



## **Realisierung interaktiver Formeln innerhalb des e-learning Umfeldes thekey**

Semesterarbeit

von

Silvan Schmid

an der

ETH Zürich,

Institut für mechanische Systeme,

Zentrum für Produkt-Entwicklung

Verantwortlicher Professor:

Prof. Dr. Markus Meier

Betreuer:

PD Dr. Ing. habil. Wilfried J. Elspass

Juli 2001

## **Zusammenfassung**

Das Zentrum für Produkt-Entwicklung an der ETH Zürich will die Funktionalität ihres e-learning Umfeldes «thekey to innovation» erhöhen. Es sollen interaktive Formeln implementiert werden. Die Studierenden sollen die Möglichkeit haben, direkt mit den Formeln aus dem Onlinescript selbstständig in Matlab Berechnungen durchzuführen.

Die vorliegende Semesterarbeit beschreibt die Realisierung dieser Implementierung, welcher MathML als Basis dient. Es wird die Wahl von MathML als Notation erläutert. Weiter werden die verschiedenen Möglichkeiten gezeigt, wie sich MathML in einem Webbrowser darstellen lässt. Anhand der gegebenen Anforderungen wird WebEQ, eine Java Programmfamilie, gewählt.

Der Knackpunkt dieser Arbeit bestand darin, MathML für weitere Berechnungen mit Matlab bereitzustellen. Dieses Problem wurde durch ein Perl Übersetzungsprogramm gelöst. Dieses Perl-Script basiert auf einem DOM- und einem XML-Parser- Modul.

Es wird die gesamte Realisierung aufgezeigt, welche auf weitere Hilfsmittel wie PHP, CGI, ActiveX und JavaScript aufbaut. Es wird die Rollenaufteilung und das Zusammenspiel von Client und Server vermittelt.

Diese Semesterarbeit soll auch als Anleitung dienen, damit die Implementierung von interaktiven Formeln durch einen Webadministrator realisiert werden kann. Im Anhang ist ein Manual, welches alle verwendeten Tools erklärt und deren Funktionen und Konfigurationsmöglichkeiten aufzeigt.

## **Aufgabenstellung**

«thekey to innovation» ist eine neuartige Wissens- und Lernumgebung für Produkt-Innovation und -Entwicklung. Den Studierenden werden online Kursunterlagen angeboten. In diesem e-learning Umfeld kommen in den Dokumenten mathematische Ausdrücke vor. Formeln oder mathematische Symbole wurden bisher im Webbrowser durch GIF-Bilder dargestellt. Diese GIF's präsentieren die Formeln gut, enthalten aber keine Formelinformationen und sind daher nicht interaktiv.

thekey will interaktive Formeln implementieren. Es soll die Möglichkeit offeriert werden, direkt mit einer Formel aus dem Webdokument weitere Berechnungen durchzuführen. Durch anklicken einer Formel im thekey-Umfeld soll das Mathematikprogramm Matlab auf dem Client mit dem ausgewählten Ausdruck geöffnet werden. Studierende am Departement Maschinenbau und Verfahrenstechnik lernen im Grundstudium den Umgang mit Mathematikwerkzeugen wie Matlab. Der Studierende soll im Umgang mit der Formel die ganze Funktionalität von Matlab auskosten können. Auf einengende Massnahmen wie starre Parameterübergabe oder Beispiel-Berechnungen soll im Moment verzichtet werden.

Eine Anforderung für die Realisierung besteht darin, dass einerseits für die Darstellung im Browser und andererseits für die weitere Verwendung durch Matlab ein und dieselbe Notation die Basis bildet. Der Inhalt einer Formel sollte nur einmal definiert werden müssen.

Es ist wichtig, dass keine proprietäre Lösung realisiert wird. Die Implementierung muss vom Browsertyp und Betriebssystem unabhängig sein. Es sollen wenn möglich offene und weit verbreitete Standards zur Anwendung kommen. Weiter soll bei der Auswahl dieser Normen darauf geachtet werden, dass ihre Verbreitung und der Gebrauch auch in der Zukunft gewährleistet ist.

