Größen einer Schwingung - Beschleunigung, Geschwindigkeit und Verschiebung - unmittelbar zu erfassen, etwa an großen Maschinenbauteilen, Getrieben usw..
Ein Nachteil ist allerdings, dass die Messspitze bzw. der Sensor ebenso wie bei den Beschleunigungsaufnehmern Kontakt zur vermessenden Oberfläche hat, was wiederum die Schwingung in diesem Punkt beeinflusst. Die Tatsache, dass das Gerät mit dem Sensor in der Hand gehalten wird, verfälscht das Ergebnis der Messung nochmals.
Das Gerät eignet sich dadurch nur eingeschränkt zur Messung von Vibrationen in mehreren Punkten einer Oberfläche, insbesondere, wenn der schwingende Körper eine geringe Masse hat.

2.3 Das Polytec Scanning-Laser-Doppler-Vibrometer PSV-400

Das Scanning Vibrometer der Firma Polytec (vgl. [Poly-B]) basiert auf dem Einpunkt-Vibrometer-Messkopf OFV-505, einer Weiterentwicklung des in dieser Arbeit verwendeten Messkopfes OFV-303, und bietet im Vergleich zu diesem zusätzlich eine Autofokus- und Autofokus-Memory-Funktion. Als Controller dient der OFV-5000, ein Nachfolgemodell des OFW-3001 (siehe Abb. 3). Das Messsystem PSV-400 umfasst ferner eine Software zur Steuerung des Scanvorgangs, zur Datenaufnahme, zur Auswertung und zur Visualisierung der Schwingungen. Im Vergleich zu den oben vorgestellten Messgeräten ist mit dem Vibrometer nicht nur eine berührungslose Messung möglich, die Scanfunktion

gestattet zudem, Schwingungen in bis zu 512 x 512 Punkten einer Fläche von wenigen mm² bis zu einigen m² mit einer Geschwindigkeit von mehr als 100 Punkten pro Sekunde zu erfassen (Vorabscan).

Abbildung 3: PSV-400 Scanning Vibrometer (*Quelle: [Poly-B]*)

Die Scannereinheit hat einen Arbeitsbereich von ungefähr ± 20° in x- und y-Richtung und arbeitet mit einer maximalen Auflösung von unter 0,002°. Die Geschwindigkeits-Messbereiche dieses Systems entsprechen denen des Vorgängermodells, das in Abschnitt 3.3 beschrieben wird. Je nach installiertem Decoder-Typ können Beschleunigungen bis zu 200.000g und Frequenzen bis zu 1,5 MHz gemessen werden (vgl. [Poly-B] und [Poly97], Kapitel 8).
Durch eine integrierte Video-Farbkamera können die visualisierten Schwingungen direkt mit dem Messobjekt überlagert werden, so dass eine optimale Anschauung der Messung ermöglicht wird.

2.4 Motivation der Entwicklung eines eigenen Scanning-Laser-Vibrometers

Nachdem in den vorherigen Abschnitten verschiedene Systeme zur Messung von Schwingungen auf Oberflächen vorgestellt wurden, sollen nun abschließend die Beweggründe zum Bau des dieser Arbeit zu Grunde liegenden Scanning-Laser-Vibrometers aufgeführt werden.

Wie bereits angesprochen, gibt es viele Möglichkeiten, mit Hilfe von Beschleunigungsaufnehmern oder Handmessgeräten Schwingungen auf Oberflächen punktweise zu vermessen. Der Nachteil dieser Methoden liegt auf der Hand: Es können immer nur wenige Punkte gleichzeitig oder nur ein Punkt zur Zeit gemessen werden. Die Positionierung, Fixierung und Verkabelung der Beschleunigungsaufnehmer kostet viel Zeit und die Messdatenerfassung mit einem Handgerät erfolgt in der Regel nicht automatisiert. Zudem verfälschen diese Kontaktmessmethoden das Ergebnis, wenn die Eigenmasse der Sensoren im Verhältnis zur Masse des Messobjekts zu groß ist.

Ein fertiges Messsystem wie das PSV-400 Scanning-Vibrometer der Firma Polytec bietet zwar eine berührungslose Messung von mehr als 250.000 Messpunkten auf einer Oberfläche und eine optimale visuelle Darstellung, der Anschaffungspreis ist überdies jedoch sehr hoch.

Berücksichtigt man nun, dass bereits ein älteres Modell eines Ein-Punkt-Laser-Vibrometers vorhanden ist, liegt es nahe, dieses in ein System zum Scannen von schwingenden Oberflächen umzubauen.

Die Anforderungen, die dieses System erfüllen soll, sind mit dem Thema dieser Examensarbeit bereits vorgegeben worden: Zur Visualisierung schwingender Oberflächen ist es notwendig, eine automatische Positionierung des Messkopfes

zu realisieren, so dass die Schwingungen (genauer: die Geschwindigkeit der Oberfläche in einem Punkt) in mehreren Punkten nacheinander erfasst und dargestellt werden können. Zudem sollte die Software dieses Systems eine Vielzahl an Einstellungs- und Berechnungsmöglichkeiten bieten, die eine umfassende Datenanalyse zulassen.

Um im Rahmen dieser Arbeit ein möglichst kostengünstiges System aufzubauen, sollten weitestgehend vorhandene Geräte verwendet werden. Hier bietet sich u.a. ein verfügbarer Schrittmotorcontroller der Firma ISEL an, der bis zu drei Motoren bzw. Achsen steuern und über eine serielle Schnittstelle mit einem PC verbunden werden kann. Die günstigste und einfachste Möglichkeit, hieraus ein Positionierungssystem für das Vibrometer zu realisieren, besteht darin, die Dreh- und Kippbewegung des Messkopfes mit Hilfe zweier Kugelgewindevorschübe umzusetzen.

Als Basis für die Softwareentwicklung wird auf das zur Verfügung stehende Programm Matlab zurückgegriffen, da hiermit eine umfassende Datenauswertung und Visualisierung erfolgen kann.

Kapitel 3

Funktionsweise und Aufbau eines Laser-Doppler-Vibrometers

In diesem Kapitel sollen die theoretischen Grundlagen der Messung von Schwingungen mit Hilfe eines Laservibrometers erarbeitet werden. Das Messen von Schwingungen bzw. Vibrationen mit Hilfe eines Laservibrometers basiert auf der Nutzung des Doppler-Effekts und der Interferometrie: Die Frequenz eines Laserstrahls wird durch die Schwingungen des Objektes moduliert und anschließend interferometrisch ausgewertet. Zusammen mit einigen Grundlagen der Signalanalyse sollen diese Begriffe in diesem Abschnitt erörtert werden.

3.1 Der Dopplereffekt

Die Funktionsweise eines Laser-Doppler-Vibrometers basiert auf der Messung der Dopplerverschiebung eines kohärenten Lichtstrahls, der auf eine vibrierende Oberfläche trifft und reflektiert wird.

Dieser Dopplerverschiebung zugrunde liegt der Dopplereffekt (vgl. [Alo92], S. 575), benannt nach dem in Deutschland geborenen österreichischen Physiker C. J. Doppler (1803-1853). Er beschreibt das Phänomen, dass die Frequenz einer Welle, deren Ursprung sich relativ zur Position eines Empfängers bewegt, verschieden ist von der Frequenz der Quelle. Beobachtet wurde dies zuerst an Schallwellen, hier beziehen sich die Geschwindigkeit des Empfängers und die

Geschwindigkeit des Senders auf das als ruhend angesehene Medium Luft mit der Schallgeschwindigkeit c.
Entfernt sich nun der Empfänger vom Sender, verkleinert sich die Relativgeschwindigkeit zwischen Schallwelle und Empfänger. Bewegt sich andererseits der Sender auf den Empfänger zu, verkürzt sich die Wellenlänge um den Weg, den der Sender während der Dauer einer Schwingung zurücklegt.

Das Prinzip der Dopplerverschiebung gilt auch für elektromagnetische Wellen, in diesem Fall Licht. Allerdings ist hier kein Medium vorhanden, auf das die Geschwindigkeit des Senders oder des Empfängers bezogen wird. Zudem ist die Fortpflanzungsgeschwindigkeit der elektromagnetischen Wellen für alle Beobachter gleich der Lichtgeschwindigkeit c_0 - der Dopplereffekt muss hier notwendigerweise mit Hilfe des Relativitätsprinzips berechnet werden. Hierbei ergibt sich für die von einem Beobachter empfangene Frequenz ν'.

$$\nu' = \nu \frac{1-\frac{v}{c_0}}{\sqrt{1-\frac{v^2}{c_0^2}}} = \nu \frac{\sqrt{1-\frac{v^2}{c_0^2}}}{1+\frac{v}{c_0}} = \nu \sqrt{\frac{1-\frac{v}{c_0}}{1+\frac{v}{c_0}}} = \nu \sqrt{\frac{c_0-v}{c_0+v}}. \qquad (1)$$

für eine Bewegung des Empfängers vom Sender weg. Bei einer Bewegung aufeinander zu ist entsprechend -v einzusetzen.

Für kleine Geschwindigkeiten v ergibt sich

$$\nu' = \nu\left(1+\frac{v}{c_0}-\frac{v^2}{2c_0^2}+-...\right), \qquad (2)$$

wobei hier bereits der quadratische Term (v^2 / $2c_0^2$) und alle folgenden vernachlässigt werden können. Es ergibt sich also die Beziehung

$$\nu' \approx \nu\left(1\pm\frac{v}{c_0}\right), \qquad (3)$$

mit einem „+“ für eine Annäherung zwischen Sender und Empfänger und einem „-“ für das Entfernen des Senders vom Empfänger. Für die Frequenzverschiebung erhält man die sogenannte Dopplerfrequenz

$$\nu_D = \Delta\nu = \nu\frac{v}{c_0}. \tag{4}$$

Betrachtet man nun das Laservibrometer, trifft hier kohärentes Licht auf ein vibrierendes Objekt, wird reflektiert und in der Nähe des Senders wieder detektiert. Das vibrierende Objekt als Empfänger einerseits kann also andererseits auch als Sender betrachtet werden, so dass sich insgesamt die doppelte Frequenzverschiebung

$$\nu_D = \Delta\nu = \nu\frac{2v}{c_0} = \frac{2v}{\lambda} \tag{5}$$

mit v gleich der Geschwindigkeit des Objektes und λ der Wellenlänge der ursprünglichen Welle ergibt.

Soll nun umgekehrt die Geschwindigkeit eines Objektes bestimmt werden, muss die Frequenzverschiebung bei einer bekannten Wellenlänge des Lasers (hier $\lambda = 632{,}8$ nm) gemessen werden.

2.2 Interferometrie

Mit Hilfe eines Interferometers wird im Laser-Doppler-Vibrometer die Frequenzverschiebung gemessen, die zur Bestimmung der Geschwindigkeit des Objektes benötigt wird. Optische Interferenz entsteht immer dann, wenn sich kohärente Wellen überlagern - treffen sie etwa nach unterschiedlich langen zurückgelegten Wegen wieder zusammen, tritt eine Verstärkung der Intensität auf, wenn der Gangunterschied ein geradzahliges Vielfaches von $\lambda/2$ beträgt und eine

Schwächung, wenn der Gangunterschied ein ungeradzahliges Vielfaches von λ/2 ist.

2.2.1 Das Mach-Zehnder-Interferometer

Ein typischer Versuchsaufbau zur Messung der Interferenz ist das Mach-Zehnder-Interferometer, das in Abb. 4 dargestellt ist, und welches auch in modifizierter Art im Laservibrometer verbaut ist (vgl. [Poly97], Kapitel 5).

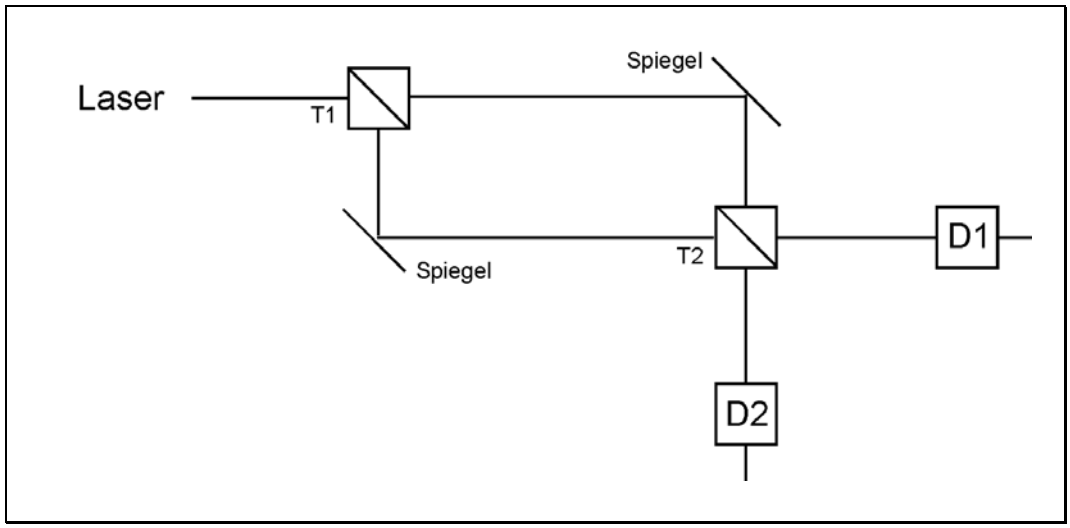


Abbildung 4: Mach-Zehnder-Interferometer

Beim Mach-Zehnder-Interferometer wird die vom Laser ausgesandte kohärente Welle mit Hilfe zweier gekitteter Prismen oder halbdurchlässiger Spiegel (T1) aufgeteilt, jeweils über einen Spiegel um 90° umgelenkt und in T1 wieder zusammengeführt. Die beiden so entstandenen überlagerten Wellenzüge werden auf zwei Schirmen (bzw. Photodetektoren) D1 und D2 dargestellt, wo sich ein Interferenzmuster ergibt.

An dieser Stelle noch eine kurze Anmerkung: Die gekitteten Prismen bzw. halbdurchlässige Spiegel werden im angloamerikanischen Sprachraum als

‚Beamsplitter', also ‚Strahlteiler' bezeichnet. Da bei einem Interferometer aber gerade die *Wellen*natur des Lichts Grundlage der Interferenz ist, ist diese Benennung irreführend. Ich werde die gängige Bezeichnung im Folgenden trotzdem beibehalten und von ‚Strahlteilern', sowie ‚Teilstrahlen' reden, auch wenn Licht als elektromagnetische Welle gemeint ist.
Um das Phänomen der Interferenz nun genauer beschreiben zu können, stellt man die ebene, kohärente Welle allgemein als

$$a_0 = A \cdot e^{i\omega t} \cdot e^{-ikz} = A \cdot e^{i(\omega t - kz)} \tag{6}$$

dar (mit der Amplitude A, der Kreisfrequenz des Lichts ω, der Zeit t und der Ausbreitungsrichtung z, sowie $k = 2\pi/\lambda$ mit der Wellenlänge λ).

Da die Welle beim Durchgang durch ein gekittetes Prisma aufgeteilt wird, verringert sich die Amplitude um den Faktor zwei. Zudem ergibt sich durch die unterschiedliche Weglänge eine Phasendifferenz θ für beide Teilstrahlen, so dass diese wie folgt beschrieben werden können:

$$\begin{aligned} a_1 &= \frac{1}{2} A \cdot e^{i(\omega t - \theta_1)} \\ a_2 &= \frac{1}{2} A \cdot e^{i(\omega t - \theta_2)} \end{aligned} \tag{7}$$

Die Phasendifferenzen $\Delta\theta = \theta_2 - \theta_1$ hängen von den Wegunterschieden $\Delta z = z_2 - z_1$ der beiden Teilstrahlen ab: $\Delta\theta = k \cdot \Delta z = \frac{2\pi}{\lambda} \Delta z$.

Die Amplitude der interferierenden Wellen in D2 ergibt sich nun durch Addition der Einzelamplituden ($a = a_1 + a_2$), die Intensität berechnet sich aus dem Betragsquadrat, bei komplexen Zahlen ist dies $I_i = a_i \cdot \overline{a_i}$, also aus dem Produkt der Amplitude mit der konjugiert komplexen Amplitude.

Für die Intensität in D2 erhält man damit den folgenden Zusammenhang:

$$I_{D2} = a \cdot \bar{a} = \frac{1}{2} A^2 \left[1 + \cos(\theta_2 - \theta_1)\right] \tag{8}$$

Da diese Intensität zusammen mit der in D1 der Intensität des Lasers (A^2) entsprechen muss, ergibt sich für D1:

$$I_{D1} = \frac{1}{2} A^2 \left[1 - \cos(\theta_2 - \theta_1)\right] \tag{9}$$

Mit der obigen Gleichung für die Phasendifferenzen und diesen beiden Gleichungen für die Intensitäten kann nun ein solches Interferometer dazu verwendet werden, kleinste Wegunterschiede Δz in der Größenordnung der halben Wellenlänge des verwendeten Lichts zu messen.

2.2.2 Modifiziertes Mach-Zehnder-Interferometer

Mit dem im vorherigen Abschnitt beschriebenen Mach-Zehnder-Interferometer ist es noch nicht möglich, die Bewegung eines außerhalb des Interferometers liegenden Objekts zu messen. Hierzu müsste eine Teilwelle nach außen treten, auf das sich bewegende Objekt treffen und die (messbare) Phasendifferenz der reflektierten Welle erfasst werden.

Dies kann durch eine kleine Modifizierung erreicht werden (siehe Abb. 5). Dazu wird ein Spiegel durch zwei weitere gekittete (polarisierende) Prismen (T2) ersetzt und zusätzlich ein Lambda-Viertel-Plättchen in den Strahlengang eingebracht (vgl. [Poly97], Kapitel 5).

Diese beiden Elemente haben den Effekt, dass die Teilwelle des Lasers in Richtung des Messobjekts (schraffiert dargestellt) ungehindert passieren kann, auf

dem Rückweg jedoch direkt in Richtung T3 umgeleitet und mit dem Referenzstrahl überlagert wird.

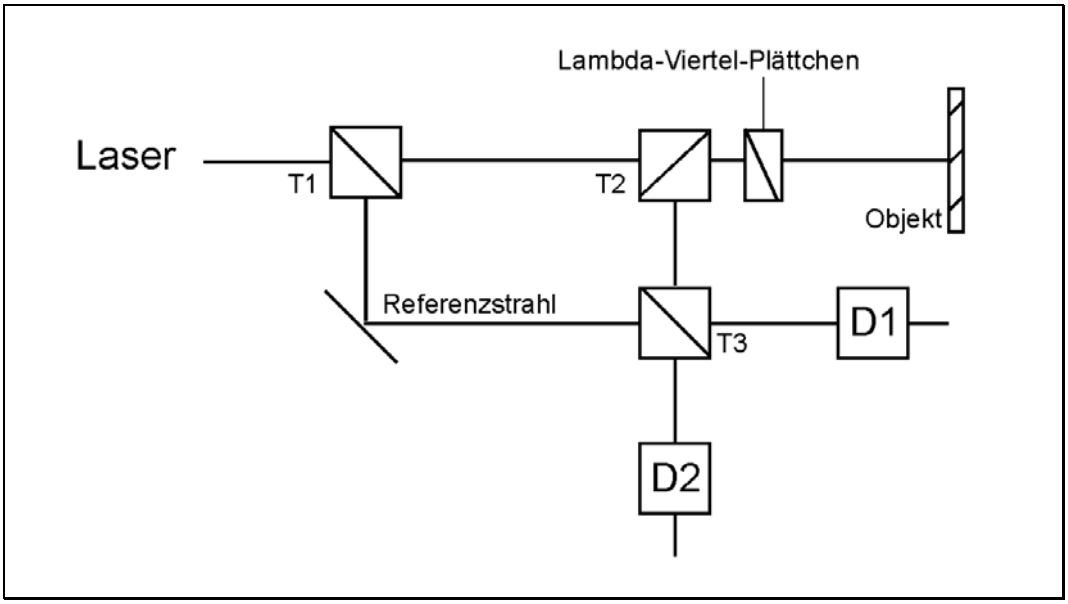


Abbildung 5: Modifiziertes Mach-Zehnder-Interferometer

Wird nun weiterhin angenommen, dass das „interne“ Interferometer symmetrisch aufgebaut ist, hängt die Phasendifferenz $\Delta\theta$ von der Weglänge L (Strahlteiler T2 zum Messobjekt) wie folgt ab:

$$\Delta\theta = 4\pi \cdot \frac{L}{\lambda} \tag{10}$$

Angenommen, das Objekt bewegt sich mit einer konstanten Geschwindigkeit v in Richtung des Lasers, dann erhält man eine zeitabhängige Gleichung für L = v t.
In Gleichung (10) eingesetzt und in Verbindung mit (5) erhält man

$$\Delta\theta = 4\pi \cdot \frac{v \cdot t}{\lambda} = 2\pi \cdot \nu_D \cdot t$$

mit der Dopplerfrequenz ν_D . Diese Gleichung für $\Delta\theta$ kann nun wiederum in die Gleichungen (8) und (9) eingesetzt werden:

$$I_{D1} = \frac{1}{2} A^2 \left[1 + \cos(2\pi\nu_D t)\right] \tag{11}$$

$$I_{D2} = \frac{1}{2} A^2 \left[1 - \cos(2\pi\nu_D t)\right] \tag{12}$$

Mit Hilfe dieser Beziehung zwischen den Intensitäten und der Dopplerfrequenz kann nun letztere über die Hell-Dunkel-Zyklen des Interferenzmusters bestimmt werden.

Ein Problem ergibt sich hier jedoch: Wie in den Gleichungen (11) und (12) zu erkennen, „sehen“ die beiden Photodetektoren D1 und D2 die gleiche Dopplerfrequenz ν_D. Dies ist verständlich, denn eine Objektbewegung vom Interferometer weg erzeugt das gleiche Muster von Hell-Dunkel-Zyklen wie eine Bewegung des Objekts auf das Interferometer zu. Die Richtung der Bewegung kann mit diesem modifizierten Mach-Zehnder-Interferometer also noch nicht bestimmt werden.

Gelöst wird dieses Problem indem eine Bragg-Zelle, also ein akusto-optischer Modulator, in den Referenzstrahl eingebracht wird (siehe Abb. 7). Diese bewirkt, dass die Frequenz des Referenzstrahls um 40 MHz verschoben wird und somit eine Modulationsfrequenz des Interferenzmusters von 40 MHz erzeugt wird, wenn das Messobjekt still steht. Bewegt sich nun das Objekt auf das Interferometer zu, wird diese Modulationsfrequenz erniedrigt, bei einer Bewegung vom Interferometer weg wird sie erhöht. Somit kann auch die Bewegungsrichtung eindeutig bestimmt werden. Die Photodetektoren registrieren nun die Frequenz

$$\nu_{res} = \nu_{Bragg} + \nu_D = \nu_{Bragg} + \frac{2v}{\lambda}, \tag{13}$$

wobei $\nu_{Bragg} = 40$ MHz beträgt.

2.2.3 Das Polytec OFV-303 Single Point Interferometer

Die im vorherigen Abschnitt beschriebene Modifikation eines Mach-Zehnder-Interferometers ist dafür geeignet, die Geschwindigkeit eines sich in Richtung des Interferometers bewegenden Objekts zu messen.
Der Standard-Messkopf OFV-303 (siehe Abb. 6) des verwendeten Polytec-Vibrometers (vgl. [Poly97], Kapitel 5) beinhaltet ein solches Interferometer (siehe schematische Darstellung in Abb. 7).

Abbildung 6: Der OFV-303 Standard-Messkopf

Hier wird ein Helium-Neon-Laser als kohärente Lichtquelle verwendet. Das um 45° zur Horizontalen polarisierte Licht trifft auf den (polarisierenden) Strahlteiler BS1, welcher zwei wiederum senkrecht zueinander polarisierte Teilstrahlen erzeugt: einen, der über den Strahlteiler BS2, welcher ein Lambda-Viertel-Plättchen enthält, durch eine Linse fokussiert auf das Objekt trifft, und einen Referenzstrahl, der eine Bragg-Zelle durchläuft, wo er mit einer Frequenz von 40 MHz moduliert wird.

Diese beiden Strahlen treffen im Strahlteiler BS3 wieder aufeinander und werden hier überlagert, was ein moduliertes Interferenzsignal erzeugt. Über zwei

Photodioden D1 und D2 wird dieses dann in ein elektrisches Signal umgewandelt. Die resultierende Ausgangsspannung ist gegeben durch

$$V = K \cdot \cos\left[2\pi(\nu_{Bragg} + 2v/\lambda) \cdot t\right] \tag{14}$$

mit einer Konstanten K, der Modulationsfrequenz ν_{Bragg} und der Dopplerfrequenz $\nu_D = 2v/\lambda$.

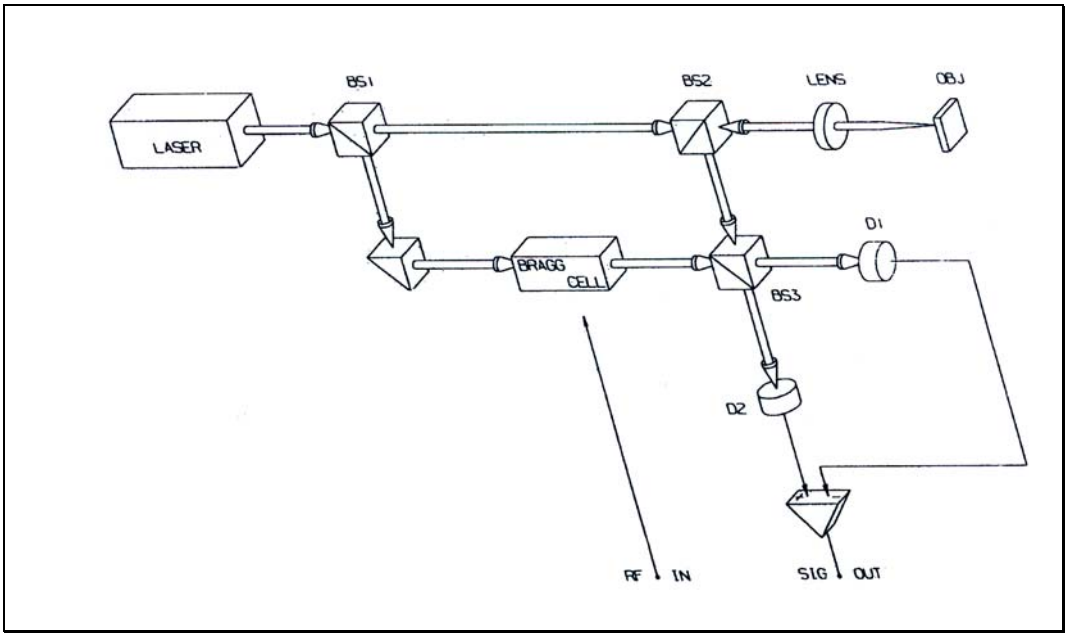


Abbildung 7: Schematische Darstellung des OFW-303 Single Point Interferometers (*Quelle: Poly97, Kapitel 5*)

Dieses Signal wird anschließend an den OFW-3001 Controller Processor (siehe Abb. 8) übermittelt.

In dieser zum Messgerät gehörenden Einheit werden daraus die gewünschten Informationen über die Geschwindigkeit (im velocity decoder) bzw. Auslenkung (im deplacement decoder) des Objekts ermittelt.

2.3 Das Polytec Vibrometer 3000

Nachdem in den vorherigen Abschnitten die Funktionsweise eines Laser-Doppler-Vibrometers erläutert wurde, soll nun kurz auf das verwendete Messgerät der Firma Polytec eingegangen werden. Das System besteht aus den oben bereits genannten zwei Komponenten: dem Ein-Punkt-Messkopf (Standard-Messkopf OFV-303) und dem Controller Processor OFW-3001 (siehe Abb. 8).

Abbildung 8: Der OFW-3001 Controller Processor

Nachdem der Controller Processor über das mitgelieferte D-Sub-Kabel mit dem optischen Messkopf verbunden wurde, ist das Messsystem bereits einsatzbereit.

An der Vorderseite des Controllers befinden sich zwei Ausgänge: Zum einen ein BNC-Anschluss „Velocity Output“, an dem eine proportional zur Schnelle modulierte Wechselspannung anliegt und zum anderen ein BNC-Anschluss „Displacement Output“, an dem eine Gleichspannungssignal proportional zur Auslenkung (Amplitude) entnommen werden kann.

Der Controller kann über eine serielle RS-232-Schnittstelle (nicht-gekreuztes Kabel) oder über ein paralleles GPIB/IEEE-488-Interface an einen PC angeschlossen werden. Über die serielle Schnittstelle können mit Hilfe von ASCII-Zeichenketten Steuerungsbefehle an den Controller gesendet werden, etwa um die Messbereiche auszulesen bzw. zu ändern oder Informationen über die Signalstärke zu erhalten.

Neben der Steuerung über einen PC kann der Controller auch über vier Tasten neben dem Display bedient werden. Neben den Informationen zur Signalstärke und zum installierten Typ des ‚Velocity Decoders' können hier die unterschiedlichen Messbereiche für die Schnelle, die in unserem Fall hauptsächlich gemessen werden soll, eingestellt werden:

Sie betragen 5, 25, 125 und 1000 $\frac{\text{mm}/\text{s}}{\text{V}}$. Zudem kann ein Filter mit 100 kHz, 20 kHz oder 5 kHz bei Bedarf aktiviert werden.

Kapitel 4

Aufbau des Scanning-Laser-Vibrometers

Wie im zweiten Kapitel bereits angesprochen, soll das Scanning-Vibrometer mit Hilfe zweier Schrittmotor-Kugelgewindevorschübe, einer Rahmenkonstruktion zur Positionierung, der Software Matlab und mit einem PC zur Steuerung und Auswertung realisiert werden. Dazu sind zu Anfang u.a. Überlegungen notwendig, wie die Kugelgewindevorschübe dimensioniert und in die Rahmenkonstruktion eingebaut werden müssen. Der Bau einer entsprechenden Rahmenkonstruktion soll durch die hauseigene Werkstatt erfolgen.
Die Entwicklung der Software erfolgt schrittweise - zunächst wird die Ansteuerung der Hardware erprobt, danach sollen einzelne Messpunkte eingelesen und ausgewertet werden. Im Anschluss hieran werden die einzelnen Programmteile zusammengefügt.

4.1 Die Hardware

In den folgenden Abschnitten soll auf die Entwicklung und den Aufbau der Hardware näher eingegangen werden. Diese besteht zum einen aus den bereits vorhandenen Geräten, etwa dem Schrittmotorcontroller und einem PC, und zum anderen aus den Teilen, die bestellt, neu konstruiert und gebaut werden müssen, also den Kugelgewindevorschüben und der Rahmenkonstruktion zur Positionierung des Messkopfes.

In Abb. 9 ist der Aufbau des Scanning-Laser-Vibrometers schematisch dargestellt. Als zentrales Element für die Steuerung, zur Erfassung von Messwerten und zur Auswertung dient ein Laptop oder PC mit der Software Matlab. Der Line-In-Eingang der Soundkarte wird dabei mit dem Schnellesignal-Ausgang des Polytec Controller Procerssors verbunden.

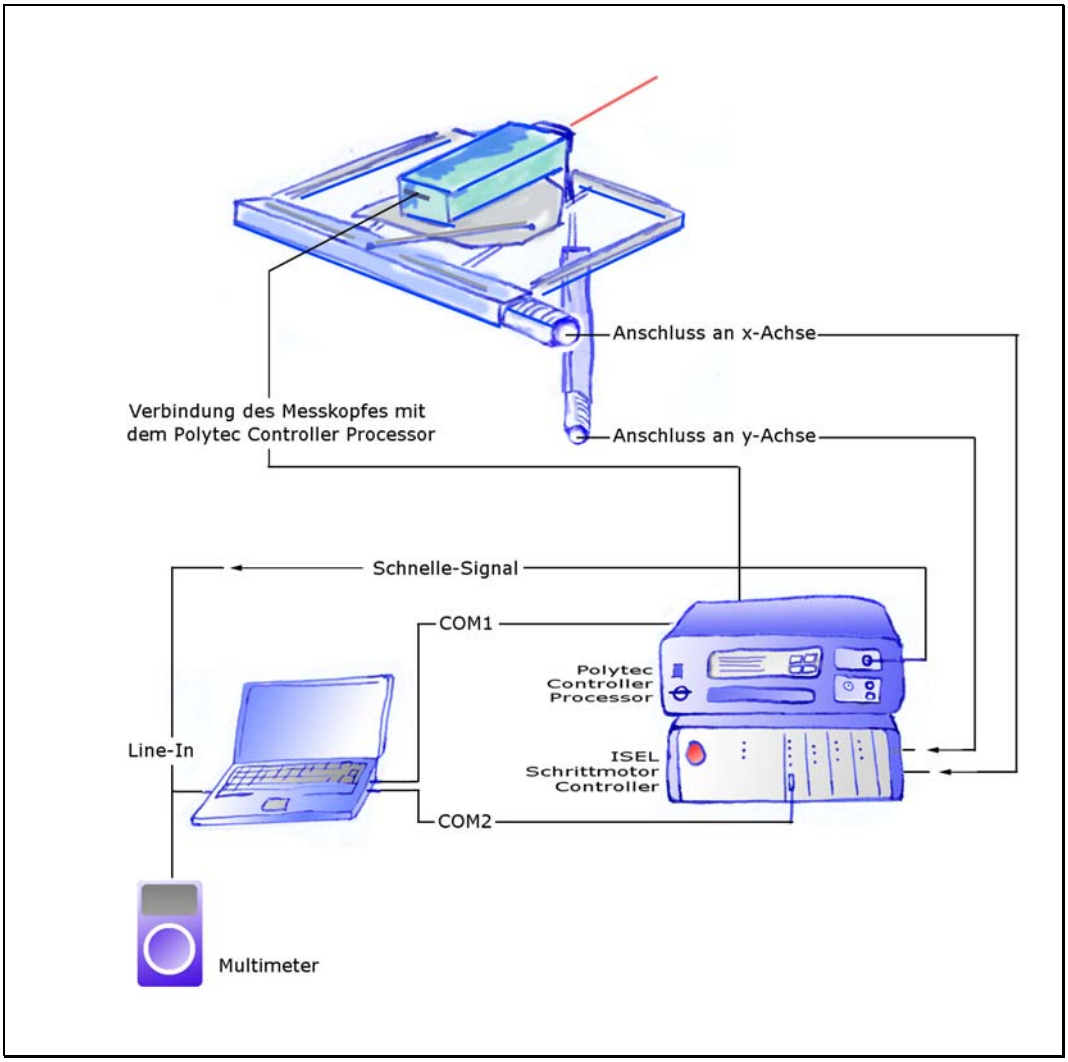


Abbildung 9: Schematischer Aufbau des Scanning Laservibrometers

Damit der Eingang der Soundkarte nicht übersteuert wird und dies zu einem Defekt führt, kann die Spannung dieses Signals zusätzlich mit einem Multimeter überprüft werden. Ist die Spannung zu hoch, muss der Messbereich am Controller angepasst werden. Der Controller selber wird an die serielle Schnittstelle des PCs angeschlossen (COM1), ebenso wie der Schrittmotorcontroller (COM2). An letzteren werden zudem die Schrittmotoren über zwei Kabel mit Amphenolstecker angeschlossen. Die Verbindung des Messkopfes mit dem Polytec Controller erfolgt über das mitgelieferte D-SUB-Connector-Kabel.

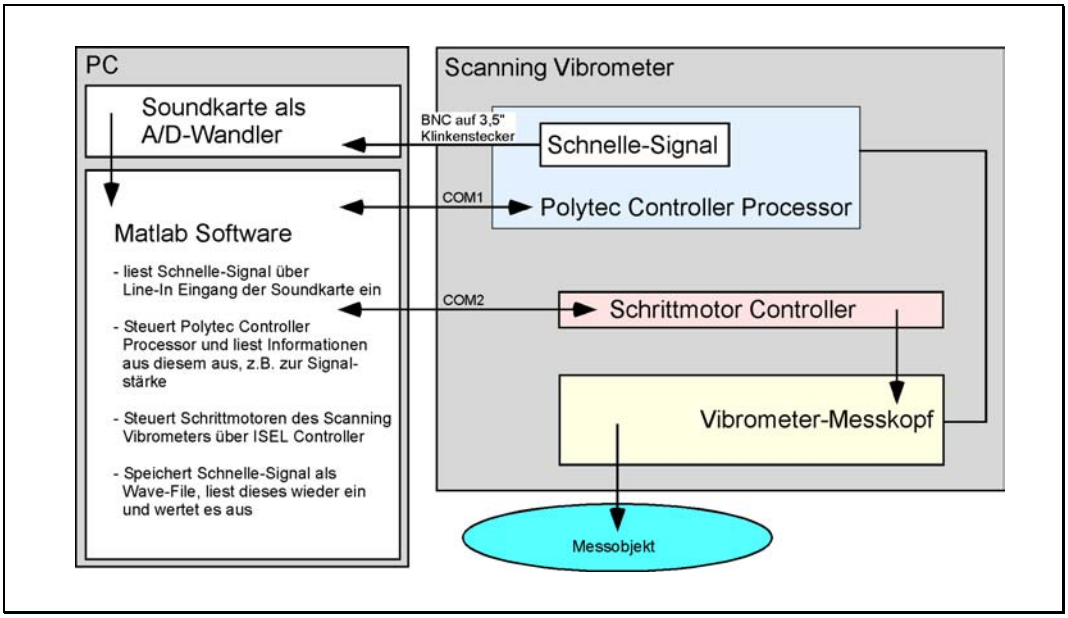


Abbildung 10: Schematische Darstellung der Einzelkomponenten und ihr Zusammenspiel

In Abb. 10 ist zusätzlich noch das Zusammenspiel der Einzelkomponenten schematisch dargstellt.

4.1.1 Schrittmotorsteuerung und Kugelgewindevorschübe

Da bereits ein Schrittmotorcontroller ISEL C 142-1 vorhanden ist, liegt es nahe, die Positionierung des Messkopfes über zwei Schrittmotoren zu realisieren. Diese sollte so erfolgen, dass ein Kugelgewindevorschub den Messkopf auf der y-Achse hebt und senkt und ein weiterer den Messkopf auf einer drehbaren Platte in x-Richtung schwenkt. In Abb. 11 ist dies dargestellt.

Abbildung 11: Aufbau der Rahmenkonstruktion mit Kugelgewinde-vorschüben und Messkopf

Die Schrittmotoren der Firma ISEL sind bereits in Kombination mit den Kugelgewindevorschüben erhältlich. Der in diesem System verbaute Schrittmotor MS 160 hat ein bipolares Haltemoment von 1,6 Nm und zeichnet sich durch ein annähernd konstantes Drehmoment von etwa 100 Ncm im Bereich einer Geschwindigkeit von 1 bis 10.000 Schritten pro Sekunde aus. Da die Kugelgewindevorschübe nicht selbsthemmend sind, ist der Motor für die y-Achsen-Positionierung mit einer entsprechenden Haltebremse ausgestattet.

Die Kugelgewindevorschübe bestehen aus einem Kugelgewindetrieb mit 2,5 mm Steigung und einem Durchmesser von 16 mm, sowie einem Alu-Wellenschlitten, der darauf spielfrei angebracht ist (Herstellerangabe). Der Schlitten bewegt sich bei 40.000 Schritten des Motors ca. 25,0 ± 0,1 cm, also pro Schritt etwa 6,250 ± 0,025 µm. Insgesamt beträgt der mögliche Fahrweg von der Mittelposition aus etwa ± 22.000 Schritte (13,8125 cm).

Angeschlossen werden die beiden Schrittmotoren über die Amphenol-Stecker an der Rückseite des Controllers (siehe Abb. 12).

