Praxissemesterbericht

Erstes Industriepraktikum WS 2002/2003

Studiengang Computer-Engineering
an der

von Andreas Amann
am

der

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

in Zürich (CH)

Inhaltsverzeichnis

1. Einführung

Der Studiengang Computer-Engineering an der Fachhochschule beinhaltet neben der Vermittlung des Theoretischen Wissens auch zwei Praxissemester. In diesen Praxissemestern soll das Wissen ,welches man an der Fachhochschule erlernt hat , in einem Unternehmen in die Praxis umgesetzt werden. Dabei hat man die Möglichkeit in folgenden Bereiche zu arbeiten:

- Softwareentwicklung
- Hardwareentwicklung
- Systemadministration
- Systembetreuung

Meine Vorliebe galt der Softwareentwicklung oder der Mitentwicklung von Embedded-Systemen.

2. Wie kam ich zur ETH-Zürich

Im Verlauf des zweiten Semesters stellte sich die Frage, wo ich mein Praxissemester absolvieren möchte. Nach der Auswahl verschiedener Firmen im Südbadener-Raum, versuchte ich mein Glück im Grenznahenbereich der Schweiz. Dort wurde ich auch fündig.
Das Institut für virtuelle Produktion [I-01] an der Eidgenössichen Hochschule in Zürich [I-02] fiel mir durch Ihren sehr positiven Ruf auf. Nach diversen Emails wie auch Telefonanrufen, konnte sobald ein Vorstellungstermin vereinbart werden. Während dem Treffen, wurden neben der „Qualifikation“ auch diverse vertragliche Punkte, wie zum Beispiel: Dauer des Praktikums und Umfang des Gleichen geregelt. Eigentlich war schon alles was es zwischen meiner Person und dem Institut zu regeln gab erledigt, doch leider liess sich die Schweizerische Fremdenpolizei des Kantons Zürich mit der Austellung der notwendigen Papiere ein bisschen mehr Zeit wie erwartet. So dass sich das Praxissemester um einen Monat verschoben hat.

3. Informationen zum Institut und Praktikum

3.1 Allgemeines

Das Praxissemester dauerte vom 16.09.2002 bis zum 16.01.2003. Die wöchentlich zu erbringe Arbeitszeit betrug 40 Stunden. Als Urlaubszeit wurde die Zeit zwischen heilig Abend und dem 4.01.2003 gewählt, da in dieser Zeitspanne das Institut sowieso geschlossen war.

Gleich zu Begin des Praktikums wurden mir sämtliche Mitarbeiter und Hilfsassistenten des Institues vorgestellt. Nach einer kurzen Führung durch das Gebäude wurde mir dann auch ein Arbeitsplatz zur Verfügung gestellt und diverse Zugänge für die Windows- und Unixumgebungen freigegeben, wie aber auch für den lokalen FTP und Fileserver.

Mein Ansprechpartner während des ganzen Praktikums war der stellvertretende Institusleiter Prof. Dr.Pavel Hora .

3.2 Profil der ETH

Die Eidgenössische Technische Hochschule Zürich (ETH Zürich) wurde im Jahre 1854 von der Schweizerischen Eidgenossenschaft als Polytechnikum gegründet und 1855 in Zürich eröffnet. Sie war bis 1969 die einzige Bundeshochschule der Schweiz.
83 Institute, Laboratorien sowie 330 Professuren und rund 840 Lehrbeauftragte pro Semester sind an der ETH Zürich die Träger von Forschung und Lehre. Über 7500 Mitarbeiter - darunter rund 25% Frauen - sind in Forschung und Lehre sowie in der Verwaltung tätig. Die Statistik der ETH Zürich weist derzeit etwa eingeschriebene 11'700 Studierende aus.

3.3 Profil des Institutes für virtuelle Produktion

Das Institut für virtuelle Produktion hat derzeit 27 Mitarbeiter und ihren Sitz ebenfalls in Zürich.
Aufgrund der Grösse und des Umfangs der Forschungs- und Testumgebungen wurde auch eine Aussenstelle in Zürich Hardturm bezogen. Neben dem reinen Lehrbetrieb investiert das Institut auch sehr viel Zeit mit Forschungsprojekten in Zusammenarbeit mit der Industrie. In Kooperation mit den sogenannten Spin-Off Unternehmen [I-03] können dabei auch grösste und fortschrittlichste Projekte verwirklicht werden.

3.4 Organisation / Abläufe innerhalb des Institutes

Im grossen und ganzen ist die Arbeitsatmosphäre sehr angenehm. Man bekommt nirgends das Gefühl, als würde der „Chef“ direkt hinter einem mit der Peitsche stehen. Es wird deshalb sehr grossen Wert auf die selbständige Arbeit und Problemlösung gelegt, wobei das nicht heisst, das man bei einem Problem keinen Arbeitskollgen fragen darf. Eins der aufälligsten Arbeitsweisen war das unkonventionelle „Meeting“. Beim sogenannten „z-Nünii“, das entspricht unserer Vesperpause, wurden diverse Projektmeetings abgehalten.
Die Meetings hatten nicht unbedingt immer eine feste Zeit, es wurde abgehalten wie es nötig war. Wenn etwas besprochen musste, so hat man dies schnell und unkomplizert direkt mit den entsprechenden Personen geklärt.

4. Produkte und Tätigkeiten des Institutes

Die Software die an diesem Institut entwickelt wird, kann ich folgende Klassen seperiert werden.

- Planungs-Systeme

  Diese Tools werden bei der Konstruktion von Halbzeugen verwendet. Folgende Programme wurde in diesem Bereich am Institut entwickelt

  - IHU-Plan
  - Fineblanking-Plan
  - Roll-Plan

- Prozess-Simulationen

  Die Prozesssimulation überprüft Konstruktionsvorschläge der Werkzeugplanung auf ihre Durchführbarkeit. Basierend auf der Finite-Elemente Methode werden die geometrischen Formänderungen während des Umformprozesses simuliert. Die Berechnungsalgorithmen greifen dabei auf Daten aus der Gefüge- und Tribosimulation zurück, die wiederum aufgrund der geometrischen Verformung berechnet werden. Das Schwergewicht des Forschungsbereichs Prozesssimulation liegt auf dem Gebiet der Beschreibung der Berechnungsalgorithmen, welche die geometrischen Formänderungen möglichst realistisch beschreiben sollen. Diese Algorithmen basieren auf Formänderungs- und Spannungs- hypothesen. Durch die Prozesssimulation berechnete Daten geben Auskunft über Spannungs- und Dehnungsverteilungen. Die Resultate dieser Berechnungen werden über ein graphisches User-Interface dargestellt.

  - Ex-Form
  - Streck-Form
  - Press-Form
  - Pre-Form

- Tribo-Simulationen

  Für die Industrie ist die Tribologie, d.h. Reibung und Verschleiss, wichtig, da sie das Umformen und Gleiten wesentlich beeinflussen. Untersuchungen in diesem Bereich haben gezeigt, dass der Kontaktbereich ein sehr komplexes dynamisches System darstellt. Unterschiedliche Reibungs-formen, wie Festkörperreibung, Grenzreibung und hydrodynamische Reibung liegen in unterschiedlichen Grössenverhältnissen vor, die sich während des gesamten Prozesses laufend ändern. Im Rahmen des Forschungsprojekts Tribodesign werden verschiedene Programme entwickelt, mit dem komplexe Tribosysteme untersucht werden können. Einerseits wird durch gezielte Modellierung der Oberflächen ein optimales Tribosystem ausgelegt. Andererseits ermöglichen Simulationen, den Reibungsprozess zu untersuchen und optimale Lösungen für

Tribosysteme zu entwickeln. Mit Hilfe eines Finiten-Elemente-Programms werden Tribosysteme im Mikrometerbereich simuliert, wobei verschiedene Parameter wie lokale Spannungen während dem Reibprozess ermittelt werden. Ein Molekular- Dynamik Programm simuliert die Festkörper- und Grenzreibung im Nanometerbereich. Die Verbindung dieser Rechenpakete erlauben ein gezieltes Design von Tribosystemen.

Für die Überprüfung der Simulationen stehen verschiedene Analyse- und Versuchsapparaturen zur Verfügung. Neben einem laseroptischen Oberflächenmessgerät (UBM) befindet sich auch eine Streifenziehanlage im Besitz des IfU. Ausserdem wird ein Atomic Force Mikroskop (AFM) für Experimente im Nanometerbereich verwendet.

5. MatForm-QT

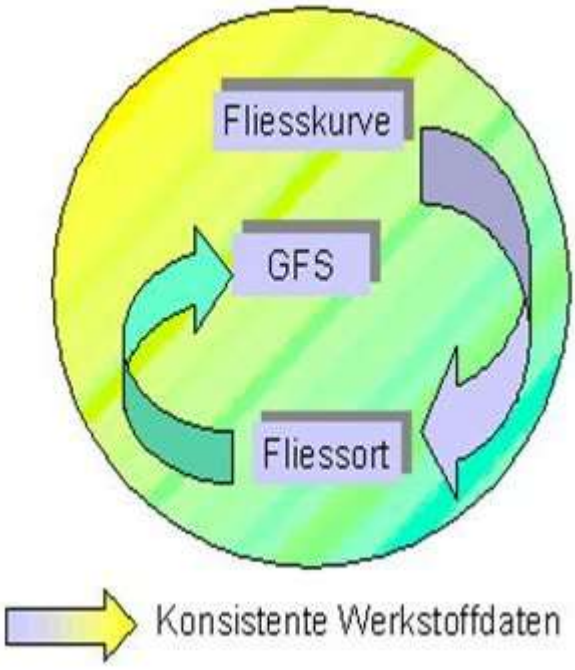
5.1 Was ist Matform

Das Projekt an dem ich arbeitete heisst MatForm. Es ist in der obengenannten Auflistung noch nicht vorhanden, da das Programm immernoch in der Entwicklerphase ist. MatForm-Qt ist ein Programm zur Bildung mathematischer Werkstoffmodelle basierend auf den experimentellen Daten unterschiedlichster Versuche.

Dank einer FEM-spezifischen Ausgabe können die Werte später direkt als Input für die jeweiligen FEM-Programme eingesetzt werden.
Eine zentrale Eigenschaft von MatForm ist die mathematische Berechnung von Grenzformänderungskurven (GFS) nach diversen Versagensmodellen. Dies macht es möglich - ohne aufwendige Nakazima-Versuche - die GFS aus den einfach im Zugversuch zu ermittelnden

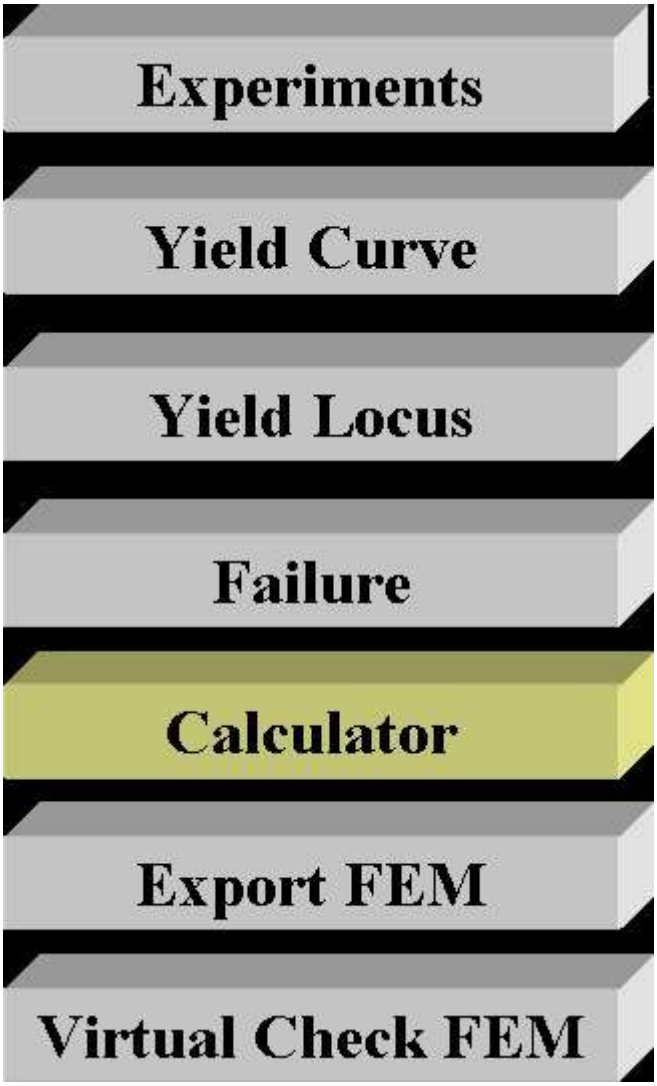
- Fliesskurvendaten und
  - Anisotropiewerten
- rechnerisch zu bestimmen.