# Inhaltsverzeichnis

<b>Zusammenfassung .....</b>	<b>II</b>
<b>Aufgabenstellung .....</b>	<b>III</b>
<b>Inhaltsverzeichnis .....</b>	<b>IV</b>
<b>Tabellenverzeichnis.....</b>	<b>VII</b>
<b>Bildverzeichnis .....</b>	<b>VIII</b>
<b>1 Einleitung .....</b>	<b>1</b>
1.1 Status .....	1
1.2 Problemstellungen .....	2
1.3 Road Map.....	3
<b>2 Notation.....</b>	<b>5</b>
2.1 Anforderungen .....	5
2.2 Standards.....	5
2.2.1 MathML.....	5
2.2.2 OpenMath .....	7
2.2.3 TeX/LaTeX.....	8
2.3 Wahl der Notation .....	8
<b>3 MathML Render-Tool für Webbrowser.....</b>	<b>11</b>
3.1 Anforderungen .....	11
3.2 Tools.....	11
3.2.1 IBM techexplorer.....	11
3.2.2 WebEQ .....	12

3.2.3	Amaya .....	14
3.2.4	IceStorm.....	15
3.2.5	EZMath.....	15
3.3	Wahl des Render-Tools .....	16
<b>4</b>	<b>MathML zu Matlab Translator .....</b>	<b>18</b>
4.1	Konzept.....	18
4.2	Das Perl-Programm (MML2Matlab.pl).....	19
<b>5</b>	<b>Realisierung.....</b>	<b>22</b>
<b>6</b>	<b>Schlussfolgerungen und Ausblick.....</b>	<b>31</b>
6.1	MathML im WWW.....	32
6.2	Ausbau von MML2Matlab.pl .....	32
6.2.1	Technische Verbesserungen.....	33
6.2.2	Funktionale Erweiterungsmöglichkeiten.....	33
6.3	Alternative Konzepte.....	34
6.3.1	Java Applets.....	34
6.3.2	webMathematica .....	35
<b>A</b>	<b>Manual.....</b>	<b>37</b>
A.1	WebEQ.....	37
A.2	Apache Konfiguration für PHP und CGI.....	37
A.3	PHP Scripting, Formel ins Web einfügen.....	39
A.4	MML2Matlab.pl .....	41
A.5	System Struktur.....	42
<b>B</b>	<b>Source Code .....</b>	<b>43</b>

B.1	Formel_handler.inc .....	43
B.2	MML2Matlab.pl .....	45
<b>C</b>	<b>Referenzen.....</b>	<b>58</b>

## **Tabellenverzeichnis**

- 2.1 Ein kleines MathML Beispiel für den Gebrauch von Presentation und Content Markup.
- 2.2 Die drei webtauglichen Mathematiknotationen in der Übersicht.
  
- 3.1 Die MathML-Renderer in der Übersicht.
  
- A.1 Liste der momentan von MML2Matlab.pl unterstützten Operationen.

## **Bildverzeichnis**

- 3.1 Der visuelle WebEQ 2.5 Formel Editor.
- 3.2 W3C's Browser Amaya.
- 3.3 EZMath Formel Editor.
  
- 5.1 Überblick über die vollständige Realisierung.
- 5.2 MathML-Formel in Texteditor von Hand geschrieben.
- 5.3 Formeln in einem Webbrowser, gerendert mit dem WebEQ Java Applet. Onmouseover-Effekt ist erkennbar.
- 5.4 Matlab Terminal mit übersetzter Formel hinter dem Prompt.
- 5.5 Matlab Editor mit geöffnet m-File das die übersetzte Formel enthält.
  
- 6.1 Mohrscher Spannungskreis erklärt mit einem interaktivem Java Applet.
- 6.2 webMathematica Beispiel mit einer vom Mathematica Server generierter Grafik.
  
- A.1 PHP Datei mit eingefügter Formel.
- A.2 Beispiel einer Verzeichnisstruktur auf dem Server.

# **1 Einleitung**

«Mathematik auf dem Web» ist ein aktuelles Thema. Obwohl ein Wissenschaftler (Tim Berners-Lee, CERN) das World Wide Web für den Austausch von Informationen zwischen Forschungsprojekten entwickelte, ist dabei die Implementierung von Formeln und mathematischen Symbolen weitgehend vergessen gegangen. HTML kann mathematische Ausdrücke nicht darstellen. Formeln werden in wissenschaftlichen Webdokumenten durch ASCII-Konstrukte oder als Bilder dargestellt. Bilder (GIF oder JPEG) sind absolut plattformunabhängig und werden in jedem Webbrowser auf jedem Betriebssystem genau gleich dargestellt. Ein Bild einer Formel repräsentiert einen mathematischen Ausdruck gut, die Formelinformation lässt sich aber nicht extrahieren. Eine echte dynamische Verarbeitung ist nicht möglich. So wird auf viele verschiedene Arten versucht, das Problem von dynamischer Mathematik im Web zu umgehen.