Abbildung 12: Rückseite des Controllers (unten)

An der Vorderseite kann der Controller über die serielle RS-232-Schnittstelle mit dem PC verbunden werden (siehe Abb. 13, Kreis). Hier befinden sich auch die wichtigsten Schalter „Not-Aus“, „Start“, „Stop“ und „Reset“, mit denen die Bewegung der Motoren im Notfall gestoppt werden kann.

Abbildung 13: Vorderseite des Schrittmotor-Controllers

4.1.2 Die Rahmenkonstruktion

Die Konstruktion des Positionierungsrahmens auf Grundlage einer handgefertigten Skizze ist durch die hauseigene Werkstatt erfolgt. Durch die Verwendung von verschraubten Aluminium-Profilen kann der Rahmen ggf. leicht demontiert oder verändert werden. So sind auf der Unterseite z.B. bereits Winkelelemente vorgesehen, an die eine Platte zum Anbringen eines Stativs angeschraubt werden kann.

Besonders zu berücksichtigen bei dieser Konstruktion ist, dass die Bewegung der Kugelgewindevorschübe nicht linear in die Dreh- und Kippbewegung umgesetzt wird. In Abb. 14 ist die Geometrie der Drehbewegung dargestellt.

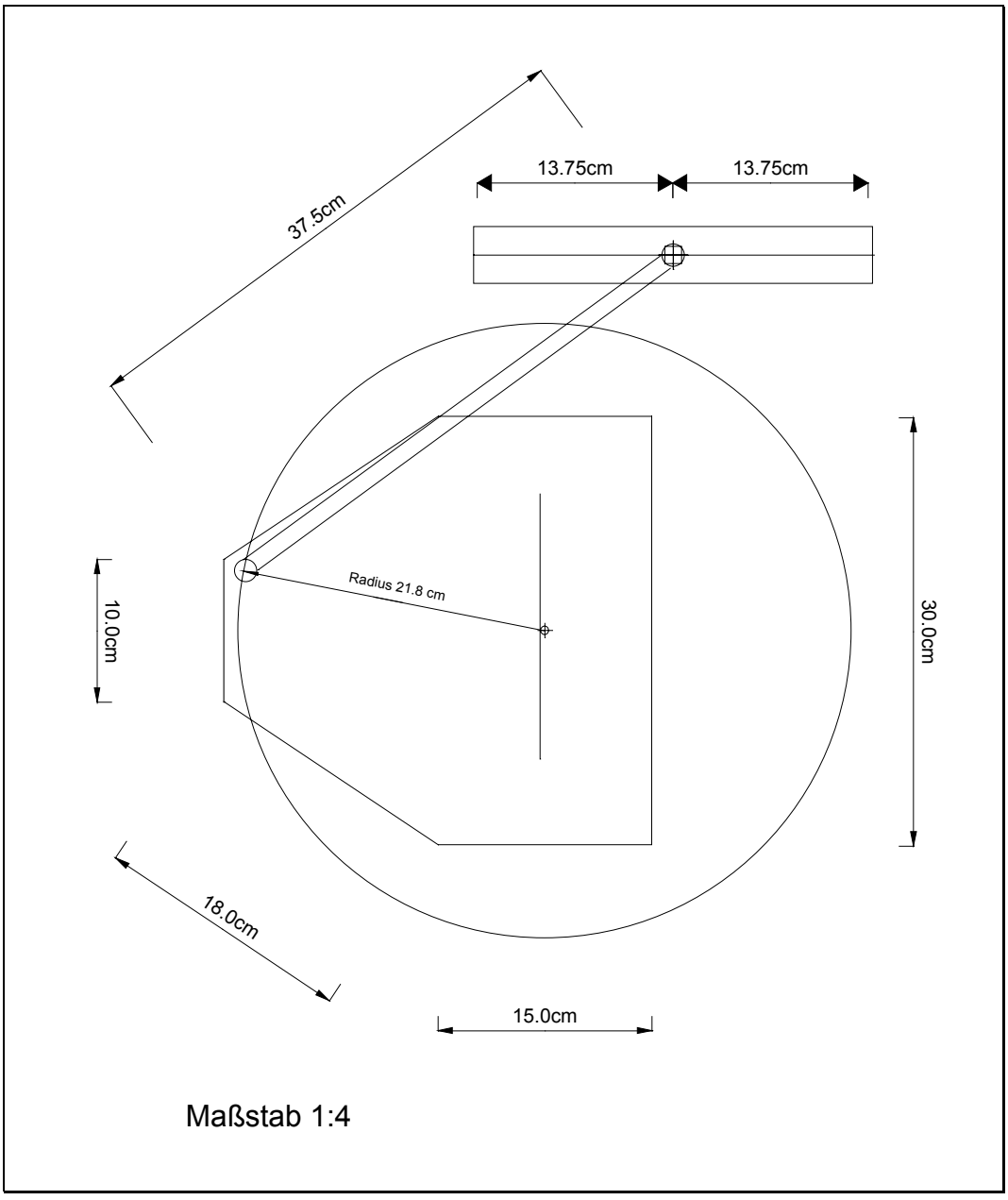


Abbildung 14: Schematische Darstellung der Geometrie der Drehbewegung, alle Maße mit Fehler ± 0,1 cm

Die relativ ungenaue Messung der einzelnen Abstände (Genauigkeit ± 1 mm) hätte zur Folge, dass die *Berechnung* des Drehwinkels in Abhängigkeit von der linearen Bewegung des Kugelgewindevorschubs einen (vielfach) höheren Fehler zur Folge hätte.

Aus diesem Grund wird die Abhängigkeit des Drehwinkels zur Anzahl der Schritte des Motors bestimmt, indem jeweils für eine definierte Anzahl von Schritten die Markierung des Lasers auf einer („weit“) entfernten Wand markiert und vermessen wird. Durch einen genügend großen Abstand des Lasers zur Wand können hier die Winkelpositionen einfach und wesentlich genauer über Dreiecksberechnungen erfasst werden. Dies soll nun erfolgen:

Der Drehpunkt des Lasers ist senkrecht zur gegenüberliegenden Wand in einem Abstand von 2293 ± 1 mm positioniert. Nachdem bereits bekannt ist, dass sich der Kugelgewindevorschub pro Schritt des Motors etwa 6,250 ± 0,025 µm bewegt, wird nun nach jeweils 500 Schritten in beide Richtungen die Position des Lasers an der Wand markiert und der Abstand zum Ausgangspunkt gemessen. Es ergibt sich die in Anhang 1 dargestellte Tabelle.

Trägt man nun die Anzahl der Schritte und die dazugehörigen Winkel übereinander auf, erhält man den in Abb. 15 dargestellten Zusammenhang.

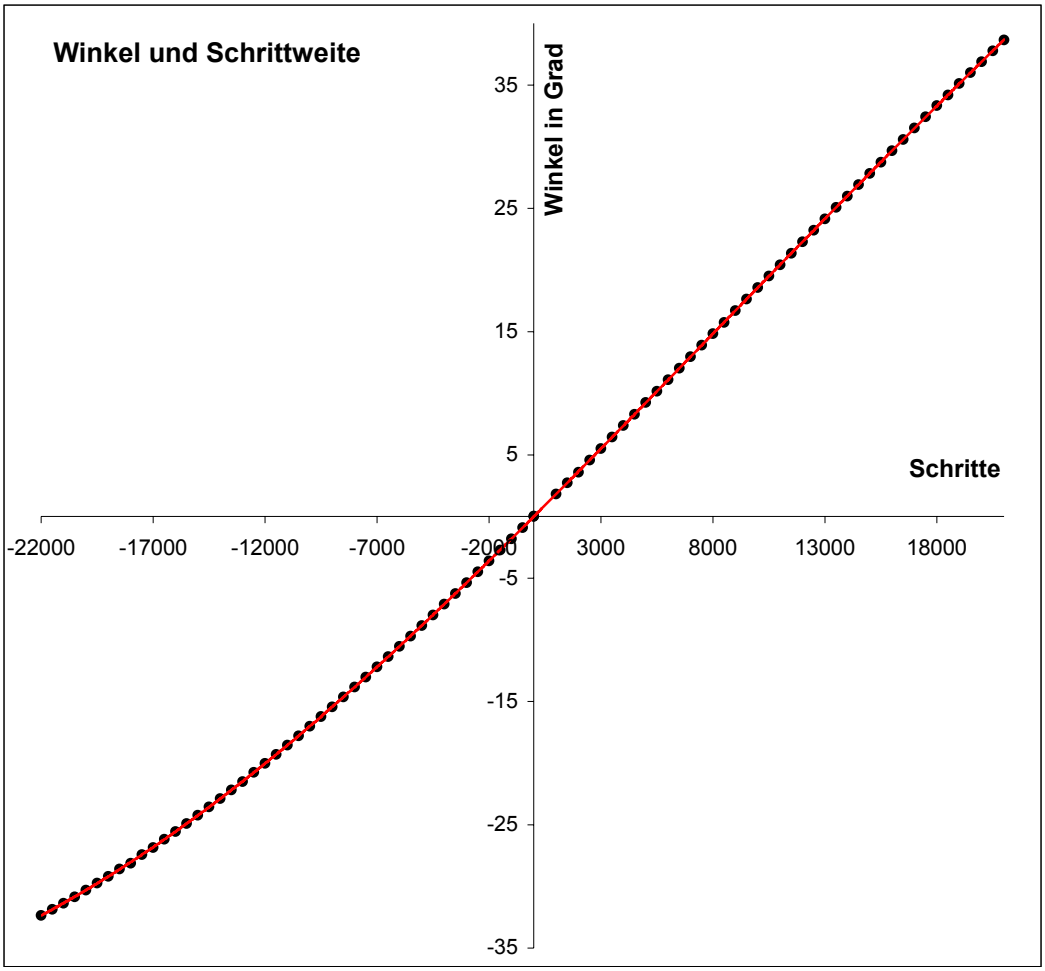


Abbildung 15: Graphische Darstellung der Abhängigkeit des Drehwinkels (x-Achsen-Bewegung des Messkopfes) zur Anzahl der Schritte

Die eingezeichnete Regression eines Polynoms fünften Grades hat die Funktionsvorschrift

$$y = 8{,}258140389714E\text{-}23 \cdot X^5 + 1{,}419963374671E\text{-}18 \cdot X^4 - 3{,}721212464506E\text{-}13 \cdot X^3 + 7{,}668007150107E\text{-}09 \cdot X^2 + 1{,}815288941525E\text{-}03 \cdot X$$

mit einem Korrelationskoeffizienten von $R^2 = 0{,}9999993862066$.

Diese Gleichung wird später in der Software zur Berechnung der Position benötigt.

Analog erhält man für die Bewegung in vertikaler Richtung mit jeweils 500 Schritten die ebenfalls in Anhang 1 aufgeführte Tabelle für den Kippwinkel zur Horizontalen. Trägt man hier ebenfalls den vertikalen Kippwinkel über die Anzahl der Schritte auf, ergibt sich das in Abb. 16 dargestellte Bild.

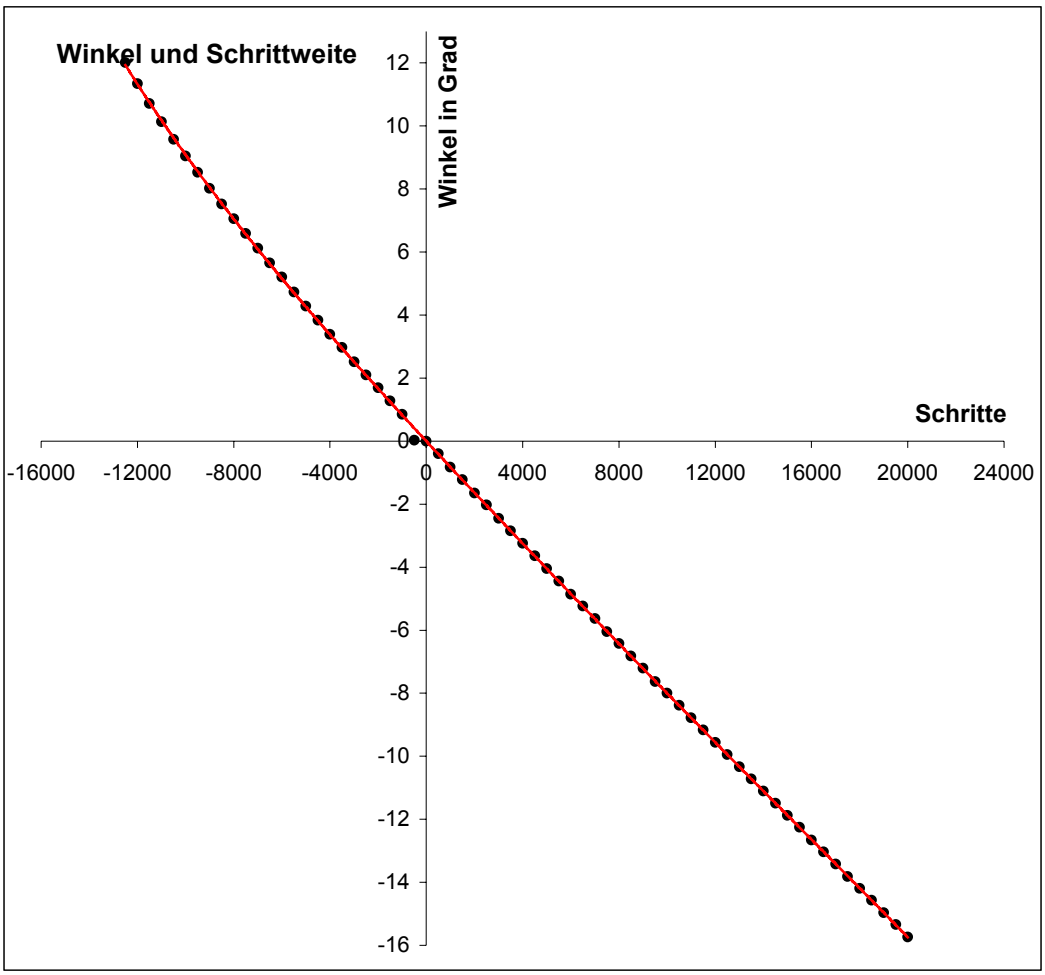


Abbildung 16: Graphische Darstellung der Abhängigkeit des vertikalen Kippwinkels (y-Achsen-Bewegung des Messkopfes) zur Anzahl der Schritte

Die eingezeichnete Regression eines Polynoms fünften Grades hat die Funktionsvorschrift

```
y = - 4,010580177957E-22*X^5 + 1,620329832712E-17*X^4
    - 2,620398895689E-13*X^3 + 3,815650172908E-09*X^2
    - 8,242669370255E-04*X
```

mit einem Korrelationskoeffizienten von $R^2 = 0{,}9999562245359$. Diese Gleichung wird später ebenfalls in der Software zur Berechnung der Position benötigt.

Die Genauigkeit der Positionierung der Kugelgewindevorschübe wird vom Hersteller mit ± 0,02 mm (≅ 3 Schritte) angegeben, was umgerechnet 0,0024° für die vertikale und 0,0054° für die horizontale Bewegung entspricht. Diese Fehler sind in den Tabellen (siehe Anhang 1) bereits berücksichtigt.

Die Bewegung in horizontaler Richtung kann also im Bereich von etwa ± 10 Grad und in vertikaler Richtung etwa im Bereich von +10 und -6 Grad mit der angegebenen Genauigkeit < 0,02° bestimmt werden.

Zum Vergleich: Die Winkelauflösung des Polytec PSV-400 Scanning Vibrometers liegt im Scanfeld (± 20°) bei einem Wert von unter 0,002°, also einen Faktor 10 kleiner. Berücksichtigt man hier jedoch noch die Punktstabilität von < 0,01° pro Stunde, die auf die Erwärmung der Scannereinheit zurückzuführen ist (Herstellerangabe), ergibt sich ein Gesamtfehler beim Polytec Scanning Vibrometer von etwa ± 0,012°. Damit ist die Positionierungsgenauigkeit des Polytec-Systems etwa doppelt so hoch.

4.1.3 Richtiges Positionieren des Messkopfes

Ein Eigenart des Scanning-Laser-Vibrometers bezüglich der Positionierung des Messkopfes muss an dieser Stelle noch aufgeführt werden.

Die Kugelgewindevorschübe haben die Eigenschaft, bei einem „Richtungswechsel“ des Fahrweges etwa 20 Schritte im Umkehrpunkt zu „verlieren“. Dies macht sich z.B. dann bemerkbar, wenn der Laserstrahl des Messkopfes aus der Ausgangsposition eine definierte Anzahl von Schritten in die eine Richtung und anschließend die gleiche Anzahl der Schritte in die Gegenrichtung gefahren wird. Hier ergibt sich eine Differenz von etwa 0,0686° zum Ausgangspunkt.

Beim Abscannen einer Fläche lässt sich dieser Fehler nun vermeiden, indem der Messkopf, nachdem die erste Zeile (von links nach rechts) durchfahren wurde, nicht direkt zum ersten Punkt der zweiten Zeile zurückgefahren, sondern ein Stück (> 20 Schritte) über diesen Punkt hinaus gefahren wird, um sich ihm dann aus der anderen Richtung zu nähern (siehe Abb. 17).

Dadurch, dass nun zwei Mal bei entgegengesetzten Fahrtwegen in den Umkehrpunkten „Schritte verloren“ gehen, gleicht sich dieser Fehler genau aus und man kann mit der weiter oben aufgeführten Genauigkeit rechnen.

Durch mehrere Testfahrten mit unterschiedlichen Schrittweiten wurde festgestellt, dass es sich hierbei tatsächlich um einen Fehler in den Kugelgewindevorschüben bzw. in der Mechanik handelt, da der „Verlust“ von etwa 20 Schritten unabhängig von der Anzahl der Gesamtschritte und der Schrittmotorgeschwindigkeit ist, es also immer etwa 20 Schritte nach einem Umkehrpunkt „dauert“, bis die gewünschte „wirkliche“ Schrittweite ausgeführt wird.

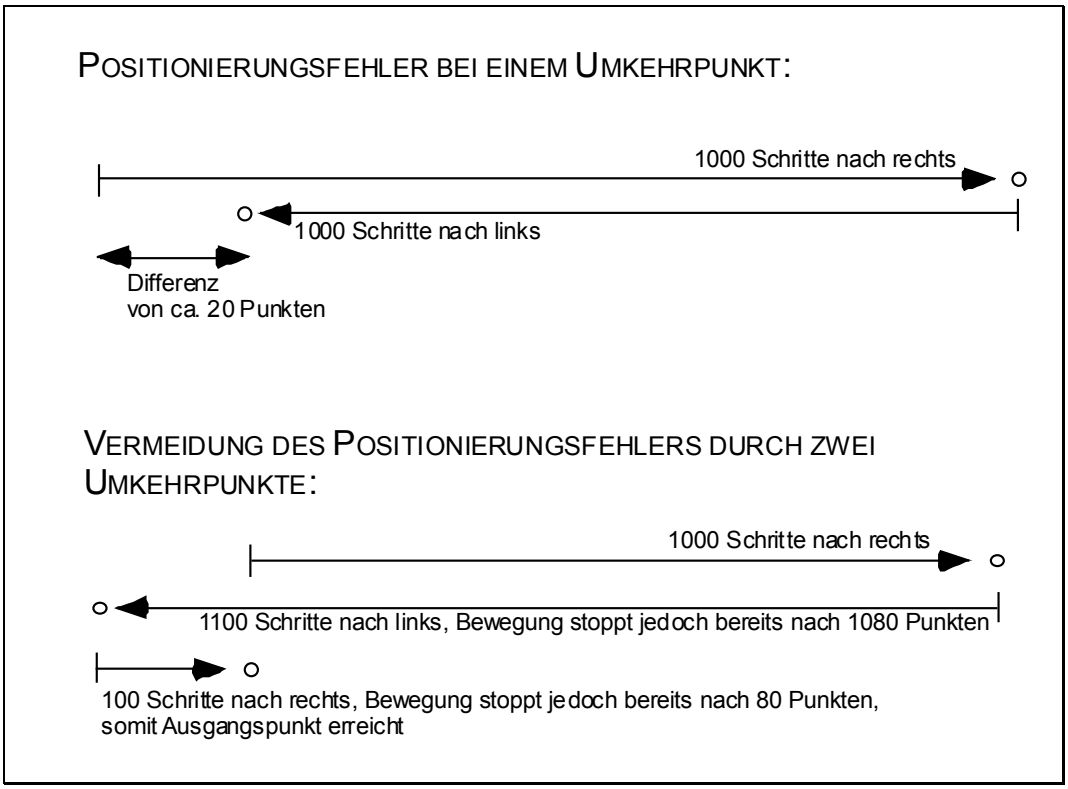


Abbildung 17: Vermeidung des Positionierungsfehlers der Kugelgewindevorschübe

Dies ist auch der Grund dafür, dass durch die in Abb. 17 dargestellte Verfahrensweise beim Scannen der Postitionierungsfehler unberücksichtigt bleiben kann, da er vermieden wird.

4.2 Entwicklung der Software

Um eine für Änderungen möglichst flexible und durchschaubare Software zur Steuerung und Auswertung des Scanning-Laser-Vibrometers zu entwickeln, wird das Programm Matlab in der Version 6.1 (Release 12.1) der Firma „The MathWorks“ als Grundlage genommen. Es bietet zum einen die Möglichkeit, mit wenig Aufwand eine graphische Oberfläche zu erstellen, der modulare Aufbau mit

Funktionen ermöglicht es zum anderen, unkompliziert Änderungen an einzelnen Programmteilen durchzuführen.
Die Entwicklung der Software ist so aufgebaut, dass zunächst die Ansteuerung der Hardware erprobt wird, danach werden die ersten Signale des Vibrometers über die Soundkarte des PCs als Wave-File eingelesen. Im nächsten Schritt wird ein Programm zur Kalibrierung der Soundkarte erstellt, woraufhin die Programme zur Positionierung und zum Scannen fertiggestellt werden. Im letzten Abschnitt „Postprocessing“ wird beschrieben, wie die gewonnenen Daten präsentiert und dargestellt werden können. In Anhang 2 ist eine Übersicht über die Programmabläufe dargestellt.
Im Folgenden werden vereinzelt zentrale Funktionen und Routinen vorgestellt und beschrieben, die komplette Software mit allen Programmteilen ist im Anhang aufgeführt und liegt ebenfalls auf CD-ROM als Bestandteil dieser Arbeit bei.

4.2.1 Ansteuerung der Hardware

Die Hardware des Scanning-Laser-Vibrometers teilt sich in den Schrittmotor-controller und den Controller Processor OFW-3001. Beide Geräte werden über eine serielle RS-232-Schnittstelle an den PC angeschlossen und können mittels einfacher ASCII-Befehle angesteuert werden.

4.2.1.1 Öffnen und schließen einer seriellen Schnittstelle

In Abb. 18 ist ein Programm dargestellt, wie z.B. der COM1-Port für den Controller Processor initialisiert und geöffnet wird. Im ersten Schritt wird ein serieller Port als Objekt definiert und in die Variable `s` geschrieben. Im zweiten Schritt erfolgt das Setzen der Parameter, welche der Bedienungsanleitung zu entnehmen sind (vgl. [Poly97]). Daraufhin wird mit dem Befehl `fopen(s)` die serielle Schnittstelle geöffnet und mit `fprintf(s,'ECHOON')` der ASCII-Befehl `ECHOON` an den Controller gesendet. Hiermit wird bezweckt, dass der

Controller nach Empfang eines Befehls eine Rückmeldung sendet. Diese wird daraufhin in die Variable `echo` mit dem Befehl `fscanf(s)` geschrieben. Mit den letzten drei Programmzeilen wird die serielle Schnittstelle wieder geschlossen.

```
% ----------------------------------------------------------------
% COM1 Port wird initialisiert
% ----------------------------------------------------------------
s   =  serial('COM1');
set(s,'Baudrate',9600,'Parity','none','StopBits',1);
set(s,'InputBufferSize',1024,'Terminator','LF','Timeout',1);

% ----------------------------------------------------------------
% Öffnen der seriellen Schnittstelle COM1
% ----------------------------------------------------------------
fopen(s);

% ----------------------------------------------------------------
% Senden der Einstellungen an das Vibrometer
% ----------------------------------------------------------------
fprintf(s,'ECHOON');
echo = fscanf(s);
fprintf(s, 'ASCII-Befehl')

% ----------------------------------------------------------------
% COM1 wird geschlossen
% ----------------------------------------------------------------
fclose(s)
delete(s)
clear s
```

Abbildung 18: Programm zum Öffnen und Schließen einer seriellen Schnittstelle

Dieses Beispiel für das Öffnen und Schließen einer seriellen Schnittstelle wird in allen Funktionen verwendet, die Einstellungen und Befehle an die angeschlossenen Controller senden oder von ihnen empfangen müssen.

4.2.1.2 Bewegen des Messkopfes

Als nächstes soll auf die Steuerung des Schrittmotorcontrollers detaillierter eingegangen werden.

Als Beispiel für die Bewegung des Messkopfes wird hier die Funktion `bewegung_rechts` beschrieben (siehe Abb. 19):

```
function bewegung_rechts

% ------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------
global  schritte...
        v_x...
        summe_schritte_rechts


% ------------------------------------------------------------------
% Bewegungsvariable erzeugen
% ------------------------------------------------------------------
bewegung_r = ['@0A'...                % Bewegung initialisieren
              int2str(schritte)...    % Schritte auf der x-Achse
              ','...                  % (jeweils als "String")
              int2str(v_x)...         % Geschwindigkeit x-Achse
              ','...
              int2str(0)...           % 1. Bewegung y-Achse
              ','...
              int2str(v_x)...         % Geschwindigkeit y-Achse
              ','...
              int2str(0)...           % 2. Bewegung y-Achse
              ','...
              int2str(v_x)]           % Geschwindigkeit y-Achse


% ------------------------------------------------------------------
% Die aktuell ausgeführte Schrittzahl nach oben wird in einer
% Variable gespeichert
% ------------------------------------------------------------------
summe_schritte_rechts = summe_schritte_rechts + schritte

% ------------------------------------------------------------------
% COM2 Port initialisieren
% ------------------------------------------------------------------
s2 = serial('COM2');
set(s2,...
    'Baudrate',9600,...
    'Parity','none',...
    'StopBits',1,...
    'Terminator','CR/LF',...
    'Timeout',1);
```

```
% ------------------------------------------------------------
% Öffnen des seriellen Ports COM2
% ------------------------------------------------------------
fopen(s2);


% ------------------------------------------------------------
% Achse initialisieren
% ------------------------------------------------------------
fprintf(s2,'@05');
fprintf(s2,'@0C1');


% ------------------------------------------------------------
% Befehl zum Bewegen setzen
% ------------------------------------------------------------
fprintf(s2,bewegung_r);


% ------------------------------------------------------------
% Schließen des seriellen Ports COM2
% ------------------------------------------------------------
fclose(s2)
delete(s2)
clear s2
```

Abbildung 19: Programm zum Bewegen des Messkopfes nach rechts

Als globale Variablen werden zu Beginn dieser Funktion die Anzahl der Schritte `schritte`, die Bewegungsgeschwindigkeit `v_x`, sowie die Gesamtzahl der Schritte `summe_schritte_rechts` definiert. Die ersten beiden können in der graphischen Oberfläche in sogenannte „Edit“-Felder eingegeben werden und sind zuvor von dort ausgelesen und ihre Werte in die Variablen gespeichert worden.

Die letzte der drei Variablen dient der Erfassung der Gesamtzahl der Schritte in die betreffende Richtung. Diese Gesamtzahl wird später benötigt, um die Position des Messkopfes relativ zum Ausgangspunkt zu berechnen, damit der Kopf zu Beginn des Scanvorgangs an die Startposition zurückgefahren werden kann.

Da der Controller zur Steuerung ASCII-Befehle verwendet, muss zunächst ein String für die Bewegung erzeugt werden. Dies geschieht im nächsten Schritt: Ein Befehl zum Bewegen der Schrittmotoren ist immer nach dem Schema

$$@0A\ S_x,\ G_x,\ S_{z1},\ G_{z1},\ S_{z2},\ G_{z2}$$

aufgebaut, wobei die Prozessorkarte mit `@0` adressiert wird und das „`A`“ angibt, dass eine Bewegung erfolgen soll. Diese besteht immer aus dem Weg `S` und der Geschwindigkeit `G`, wobei für die z-Achse (bei der Beschreibung der Fahrtwege des Vibrometers ist dies die y-Achse) zwei Zahlenpaare S_{z1}, G_{z1}, S_{z2}, G_{z2} anzugeben sind. Dies hat den Ursprung, dass bei Steuerungen im CNC-Bereich oftmals für die vertikale Achse nach einem Befehl „Senken“ der Befehl „Anheben“ folgt und dieser so im Anschluss direkt ausgeführt werden kann. Bei der Verwendung als Steuerung für den Messkopf wird im Folgenden S_{z1} immer auf null gesetzt, eine Geschwindigkeit im zulässigen Bereich zwischen 30 und 10.000 muss jedoch immer für alle Strecken `S` gesetzt werden.
Nachdem nun also der String `bewegung_r` erzeugt wurde, kann dieser an den Controller auf die bereits oben beschriebene Weise gesendet werden.

Durch die Befehle `fprintf(s2,'@05');` und `fprintf(s2,'@0C1');` wird dem Schrittmotorcontroller zuvor angekündigt, dass nur zwei Achsen (x- und z-Achse) verwendet werden sollen, und außerdem, dass nach jedem gesendeten Befehl ein CR/LF gesetzt werden soll.

4.2.1.3 Ansteuerung des Controllers - Fokussieren des Lasers

Nachdem die Bewegung des Messkopfes anhand eines Beispiels beschrieben wurde, soll nun die Ansteuerung des Vibrometer Controllers und das Auslesen bzw. Setzen der wichtigsten Parameter am Beispiel der Funktion `manuellfocus` erläutert werden.

Diese Funktion kann einerseits manuell (daher der Name) über die Programmoberfläche ausgelöst und zum anderen beim Scannen aktiviert werden, so dass der Laserstrahl vor dem Einlesen der Messdaten fokussiert wird.
Die Programmierung einer solchen Funktion war unvermeidbar, da der Messkopf des Laser-Vibrometers nicht über einen eingebauten Autofokus verfügt. Die in Abb. 20 dargestellte Funktion soll nun im einzelnen beschrieben werden:

```
function manuellfocus

% ---------------------------------------------------------------
% Benötigte globale Variablen
% ---------------------------------------------------------------
global signallevel

% ---------------------------------------------------------------
% Voreinstellungen der "empirisch" gefundenen Werte ttrechts
% und ttlinks, zaehler wird auf null gesetzt
% ---------------------------------------------------------------
zaehler      = 0;
ttrechts     = 6.7;
ttlinks      = 9.6;

% ---------------------------------------------------------------
% COM1 Port initialisieren
% ---------------------------------------------------------------
s = serial('COM1');
set(s,'Baudrate',9600,'Parity','none','StopBits',1);
set(s,'InputBufferSize',1024,'Terminator','LF','Timeout',1);

% ---------------------------------------------------------------
% Öffnen der seriellen Schnittstelle COM1
% ---------------------------------------------------------------
fopen(s);

% ---------------------------------------------------------------
% Auslesen der Ausgangssignalstärke "signallevela"
% ---------------------------------------------------------------
fprintf(s,'ECHOON');
echo = fscanf(s);


fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);
if isempty(level)
    warndlg...
    ('Konnte den Befehl "LEV" nicht an das Vibrometer senden','!!
Warnung !!')
end
```

```
fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);

for x = 0:40
    levelzahl = ['LEV',int2str(x)];
    if strcmp(level,levelzahl) == 1
    signallevela = x;
    end
end

% ------------------------------------------------------------------
% A U T O F O K U S
% ------------------------------------------------------------------
signallevel = signallevela;

    while (signallevel == 0) & (zaehler <= 4)
           ttrechts = 1.5 * ttrechts
           ttlinks  = 1.5 * ttlinks
           zaehler = zaehler + 1;

           fprintf(s,'r');
              for ta = 0:ttrechts
                 fprintf(s,'LEV');
                 level = fscanf(s);
                 level = cellstr(level);

                    for xa = 0:40
                       levelzahl = ['LEV',int2str(xa)];
                         if strcmp(level,levelzahl) == 1
                         signallevel = xa;
                         end
                    end

                    if signallevel > signallevela
                    fprintf(s,'N');
                    break
                    end


            fprintf(s,'l');
               for tb = 0:ttlinks
                  fprintf(s,'LEV');
                  level = fscanf(s);
                  level = cellstr(level);

                     for xb = 0:40
                        levelzahl = ['LEV',int2str(xb)];
                           if strcmp(level,levelzahl) == 1
                           signallevel = xb;
                           end

                     end

                     if signallevel > signallevela
                     fprintf(s,'N');
                     break
                     end
```

```
            end
    end

% --------------------------------------------------------------------
% Stop-Befehl an Motor senden
% --------------------------------------------------------------------
fprintf(s,'N');

% --------------------------------------------------------------------
% Signalstärke in Feld schreiben
% --------------------------------------------------------------------
handles = guihandles(gcbo);
set(handles.text81,'String',num2str(signallevel));

% --------------------------------------------------------------------
% Serielle Schnittstelle schließen
% --------------------------------------------------------------------
fclose(s)
delete(s)
clear s
```

Abbildung 20: Programm zum Fokussieren des Laserstrahls

Nachdem hier zunächst die globale Variable `signallevel` definiert wird, erfolgt das Zuweisen zweier „empirisch“ gefundener Werte in die Variablen `ttrechts` und `ttlinks`. Diese werden weiter unten detaillierter beschrieben.

Im folgenden Schritt wird nun nach Öffnen der seriellen Schnittstelle einmalig die aktuelle Signalstärke ausgelesen. Diese wird mit dem Befehl `LEV` angefordert und in die Variable `level` geschrieben. Der Wert der Signalstärke liegt im Bereich von 0 bis 40, der Controller gibt jedoch Strings der Form `LEV0` bis `LEV40` zurück - aus diesem Grund wird der empfangene String in der folgenden Schleife mit dem hier generierten String `LEV0` bis `LEV40` so lange verglichen, bis eine wahre Aussage gegeben ist (`if strcmp(level,levelzahl) == 1`) und dann der Wert der Schleife in die Variable `signallevela` geschrieben. Dies ist die „Ausgangssignalstärke“, mit der die beim Fokussieren erzielten Signalstärken verglichen werden können.

Der Autofokus selber wird nun im nächsten Schritt der Funktion formuliert: Zu Beginn steht hier eine `while`-Schleife, die den Autofokus nur auslöst, wenn die Ausgangssignalstärke gleich null ist. Dies kann an dieser Stelle noch variiert werden, etwa dass die Signalstärke immer ≥ 1 sein muss o.ä.. Insgesamt wird die Fokussierungsschleife maximal drei Mal durchlaufen.

Eine Eigenschaft des Fokussierungsmotors im Messkopf ist, dass dieser nicht mit einer definierten Anzahl von Schritten bewegt werden kann. Hier ist ein Trick erforderlich, bei dem die beiden o.a. Variablen `ttrechts` und `ttlinks` die zentrale Rolle spielen. Sobald der Motor für die Fokussierung mit dem Befehl `fprintf(s,'r')` in Bewegung (genauer: in eine langsame Rechtsbewegung) gesetzt wurde, lässt er sich nur noch durch den Befehl `fprintf(s,'N')` stoppen.

Um nun den Motor eine *bestimmte* Zeit in diese Richtung laufen zu lassen, wird in einer Schleife immer wieder die Signalstärke ausgelesen und mit dem Ausgangswert verglichen. Dafür benötigt der PC eine gewisse Zeit, die der Motor nun in die betreffende Richtung läuft. Wird bei diesem Fokussierungsvorgang irgendwann eine Signalstärke ungleich null ausgelesen, stoppt der Motor und die Schleife ist beendet. Die Anzahl der Schleifendurchläufe ist dabei gerade der Wert der Variable `ttrechts`.

Wird beim Rechtslaufen des Motors keine höhere Signalstärke erreicht, erfolgt eine Linksbewegung mit der Schleifenlänge `ttlinks`. Dieser Wert ist etwas höher, da der Motor schneller nach links läuft als nach rechts. Um hier nun die gleiche Bewegung zu erhalten wie beim Rechtslauf, wurden die Werte für die Schleifenlängen wie oben beschrieben „empirisch“ ermittelt.

Sollte nun nach einem Rechts- und Linkslauf des Fokussierungsmotors noch immer keine Signalstärke ungleich null vorliegen, wird die Fokussierungsschleife erneut durchlaufen, allerdings mit 1,5fach höheren Werten für die Variablen `ttrechts` und `ttlinks`, damit so ein größerer Brennweitenbereich mit einbezogen wird.

Nachdem die Fokussierungsschleife maximal drei Mal durchlaufen wurde und die Signalstärke noch immer gleich null ist, endet sie und der Motor wird gestoppt. Im Anschluss wird die aktuelle Signalstärke noch in das entsprechende Textfeld der Programmoberfläche geschrieben.

4.2.1.4 Speichern eines Messsignals

Ein letztes Programmbeispiel für das Ansteuern der Hardware betrifft das Auslesen des Schnelle-Signals am Controller Processor des Laser-Vibrometers. Dieses Signal wird an den LineIn-Eingang der Soundkarte übermittelt und von hier aus über die Befehle `wavrecord` und `wavwrite` in das Matlab-Programmverzeichnis als Wave-File geschrieben (siehe Abb. 21).

```
function einlesen_ep

% ---------------------------------------------------------------
% Benötigte globale Variablen
% ---------------------------------------------------------------
global  dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs

% ---------------------------------------------------------------
% Samplingfrequenz setzen, allg. Wert der Soundkarte
% ---------------------------------------------------------------
Fs = 44100;

% ---------------------------------------------------------------
% Einlesen des Einpunkt-Signals mit der Länge
% (signallaenge_einpunkt*Fs) in
% Sekunden und der Samplingfrequenz Fs = 44100 Hz
% ---------------------------------------------------------------
einpunktsignal = wavrecord(signallaenge_einpunkt*Fs,Fs);
```

```
% ------------------------------------------------------------
% Schreiben des Einpunkt-Signals ins Matlab-Verzeichnis
% ------------------------------------------------------------
wavwrite(einpunktsignal,Fs,dateiname_einpunkt);
```

Abbildung 21: Programm zum Einlesen des Schnelle-Signals

Die Samplingfrequenz wird mit der Variable `Fs` auf den Wert 44.100 gesetzt, was der Samplingrate der Soundkarte entspricht. Die Signallänge in Sekunden wurde zuvor aus dem entsprechenden Edit-Feld in der Programmoberfläche ausgelesen und wird hier mit `Fs` multipliziert, was ein entsprechend langes Signal erzeugt. Dieses Signal wird anschließend mit dem Dateinamen `dateiname_einpunkt` in das Matlab-Programmverzeichnis geschrieben.

Das hier aufgeführte Programmbeispiel dient dem Einlesen eines einzigen Messpunktes - daher die gewählte Bezeichnung „Einpunktsignal“.