Da die so ermittelte GFS mit den Fliesskurven- und den Fliessortdaten gekoppelt sind, kann von so genannten konsistenten Werkstoffdaten gesprochen werden. Diese sind eine Voraussetzung für eine exakte Voraussage des Versagens bei komplexen FEM-Anwendungen.



Aus welchen Modulen besteht MatForm...?

Das Matform besteht aus folgenden 7 Modulen:



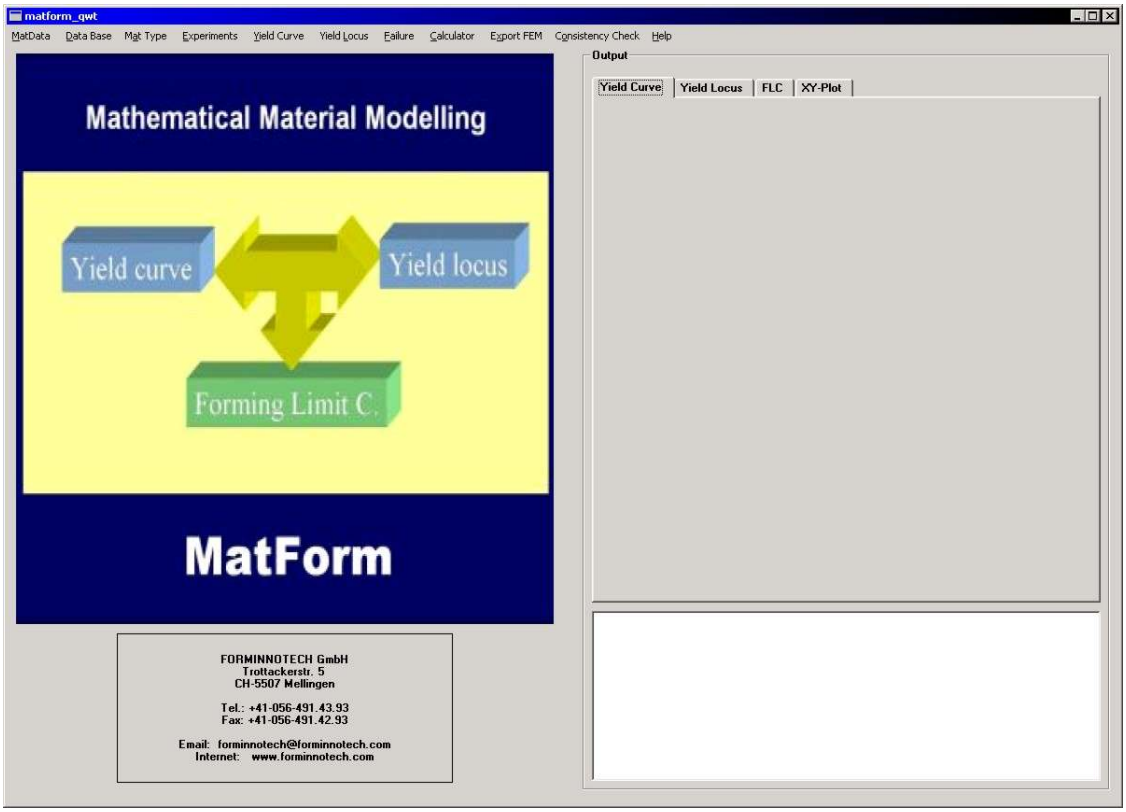
5.2 Welches sind die Einsatzgebiete von MatForm?

MatForm dient zur Erstellung von mathematischen Werkstoffmodellen. Dies ermöglicht den Einsatz für folgende Aufgaben:

- Quantitative Auswertung unterschiedlichster Experimente wie Zug, Bulge, Miyauchi, Torsionsversuch, Dilatometerversuch und Rohrversuch.
- Automatische Umrechnung der Kraft-Weg-Verläufe in eine Fliesskurve
- Mathematische Approximation der Fliesskurve mit Hilfe diverser Ansatzfunktionen zur Extrapolation der Werte über Phi-Gleichmass
- Spezifikation der Fliessortparameter für Hill`90 und Barlat-Lian aus den R-Werten
- Graphische Darstellung der Fliessortskurven für die gängigen Modelle
- Berechnung der Grenzformänderungsschaubilder (GFS) nach mathematischen VK
- Visualisierung der Zusammenhänge - Fliesskurve - Fliessort - Versagen
- Ausgabe der Werkstoffparameter in FEM-spezifischer Form für gängige FEM-Programme wie AUTOFORM und PamStamp.
- Calculator zur Berechnung von Spannungen bei vorgegebener Blechdeformation
- Calculator zur Ermittlung von Hydroforming-Limit-Curves zur Auslegung von IHU-Prozessen an Rohren

Obwohl der Einsatz von MatForm sicherlich in Verbindung mit den heute sehr verbreiteten FEM-Applikazionen zu sehen ist, ist der Einsatz des Programms auch auf Gebieten der Qualitätssicherung interessant.

Menüs
MatForm stellt nach dem Start die oben gezeigte UI-Oberfläche dar. Die einzelnen Menüpunkte beinhalten folgende Funktionen:



-MatData ..... Einlesen von bestehenden Modellen
-Experiments ..... Auswertung von experimentellen Daten
Tensile test
Miyauchi test
Bulge
Torsion
Tube
Dilatometer

-Yield Curve ..... Mathematische Approximation der Fliesskurvedaten
Approximation
Yield Curve Plot

-Yield Locus ..... Mathematische Beschreibung des Fliessortes

-Failure ..... Mathematische Berechnung der GFS Daten
MMFC
Stress limit curve

Marciniak-Kuczyski
-Calculator ..... Mathematische Berechnung der GFS Daten
Sheet 2D-RP
Tube 2D-RP

-Export ..... Ausgabe der Daten in FEM-Format

Export AutoForm
Export ExForm
Export PressForm

Einlesen von bestehenden Datenmodellen

Bestehende Datenmodelle können unter dem ersten Menüpunkt

MatData -> Open

aus dem Verzeichnis Experiments/MatData/*.mat eingelesen werden. Diese Modelle beinhalten die mathematische Flisskurvenapproximation und die Angaben zum Fliessort (s. spätere Erläuterungen). Die GFS-Daten sind auf dem File nicht enthalten und müssen unter <FAILURE> neu errechnet werden.

Die Modelle können durch Veränderung der jeweiligen Parameter abgeändertwerden um dann entweder unter

MatData -> Save as

als *.mat Modell abgespeichert oder in einem der vorgesehenen FEM Formate unter

Export

ausgeschrieben werden.

Auswertung von Experimentellen Daten

Art der Versuche

Das Programm ermöglicht unter dem Menüpunkt EXPERIMENTS die Auswertung folgender Versuchstypen

- Experiments ..... Auswertung von experimentellen Daten
- Tensile test
- Miyauchi test
- Bulge
- Torsion
- Tube

- Dilatometer

Einleseformate

Die entsprechenden Daten werden von durch den Benutzer erstellten Filen gelesen oder können direkt von Hand eingegeben werden.
Die Dateien müssen als ASCII (Text-Datei) vorliegen. Im Fall, dass diese zuvor als Excel-Dateien abgespeichert wurden, müssen sie vor dem Einlesen ins MatForm in Excel in Text-Dateien umgewandelt werden.

Wahl der Zeilen und Kolonnen

Der Benutzer hat die Möglichkeit den Datenbereich, den er lesen möchte, entweder interaktiv zu bestimmen oder die jeweiligen Angaben fest anzugeben.

Vorgehensweise A: ‚Vordefinierte Datensätze
Jedem der oben aufgezählten Versuchstypen kann ein vorgewählter Lese-Datenbereich mit folgenden Angaben zugewiesen werden:

- Kolonne X
- Kolonne Y
- Zeile Start
- Zeile Ende
- Faktor X
- Faktor Y

Die entsprechenden Angaben sind auf den File
Matdata.Exptypes
abgespeichert. Das File weist in der Version 0.0.3 folgendes Format auf:

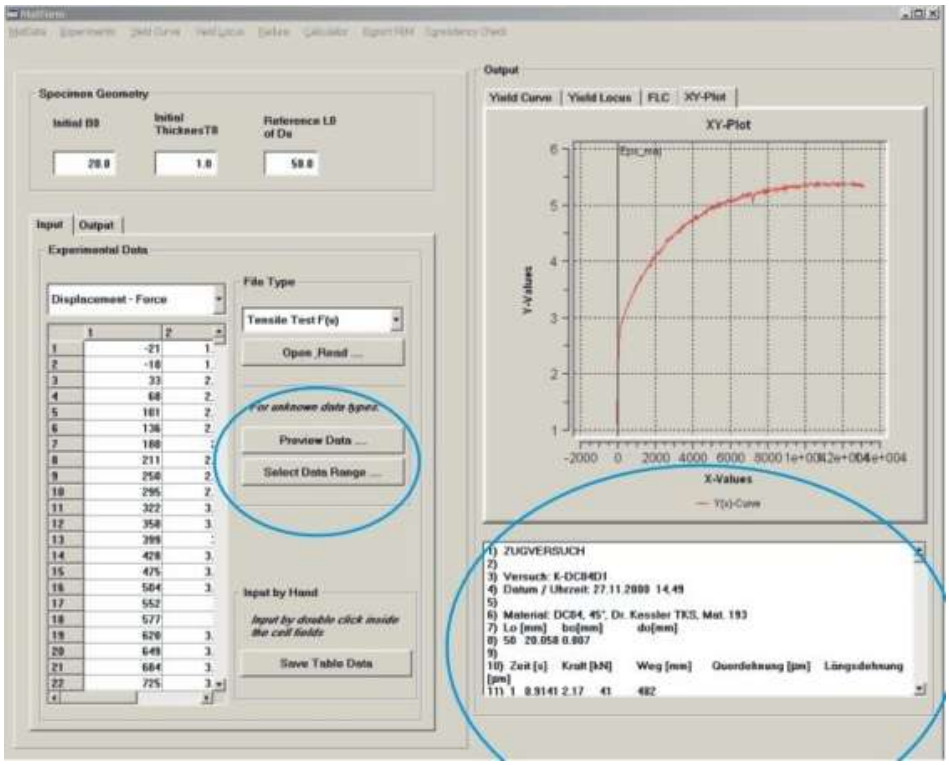
```
# ------------------------------------------------------------
#  Exp.Name             L1      LE      Kx      Ky    FakX  FakY
#123456789.123456789.123456789.123456789.1234567890
# ------------------------------------------------------------
"Tensile Test F(u)"     11      EOF     5       2     0.001 1000.
"Tensile Test R(u)"     11      eof     5       4     0.001 0.001
"Miyauchi Test"         11      eof     5       2     1     1
"Bulge Test"            11      eof     5       2     1     1
"Torsion Test"          11      eof     5       2     1     1
"Tube Hydro Test"       65      eof     3       2     1.    0.1
"Dilatometer Test"      11      eof     5       2     0.001 1000.
"kf-Curve log/Cauchy"   11      eof     2       3     1     1
"kf-Curve technical "   11      eof     2       3     1     1
```

Bitte beachten Sie, dass die Gross-Klein-Schreibung bei den Code-Words entscheidend ist und nicht verändert werden darf.

Vorgehensweise B: ‚Interaktive Wahl der Datensätze

Liegt ein neuner Datensatz vor, so kann mit Hilfe einer Preview-Option das File zuerst im rechten unteren Fenster eingelesen werden.

1.)Preview des Datensatzes



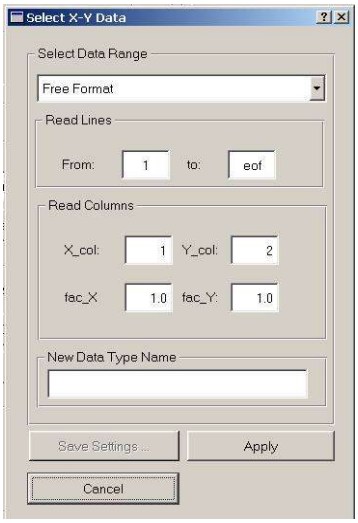
Mit Hilfe des rechten unteren Scroll-Fensters kann der Bereich, der gelesen werden soll, spezifiziert werden.