## **1.1 Status**

Eine weit verbreitete und mächtige Lösung des Problem sieht so aus, dass alle mathematischen Berechnungen auf einem Server mit einem Mathematik-Kernel bearbeitet werden [15, 16, 20, 46]. Für die resultierenden Formeln werden, für die Ausgabe im Webbrowser, «on the fly» GIF's generiert. Diese Lösung ist dynamisch, erfüllt aber die Anforderungen der Aufgabenstellung nicht. Die eigentliche Formelinformation muss nämlich zusätzlich dem Bild mitgegeben werden, und die Berechnungen finden nicht, wie verlangt, auf dem Client-Rechner statt. Sobald aber Grafiken ausgegeben werden sollen,

so ist diese Art der dynamischen mathematischen Datenverarbeitung im Moment (siehe Kapitel 6.1) noch die einzige Lösung.

Es gibt Softwarehersteller, welche eine proprietäre Gesamtlösung für die Implementierung von Mathematik im Web anbieten [17]. Sie beinhaltet Autorensoftware und die dazu passenden Plug-ins.

Es existieren standardisierte Formelnotationen, welche mit geeigneten Render-Tools ins Web implementiert werden können. Diese Formelsprachen sind viel weniger speicherintensiv als Bilder und lassen Formeln echt dynamisch im Web präsentieren. Es gibt einige wenige Softwarelösungen [43], die mathematische Notationen für die Darstellung in einem Browser verwenden.

Eine kommerzielle Gesamtlösung, welche die Anforderungen an eine Implementierung von interaktiven Formeln für das e-learning Umfeld thekey erfüllen würde, gibt es nicht.

## **1.2 Problemstellungen**

Es muss eine eigene auf thekey zugeschnittene Lösung entwickelt werden. Es wird dabei auf Basisstandards für den Umgang von Mathematik im Web zurückgegriffen. Um diese Standards den Anforderungen entsprechend ins Web einzufügen, müssen Software Tools gefunden oder selbst geschrieben werden.

Nach der Wahl einer geeigneten Notation, mit der sich mathematische Ausdrücke beschreiben lassen, kann die Arbeit grob in zwei Hauptprobleme aufgeteilt werden:

1. Der mathematische Ausdruck muss basierend auf der ausgewählten Notation in einem beliebigen Webbrowser dargestellt werden können. Dafür muss nach geeigneten Tools gesucht werden.

2. Die ausgewählte Formel soll auf der selben Notation basierend für Matlab zur Verfügung gestellt werden. Es wird ein Übersetzungsprogramm benötigt.

Um diese beiden Hauptpunkte zu einer ganzheitlichen Lösung zusammenzufügen, werden weitere Hilfsmittel benötigt.

- Es braucht einen Formeleditor, welcher die Notation generiert, damit diese nicht von Hand geschrieben werden muss.
- Die Notation muss an einem geeigneten Ort gespeichert werden und es muss dynamisch darauf zugegriffen werden können.
- Auf dem Client muss aus dem Browser auf Matlab zugegriffen werden können. Die übersetzte Formel muss dabei in den Matlabterminal transferiert werden.

In den nächsten Kapiteln wird die Erarbeitung einer Lösung diskutiert.

### **1.3 Road Map**

Kapitel 2 befasst sich mit der Auswahl einer geeigneten Notation, mit der mathematische Objekte im Web dargestellt und berechnet werden können. Diese Notation bildet die Basis der ganzen Arbeit. Alle weiteren Schritte hängen von der Wahl dieses Standards ab.

Kapitel 3 diskutiert die Wahl der Software um die in Kapitel 2 gewählte Notation MathML im Web zu präsentieren.

Kapitel 4 diskutiert das Problem, MathML für Matlab lesbar zu machen. Es wird ein Übersetzungsprogramm benötigt.

Kapitel 5 zeigt die vollständige Realisation der Arbeit. Es werden Anhand einer Übersicht alle eingesetzten Hilfsmittel und Softwaretools erklärt.

Kapitel 6 präsentiert Schlussfolgerungen und gibt einen Ausblick in die Zukunft. Im Unterkapitel 6.3 wird, losgelöst von der Aufgabenstellung dieser Arbeit, speziell nach alternativen Konzepten gesucht.

Anhang A dient als Anleitung um die Implementierung von dynamischen Formeln in thekey zu realisieren. Es werden alle Tools und ihre Konfigurationsmöglichkeiten erklärt.

Anhang B präsentiert den Programmcode, der im Rahmen dieser Semesterarbeit geschrieben wurde.