4.2.2 Verarbeitung der eingelesenen Signale

Nachdem das eingelesene Schnellesignal als Wave-File in das Matlab-Programmverzeichnis geschrieben wurden, muss es nun gelesen und ausgewertet werden. Das Ziel ist hier, das Signal über eine diskrete Fast Fourier Transformation (FFT) spektral zu zerlegen und die Schnelle über die Frequenz darzustellen.
Matlab bietet eine einfache Möglichkeit, mit Hilfe des `wavread`-Befehls Wave-Files einzulesen, allerdings werden hierbei die Werte der Amplitude in das Intervall [-1,1] geschrieben, so dass die tatsächlichen Amplituden nur schwer und mit einer geringeren Genauigkeit rekonstruiert werden können.
Aus diesem Grund werden die Wave-Files im Folgenden als binäre Datei eingelesen und in eine Variable geschrieben. Dazu ist es notwendig zu wissen, dass bei einem Wave-File die eigentlichen Daten ab dem 44. Byte zu finden sind,

die Bytes zuvor enthalten z.B. Informationen zum Dateiformat, zur Anzahl der Kanäle und zur Samplingrate. In Abb. 22 ist eine Funktion dargestellt, die ein Wave-File zunächst binär einliest und die so erhaltenen Daten über eine FFT spektral darstellt.

```
function einpunkt_berechnen

% ------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------
global  dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs...
        xlinks...
        xrechts...
        ymax...
        effektivspannung...
        kalibmax...
        kalibmin...
        peaktopeak...
        messb_schnelle...
        messb_ausl...
        filter...
        st_filter...
        eff_sig

handles = guihandles(gcbo);


% ------------------------------------------------------------------
% Einlesen des wave-Files
% ------------------------------------------------------------------
fid = fopen(dateiname_einpunkt, 'r');  % Handle erzeugen
fseek(fid, 22, 'bof');                 % überspringe 22 Byte
numchannels = fread(fid, 1, 'int16');  % lese numchannels


samplerate  = fread(fid, 1, 'int32');  % lese samplerate
fseek(fid, 44, 'bof');                 % springe zum 44. Byte
einpunktsignal = fread(fid, inf, 'int16');  % lese messsignal
fclose(fid);                                % fid schließen


% ------------------------------------------------------------------
% Bestimmen der Länge des Eingangssignals
% ------------------------------------------------------------------
N = length(einpunktsignal);
```

```
% ---------------------------------------------------------------
% Berechnen der FFT
% ---------------------------------------------------------------
Y    = fft(einpunktsignal,N);

% ---------------------------------------------------------------
% Berechnung Skalierungsfaktors aus den Variablen der
% Kalibrierung
% ---------------------------------------------------------------
skalierungsfaktor = effektivspannung/Kali_Pyy

% ---------------------------------------------------------------
% Umrechnen in Schnelle über Messbereich des Vibrometers
% ---------------------------------------------------------------
if messb_schnelle == 6
    y_range = 5;                  % 5 mm/s/V
elseif messb_schnelle == 7
    y_range = 25;                 % 25 mm/s/V
elseif messb_schnelle == 8
    y_range = 125;                % 125 mm/s/V
elseif messb_schnelle == 9
    y_range = 1000;               % 1000 mm/s/V
end

% ---------------------------------------------------------------
% Darstellen als Funktion von x=0 bis x=(Samplingfrequenz/2) und
% Skalierung der y-Achse auf Grundlage der
% Vibrometereinstellungen
% ---------------------------------------------------------------
a    = N/2;               % halbe Signallänge
b    = a+1;               % halbe Signallänge + 1
f = 44100*(0:a)/N;        % x-Achse
Ay  = abs(Y);             % entspr. Ay = sqrt(Y .* conj(Y));
Ay  = 2.*Ay./N./sqrt(2); % 'normalized discrete fourier transf.,
                          % siehe Matlab Hilfe
% Multipl. mit Skalierungsfaktor und
% Einstellung des Vibrometers
Ay  =  y_range*skalierungsfaktor*Ay;
plot(f,(Ay(1:b)),'-')     % Plot des Amplitudenspektrums (wegen
                          % Symmetrie nur eine Hälfte)
xlim([xlinks xrechts]);   % Grenzen, Einzustellen über Slider
ylim([0 ymax]);           % -"-
title('Amplitudenspektrum des Signals')         % Titel usw...
xlabel('Frequenz [Hz]')
ylabel('Schnelle [mm/s]')
drawnow

if  ymax < y_range        % Umschaltung des y-Bereichs nur
    set(handles.slider3,'Max',2*y_range);  % zulässig, wenn
                                           % gewählter Bereich
end;
```

Abbildung 22: Programm zum Einlesen eines Wave-Files und zur Berechnung des Amplitudenspektrums

Nachdem in diesem Programm zunächst die benötigten globalen Variablen definiert wurden, wird das Wave-File mit dem Befehl `fread` ausgelesen. Die Daten ab dem 44. Byte werden dabei in die Variable `einpunktsignal` geschrieben, dessen Länge anschließend mit `N` bestimmt wird. Im nächsten Schritt erfolgt die Berechnung der diskreten N-Punkt FFT mit der zuvor bestimmten Länge N des Signals. Da die Signallänge variieren kann, ändert sich hiermit auch die Frequenzauflösung – bei einer Signallänge von einer Sekunde erhält man eine Auflösung von einem Hertz.

Um nun im nächsten Schritt die Amplitude in Volt und somit eine Darstellung der Schnelle auf Grund des gewählten Messbereiches in mm/s angeben zu können, muss ein Skalierungsfaktor bestimmt werden, der für jede Soundkarte erneut zu berechnen ist. Diese Kalibrierung der Soundkarte wird im anschließenden Abschnitt beschrieben, an dieser Stelle soll der Skalierungsfaktor als gegeben angenommen werden.

Nachdem als nächstes der Messbereich des Vibrometers ausgelesen wurde, wird das Amplitudenspektrum `Ay = abs(Y)` noch mit einem Faktor 2/N multipliziert, welcher eine Normalisierung der diskreten FFT zur Folge hat. Nähere Angaben hierzu liefert die Matlab-Programmhilfe im Kapitel „Signal Processing Toolbox: fft“.

Anschließend erfolgt eine Multiplikation mit dem Skalierungsfaktor und dem Messbereich, so dass nun das Spektrum über die Frequenz im Bereich 0 bis 22050 Hz aufgetragen und in der Programmoberfläche geplottet werden kann.

4.2.3 Kalibrieren der Soundkarte

Im vorherigen Abschnitt wurde bereits beschrieben, wie mit Hilfe der FFT das Amplitudenspektrum des eingelesenen Wave-Files berechnet wurde. Um dieses letztlich in der Einheit mm/s darstellen zu können, ist eine Kalibrierung der Soundkarte notwendig.

Das Schnelle-Signal des Vibrometers Controllers wird über den LineIn-Eingang der Soundkarte in den PC eingelesen. Für eine Kalibrierung ist es nun erforderlich, dass ein Signal mit bekannter Amplitude bzw. mit einer bekannten Effektivspannung eingelesen und als Wave-File abgespeichert wird. Das Verhältnis der effektiven Spannung des Eingangssignals zum berechneten Effektivwert des eingelesenen Wave-Files entspricht dann dem Skalierungsfaktor. Dieser hat die Dimension V/Punkte, wobei die Soundkarte mit 16 Bit maximal ± 32.786 Punkte auflösen kann.

Das Kalibrierungsprogramm `kalibrierung.m` (siehe Anhang) bietet eine komfortable Möglichkeit, den gesuchten Faktor zu finden. Hier ist schrittweise erklärt, wie zunächst ein geeignetes Signal mit Hilfe eines Frequenzgenerators erzeugt und eingelesen werden kann. Dazu wird zunächst die Effektivspannung des Signals mit einem Multimeter gemessen und in das entsprechende Feld eingetragen (siehe Abb. 23).

Mit einem Pushbutton können im nächsten Schritt fünf Signale mit einer Länge von jeweils fünf Sekunden eingelesen und abgespeichert werden. Über einen weiteren Button werden diese Signale anschließend geladen und so verschoben, so dass die maximale Amplitude gleich der minimalen Amplitude ist. Hiermit wird der vorhandene Offset der Soundkarte eliminiert. Zur Kontrolle werden nun die ersten 1.000 Punkte des ersten Signals in einem Plot dargestellt und der maximale und minimale Wert der Amplituden ausgegeben.

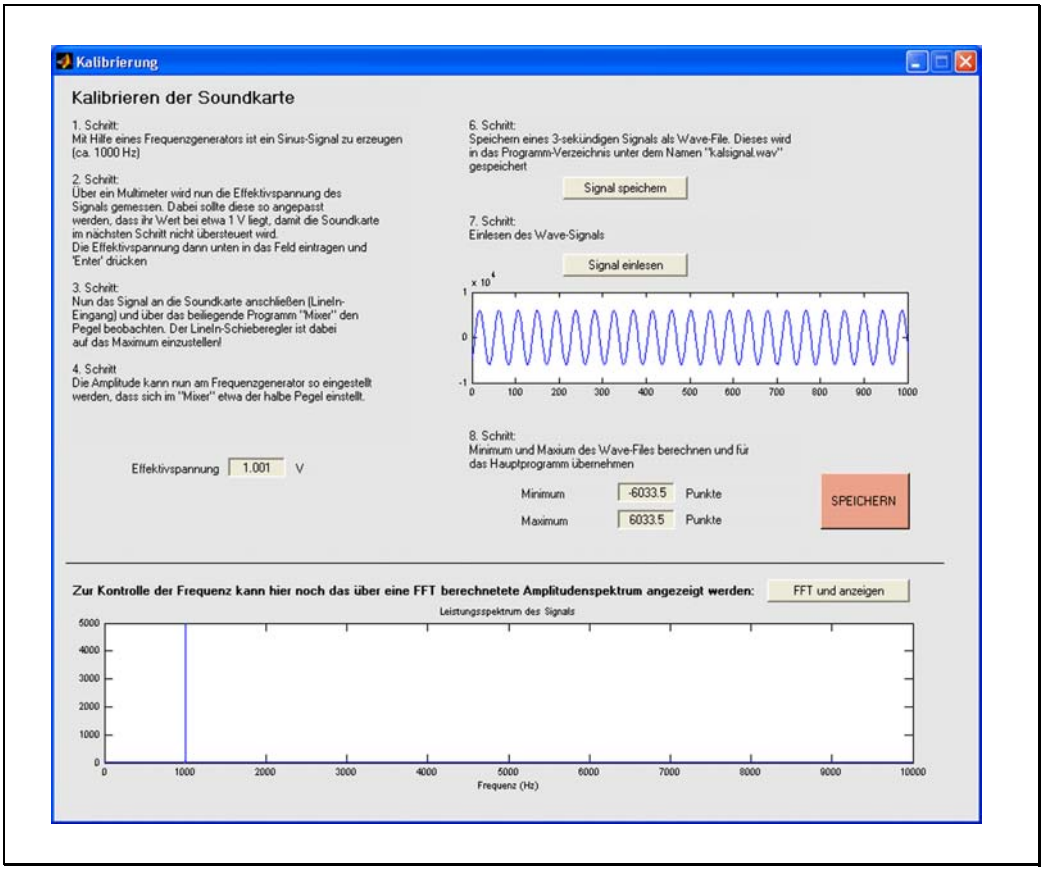


Abbildung 23: Screenshot des Programms zur Kalibrierung der Soundkarte

In einem weiteren Schritt werden die einzelnen FFTs und Effektivwerte der fünf Signale berechnet. Hierbei wird das arithmetische Mittel der fünf Effektivwerte `Kali_Pyy_#` (siehe Programm in Anhang 3) zur Kalibrierung verwendet. Mit der eingegebenen Effektivspannung zusammen erhält man den für die Soundkarte charakteristischen Quotienten bzw. Skalierungsfaktor (`effektivspannung / Kali_Pyy`).

Mit einem letzten Klick auf den Button „Speichern“ werden die wichtigsten Variablen `effektivspannung` und `Kali_Pyy` im File `kalibrierung.mat` gespeichert, von wo aus sie später beim Aufruf des Positionierungsprogramms automatisch geladen werden.

Im fünften Kapitel dieser Arbeit wird bei der Durchführung einer Messung noch einmal genauer auf das Kalibrierungsprogramm eingegangen und in diesem Zusammenhang werden auch konkrete Werte für die verwendete Soundkarte ermittelt.

4.2.4 Das SLDV[1]-Positionierungs-Programm unter Matlab

Die eigentlichen Hauptprogramme zum Positionieren und Einlesen der Messpunkte einer schwingenden Oberfläche sind das SLDV-Positionierungs- und das SLDV-Scan-Programm. Mit ersterem werden im Wesentlichen zunächst die Eckpunkte der zu scannenden Fläche bestimmt und der Abstand zum Messobjekt eingegeben (siehe Abb. 24).

Weiterhin können mit diesem Programm einzelne Messpunkte eingelesen werden, um abzuschätzen, welcher Messbereich am Vibrometer gewählt werden sollte. Diese Messbereiche können hier dann ebenfalls eingestellt und an das Vibrometer gesendet werden.

Das SLDV-Positionierungs-Programm ist so aufgebaut, dass die erforderlichen Schritte zur Vermessung einer Oberfläche praktisch in der sichtbaren Reihenfolge von oben links nach unten rechts abgearbeitet werden können.

[1] Scanning Laser Doppler Vibrometer

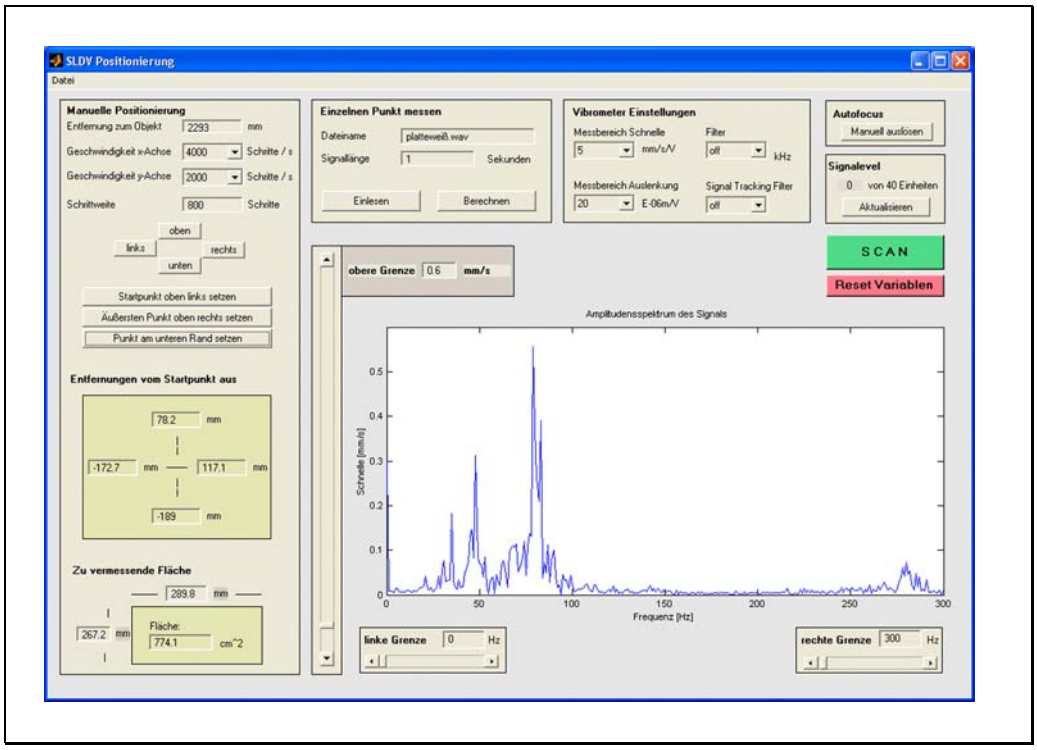


Abbildung 24: Screenshot des SLDV -Positionierungs-Programms (im Folgenden werden hieraus Ausschnitte vergrößert dargestellt)

Als erstes sollten daher die wichtigsten Daten zur Position, zur Schrittmotorgeschwindigkeit und zur Anzahl der auszuführenden Schritte eingegeben werden. Die betreffenden Felder sind in Abb. 25 dargestellt. Als erstes ist hier die Entfernung zum Objekt in mm einzutragen, danach die Geschwindigkeiten für die einzelnen Achsen bei der manuellen Positionierung, sowie die gewünschte Schrittweite.

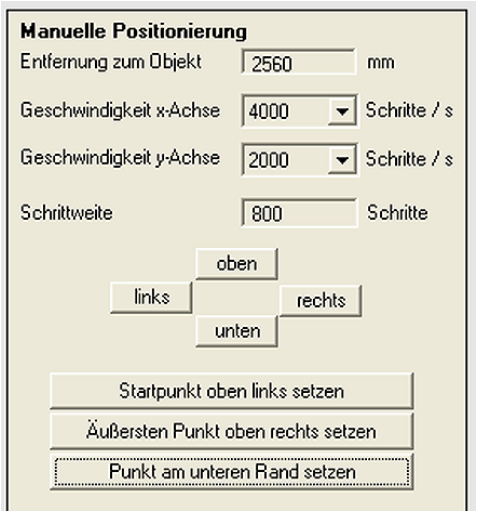


Abbildung 25: Felder zur manuellen Positionierung

Mit einem Klick auf einen der Buttons „oben“, „unten“, „links“ oder „rechts“ fährt der Messkopf anschließend mit der eingestellten Geschwindigkeit die Anzahl der Schritte in diese Richtung. Um nun das Scanfeld zu definieren, muss der Laserstrahl zum oberen linken (Start-) Punkt gefahren und dort der Button „Startpunkt oben links setzen“ betätigt werden. Sofort erscheinen die Werte für den Abstand zum Ausgangspunkt in mm in den Feldern oben und links (siehe Abb. 26).

Fährt man nun den Laser zum äußersten rechten Punkt des Scanfeldes und betätigt den Button „Äußersten Punkt oben rechts setzen“, wird auch dieser Abstand in das entsprechende Feld eingetragen; zudem der Gesamtabstand der beiden definierten Punkte im Bereich „Zu vermessende Fläche“.

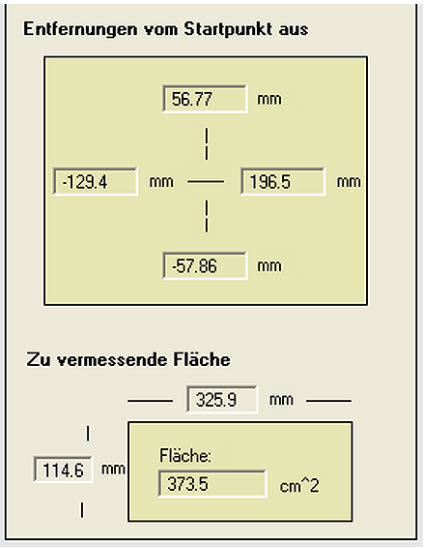


Abbildung 26: Berechnete Entfernungen des Laserstrahls vom Startpunkt aus

Als letzter Punkt des Scanfeldes muss nun noch ein Punkt am unteren Rand definiert werden. Dazu ist der Laserstrahl vom oberen rechten Punkt aus nach unten zu fahren und dort der Button „Punkt am unteren Rand setzen“ zu drücken. Damit vervollständigen sich auch die Werte für die zu vermessende Fläche und die Entfernungen zum Ausgangspunkt.

Da die Matlab-Funktionen für die Bewegung bereits weiter oben beschrieben wurden, soll nun die Berechnung der Abstände näher erläutert werden.

Die Grundlage für die Berechnung der Messpunktabstände auf der zu scannenden Oberfläche bilden die in Abschnitt 4.1.2 gefundenen Näherungen für die Abhängigkeit der Schrittanzahl der Kugelgewindevorschübe zum Dreh- bzw.

Kippwinkel. Als Beispiel soll hier die Funktion `strecke_li_ausr` vorgestellt werden. Hier wird nach dem Anfahren des oberen linken Eckpunktes der horizontale Abstand zum Ausgangspunkt berechnet (siehe Abb. 27).

```
function strecke_li_ausr

% --------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% --------------------------------------------------------------
global  entfernung...
        summe_schritte_links...
        summe_schritte_rechts...
        linke_strecke...
        ges_schritte_li...

handles = guihandles(gcbo);

% --------------------------------------------------------------
% Auslesen der Entfernung zum Objekt
% --------------------------------------------------------------
entfernung = str2num(get(handles.edit4,'String'))

% --------------------------------------------------------------
% Berechnen der eff. Schrittzahl nach links
% --------------------------------------------------------------
ges_schritte_li = summe_schritte_links...
                  + summe_schritte_rechts


% --------------------------------------------------------------
% Berechnen des Winkels aus Anzahl der Schritte
% --------------------------------------------------------------
winkel =   8.249357569520E-23*(ges_schritte_li)^5 ...
         + 1.419873939066E-18*(ges_schritte_li)^4 ...
         - 3.720683184245E-13*(ges_schritte_li)^3 ...
         + 7.668028385937E-09*(ges_schritte_li)^2 ...
         + 1.815281965615E-03*(ges_schritte_li)

% --------------------------------------------------------------
% Berechnen des Abstandes aus dem Winkel
% --------------------------------------------------------------
linke_strecke = tan((2*pi*winkel/360))*entfernung

% --------------------------------------------------------------
% Horizontale Entfernung zum Ausgangspunkt wird in
% Feld geschrieben
% --------------------------------------------------------------
set(handles.edit7,'String',num2str(linke_strecke,4));
```

Abbildung 27: Programm zur Berechnung des horizontalen Abstandes zum Ausgangspunkt

Zunächst wird hier die Entfernung zur Oberfläche aus dem betreffenden Feld ausgelesen. Da beim Ansteuern des oberen linken Eckpunktes evtl. mehrere Bewegungen in positiver und negativer horizontaler Richtung nötig gewesen sein könnten, muss nun die effektive Anzahl der Schritte in die linke Richtung berechnet werden. Dazu werden alle bisher ausgeführten Schritte in die rechte und linke Richtung aufsummiert. Das Ergebnis entspricht der Anzahl der effektiven Schrittzahl nach links, da Schritte nach links immer ein negatives Vorzeichen und Schritte nach rechts immer ein positives Vorzeichen haben.

Als nächstes wird nun der Winkel in der Horizontalen zum äußersten linken Punkt berechnet. Dies geschieht über die bereits oben erwähnte Näherungsgleichung mit einem Polynom fünften Grades. Über diesen Winkel kann nun mittels einer einfachen Dreiecksberechnung der horizontale Abstand des oberen linken Eckpunktes zum Ausgangspunkt berechnet werden. Dieser wird anschließend in das betreffende Feld geschrieben.

Für die anderen Bewegungsrichtungen rechts, oben und unten sind diese Funktionen analog programmiert worden. Sie bilden die Grundlage für die spätere Berechnung der Gesamtzahl der Scanpunkte in einer Zeile und die Anzahl der Zeilen selbst. Ebenso wird aus den Näherungsgleichungen die benötigte Anzahl der Schritte berechnet, die beim Scannen zwischen zwei Punkten liegt.

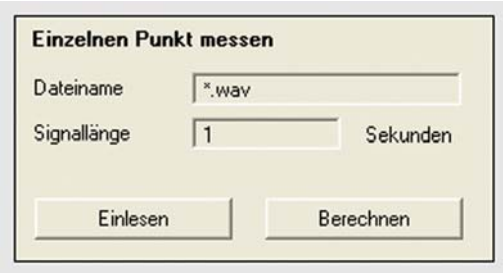


Abbildung 28: Dialog zum Einlesen eines einzelnen Messpunktes

Weiterhin bietet das SLDV Positionierungs-Programm die Möglichkeit, die Schwingungen auf der Oberfläche in einem einzelnen Punkt zu messen. Dazu muss zunächst das Signal des Vibrometer Controllers als Wave-File eingelesen werden. Der in Abb. 28

dargestellte Ausschnitt des Programms lässt hier eine individuelle Benennung des zu erzeugenden Files und die Wahl der Signallänge zu. Über den Button „Einlesen“ kann nun das Signal, sofern der Velocity-Ausgang des Vibrometer Controllers mit dem LineIn-Eingang der Soundkarte verbunden ist, eingelesen und damit in das Matlab-Programmverzeichnis gespeichert werden. Zu beachten ist hier, das beim Einlesen der zuvor eingestellte Messbereich und die Filter-einstellungen (siehe Abb. 29) an den Vibrometer Controller gesendet wurden und verwendet werden.

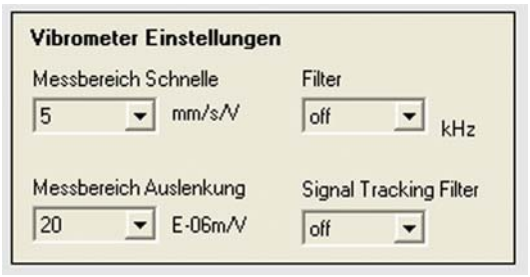


Abbildung 29: Dialog zum Setzen der Vibrometer-Controller-Einstellungen

Danach wird das Signal automatisch eingelesen, eine FFT berechnet, und das Amplitudenspektrum in der Programmoberfläche dargestellt. Möchte man nun ein bereits vorhandenes Wave-File erneut einlesen, ist der Name in das Feld einzutragen und der Button „Berechnen“ zu drücken.

Nachdem nun ein Wave-File mit dem Signal erzeugt und abgespeichert worden ist, berechnet das Programm die FFT und stellt das Amplitudenspektrum in der Programmoberfläche dar (siehe Abb. 30).

Die Skalierung der y-Achse wird hierbei aus der Vibrometer-Controller-Einstellung „Messbereich Schnelle“ berechnet. Über die Slider kann nun die obere Grenze so variiert werden, dass das Spektrum in das Schaubild passt. Über die Slider für die linke und rechte Grenze können Ausschnitte vergrößert dargestellt oder „interessante“ Abschnitte herausgegriffen werden. Zusätzlich zur Auswahl über die Slider können die gewünschten Grenzen auch direkt in die Textfelder eingegeben werden.

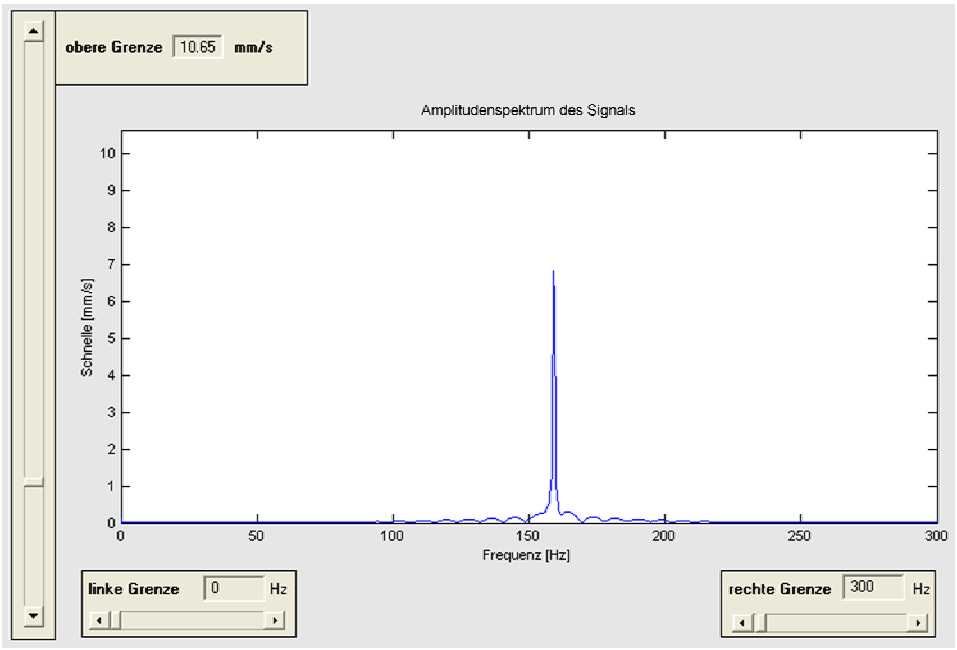


Abbildung 30: Darstellung des Amplitudenspektrums eines einzelnen Messpunktes als Schnelle in mm/s über die Frequenz

Die Wahl dieser Grenzen ist wichtig für das anschließende Scan-Programm. Es ist oftmals nicht notwendig, eine FFT und das Amplitudenspektrum für den gesamten Frequenzbereich von 0 bis 22050 Hz zu berechnen, daher werden zur Reduktion der Datenmenge (dazu im nächsten Abschnitt mehr) die FFT und das Amplitudenspektrum nur für die Frequenzen berechnet, die zwischen den mit den Slidern zu wählenden Grenzen liegen. Es ist daher unbedingt notwendig, darauf zu achten, dass nur die Frequenzen in den hier eingestellten Grenzen beim Scannen berücksichtigt werden.

Die Einzelheiten zur Berechnung der FFT und des Amplitudenspektrums, der Umrechnung in Volt über den Kalibrierungsfaktor und das Multiplizieren mit dem am Vibrometer Controller gewählten Messbereich im Programm `ein-punkt_berechnen.m` wurden bereits in Abschnitt 4.2.2 beschrieben.

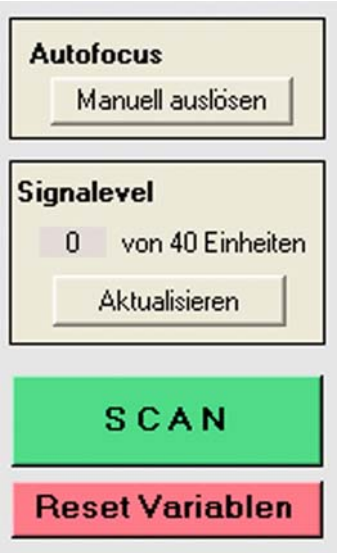


Abbildung 31: Sonstige Einstellungen und Scan-Button

Mit dem Button „Manuell auslösen“ im Feld „Autofocus“ der Programmoberfläche kann auf Wunsch noch der Autofokus ausgelöst werden. Sollte dabei eine Signalstärke ungleich null erzielt werden, wird diese im Feld „Signallevel“ angezeigt (siehe Abb. 31).

Über den Button „Reset Variablen“ werden alle Variablen auf null zurückgesetzt, die zur Berechnung der Position des Laserstrahls notwendig sind. Dies ist mit einem kompletten Neustart des Programms zu vergleichen und z.B. dann anzuwenden, wenn ein neuer Startpunkt gesetzt werden soll oder die Position des Messobjekts verändert wurde. Ebenso werden mit diesem Button die seriellen Verbindungen beendet, falls diese nach einem Abbruch noch offen sein sollten.

Über den Button „SCAN“ startet man letztlich das SLDV-Scanning-Programm und überträgt hiermit auch automatisch den zuvor gewählten Frequenzbereich mit den Variablen `xlinks` und `xrechts`, sowie den Maximalwert für die angezeigte Schnelle `ymax`. Letzterer kann später selbstverständlich noch geändert werden, er dient nur zur einmaligen Bestimmung des darzustellenden Maximums während des Scanvorgangs.

4.2.5 Das SLDV-Scanning-Programm unter Matlab

Das Hauptprogramm zum Scannen von Oberflächen ist ähnlich aufgebaut wie das Programm zur Positionierung des Messkopfes. Auch hier werden zunächst oben links in der Programmoberfläche (siehe Abb. 32) die gewünschten Einstellungen

für den Scanvorgang ausgewählt. Dazu gehören die Schrittgeschwindigkeit der Motoren (in Schritte pro Sekunde) und die Schrittweite der einzelnen Messpunkte (in mm). Weiterhin kann auf Wunsch eine Autofokus-Funktion gewählt werden, die das Fokussieren vor jedem Messpunkt bewirkt, sofern die Signalstärke gleich null ist. Während einer Messung wird dann die Signalstärke in dem Feld darunter mit jedem Messpunkt aktualisiert.

Nachdem mit dem Positionierungsprogramm die drei Eckpunkte der zu vermessenden Fläche bestimmt wurden, sollte der Messkopf noch immer auf den unteren, rechten Punkt zeigen. Mit einem Klick auf den Button „Startposition: Hinfahren“ kann nun der erste Punkt oben links automatisch angefahren werden.

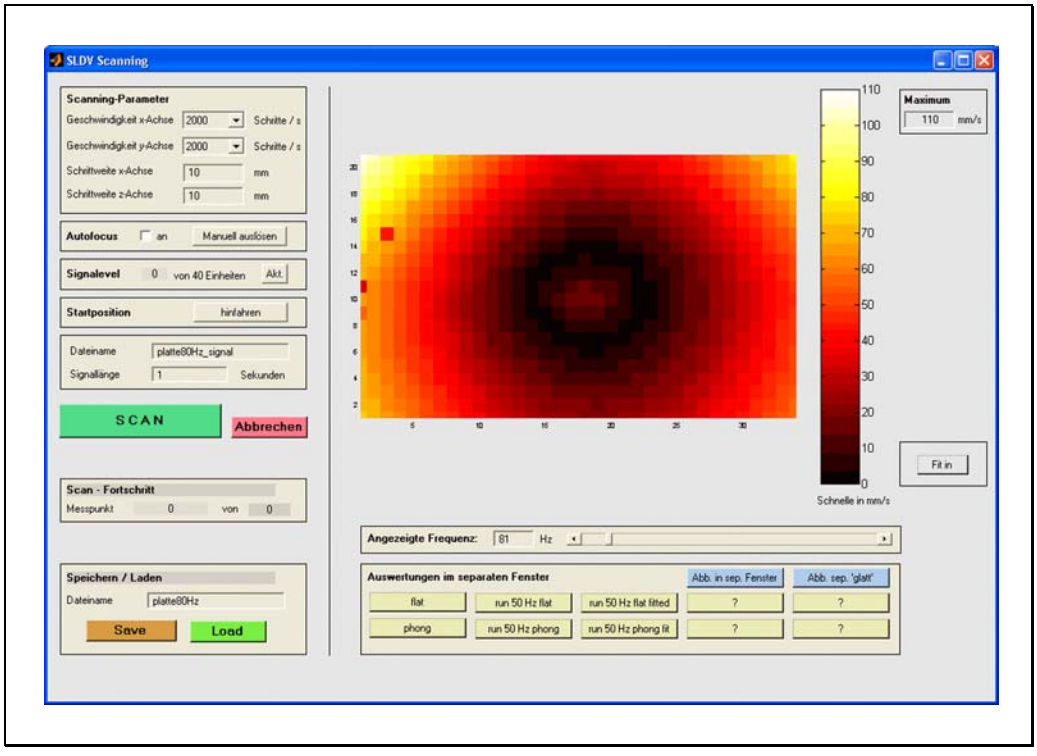


Abbildung 32: Das SLDV-Scanning-Programm unter Matlab

Damit bereits während des Scanvorgangs ein Bild angezeigt werden kann, sollte nun noch die gewünschte angezeigte Frequenz in das entsprechende Feld unter der Matrixdarstellung eingetragen werden (in Abb. 32 z.B 81 Hz).

Mit der Benennung der eingelesenen Messpunkt-Signale im Feld „Dateiname“ und der Wahl der Signallänge sind die Vorbereitungen zum Scannen bereits abgeschlossen.

Der Scanvorgang kann nun mit einem Klick auf den Button „SCAN“ gestartet werden. In der Fortschrittsanzeige werden daraufhin die Anzahl der bereits gemessenen Punkte und die Gesamtzahl der Messpunkte angezeigt. Mit jedem Messpunkt wird die Legende der Matrixdarstellung automatisch auf den Höchstwert skaliert.

Während des Scanvorgangs kann die angezeigte Frequenz mit Hilfe der beiden Slider vorsichtig verändert werden. Zu beachten ist hier, dass es bei einer „Überlastung“ von Matlab leicht zu einem Programmabbruch kommen kann.

Nach dieser Beschreibung der Programmoberfläche soll nun ein kurzer Einblick in den Aufbau der Matlab-Funktion erfolgen. Die gesamte Funktion ist im Anhang 5.8 („scan_autoscan.m“) aufgeführt, an dieser Stelle sollen nur die wichtigsten Ausschnitte daraus betrachtet werden.

Zu Beginn dieses Programms werden die Schleifenvariablen ermittelt (siehe Abb. 33). Der gesamte Scanvorgang spielt sich dabei innerhalb zweier ineinander verschachtelter Schleifen ab: eine für das Bewegen und Scannen in x-Richtung und eine für die y-Richtung. Die Variablen `x_schleife_ende` und `y_schleife_ende` sind daher auch gleichzeitig die Werte für die Anzahl der Scanpunkte in x- und y-Richtung.

```
% ------------------------------------------------------------------
% Schleifenvariablen setzen
% ------------------------------------------------------------------
x_schleife_anfang      =  1;
x_schleife_ende        =  fix(horizontale/scan_schrittweite_x)+1;
y_schleife_anfang      =  1;
y_schleife_ende        =  fix(vertikale/scan_schrittweite_y)+1;
```

Abbildung 33: SLDV-Scanning-Programm: Ermittlung der Schleifenvariablen

Die Gesamtzahl der Scanpunkte wird dann im Folgenden aus dem Produkt dieser beiden Variablen berechnet und in das entsprechende Feld der Programmoberfläche geschrieben.

Ebenfalls zu Beginn dieser Funktion wird der Soundkarten-spezifische Skalierungsfaktor berechnet (siehe Ab.. 34).

```
% ------------------------------------------------------------
% Berechnung Skalierungsfaktors aus den Variablen der
  Kalibrierung
% ------------------------------------------------------------
skalierungsfaktor = effektivspannung/Kali_Pyy;
```

Abbildung 34: SLDV-Scanning-Programm: Berechnung des Skalierungsfaktors

Dieser ist der Quotient aus der im Kalibrierungsprogramm eingegebenen Effektivspannung und dem gemittelten Effektivwert der eingelesenen Kalibrierungssignale (siehe auch Abschnitt 4.2.3). Er besitzt die Dimension „Volt pro Punkte“, wobei die maximale Anzahl der „Punkte“ die Auflösung der Soundkarte (16 Bit) bedeutet.

Nachdem in einem weiteren Schritt die Schleife für die Bewegung des Scanning-Vibrometers in y-Richtung initialisiert wurde (‚`for y_scanschleife = y_schleife_anfang : y_schleife_ende`'), wird der gesamte vertikale Scanwinkel berechnet (siehe Abb. 35).

```
% ------------------------------------------------------------
% Anzahl des gesamten Scanwinkels in y-Richtung wird
  berechnet
% ------------------------------------------------------------
ges_scanwinkel_y  = 2*((360/(2*pi))* ...
   atan(0.5*y_schleife_ende*scan_schrittweite_y/entfernung));
```

Abbildung 35: SLDV-Scanning-Programm: Berechnung des Scanwinkels

Dieser gesamte Winkel ergibt sich aus der Summe zweier rechtwinkeliger Dreiecke mit der Entfernung zum Messobjekt als Ankathete und dem Produkt aus der Anzahl der Messpunkte mit deren Abstand zueinander als Gegenkathete (siehe Abb. 36).

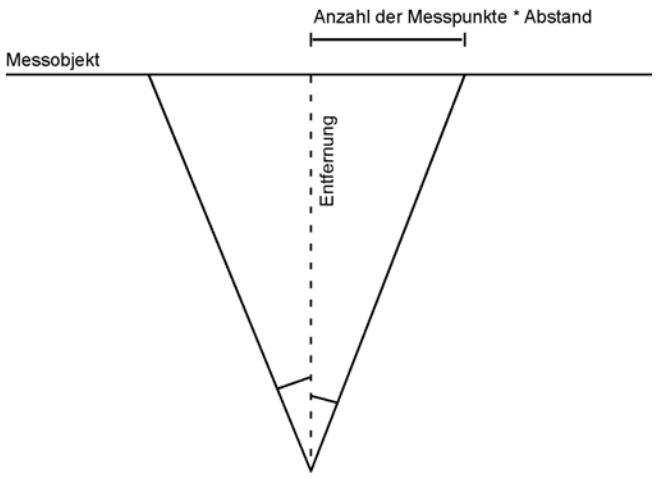


Abbildung 36: Geometrische Berechnung des gesamten Scanwinkels

Die durchzuführenden Schritte von Messpunkt zu Messpunkt werden mit Hilfe der in Abschnitt 4.1.2 berechneten Näherungsgleichungen bestimmt. Da dieser Zusammenhang nicht linear ist, sondern vom bereits zurückgelegten Weg abhängig ist, muss zunächst der Winkel zweier Messpunkte zueinander bestimmt werden (siehe Abb. 37).

```
% ------------------------------------------------------------------
% Aktuelle Schrittzahl in y-Richtung berechnen
% ------------------------------------------------------------------
   scanwinkel_y_o    = 0.5*ges_scanwinkel_y - ((360/(2*pi))* ...
                       atan((y_scanschleife-
                       1)*scan_schrittweite_y/entfernung));

   scanwinkel_y_u    = 0.5*ges_scanwinkel_y - ((360/(2*pi))* ...
                       atan(((y_scanschleife)*
                       scan_schrittweite_y)/entfernung));

   scan_schritte_y = round(abs((...
                       3.353899071668E-04*(scanwinkel_y_o)^5 ...
                     + 1.280602602320E-02*(scanwinkel_y_o)^4 ...
                     + 3.287381441029E-01*(scanwinkel_y_o)^3 ...
                     + 7.314062013423E+00*(scanwinkel_y_o)^2 ...
                     - 1.207856559729E+03*(scanwinkel_y_o))  ...
                     - ...
                     +(3.353899071668E-04*(scanwinkel_y_u)^5 ...
                     + 1.280602602320E-02*(scanwinkel_y_u)^4 ...
                     + 3.287381441029E-01*(scanwinkel_y_u)^3 ...
                     + 7.314062013423E+00*(scanwinkel_y_u)^2 ...
                     - 1.207856559729E+03*(scanwinkel_y_u))));

   bisherige_y_schritte  =  bisherige_y_schritte + scan_schritte_y;
```

Abbildung 37: SLDV-Scanning-Programm: Berechung des Winkels zwischen zwei Messpunkten und die notwendige Anzahl der Schritte zwischen ihnen

Hier werden die Winkel des anzufahrenden Messpunktes und des bereits gemessenen Punktes bestimmt und mit Hilfe der Näherungsgleichung die Differenz der dazwischen liegenden Schritte errechnet. In der Variable `bisherige_y_schritte` wird die bisherige Anzahl der durchgeführten Schritte gespeichert.