Wahl des Bereiches

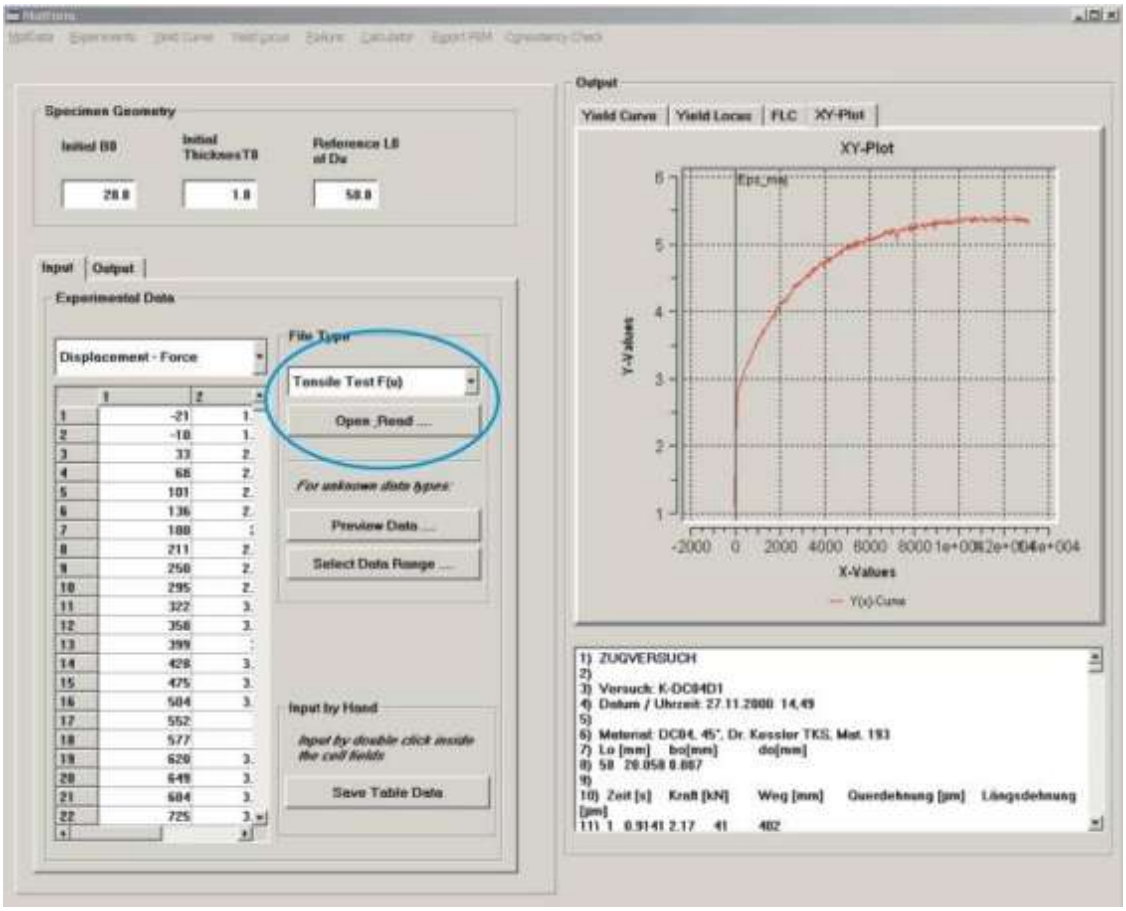
Die notwendigen Angaben

- Zeile Anfang
- Zeile Ende (eof = ‚Lesen bis zum Ende')
- Kolonne X
- Kolonne Y
- FakX
- FakY

können wie folgt auch interaktiv definiert werden:



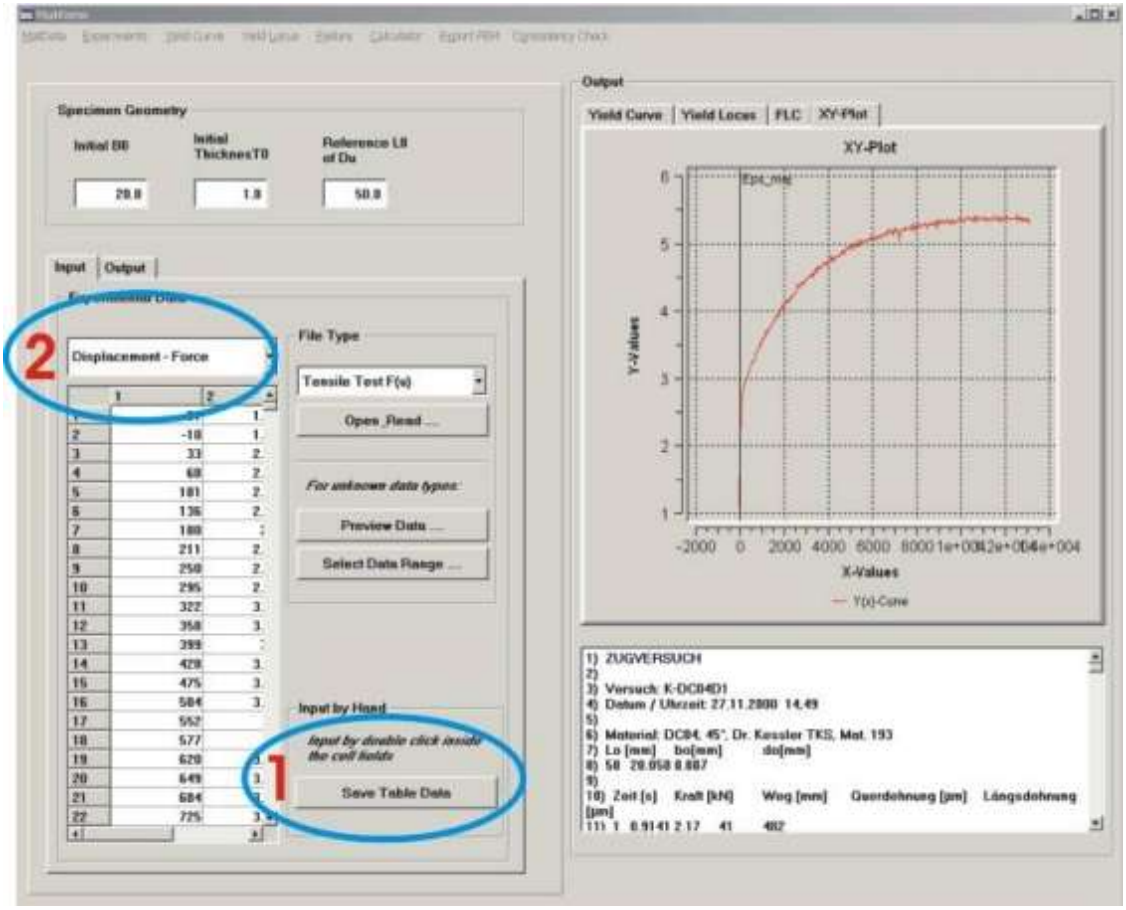
Nach erfolgter Spezifikation des Bereiches werden die Daten über den Befehl Open & read eingelesen und graphisch als x-y- Plot dargestellt.



Wir empfehlen den Anwendern trotz dieser Möglichkeit, die Datenformate Ihrer Versuche nach Möglichkeit zu standardisieren, so dass der Eintrag der festen Datenbereiche im File <Matform.Exptypes> genutzt werden kann.

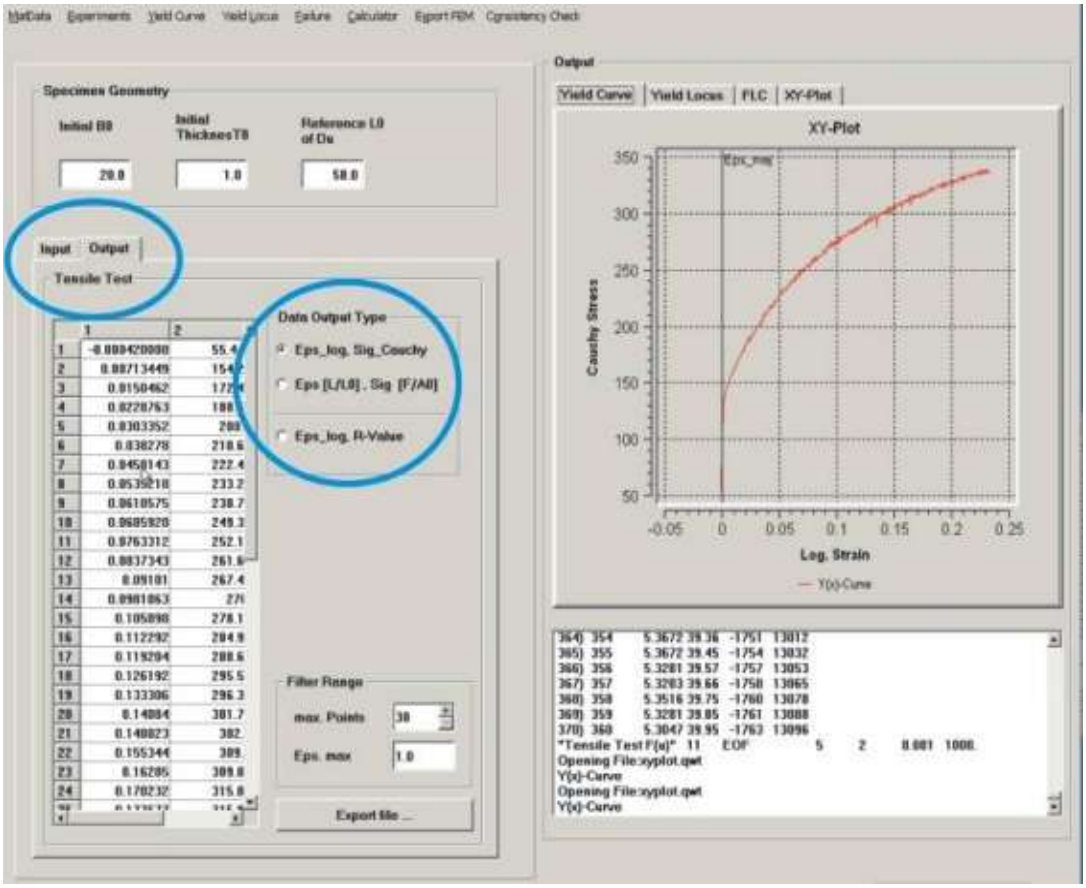
Vorgehensweise C: ‚Manuelle Eingabe'

Es ist ebenfalls möglich, die entsprechenden Wertepaare direkt in die Tabelle von Hand einzutragen. Der Input wird durch ein Doppelclick in dem jeweiligen Tab-Fenster aktiviert (1). Damit das Programm weis, um welche Daten es sich handelt, müssen diese in dem über der Tabelle liegenden Menü (2) spezifiziert werden



Auswertung der Fliesskurven-Daten

Die experimentelle Daten stellen in der Regel Kraft-Weg-Verläufe dar. Um die Fliesskurve zu bestimmen, müssen diese in Spannungen und Dehnungen umgerechnet werden.



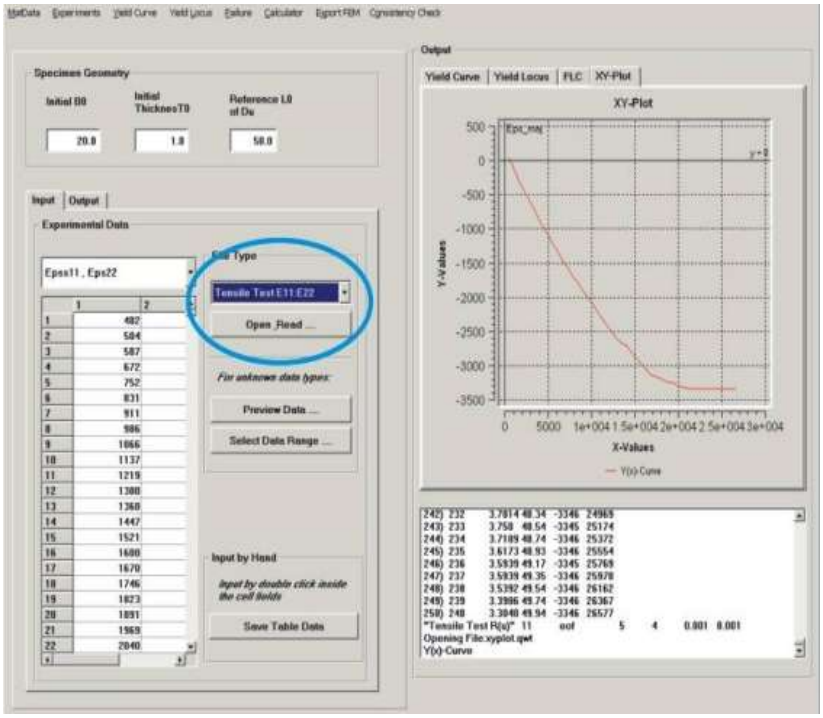
Zu diesem Zweck wählen Sie unter Output eine der beiden Optionen

- Eps-Log / Sig_Cauchy
- Eps [L/Lo] / Sig [F/Ao]