## **2 Notation**

Die Grundlage, um mit mathematischen Formeln im Web zu arbeiten, ist eine geeignete Notation. In diesem Abschnitt werden kurz die Anforderungen an eine solche Sprache erörtert. Es wird eine Auswahl an möglichen Standards gegeben und Anhand ihrer Spezifikationen und Möglichkeiten eine geeignete Wahl getroffen.

### **2.1 Anforderungen**

Die Notation muss die Semantik eines mathematischen Ausdruckes exakt wiedergeben können. Es müssen Implementierungen für das World Wide Web existieren. Der Standard soll zukunftsorientiert sein und eine weite Verbreitung und grosse Akzeptanz gewährleisten. Es muss möglich sein die Notation relativ leicht durch vorhandene Standards nach Matlab-Code übersetzen zu können.

### **2.2 Standards**

Hier werden die geläufigsten Notationen, welche im Web in Gebrauch sind, aufgelistet und kurz beleuchtet.

#### **2.2.1 MathML**

MathML [1] (Mathematical Markup Language) ist ein W3C Standard. Das W3C ist eine Organisation wo in Arbeitsgruppen zusammen mit der Industrie neue Webstandards erarbeitet und veröffentlicht werden. Das Ziel des W3C ist es, das Potential des Web voll und ganz ausnützen zu können.

MathML ermöglicht die Verwendung von mathematischen und wissenschaftlichen Formeln im Web. Weiter dient MathML als plattform-

unabhängige Notation um Formeln zwischen verschiedenen Applikationen zu kommunizieren. MathML soll es ermöglichen, mathematische Ausdrücke im Web bereitzustellen, zu empfangen und zu berechnen wie dies HTML für Text tut. MathML vervollständigt HTML (XHTML in Zukunft), um auch Formeln in einem Webbrowser darstellen zu können.

MathML ist eine XML-Notation [2]. XML ist ein uniformelles Format für strukturierte, baumförmige Dokumente und Daten im Web. Somit besitzt MathML alle Möglichkeiten und Vorzüge die XML so beliebt und erfolgreich machen. MathML lässt sich mit den selben Methoden parsen wie XML.

MathML ist ein junger Standard. Eine weite Verbreitung steht noch aus ist aber unaufhaltsam im Gang. MathML 1.0 wurde das erste mal im April 1998 veröffentlicht. Seither ist MathML 2.0 publiziert worden (Februar 2001) und MathML 3.0 ist bereits in Arbeit. Eine grosse Anzahl von Softwarefirmen, die bei der Entwicklung des Standards mitgeholfen haben, stehen mit Ihren MathML-kompatiblen Produkten am Start. Eine weite Verbreitung von MathML ist sicher.

Die Mathematische Markup Language bietet die Möglichkeit eine Formel in Bezug auf das Layout oder die Semantik zu beschreiben. Es werden dafür zwei verschiedene Tag-Sätze angeboten, die sich ohne weiteres für spezielle Zwecke mischen lassen.

- Es gibt Presentation Markup, welches die Formel in ihrer Struktur beschreibt.
- Um den Inhalt einer Formel zu erfassen steht Content Markup zur Verfügung.

Es ist möglich aus reinem Content Markup Presentation Markup zu generieren [21]. Eine umgekehrte Transformation, Darstellung zur Semantik, ist nicht möglich, ebensowenig wie aus einem Bild die Formelinformation herauszufiltern.

In Tabelle 2.1 soll die Darstellung vom Term  $(a+b)^2$  mit MathML zeigen:

Presentation Markup	Content Markup
<pre> &lt;math&gt;   &lt;mrow&gt;     &lt;msup&gt;       &lt;mfenced&gt;         &lt;mi&gt;a&lt;/mi&gt;         &lt;mo&gt;+&lt;/mo&gt;         &lt;mi&gt;b&lt;/mi&gt;       &lt;/mfenced&gt;       &lt;mn&gt;2&lt;/mn&gt;     &lt;/msup&gt;   &lt;/mrow&gt; &lt;/math&gt; </pre>	<pre> &lt;math&gt;   &lt;apply&gt;     &lt;power/&gt;     &lt;apply&gt;       &lt;plus/&gt;       &lt;ci&gt;a&lt;/ci&gt;       &lt;ci&gt;b&lt;/ci&gt;     &lt;/apply&gt;     &lt;cn&gt;2&lt;/cn&gt;   &lt;/apply&gt; &lt;/math&gt; </pre>

Tabelle 2.1: Ein kleines MathML Beispiel für den Gebrauch von Presentation und Content Markup.