Diese Berechnung wird später innerhalb der Schleife für die Bewegung in x-Richtung analog ausgeführt.

Die Scanbewegung des Messkopfes hat zur Folge, dass bei einem „schiefen“ (d.h. nicht rechtwinkeligen) Auftreffen des Laserstrahls auf das Messobjekt lediglich die Schwingungen senkrecht zum ihm erfasst werden. Hier muss nun eine Korrektur erfolgen, indem der Winkel des Laserstrahls zum Lot der Oberfläche des Messobjekts in jedem Messpunkt berechnet wird und das dort ausgelesene Zeitsignal (als Vektor in Matlab) durch den Kosinus dieses Winkels geteilt wird (siehe Abb. 38).

```
% ------------------------------------------------------------
% Korrekturfaktor für "schiefes" Messen wird berechnet
% ------------------------------------------------------------
if ((0.5*horizontale-x_scanschleife*scan_schrittweite_x) > 0 ...
      | (0.5*vertikale-y_scanschleife*scan_schrittweite_y) > 0)

   winkelkorrektur = 1/(cos((pi/2)- ...
                     atan(entfernung/ ...
                     (sqrt((0.5*horizontale-
                        x_scanschleife * scan_schrittweite_x)^2 ...
                     + (0.5*vertikale-
                        y_scanschleife*scan_schrittweite_y)^2)))))
 else
   winkelkorrektur = 1;

end;
```

Abbildung 38: SLDV-Scanning-Programm: Berechnung des Korrekturfaktors für „schiefes“ Messen

Nachdem nun die Anzahl der auszuführenden Schritte, der Kalibrierungsfaktor und der Korrekturfaktor für den Winkel bekannt sind, wird das Signal für den aktuellen Messpunkt als Wave-File eingelesen und unter dem gewählten Dateinamen (mit den Koordinaten des Punktes zusammen, z.B.

„messpunkt_1_1“) in das Matlab-Verzeichnis gespeichert. Daraufhin erfolgt die Bewegung zum nächsten Messpunkt und eine Pause („`pause((scan_schritte_x/scan_v_x)+0.3)`“) mit der Dauer dieser Bewegung.

Im Anschluss daran wird das Zeitsignal wieder eingelesen und wie in Abschnitt 4.2.2 beschrieben die FFT und das Amplitudenspektrum berechnet. Damit die Gesamtmenge der Daten nicht zu groß wird, wird hier nur der im Positionierungsprogramm gewählte Frequenzausschnitt (`[xlinks xrechts]`) berücksichtigt und in die Messwertematrix geschrieben, die daraufhin im Scanning-Programm dargestellt wird (siehe Abb. 39).

```
% ------------------------------------------------------------------------
% Einlesen des wave-Files
% ------------------------------------------------------------------------
fid            = fopen(scan_dateiname_komplett, 'r'); % Handle erzeugen
fseek(fid, 22, 'bof');                    % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');  % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');  % lese samplerate
fseek(fid, 44, 'bof');                    % springe zum 44. Byte
scan_signal    = fread(fid, inf, 'int16');  % lese ab hier messsignal
fclose(fid);                               % fid schließen

% ------------------------------------------------------------------------
% Verschieben des Zeitsignals, so dass max = min (Offset der
% Soundkarte!)
% ------------------------------------------------------------------------
max_wav = max(scan_signal);              % Maximum des Zeitsignals
min_wav = min(scan_signal);              % Minimum des Zeitsignals
differenz_ampl = max_wav + min_wav;
scan_signal  = scan_signal-0.5*differenz_ampl;

% ------------------------------------------------------------------------
% Multiplizieren des Signals mit Winkelkorrekturfaktor
% ------------------------------------------------------------------------
scan_signal  = winkelkorrektur*scan_signal;

% ------------------------------------------------------------------------
% Bestimmen der Länge des Eingangssignals
% ------------------------------------------------------------------------
N = length(scan_signal);

% ------------------------------------------------------------------------
% Berechnen der FFT
% ------------------------------------------------------------------------
```

```
fft_scan_signal    = fft(scan_signal,N);

clear scan_signal

% ------------------------------------------------------------------------
% Umrechnen in Schnelle
% ------------------------------------------------------------------------
        if messb_schnelle == 6
            y_range = 5;                    % 5 mm/s/V
        elseif messb_schnelle == 7
            y_range = 25;                   % 25 mm/s/V
        elseif messb_schnelle == 8
            y_range = 125;                  % 125 mm/s/V
        elseif messb_schnelle == 9
            y_range = 1000;                 % 1000 mm/s/V
        end


% ------------------------------------------------------------------------
% Darstellen als Funktion von x=0 bis x=(Samplingfrequenz/2) und
% Skalierung der y-Achse auf Grundlage der Vibrometereinstellungen
% ------------------------------------------------------------------------
        a    = N/2;                 % halbe Signallänge
        b    = a+1;                 % halbe Signallänge + 1

        f = 44100*(0:a)/N;          % x-Achse

        Ay  = abs(fft_scan_signal);  % entspr. Ay = sqrt(Y .* conj(Y));

        clear fft_scan_signal

        Ay  = 2.*Ay./N./sqrt(2);    % 'normalized discrete fourier-
                                    % transf., siehe Matlab Hilfe

        Ay  =  y_range*skalierungsfaktor*Ay;  % Skalierungsfaktor und
                                % Einstellung des Vibrometers
        max_Ay = max(Ay)        % Ausgabe des Maximalwertes in Command.


% ------------------------------------------------------------------------
% auf Wunsch: Nur Messpunkte verwenden, wenn Signal vorhanden
% ------------------------------------------------------------------------
% scan_signallevellesen
% if scan_signallevel == 0
%     Ay = 0.*Ay;
% end;

% ------------------------------------------------------------------------
% Berechnen der Grenzen für den Frequenzauschnitt
% ------------------------------------------------------------------------
        ug  = xlinks*N/Fs;
        og  = xrechts*N/Fs;
        Ay = Ay(ug+1:og);
        hoh = length(Ay);

% ------------------------------------------------------------------------
% Im ersten Schritt der Schleife wird eine mit Einsen gefüllte
% Messwertematrix erzeugt
% ------------------------------------------------------------------------
```

```
        if y_scanschleife == y_schleife_anfang...
                & x_scanschleife == x_schleife_anfang
            Y = zeros(y_schleife_ende,x_schleife_ende,hoh);
        end;

% ------------------------------------------------------------------------
% Schreiben des FFT-Betrags in die 3. Komponente der Messwerte-
% matrix
% ------------------------------------------------------------------------
        Y(y_scanschleife,x_scanschleife,:) = Ay;
        clear Ay

% ------------------------------------------------------------------------
% Anzeige der gemessenen Werte in axes2 als Matrix mit Skala
% ------------------------------------------------------------------------
        axes(handles.axes2)
        Messwerte_Matrix = Y;
        set(gcf,'Doublebuffer','on');
        colormap(hot)
        colorscale = [0 ymax];
        set(handles.slider1,'SliderStep',...
            [1/((xrechts-xlinks)*scan_signallaenge) ...
                10*(1/((xrechts-xlinks)*scan_signallaenge))])


        set(handles.slider1,'Min',xlinks)
        set(handles.slider1,'Max',xrechts)

        Messwerte_Matrix(:,:,((angezeigte_frequenz-
            xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,...
            ((angezeigte_frequenz-xlinks)*scan_signallaenge)));

        imagesc(Messwerte_Matrix(:,:,...
            ((angezeigte_frequenz-
              xlinks)*scan_signallaenge)),'parent',handles.axes2);
        colorbar
        set(gca,'ZLim',[0 ymax])
        xlim([0.999 x_schleife_ende])
        ylim([0.999 y_schleife_ende])
        drawnow
```

Abbildung 39: SLDV-Scanning-Programm: Einlesen des Zeitsignals, Berechnung des Amplitudenspektrums und Darstellung als Matrix

Die mit der in Abb. 39 dargestellten Funktion erzeugte „`Messwerte_Matrix`“ hat als Zeilen die y-Koordinaten des Messpunktes und als Spalten die x-Koordinaten. In die dritte Komponente dieser Matrix (dreidimensional sozusagen „nach oben“)

wird nun für jeden Messpunkt das mit dem Skalierungsfaktor und dem Messbereich multiplizierte Amplitudenspektrum geschrieben. Diese dritte Komponente beinhaltet also den Wert für die Schnelle in der richtigen Einheit und für jede Frequenz im gewählten Frequenzausschnitt. Daher ist es auch leicht einzusehen, dass die Datenmenge hier möglichst klein gehalten werden muss. Würde man für jeden Punkt das gesamte Amplitudenspektrum mit seinen 22.050 Punkten (bei einer Signallänge von 1 Sekunde) in diese Matrix schreiben, hätte diese bei einer Scanfläche von 50 x 50 Punkten eine Größe von 55.125.000 Einträgen zu je 8 Bytes, also einen Speicherbedarf von 441 MBytes (im Arbeitsspeicher!). Beschränkt man sich hier auf einen Ausschnitt von z.B. 0 bis 1000 Hz, verringert sich der Speicherbedarf bereits auf 22 MBytes.

Die Darstellung der Messwerte-Matrix erfolgt anschließend über den Befehl `imagesc` für die gewählte Frequenz im Feld „`axes2`“ zusammen mit der Legende („`colorbar`“).

Nachdem das Einlesen der Messpunkte abgeschlossen ist, kann die berechnete Matrix nach Eingabe eines Dateinamens in der Programmoberfläche abgespeichert und auf Wunsch später geladen werden. Das Speichern sichert die verwendeten Variablen in einer entsprechenden Datei „dateiname.mat“ im Matlab-Programmverzeichnis.

4.2.6 Postprocessing

In diesem Abschnitt soll auf die Möglichkeiten der Darstellung der Messwerte-Matrix eingegangen werden. In der SLDV-Scanning-Programmoberfläche befinden sich unten rechts (siehe Abb. 32) diverse Buttons, mit denen die Matrix in separaten Fenstern („figures“) abgebildet werden kann. Dies ist wesentlich praktischer als die Darstellung in der Programmoberfläche, weil Matlab hier automatisch zusätzliche Funktionen bietet, etwa das Exportieren als Bitmap, das

Drehen von dreidimensionalen Darstellungen, das individuelle Skalieren der Matrix oder die Verwendung einer anderen Colormap. Die entsprechenden Funktionen sind bei Bedarf in der Matlab-Programmhilfe nachzulesen.

Die angesprochenen Buttons zur Auswertung und Darstellung sind jeweils mit Funktionen Verknüpft, die mit „scan_ausw_*“ benannt sind und ebenfalls im Matlab-Programmverzeichnis zu finden sind. Diese können auf Wunsch abgeändert oder ergänzt werden - sie bilden nur einen Anhaltspunkt für die vielfältigen Möglichkeiten, die Matrix darzustellen.

Über den Button „Abb. in sep. Fenster“ kann die aktuell in der Programmoberfläche dargestellte Matrix in einem separaten Fenster geöffnet werden. Der Button „Abb. sep. ‚glatt’“ liefert ein ähnliches Bild, hier werden jedoch ± 10 Ebenen der Matrix (also bei einer Auflösung der Frequenz von 1 Hz der Bereich [-10Hz ... (dargestellte Frequenz) ... +10Hz]) addiert und abgebildet. Dies ist oft von Vorteil, wenn die Abbildung der Matrix sehr „pixelig“ ist und die Einbeziehung der Frequenzen um die aktuell angezeigte herum gewünscht ist.

Die Buttons „flat“ und „phong“ liefern jeweils eine dreidimensionale Darstellung der Matrix, die mit dem Matlab-Befehl „`surf`“ erzeugt wird. Neben den Koordinaten des Messpunktes in x- und y-Richtung ist hier die Schnelle des jeweiligen Punktes in z-Richtung aufgetragen.

Über die Buttons „run ...“ können nun die dreidimensionalen Matrixansichten als Animation für die nächsten 50 Hz betrachtet werden - dies ist u.a. sehr nützlich, wenn bestimmte Moden auf der abgescannten Oberfläche schneller erkannt werden sollen. Die Buttons mit der Beschriftung „fitted“ skalieren jeweils automatisch die Darstellung auf das Maximum der z-Achse.

Neben diesen „vorgefertigten“ Buttons kann selbstverständlich auch das abgespeicherte „*.mat“-File per Doppelklick in Matlab eingelesen werden, womit die Messwerte als Matrix im Hauptspeicher zur Verfügung stehen und individuell bearbeitet werden können.

Kapitel 5

Messung verschiedener Oberflächen-schwingungen

In diesem Kapitel sollen drei Messungen mit dem Scanning-Laservibrometer vorgestellt werden - zum einen wird eine mittig aufgehängte Platte (Panel) mit einem Shaker angeregt und abgescannt, zum anderen ein Breitbandlautsprecher, der an einen Frequenzgenerator angeschlossen ist, und weiterhin die Seite eines PC-Gehäuses. Zuvor werden die notwendigen Vorbereitungen, der Versuchsaufbau und das Kalibrieren der Soundkarte für den verwendeten Laptop beschrieben.

5.1 Vorbereitungen und Versuchsaufbau

Bevor das Scanning-Laservibrometer (wie in Abb. 42) aufgebaut wird, sollte die verwendete Matlab-Software gepatcht werden. Dies ist notwendig, da in der verwendeten Version 6 Release 12.1 die Erkennung von seriellen Schnittstellen fehlerhaft ist. Die Korrektur dieses Fehlers ist simpel: Es ist lediglich die Datei „javax.comm.properties“ vom Verzeichnis „matlabroot\java\jarext\commapi\win32“ in das Verzeichnis „matlabroot\sys\java\jre\win32\jre\lib“ zu kopieren. Bei späteren Matlab-Versionen soll dieser Fehler behoben werden.
Der in Abb. 42 dargestellte Versuchsaufbau stellt das Scanning-Laservibrometer auf einem Wagen mit einer Kunststoffbox als Erhöhung dar. In Zukunft ist es vorstellbar, dass der Positionierungsrahmen auf einem Stativ montiert wird.

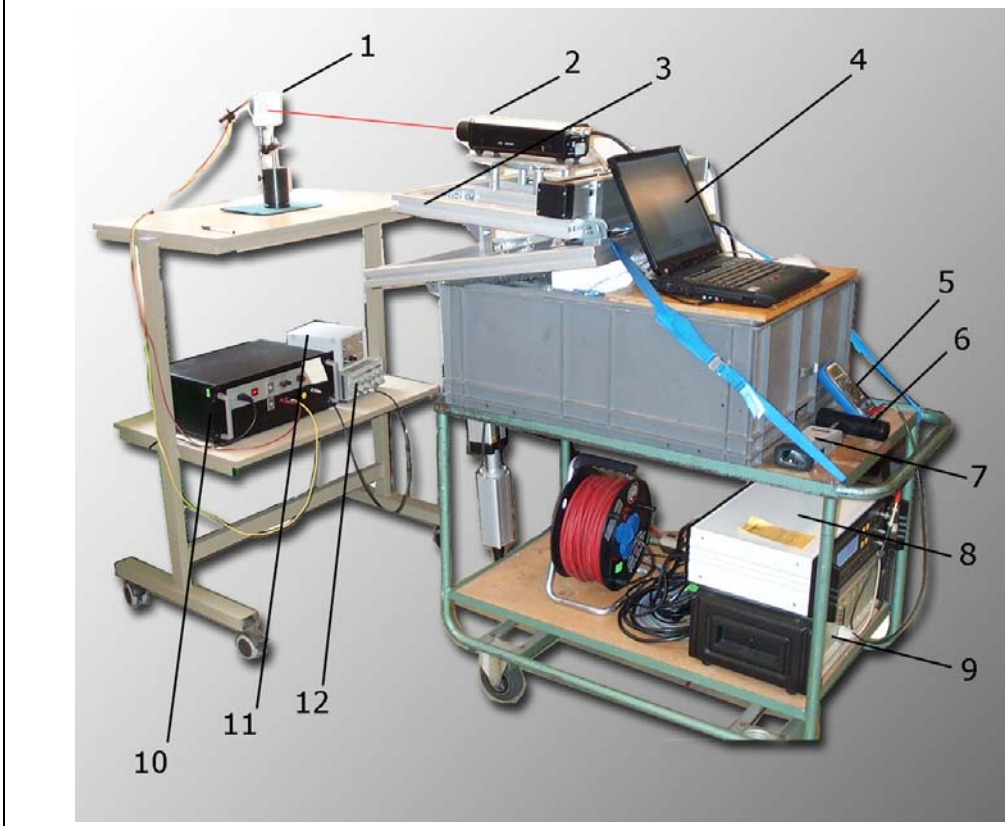


Abbildung 40: Versuchsaufbau

1	Messobjekt, hier: 10cm Breitbandlautsprecher VISATON R 10S-4
2	Polytec Vibrometer Single-Point Messkopf OFV-303
3	Positionierungsrahmen
4	Laptop IBM A31 mit Software Matlab V6.1 Release 12.1
5	Multimeter Metrix
6	Kalibrator B&K
7	Laser-Entfernungsmesser Leica
8	Polytec Vibrometer Controller OFV-3001
9	ISEL Schrittmotorcontroller C 142-1
10	Verstärker
11	Rauschgenerator
12	Frequenzgenerator

Wichtig bei dieser Art des Aufbaus ist hier, dass der untere Teil des Rahmens einen Neigungswinkel von etwa 20° nach unten aufweist, damit dem oberen Teil ein genügend großer Fahrtweg nach unten zur Verfügung steht.

Der Abstand der Frontplatte des Messkopfes zum Messobjekt ist gem. [Poly97, Kap. 5-11] mit 232, 437 oder 642 mm usw. (+205 mm) zu wählen, damit eine optimale Signalstärke erreicht wird. Dies ist jedoch nicht unbedingt notwendig, in mehreren Messungen wurden bei unterschiedlichen Entfernungen ebenfalls optimale Signalstärken erreicht. Bei Messobjekten mit einer wenig reflektierenden Oberfläche sollte dies jedoch berücksichtigt werden.

Abbildung 41: Messung des Abstandes zum Messobjekt

Die einzelnen Komponenten sind wie in Abschnitt 4.1 beschrieben miteinander zu verbinden. Mit Hilfe des Laser-Entfernungsmessers kann nun der Abstand des Messkopfes (gemessen ab Drehpunkt) zum Messobjekt bestimmt werden (siehe Abb. 41). Dieser ist später in das SLDV-Positionierungs-Programm einzutragen.

5.2 Kalibrieren der Soundkarte

Bevor überhaupt eine Messung durchgeführt wird, sollte eine Kalibrierung der Soundkarte des verwendeten PCs oder Laptops erfolgen. Die Funktionsweise des entsprechenden Programms hierfür wurde bereits in Abschnitt 4.2.3 erörtert. An dieser Stelle soll noch einmal kurz die Kalibrierung des Versuchsaufbaus mit dem verwendeten Laptop A31 von IBM beschrieben werden.

Mit Hilfe eines Multimeters wird dazu die Effektivspannung des Ausgangssignal eines Frequenzgenerators (Sinusfunktion) so eingestellt, dass diese 1 Volt nicht übersteigt (hier: 0,067 V). Die Frequenz sollte dabei so gewählt werden, dass sie

ungefähr der später mit dem Vibrometer zu messenden Frequenz entspricht - in diesem Fall wurden 1000 Hz am Frequenzgenerator eingestellt.

Dieses vom Frequenzgenerator generierte Sinussignal wird anschließend an den Line-In-Eingang der Soundkarte gelegt. Der Regler für den Pegel des Line-In-Eingangs der Soundkarte muss auf das Maximum eingestellt werden, anschließend sollte zur Kontrolle das beiliegende Programm „Mixer“ gestartet werden, das zusätzlich die Anzeige des Eingangssignals als Zeitplot erlaubt. Hier ist gut zu erkennen, ob die Soundkarte übersteuert wird. Falls dies der Fall sein sollte, ist ein entsprechend geringerer Wert für die effektive Spannung des Signals einzustellen.

Nachdem die Software Matlab gestartet und das Programmverzeichnis ausgewählt wurde, wird über das Matlab-Command-Window das Programm „kalibrierung“ gestartet und der zuvor gemessene Effektivwert der Spannung in das entsprechende Feld eingetragen (siehe Abb.42).

Über den Button „Signal speichern“ werden anschließend fünf Signale mit einer Länge von jeweils fünf Sekunden eingelesen und gespeichert. Über den Button „Signal einlesen“ werden diese Signale wieder geladen, sowie die FFT und das Amplitudenspektrum berechnet und gemittelt. Zur Kontrolle, ob ein „sauberes“ Signal eingelesen wurde, werden die ersten 1000 Punkte des Signals und das Amplitudenspektrum in einem Plot dargestellt. Hier ist gut der Peak bei der am Frequenzgenerator eingestellten Frequenz von 1000 Hz zu erkennen. Über den Button „Speichern“ werden die wichtigsten Variablen, insbesondere die für den Kalibrierungsfaktor benötigten `effektivspannung` und `Kali_Pyy` gespeichert.

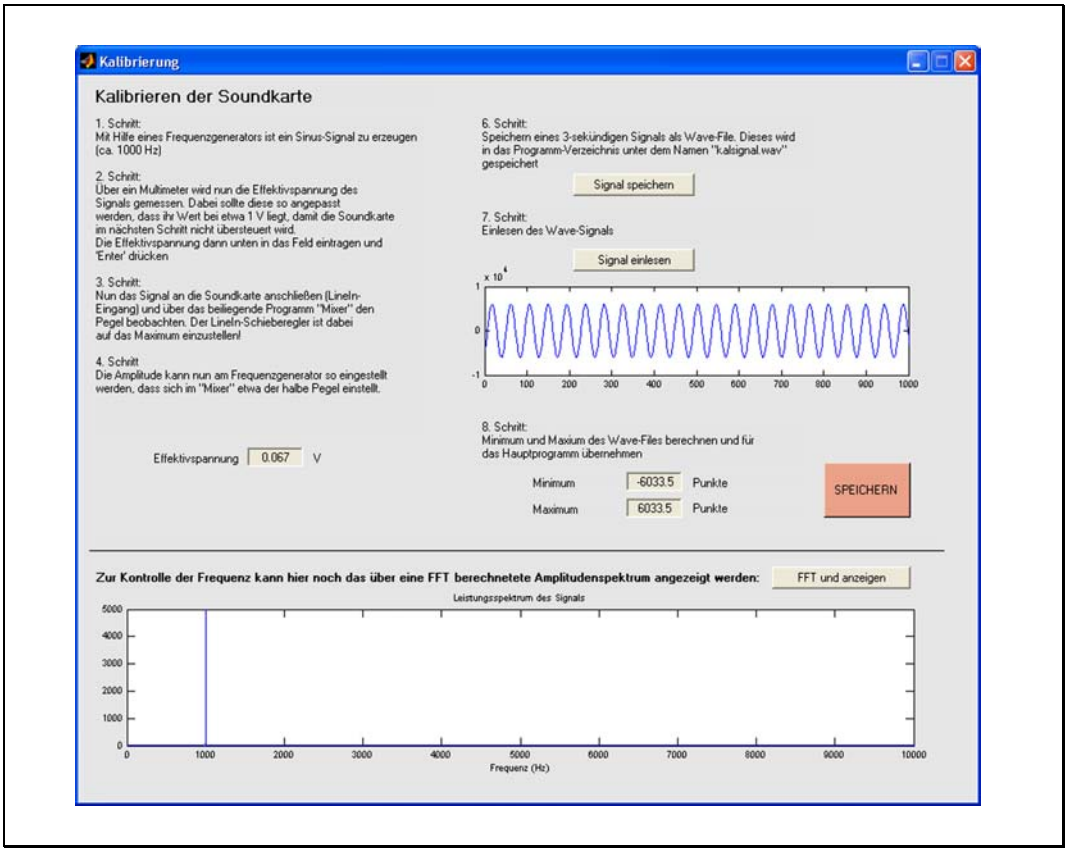


Abbildung 42: Kalibrierung der Soundkarte des Laptops IBM A31

Für den verwendeten Laptop A31 von IBM beträgt die gemessene Effektivspannung des Sinussignals 0,067 V und der Effektivwert des Amplitudenspektrums 4498,7 Punkte. Hiermit ergibt sich also ein Kalibrierungsfaktor von 1,4893e-005 Volt/Punkte.

5.3 Scan einer schwingenden Platte

In diesem und den nächsten beiden Abschnitten sollen die Ergebnisse von drei Messungen vorgestellt werden, die mit dem Aufbau aus Abb. 40 im Rahmen dieser Examensarbeit durchgeführt wurden. Als erstes wurde ein Panel (ca. 34 x

21 cm) mittig mit einem Shaker verbunden und an einen Frequenzgenerator angeschlossen (siehe Abb. 43).

Abbildung 43: Panel mit Shaker

Das Panel wurde anschließend mit Frequenzen 80, 160 und 320 Hz angeregt und mit einer Auflösung von 10 mm horizontal und vertikal in einem Abstand von 938 mm abgescannt. Die 714 Messpunkte wurden als Matrix in einem Frequenzbereich von 0 bis 1000 Hz abgespeichert.

In Abb. 44 ist das Resultat dieses Scans für die Frequenz 81 Hz dargestellt (siehe auch Abb. 32). Gut zu erkennen ist die Position der Aufhängung in der Mitte des Panels. An den Eckpunkten wird für diese Frequenz eine Schnelle von bis zu 110 mm/s gemessen, während in der Nähe der Aufhängung lediglich ca. 30 mm/s gemessen werden.

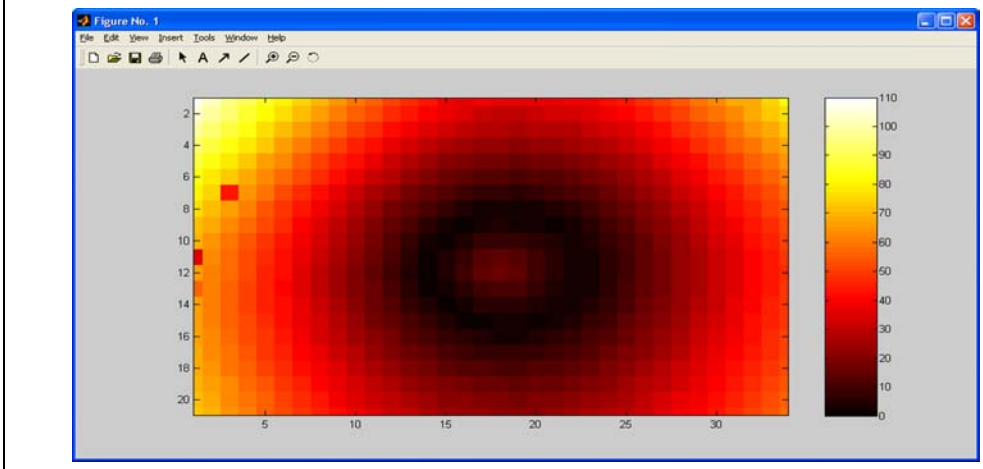


Abbildung 44: Ergebnis eines Scans des Panels, angezeigt wird die Frequenz 81 Hz, die Legende zeigt die Schnelle in mm/s

Regt man das Panel mit einer Frequenz von 160 Hz an, ergibt sich das in Abb. 45 dargestellte Bild:

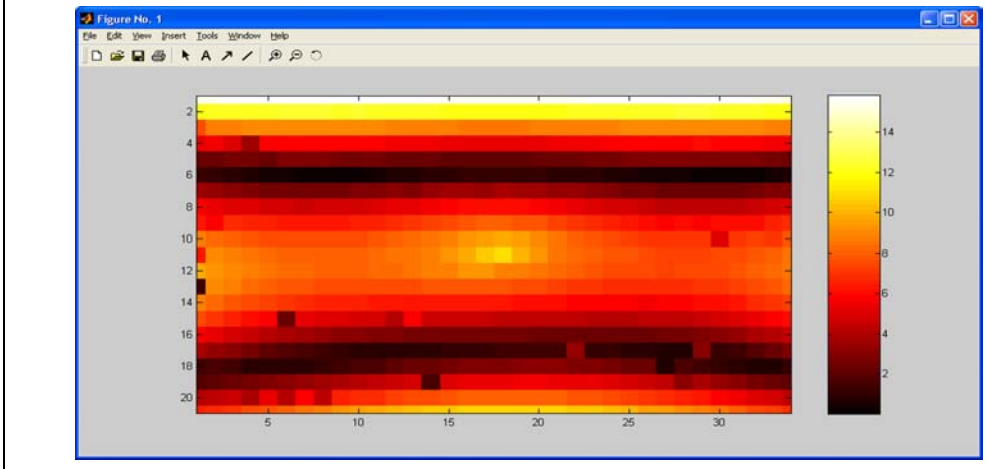


Abbildung 45: Ergebnis eines Scans des Panels, angezeigt wird die Frequenz 161 Hz

Hier ist sehr gut der waagerechte Verlauf der Moden zu erkennen. An der oberen und unteren Kante des Panels wird eine Schnelle von bis zu 15 mm/s gemessen. Regt man weiterhin das Panel mit einer Frequenz von 320 Hz an, ergibt sich das in Abb. 46 dargestellte Bild der Messwerte-Matrix.

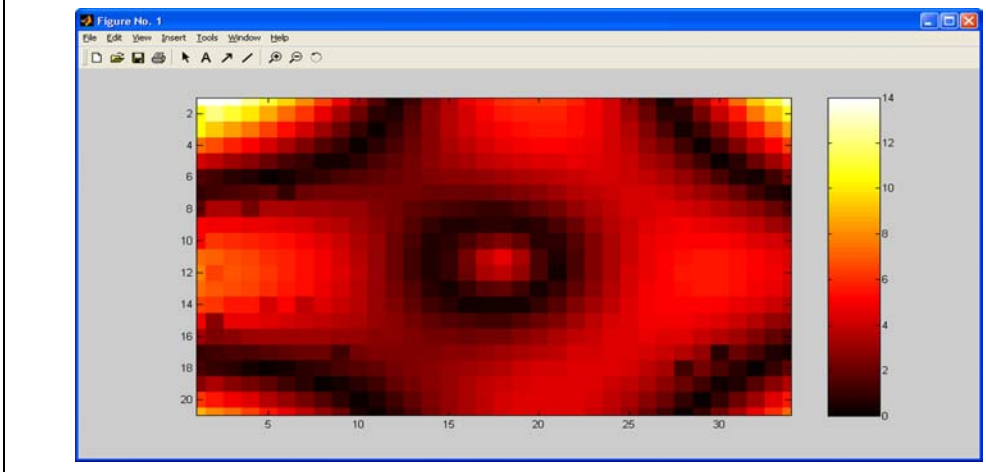


Abbildung 46: Ergebnis eines Scans des Panels, angezeigt wird die Frequenz 321 Hz

Auch hier ist gut zu erkennen, dass an den Eckpunkten des Panels eine etwa doppelt so große Schnelle (bis zu 14 mm/s) erreicht wird als im Aufhängepunkt.

Über das SLDV-Scanning-Programm kann dieses Bild zusätzlich als Oberfläche („Surface“) in Matlab dargestellt werden. Dabei entsprechen die x- und y-Koordinaten des Messpunktes der Spalte und Zeile der Matrix, in Abb. 47 in der x-y-Ebene dargestellt. Die Schnelle wird auf der z-Achse aufgetragen.

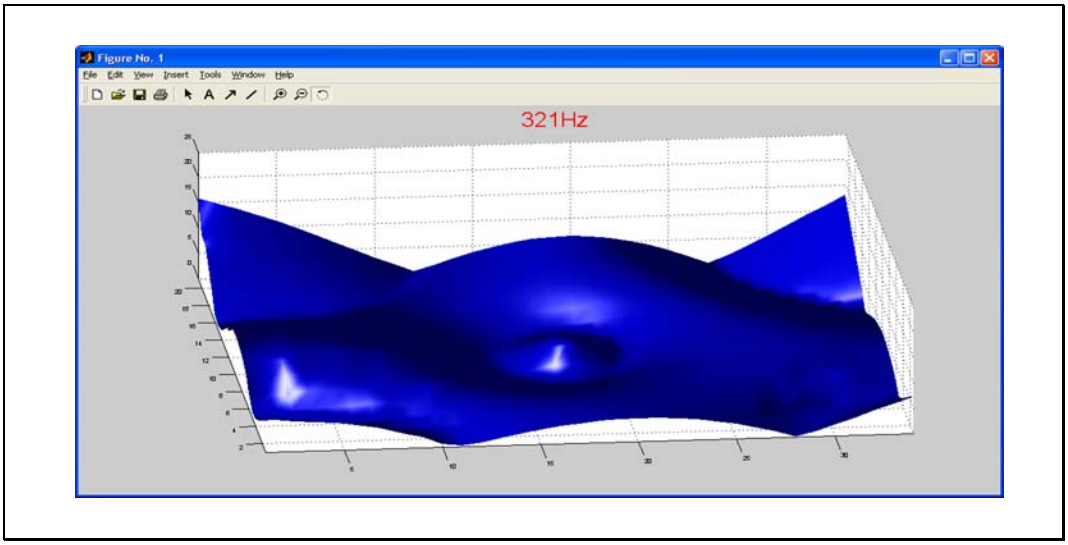


Abbildung 47: Darstellung der Schnelle für ein mit 320 Hz angeregtes Panel als dreidimensionale Oberfläche

Zur besseren Visualisierung können die in Abb. 44 bis 46 dargestellten Moden mit einem Bildbearbeitungsprogramm mit dem Messobjekt überlagert werden (siehe Abb. 48).

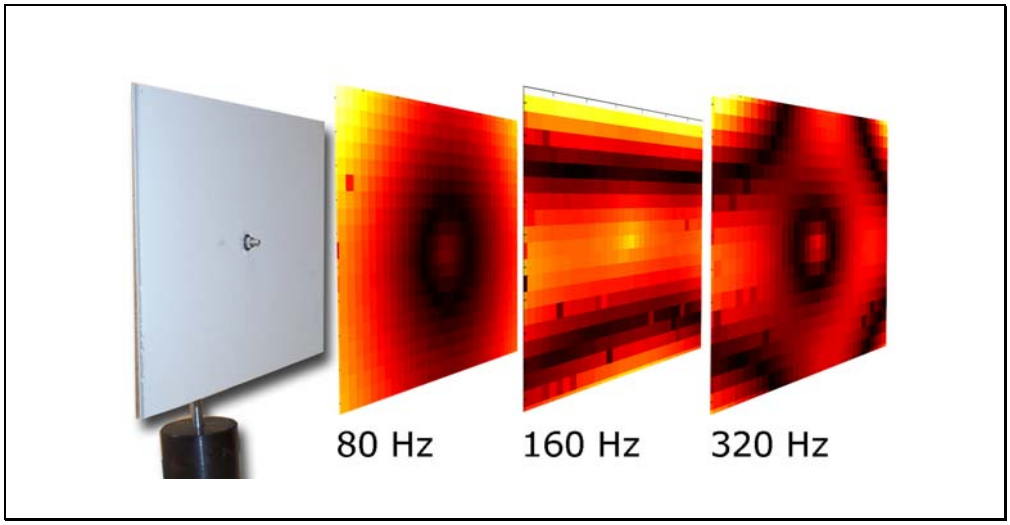


Abbildung 48: Montage der Ergebnisse mit einem Bildbearbeitungsprogramm

5.4 Scan eines Breitbandlautsprechers

In der zweiten Messung mit dem Scanning-Laservibrometer wurde ein 10cm-Breitbandlautsprecher der Firma Visaton abgescannt. Der Lautsprecher wurde an einen Frequenzgenerator mit der eingestellten Frequenz 9000 Hz angeschlossen und zunächst das Schnellesignal für einen einzigen Messpunkt in der Mitte des Lautsprechers mit dem SLDV-Positionierungs-Programm eingelesen (siehe Abb. 49). Um eine höhere Signalstärke zu erreichen, wurde die schwarze Membran weiß lackiert, was allerdings das Messergebnis verfälschen könnte.

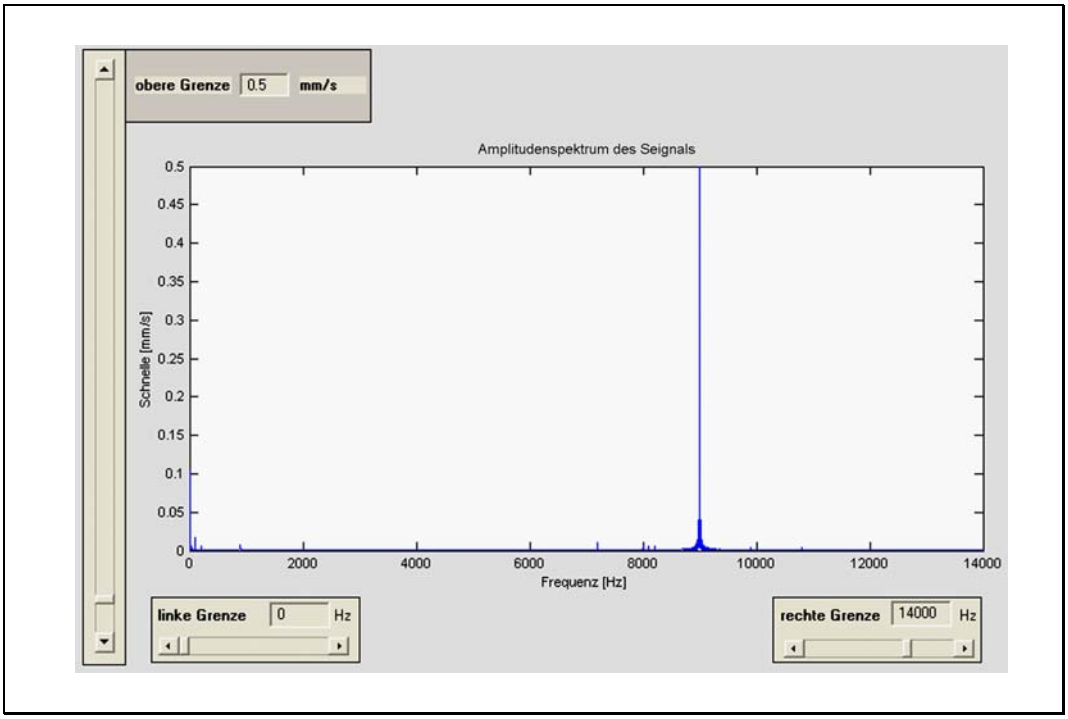


Abbildung 49: Amplitudenspektrum des Signals eines einzelnen Messpunktes

Als linke Grenze wurde anschließend 8500 Hz und als rechte Grenze 9500 Hz eingestellt, damit die Datenmenge beim Scannen und beim Erstellen der Messwerte-Matrix nicht zu groß wird. Der Abstand der Messpunkte beim Scannen betrug horizontal und vertikal jeweils 2 mm, die Gesamtzahl der

eingescannten Punkte 3224. Diese hohe Anzahl der Messpunkte, der geringe Abstand zwischen ihnen und die Länge des eingelesenen Signals von 1 Sekunde waren der Grund für die Dauer des Scans von ca. 2 Stunden. Generell kann man die Dauer eines Scans abschätzen, indem man die Gesamtzahl der Messpunkte mit der doppelten Signallänge in Sekunden multipliziert.