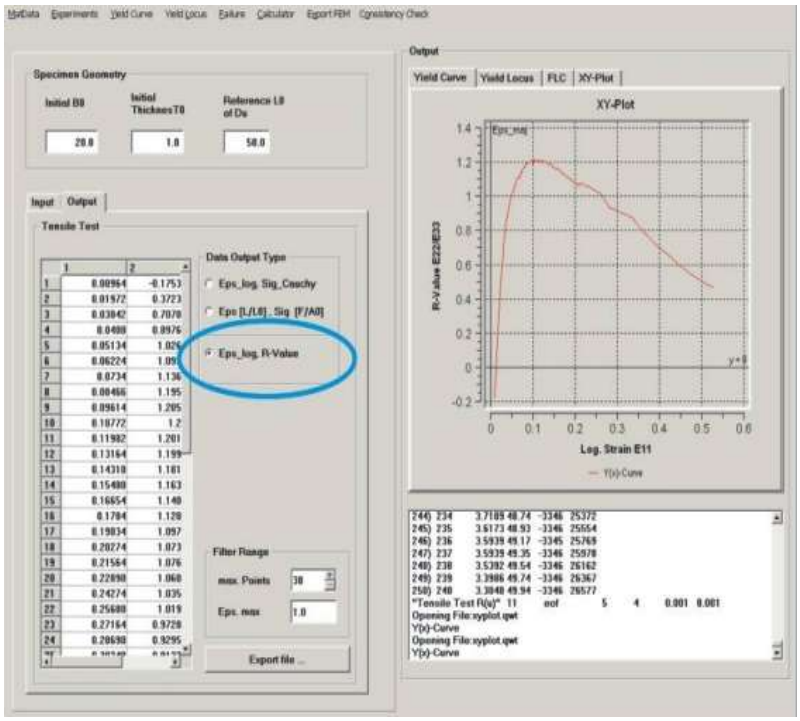
Im ersteren Fall erfolgt die Umrechnung in die logarithmische Dehnung und die Wahre Spannung. Im zweiten Fall in die weniger übliche technische Dehnung und die technische (auf den Ausgagsquerschnitt bezogene) Spannung.
Bei der Spannungsauswertung ist unbedingt zu beachten, dass die geometriedaten der Probenabmessung (Eingabefelder oben links Specimen Geometry) der verwendeten Probengeometrie entsprechen müssen.

Auswertung der Anisotropie-Daten

Liegen vom Zugversuch sowohl die Quer- als auch die Längsdehnung vor, so kann ebenfalls der Verlauf des R-wertes in Funktion der Dehnungsgeschichte dargestellt werden. Bei der Eingabe sind die E11-E22-Daten durch Wahl der 2. Option einzulesen

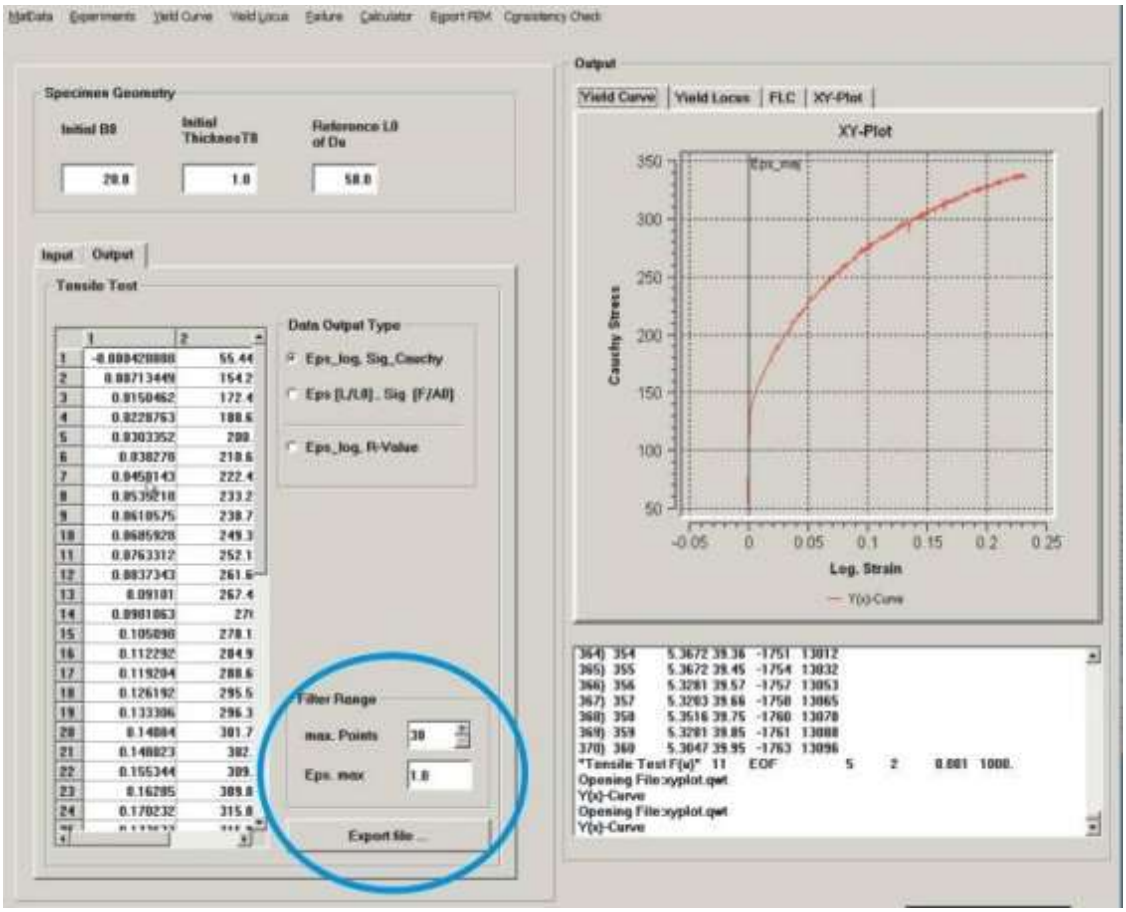


Die Umrechnung zu R-Werten erfolgt unter der Output -Option durch Wahl der Eps-log / R-Wert -Ausgabe:



Ausgabe der Fliesskurvendaten - Schnittstelle zur FLK-Approximation

Die im Zugversuch ermittelten Daten decken in der Regel nicht den Umformbereich ab, wie er bei komplexen mehrachsigen Umformzuständen auftritt. Aus diesem Grund ist eine Extrapolation der Fliesskurvendaten über den gemessenen Bereich notwendig, wozu wiederum eine mathematische Approximation des Fliesskurve notwendig ist.



Da die Messung einerseits ein sehr dichtes und streuendes Messpunktenetz liefert und andererseits die Messpunkte ausserhalb des Gleichmassbereiches nicht in die Auswertung einbezogen werden dürfen, bietet die Option zum Speichern der Daten die Parameterwahl

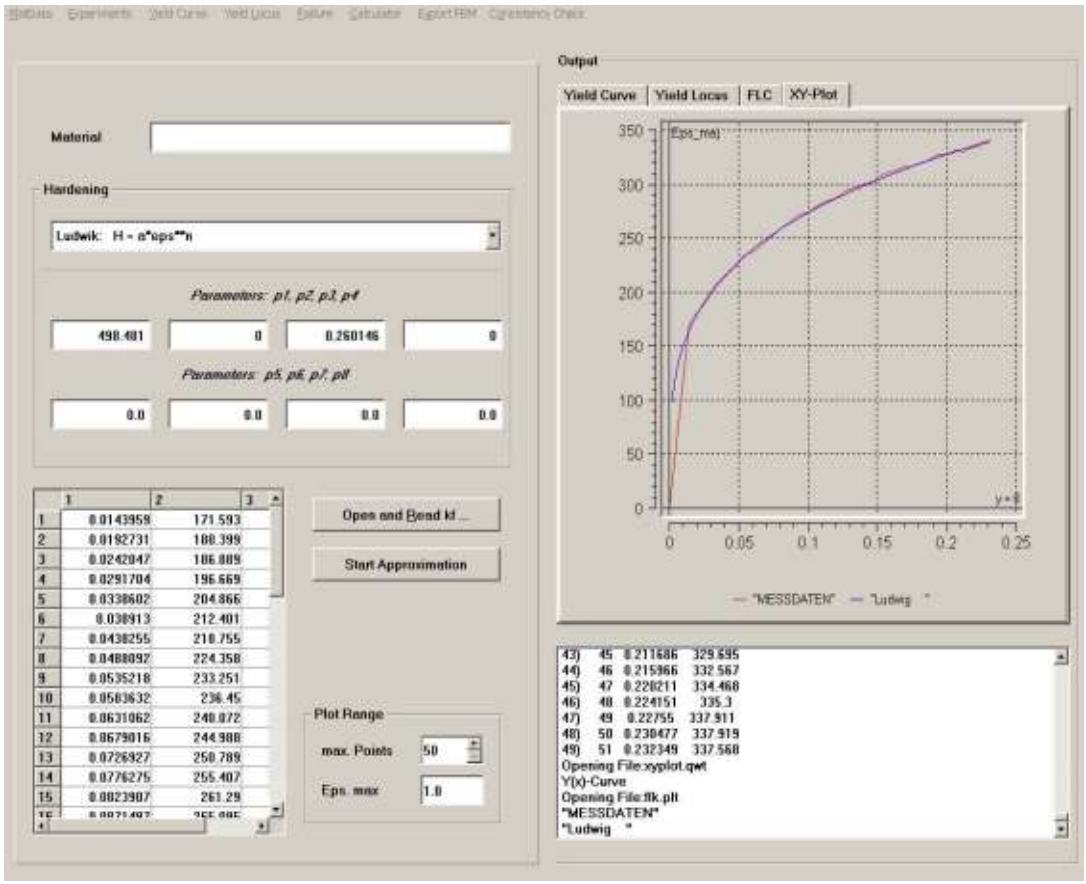
FILTER RANGE

wie auch **EPS_max**.

an.it der ersten Option wird die maximale Anzahl auszuschreibender Punkte festgelegt (per default 30) , mit EPS_max der Dehnungsbereich definiert, in dem die Messpunkte berücksichtigt werden.

YIELD CURVE - Mathematische Approximation der Fliesskurve

Die so gefilterten Daten werden unter YIELD CURVE - Open&Read kf erneut eingelesen.



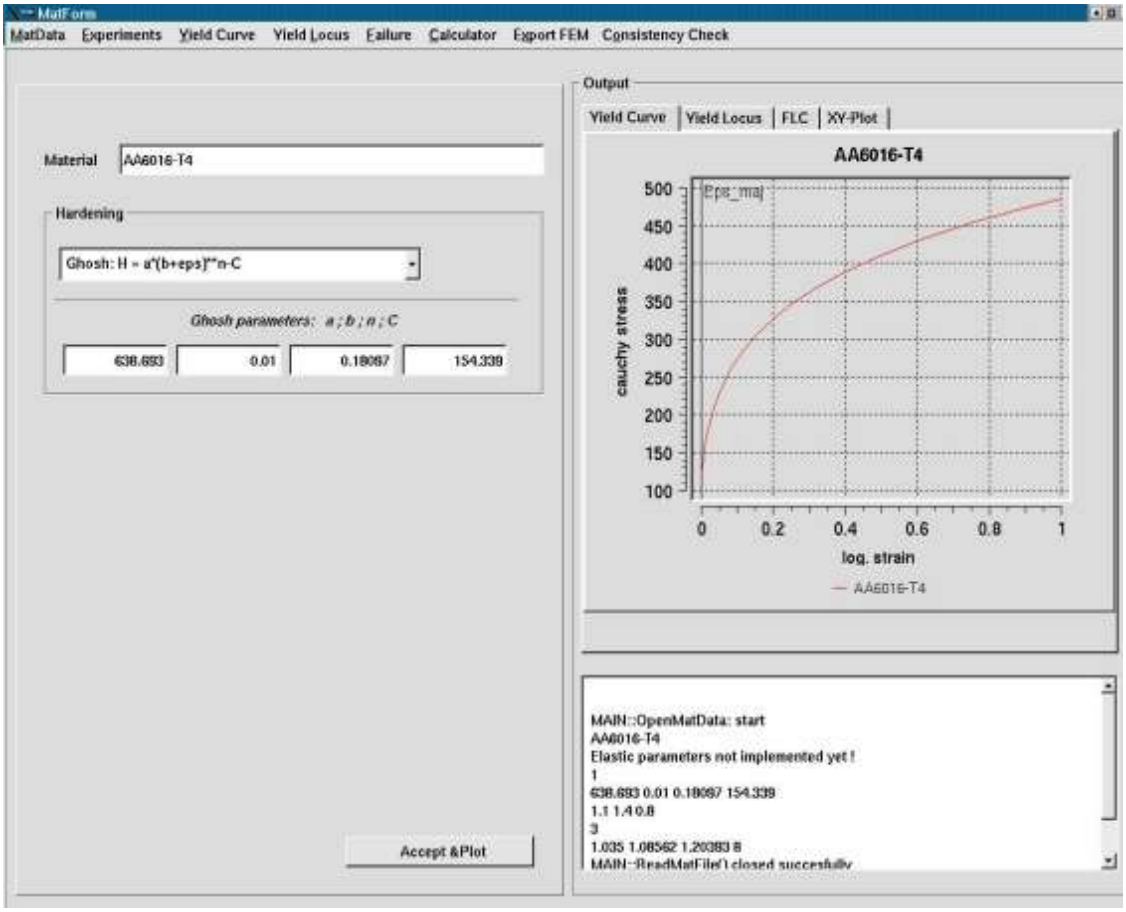
Für die mathematische Approximation sind folge Fliesskurventypen implementiert:

- Ludwik
- Swift
- Ghosh
- Voce
- Hocket-Sherby
- Ghosh-invers

Die Auswahl erfolgt durch Wahl des erwünschten Typs und der Start der Berechnung mit Start Approximation.
Nach erfolgter Approximation kann unter YIELD CURVE - Yield Curve Plot ein nochmaliges Ploten erfolgen. Da alle obigen Ansatzfuktionen sich auf die beiden Typen

- Ghosh
- Hocket-Sherby

reduzieren lassen, werden in der Folge nur noch diese beiden Approximationsarten mit der entsprechenden Nullsetzung der nicht benötigten Parameter weitergeführt.