### 2.2.2 OpenMath

OpenMath ist ein XML-Standard um mathematische Ausdrücke in Bezug auf ihre Semantik zu repräsentieren. Alle Aktivitäten werden von der OpenMath Society [3] koordiniert. Die Society arbeitet eng mit Softwareentwicklern zusammen.

OpenMath ermöglicht den Austausch von Formeln zwischen Computerprogrammen und Datenbanken. Die Society arbeitet eng mit Maple zusammen [18, 15, 19]. Es gibt Tools, welche die Kommunikation und Berechnung von OpenMath mit Maple ermöglichen.

Die Notation lässt sich auch im Web verwenden, d.h. in einem Browser darstellen. Es gibt in dieser Hinsicht mit MathML einige Überschneidungen. OpenMath ist mehr auf die Semantik eines mathematischen Ausdrucks spezialisiert, wobei MathML vor allem auf

die dessen Präsentation versiert ist. Die beiden Technologien sind komplementär.

Als Beispiel wird der einfache Term  $(a+b)$  mit OpenMath dargestellt:

```
<OMOBJ>  
  <OMA>  
    <OMS name="plus" cd="arith" />  
    <OMV name="a" />  
    <OMV name="b" />  
  </OMA>  
</OMOBJ>
```

### **2.2.3 TeX/LaTeX**

TeX ist eine Notation die speziell für die Typographie am Computer in den 70ern von Donald Knuth entwickelt wurde. TeX eignet sich vor allem für die Gestaltung von Dokumenten mit mathematischen Inhalten. LaTeX ist ein weiterentwickelter Dialekt von TeX.

Oft wird LaTeX im Web verwendet um Formeln mit LaTeX-Render-Programmen darzustellen. TeX/LaTeX dient allein der Darstellung. Eine Extraktion des Inhalts einer Formel, basierend auf LaTeX, ist nicht möglich und nur heuristisch abzuschätzen.

## **2.3 Wahl der Notation**

MathML wird als Formel Notation gewählt. MathML bietet die Möglichkeit eine Formel in Bezug auf das Aussehen und den Inhalt zu beschreiben. Dies ist wichtig, da aus dem MathML-Code sowohl die Darstellung der Formel im Browser, als auch die Übersetzung nach Matlab-Code abgeleitet werden soll. Um das mathematische Objekt richtig zu rendern, muss man zum Beispiel wissen, ob bei einer Division

ein waagrechter Bruchstrich oder ein Divisionszeichen („/“) verlangt wird. MathML ist von den drei hier vorgestellten Standards der zukunftsträchtigste von allen. Das W3C hat Normen wie XML, HTML, XHTML, CSS, DOM, XSL, etc. veröffentlicht und arbeitet im MathML-Projekt fest mit Softwarefirmen wie Microsoft, Wolfram Research, IBM, Waterloo Maple Inc., Design Science Inc, etc. zusammen. Diese enge Bindung an die Industrie gewährleistet eine weite Verbreitung und Implementierung. MathML wird in Zukunft das «HTML» für mathematische Ausdrücke im Web. MathML ergänzt HTML und wird zu der Formelnotation im World Wide Web schlechthin. In Zukunft sollte MathML von den Standardwebbrowsern ohne spezielle Plug-Ins oder anderer Hilfsmittel dargestellt werden können.

OpenMath ist in erster Linie dafür entwickelt worden um mathematische Objekte zwischen verschiedenen Software Tools auszutauschen. Die Beschreibung ist stark nach der Semantik einer Formel orientiert. Die Präsentation einer Formel wird wenig berücksichtigt. OpenMath ist sicher umfangreicher und geht tiefer als MathML, dessen Möglichkeiten reichen aber für diese Arbeit bei weitem aus. Die hohe Kompatibilität mit Maple stellt ein schweres Argument für OpenMath dar. In MathML liegt aber eindeutig die Zukunft. Viele grosse Softwarehersteller haben die MathML Kompatibilität ihrer Produkte bereits angekündigt. OpenMath verliert das Rennen gegen MathML und wird ausgeschlossen.

Die Tage von LaTeX sind langsam gezählt. HTML und die neuen Textverarbeitungsprogramme laufen diesem alten Standard den Rang ab. Seit MathML von W3C veröffentlicht wurde ist die Domäne der Formeldarstellung von LaTeX nun auch geknackt. TeX/LaTeX liegt zu den anderen beiden Standards ausser Konkurrenz und kommt als Notation, vor allem wegen der fehlenden semantischen Beschreibung, für diese Arbeit nicht in Frage.

In der Tabelle 2.2 ist eine Übersicht über die drei Notationen und ihre Merkmale gezeigt.