Das Ergebnis dieses Scans ist in den Abb. 50 und 51 dargestellt.

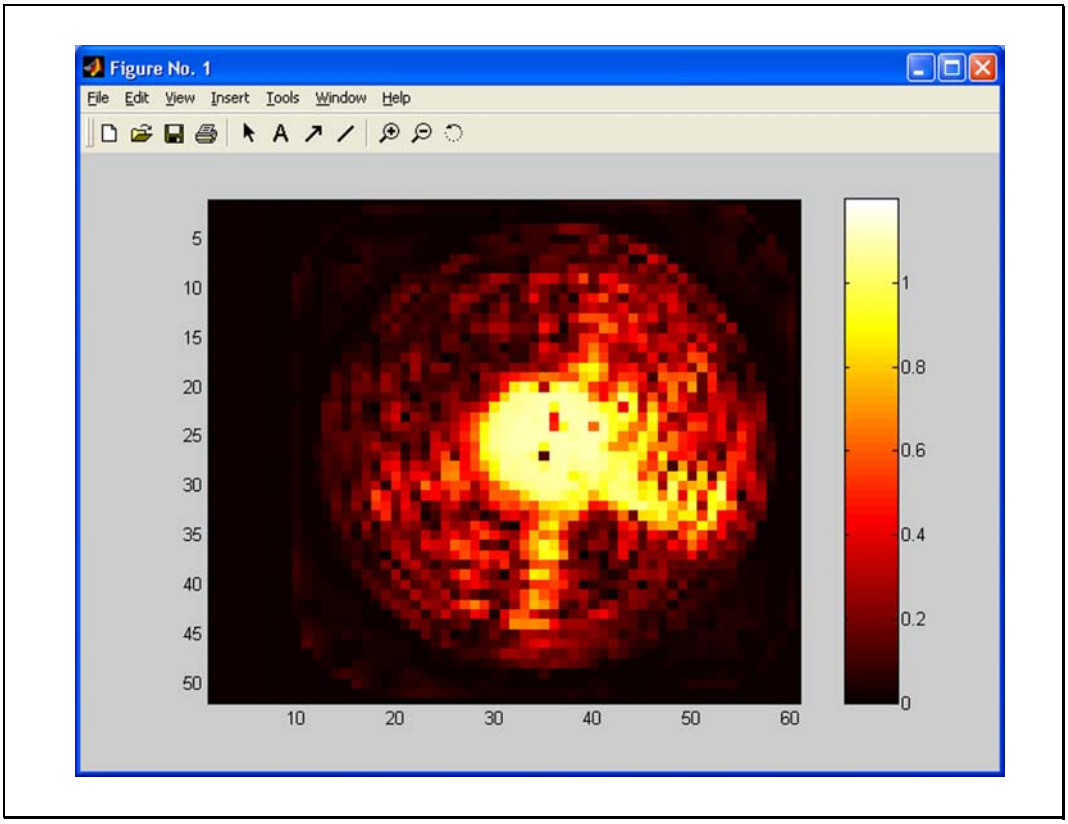


Abbildung 50: Ergebnis des Scans eines Breitbandlautsprechers, Dargestellt ist die Frequenz 9000 Hz

Gut zu erkennen sind hier die etwas unregelmäßigen sternförmigen Moden auf der Lautsprechermembran und das Zentrum mit der größten Schnelle auf der Dustcap in der Mitte.

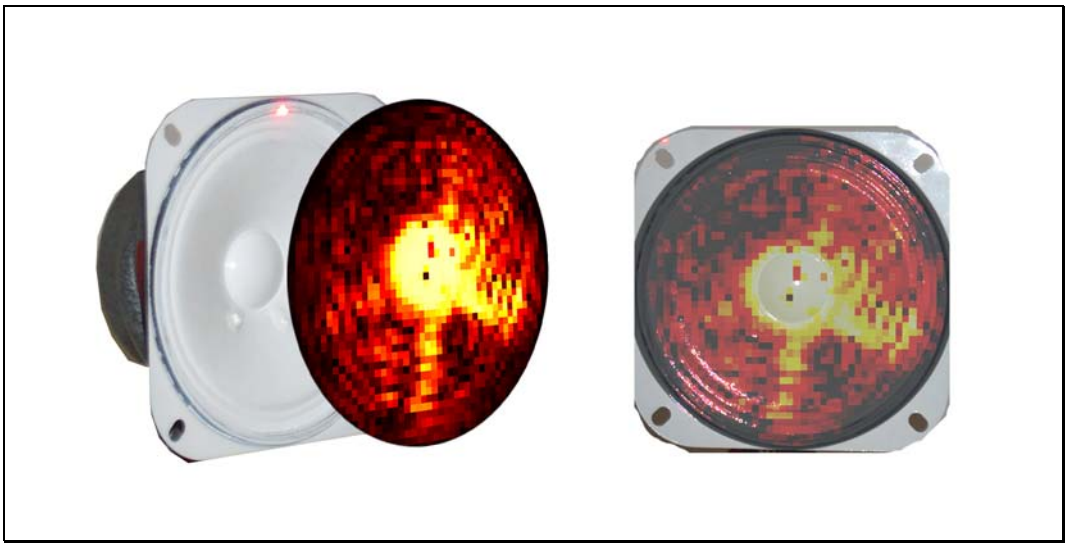


Abbildung 51: Montage des Scan-Ergebnisses mit einem Bild des Lautsprechers

5.5 Scan eines PC-Gehäuses

In der dritten Messung mit dem Scanning-Laservibrometer wurde eine Seite eines PC-Gehäuses gemessen, in dem lediglich der Netzteil- und der Prozessor-Lüfter liefen. Die Plattenlaufwerke wurden im Messzeitraum nicht angesteuert und trugen somit auch nicht zum Ergebnis bei. Die Gehäuseseite wurde in 1505 Punkten abgescannt, die jeweils einen horizontalen und vertikalen Abstand von 10 mm zueinander hatten.

Mit Hilfe des SLDV-Positionierungsprogramms wurde zunächst das Amplitudenspektrum der Schwingungen in einem einzigen Punkt erfasst, um abschätzen zu können, in welchem Frequenzbereich am besten zu messen ist (siehe Abb. 52).

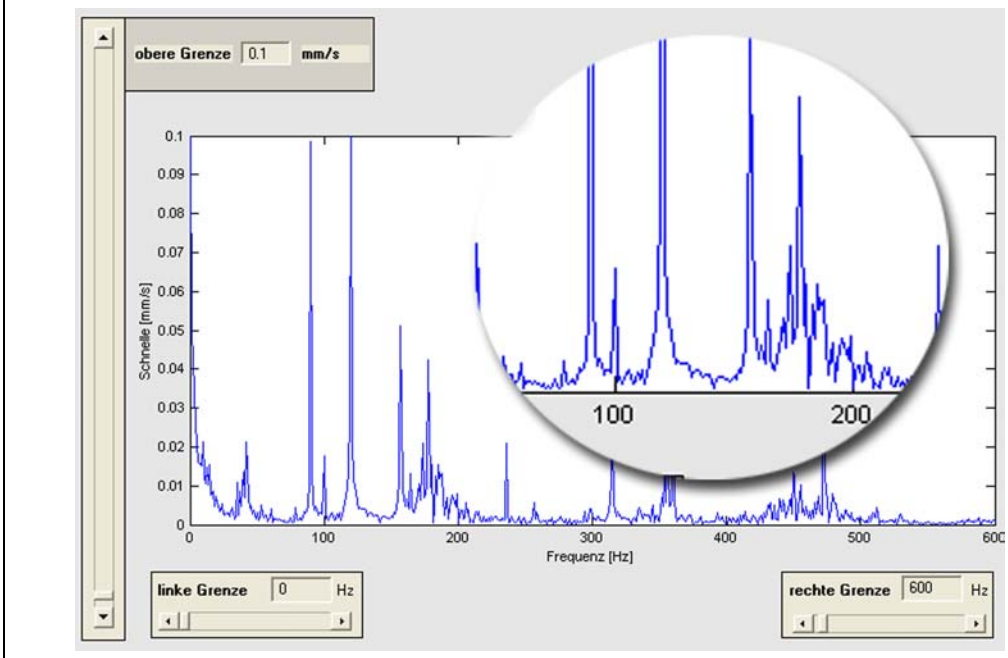


Abbildung 52: Amplitudenspektrum des Signals eines einzelnen Messpunktes auf dem PC-Gehäuse

Aus dieser Abbildung ergibt sich, dass die ersten drei deutlichsten Peaks bei ca. 90, 100 und 120 Hz zu finden sind. Die rechte Grenze wurde mit 600 Hz gewählt, darüber hinaus war kaum noch eine Schnelle-Peak zu erkennen.

Der anschließende Scan der Oberfläche erfolgte mit einer gewählten Signallänge von 1 Sekunde, so dass sich eine Frequenzauflösung von 1 Hz ergibt.

Nach Abschluss des Scans konnten tatsächlich bei den vermuteten Frequenzen Moden festgestellt werden, die sich auf der Gehäuseoberfläche eingestellt hatten.

Diese sind in den folgenden Abbildungen 53-55 dargestellt.

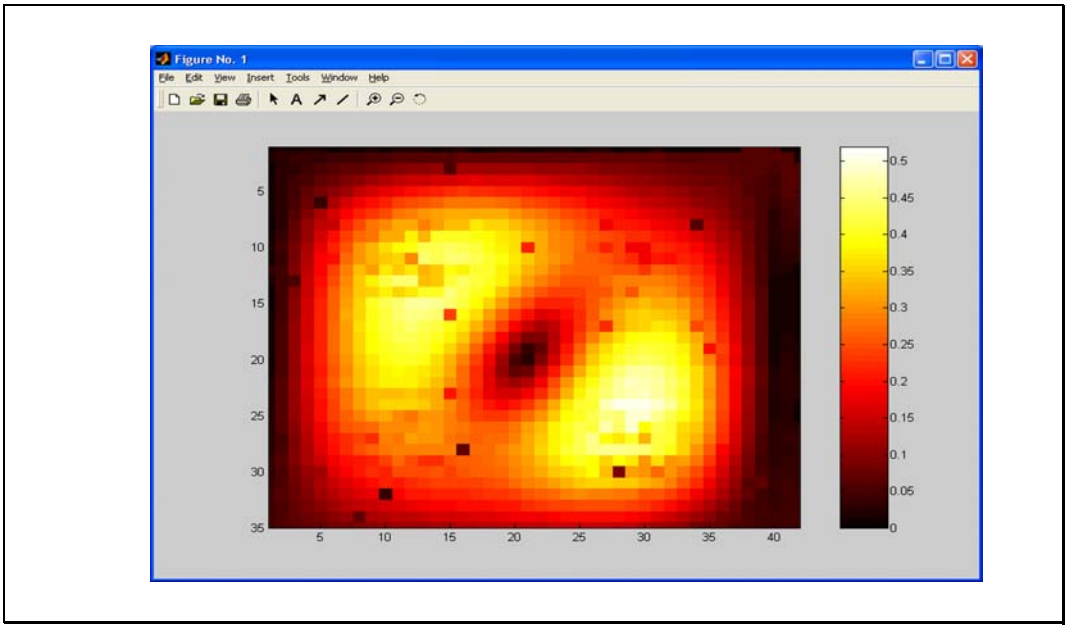


Abbildung 53: Ergebnis des Scans einer PC-Gehäuseseite, hier für 91 Hz

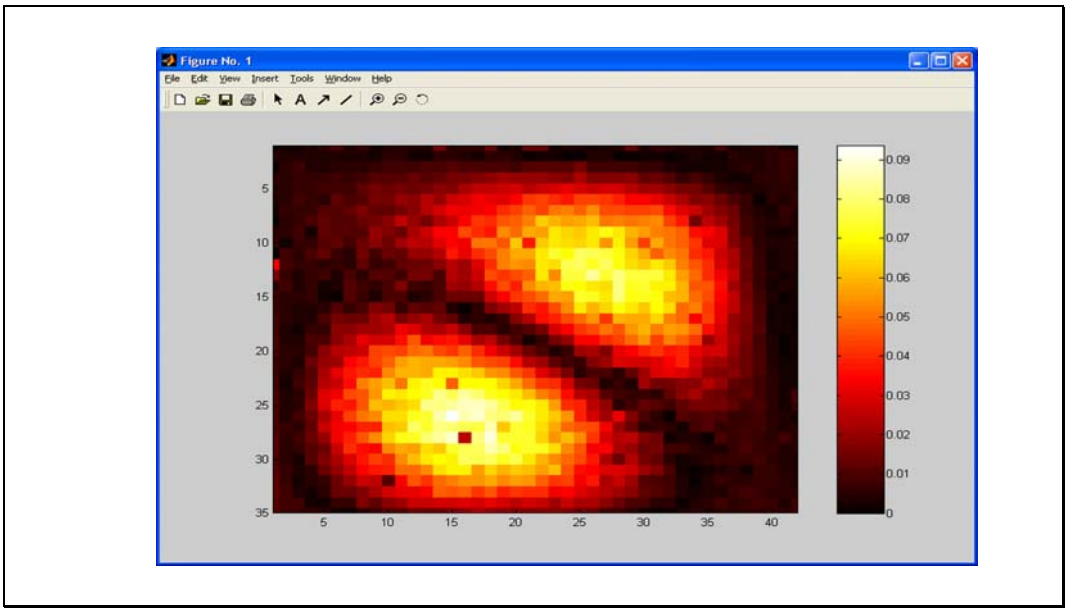


Abbildung 54: Ergebnis des Scans einer PC-Gehäuseseite, hier für 101 Hz

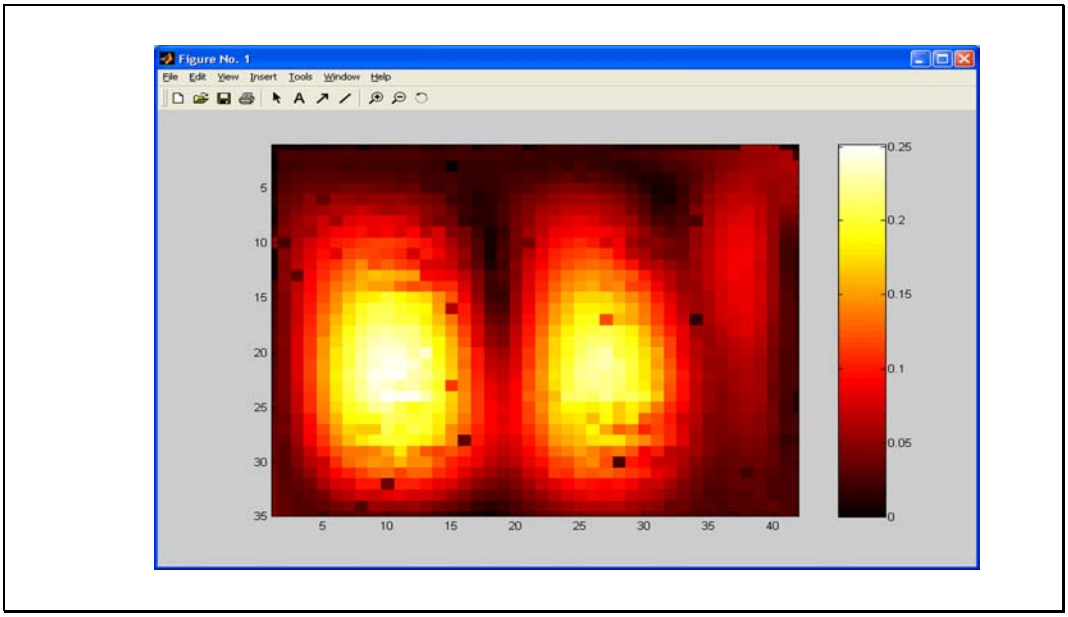


Abbildung 55: Ergebnis des Scans einer PC-Gehäuseseite, hier für 121 Hz

Stellt man diese drei Moden zusammen mit dem PC-Gehäuse dar, ist gut zu erkennen, dass in der Gehäusemitte für die drei angezeigten Frequenzen die Schnelle nahezu gleich null ist (siehe Abb. 56).

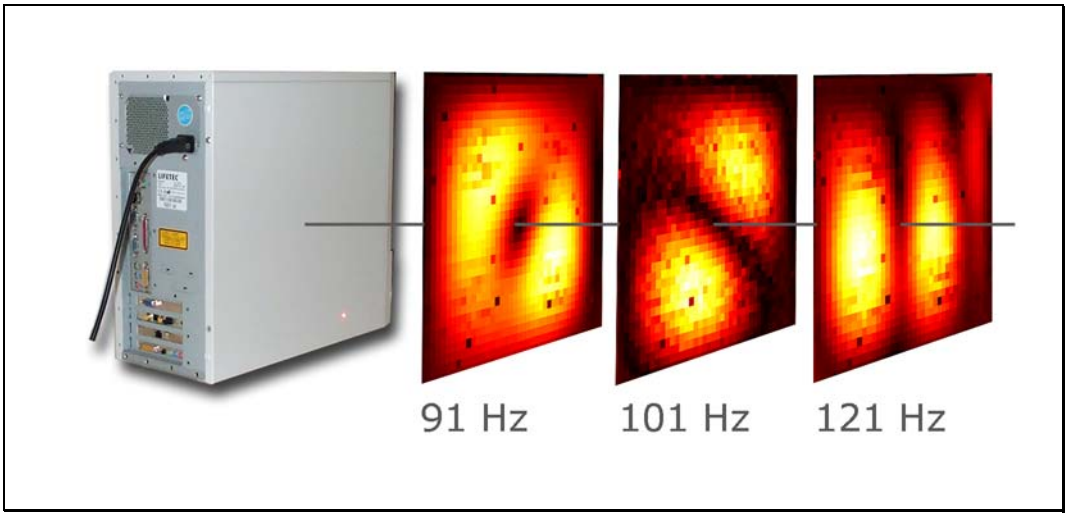


Abbildung 56: Montage des Scan-Ergebnisses mit dem Bild des PC-Gehäuses

5.6 Fehlerbetrachtung

Auffällig in allen vorigen Abbildungen sind vereinzelte Messpunkte, die zu hell oder zu dunkel sind - man könnte diese als „Fehlpixel“ oder „Fehlmessungen“ bezeichnen. Ihre Ursache dürfte in einem unsauberen Eingangssignal liegen, das evtl. auf die Bewegung des Messkopfes oder auf Störungen auf der Oberfläche zurückzuführen ist. Auf Wunsch können diese Fehlmessungen ausgeblendet werden, dazu sind in der Matlab-Funktion „scan_autoscan“ (siehe Anhang 5.8 auf Seite 192) die entsprechenden Zeilen 367-370 zu „ent“-kommentieren.

Bei der Messung der Schnelle einer Oberfläche tritt weiterhin eine Abweichung auf, die auf der Berechnung der FFT mit einem Rechteck-Fenster beruht. Misst man beispielsweise die Schnelle eines Vibrations-Kalibrators, ergibt sich durch einen „unscharfen“ Peak im Amplitudenspektrum nie der Sollwert von 10 mm/s. Die Werte schwanken hier zwischen 8 und 9,7 mm/s. Diese Abweichung ist nicht zu korrigieren, sie sollte jedoch jedem bekannt sein, der ein Amplitudenspektrum über eine FFT berechnet. Durch die Multiplikation des Messsignals mit einem anderen als einem Rechteckfenster könnte das Spektrum „zusammengeholt“ werden, die „Verschmierung“ würde geringer, der Hauptpeak dafür jedoch breiter.

Die Positionierung des Messkopfes erfolgt auf Grundlage der in Abschnitt 4.1.2 gefundenen Näherungsgleichungen - die Genauigkeit wurde dort bereits in den Tabellen mit 0,0304° für die horizontale Bewegung und 0,0274° für die vertikale Bewegung angegeben. Für die in diesem Abschnitt durchgeführten Messungen ergibt sich also ein Positionierungsfehler von 0,53 mm in der Horizontalen und 0,48 mm in der Vertikalen bei einem Messabstand von ca. 1 m.

Ein weiterer Fehler, der sich bei der Positionierung ergibt, resultiert aus der Drehbewegung des Messkopfes. Stellt man sich vor, dass der Messkopf deutlich nach oben geneigt misst, wird in der horizontalen Bewegung auf einem

entfernten, ebenen Messobjekt eine Ellipse beschrieben (Projektion einer Kreisbahn). Diese Bewegung führt bei großen Scanwinkeln in der Vertikalen in Verbindung mit einem kurzen Abstand zum Messobjekt zu einer Verzerrung des Scanbildes. Dieser Fehler wurde in der vorliegenden Programmversion noch nicht korrigiert, da er sehr gering ist und bei einer Vergrößerung des Messabstandes minimiert werden kann. Werden hingegen große Messobjekte in relativ geringem Abstand eingescannt, sollte hier eine Korrektur erfolgen.

Die Abweichung beträgt bei einem Messabstand von 4 m, einem horizontalen Scanwinkel von ca. ± 20° und einem vertikalen Winkel des Messkopfes zur Ausgangsposition von ca. 10° nur wenige Zentimeter (siehe Abb. 57).

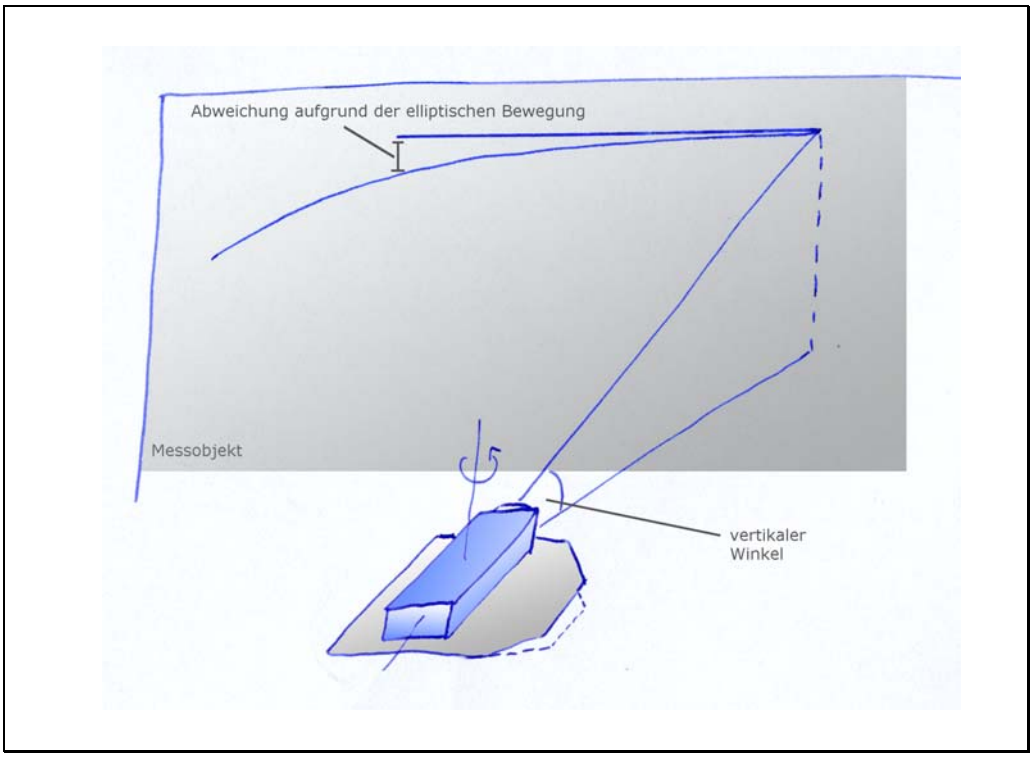


Abbildung 57: Darstellung des Positionierungsfehlers aufgrund der Drehbewegung des Messkopfes

Kapitel 6

Ergebnis und Ausblick

Das Ziel dieser Examensarbeit war der Aufbau eines kostengünstigen Scanning-Laservibrometers zur Visualisierung schwingender Oberflächen mit möglichst einfachen Mitteln und unter Zuhilfenahme vorhandener Geräte und Materialien. Dass dieses Ziel erreicht worden ist, zeigen die in Kapitel 5 durchgeführten Scans. Mit dem konstruierten Gerät können aufgrund der einfach gestalteten Programmoberfläche auf Grundlage der Software Matlab Messungen von Schwingungen auf Oberflächen durchgeführt und leicht ausgewertet werden.

Die Dauer einer Messung ist hauptsächlich von der Länge des Signals abhängig, das über die Soundkarte eingelesen und gespeichert wird. Ein Scan mit ca. 3000 Punkten und einer Frequenzauflösung von 1 Hz (also einer Signallänge von 1 Sekunde) dauert etwa zwei Stunden. Das kommerziell vertriebene Polytec Scanning-Vibrometer PSV-400 benötigt bei dieser Frequenzauflösung ebenfalls 1 Sekunde pro Messpunkt.

Die Dauer eines solchen Oberflächenscans mit mehreren Tausend Punkten hat zur Folge, dass nur Objekte vermessen werden können, die in dieser Zeit möglichst konstant und für die gesamte Dauer der Messung gleichbleibend schwingen.

Ebenfalls vergleichbar mit dem Polytec Scanning-Vibrometer ist die mögliche Gesamtzahl der Scanpunkte, sofern man sich bei der Messung auf einen kleinen Frequenzbereich beschränkt. Das PSV-400 kann maximal 512 x 512, also 262144

Punkte einscannen. Diese Anzahl von Punkten würde bei dem in dieser Arbeit entwickelten Scanning-Vibrometer bei einem betrachteten Frequenzbereich von 50 Hz etwa 100 MByte Arbeitsspeicher in Anspruch nehmen. Hier können also durchaus noch mehr Punkte eingescannt werden, als mit dem kommerziellen Produkt.

Ebenso vergleichbar ist die mögliche Scanfläche. Die Scannereinheit des PSV-400 hat einen Arbeitsbereich von etwa ±20° in vertikaler und horizontaler Richtung, der Messkopf auf dem hier gebauten Positionierungsrahmen deckt einen Scanbereich von ca. ±12° in vertikaler und ca. ±32° in horizontaler Richtung ab. Wird dieser Bereich in einer Messung voll ausgenutzt, muss allerdings die Drehbewegung des Messkopfes aufgrund des Drehtellers korrigiert werden – es ergibt sich hier eine elliptische Verzerrung.

Verbesserungsvorschläge

Möchte man das hier entwickelte Scanning-Laservibrometer verbessern, bietet es sich zunächst an, die Rahmenkonstruktion zu verkleinern. Da diese aus verschraubten Aluminiumprofilen besteht, ist eine Reduzierung der Größe und damit auch des Gewichts relativ unkompliziert.

Weiterhin ist es denkbar, die Motoren der Kugelgewindevorschübe seitlich anzuflanschen, was deren Baulänge um ca. 20 cm verringert, oder diese gar so positioniert, dass das Anheben des Messkopfes über einen *waagerecht* montierten Kugelgewindevorschub realisiert wird.

Vorstellbar ist auch eine gänzliche Neukonstruktion der Positionierungseinheit. Dies kann zum einen durch eine andere Antriebsart geschehen, etwa wenn der Messkopf auf einem Hexapod montiert wird. Die in Matlab erstellten Funktionen zur Steuerung können hierfür leicht angepasst werden, da sie modular aufgebaut sind und von den anderen Programmteilen separiert sind.

Zum anderen kann die Positionierung des Laserstrahls auch über zwei Spiegel erreicht werden. Diese Möglichkeit wurde zu Beginn der Arbeit wieder verworfen, da die Vibrationen der Spiegel, verursacht durch deren Bewegung in Verbindung mit Schrittmotoren, das Messergebnis stark beeinträchtigen können.

Ein solches Scanning-Laservibrometers auf Grundlage eines Einpunkt-Laservibrometers mit Spiegeln als Positionierungselemente wurde bereits von Berthold Käferstein an der Technischen Universität Clausthal gebaut (vgl. [Käf02]). Mit Schrittmotoren aus zwei 5¼“-Laufwerken und einer handelsüblichen Schrittmotorsteuerkarte der Firma Conrad wurden in diesem Aufbau zwei Spiegel so gesteuert, dass sie den Laserstrahl horizontal und vertikal ablenken. Zur Auswertung wurde auf Funktionen der Software Matlab zurückgegriffen. Leider gibt diese Arbeit keine Auskunft über die erzielten Messergebnisse.

Abschließend kann man zusammenfassen, dass mit dem hier entwickelten Scanning-Laservibrometer Messungen durchgeführt und ausgewertet werden können. Die Dauer des Scans, die Art, Größe und das Gewicht des Aufbaus lassen jedoch nur Messungen zu, bei denen das Messobjekt ungefähr in gleicher Höhe mit dem Messkopf aufgebaut ist.

Sehr interessant war die in Kapitel 5 beschriebene Messung von Schwingungen an PC-Gehäusen. Hier ist es durchaus vorstellbar, dass diese Ergebnisse dazu benutzt werden, durch gezielte Verstärkung der Gehäuse oder durch den Einsatz von Dämmmaterialien die Schwingungen zu verringern und somit die Betriebsgeräusche von Arbeitsplatz-PCs zu minimieren.

Aufgrund der einfach gestalteten Programmoberfläche und des übersichtlichen Versuchsaufbaus ist es zudem denkbar, dass im Rahmen physikalischer Praktika Messungen mit dem Scanning-Laservibrometer durchgeführt werden. Hierbei

sollte jedoch berücksichtigt werden, dass die einzelnen Messungen lange dauern und dass ausreichende Matlab-Kenntnisse vorhanden sein sollten - diese können jedoch auch - als Lernziel formuliert - beim Umgang mit der Software erarbeitet werden. Ebenso kann anhand dieses Versuchs ein grundlegender Einblick in die Signalanalyse und -auswertung erfolgen.

Literaturverzeichnis

[Alo92] Alonso, Marcelo und Finn, Edward J.: *Physik.* (Übers.: Anneliese Schimpfl), 2. Auflage, Bonn; München; Reading, Mass. (u.a.): Addison-Wesley 1992

[Feyn97] Feynman, Richard P.: *Vorlesungen über Physik, Bd. 1 - Hauptsächlich Mechanik, Strahlung und Wärme.* 2. Auflage, München; Wien: Oldenbourg 1997

[Ger95] Gerthsen, Christian und Vogel, Helmut: *Gerthsen Physik.* 19. Auflage, Heidelberg: Springer 1997

[Poly97] Polytech GmbH (Hrsg.): *Vibrometer Operator's Manual for Polytec Vibrometer Series 3000*. Waldbronn, 1997

[Käf02] Käferstein, Berthold: *Erfassung vollflächiger Schwingungen von Feinblechstrukturen mit dem Einpunktvibrometer?*. In: IMW-Institutsmitteilung Nr. 27. TU Clausthal, 2002

Online im Internet unter: http://www.imw.tu-clausthal.de/inhalte/forschung/publikationen/mitteilungen/27/pdf/2002_14.pdf

[Kuch01] Kuchling, Horst: *Taschenbuch der Physik*. 17. Auflage, Leipzig: Fachbuchverlag 2001

[Tip94] Tipler, Paul A.: *Physik*. Aus dem Amerikan. übers. von M. Baumgartner, Hrsg. der dt. Ausg.: Dieter Gerlich und Götz Jerke, Heidelberg; Berlin; Oxford: Spektrum Akad. Verl. 1994

Internetquellen

[MMF] Metra Mess- und Frequenztechnik Radebeul (Hrsg.): *Theorie der piezoelektrischen Schwingungsmesstechnik*. Online im Internet unter: http://www.mmf.de/theorie.htm, [Stand: 22.05.2004]

[PCE] PCE Group (Hrsg.): *Online-Handel: Vibrationsmeßgeräte - Schwingungsmessgeräte*. Online im Internet unter: http://www.warensortiment.de/messtechnik/messgeraete/vibrationsmessgeraete.htm, [Stand: 22.05.2004]

[Poly-A] Polytech GmbH (Hrsg.): *Möglichkeiten und Grundlagen der Vibrometrie*. Online im Internet unter: Vibrometer University. http://www.polytec.com/ger/_files/LM_INFO_Special_2003_01_D_Suppl.pdf, [Stand 01.04.2004]

[Poly-B] Polytech GmbH (Hrsg.): *PSV-400 Datenblatt*. Online im Internet unter: LM_DS_PSV-400. http://www.polytec.com/ger/_files/LM_DS_PSV-400_2003_10_D.pdf, [Stand 22.05.2004]

Anhang 1 Tabellen zur Dreh- und Kippwinkelberechnung

Tabelle zur Berechnung der Abhängigkeit der Schrittanzahl zum Drehwinkel (horizontale x-Achsen-Bewegung):

Anzahl der Schritte	**Abstand an der Wand in cm**	**Winkel in Grad (berechnet)**
-22000	-145,2 ± 0,1	-32,3433 ± 0,0304
-21500	-142,5 .	-31,8592 .
-21000	-139,8 .	-31,3699 .
-20500	-136,9 .	-30,8387 .
-20000	-134,0	-30,3015
-19500	-131,0	-29,7395
-19000	-128,0	-29,1712
-18500	-125,0	-28,5965
-18000	-122,5	-28,1127
-17500	-118,9	-27,4083
-17000	-116,0	-26,8343
-16500	-112,7	-26,1739
-16000	-109,6	-25,5467
-15500	-106,4	-24,8923
-15000	-103,2	-24,2309
-14500	- 99,9	-23,5415
-14000	- 96,7	-22,8660
-13500	- 93,5	-22,1838
-13000	- 90,3	-21,4949
-12500	- 86,9	-20,7557
-12000	- 83,5	-20,0092
-11500	- 80,3	-19,3001
-11000	- 76,9	-18,5398
-10500	- 73,5	-17,7728
-10000	- 70,1	-16,9991
- 9500	- 66,7	-16,2189
- 9000	- 63,4	-15,4558
- 8500	- 59,9	-14,6402
- 8000	- 56,5	-13,8421
- 7500	- 53,1	-13,0384
- 7000	- 49,6	-12,2056
- 6500	- 46,1	-11,3676
- 6000	- 42,6	-10,5246
- 5500	- 39,2	- 9,7012
- 5000	- 35,7	- 8,8494
- 4500	- 32,2	- 7,9936
- 4000	- 28,6	- 7,1096
- 3500	- 25,1	- 6,2469
- 3000	- 21,5	- 5,3566
- 2500	- 18,0	- 4,4885
- 2000	- 14,4	- 3,5934

- 1500	- 10,9	- 2,7216
- 1000	- 7,3	- 1,8235
- 500	- 3,6	- 0,8995
0	0,0	0,0000
500	3,6	0,8995
1000	7,3	1,8235
1500	11,0	2,7465
2000	14,4	3,5934
2500	18,4	4,5878
3000	22,1	5,5052
3500	25,9	6,4444
4000	29,7	7,3801
4500	33,4	8,2875
5000	37,3	9,2393
5500	41,1	10,1619
6000	45,0	11,1032
6500	48,9	12,0384
7000	52,8	12,9672
7500	56,7	13,8892
8000	60,7	14,8272
8500	64,7	15,7571
9000	68,8	16,7015
9500	72,9	17,6367
10000	77,1	18,5847
10500	81,2	19,5001
11000	85,4	20,4272
11500	89,7	21,3650
12000	94,0	22,2908
12500	98,4	23,2257
13000	102,7	24,1269
13500	107,2	25,0566
14000	111,7	25,9723
14500	116,4	26,9138
15000	121,0	27,8203
15500	125,7	28,7311
16000	130,6	29,6641
16500	135,5	30,5801
17000	140,5	31,4972
17500	145,6	32,4146
18000	150,7	33,3136
18500	155,8	34,1945
19000	161,3	35,1243
19500	166,6	36,0006
20000	172,1	36,8899
20500	177,7	37,7745
21000	183,3	38,6385

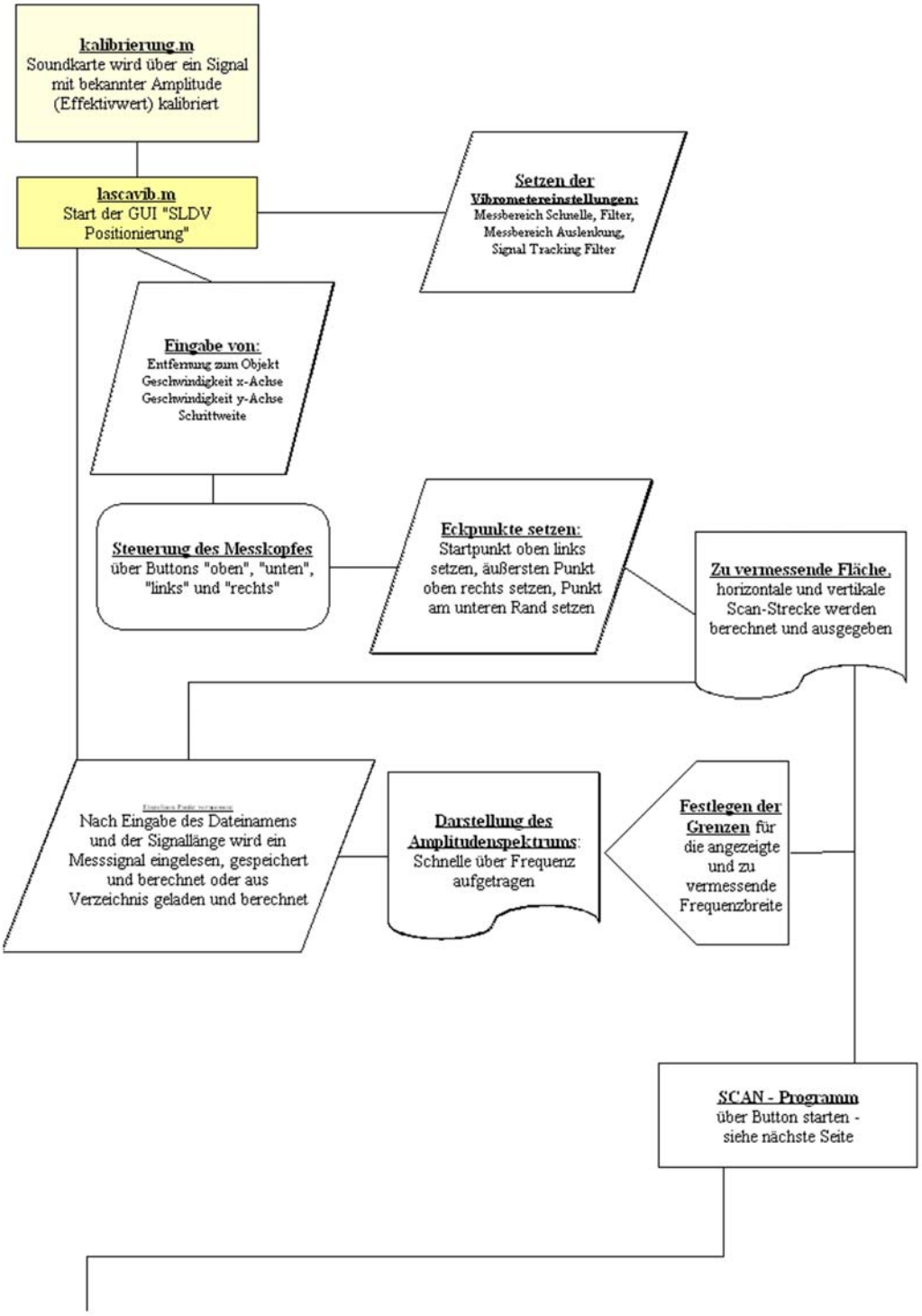
Tabelle zur Berechnung der Abhängigkeit der Schrittanzahl zum Kippwinkel (vertikale y-Achsen-Bewegung):