YIELD LOCUS - Mathematische Beschreibung der Fliessortkurve

Sowohl die Anisotropie als auch der Typ des Werkstoffes beeinflusst die Form des Fliessortes. Matform ermöglicht die Eingabe und graphische Darstellung für folgende FOK-Modelle:

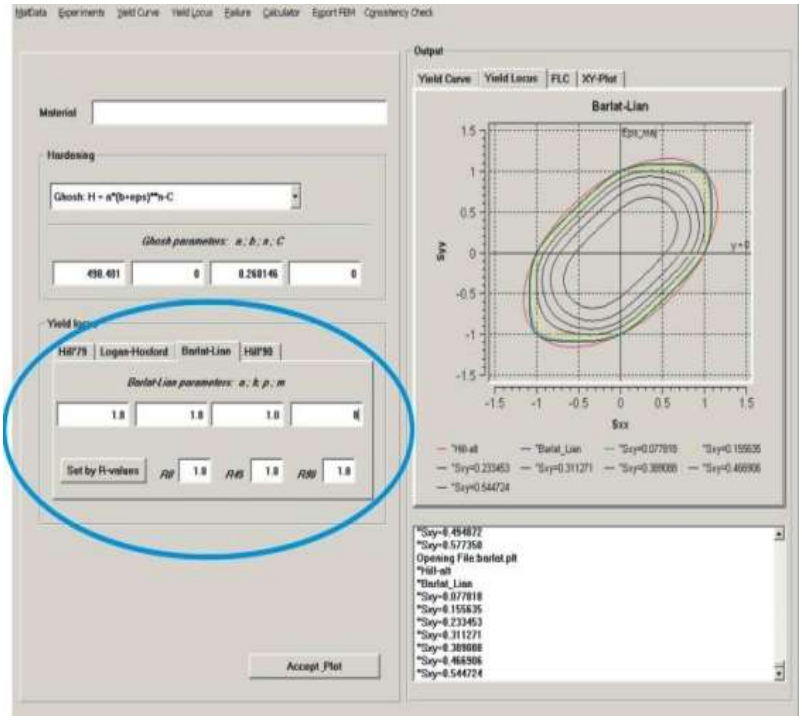
- Hill'79
- Logan-Hosford
- Barlat-Lian
- Hill`90

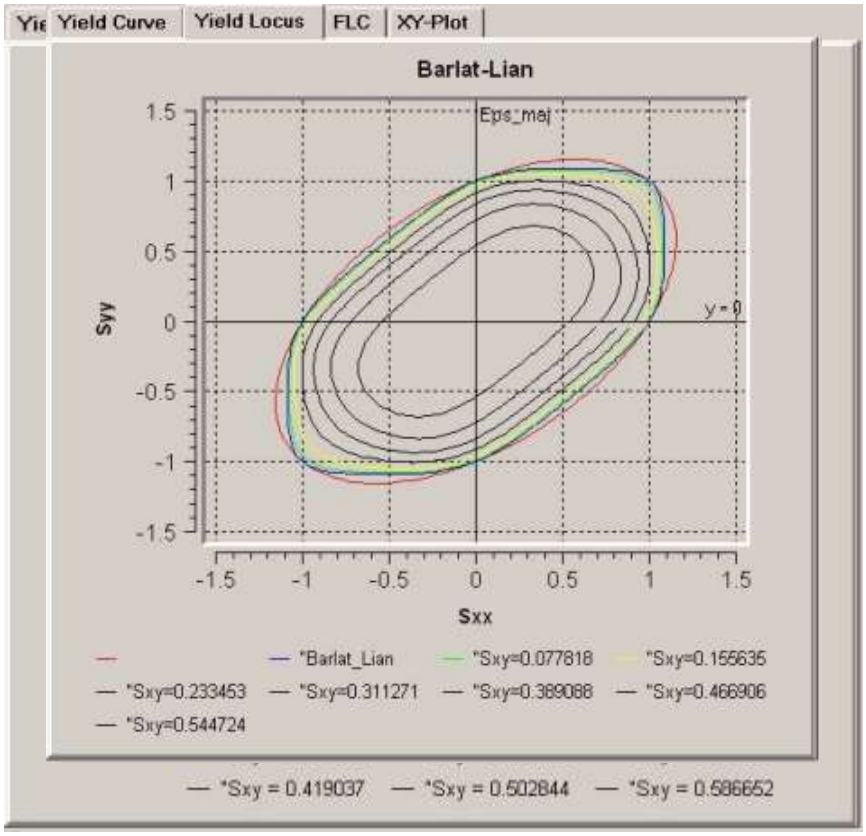
Alle obigen Fliessortmodelle erlauben die Eingabe von nicht quadratischen Exponenten, womit sie auch die nicht-quadratische Form annehmen können.
Zur Darstellung der bekannten Hill`48-Beschreibung kann z.B. die Hill`90 Beschreibung mit m=2 eingesetzt werden.

Die ersten beiden Modelle erlauben nur die Eingabe einer Normalanisotropie R.

Barlat-Lian und Hill`90 beruhen auf der Kenntnis von R0, R45 und R90.
Die Implementierung dieser beiden Modelle wurde so vorgenommen, dass die entsprechenden modellspezifischen Parameter a, h und p aus den R-Werten ermittelt werden.

Für den Barlat-Lian-Fliessort werden üblicherweise Exponenten m > 2 eingesetzt. Die Verwendung von m-Werten < 2 ist ebenfalls möglich. Diese Parameterwahl wird aber nicht empfohlen.





FAILURE - Grenzformänderungsschaubilder n. Keeler

Sind die Fliesskurve und die Fliessort bekannt, so können mit diesen Angaben die GFS rechnerisch ermittelt werden.Gegenwärtig sind folgende Versagenskriterien (VK) eingebaut

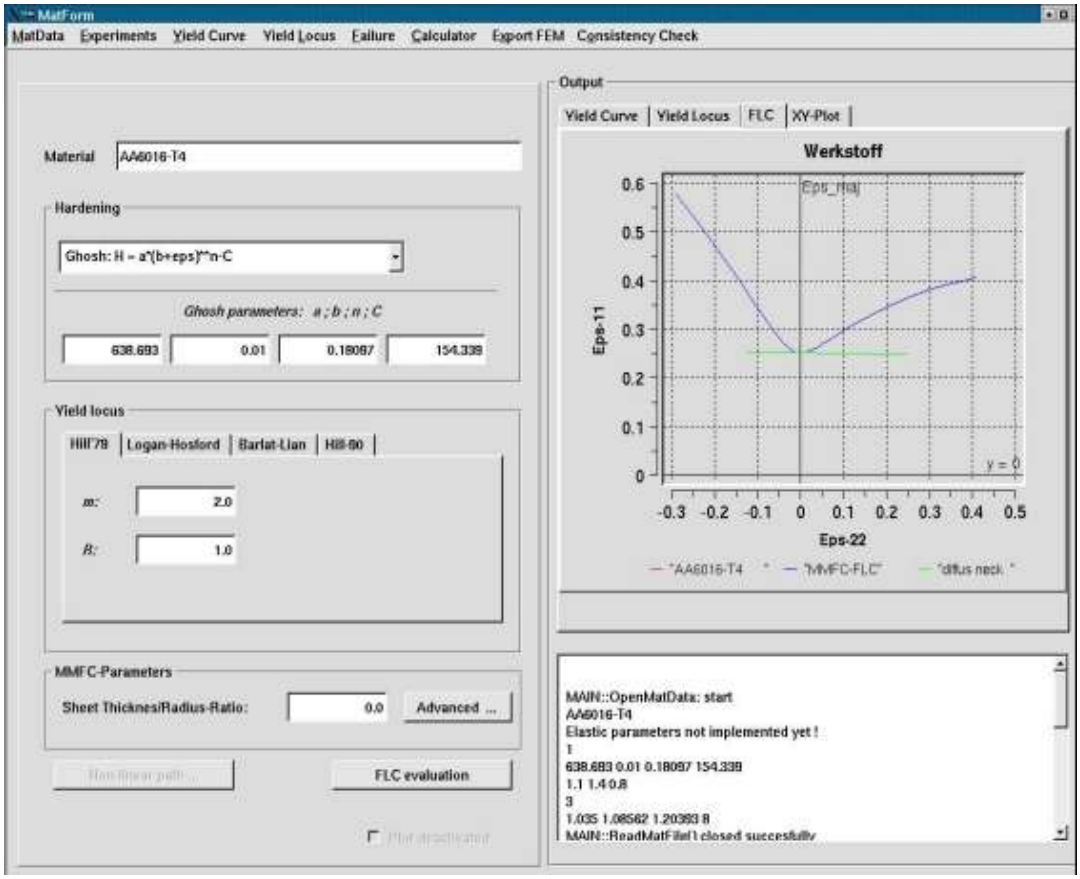
- Modified Maximum Force Criterion (MMFC n. Hora-Tong)
- Grenzspannungskriterium
- Rice (Deformationstheorie) (in Vorbereitung)
- Marciniak-Kuczynski (M-K) (in Vorbereitung)

Zu einem späteren Zeitpunkt wird auch die Implementierung von Versagenskriterien für die Massivumformung erfolgen.

An dieser Stellesoll darauf hingewiesen werden, dass die theoretischen Modelle das reale Verhalten nicht exakt beschreiben. Der Leser sei hier auf die zahlreiche Literatur zu dieser Problematik hingewiesen.
sei hier auf die zahlreiche Literatur zu dieser Problematik hingewiesen. Die Kopplung des GFS mit der Fliesskurve und dem Fliessort bietet aber vor allem die Möglichkeit zur Untersuchung von Schwankungseinflüssen einzelner Eingabeparameter, wie z.B der R-Wert-Variation

Option ‚MMFC n. Hora-Tong'