	MathML	OpenMath	TeX/LaTeX
Beschreibung der Semantik	+	++	-
Beschreibung der Darstellung	+	-	++
Implementierungen für WWW	+	0	+
Verbreitung jetzt	0	0	+
Verbreitung in der Zukunft	++	0	-
Übersetzung nach Matlab	0	+	-

Tabelle 2.2: Die drei webtauglichen Mathematiknotationen in der Übersicht.

## **3 MathML Render-Tool für Webbrowser**

MathML soll in Zukunft ohne Hilfsmittel in den Webbrowsern dargestellt werden können. MathML ist drei Jahre alt und trotzdem gibt es noch keinen Browser, der mit der XML-Formelnotation zurecht kommt. Obwohl die geläufigen Webbrowser gratis zu haben sind, benutzen zudem viele Websurfer noch immer alte Browserversionen. Auch wenn ein neuer Browser MathML rendern kann, so ist nicht gewährleistet, dass jeder User diesen neusten Browser auch benutzt.

Das W3C gibt auf ihrer Webseite 17 Implementierungen an, welche für dieses Projekt auf ihre Tauglichkeit geprüft werden müssen.

### **3.1 Anforderungen**

Das auszuwählende Tool muss plattform- und browserunabhängig sein. Es soll so einfach wie möglich auf dem Client installiert werden können. Der MathML-Code sollte, ohne viel zusätzlichen Hilfscod, einfach in die HTML-Umgebung einzufügen sein.

### **3.2 Tools**

Hier werden aus den 17 Implementierungen von W3C die wenigen echten MathML-Viewer vorgestellt und besprochen.

#### **3.2.1 IBM techexplorer**

IBM techexplorer [10] ist ein Plug-in für den Netscape Navigator und den Microsoft Internet Explorer. techexplorer ermöglicht die Darstellung von TeX, LaTeX and MathML Dokumenten und das Veröffentlichen von interaktivem wissenschaftlichem Material auf dem Web.

Es existiert die Möglichkeit auf dem Client mit ActiveX auf PowerPoint und Word zuzugreifen. Das Plug-in lässt sich durch eine Skript Schnittstelle mit Java und JavaScript kontrollieren.

Der techexplorer läuft auf allen gängigen Betriebssystemen. Es muss jedoch für jeden User eine neue Lizenz gelöst werden. Das Plug-in hat in der kleinsten Version eine beachtliche Datengrösse von 4.6 MByte, die professionelle Version kommt sogar auf über 11 MByte! Das Plug-in muss auf jedem einzelnen Client installiert werden.

### **3.2.2 WebEQ**

WebEQ [11] ist eine Java Programmfamilie um MathML 1.0 zu verfassen und zu rendern. Das Softwarepaket beinhaltet:

- Ein Math Viewer Applet um MathML und LaTeX in einem Webbrowser darzustellen.
- Einen auf MathML basierten visuellen Formeleditor, der die Formeln Math Viewer Applet gerecht speichern kann.
- Einen Formeleditor als Java Applet, mit welchem man von einem Browser aus Formeln erstellen und manipulieren kann.
- Einen Page Wizard der MathML in bestehenden Websites aufspürt und diese aufbereitet um später mit dem Math Viewer gerendert zu werden.
- Einen Command Line Page Wizard mit derselben Funktionalität wie der Page Wizard. Er besitzt kein Interface und lässt sich aus einem beliebigen Skript aufrufen.

Das wichtigste Tool von WebEQ ist sicher das Math Viewer Applet, welches eine echte dynamische Darstellung von MathML in einem Webbrowser ermöglicht. Der Gebrauch von einem Java Applet ist

plattformunabhängig und wird heute von jedem Browser unterstützt. Das Applet kann durch seine geringe Grösse von 343KB direkt mit der Webseite geladen werden. Dieses Runterladen des Applets geschieht nur beim ersten Aufruf einer Webseite, welche den Math Viewer enthält. Danach wird das Programm aus dem Zwischenspeicher geladen. Es gibt aber auch die Möglichkeit, eine spezielle Math Viewer Installationssoftware zu gebrauchen. Diese installiert das Applet vom Server aus auf dem Client. Danach muss der Viewer nicht mehr jedesmal neu vom Server geladen werden.

Der WebEQ 2.5 Formel Editor kann nur Presentation Markup speichern. Es existiert aber schon eine Betaversion von WebEQ 3.0, welche auch das für diese Arbeit gewünschte Content Markup unterstützt. Diese neue Version wird dann auch vollständig MathML 2.0 unterstützen.