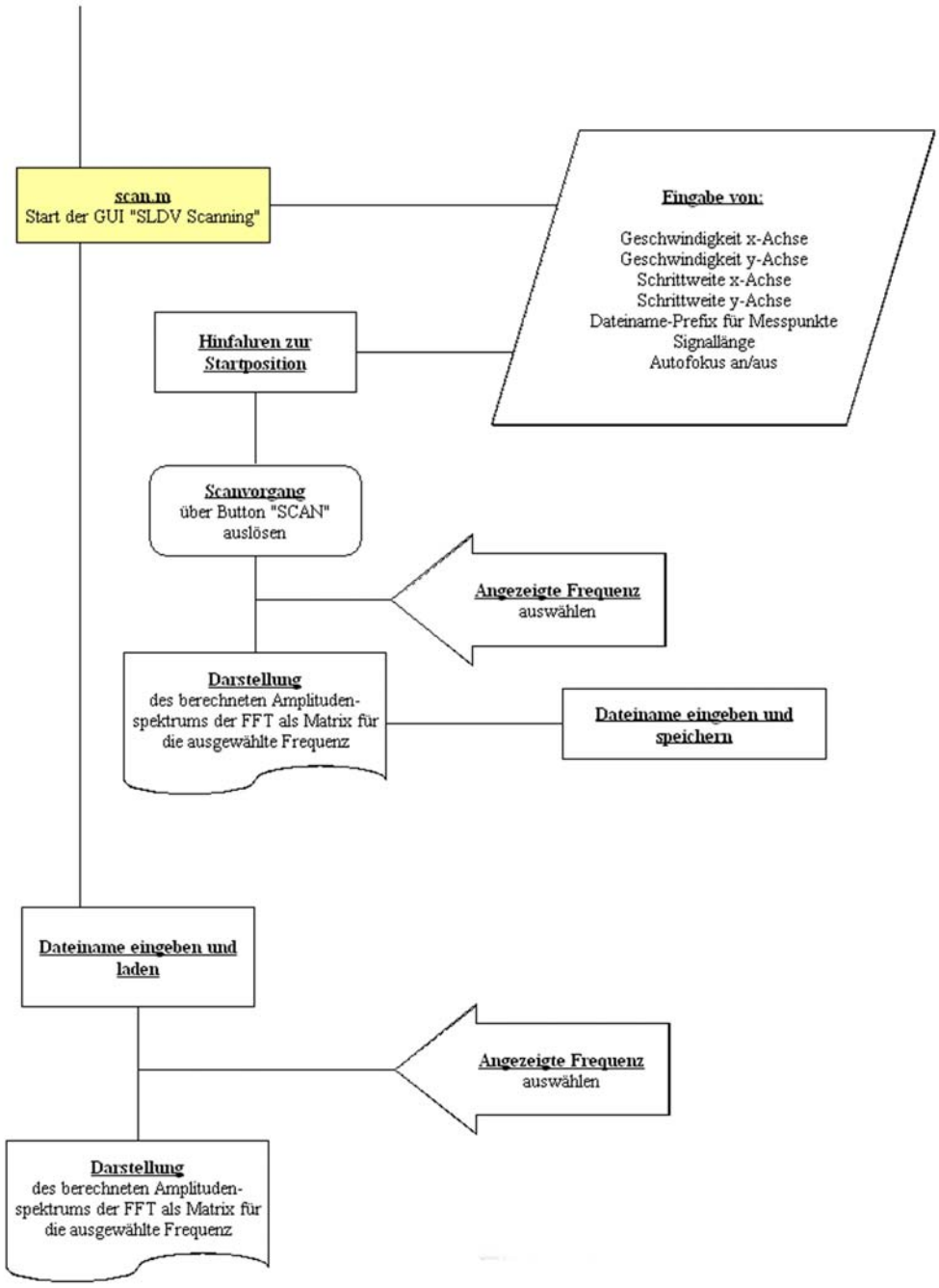
Anzahl der Schritte	**Abstand an der Wand in cm**	**Winkel in Grad (berechnet)**
20000	-64,6 ± 0,1	-15,7340 ± 0,0274
19500	-62,9 .	-15,3397 .
19000	-61,3 .	-14,9672 .
18500	-59,6 .	-14,5700 .
18000	-58,0	-14,1949
17500	-56,4	-13,8185
17000	-54,7	-13,4173
16500	-53,1	-13,0384
16000	-51,5	-12,6584
15500	-49,8	-12,2534
15000	-48,2	-11,8710
14500	-46,6	-11,4876
14000	-45,0	-11,1032
13500	-43,4	-10,7177
13000	-41,8	-10,3312
12500	-40,2	- 9,9438
12000	-38,6	- 9,5555
11500	-37,0	- 9,1663
11000	-35,4	- 8,7762
10500	-33,8	- 8,3853
10000	-32,2	- 7,9936
9500	-30,7	- 7,6257
9000	-29,0	- 7,2080
8500	-27,4	- 6,8142
8000	-25,8	- 6,4197
7500	-24,3	- 6,0493
7000	-22,6	- 5,6289
6500	-21,0	- 5,2327
6000	-19,5	- 4,8608
5500	-17,8	- 4,4388
5000	-16,2	- 4,0412
4500	-14,6	- 3,6432
4000	-13,0	- 3,2449
3500	-11,4	- 2,8462
3000	- 9,8	- 2,4473
2500	- 8,1	- 2,0231
2000	- 6,6	- 1,6487
1500	- 4,9	- 1,2242
1000	- 3,3	- 0,8245
500	- 1,6	- 0,3998
0	0,0	0,0000
- 500	1,7	0,4248
- 1000	3,4	0,8495
- 1500	5,1	1,2741
- 2000	6,8	1,6986
- 2500	8,4	2,0980
- 3000	10,1	2,5221
- 3500	11,9	2,9708
- 4000	13,6	3,3943

- 4500	15,4	3,8423
- 5000	17,2	4,2898
- 5500	19,0	4,7368
- 6000	20,9	5,2079
- 6500	22,7	5,6537
- 7000	24,6	6,1234
- 7500	26,5	6,5924
- 8000	28,4	7,0604
- 8500	30,3	7,5275
- 9000	32,3	8,0181
- 9500	34,4	8,5320
-10000	36,5	9,0445
-10500	38,7	9,5798
-11000	41,0	10,1376
-11500	43,4	10,7177
-12000	46,0	11,3436
-12500	48,8	12,0145

Anhang 2 Übersicht über die Programmabläufe

Auf den folgenden beiden Seiten ist eine schematische Übersicht der Programmabläufe dargestellt.


kalibrierung.m
Soundkarte wird über ein Signal mit bekannter Amplitude (Effektivwert) kalibriert
lascavib.m
Start der GUI "SLDV Positionierung"
Setzen der Vibrometereinstellungen:
Messbereich Schnelle, Filter, Messbereich Auslenkung, Signal Tracking Filter
Eingabe von:
Entfernung zum Objekt
Geschwindigkeit x-Achse
Geschwindigkeit y-Achse
Schrittweite
Steuerung des Messkopfes
über Buttons "oben", "unten", "links" und "rechts"
Eckpunkte setzen:
Startpunkt oben links setzen, äußersten Punkt oben rechts setzen, Punkt am unteren Rand setzen
Zu vermessende Fläche,
horizontale und vertikale Scan-Strecke werden berechnet und ausgegeben
Nach Eingabe des Dateinamens und der Signallänge wird ein Messsignal eingelesen, gespeichert und berechnet oder aus Verzeichnis geladen und berechnet
Darstellung des Amplitudenspektrums:
Schnelle über Frequenz aufgetragen
Festlegen der Grenzen für die angezeigte und zu vermessende Frequenzbreite
SCAN - Programm
über Button starten - siehe nächste Seite


scan.m
Start der GUI "SLDV Scanning"
Eingabe von:
Geschwindigkeit x-Achse
Geschwindigkeit y-Achse
Schrittweite x-Achse
Schrittweite y-Achse
Dateiname-Prefix für Messpunkte
Signallänge
Autofokus an/aus
Hinfahren zur Startposition
Scanvorgang über Button "SCAN" auslösen
Angezeigte Frequenz auswählen
Darstellung des berechneten Amplituden-spektrums der FFT als Matrix für die ausgewählte Frequenz
Dateiname eingeben und speichern
Dateiname eingeben und laden
Angezeigte Frequenz auswählen
Darstellung des berechneten Amplituden-spektrums der FFT als Matrix für die ausgewählte Frequenz

Anhang 3 kalibrierung.m

Mit diesem Kalibrierungsprogramm wird die Soundkarte des verwendeten PCs einmalig kalibriert. Das hier aufgeführte Programm kalibrierung.m beinhaltet neben den verwendeten globalen Variablen alle verwendeten Routinen, der einzige separate Teil ist die Funktion save_kalibrierung.m, die im Unterabschnitt 3.1 aufgeführt ist.

```
function varargout = kalibrierung(varargin)

% ================================================================
% ||                                                            ||
% ||           /       __  __       \    /     _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% KALIBRIERUNG Application M-file for kalibrierung.fig
%    FIG = KALIBRIERUNG launch kalibrierung GUI.
%    KALIBRIERUNG('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 27-May-2004 14:54:33


global  effektivspannung...
        kalsignal...
        kalibsignal...
        kalibmax...
        kalibmin...
        peaktopeak...
        eff_sig...
        Kali_Pyy

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
```

```
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                if (nargout)
                        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL
switchyard
                else
                        feval(varargin{:}); % FEVAL switchyard
                end
        catch
                disp(lasterr);
        end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.



% --------------------------------------------------------------------
function varargout = effektivspannung_Callback(h, eventdata, handles, varargin)
global  effektivspannung
effektivspannung = str2num(get(handles.effektivspannung,'String'));



% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
```

```
global  kalsignal
% ---------------------------------------------------------
% Speichern von fünf 5-sekündigen Signalen in wave-Files
% ---------------------------------------------------------
Fs = 44100;
kalsignal = wavrecord(5*Fs,Fs);
wavwrite(kalsignal,Fs,'kalsignal_1.wav');
kalsignal = wavrecord(5*Fs,Fs);
wavwrite(kalsignal,Fs,'kalsignal_2.wav');
kalsignal = wavrecord(5*Fs,Fs);
wavwrite(kalsignal,Fs,'kalsignal_3.wav');
kalsignal = wavrecord(5*Fs,Fs);
wavwrite(kalsignal,Fs,'kalsignal_4.wav');
kalsignal = wavrecord(5*Fs,Fs);
wavwrite(kalsignal,Fs,'kalsignal_5.wav');

% ---------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
global  effektivspannung...
        kalsignal...
        kalibsignal...
        kalibmax...
        kalibmin...
        peaktopeak
% ---------------------------------------------------------
% Einlesen der wave-Files
% ---------------------------------------------------------
fid            = fopen('kalsignal_1.wav', 'r');      % Handle erzeugen
fseek(fid, 22, 'bof');                              % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                              % springe zum 44. Byte
kalibsignal_1   = fread(fid, inf, 'int16');         % lese ab hier Messsignal
fclose(fid);                                        % fid schließen

fid            = fopen('kalsignal_2.wav', 'r');      % Handle erzeugen
fseek(fid, 22, 'bof');                              % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                              % springe zum 44. Byte
kalibsignal_2  = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                        % fid schließen

fid            = fopen('kalsignal_3.wav', 'r');      % Handle erzeugen
fseek(fid, 22, 'bof');                              % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                              % springe zum 44. Byte
kalibsignal_3  = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                        % fid schließen

fid            = fopen('kalsignal_4.wav', 'r');      % Handle erzeugen
fseek(fid, 22, 'bof');                              % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                              % springe zum 44. Byte
kalibsignal_4  = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                        % fid schließen

fid            = fopen('kalsignal_5.wav', 'r');      % Handle erzeugen
fseek(fid, 22, 'bof');                              % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                              % springe zum 44. Byte
kalibsignal_5  = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                        % fid schließen

% ---------------------------------------------------------
```

```
% Länge des ersten Signals
% --------------------------------------------------------
N              = length(kalibsignal_1);

% --------------------------------------------------------
% Berechnen von Min und Max der Signale
% --------------------------------------------------------
kalibmax_1       = max(kalibsignal_1);                % Maximum
kalibmin_1      = min(kalibsignal_1);                 % Minimum
differenz_ampl_1 = kalibmax_1+kalibmin_1;

kalibmax_2       = max(kalibsignal_2);                % Maximum
kalibmin_2      = min(kalibsignal_2);                 % Minimum
differenz_ampl_2 = kalibmax_2+kalibmin_2;

kalibmax_3       = max(kalibsignal_3);                % Maximum
kalibmin_3      = min(kalibsignal_3);                 % Minimum
differenz_ampl_3 = kalibmax_3+kalibmin_3;

kalibmax_4       = max(kalibsignal_4);                % Maximum
kalibmin_4      = min(kalibsignal_4);                 % Minimum
differenz_ampl_4 = kalibmax_4+kalibmin_4;

kalibmax_5       = max(kalibsignal_5);                % Maximum
kalibmin_5      = min(kalibsignal_5);                 % Minimum
differenz_ampl_5 = kalibmax_5+kalibmin_5;

% --------------------------------------------------------
% Verschieben der Signale in y-Richtung, s.d. Min = Max
% wegen Soundkarten-Offset
% --------------------------------------------------------
kalibsignal_1  = kalibsignal_1-0.5*differenz_ampl_1;
kalibsignal_2  = kalibsignal_2-0.5*differenz_ampl_2;
kalibsignal_3  = kalibsignal_3-0.5*differenz_ampl_3;
kalibsignal_4  = kalibsignal_4-0.5*differenz_ampl_4;
kalibsignal_5  = kalibsignal_5-0.5*differenz_ampl_5;

% --------------------------------------------------------
% Zur Kontrolle nochmal Min und Max des ersten Signals
% auslesen und in Programmoberfläche anzeigen lassen
% --------------------------------------------------------
max_wav=max(kalibsignal_1);
min_wav=min(kalibsignal_1);

set(handles.minimum,'String',min_wav);
set(handles.maximum,'String',max_wav);

axes(handles.wave)
plot(kalibsignal_1)
xlim([0 1000]);



% --------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
global  effektivspannung...
        kalsignal...
        kalibsignal...
        kalibmax...
        kalibmin...
        peaktopeak...
        eff_sig...
        Kali_Pyy

% --------------------------------------------------------
% Einlesen der wave-Files
% --------------------------------------------------------
```

```
fid             = fopen('kalsignal_1.wav', 'r');     % Handle erzeugen
fseek(fid, 22, 'bof');                               % überspringe ersten 22 Byte
numchannels     = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate      = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                               % springe zum 44. Byte
kalibsignal_1   = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                         % fid schließen

fid             = fopen('kalsignal_2.wav', 'r');     % Handle erzeugen
fseek(fid, 22, 'bof');                               % überspringe ersten 22 Byte
numchannels     = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate      = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                               % springe zum 44. Byte
kalibsignal_2   = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                         % fid schließen

fid             = fopen('kalsignal_3.wav', 'r');     % Handle erzeugen
fseek(fid, 22, 'bof');                               % überspringe ersten 22 Byte
numchannels     = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate      = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                               % springe zum 44. Byte
kalibsignal_3   = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                         % fid schließen

fid             = fopen('kalsignal_4.wav', 'r');     % Handle erzeugen
fseek(fid, 22, 'bof');                               % überspringe ersten 22 Byte
numchannels     = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate      = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                               % springe zum 44. Byte
kalibsignal_4   = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                         % fid schließen

fid             = fopen('kalsignal_5.wav', 'r');     % Handle erzeugen
fseek(fid, 22, 'bof');                               % überspringe ersten 22 Byte
numchannels     = fread(fid, 1, 'int16');            % lese numchannels aus datei
samplerate      = fread(fid, 1, 'int32');            % lese samplerate
fseek(fid, 44, 'bof');                               % springe zum 44. Byte
kalibsignal_5   = fread(fid, inf, 'int16');          % lese ab hier Messsignal
fclose(fid);                                         % fid schließen

% ---------------------------------------------------------
% Länge des ersten Signals
% ---------------------------------------------------------
N               = length(kalibsignal_1);

% ---------------------------------------------------------
% Berechnen von Minimum und Maximum
% ---------------------------------------------------------
kalibmax_1       = max(kalibsignal_1);               % Maximum
kalibmin_1      = min(kalibsignal_1);                % Minimum
differenz_ampl_1 = kalibmax_1+kalibmin_1;

kalibmax_2       = max(kalibsignal_2);               % Maximum
kalibmin_2      = min(kalibsignal_2);                % Minimum
differenz_ampl_2 = kalibmax_2+kalibmin_2;

kalibmax_3       = max(kalibsignal_3);               % Maximum
kalibmin_3      = min(kalibsignal_3);                % Minimum
differenz_ampl_3 = kalibmax_3+kalibmin_3;

kalibmax_4       = max(kalibsignal_4);               % Maximum
kalibmin_4      = min(kalibsignal_4);                % Minimum
differenz_ampl_4 = kalibmax_4+kalibmin_4;

kalibmax_5       = max(kalibsignal_5);               % Maximum
kalibmin_5      = min(kalibsignal_5);                % Minimum
differenz_ampl_5 = kalibmax_5+kalibmin_5;
```

```
% --------------------------------------------------------
% Effektivwert des 1. Zeitsignals ausrechnen
% --------------------------------------------------------
eff_sig        = sqrt(sum(kalibsignal_1.^2)./N);

% --------------------------------------------------------
% Verschieben der Signale in y-Richtung, s.d. Min = Max
% wegen Soundkarten-Offset
% --------------------------------------------------------
kalibsignal_1  = kalibsignal_1-0.5*differenz_ampl_1;
kalibsignal_2  = kalibsignal_2-0.5*differenz_ampl_2;
kalibsignal_3  = kalibsignal_3-0.5*differenz_ampl_3;
kalibsignal_4  = kalibsignal_4-0.5*differenz_ampl_4;
kalibsignal_5  = kalibsignal_5-0.5*differenz_ampl_5;

% --------------------------------------------------------
% Zur Kontrolle nochmal Min und Max des ersten Signals
% auslesen und in Programmoberfläche anzeigen lassen
% --------------------------------------------------------
max_wav=max(kalibsignal_1) ;
min_wav=min(kalibsignal_1) ;

set(handles.minimum,'String',min_wav);
set(handles.maximum,'String',max_wav);

% --------------------------------------------------------
% Berechnen der FFTs und der Amplitudenspektren
% --------------------------------------------------------
Y_1    = fft(kalibsignal_1);
Y_2    = fft(kalibsignal_2);
Y_3    = fft(kalibsignal_3);
Y_4    = fft(kalibsignal_4);
Y_5    = fft(kalibsignal_5);

Kali_Pyy_1  = sqrt(sum(Y_1.*conj(Y_1)))/N;
Kali_Pyy_2  = sqrt(sum(Y_2.*conj(Y_2)))/N;
Kali_Pyy_3  = sqrt(sum(Y_3.*conj(Y_3)))/N;
Kali_Pyy_4  = sqrt(sum(Y_4.*conj(Y_4)))/N;
Kali_Pyy_5  = sqrt(sum(Y_5.*conj(Y_5)))/N;

% --------------------------------------------------------
% Wert für Skalierungsfaktor
% --------------------------------------------------------
Kali_Pyy    = (1/5)*(Kali_Pyy_1+Kali_Pyy_2+...
               Kali_Pyy_3+Kali_Pyy_4+...
               Kali_Pyy_5);


% ------------------------------------------------------------------------------
% Darstellen als Funktion von x=0 bis x=(Samplingfrequenz/2)
% ------------------------------------------------------------------------------
a = N/2;
b = a+1;
Ay   = abs(Y_1);
Ay   = 2.*Ay./N;
f = 44100*(0:a)/N;
axes(handles.fft)
plot(f,Ay(1:b),'-')
xlim([0 10000]);
ylim([0 5000]);
title('Amplitudenspektrum des Signals')
xlabel('Frequenz (Hz)')

% --------------------------------------------------------------------
function varargout = edit3_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
```

```
function varargout = edit4_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
global  effektivspannung...
        kalibmax...
        kalibmin...
        peaktopeak...
        eff_sig...
        Kali_Pyy

save_kalibrierung
```

Anhang 3.1 save_kalibrierung.m

```
function save_kalibrierung

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\     \  /   | |_)             ||
% ||         /__ /--\ __| |__ /--\     \/    | |__)            ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  effektivspannung...
        kalibmax...
        kalibmin...
        peaktopeak...
        eff_sig...
        Kali_Pyy


% ------------------------------------------------------------------------------
% Speichern der o.a. Variablen unter dem Dateinamen "dateiname_messung"
% ------------------------------------------------------------------------------
save('kalibrierung')
```

Anhang 4 lascavib.m

Mit diesem ersten der beiden Hauptprogramme wird die Programmoberfläche „SLDV Positionierung“ gestartet. Die Funktion beinhaltet die verwendeten globalen Variablen für alle eingebetteten Funktionen und die Zuordnung der einzelnen Funktionen bezüglich der Pushbuttons, Edit-Felder usw.

In diesem Abschnitt werden nun alle zum Programm „lascavib.m“ gehörigen Funktionen in der Reihenfolge ihrer Verwendung aufgeführt.

```
function varargout = lascavib(varargin)

% ================================================================
% ||                                                            ||
% ||           /       __  __       \    /     _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% LASCAVIB Application M-file for lascavib.fig
%    FIG = LASCAVIB launch lascavib GUI.
%    LASCAVIB('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 13-May-2004 14:22:46

global  signallevel...
        v_x...
        v_y...
        schritte...
        entfernung...
        summe_schritte_links...
        summe_schritte_rechts...
        summe_schritte_oben...
        summe_schritte_unten...
```

```
        linke_strecke...
        rechte_strecke...
        strecke_unten...
        strecke_oben...
        ges_schritte_li...
        ges_schritte_re...
        ges_schritte_o...
        ges_schritte_u...
        messb_schnelle...
        messb_ausl...
        filter...
        st_filter...
        messb_schnelle_string...
        messb_ausl_string...
        filter_string...
        st_filter_string...
        horizontale...
        vertikale...
        dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs...
        xlinks...
        xrechts...
        ymax...
        rueckschritte_links...
        rueckschritte_oben...
        effektivspannung...
        kalsignal...
        kalibsignal...
        kalibmax...
        kalibmin...
        eff_sig...
        Kali_Pyy...
        messb_schnelle...
        y_range...
        scan_signallevel


if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end
        load('kalibrierung')
        effektivspannung;
        kalibmax;
        kalibmin;
        eff_sig;
        Kali_Pyy

        summe_schritte_links    = 0;
        summe_schritte_rechts   = 0;
        summe_schritte_oben     = 0;
        summe_schritte_unten    = 0;
        ges_schritte_li         = 0;
        ges_schritte_re         = 0;
        ges_schritte_u          = 0;
        xlinks                  = 0;
        xrechts                 = 1000;
        ymax                    = 1;
        set(handles.edit17,'String',num2str(xlinks));
```

```
        set(handles.slider1,'Value',xlinks);
        set(handles.edit18,'String',num2str(xrechts));
        set(handles.slider2,'Value',xrechts);
        %set(handles.edit21,'String',num2str(ymax));
        %set(handles.slider3,'Value',ymax);

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                if (nargout)
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
                else
                feval(varargin{:}); % FEVAL switchyard
                end
        catch
                disp(lasterr);
        end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.


% --------------------------------------------------------------------
function varargout = popupmenu1_Callback(h, eventdata, handles, varargin)
vxsetzen


% --------------------------------------------------------------------
function varargout = popupmenu2_Callback(h, eventdata, handles, varargin)
vysetzen


% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
```

```
vysetzen
schrittesetzen
bewegung_oben


% --------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
vysetzen
schrittesetzen
bewegung_unten


% --------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
vxsetzen
schrittesetzen
bewegung_links


% --------------------------------------------------------------------
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
vxsetzen
schrittesetzen
bewegung_rechts

% --------------------------------------------------------------------
function varargout = edit1_Callback(h, eventdata, handles, varargin)
schrittesetzen


% --------------------------------------------------------------------
function varargout = figure1_ResizeFcn(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = popupmenu3_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = popupmenu4_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = edit2_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = edit3_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
entfernungsetzen
strecke_li_ausr
strecke_o_ausr
startpunkt_oben_li_setzen


% --------------------------------------------------------------------
function varargout = pushbutton7_Callback(h, eventdata, handles, varargin)
entfernungsetzen
strecke_re_ausr
strecke_o_ausr
horizontale
startpunkt_oben_re_setzen
```

```
% --------------------------------------------------------------------
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
entfernungsetzen
strecke_u_ausr
vertikale
flaeche_ausr
startpunkt_unten_setzen


% --------------------------------------------------------------------
function varargout = edit4_Callback(h, eventdata, handles, varargin)
entfernungsetzen


% --------------------------------------------------------------------
function varargout = radiobutton1_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = radiobutton2_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = pushbutton20_Callback(h, eventdata, handles, varargin)
manuellfocus


% --------------------------------------------------------------------
function varargout = edit5_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
linke_grenze_slider
rechte_grenze_slider
linke_grenze_edit
rechte_grenze_edit
y_slider
obere_grenze_edit
xrechts = str2double(get(handles.edit18,'String'))
xlinks  = str2double(get(handles.edit17,'String'))
ymax    = max((str2double(get(handles.edit21,'String'))),0.00000001)
scan


% --------------------------------------------------------------------
function varargout = edit6_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = edit7_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = edit8_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = edit9_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = edit13_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = radiobutton5_Callback(h, eventdata, handles, varargin)
```

```
% --------------------------------------------------------------------
function varargout = radiobutton6_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = radiobutton7_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = radiobutton8_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = radiobutton9_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = popupmenu5_Callback(h, eventdata, handles, varargin)
einstellungen_setzen
einpunkt_berechnen
y_slider
obere_grenze_edit

% --------------------------------------------------------------------
function varargout = popupmenu6_Callback(h, eventdata, handles, varargin)
einstellungen_setzen
einpunkt_berechnen
y_slider
obere_grenze_edit

% --------------------------------------------------------------------
function varargout = popupmenu7_Callback(h, eventdata, handles, varargin)
einstellungen_setzen
einpunkt_berechnen
y_slider
obere_grenze_edit

% --------------------------------------------------------------------
function varargout = popupmenu8_Callback(h, eventdata, handles, varargin)
einstellungen_setzen
einpunkt_berechnen
y_slider
obere_grenze_edit

% --------------------------------------------------------------------
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
einstellungen_setzen
y_slider
obere_grenze_edit

% --------------------------------------------------------------------
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton21_Callback(h, eventdata, handles, varargin)
signallevellesen

% --------------------------------------------------------------------
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
resetvariablen
```

```
% --------------------------------------------------------------------
function varargout = Datei_1_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = oeffnen_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = schliessen_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = edit10_Callback(h, eventdata, handles, varargin)
horizontale

% --------------------------------------------------------------------
function varargout = edit11_Callback(h, eventdata, handles, varargin)
vertikale

% --------------------------------------------------------------------
function varargout = edit12_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = edit14_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = edit16_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
dateiname_ep_setzen
signallaenge_ep_setzen
einlesen_ep
einstellungen_setzen
einpunkt_berechnen

% --------------------------------------------------------------------
function varargout = pushbutton19_Callback(h, eventdata, handles, varargin)
dateiname_ep_setzen
signallaenge_ep_setzen
einstellungen_setzen
einpunkt_berechnen

% --------------------------------------------------------------------
function varargout = slider1_Callback(h, eventdata, handles, varargin)
linke_grenze_slider
einpunkt_berechnen

% --------------------------------------------------------------------
function varargout = slider2_Callback(h, eventdata, handles, varargin)
rechte_grenze_slider
einpunkt_berechnen

% --------------------------------------------------------------------
function varargout = edit17_Callback(h, eventdata, handles, varargin)
linke_grenze_edit
einpunkt_berechnen
```

```
% --------------------------------------------------------------------
function varargout = edit18_Callback(h, eventdata, handles, varargin)
rechte_grenze_edit
einpunkt_berechnen


% --------------------------------------------------------------------
function varargout = slider3_Callback(h, eventdata, handles, varargin)
y_slider
einpunkt_berechnen


% --------------------------------------------------------------------
function varargout = edit21_Callback(h, eventdata, handles, varargin)
obere_grenze_edit
einpunkt_berechnen
```

Anhang 4.1 vxsetzen

```
function vxsetzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\   |_  |    /\    \  /   | |_)             ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)            ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global v_x

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Schrittgeschwindig aus Dropdown-Feld auslesen und
% v_x setzen
% ---------------------------------------------------------------
v_x_ein = get(handles.popupmenu1,'Value');

if v_x_ein == 1
    v_x = 50;
elseif v_x_ein == 2
    v_x = 100;
elseif v_x_ein == 3
    v_x = 250;
elseif v_x_ein == 4
    v_x = 500;
elseif v_x_ein == 5
    v_x = 1000;
elseif v_x_ein == 6
    v_x = 2000;
elseif v_x_ein == 7
    v_x = 3000;
elseif v_x_ein == 8
    v_x = 4000;
elseif v_x_ein == 9
    v_x = 5000;
end
```

Anhang 4.2 vysetzen

```
function vysetzen

% ================================================================
% ||                                                            ||
% ||           /          __  __        \    /     _           ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers     ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global v_y

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Schrittgeschwindig aus Dropdown-Feld auslesen und
% v_y setzen
% ---------------------------------------------------------------
v_y_ein = get(handles.popupmenu2,'Value');

if v_y_ein == 1
    v_y = 50;
elseif v_y_ein == 2
    v_y = 100;
elseif v_y_ein == 3
    v_y = 250;
elseif v_y_ein == 4
    v_y = 500;
elseif v_y_ein == 5
    v_y = 1000;
elseif v_y_ein == 6
    v_y = 2000;
elseif v_y_ein == 7
    v_y = 3000;
end
```

Anhang 4.3 schrittesetzen

```
function schrittesetzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global schritte
handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Anzahl der Schritte wird aus Feld ausgelesen und in Variable 'schritte' als
% Zahl geschrieben
% ------------------------------------------------------------------------------
schritte = str2num(get(handles.edit1,'String'));
```

Anhang 4.4 bewegung_oben

```
function bewegung_oben

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _             ||
% ||          /   /\   |_  |    /\    \  /   | |_)             ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)            ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  schritte...                         % wurden zuvor aus Feld ausgelesen
        v_y...                              % wurde aus Dropdown ausgelesen
        summe_schritte_oben...
        summe_schritte_unten...


% ------------------------------------------------------------------------------
% Bewegungsvariable erzeugen
% ------------------------------------------------------------------------------
bewegung_o      =   ['@0A'...               % Bewegung initialisieren
                    int2str(0)...           % Schritte auf der x-Achse
                    ',' ...                 % (jeweils als "String")
                    int2str(v_y)...         % Geschwindigkeit x-Achse (=v_y)
                    ',' ...
                    int2str(0)...           % 1. Bewegung y-Achse
                    ',' ...
                    int2str(v_y)...         % Geschwindigkeit y-Achse
                    ',' ...
                    int2str(-schritte)...   % 2. Bewegung y-Achse
                    ',' ...
                    int2str(v_y)]           % Geschwindigkeit y-Achse


% ------------------------------------------------------------------------------
% Die aktuell ausgeführte Schrittzahl nach oben wird in einer Variable
% gespeichert
% ------------------------------------------------------------------------------
summe_schritte_oben = summe_schritte_oben + schritte


% ------------------------------------------------------------------------------
% COM2 Port initialisieren
% ------------------------------------------------------------------------------
```

```
s2 = serial('COM2');
set(s2,...
    'Baudrate',9600,...
    'Parity','none',...
    'StopBits',1,...
    'Terminator','CR/LF',...
    'Timeout',1);


% --------------------------------------------------------------------------
% Öffnen des seriellen Ports COM2
% --------------------------------------------------------------------------
fopen(s2);


% --------------------------------------------------------------------------
% Achse initialisieren
% --------------------------------------------------------------------------
fprintf(s2,'@05');
fprintf(s2,'@0C1');


% --------------------------------------------------------------------------
% Befehl zum Bewegen setzen
% --------------------------------------------------------------------------
fprintf(s2,bewegung_o);


% --------------------------------------------------------------------------
% Schließen des seriellen Ports COM2
% --------------------------------------------------------------------------
fclose(s2)
delete(s2)
clear s2
```

Anhang 4.5 bewegung_unten

```
function bewegung_unten

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _             ||
% ||          /   /\   |_  |    /\    \  /   | |_)             ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)            ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  schritte...                         % wurden zuvor aus Feld ausgelesen
        v_y...                              % wurde aus Dropdown ausgelesen
        summe_schritte_oben...
        summe_schritte_unten...


% ------------------------------------------------------------------------------
% Bewegungsvariable erzeugen
% ------------------------------------------------------------------------------
bewegung_u      =   ['@0A'...               % Bewegung initialisieren
                    int2str(0)...           % Schritte auf der x-Achse
                    ',' ...                 % (jeweils als "String")
                    int2str(v_y)...         % Geschwindigkeit x-Achse (=v_y)
                    ',' ...
                    int2str(0)...           % 1. Bewegung y-Achse
                    ',' ...
                    int2str(v_y)...         % Geschwindigkeit y-Achse
                    ',' ...
                    int2str(schritte)...    % 2. Bewegung y-Achse
                    ',' ...
                    int2str(v_y)]           % Geschwindigkeit y-Achse


% ------------------------------------------------------------------------------
% Die aktuell ausgeführte Schrittzahl nach oben wird in einer Variable
% gespeichert
% ------------------------------------------------------------------------------
summe_schritte_unten = summe_schritte_unten - schritte


% ------------------------------------------------------------------------------
% COM2 Port initialisieren
% ------------------------------------------------------------------------------
```

```
s2 = serial('COM2');
set(s2,...
    'Baudrate',9600,...
    'Parity','none',...
    'StopBits',1,...
    'Terminator','CR/LF',...
    'Timeout',1);


% --------------------------------------------------------------------------
% Öffnen des seriellen Ports COM2
% --------------------------------------------------------------------------
fopen(s2);


% --------------------------------------------------------------------------
% Achse initialisieren
% --------------------------------------------------------------------------
fprintf(s2,'@05');
fprintf(s2,'@0C1');


% --------------------------------------------------------------------------
% Befehl zum Bewegen setzen
% --------------------------------------------------------------------------
fprintf(s2,bewegung_u);


% --------------------------------------------------------------------------
% Schließen des seriellen Ports COM2
% --------------------------------------------------------------------------
fclose(s2)
delete(s2)
clear s2
```

Anhang 4.6 bewegung_links

```
function bewegung_links

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /   | |_)              ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)             ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  schritte...
        v_x...
        summe_schritte_links...
        summe_schritte_rechts

% ------------------------------------------------------------------------------
% Bewegungsvariable erzeugen
% ------------------------------------------------------------------------------
bewegung_l      =   ['@0A'...               % Bewegung initialisieren
                    int2str(-schritte)...   % Schritte auf der x-Achse
                    ','...                  % (jeweils als "String")
                    int2str(v_x)...         % Geschwindigkeit x-Achse
                    ','...
                    int2str(0)...           % 1. Bewegung y-Achse
                    ','...
                    int2str(v_x)...         % Geschwindigkeit y-Achse (=v_x)
                    ','...
                    int2str(0)...           % 2. Bewegung y-Achse
                    ','...
                    int2str(v_x)]           % Geschwindigkeit y-Achse (=v_x)


% ------------------------------------------------------------------------------
% Die aktuell ausgeführte Schrittzahl nach oben wird in einer Variable
% gespeichert
% ------------------------------------------------------------------------------
summe_schritte_links = summe_schritte_links - schritte


% ------------------------------------------------------------------------------
% COM2 Port initialisieren
% ------------------------------------------------------------------------------
s2 = serial('COM2');
```

```
set(s2,...
    'Baudrate',9600,...
    'Parity','none',...
    'StopBits',1,...
    'Terminator','CR/LF',...
    'Timeout',1);


% --------------------------------------------------------------------------------
% Öffnen des seriellen Ports COM2
% --------------------------------------------------------------------------------
fopen(s2);


% --------------------------------------------------------------------------------
% Achse initialisieren
% --------------------------------------------------------------------------------
fprintf(s2,'@05');
fprintf(s2,'@0C1');


% --------------------------------------------------------------------------------
% Befehl zum Bewegen setzen
% --------------------------------------------------------------------------------
fprintf(s2,bewegung_l);


% --------------------------------------------------------------------------------
% Schließen des seriellen Ports COM2
% --------------------------------------------------------------------------------
fclose(s2)
delete(s2)
clear s2
```

Anhang 4.7 bewegung_rechts

```
function bewegung_rechts

% ================================================================
% ||                                                            ||
% ||           /          __  __        \    /    _           ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  schritte...
        v_x...
        summe_schritte_links...
        summe_schritte_rechts

% ------------------------------------------------------------------------------
% Bewegungsvariable erzeugen
% ------------------------------------------------------------------------------
bewegung_r      =   ['@0A'...               % Bewegung initialisieren
                    int2str(schritte)...    % Schritte auf der x-Achse
                    ','...                  % (jeweils als "String")
                    int2str(v_x)...         % Geschwindigkeit x-Achse
                    ','...
                    int2str(0)...           % 1. Bewegung y-Achse
                    ','...
                    int2str(v_x)...         % Geschwindigkeit y-Achse (=v_x)
                    ','...
                    int2str(0)...           % 2. Bewegung y-Achse
                    ','...
                    int2str(v_x)]           % Geschwindigkeit y-Achse (=v_x)


% ------------------------------------------------------------------------------
% Die aktuell ausgeführte Schrittzahl nach oben wird in einer Variable
% gespeichert
% ------------------------------------------------------------------------------
summe_schritte_rechts = summe_schritte_rechts + schritte


% ------------------------------------------------------------------------------
% COM2 Port initialisieren
% ------------------------------------------------------------------------------
s2 = serial('COM2');
```

```
set(s2,...
    'Baudrate',9600,...
    'Parity','none',...
    'StopBits',1,...
    'Terminator','CR/LF',...
    'Timeout',1);


% --------------------------------------------------------------------------
% Öffnen des seriellen Ports COM2
% --------------------------------------------------------------------------
fopen(s2);


% --------------------------------------------------------------------------
% Achse initialisieren
% --------------------------------------------------------------------------
fprintf(s2,'@05');
fprintf(s2,'@0C1');


% --------------------------------------------------------------------------
% Befehl zum Bewegen setzen
% --------------------------------------------------------------------------
fprintf(s2,bewegung_r);


% --------------------------------------------------------------------------
% Schließen des seriellen Ports COM2
% --------------------------------------------------------------------------
fclose(s2)
delete(s2)
clear s2
```

Anhang 4.8 entfernungsetzen

```
function entfernungsetzen

% ================================================================
% ||                                                            ||
% ||           /          __  __        \    /     _          ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global entfernung

handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Auslesen der Entfernung aus dem Feld Edit4, Schreiben als Num in Variable
% ------------------------------------------------------------------------------
entfernung = str2num(get(handles.edit4,'String'))
```