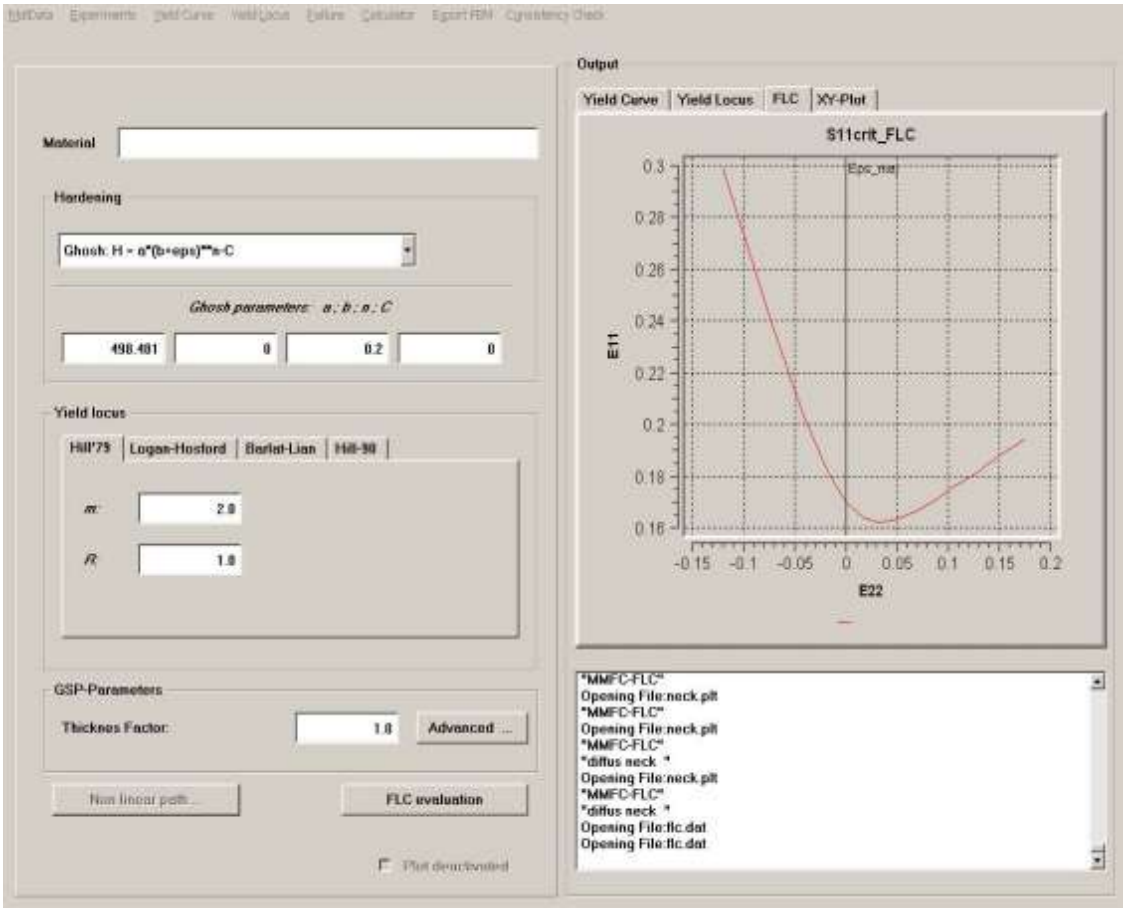
Als Standard-Kriterium ist das Modifizierte Kraftmaximum-Kriterium eingebaut.



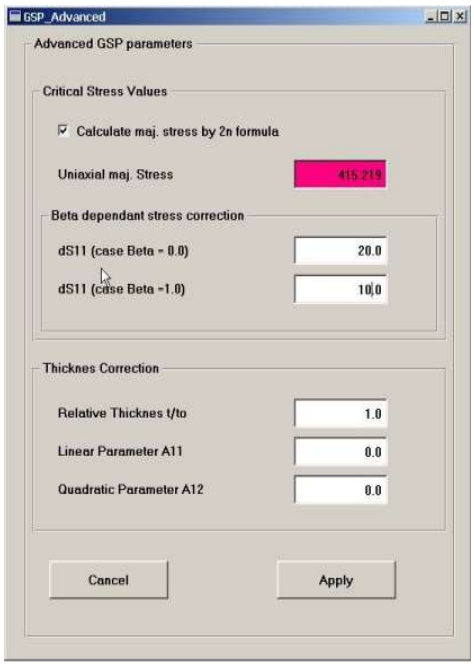
Die mit dem MMFC ermittelten Grenzwerte stellen den Beginn der lokalen Einschnürung am unendlich dünnen Blech dar und liegen erfahrungsgemäss etwas unter den experimentell ermittelten Grenzwerten. Die mit diesem konservativen Kriterium ausgelegten Prozesse sind deshalb in der Praxis gut umsetzbar. Leichte Überschreitungen der Grenzkurve bedeuten andererseits nicht, dass auch ein Versagen bis zum Blechriss auftreten wird.

Grenzspannungskriterium

Im Unterschied zum auf der plastischen Instabilität besierenden MMFC errechnet das Grenzspannungskriterium den Grenzwert an Hand einer vorgegebenen kritischen Spannung. Das untere Diagramm zeigt den GFS-Verlauf unter der Annahme eines konstanten Spannungswertes.



Mit Hilfe des ADAVANCED-Menüs kann der Grenzspannungswert einerseits vom Beta-Verhältnis E22/E11 als auch von der Dicke abhängig gemacht werden.



Obige Diagramme zeigen die so erzielte GFS-Modifikation.

Export FEM - Ausgabe der Daten in FEM-Format

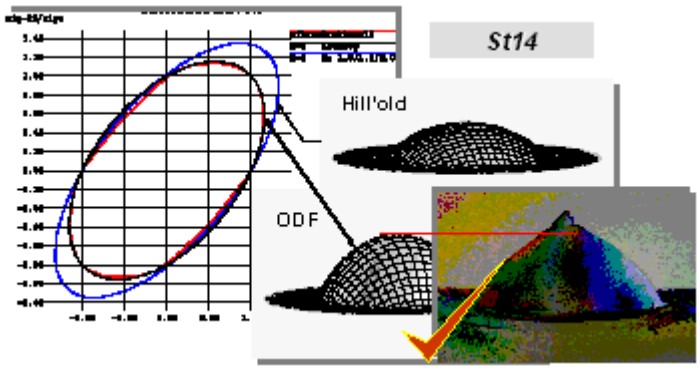
Mit der Fliesskurve, dem Fliessort und der Gernzformänderungskurve sind alle 3 Kenngrössen bestimmt, welche die FEM-Programme zur nichtlinearen plastischen Berechnung benötigen. Als Ausgabetypen sind vorgesehen

- Export AutoForm
- Export ExForm
- Export PressForm

Für die noch nicht implementierten Ausgabetypen wird wegen der Formatverwandschaft empfohlen diese direkt von der ExForm-Ausgabe abzuleiten. Die Verteilung der Fliesskurven-Stützpunkte wird so bestimmt, dass die Abweichung zwischen dem exakte Verlauf und der stützpunktbasierten linearen Interpolation nie grösser als 2 N/mm2 wird. Aus diesem Grund liegen die errechneten Stützpunkte nicht äquidistant.

Consistency Check - Virtuelle Abbildung der Experimente

Wie gut die in ‚einfachen' Versuchen ermittelten Daten in Verbindung mit den entsprechenden plastomechanischen Annahmen das reale Verhalten beschreiben, kann am besten durch eine virtuelle Abbildung gleicher oder anderer Versuche verifiziert werden. Die Grundidee dieses Abgleich ist aus dem nachfolgend Bild ersichtlich



Die Werkstoffdaten gelten erst dann als korrekt, wenn die so erzielte virtuelle Abbildung des Prozesses mit dem entsprechenden versuch wiederum korreliert.Treten Abweichungen auf, so muss das mathematische Werkstoffmodell als unexakt bezeichnet werden. Eine Verwendung für reale Berechnungen sollte mit diesen auf keinen Fall durchgeführt werden.
Solche Kontrollrechnungen können mit dem FEM-PAket ExForm oder einem anderen Softwareprodukt durchgeführt werden. Die direkte Definition einer Schnittstelle zum Programm ExForm wird zu einem späteren Zeitpunkt bereitgestellt. Vorderhand liegen die Inputdecks für eine Stand-Alone Eingabe vor.

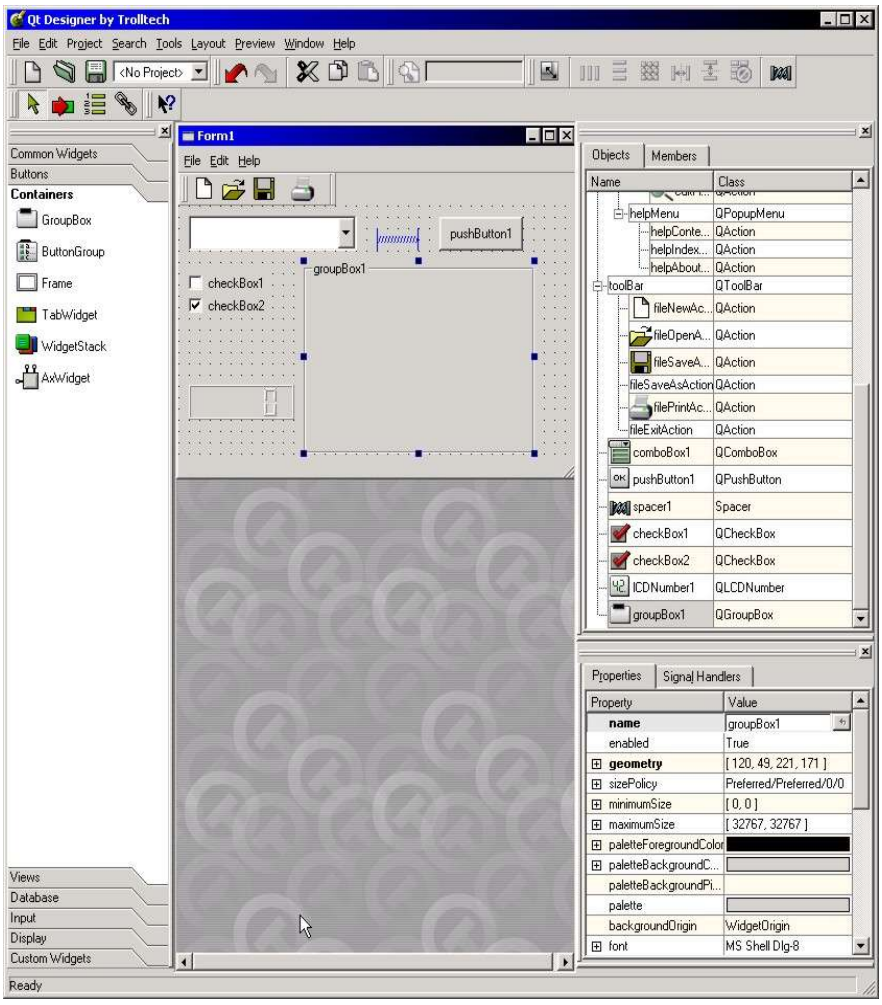
6 Tätigkeiten während des Praktikums

6.1 Verwendete Softwarepackete

Während des 4 Monatigen Praxissemsters bestand meine Hauptaufgabe darin, das Programm Matform-QT weiter zu entwickeln. Um dieser Anforderung gewachsen zu sein, bedurfte es einer gewissen Einarbeitungszeit. Während dieser Phase erlernte ich die Entwicklungsumgebung Visual Studio [I-04] , wie aber auch das Softwarepacket QT der Firma Trolltech [I-05] und auch die Zusatzlibraries QWT-PLOT [I-06] einzusetzen.

Die QT-Library, welche nachfolgend der Einfachheit halber „QT“ genannt wird, beinhaltet neben der eigentlichen Library noch sehr viel Zubehör (Dokumentation, Beispielprogramme, Zusatztools...). Im folgenden versuche ich die speziellen Softwarepackete kurz und prägnannt darzustellen.

Designer



Erwärnenswert sei hierbei das mitgeliferte Programm „Designer“ mit welchem man in der Lage ist, sich eine Grafische Benutzer Oberfläche ohne sehr viel Aufwand

zusammen zuklicken. Es ist zwar möglich sich die GUI auch ohne dem Programm zu erstellen, doch damit ersprat man sich viel Entwicklungszeit. Als einziges Manko kann man hierbei nur erwähnen, das falls es nicht möglich ist, das eigentliche Programm und die GUI strikt zu trennen , so muss man bei änderungen der GUI sehr viel von Hand ändern.