Sehr interessant ist der Command Line Page Wizard, welcher durch seine Fähigkeit MathML-Code «on the fly» für das Math Viewer Applet aufzubereiten, einen echten dynamischen Einsatz von MathML ermöglicht.

WebEQ wurde vor kurzem von Design Science Inc. (MathType) übernommen und wird nun von dieser Firma vertrieben. Die Lizenz für das Math Viewer Applet alleine ist gratis. Es muss nur für das ganze Softwarepaket eine Lizenz gelöst werden.

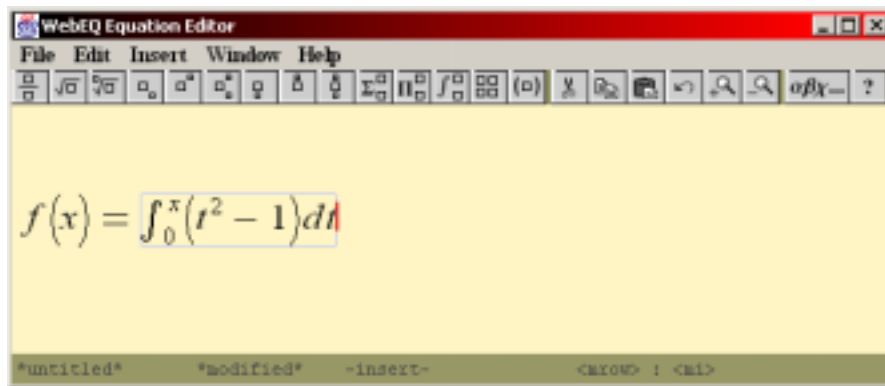


Bild 3.1: Der visuelle WebEQ 2.5 Formel Editor.

### 3.2.3 Amaya

Amaya [12] ist der W3C Test Browser und Editor. Er wird gebraucht um alle neuen Entwicklungen von Webprotokollen und Datenformaten zu demonstrieren und zu testen. Amaya rendert MathML. Der Browser ist für beide Unix und Windows '95/NT Plattformen erhältlich.

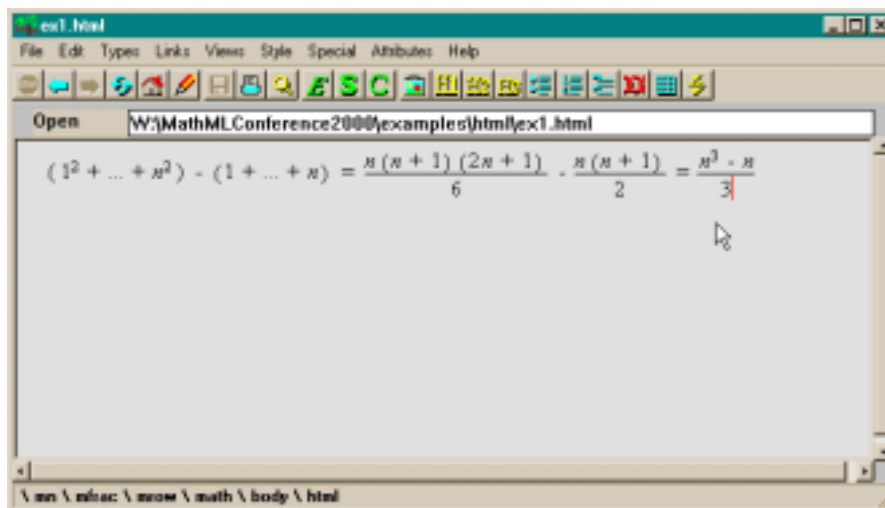


Bild 3.2: W3C's Browser Amaya.

### 3.2.4 IceStorm

IceStorm [13] ist eine Java Webbrowser-Komponente. Der Browser kann in jede beliebige Java Applikation eingebaut werden. Der IceStorm Browser beinhaltet einen MathML Pilot der MathML rendern kann. Der Renderer basiert auf WebEQ.

### 3.2.5 EZMath

EZMath [14] ist ein Tool um mathematische Ausdrücke im Web zu präsentieren. Es beinhaltet neben einem Formeleditor auch ein Browser Plug-in. Neben einer eigenen speziell entwickelten Formelnotation unterstützt EZMath auch MathML. Die Software wurde von Dave Raggett, einem W3C Mitglied, mit der Unterstützung von Hewlett Packard entwickelt.