Anhang 2.9 strecke_li_ausrechnen

```
function strecke_li_ausr

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  entfernung...
        summe_schritte_links...
        summe_schritte_rechts...
        linke_strecke...
        ges_schritte_li...

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Auslesen der Entfernung zum Objekt
% ---------------------------------------------------------------
entfernung = str2num(get(handles.edit4,'String'))

% ---------------------------------------------------------------
% Berechnen der eff. Schrittzahl nach links
% ---------------------------------------------------------------
ges_schritte_li = summe_schritte_links...
                  + summe_schritte_rechts

% ---------------------------------------------------------------
% Berechnen des Winkels aus Anzahl der Schritte
% ---------------------------------------------------------------
winkel =   8.249357569520E-23*(ges_schritte_li)^5 ...
         + 1.419873939066E-18*(ges_schritte_li)^4 ...
         - 3.720683184245E-13*(ges_schritte_li)^3 ...
         + 7.668028385937E-09*(ges_schritte_li)^2 ...
         + 1.815281965615E-03*(ges_schritte_li)

% ---------------------------------------------------------------
% Berechnen des Abstandes aus dem Winkel
% ---------------------------------------------------------------
linke_strecke = tan((2*pi*winkel/360))*entfernung
```

```
% ---------------------------------------------------------------
% Horizontale Entfernung zum Ausgangspunkt wird in
% Feld geschrieben
% ---------------------------------------------------------------
set(handles.edit7,'String',num2str(linke_strecke,4));
```

Anhang 4.10 strecke_o_ausrechnen

```
function strecke_o_ausr

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _             ||
% ||          /   /\   |_  |    /\    \  /   | |_)              ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)             ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  entfernung...
        summe_schritte_oben...
        summe_schritte_unten...
        strecke_unten...
        ges_schritte_o...
        strecke_oben...

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Auslesen der Entfernung zum Objekt
% ---------------------------------------------------------------
entfernung = str2num(get(handles.edit4,'String'))

% ---------------------------------------------------------------
% Berechnen der eff. Schrittzahl nach oben
% ---------------------------------------------------------------
ges_schritte_o = summe_schritte_unten+summe_schritte_oben


% ---------------------------------------------------------------
% Berechnen des Winkels aus Anzahl der Schritte
% ---------------------------------------------------------------
winkel = -(- 4.067378106555E-22*(ges_schritte_o)^5 ...
         + 1.615418578968E-17*(ges_schritte_o)^4 ...
         + 2.580519946375E-13*(ges_schritte_o)^3 ...
         + 3.818092031355E-09*(ges_schritte_o)^2 ...
         - 8.246903811402E-04*(ges_schritte_o))

% ---------------------------------------------------------------
% Berechnen des Abstandes aus dem Winkel
% ---------------------------------------------------------------
strecke_oben = tan(2*pi*winkel/360)*entfernung
```

```
% ---------------------------------------------------------------
% Horizontale Entfernung zum Ausgangspunkt wird in
% Feld geschrieben
% ---------------------------------------------------------------
set(handles.edit13,'String',num2str(strecke_oben,4));
```

Anhang 4.11 startpunkt_oben_li_setzen

```
function startpunkt_oben_li_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen
% ---------------------------------------------------------------
global  ges_schritte_li rueckschritte_links

% ---------------------------------------------------------------
% Schreiben der Gesamtschritte in neue Variable
% ---------------------------------------------------------------
rueckschritte_links = ges_schritte_li
```

Anhang 4.12 strecke_re_ausrechnen

```
function strecke_re_ausr

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  entfernung...
        summe_schritte_links...
        summe_schritte_rechts...
        rechte_strecke...
        ges_schritte_re...

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Auslesen der Entfernung zum Objekt
% ---------------------------------------------------------------
entfernung = str2num(get(handles.edit4,'String'))

% ---------------------------------------------------------------
% Berechnen der eff. Schrittzahl nach rechts
% ---------------------------------------------------------------
ges_schritte_re = summe_schritte_links ...
                  + summe_schritte_rechts

% ---------------------------------------------------------------
% Berechnen des Winkels aus Anzahl der Schritte
% ---------------------------------------------------------------
winkel =   8.249357569520E-23*(ges_schritte_re)^5 ...
         + 1.419873939066E-18*(ges_schritte_re)^4 ...
         - 3.720683184245E-13*(ges_schritte_re)^3 ...
         + 7.668028385937E-09*(ges_schritte_re)^2 ...
         + 1.815281965615E-03*(ges_schritte_re)

% ---------------------------------------------------------------
% Berechnen des Abstandes aus dem Winkel
% ---------------------------------------------------------------
rechte_strecke = tan(2*pi*winkel/360)*entfernung
```

```
% --------------------------------------------------------------
% Horizontale Entfernung zum Ausgangspunkt wird in
% Feld geschrieben
% --------------------------------------------------------------
set(handles.edit8,'String',num2str(rechte_strecke,4));
```

Anhang 4.13 horizontale

```
function horizontale

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global  linke_strecke...
        rechte_strecke...
        horizontale

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Berechnen der horizontalen Scanstrecke
% ------------------------------------------------------------------------------
horizontale = abs(linke_strecke)+abs(rechte_strecke)


% ------------------------------------------------------------------------------
% Schreiben der horiz. Scanstrecke in Feld Edit10 als String mit 4 Stellen
% ------------------------------------------------------------------------------
set(handles.edit10,'String',num2str(horizontale,4));
```

Anhang 4.14 startpunkt_oben_re_setzen

```
function startpunkt_oben_re_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\     \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\     \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen
% ---------------------------------------------------------------
global  ges_schritte_re...
        rueckschritte_links...
        rueckschritte_oben...
        summe_schritte_unten...
        summe_schritte_oben

% ---------------------------------------------------------------
% Schreiben der Gesamtschritte in neue Variable
% ---------------------------------------------------------------
rueckschritte_links = rueckschritte_links-ges_schritte_re
rueckschritte_oben  = summe_schritte_oben+summe_schritte_unten
```

Anhang 4.15 strecke_u_ausrechnen

```
function strecke_u_ausr

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  entfernung...
        summe_schritte_oben...
        summe_schritte_unten...
        strecke_unten...
        ges_schritte_u...

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Auslesen der Entfernung zum Objekt
% ---------------------------------------------------------------
entfernung = str2num(get(handles.edit4,'String'))

% ---------------------------------------------------------------
% Berechnen der eff. Schrittzahl nach oben
% ---------------------------------------------------------------
ges_schritte_u = summe_schritte_unten+summe_schritte_oben

% ---------------------------------------------------------------
% Berechnen des Winkels aus Anzahl der Schritte
% ---------------------------------------------------------------
winkel = -(- 4.067378106555E-22*(ges_schritte_u)^5 ...
         + 1.615418578968E-17*(ges_schritte_u)^4 ...
         + 2.580519946375E-13*(ges_schritte_u)^3 ...
         + 3.818092031355E-09*(ges_schritte_u)^2 ...
         - 8.246903811402E-04*(ges_schritte_u))

% ---------------------------------------------------------------
% Berechnen des Abstandes aus dem Winkel
% ---------------------------------------------------------------
strecke_unten = tan(2*pi*winkel/360)*entfernung
```

```
% ---------------------------------------------------------------
% Horizontale Entfernung zum Ausgangspunkt wird in
% Feld geschrieben
% ---------------------------------------------------------------
set(handles.edit9,'String',num2str(strecke_unten,4));
```

Anhang 4.16 vertikale

```
function vertikale

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global strecke_unten strecke_oben vertikale

handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Berechnen der vertiaklen Scanstrecke
% ------------------------------------------------------------------------------
vertikale = abs(strecke_unten)+abs(strecke_oben)

% ------------------------------------------------------------------------------
% Schreiben der vert. Scanstrecke in Feld Edit11 als String mit 4 Stellen
% ------------------------------------------------------------------------------
set(handles.edit11,'String',num2str(vertikale,4));
```

Anhang 4.17 flaeche_ausr

```
function flaeche_ausr

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global horizontale vertikale

handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Ausrechnen der zu scannenden Fläche in cm^2 (deswegen /100)
% ------------------------------------------------------------------------------
flaeche = horizontale*vertikale/100


% ------------------------------------------------------------------------------
% Schreiben der Fläche in das Feld Edit12 mit 4 Stellen als String
% ------------------------------------------------------------------------------
set(handles.edit12,'String',num2str(flaeche,4));
```

Anhang 4.18 startpunkt_unten_setzen

```
function startpunkt_unten_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\    \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen
% ---------------------------------------------------------------
global  ges_schritte_u...
        ges_schritte_o...
        rueckschritte_oben

% ---------------------------------------------------------------
% Schreiben der bisher ausgeführten eff. Bewegung nach unten
% in neue Variable
% ---------------------------------------------------------------
rueckschritte_oben = abs(ges_schritte_o)+abs(ges_schritte_u)
```

Anhang 4.19 manuellfocus

```
function manuellfocus

% ================================================================
% ||                                                            ||
% ||           /         __  __        \    /    _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  signallevel...
        scan_signallevel


% ------------------------------------------------------------------------------
% Voreinstellungen, die "empirisch" gefundenen Werte ttrechts und ttlinks
% sind später in der Schleife sozusagen die "Länge" für die Motorbewegung
% in eine Richtung. Da de Motor schneller nach rechts läuft als nach links,
% ist dieser Wert höher. (s.u.)
% ------------------------------------------------------------------------------
zaehler      = 0;
ttrechts     = 6.7;
ttlinks      = 9.6;


% ------------------------------------------------------------------------------
% COM1 Port initialisieren
% ------------------------------------------------------------------------------
s = serial('COM1');
set(s,'Baudrate',9600,'Parity','none','StopBits',1);
set(s,'InputBufferSize',1024,'Terminator','LF','Timeout',1);


% ------------------------------------------------------------------------------
% Öffnen der seriellen Schnittstelle COM1
% ------------------------------------------------------------------------------
fopen(s);


% ------------------------------------------------------------------------------
% Auslesen der Ausgangssignalstärke "signallevela"
% ------------------------------------------------------------------------------
fprintf(s,'ECHOON');
```

```
echo = fscanf(s);

fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);
if isempty(level)
    warndlg...
    ('Konnte den Befehl "LEV" nicht an das Vibrometer senden','!! Warnung !!')
end

fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);

for x = 0:40
    levelzahl = ['LEV',int2str(x)];
    if strcmp(level,levelzahl) == 1
    signallevela = x;
    end
end


% ------------------------------------------------------------------------------
% ------------------------- A U T O F O K U S ----------------------------------
% ------------------------------------------------------------------------------
% Die Funktionsweise ist wie folgt: Nachdem der Befehl zum Rechtsdrehen
% an den Fokussierungs-Motor gesendet wurde, wird in einer Schleife mehrmals
% die Signalstärke ausgelesen. Da dies "Zeit kostet", dreht der Motor mit einer
% bestimmten Zeit in diese Richtung. Dies ist leider notwendig, da man dem
% Motor nur die Befehle "start" und "stopp" geben kann, aber nicht, wie lange
% er laufen soll. Deswegen diese Notlösung...
% Die Bewegungen laufen wie folgt ab: erst Rechtslauf, dann Linkslauf mit den
% o.a. Schleifenlängen. Diese werden für die nächste Schleife um den Faktor
% 1.5 vergrößert, damit ein größerer Bereich in den Fokus gelangt.
% Insgesamt läuft die Fokusschleife entweder, bis die Signalstärke ungleich
% Null ist, oder bis der Zähler den Stand 4 erreicht hat, also vier
% Durchläufe
% ------------------------------------------------------------------------------
signallevel = signallevela;
    while (signallevel == 0) & (zaehler <= 4)
                        ttrechts = 1.5 * ttrechts
                        ttlinks  = 1.5 * ttlinks
                        zaehler = zaehler + 1;
                        fprintf(s,'r');
                        for ta = 0:ttrechts
                                fprintf(s,'LEV');
                                level = fscanf(s);
                                level = cellstr(level);

                                        for xa = 0:40
                                            levelzahl = ['LEV',int2str(xa)];
                                            if strcmp(level,levelzahl) == 1
                                            signallevel = xa;
                                            end
                                        end

                                if signallevel > signallevela
                                fprintf(s,'N');
                                break
                                end
                        end
                        fprintf(s,'N');

                        fprintf(s,'l');
                        for tb = 0:ttlinks
                                fprintf(s,'LEV');
                                level = fscanf(s);
                                level = cellstr(level);
```

```
                                        for xb = 0:40
                                            levelzahl = ['LEV',int2str(xb)];
                                            if strcmp(level,levelzahl) == 1
                                            signallevel = xb;
                                            end
                                        end

                                    if signallevel > signallevela
                                    fprintf(s,'N');
                                    break
                                    end
                            end
    end

    scan_signallevel = signallevel;

% --------------------------------------------------------------------------------
% Stop-Befehl an Motor senden
% --------------------------------------------------------------------------------
fprintf(s,'N');


% --------------------------------------------------------------------------------
% Signalstärke in Feld schreiben
% --------------------------------------------------------------------------------
handles = guihandles(gcbo);
set(handles.text81,'String',num2str(signallevel));

% --------------------------------------------------------------------------------
% Serielle Schnittstelle schließen
% --------------------------------------------------------------------------------
fclose(s)
delete(s)
clear s
```

Anhang 4.20 linke_grenze_slider

```
function linke_grenze_slider

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global xlinks

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Linke Grenze auf wird von Slider ausgelesen und in Edit17-Feld geschrieben
% ------------------------------------------------------------------------------
set(handles.edit17,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Wert wird erneut aus Feld ausgelesen und in Variable xlinks geschrieben
% ------------------------------------------------------------------------------
val = str2double(get(handles.edit17,'String'));
xlinks = val;
```

Anhang 4.21 rechte_grenze_slider

```
function rechte_grenze_slider

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\     \  /   | |_)             ||
% ||         /__ /--\ __| |__ /--\     \/    | |__)            ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global xrechts

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Rechte Grenze auf wird von Slider ausgelesen und in Edit18-Feld geschrieben
% ------------------------------------------------------------------------------
set(handles.edit18,'String',num2str(get(handles.slider2,'Value')));


% ------------------------------------------------------------------------------
% Wert wird erneut aus Feld ausgelesen und in Variable xrechts geschrieben
% ------------------------------------------------------------------------------
val = str2double(get(handles.edit18,'String'));
xrechts = val;
```

Anhang 4.22 linke_grenze_edit

```
function linke_grenze_edit

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global xlinks

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Linke Grenze auf x-Achse wird ausgelesen und auf den Slider übertragen
% ------------------------------------------------------------------------------
val = str2double(get(handles.edit17,'String'));
if isnumeric(val) & length(val)==1 & ...
        val >= get(handles.slider1,'Min') & val <= get(handles.slider1,'Max')
    set(handles.slider1,'Value',val);
    xlinks = val;
else
    set(handles.edit17,'String','Fehler');
end
```

Anhang 4.23 rechte_grenze_edit

```
function rechte_grenze_edit

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global xrechts

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Rechte Grenze auf x-Achse wird ausgelesen und auf den Slider übertragen
% ------------------------------------------------------------------------------
val = str2double(get(handles.edit18,'String'));
if isnumeric(val) & length(val)==1 & ...
        val >= get(handles.slider2,'Min') & val <= get(handles.slider2,'Max')
    set(handles.slider2,'Value',val);
    xrechts = val;
else
    set(handles.edit18,'String','Fehler');
end
```

Anhang 4.24 y_slider

```
function y_slider

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\     \  /   | |_)             ||
% ||         /__ /--\ __| |__ /--\     \/    | |__)            ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global  ymax...

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Obere Grenze auf y-Achse wird gesetzt
% ------------------------------------------------------------------------------
set(handles.edit21,'String',num2str(get(handles.slider3,'Value')));
val = str2num(get(handles.edit21,'String'));
ymax = max(val,0.00000001);
```

Anhang 4.25 obere_grenze_edit

```
function obere_grenze_edit

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\    \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global  ymax...
        effektivspannung...
        kalibmax...
        kalibmin...
        peaktopeak

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Obere Grenze auf x-Achse wird ausgelesen und auf den Slider übertragen
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit21,'String'));
if isnumeric(val) & length(val)==1 & ...
        val >= get(handles.slider3,'Min') & val <= get(handles.slider3,'Max')
    set(handles.slider3,'Value',val);
    ymax = val;
else
    set(handles.edit21,'String','Fehler');
end
```

Anhang 4.26 einstellungen_setzen

```
function einstellungen_setzen           % Einstellungen an das Vibrometer
                                        % senden

% ================================================================
% ||                                                            ||
% ||           /          __  __        \    /     _          ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global  messb_schnelle...
        messb_ausl...
        filter...
        st_filter

handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Dropdown-Wert für den Messbereich der Schnelle wird aus Feld ausgelesen
% ------------------------------------------------------------------------------
messb_schnelle = get(handles.popupmenu5,'Value');

if messb_schnelle == 1
    messb_schnelle = 6;                 % Wert "6" gem. Bed-Anleitung für das
elseif messb_schnelle == 2              % Vibrometer
    messb_schnelle = 7;
elseif messb_schnelle == 3
    messb_schnelle = 8;
elseif messb_schnelle == 4
    messb_schnelle = 9;
end
messb_schnelle_string = ['VELO' num2str(messb_schnelle)];


% ------------------------------------------------------------------------------
% Dropdown-Wert für den Messbereich der Auslenkung wird aus Feld ausgelesen
% ------------------------------------------------------------------------------
messb_ausl = get(handles.popupmenu6,'Value');

if messb_ausl == 1
    messb_ausl = 3;
elseif messb_ausl == 2
```

```
    messb_ausl = 4;
elseif messb_ausl == 3
    messb_ausl = 5;
elseif messb_ausl == 4
    messb_ausl = 6;
elseif messb_ausl == 5
    messb_ausl = 7;
end
messb_ausl_string = ['AMPL' num2str(messb_ausl)];


% --------------------------------------------------------------------------------
% Dropdown-Wert für den Filter wird aus Feld ausgelesen
% --------------------------------------------------------------------------------
filter = get(handles.popupmenu7,'Value');

if filter == 1
    filter = 1;
elseif filter == 2
    filter = 2;
elseif filter == 3
    filter = 3;
elseif filter == 4
    filter = 4;
end
filter_string = ['FILT' num2str(filter)];


% --------------------------------------------------------------------------------
% Dropdown-Wert für den Signal Tracking Filter wird aus Feld ausgelesen
% --------------------------------------------------------------------------------
st_filter = get(handles.popupmenu8,'Value');

if st_filter == 1
    st_filter = 1;
elseif st_filter == 2
    st_filter = 4;
elseif st_filter == 3
    st_filter = 3;
end
st_filter_string = ['TRACK' num2str(st_filter)];



% --------------------------------------------------------------------------------
% COM1 Port wird initialisiert
% --------------------------------------------------------------------------------
s   =  serial('COM1');
set(s,'Baudrate',9600,'Parity','none','StopBits',1);
set(s,'InputBufferSize',1024,'Terminator','LF','Timeout',1);

% --------------------------------------------------------------------------------
% Öffnen der seriellen Schnittstelle COM1
% --------------------------------------------------------------------------------
fopen(s);


% --------------------------------------------------------------------------------
% Senden der Einstellungen an das Vibrometer
% --------------------------------------------------------------------------------
fprintf(s,'ECHOON');
echo = fscanf(s);
fprintf(s,messb_schnelle_string)
fprintf(s,messb_ausl_string)
fprintf(s,filter_string)
fprintf(s,st_filter_string)
```

```
% --------------------------------------------------------------------------
% COM1 wird geschlossen
% --------------------------------------------------------------------------
fclose(s)
delete(s)
clear s
```

Anhang 4.27 einpunkt_berechnen

```
function einpunkt_berechnen

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _             ||
% ||          /   /\   |_  |    /\     \  /   | |_)            ||
% ||         /__ /--\ __| |__ /--\     \/    | |__)            ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ------------------------------------------------------------------------------
global  dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs...
        xlinks...
        xrechts...
        ymax...
        effektivspannung...
        kalibmax...
        kalibmin...
        peaktopeak...
        messb_schnelle...
        y_range...
        messb_ausl...
        filter...
        st_filter...
        eff_sig...
        Kali_Pyy

handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Einlesen des wave-Files
% ------------------------------------------------------------------------------
fid            = fopen(dateiname_einpunkt, 'r');   % Handle erzeugen
fseek(fid, 22, 'bof');                             % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');           % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');           % lese samplerate
fseek(fid, 44, 'bof');                             % springe zum 44. Byte
einpunktsignal = fread(fid, inf, 'int16');         % lese ab hier messsignal
fclose(fid);                                       % fid schließen
```

```
% --------------------------------------------------------------------------
% Verschieben des Zeitsignals, so dass max = min (Offset der Soundkarte!)
% --------------------------------------------------------------------------
max_wav=max(einpunktsignal);                        % Maximum des Zeitsignals
min_wav=min(einpunktsignal);                        % Minimum des Zeitsignals
differenz_ampl = max_wav + min_wav
einpunktsignal  = einpunktsignal-0.5*differenz_ampl;


% --------------------------------------------------------------------------
% Bestimmen der Länge des Eingangssignals
% --------------------------------------------------------------------------
N = length(einpunktsignal)

%einpunktsignal = einpunktsignal(1:2^15);



% --------------------------------------------------------------------------
% Effektivwert des Zeitsignals
% --------------------------------------------------------------------------
eff_einpunktsignal  =  sqrt(sum(einpunktsignal.^2)./N)


% --------------------------------------------------------------------------
% Berechnen der FFT und des Amplitudenspektrums
% --------------------------------------------------------------------------
Y    = fft(einpunktsignal,N);


% --------------------------------------------------------------------------
% Berechnung Skalierungsfaktors aus den Variablen der Kalibrierung
% --------------------------------------------------------------------------
skalierungsfaktor = effektivspannung/Kali_Pyy


% --------------------------------------------------------------------------
% Umrechnen in Schnelle
% --------------------------------------------------------------------------
if messb_schnelle == 6
    y_range = 5;                    % 5 mm/s/V
elseif messb_schnelle == 7
    y_range = 25;                   % 25 mm/s/V
elseif messb_schnelle == 8
    y_range = 125;                  % 125 mm/s/V
elseif messb_schnelle == 9
    y_range = 1000;                 % 1000 mm/s/V
end


% --------------------------------------------------------------------------
% Darstellen als Funktion von x=0 bis x=(Samplingfrequenz/2) und
% Skalierung der y-Achse auf Grundlage der Vibrometereinstellungen
% --------------------------------------------------------------------------
a    = N/2;                             % halbe Signallänge
b    = a+1;                             % halbe Signallänge + 1

f = 44100*(0:a)/N;                      % x-Achse

Ay  = abs(Y);                           % entspr. Ay = sqrt(Y .* conj(Y));
Ay  = 2.*Ay./N./sqrt(2);                % 'normalized discrete fourier transf.,
                                        % siehe Matlab Hilfe

Ay  =  y_range*skalierungsfaktor*Ay;    % Multipl. mit Skalierungsfaktor und
                                        % Einstellung des Vibrometers
max_Ay = max(Ay)                        % Ausgabe des Maximalwertes in Command.
```

```
plot(f,(Ay(1:b)),'-')                    % Plot des Amplitudenspektrums (wegen
                                         % Symmetrie nur eine Hälfte)
xlim([xlinks xrechts]);                  % Grenzen, Einzustellen über Slider
ylim([0 ymax]);                          % -"-
title('Amplitudenspektrum des Signals')  % Titel usw...
xlabel('Frequenz [Hz]')
ylabel('Schnelle [mm/s]')
drawnow

if  ymax < y_range                           % Umschaltung des y-Bereichs nur
    set(handles.slider3,'Max',2*y_range);    % zulässig, wenn gewählter Bereich
end;                                         % kleiner ist als der angezeigte
```

Anhang 4.28 signallevellesen

```
function signallevellesen

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers     ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global signallevel

signallevel  = 0;

% ------------------------------------------------------------------------------
% COM1 Port wird initialisiert
% ------------------------------------------------------------------------------
s = serial('COM1');
set(s,'Baudrate',9600,'Parity','none','StopBits',1);
set(s,'InputBufferSize',1024,'Terminator','LF','Timeout',1);

% ------------------------------------------------------------------------------
% Öffnen der seriellen Schnittstelle COM1
% ------------------------------------------------------------------------------
fopen(s);

% ------------------------------------------------------------------------------
% Empfangen des Signal-Levels vom Vibrometer
% ------------------------------------------------------------------------------
fprintf(s,'ECHOON');
echo = fscanf(s);

fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);
if isempty(level)
    warndlg('Konnte den Befehl "LEV" nicht an das Vibrometer senden','!! Warnung
!!')
end

fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);
```

```
for x = 0:40
    levelzahl = ['LEV',int2str(x)];
    if strcmp(level,levelzahl) == 1
    signallevel = x;
    end
end

fprintf(s,'N');

% --------------------------------------------------------------------------------
% GUI initialisieren und schreiben der Signalstärke in Feld
% --------------------------------------------------------------------------------
handles = guihandles(gcbo);
set(handles.text81,'String',num2str(signallevel));

% --------------------------------------------------------------------------------
% COM1 wird geschlossen
% --------------------------------------------------------------------------------
fclose(s)
delete(s)
clear s
```

Anhang 4.29 resetvariablen

```
function resetvariablen

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  summe_schritte_links...
        summe_schritte_rechts...
        summe_schritte_oben...
        summe_schritte_unten...
        linke_strecke...
        rechte_strecke...
        strecke_unten...
        ges_schritte_li...
        ges_schritte_re...
        ges_schritte_u...
        schritte_nach_links...
        schritte_nach_oben...
        schritte_nach_unten...
        schritte_nach_rechts...
        summe_schritte_nach_links...
        summe_schritte_nach_oben

% ------------------------------------------------------------------------------
% Geometrische Variablen werden auf Null gesetzt
% ------------------------------------------------------------------------------
        summe_schritte_links    = 0;
        summe_schritte_rechts   = 0;
        summe_schritte_oben     = 0;
        summe_schritte_unten    = 0;
        ges_schritte_li         = 0;
        ges_schritte_re         = 0;
        ges_schritte_u          = 0;
        linke_strecke           = 0;
        rechte_strecke          = 0;
        strecke_unten           = 0;
        rueckschritte_links     = 0;
        rueckschritte_oben      = 0
```

Anhang 4.30 dateiname_ep_setzen

```
function dateiname_ep_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global dateiname_einpunkt
handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Dateiname für die Testmessung wird aus Feld ausgelesen und als String
% in eine Variable geschrieben
% ------------------------------------------------------------------------------
dateiname_einpunkt = get(handles.edit14,'String');
```

Anhang 4.31 signallaenge_ep_setzen

```
function signallaenge_ep_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global signallaenge_einpunkt

handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Auslesen der Signallänge in Sekunden
% ------------------------------------------------------------------------------
signallaenge_einpunkt = str2num(get(handles.edit16,'String'));
```

Anhang 4.32 einlesen_ep

```
function einlesen_ep

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs

% ------------------------------------------------------------------------------
% Samplingfrequenz setzen, allg. Wert der Soundkarte
% ------------------------------------------------------------------------------
Fs = 44100;

% ------------------------------------------------------------------------------
% Einlesen des Einpunkt-Signals mit der Länge (signallaenge_einpunkt*Fs) in
% Sekunden und der Samplingfrequenz Fs = 44100 Hz
% ------------------------------------------------------------------------------
einpunktsignal = wavrecord(signallaenge_einpunkt*Fs,Fs);
%pause(signallaenge_einpunkt+0.5)

% ------------------------------------------------------------------------------
% Schreiben des Einpunkt-Signals ins Matlab-Verzeichnis
% ------------------------------------------------------------------------------
wavwrite(einpunktsignal,Fs,dateiname_einpunkt);
%pause(signallaenge_einpunkt+0.5)
```

Anhang 5 scan.m

Mit dem zweiten der beiden Hauptprogramme wird die Programmoberfläche „SLDV Scan“ gestartet. Die Funktion beinhaltet die verwendeten globalen Variablen für alle eingebetteten Funktionen und die Zuordnung der einzelnen Funktionen bezüglich der Pushbuttons, Edit-Felder usw.

In diesem Abschnitt werden nun alle zum Programm „scan.m“ gehörigen Funktionen in der Reihenfolge ihrer Verwendung aufgeführt.

```
function varargout = scan(varargin)

% ================================================================
% ||                                                            ||
% ||           /       __  __       \    /     _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% SCAN Application M-file for scan.fig
%    FIG = SCAN launch scan GUI.
%    SCAN('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 04-Jun-2004 16:20:31

global  signallevel...
        v_x...
        v_y...
        schritte...
        entfernung...
        summe_schritte_links...
        summe_schritte_rechts...
        summe_schritte_oben...
        summe_schritte_unten...
        linke_strecke...
        rechte_strecke...
```

```
        strecke_unten...
        strecke_oben...
        ges_schritte_li...
        ges_schritte_re...
        ges_schritte_o...
        ges_schritte_u...
        messb_schnelle...
        messb_ausl...
        filter...
        st_filter...
        messb_schnelle_string...
        messb_ausl_string...
        filter_string...
        st_filter_string...
        horizontale...
        vertikale...
        dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs...
        xlinks...
        xrechts...
        ymax...
        scan_schrittweite_x...
        scan_schrittweite_y...
        scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        rueckschritte_links...
        rueckschritte_oben...
        scan_dateiname...
        scan_signallaenge...
        angezeigte_frequenz...
        dateiname_messung...
        angezeigte_frequenz...
        Pyy...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        effektivspannung...
        Kali_Pyy...
        messb_schnelle...
        y_range...
        scan_signallevellesen...
        hoh


if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

    scan_schrittweite_x     = 100;
    scan_schrittweite_y     = 100;
    scan_v_x                = 50;
    scan_v_y                = 50;
```

```
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

	try
		if (nargout)
			[varargout{1:nargout}] = feval(varargin{:}); % FEVAL
switchyard
		else
			feval(varargin{:}); % FEVAL switchyard
		end
	catch
		disp(lasterr);
	end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.



% --------------------------------------------------------------------
function varargout = popupmenu3_Callback(h, eventdata, handles, varargin)
scan_vxsetzen

% --------------------------------------------------------------------
function varargout = popupmenu4_Callback(h, eventdata, handles, varargin)
scan_vysetzen

% --------------------------------------------------------------------
function varargout = edit3_Callback(h, eventdata, handles, varargin)
scan_schrittweite_x

% --------------------------------------------------------------------
function varargout = edit4_Callback(h, eventdata, handles, varargin)
scan_schrittweite_y
```

```
% --------------------------------------------------------------------
function varargout = radiobutton1_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = radiobutton2_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
scan_focus

% --------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
scan_vxsetzen
scan_vysetzen
scan_schrittweite_x
scan_schrittweite_y
scan_dateiname_setzen
scan_signallaenge_setzen
scan_autoscan

% --------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)


% --------------------------------------------------------------------
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
scan_signallevellesen

% --------------------------------------------------------------------
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
startposition

% --------------------------------------------------------------------
function varargout = edit5_Callback(h, eventdata, handles, varargin)
scan_dateiname_setzen

% --------------------------------------------------------------------
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton7_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = edit6_Callback(h, eventdata, handles, varargin)
scan_signallaenge_setzen

% --------------------------------------------------------------------
function varargout = slider1_Callback(h, eventdata, handles, varargin)
scan_frequenzslider

% --------------------------------------------------------------------
function varargout = edit8_Callback(h, eventdata, handles, varargin)
scan_frequenz_edit

% --------------------------------------------------------------------
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
scan_dateiname_messung
load_dateiname_messung

% --------------------------------------------------------------------
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
scan_dateiname_messung
save_dateiname_messung

% --------------------------------------------------------------------
function varargout = edit9_Callback(h, eventdata, handles, varargin)
scan_dateiname_messung
```

```
% --------------------------------------------------------------------
function varargout = edit11_Callback(h, eventdata, handles, varargin)
scan_ymax_setzen

% --------------------------------------------------------------------
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
scan_ausw_flat

% --------------------------------------------------------------------
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
scan_ausw_phong

% --------------------------------------------------------------------
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
scan_ausw_flat_50

% --------------------------------------------------------------------
function varargout = pushbutton18_Callback(h, eventdata, handles, varargin)
scan_ausw_phong_50

% --------------------------------------------------------------------
function varargout = pushbutton21_Callback(h, eventdata, handles, varargin)
scan_ausw_flat_50_fit

% --------------------------------------------------------------------
function varargout = pushbutton22_Callback(h, eventdata, handles, varargin)
scan_ausw_phong_50_fit

% --------------------------------------------------------------------
function varargout = pushbutton25_Callback(h, eventdata, handles, varargin)
scan_fit_in

% --------------------------------------------------------------------
function varargout = pushbutton26_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton27_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton28_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton29_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton30_Callback(h, eventdata, handles, varargin)
scan_ausw_sep_fig

% --------------------------------------------------------------------
function varargout = pushbutton31_Callback(h, eventdata, handles, varargin)
scan_ausw_sep_fig_glatt
```

Anhang 5.1 scan_vxsetzen

```
function scan_vxsetzen

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _              ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers     ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global scan_v_x

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Schrittgeschwindig aus Dropdown-Feld auslesen und
% scan_v_x setzen
% ---------------------------------------------------------------
scan_v_x_ein = get(handles.popupmenu3,'Value');

if scan_v_x_ein == 1
    scan_v_x = 50;
elseif scan_v_x_ein == 2
    scan_v_x = 100;
elseif scan_v_x_ein == 3
    scan_v_x = 250;
elseif scan_v_x_ein == 4
    scan_v_x = 500;
elseif scan_v_x_ein == 5
    scan_v_x = 1000;
elseif scan_v_x_ein == 6
    scan_v_x = 2000;
elseif scan_v_x_ein == 7
    scan_v_x = 3000;
elseif scan_v_x_ein == 8
    scan_v_x = 4000;
elseif scan_v_x_ein == 9
    scan_v_x = 5000;
end
```

Anhang 5.2 scan_vysetzen

```
function scan_vysetzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global scan_v_y

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Schrittgeschwindig aus Dropdown-Feld auslesen und
% scan_v_y setzen
% ---------------------------------------------------------------
scan_v_y_ein = get(handles.popupmenu4,'Value');

if scan_v_y_ein == 1
    scan_v_y = 50;
elseif scan_v_y_ein == 2
    scan_v_y = 100;
elseif scan_v_y_ein == 3
    scan_v_y = 250;
elseif scan_v_y_ein == 4
    scan_v_y = 500;
elseif scan_v_y_ein == 5
    scan_v_y = 1000;
elseif scan_v_y_ein == 6
    scan_v_y = 2000;
elseif scan_v_y_ein == 7
    scan_v_y = 3000;
end
```

Anhang 5.3 scan_schrittweite_x

```
function scan_schrittweite_x

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _              ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  scan_schrittweite_x

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Scan-Schrittweite aus Feld auslesen
% ---------------------------------------------------------------
scan_schrittweite_x = str2num(get(handles.edit3,'String'))
```

Anhang 5.4 scan_schrittweite_y

```
function scan_schrittweite_y

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  scan_schrittweite_y

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Scan-Schrittweite aus Feld auslesen
% ---------------------------------------------------------------
scan_schrittweite_y = str2num(get(handles.edit4,'String'));
```

Anhang 5.5 scan_focus

```
function scan_focus

% ================================================================
% ||                                                            ||
% ||           /          __  __        \    /    _           ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  signallevel...
        scan_signallevel

% ------------------------------------------------------------------------------
% Voreinstellungen, die "empirisch" gefundenen Werte ttrechts und ttlinks
% sind später in der Schleife sozusagen die "Länge" für die Motorbewegung
% in eine Richtung. Da de Motor schneller nach rechts läuft als nach links,
% ist dieser Wert höher. (s.u.)
% ------------------------------------------------------------------------------
zaehler      = 0;
ttrechts     = 6.7;
ttlinks      = 9.6;

% ------------------------------------------------------------------------------
% COM1 Port initialisieren
% ------------------------------------------------------------------------------
s = serial('COM1');
set(s,'Baudrate',9600,'Parity','none','StopBits',1);
set(s,'InputBufferSize',1024,'Terminator','LF','Timeout',1);

% ------------------------------------------------------------------------------
% Öffnen der seriellen Schnittstelle COM1
% ------------------------------------------------------------------------------
fopen(s);

% ------------------------------------------------------------------------------
% Auslesen der Ausgangssignalstärke "signallevela"
% ------------------------------------------------------------------------------
fprintf(s,'ECHOON');
echo = fscanf(s);

fprintf(s,'LEV');
level = fscanf(s);
```

```
level = cellstr(level);
if isempty(level)
    warndlg...
    ('Konnte den Befehl "LEV" nicht an das Vibrometer senden','!! Warnung !!')
end

fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);

for x = 0:40
    levelzahl = ['LEV',int2str(x)];
    if strcmp(level,levelzahl) == 1
    signallevela = x;
    end
end

% ------------------------------------------------------------------------
% ------------------------- A U T O F O K U S ----------------------------
% ------------------------------------------------------------------------
% Die Funktionsweise ist wie folgt: Nachdem der Befehl zum Rechtsdrehen
% an den Fokussierungs-Motor gesendet wurde, wird in einer Schleife mehrmals
% die Signalstärke ausgelesen. Da dies "Zeit kostet", dreht der Motor mit einer
% bestimmten Zeit in diese Richtung. Dies ist leider notwendig, da man dem
% Motor nur die Befehle "start" und "stopp" geben kann, aber nicht, wie lange
% er laufen soll. Deswegen diese Notlösung...
% Die Bewegungen laufen wie folgt ab: erst Rechtslauf, dann Linkslauf mit den
% o.a. Schleifenlängen. Diese werden für die nächste Schleife um den Faktor
% 1.5 vergrößert, damit ein größerer Bereich in den Fokus gelangt.
% Insgesamt läuft die Fokusschleife entweder, bis die Signalstärke ungleich
% Null ist, oder bis der Zähler den Stand 4 erreicht hat, also vier
% Durchläufe
% ------------------------------------------------------------------------
signallevel = signallevela;
    while (signallevel == 0) & (zaehler <= 4)
                        ttrechts = 1.5 * ttrechts
                        ttlinks  = 1.5 * ttlinks
                        zaehler = zaehler + 1;
                        fprintf(s,'r');
                        for ta = 0:ttrechts
                                fprintf(s,'LEV');
                                level = fscanf(s);
                                level = cellstr(level);

                                        for xa = 0:40
                                            levelzahl = ['LEV',int2str(xa)];
                                            if strcmp(level,levelzahl) == 1
                                            signallevel = xa;
                                            end
                                        end

                                if signallevel > signallevela
                                fprintf(s,'N');
                                break
                                end
                        end
                        fprintf(s,'N');

                        fprintf(s,'l');
                        for tb = 0:ttlinks
                                fprintf(s,'LEV');
                                level = fscanf(s);
                                level = cellstr(level);

                                        for xb = 0:40
                                            levelzahl = ['LEV',int2str(xb)];
                                            if strcmp(level,levelzahl) == 1
                                            signallevel = xb;
```

```
                                        end
                                    end

                                if signallevel > signallevela
                                fprintf(s,'N');
                                break
                                end
                            end
    end

    scan_signallevel = signallevel;

% --------------------------------------------------------------------------------
% Stop-Befehl an Motor senden
% --------------------------------------------------------------------------------
fprintf(s,'N');


% --------------------------------------------------------------------------------
% Signalstärke in Feld schreiben
% --------------------------------------------------------------------------------
handles = guihandles(gcbo);
set(handles.text21,'String',num2str(signallevel));

% --------------------------------------------------------------------------------
% Serielle Schnittstelle schließen
% --------------------------------------------------------------------------------
fclose(s)
delete(s)
clear s
```

Anhang 5.6 scan_dateiname_setzen

```
function scan_dateiname_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global scan_dateiname

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Dateiname aus Feld auslesen
% ---------------------------------------------------------------
scan_dateiname = get(handles.edit5,'String');
```