1. UIC

Das uic (User-Interface-Compiler) ist ein kleines Kommandozeilen basierdes Programm, welches die Verwendung des Designers zu Erstellung von GUIs ermöglicht. Der Designer erstellt aus dem zusammengeklickten Fenster eine XML-Datei, welche dieses Fenster exakt beschreibt. Durch das UIC wird eine der besagten Datei entsprechene C++ Source und Header Datei erstellt.

```
<widget class="QPushButton">
        <property name="name">
            <cstring>pushButton1</cstring>
        </property>
        <property name="geometry">
            <rect>
                <x>250</x>
                <y>10</y>
                <width>80</width>
                <height>30</height>
            </rect>
        </property>
        <property name="text">
            <string>pushButton1</string>
        </property>
    </widget>
```

Buttonbeschreibung, im Dateiformat des Designers

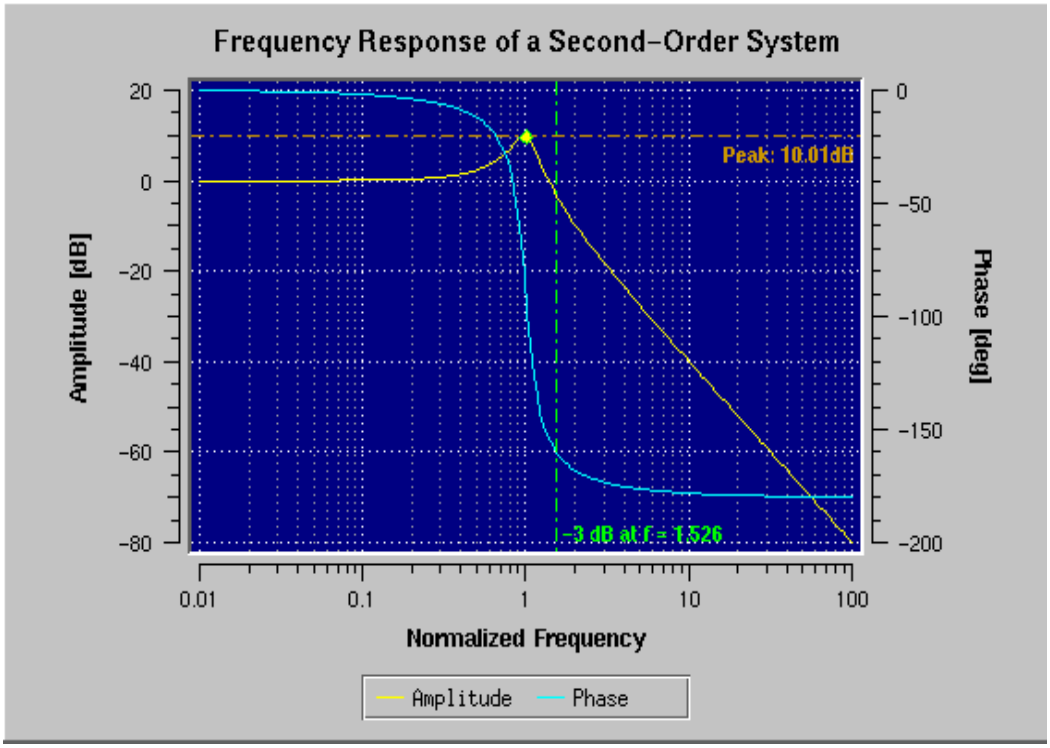
```
pushButton1 = new QPushButton( centralWidget(),
"pushButton1" );

    pushButton1->setGeometry( QRect( 250, 10, 80,
30 ) );

    pushButton1->setText( tr( "pushButton1" ) );
```

Buttonbeschreibung, als C++-Source

2. MOC

Dieses Zusatzprogramm ist elementar für die Benutzung der QT-Library. Denn die QT-API beinhaltet Schlüsselwörter welche nicht C++ konform sind. Aber durch diesen Precompiler ist die Verwendung ebenfalls möglich.

3. Qwt-Plot



Diese Library dient zur einfachen Erstellung von Plots. Diese Opensource-Lib beeindruckt durch sehr umfangreiche Einstellungsmöglichkeiten der gewünschten Plots. Nur die vorhandene Dokumentation ist nicht gerade einfach zuverstehen, aber wenn diese Hürde überwunden ist, so ist die Handhabung sehr unkompliziert.

```
double *x = new double[100];
double *y = new double[100];
double *z = new double[100];

//Befüllen der obigen Arrays mit Werten
for (i = 0; i< 100; i++)
{
 x[i] = 0.1 * double(i);
 y[i] = sin(x[i]);
 z[i] = cos(x[i])
}

// Neue Kurven erstellen
cSin = plot->insertCurve("y = sin(x)");
cCos = plot->insertCurve("y = cos(x)");

// Setzen von Kurveneigenschaften
plot->setCurvePen(cSin, Qpen(red));
plot->setCurvePen(cCos, QPen(blue));

// Kopieren der Daten
plot->setCurveData(cSin, x, y, 100);
plot->setCurveData(cCos, x, z, 100);

// Erstellen von Hilfslinien
// .. horizontale Linie y = 0
mY = plot->insertLineMarker("y = 0", QwtPlot::yLeft);
plot->setMarkerYPos(mY, 0.0);

// .. vertikale Linie x = 2 * pi
mX = plot->insertLineMarker("x = 2 pi", QwtPlot::xBottom);
plot->setMarkerXPos(mX, 6.284);

// Setzen der Axen-Beschriftung
plot->setAxisTitle(QwtPlot::xBottom, "x -->");
plot->setAxisTitle(QwtPlot::yLeft, "y -->");

// Legende einfügen
plot->enableLegend(TRUE);

// Löschen der Arrays
delete[] x;
delete[] y;
delete[] z;
```

Beispiel einer Sinus und Cosinus-Kurve mit Hilfe der QWT-Library

6.2 Programmieraufgaben

6.2.1 Grundsätzliches

Anfänglich bestand Matform ausschlieslich aus einer kalkulationsmethode. Es wurde

Programmeirt um die Mitarbeiter des Institutes von den alltäglichen recharbeiten zu entlasten. Dies gesch noch in der Programmeirsprache Fortran. Die Bedienung des Programms erfolgte ausschlieslich über die Kommandozeile. So über die jahre hinweg das Programm um diverse Funktionen erweitert. Zuerst wurden die Erweiterungen in Fortran, anschliessend in C und zum Schluss in c++ realisiert. Doch die Benutzung stellte sich durch die zusätzlichen Routinen als sehr komplex dar. Darauf entschloss man sich für das Matform-Programm eine grafische-Benutzer-Obverfläche zu programmieren. Soviel zur Vorgeschichte dieses Projektes.

6.2.2 GUI-Anpassung

Nach relativ kurzer Einarbeitungszeit begann dann solangsam auch die Produktive Phase. Während dieser Zeit wurde die bisher provisorisch angelegte GUI erweitert. Bei der ersten Problemstellung musste entschieden werden , bei welchen Auflösungen das Programm betrieben werden sollte. Wir dachten, das die Auflösung von 1024x768 nicht unterschritten werden darf, da sonst die GUI komplett umgestaltet werden musste, um alle Informationen darstellen zu können. Als Maximaldarstellung wurde die Auflösung von 1600x1200 gewählt, da diese normalerweise die grösste sich im Gebrauch befindliche Auflösung ist. Desweiteren werden die dargestellten Informationen sonst noch kleiner.

Diese Aufgabe klingt ansich nach wirklich viel Arbeit, aber weit gefehlt. Es musste grösstenteils die Grafische Oberfläche komplett neugestaltet werden. QT liefert zwar Methoden um eine automatische Grössenanpassung beziehungsweise Verschiebung bei der Änderung der Windoswsgrösse, doch diese Methoden stellen sich als unpraktikabel heraus. Denn durch diese Methoden wird der Quellcode extrem unübersichtlich , es geht sogar soweit, das man sich teilweise überhaupt nicht im Source zurechtfindet.

Als Alternative dazu Verwendete ich globale Variablen , welche sich linear zur Grössenänderung des Minimalfensterns im Vergleich zum dargestellten Fenster ändert. Dies ist zwar nicht ganz so performant wie die Trolltech´sche Variante, doch hier musste eindeutig die Entscheidung zugunsten des übersichlicheren Quellcodes gewählt werden.

6.2.3 Portierung auf Windows

Als weitere Aufgabe musste dann das Programm von Unix / Linux auf Windows Portiert werden. Durch die sehr klare und durchdachte Programmierung und die Verwendugn der QT-Library war die Portierung dann im grossen und ganzen kein wirklich grosses Problem. Nur die Verwendung der moc-files handhabte sich etwas

schwieriger als gedacht. Aber aller Anfang ist schwer.

Das nächste noch grössere Problem stellt die Einbindung der Fortran-Routinen in das Visual Studio dar. Auch dabei gab es mehrere Methoden .

Die erste beinhaltete die benutzung eines Zusatz-Tools und Libs namens f2c [I-07], welche dazu dar war , die Fortranroutinen in C bzw. C++ Dateien umzuwandeln. Diese Möglichkeit besitzt jedoch den Nachteil, das bei einer Änderung der Fortran-dateien, man erst den Fortran-Precompiler starten musste und anschliessend dieses f2c Programm , um dann die Fortran-datei entsprechend zu übersetzen .

```
        *fm = *fm;
        ndiv = 50;
        dthet = (float)3.14159265 / ndiv;
        io___16.ciunit = ixg;
        s_wsle(&io___16);
        e_wsle();
        io___17.ciunit = ixg;
        s_wsfe(&io___17);
        do_fio(&c__1, " \" R = ", (ftnlen)7);
        do_fio(&c__1, (char *)&r__,           (ftnlen)
    sizeof(real));

        do_fio(&c__1, "M = ", (ftnlen)4);
        do_fio(&c__1, (char *)&(*fm), (ftnlen)sizeof
    (real));

        e_wsfe();
        i__1 = ndiv;
        for (i__ = 0; i__ <= i__1; ++i__) {
    thet = dthet * i__ - (float)1.5708;

    if (thet < (float)-.7854 || thet > (float).7854) {
        alpha = cos(thet) / sin(thet);
        coef1 = r_sign(&c_b39, &thet) * falpha_(ifok,
    &alpha, &r__, fm);
```

Von Fortan nach c übersetzter Code (um die Komplexität aufzuzeigen)

Alternativ dazu bestand nach längerem Studieren der Intel-Fortran-Compiler-Dokumentation , wie aber auch dem Studium im MSDN, das man den Übersetzung auch automatisch bewerkstelligen konnte. Dabei wurde die Fortrandatei mittels eines Parameters durch den Intelcompiler „gejagt“. Im Anschluss daran wurde dann die compilierten Objectfiles in das aktuelle Visual-Studio-Projekt verlinkt.

Diese Variante hatte aber den entscheidenden Nachteil, das selbst ohne Änderung des Fortranquellcodes die Compilierung erfolgte. Und dies führte auf die Dauer auf eine beträchtliche Wartezeit während des Compilierens (selbst auf einem Dual Xeon Rechner mit je 2,4 Ghz). Durch die Abwägung der Vor- wie aber auch der Nachteile, wurde entschlossen, das man die kompliziertere, aber schnellere f2c möglichkeit in das Projekt übernahm. Für die Zukunft ist noch geplant die noch vorhandenen Fortrandateien neu zuschreiben. Dann würde der ganze Umstand mit f2c,... entfallen.

6.2.4 Import- / Eportfilter

Im Anschluss an die GUI-Erweiterung und die Portierung auf win32, lautete meine Aufgabe, einen Import- und Exportfilter für alternative Programme zu implementieren (Autoform [I-08] , ExForm [I-09], PressForm [I-10])

Auch dies klang auf anhieb nicht wirklich Anspruchsvoll, aber ich täuschte mich gewaltig darin. Man gab mir die einzulesenden Dateien , so das ich deren Aufbau studieren konnte. Als dann die Importfunktion entwickelt und implementiert wurde, so bekam ich dann noch weitere Dateien (z.b.Materialdaten mit Materialien aus Europa und Japan). Dabei stellte sich heraus, dass das Dateiformat nicht so einheitlich war, wie ich das ursprünglich gedacht hatte. Als sich dieser Vorfall mehrmals ereignete, wurde die Import, wie aber auch die Export routine komplett überarbeitet. Nun arbeitete der Parser nicht mehr Zeilenabhängig , sondern durchsuchte jeweils die komplette Datei nach entsprechen Schlüsselwörter. Diese Variante ziegte sich als sehr zuverlässig.

```
Material    st15, fep05, DC05
ElasticParameter      210000 0.3

ThermalParameter      0.0 0.0 0.0

Density     7.8e-005

Hardening
   0            0       144.563
   1        0.002       156.241
   2        0.005       170.508
   3        0.009       185.144
   4        0.015        202.02
   5        0.023       219.339
   6        0.033       236.334
   7        0.046       253.949
   8        0.063       272.438
   9        0.084       290.948
  10        0.111       310.398
  11        0.145       330.508
  12        0.187       351.045
  13        0.239       372.204
  14        0.303       393.988
  15        0.381       416.315
  16        0.476       439.293
  17         0.59       462.711
  18      0.72799       486.887
  19      0.89299       511.632
  20      0.99999        525.89

 FlkType    1
FlkParameter          525.255 0.004972 0.244164 0

R_Values   1.835 1.62 1.7 2.32

BiaxialStressFactor   1

FokType    3
FokParameter         0.68534  0.94066  0.91245  5
```

Materialdaten eines ST-15 Stahls, im Datei Format des Programms Matform v.0.0.4

```
Youngs                206000
######     SPCC 0.9mm =645*(0.0093+I.strain)^0.238
Poisson               0.33
SpecificWeight 7.8E-5
Hardening
0          0                      211.8647263
1          0.01                   252.0706923
2          0.02                   278.4031907
3          0.03                   298.555545
4          0.04                   315.1064416
5          0.05                   329.2658816
6          0.06                   341.7072745
7          0.07                   352.8472469
8          0.08                   362.963005
9          0.09                   372.2490823
10         0.1                    380.847679
11         0.12                   396.3876943
12         0.14                   410.1908315
13         0.16                   422.6492189
14         0.18                   434.0318108
15         0.2                    444.5317559
16         0.22                   454.292827
17         0.24                   463.4251689
18         0.26                   472.0151811
19         0.28                   480.1319888
20         0.3                    487.8318321
21         0.4                    521.4655828
22         0.5                    549.3124592
23         0.6                    573.2572224
24         0.7                    594.3704984
25         0.8                    613.323754
26         0.9                    630.5681433
27         1.0                    646.4226119

R-Value               1.09       0.79       1.29
    Failure
        0       -0.20   0.50
        1        0.00   0.275
        2        0.10   0.325
        3        0.20   0.375
        4        0.30   0.42
        5        0.40   0.43
```

Darstellung eines Japanischen Stahls im Autoform Dateiformat

6.2.5 Druckroutinen

Die nächste Herausforderung bestand darin, dem Programm Print-Routinen hinzuzufügen. Da dies noch keiner in dem Institut implementieren musste, betrat ich Neuland. Da solche Methoden schon in der QT-Lib wie auch in der QWT-Lib vorbereitet sind, jetzt sollte man nur noch in der Lage sein irgendwie darauf zuzugreifen. Nach dem Lesern sämtlicher Dokumentation und durchstöbern von diversen Foren. Stellte sich heraus , das ich den grössten Erfolg durch Try & Error erhalte. Und so war es.