Anhang 5.7 scan_signallaenge_setzen

```
function scan_signallaenge_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _             ||
% ||          /   /\  |_  |    /\    \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global scan_signallaenge

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Scan-Signalläng aus Feld auslesen
% ---------------------------------------------------------------
scan_signallaenge = str2num(get(handles.edit6,'String'));
```

Anhang 5.8 scan_autoscan

```
function scan_autoscan

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        horizontale...
        vertikale...
        scan_schrittweite_x...
        scan_schrittweite_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        Pyy...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        colorscale...
        effektivspannung...
        Kali_Pyy...
        messb_schnelle...
        y_range...
        hoh...
        entfernung

handles = guihandles(gcbo);
```

```
% --------------------------------------------------------------------------
% Benötigte Variablen erzeugen
% --------------------------------------------------------------------------
bisherige_y_schritte   = 0;

% --------------------------------------------------------------------------
% Schleifenvariablen setzen
% --------------------------------------------------------------------------
x_schleife_anfang      =  1;
x_schleife_ende        =  fix(horizontale/scan_schrittweite_x);
y_schleife_anfang      =  1;
y_schleife_ende        =  fix(vertikale/scan_schrittweite_y);


% --------------------------------------------------------------------------
% Gesamtzahl der Scanpunkte berechnen und in Feld schreiben
% --------------------------------------------------------------------------
gesamtzahl_scanpunkte  =  x_schleife_ende*y_schleife_ende;
set(handles.text35,'String',num2str(gesamtzahl_scanpunkte));


% --------------------------------------------------------------------------
% Voreinstellung für die angezeigte Frequenz und schreiben ins Edit-Feld
% --------------------------------------------------------------------------
set(handles.edit8,'String',num2str(angezeigte_frequenz));
val                     = str2num(get(handles.edit8,'String'));
angezeigte_frequenz     = max(val,1);

% --------------------------------------------------------------------------
% Berechnung Skalierungsfaktors aus den Variablen der Kalibrierung
% --------------------------------------------------------------------------
skalierungsfaktor = effektivspannung/Kali_Pyy;

% --------------------------------------------------------------------------
% Hauptschleife für den Scan-Vorgang (y-Richtung)
% --------------------------------------------------------------------------
for y_scanschleife = y_schleife_anfang:y_schleife_ende

% --------------------------------------------------------------------------
% bisherige_x_schritte zur ersten Spalte wieder auf null setzen
% --------------------------------------------------------------------------
bisherige_x_schritte   = 0;

% --------------------------------------------------------------------------
% Anzahl des gesamten Scanwinkels in y-Richtung wird berechnet
% --------------------------------------------------------------------------
ges_scanwinkel_y  = 2*((360/(2*pi))* ...
                    atan(0.5*y_schleife_ende*scan_schrittweite_y/entfernung));

halber_weg_y      = 3.353899071668E-04*(ges_scanwinkel_y/2)^5 ...
                  + 1.280602602320E-02*(ges_scanwinkel_y/2)^4 ...
                  + 3.287381441029E-01*(ges_scanwinkel_y/2)^3 ...
                  + 7.314062013423E+00*(ges_scanwinkel_y/2)^2 ...
                  - 1.207856559729E+03*(ges_scanwinkel_y/2);

   % --------------------------------------------------------------------------
   % Aktuelle Schrittzahl in y-Richtung berechnen
   % --------------------------------------------------------------------------
   scanwinkel_y_o    = -0.5*ges_scanwinkel_y + ((360/(2*pi))* ...
                       atan(y_scanschleife*scan_schrittweite_y/entfernung));

   scanwinkel_y_u    = -0.5*ges_scanwinkel_y + ((360/(2*pi))* ...
                       atan(((y_scanschleife-1)*scan_schrittweite_y)/entfernung));
```

```
scan_schritte_y = abs((...
                    3.353899071668E-04*(scanwinkel_y_o)^5 ...
                  + 1.280602602320E-02*(scanwinkel_y_o)^4 ...
                  + 3.287381441029E-01*(scanwinkel_y_o)^3 ...
                  + 7.314062013423E+00*(scanwinkel_y_o)^2 ...
                  - 1.207856559729E+03*(scanwinkel_y_o))  ...
                  - ...
                  +(3.353899071668E-04*(scanwinkel_y_u)^5 ...
                  + 1.280602602320E-02*(scanwinkel_y_u)^4 ...
                  + 3.287381441029E-01*(scanwinkel_y_u)^3 ...
                  + 7.314062013423E+00*(scanwinkel_y_u)^2 ...
                  - 1.207856559729E+03*(scanwinkel_y_u)));

bisherige_y_schritte  =  bisherige_y_schritte + scan_schritte_y;

% -------------------------------------------------------------------------
% Bewegungsvariablen erzeugen
% -------------------------------------------------------------------------
scan_bewegung_u  =  ['@0A'...
                    int2str(0)...
                    ',' ...
                    int2str(scan_v_y)...
                    ',' ...
                    int2str(0)...
                    ',' ...
                    int2str(scan_v_y)...
                    ',' ...
                    int2str(-scan_schritte_y)...
                    ',' ...
                    int2str(scan_v_y)];

% -------------------------------------------------------------------------
% Hauptschleife für den Scan-Vorgang (x-Richtung)
% -------------------------------------------------------------------------
for x_scanschleife = x_schleife_anfang:x_schleife_ende


     % ---------------------------------------------------------------------
     % Ges. Scanwinkel und halber Weg für Winkelkorrektur wird berechnet
     % ---------------------------------------------------------------------
     ges_scanwinkel_x  = 2*((360/(2*pi))* ...
                         atan(0.5*x_schleife_ende*...
                         scan_schrittweite_x/entfernung));

     halber_weg_x      =  1.537878419615E-05*(ges_scanwinkel_x/2)^5 ...
                        - 1.003307955963E-03*(ges_scanwinkel_x/2)^4 ...
                        + 3.944388752658E-02*(ges_scanwinkel_x/2)^3 ...
                        - 1.082877883392E+00*(ges_scanwinkel_x/2)^2 ...
                        + 5.505302315286E+02*(ges_scanwinkel_x/2)

     % ---------------------------------------------------------------------
     % Aktuelle Schrittzahl in x-Richtung berechnen
     %----------------------------------------------------------------------
     scanwinkel_x_o  = -0.5*ges_scanwinkel_x + ((360/(2*pi))* ...
                       atan(x_scanschleife*scan_schrittweite_x/entfernung));

     scanwinkel_x_u  = -0.5*ges_scanwinkel_x + ((360/(2*pi))* ...
                       atan(((x_scanschleife-1)*scan_schrittweite_x)...
                       /entfernung));
```

```
scan_schritte_x = abs((...
                    1.537878419615E-05*(scanwinkel_x_o)^5 ...
                  - 1.003307955963E-03*(scanwinkel_x_o)^4 ...
                  + 3.944388752658E-02*(scanwinkel_x_o)^3 ...
                  - 1.082877883392E+00*(scanwinkel_x_o)^2 ...
                  + 5.505302315286E+02*(scanwinkel_x_o))  ...
                  - ...
                  ( 1.537878419615E-05*(scanwinkel_x_u)^5 ...
                  - 1.003307955963E-03*(scanwinkel_x_u)^4 ...
                  + 3.944388752658E-02*(scanwinkel_x_u)^3 ...
                  - 1.082877883392E+00*(scanwinkel_x_u)^2 ...
                  + 5.505302315286E+02*(scanwinkel_x_u)));

bisherige_x_schritte  =  bisherige_x_schritte + scan_schritte_x;

% ------------------------------------------------------------------------
% Bewegungsvariablen erzeugen
% ------------------------------------------------------------------------
scan_bewegung_r  =  ['@0A'...                    % Bewegung initialisieren
                    int2str(scan_schritte_x)... % Schritte auf der x-Achse
                    ',' ...                      % (jeweils als "String")
                    int2str(scan_v_x)...         % Geschwindigkeit x-Achse
                    ',' ...
                    int2str(0)...                % 1. Bewegung y-Achse
                    ',' ...
                    int2str(scan_v_x)...         % Geschwindigkeit y-Achse
                    ',' ...
                    int2str(0)...                % 2. Bewegung y-Achse
                    ',' ...
                    int2str(scan_v_x)];          % Geschwindigkeit y-Achse

% ------------------------------------------------------------------------
% Korrekturfaktor für "schiefes" Messen wird berechnet
% ------------------------------------------------------------------------
if (halber_weg_x-bisherige_x_schritte > 0 ...
        | halber_weg_y-bisherige_y_schritte > 0)
    winkelkorrektur = 1/(cos((pi/2)- ...
                      atan(entfernung/ ...
                      (sqrt((halber_weg_x-bisherige_x_schritte)^2 ...
                      + (halber_weg_y-bisherige_y_schritte)^2)))));
    else
    winkelkorrektur = 1;
end;

% ------------------------------------------------------------------------
% Anzahl der abgearbeiteten Scanpunkte
%-------------------------------------------------------------------------
Spalten_Fortschritt  =   x_scanschleife;
Zeilen_Fortschritt   =   y_scanschleife;
BisherigePunkte      =   (x_schleife_ende * (Zeilen_Fortschritt-1)) ...
                           + Spalten_Fortschritt;
set(handles.text32,'String',num2str(BisherigePunkte));

% ------------------------------------------------------------------------
% Fokusroutine aktivieren, wenn Checkbox gesetzt
%-------------------------------------------------------------------------
if get(handles.radiobutton1,'Value') == 1
    scan_focus
end;
```

```
% ------------------------------------------------------------------------
% Setzen der Samplingfrequenz, Einlesen des Signals als .wav-File und
% Schreiben in das Matlab-Verzeichnis
% ------------------------------------------------------------------------
Fs = 44100;
scan_signal = wavrecord(scan_signallaenge*Fs,Fs);
scan_dateiname_komplett = [scan_dateiname...
                           '_'...
                           int2str(y_scanschleife)...
                           '-'...
                           int2str(x_scanschleife)...
                           '.wav'];
wavwrite(scan_signal,Fs,scan_dateiname_komplett);

% ------------------------------------------------------------------------
% COM2 Port initialisieren
% ------------------------------------------------------------------------
s2 = serial('COM2');
set(s2,...
    'Baudrate',9600,...
    'Parity','none',...
    'StopBits',1,...
    'Terminator','CR/LF',...
    'Timeout',1);

% ------------------------------------------------------------------------
% Öffnen des seriellen Ports COM2
% ------------------------------------------------------------------------
fopen(s2);

% ------------------------------------------------------------------------
% Achse initialisieren
% ------------------------------------------------------------------------
fprintf(s2,'@05');
fprintf(s2,'@0C1');

% ------------------------------------------------------------------------
% Befehl zum Bewegen nach rechts senden
% ------------------------------------------------------------------------
fprintf(s2,scan_bewegung_r);

% ------------------------------------------------------------------------
% Schließen des seriellen Ports COM2
% ------------------------------------------------------------------------
fclose(s2)
delete(s2)
clear s2

% ------------------------------------------------------------------------
% Einfügen einer Programmpause, während sich die Schrittmotoren
% bewegen, Dauer errechnet sich aus Schrittweite und Geschwindigkeit
% zzgl. einer 'Sicherheits'-Pause von 0.3 Sekunden
% ------------------------------------------------------------------------
pause((scan_schritte_x/scan_v_x)+0.3)

% ------------------------------------------------------------------------
% Einlesen des wave-Files
% ------------------------------------------------------------------------
fid            = fopen(scan_dateiname_komplett, 'r'); % Handle erzeugen
fseek(fid, 22, 'bof');                    % überspringe ersten 22 Byte
numchannels    = fread(fid, 1, 'int16');  % lese numchannels aus datei
samplerate     = fread(fid, 1, 'int32');  % lese samplerate
fseek(fid, 44, 'bof');                    % springe zum 44. Byte
scan_signal    = fread(fid, inf, 'int16');  % lese ab hier messsignal
fclose(fid);                              % fid schließen
```

```
% ------------------------------------------------------------------------
% Verschieben des Zeitsignals, so dass max = min (Offset der Soundkarte!)
% ------------------------------------------------------------------------
max_wav = max(scan_signal);              % Maximum des Zeitsignals
min_wav = min(scan_signal);              % Minimum des Zeitsignals
differenz_ampl = max_wav + min_wav;
scan_signal  = scan_signal-0.5*differenz_ampl;

% ------------------------------------------------------------------------
% Multiplizieren des Signals mit Winkelkorrekturfaktor
% ------------------------------------------------------------------------
scan_signal  = winkelkorrektur*scan_signal;

% ------------------------------------------------------------------------
% Bestimmen der Länge des Eingangssignals
% ------------------------------------------------------------------------
N = length(scan_signal);

% ------------------------------------------------------------------------
% Berechnen der FFT
% ------------------------------------------------------------------------
fft_scan_signal    = fft(scan_signal,N);
clear scan_signal

% ------------------------------------------------------------------------
% Umrechnen in Schnelle
% ------------------------------------------------------------------------
if messb_schnelle == 6
    y_range = 5;                  % 5 mm/s/V
elseif messb_schnelle == 7
    y_range = 25;                 % 25 mm/s/V
elseif messb_schnelle == 8
    y_range = 125;                % 125 mm/s/V
elseif messb_schnelle == 9
    y_range = 1000;               % 1000 mm/s/V
end

% ------------------------------------------------------------------------
% Darstellen als Funktion von x=0 bis x=(Samplingfrequenz/2) und
% Skalierung der y-Achse auf Grundlage der Vibrometereinstellungen
% ------------------------------------------------------------------------
a    = N/2;                           % halbe Signallänge
b    = a+1;                           % halbe Signallänge + 1

f = 44100*(0:a)/N;                    % x-Achse

Ay  = abs(fft_scan_signal);      % entspr. Ay = sqrt(Y .* conj(Y));

clear fft_scan_signal

Ay  = 2.*Ay./N./sqrt(2);     % 'normalized discrete fourier transf.,
                             % siehe Matlab Hilfe

Ay  =  y_range*skalierungsfaktor*Ay;  % Multipl. mit Skalierungsfaktor und
                               % Einstellung des Vibrometers
max_Ay = max(Ay)               % Ausgabe des Maximalwertes in Command.


% ------------------------------------------------------------------------
% Nur Messpunkte verwenden, wenn Signal vorhanden
% ------------------------------------------------------------------------
% scan_signallevellesen
% if scan_signallevel == 0
%     Ay = 0.*Ay;
% end;
```

```
    % ------------------------------------------------------------------
    % Berechnen der Grenzen
    % ------------------------------------------------------------------
    ug  = xlinks*N/Fs;
    og  = xrechts*N/Fs;
    Ay = Ay(ug+1:og);
    hoh = length(Ay);



    % ------------------------------------------------------------------
    % Im ersten Schritt der Schleife wird eine mit Einsen gefüllte
    % Messwertematrix erzeugt
    % ------------------------------------------------------------------
    if y_scanschleife == y_schleife_anfang...
            & x_scanschleife == x_schleife_anfang
        Y = zeros(y_schleife_ende,x_schleife_ende,hoh);
    end;

    % ------------------------------------------------------------------
    % Schreiben des FFT-Betragsquadrats in die 3. Komponente der Messwerte-
    % matrix
    % ------------------------------------------------------------------
    Y(y_scanschleife,x_scanschleife,:) = Ay;
    clear Ay

    % ------------------------------------------------------------------
    % Anzeige der gemessenen Werte in axes2 als Matrix mit Skala
    % ------------------------------------------------------------------
    axes(handles.axes2)
    Messwerte_Matrix = Y;
    set(gcf,'Doublebuffer','on');
    colormap(hot)
    colorscale = [0 ymax];

    set(handles.slider1,'SliderStep',...
        [1/((xrechts-xlinks)*scan_signallaenge) ...
            10*(1/((xrechts-xlinks)*scan_signallaenge))])

    set(handles.slider1,'Min',xlinks)
    set(handles.slider1,'Max',xrechts)

    Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) =
        ... flipud(Messwerte_Matrix(:,:,...
            ((angezeigte_frequenz-xlinks)*scan_signallaenge)));

    imagesc(Messwerte_Matrix(:,:,...
        ((angezeigte_frequenz-
        xlinks)*scan_signallaenge)),'parent',handles.axes2);
    colorbar
    set(gca,'ZLim',[0 ymax])
    xlim([0.999 x_schleife_ende])
    ylim([0.999 y_schleife_ende])
    drawnow

end;

    % ------------------------------------------------------------------
    % Bewegung zurück zur ersten Spalte
    % ------------------------------------------------------------------
    s2 = serial('COM2');
    set(s2,...
        'Baudrate',9600,...
        'Parity','none',...
        'StopBits',1,...
        'Terminator','CR/LF',...
        'Timeout',1);
```

```
        fopen(s2);

        fprintf(s2,'@05');
        fprintf(s2,'@0C1');

        bewegung_zurueck   =    ['@0A'...
                                int2str(-bisherige_x_schritte-1000)...
                                ','...
                                int2str(max(scan_v_x,5000))...
                                ','...
                                int2str(0)...
                                ','...
                                int2str(scan_v_x)...
                                ','...
                                int2str(0)...
                                ','...
                                int2str(scan_v_x)];

        bewegung_1000vor    =    ['@0A'...
                                int2str(1000)...
                                ','...
                                int2str(scan_v_x)...
                                ','...
                                int2str(0)...
                                ','...
                                int2str(scan_v_x)...
                                ','...
                                int2str(0)...
                                ','...
                                int2str(scan_v_x)];

        if y_scanschleife < y_schleife_ende
            fprintf(s2,bewegung_zurueck);
            pause((rueckschritte_links/max(scan_v_x,5000))+0.3)
            fprintf(s2,bewegung_1000vor);
            pause((1000/max(scan_v_x,5000))+0.3)
            fprintf(s2,scan_bewegung_u);
            pause((scan_schritte_y/scan_v_y)+0.3)
        end;

        fclose(s2)
        delete(s2)
        clear s2


end;

% ------------------------------------------------------------------------
% Bewegung zurück
% ------------------------------------------------------------------------
pause(3)
s2 = serial('COM2');
set(s2,...
    'Baudrate',9600,...
    'Parity','none',...
    'StopBits',1,...
    'Terminator','CR/LF',...
    'Timeout',1);

        fopen(s2);

        fprintf(s2,'@05');
        fprintf(s2,'@0C1');
```

```
        bewegung_ganz_zur_x   = ['@0A'...
                                int2str(-bisherige_x_schritte)...
                                ','...
                                int2str(max(scan_v_x,5000))...
                                ','...
                                int2str(0)...
                                ','...
                                int2str(scan_v_x)...
                                ','...
                                int2str(0)...
                                ','...
                                int2str(scan_v_x)];

        bewegung_ganz_zur_y   = ['@0A'...
                                int2str(0)...
                                ',' ...
                                int2str(scan_v_y)...
                                ',' ...
                                int2str(0)...
                                ',' ...
                                int2str(scan_v_y)...
                                ',' ...
                                int2str(bisherige_y_schritte)...
                                ',' ...
                                int2str(max(scan_v_y,2000))];

fprintf(s2,bewegung_ganz_zur_x);
fprintf(s2,bewegung_ganz_zur_y);

fclose(s2)
delete(s2)
clear s2
```

Anhang 5.9 scan_signallevellesen

```
function scan_signallevellesen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global scan_signallevel

scan_signallevel  = 0;

% ------------------------------------------------------------------------------
% COM1 Port wird initialisiert
% ------------------------------------------------------------------------------
s = serial('COM1');
set(s,'Baudrate',9600,'Parity','none','StopBits',1);
set(s,'InputBufferSize',1024,'Terminator','LF','Timeout',1);

% ------------------------------------------------------------------------------
% Öffnen der seriellen Schnittstelle COM1
% ------------------------------------------------------------------------------
fopen(s);

% ------------------------------------------------------------------------------
% Empfangen des Signal-Levels vom Vibrometer
% ------------------------------------------------------------------------------
fprintf(s,'ECHOON');
echo = fscanf(s);

fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);
if isempty(level)
    warndlg('Konnte den Befehl "LEV" nicht an das Vibrometer senden','!! Warnung
!!')
end

fprintf(s,'LEV');
level = fscanf(s);
level = cellstr(level);
```

```
for x = 0:40
    levelzahl = ['LEV',int2str(x)];
    if strcmp(level,levelzahl) == 1
    scan_signallevel = x;
    end
end

fprintf(s,'N');

% --------------------------------------------------------------------------------
% GUI initialisieren und schreiben der Signalstärke in Feld
% --------------------------------------------------------------------------------
handles = guihandles(gcbo);
set(handles.text21,'String',num2str(scan_signallevel));

% --------------------------------------------------------------------------------
% COM1 wird geschlossen
% --------------------------------------------------------------------------------
fclose(s)
delete(s)
clear s
```

Anhang 5.10 startposition

```
function startposition

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _              ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers     ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------
global  v_x...
        v_y...
        rueckschritte_links...
        rueckschritte_oben

% ------------------------------------------------------------------------
% Bewegungsvariablen erzeugen
% ------------------------------------------------------------------------
bewegung_lr  =  ['@0A'...
                 int2str(rueckschritte_links)...
                 ','...
                 int2str(v_x)...
                 ','...
                 int2str(0)...
                 ','...
                 int2str(v_x)...
                 ','...
                 int2str(0)...
                 ','...
                 int2str(v_x)];

bewegung_ou  =  ['@0A'...
                 int2str(0)...
                 ','...
                 int2str(v_y)...
                 ','...
                 int2str(0)...
                 ','...
                 int2str(v_y)...
                 ','...
                 int2str(-rueckschritte_oben)...
                 ','...
                 int2str(v_y)]
```

```
% --------------------------------------------------------------------
% COM2 Port initialisieren
% --------------------------------------------------------------------
s2 = serial('COM2');
set(s2,...
     'Baudrate',9600,...
     'Parity','none',...
     'StopBits',1,...
     'Terminator','CR/LF',...
     'Timeout',1);

% --------------------------------------------------------------------
% Öffnen des seriellen Ports COM2
% --------------------------------------------------------------------
fopen(s2);

% --------------------------------------------------------------------
% Achse initialisieren
% --------------------------------------------------------------------
fprintf(s2,'@05');
fprintf(s2,'@0C1');

% --------------------------------------------------------------------
% Befehl zum Bewegen senden
% --------------------------------------------------------------------
fprintf(s2,bewegung_ou);
fprintf(s2,bewegung_lr);

% --------------------------------------------------------------------
% Schließen des seriellen Ports COM2
% --------------------------------------------------------------------
fclose(s2)
delete(s2)
clear s2
```

Anhang 5.11 scan_frequenzslider

```
function scan_frequenzslider

% ================================================================
% ||                                                            ||
% ||           /       __  __       \    /    _               ||
% ||          /   /\  |_  |    /\    \  /   | |_)              ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)             ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...
        hoh


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
```

```
val = str2num(get(handles.edit8,'String'));
angezeigte_frequenz = val;


% --------------------------------------------------------------------------------
% Darstellen der Matrix im Hauptfenster
% --------------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                    % verhindert Flackern

set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge)
    10*(1/((xrechts-xlinks)*scan_signallaenge))])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];
Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));
axes(handles.axes2)
imagesc(Messwerte_Matrix(:,:,(angezeigte_frequenz-
xlinks)*scan_signallaenge),'parent',handles.axes2,colorscale);
colorbar

xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])
drawnow
```

Anhang 5.12 scan_frequenz_edit

```
function scan_frequenz_edit

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers     ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...
        hoh

handles = guihandles(gcbo);

% ------------------------------------------------------------------------------
% Der Wert aus Edit8 wird als Zahl ausgelesen und der Slider auf diesen
% Wert gesetzt
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
if isnumeric(val) & length(val)==1 & val >= get(handles.slider1,'Min') & val <=
get(handles.slider1,'Max')
    set(handles.slider1,'Value',val);
    angezeigte_frequenz = val
else
    set(handles.edit8,'String','Fehler');
end
```

```
% --------------------------------------------------------------------------
% Darstellen der Matrix im Hauptfenster
% --------------------------------------------------------------------------
Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                   % verhindert Flackern

set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge)
    10*(1/((xrechts-xlinks)*scan_signallaenge))])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];
Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));
axes(handles.axes2)
imagesc(Messwerte_Matrix(:,:,(angezeigte_frequenz-
xlinks)*scan_signallaenge),'parent',handles.axes2,colorscale);
colorbar


xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])
drawnow
```

Anhang 5.13 scan_dateiname_messung

```
function scan_dateiname_messung

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /     _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global dateiname_messung

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% Dateiname aus Feld auslesen
% ---------------------------------------------------------------
dateiname_messung = get(handles.edit9,'String')
```

Anhang 5.14 load_dateiname_messung

```
function load_dateiname_messung

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                 ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)                ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers     ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur          ||
% ||         Visualisierung schwingender Oberflächen"           ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, die geladen werden sollen
% ------------------------------------------------------------------------------
global  signallevel...
        v_x...
        v_y...
        schritte...
        entfernung...
        summe_schritte_links...
        summe_schritte_rechts...
        summe_schritte_oben...
        summe_schritte_unten...
        linke_strecke...
        rechte_strecke...
        strecke_unten...
        strecke_oben...
        ges_schritte_li...
        ges_schritte_re...
        ges_schritte_o...
        ges_schritte_u...
        messb_schnelle...
        messb_ausl...
        filter...
        st_filter...
        messb_schnelle_string...
        messb_ausl_string...
        filter_string...
        st_filter_string...
        horizontale...
        vertikale...
        dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs...
        xlinks...
        xrechts...
```

```
        ymax...
        scan_schrittweite_x...
        scan_schrittweite_y...
        scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        rueckschritte_links...
        rueckschritte_oben...
        scan_dateiname...
        scan_signallaenge...
        angezeigte_frequenz...
        dateiname_messung...
        angezeigte_frequenz...
        Pyy...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y


% ------------------------------------------------------------------------
% Laden der Variablen mit Dateinamen "dateiname_messung"
% ------------------------------------------------------------------------
load(dateiname_messung)
```

Anhang 5.15 save_dateiname_messung

```
function save_dateiname_messung

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen
% ------------------------------------------------------------------------------
global  signallevel...
        v_x...
        v_y...
        schritte...
        entfernung...
        summe_schritte_links...
        summe_schritte_rechts...
        summe_schritte_oben...
        summe_schritte_unten...
        linke_strecke...
        rechte_strecke...
        strecke_unten...
        strecke_oben...
        ges_schritte_li...
        ges_schritte_re...
        ges_schritte_o...
        ges_schritte_u...
        messb_schnelle...
        messb_ausl...
        filter...
        st_filter...
        messb_schnelle_string...
        messb_ausl_string...
        filter_string...
        st_filter_string...
        horizontale...
        vertikale...
        dateiname_einpunkt...
        signallaenge_einpunkt...
        einpunktsignal...
        Fs...
        xlinks...
        xrechts...
```

```
        ymax...
        scan_schrittweite_x...
        scan_schrittweite_y...
        scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        rueckschritte_links...
        rueckschritte_oben...
        scan_dateiname...
        scan_signallaenge...
        angezeigte_frequenz...
        dateiname_messung...
        angezeigte_frequenz...
        Pyy...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y


% ------------------------------------------------------------------------
% Speichern der o.a. Variablen unter dem Dateinamen "dateiname_messung"
% ------------------------------------------------------------------------
save(dateiname_messung)
```

Anhang 5.16 scan_ymax_setzen

```
function scan_ymax_setzen

% ================================================================
% ||                                                            ||
% ||           /        __  __       \    /    _              ||
% ||          /   /\  |_  |    /\    \  /  | |_)               ||
% ||         /__ /--\ __| |__ /--\    \/   | |__)              ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% ymax aus Feld auslesen
% ---------------------------------------------------------------
ymax = str2num(get(handles.edit11,'String'));

% ---------------------------------------------------------------
% Bild neu zeichnen
% ---------------------------------------------------------------
axes(handles.axes2)
Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                   % verhindert Flackern

set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
```

```
set(handles.slider1,'Max',xrechts)

set(gca,'ZLim',[0 ymax])
colormap(hot)
colorscale = [0 ymax];

imagesc(Messwerte_Matrix(:,:,N/44100*angezeigte_frequenz-
xlinks),'parent',handles.axes2,colorscale);
colorbar
drawnow
```

Anhang 5.17 scan_ausw_flat

```
function scan_ausw_flat

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\   |_  |    /\    \  /   | |_)              ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)             ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
```

```
angezeigte_frequenz = val;


% ------------------------------------------------------------------------
% Darstellen der Matrix in sep. Fenster
% ------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                   % verhindert Flackern


figure(1)
set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];
Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));

surf(Messwerte_Matrix(:,:,(N/44100*angezeigte_frequenz-
xlinks)),'FaceColor','blue','EdgeColor','none');

camlight headlight;
lighting flat

set(gca,'FontSize',20)
Titel = [num2str((angezeigte_frequenz)) 'Hz'];
title(Titel,'Color','r');
set(gca,'FontSize',6)

view(10,50)

set(gca,'ZLim',[0 ymax])
xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])

drawnow
```

Anhang 5.18 scan_ausw_phong

```
function scan_ausw_phong

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
```

```
angezeigte_frequenz = val;


% ------------------------------------------------------------------------
% Darstellen der Matrix in sep. Fenster
% ------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                   % verhindert Flackern


figure(1)
set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];
Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));

surf(Messwerte_Matrix(:,:,(N/44100*angezeigte_frequenz-
xlinks)),'FaceColor','blue','EdgeColor','none');
    %camlight left;
    %camlight right;
camlight headlight;
lighting phong


set(gca,'FontSize',20)
Titel = [num2str((angezeigte_frequenz)) 'Hz'];
title(Titel,'Color','r');
set(gca,'FontSize',6)

    %SHADING interp
view(10,50)

set(gca,'ZLim',[0 ymax])
xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])

drawnow
```

Anhang 5.19 scan_ausw_flat_50

```
function scan_ausw_flat_50

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================


% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
```

```
val = str2num(get(handles.edit8,'String'));
angezeigte_frequenz = val;


% --------------------------------------------------------------------------------
% Darstellen der Matrix in sep. Fenster
% --------------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                    % verhindert Flackern



for i = 0:50
pause(0.1)
figure(1)
set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];

Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));

surf(Messwerte_Matrix(:,:,i+(N/44100*angezeigte_frequenz-
xlinks)),'FaceColor','blue','EdgeColor','none');

camlight headlight;
lighting flat

set(gca,'FontSize',20)
Titel = [num2str(i+(angezeigte_frequenz)) 'Hz'];
title(Titel,'Color','r');
set(gca,'FontSize',6)

view(10,50)

set(gca,'ZLim',[0 ymax])
xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])

drawnow
end;
```

Anhang 5.20 scan_ausw_phong_50

```
function scan_ausw_phong_50

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /    _             ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================


% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
```

```
val = str2num(get(handles.edit8,'String'));
angezeigte_frequenz = val;


% --------------------------------------------------------------------------------
% Darstellen der Matrix in sep. Fenster
% --------------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                    % verhindert Flackern

for i = 0:50

figure(1)
set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];

Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));

surf(Messwerte_Matrix(:,:,(i+N/44100*angezeigte_frequenz-
xlinks)),'FaceColor','blue','EdgeColor','none');

camlight headlight;
lighting phong


set(gca,'FontSize',20)
Titel = [num2str(i+(angezeigte_frequenz)) 'Hz'];
title(Titel,'Color','r');
set(gca,'FontSize',6)

view(10,80)

set(gca,'ZLim',[0 ymax])
xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])

drawnow
end;
```

Anhang 5.21 scan_ausw_flat_50_fit

```
function scan_ausw_flat_50_fit

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
```

```
angezeigte_frequenz = val;


% ------------------------------------------------------------------------
% Darstellen der Matrix in sep. Fenster
% ------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                    % verhindert Flackern



for i = 0:50
pause(0.1)
figure(1)
set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];

Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));

surf(Messwerte_Matrix(:,:,(i+N/44100*angezeigte_frequenz-
xlinks)),'FaceColor','blue','EdgeColor','none');

camlight headlight;
lighting flat


set(gca,'FontSize',20)
Titel = [num2str(i+(angezeigte_frequenz)) 'Hz'];
title(Titel,'Color','r');
set(gca,'FontSize',6)

view(10,80)

xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])

drawnow
end;
```

Anhang 5.22 scan_ausw_phong_50_fit

```
function scan_ausw_phong_50_fit

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
```

```
angezeigte_frequenz = val;


% ------------------------------------------------------------------------
% Darstellen der Matrix in sep. Fenster
% ------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                    % verhindert Flackern

for i = 0:50

figure(1)
set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
colormap(hot)
colorscale = [0 ymax];

Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
xlinks)*scan_signallaenge)));

surf(Messwerte_Matrix(:,:,(i+N/44100*angezeigte_frequenz-
xlinks)),'FaceColor','blue','EdgeColor','none');

camlight headlight;
lighting phong


set(gca,'FontSize',20)
Titel = [num2str(i+(angezeigte_frequenz)) 'Hz'];
title(Titel,'Color','r');
set(gca,'FontSize',6)

view(10,80)

xlim([0.999 x_schleife_ende])
ylim([0.999 y_schleife_ende])

drawnow
end;
```

Anhang 5.23 scan_fit_in

```
function scan_fit_in

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _             ||
% ||          /   /\   |_  |    /\    \  /   | |_)              ||
% ||         /__ /--\ __| |__ /--\    \/    | |__)             ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ---------------------------------------------------------------
% Benötigte globale Variablen, GUI initialisieren
% ---------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...

handles = guihandles(gcbo);

% ---------------------------------------------------------------
% angezeigte Frequenz auslesen
% ---------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
if isnumeric(val) & length(val)==1 & val >= get(handles.slider1,'Min') & val <=
get(handles.slider1,'Max')
    set(handles.slider1,'Value',val);
    angezeigte_frequenz = val
else
    set(handles.edit8,'String','Fehler');
end
```

```
% ------------------------------------------------------------
% Bild neu zeichnen mit ymax = Maximum(Matrix)
% ------------------------------------------------------------
axes(handles.axes2)
Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                    % verhindert Flackern

set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge) 0.01])
set(handles.slider1,'Min',xlinks)
set(handles.slider1,'Max',xrechts)
mm = max(max(Messwerte_Matrix(:,:,N/44100*angezeigte_frequenz-xlinks)))
set(gca,'ZLim',[0 mm])
colormap(hot)
colorscale = [0 mm];

Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) = ...
            flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
            xlinks)*scan_signallaenge)));

imagesc(Messwerte_Matrix(:,:,N/44100*angezeigte_frequenz-
xlinks),'parent',handles.axes2,colorscale);
colorbar
drawnow
set(handles.edit11,'String',mm);
ymax = mm
```

Anhang 5.24 scan_ausw_sep_fig

```
function scan_ausw_sep_fig

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
```

```
angezeigte_frequenz = val;


% ------------------------------------------------------------------------
% Darstellen der Matrix in sep. Fenster
% ------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                    % verhindert Flackern


Messwerte_Matrix(:,:,(N/Fs*angezeigte_frequenz)) =
flipud(Messwerte_Matrix(:,:,(N/Fs*angezeigte_frequenz)));


figure(1)

        set(gcf,'Doublebuffer','on');
        colormap(hot)
        colorscale = [0 ymax];
        set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge)
         0.01])
        set(handles.slider1,'Min',xlinks)
        set(handles.slider1,'Max',xrechts)

        Messwerte_Matrix(:,:,((angezeigte_frequenz-xlinks)*scan_signallaenge)) =
        ...
          flipud(Messwerte_Matrix(:,:,((angezeigte_frequenz-
          xlinks)*scan_signallaenge)));

        imagesc(Messwerte_Matrix(:,:,N/Fs*angezeigte_frequenz),colorscale);
        set(gca,'DataAspectRatioMode','manual')
        colorbar
        set(gca,'ZLim',[0 ymax])
        xlim([0.999 x_schleife_ende])
        ylim([0.999 y_schleife_ende])
        drawnow
```

Anhang 5.25 scan_ausw_sep_fig_glatt

```
function scan_ausw_sep_fig_glatt

% ================================================================
% ||                                                            ||
% ||           /        __  __        \    /     _            ||
% ||          /   /\  |_  |    /\   \  /  | |_)                ||
% ||         /__ /--\ __| |__ /--\   \/   | |__)               ||
% ||                                                            ||
% ||                                                            ||
% ||  Programm zur Steuerung des Laser Scanning Vibrometers    ||
% ||              im Rahmen der Examensarbeit                   ||
% ||                                                            ||
% ||       "Aufbau eines Scanning-Laservibrometers zur         ||
% ||         Visualisierung schwingender Oberflächen"          ||
% ||                                                            ||
% ||                  Februar - Juni 2004                       ||
% ||                                                            ||
% ||                    Tobias Asendorf                         ||
% ||                   tobias@asendorf.de                       ||
% ||                                                            ||
% ================================================================

% ------------------------------------------------------------------------------
% Benötigte globale Variablen, Initialisieren des GUI
% ------------------------------------------------------------------------------
global  scan_schritte_x...
        scan_schritte_y...
        scan_v_x...
        scan_v_y...
        scan_signallaenge...
        scan_dateiname...
        scan_signal...
        xlinks...
        xrechts...
        ymax...
        angezeigte_frequenz...
        N...
        x_schleife_ende...
        y_schleife_ende...
        Y...
        Fs...


handles = guihandles(gcbo);


% ------------------------------------------------------------------------------
% Der Wert aus Slider1 wird als Zahl ausgelesen und als String in das Edit-Feld
% für die angezeigte Frequenz geschrieben
% ------------------------------------------------------------------------------
set(handles.edit8,'String',num2str(get(handles.slider1,'Value')));


% ------------------------------------------------------------------------------
% Der Wert für die angezeigte Frequenz wird ausgelesen und in eine
% Hilfsvariable 'val', sowie in 'angezeigte_frequenz' geschrieben
% ------------------------------------------------------------------------------
val = str2num(get(handles.edit8,'String'));
```

```
angezeigte_frequenz = val;


% --------------------------------------------------------------------------
% Berechnen des Durchschnitts für Frequenzen +- 10 Hz um die angezeigte
% Frequenz
% --------------------------------------------------------------------------

Messwerte_Matrix = Y;
set(gcf,'Doublebuffer','on');                   % verhindert Flackern

durchschnitt(:,:,angezeigte_frequenz-xlinks)...
    = (Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-1))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-2))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-3))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-4))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-5))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-6))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-7))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-8))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-9))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks-10))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+1))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+2))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+3))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+4))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+5))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+6))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+7))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+8))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+9))...
    + Messwerte_Matrix(:,:,(angezeigte_frequenz-xlinks+10)))/21;

durchschnitt(:,:,angezeigte_frequenz-xlinks) =
flipud(durchschnitt(:,:,angezeigte_frequenz-xlinks));



figure(1)



        set(gcf,'Doublebuffer','on');
        colormap(hot)
        colorscale = [0 ymax];
        set(handles.slider1,'SliderStep',[1/((xrechts-xlinks)*scan_signallaenge)
          0.01])
        set(handles.slider1,'Min',xlinks)
        set(handles.slider1,'Max',xrechts)
        imagesc(durchschnitt(:,:,angezeigte_frequenz-xlinks),colorscale);
        set(gca,'DataAspectRatioMode','manual')
        ymax = str2num(get(handles.edit11,'String'))
        colorbar
        set(gca,'ZLim',[0 ymax])
        xlim([0.999 x_schleife_ende])
        ylim([0.999 y_schleife_ende])
        drawnow
```

Ich bin damit einverstanden, dass die von mir gefertigte Hausarbeit mit dem Thema „Aufbau eines Scanning-Laservibrometers zur Visualisierung schwingender Oberflächen“ zur Einsicht durch andere Personen zur Verfügung gestellt wird. Ich habe auch keine Bedenken, dass meine Hausarbeit Interessenten ausgeliehen wird. Mir ist bekannt, dass eine Ausleihe erst 5 Jahre nach Ablauf des Kalenderjahres möglich ist, in dem mir das endgültige Ergebnis der Prüfung mitgeteilt worden ist.

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Ort, Datum	Unterschrift