6.2.6 Dynamische Messdatenverwaltung

Nun stand auf dem Projektplan, die Implementierung einer Speicherungsmöglichkeit für mehrere eingelesene Dateien. So das dann die Möglichkeit besteht mit Daten aus mehreren Datensätzen zu arbeiten. Dieses Modul bekam dann den Namen VFS (Virtual File System)

Im ersten galt es die Anforderungen aufzustellen. Nach diversen Diskussionen mit meinem Vorgesetzen, kamen wir zu dem Ergebnis, das wir noch nicht in der Lage sind eine komplette Anfoderungsliste zu erstellen. Dies beruht darauf, das durch den wechselnden Bedarf der Kunden auch gewisse Datenstrukturen mitwachsen müssen. So mussten wir bei der Implementierung diverse Routinen einfügen welches ein Problemloses Einfügen und Löschen der Datensätze ermöglich. Setzt man vorraus das beliebig viele Datensätze eingelesen werden können, und sich der Umfang der Datensätze stark variiren kann, so ist man dazu gezwungen, dynamische Arrays zu verwenden. Dies lässt sich in C++ ausschliesslich durch Pointer realisieren.

Es ist dann durch entsprechender Programmierung möglich auf solche Pointer wie auf ein normales Array zuzugreifen. Aber durch kleinere Unachtsamkeiten während des Implementierens, kommen die klassischen Fehler zustande. Als Beispiel möge ich hier den Zugriff auf eine Arrayposition nennen, welche aber nicht zu den Pointern gehört. Die Folge daraus sind nette Fehlermeldungen / Abstürze. Selbst durch die Verwendung von Debugger und Co.

Ist es dann nur mit sehr viel Zeit und Energieaufwand möglich einen solchen Fehler zu finden. In einigen Fällen war es mit Zuhilfenahme des Debuggers nicht möglich den Fehler ausfindig zumachen. So bleibt dann nur die Möglichkeit, den Source auszudrucken, und mit einer Person, welche nicht so nah zu dem Source steht wie ich, es von Hand durchzugehen. Meistens findet sich solch ein Fehler dann recht zügig.

6.2.7 Lizenzschutz

Im weiteren Verlauf des Praxissemesters sollte der Vorhandene Kopierschutz , welcher aus einem Hardlock bestand, zu ersetzen. Der Lizenzschutz mittels eines Hardlocks war relativ ungünstig zu Handhaben, dieser musste ja mit dem entsprechenden Daten programmiert werden. Im Anschluss daran sollte dann dieser dem Kunden zugeschickt werden. Desweiteren sind die Anschaffungskosten , wie aber auch die laufenden Kosten für diesen Hardwarekopierschutz zu hoch. So wurde die Entscheidung getroffen, einen Softwareseitigen Lizenzschutz zu entwickeln. Der Lizenzschutz beruht auf einer Maschinen-Identifikationsnummer, welche von Maschine zu Maschine unterschiedlich ist. Um die Maschinen-ID, mit den Benutzerinformationen verarbeiten zu können, musste ein weiteres Programm entwickelt werden. Dieses besagte Programm, generiert aus den Benutzerdaten und der Maschinen-ID, einen Lizenzschlüssel. Die neue Kombination aus Rechnernummer, Benutzerdaten und Lizenzschlüssel, muss natürlich in diversen Umgebungen fehlerfrei funktionieren. Somit musste neben diversen praktischen Testdurchläufen auch einige Theoretische Worst-Case-Scenarien durchspielt werden.

6.3 Zusammenfassung aller Arbeiten des Praktikums

Hier werden alle getätigten Arbeiten kurz und kommentarlos aufgeführt.

Netzwerk / Hardware

Installation / Einrichtung sowie auf Wartung diverser Windows NT / 2k / XP Rechner als, als Vertretung des Windows-Admin. Da dies aber nur von kürzerster Dauer war, habe ich diese Aufgaben im Praxissemesterbericht nicht näher erörtert.

Programmierung

Erweiterung / Pflege der GUI des Programms Matform

Import / Exportfilter für / von alternativen programme

Implementierung von Printroutinen

Erstellung eines Lizenzschutzes

Erstellung

7. Resümee des Praktikums

Mir persönlich hat das viermonatige Praktikum sehr gut gefallen. Aufgrund der abwechslungsreichen Aufgaben, dem entspannten und persönlichem Arbeitsklimas verging die Zeit wie im Fluge. An diesem Institut ist Gleitzeit vorhanden, allerdings mit einer empfholenen Kernzeit. Durch die lange Anreisezeit von täglich 3,5 Stunden, konnte ich die Gleitzeit nicht wirklich so nutzen, wie es ursprünglich gedacht war.

Durch die Absolvierung des Praktikums, habe ich sehr viel Erfahrung bezüglich Programmierung, Teamarbeit, und Problemlösung gesammelt. Durch den sehr weiten Erstreckunsgbereich der EDV, kann man das Studium als breitgefächerte Grundlagenvermittlung sämtlicher EDV-Sektionen betrachten. Es ist unumgänglich sich ständig weiter zu bilden. Sei es durch intensives Studium von Literatur, Diskussionsforen, ...

Den einzigen Wahrmutstropfen der Softwareentwicklung besteht darin, das man nicht unbedingt am Abend eines Arbeitstages ein Ergebnis sehen kann. So kann es durchaus möglich sein, das nach Tagen / Wochen fleissiger Arbeit, nur wenige Zeilen Sourcecode geschrieben wurde.

Abschliessend kann ich nur zu einem Ergebnis kommen. Obwohl ich bisher dem ersten Praxissemesters gegenüber grundsätzlich abweisend gewesen war, muss ich diese Denkweise mittlerweile revidieren.

8.Glossar

API	application programming interface (API) Schnittstelle mit Befehlen, ggf. Routinen und/oder Macros, die von einem Betriebssystem oder einer Betriebssystemerweiterung (z.B. für die Benutzung eines Netzwerkes) bereitgestellt wird. Anwendungsprogramme können dann diese Schnittstelle benutzen, um das Betriebssystem zur Ausführung der dieser Schnittstelle verborgenen und durch sie bereitgestellten Aktionen zu veranlassen. Bekannte APIs sind die CAPI im ISDN oder die TAPI.
Breakpoint	Spezieller Haltepunkt, um die Software während des Debbugens an einer bestimmten Stelle zu unterbrechen
C	Derzeit gängigste imperative Programmiersprache. C hat den großen Vorteil, maschinennah und damit sehr schnell zu sein, ohne dabei auf höhere Konzepte, z. B. Strukturierbarkeit, zu verzichten. Der Befehlssatz ist auf elementare Befehle eingeschränkt, was der Verarbeitungsgeschwindigkeit zugute kommt. Komplexe Befehle müssen zusammengesetzt werden. Diese Eigenschaft hat dazu geführt, daß ganze Betriebssysteme bis auf einen kleinen Assembler-Kern in C geschrieben werden. U. a. ist das Betriebssystem Unix auf diese Weise entstanden. Ein weiterer Vorteil von C ist die relativ leichte Übertragbarkeit (Portabilität) auf andere Rechner. Nachteilig wird von manchen die erschwerte Lesbarkeit (write only language) der Programm-Listings empfunden. Die Sprache C ist aus der Sprache B und diese wiederum aus der Sprache BCPL hervorgegangen und wurde erstmalig von Dennis Ritchie in den Bell Laboratorien für das Unix-Betriebssystem auf einer DEC PDP-11 implementiert.
C++	Programmiersprache ähnlich wie C, nur objekorientiert .
Compiler	Ein Compiler ist eine Einrichtung, die ein in einer Computerhochsprache, einer problemorientierten Programmiersprache, geschriebenes Programm in den Maschinencode des jeweiligen Prozessors umsetzt, also in ein Objectcode- Programm, das dann direkt von einem Mikroprozessor ausgeführt werden kann.
Debuging	Fehlersuche und -beseitigung, vor allem in Programmen und Datenbeständen.

IDE	Integrated- Developer-Environment Programmierumgebung , welches verschiedene Tools beinhaltet, z.b. Debugger, Compiler, Editor
Library	Sammlung von Routinen, die häufig gebraucht werden und dann aus der Library abgerufen werden können. Man unterscheidet zwischen zwei Arten von Libraries, den residenten, die während des Betriebes jederzeit verfügbar sind, und den transienten, die erst von einem Speichermedium geladen werden müssen.
MFC	MFC heißt im Englischen Microsoft Foundation Classes. Ziel der MFC Bibliothek ist es, mit Power von C++ Windows Anwendungen zu erzeugen. Die Wiederverwertung des Quellcodes ist ein besonderes Anliegen der MFC. Wie Sie vielleicht wissen, liegt ein großer Nachteil von Windows-API darin, daß sie nicht objektorientiert ist. Entwickler, die mit Windows-API arbeiten, müssen jedes Mal für jede Anwendung alles nochmals neu schreiben. Die MFC Bibliothek ruft selbst Funktionen auf, die in Windows-API implementiert worden sind, um Windows Bestandteile, wie Dialogboxen, Gerätekontext, allgemeine GDI-Objekte und andere Standardelemente zu erzeugen. Die MFC Klassen, die diese Art von Objekten erzeugen, bieten eine angenehmere Schnittstelle, da sie die Eigenschaften von C++ ausnutzen. Eine der wichtigen Besonderheiten von MFC ist ihre große Nähe zu Windows-API. Wenn Sie meinen, daß MFC nicht nah genug an Windows-API geht, können Sie für die Erstellung Ihrer Anwendung parallel zu MFC auch direkt Windows-API Funktionen aufrufen.
QT	Library, zur Programmierung von Grafischen Benutzer Schnittstellen unter verscheidenen Systemen.
QWT	Library zur Einfachen Benutzung von Kurvenplots mit zuhiklfenahme der QT-Lib

9. Literaturverzeichniss

The C++ Programming Language. Special Edition.
von Bjarne Stroustrup
Sprache: Englisch
Gebundene Ausgabe - 1019 Seiten - Addison-Wesley Professional
Erscheinungsdatum: 1. Februar 2000
Auflage: Special
ISBN: 0201700735

C Programming Language
von Brian W. Kernighan, David Ritchie
Sprache: Englisch
Broschiert - 272 Seiten - Prentice Hall PTR
Erscheinungsdatum: 1. Mai 1988
Auflage: 2nd
ISBN: 0131103628

C / C++ ge-packt.
von Herbert Schildt
Broschiert - 426 Seiten - mitp
Erscheinungsdatum: Januar 2001
ISBN: 3826606841

9. Webseitenverzeichniss

[I-01] www.ivp.ethz.ch
Internetseite des Institutes IVP

[I-02] www.ethz.ch
Internetseite der Eidgenössischen techischen Hochschule in Zürich

[I-03] www.forminnotech.com
Ein Spin-Off Unternehmen des Institutes

[I-04] msdn.microsoft.com/vstudio
Produktübersicht über MS-Visual Studio

[I-05] http://www.trolltech.com/products/qt
Produkthomepage über QT-Library

[I-06] http://qwt.sourceforge.net/
Homepage und Dokumentation über QWT-Plot Libraries

[I-07] http://www.netlib.org/f2c/
Fortran nach C / C++ Übersetzer

[I-08] http://www.autoform.de/

[I-09] http://www.ifu.ethz.ch/html/Forschung/ExForm.html
Homepage über Exform-Programm

[I-10] http://www.ifu.ethz.ch/html/Forschung/PressForm.html
Homepage über PressForm