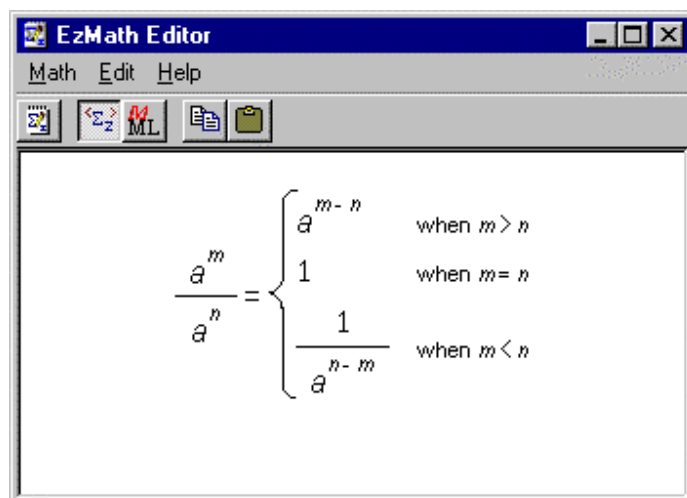


Bild 3.3: EZMath Formel Editor.

### **3.3 Wahl des Render-Tools**

Für die Implementierung von MathML ins Web wird WebEQ gewählt. Das ausschlaggebende Argument, das für WebEQ spricht, ist, dass der Math Viewer, das Herzstück von WebEQ, ein Java Applet ist und auf jedem Browser und jeder Plattform ein gesetzter Standard darstellt. Es muss nicht, wie dies bei Plug-ins nötig ist, für jeden Browsertypen eine separate Version zur Verfügung gestellt werden. Dieses Applet ist zusätzlich, im Vergleich zu den anderen Renderern, sehr klein und muss nicht auf dem Client vorinstalliert werden. WebEQ ist eine MathML Implementierung der ersten Stunde und dient für andere Software, wie zum Beispiel für IceStorm, als Basis. Die im Softwarepaket mitgegebenen Tools decken alles ab um MathML erfolgreich ins Web zu bringen.

Der IBM techexplorer, der schärfste Konkurrent von WebEQ, scheitert an seiner grossen Datenmenge und der Lizenz, die für jeden Client benötigt wird. Ein Plug-in muss, um für eine grosse Anwenderschaft zum Einsatz zu kommen, einfach und schnell auf dem Client installiert werden können. Der techexplorer erfüllt diese Anforderung nicht.

Der Testbrowser von Amaya kommt nicht in Frage da man nicht von jedem User erwarten kann, dass er diesen speziellen Browser benützt. Amaya ist kaum verbreitet und ist als MathML-Renderer für dieses Projekt nicht geeignet.

IceStorm ist keine eigenständige Software die direkt eingesetzt werden könnte. IceStorm ist eine Softwarekomponente, die zuerst in ein Tool implementiert werden müsste. Da IceStorm auf WebEQ basiert, kann IceStorm nicht mehr und nicht weniger als das gewählte Tool WebEQ.

EZMath ist zu experimentell als dass es hier ernsthaft eingesetzt werden könnte. Es sind noch einige Bugs vorhanden die von Raggett behoben werden müssen. Diese Einmannsoftware wird immer den

kommerziellen Tools hinterherlaufen. Es fehlt schlicht die Manpower. Ein tadelloser Support ist nicht gewährleistet.

In der Tabelle 3.1 werden zur Übersicht alle Tools in den wichtigsten Beurteilungskriterien noch einmal miteinander verglichen.

	techexplorer	WebEQ	Amaya	IceStorm	EZMath
Kosten	--	0	++	-	++
Support	+	+	0	+	-
Plattform-unabhängigkeit	+	++	+	+	-
Browser-unabhängigkeit	0	++		0	0
Installation auf Client	-	+	--	0	-
Ganzheitlichkeit	+	++	+	--	0

Tabelle 3.1: Die MathML-Renderer in der Übersicht..

## **4 MathML zu Matlab Translator**

Die gewünschte Formel soll durch Anklicken direkt in Matlab [42] bearbeitbar sein. Dazu muss die gewählte Notation MathML, in welcher die Formel dokumentiert ist, übersetzt werden. Aus dem baumförmigen XML-Gebilde muss ein einzeiliger Matlab-Terminal-Befehl generiert werden.

Es gibt kein Programm das MathML nach Matlab-Code übersetzt. Es muss ein eigenes Programm geschrieben werden oder aber es wird nach einem alternativen Mathematikprogramm gesucht, welches MathML unterstützt. Zum Zeitpunkt der Recherchen für diese Arbeit haben die Softwarehersteller von Mathematica [22] und Maple [18] eine MathML-Unterstützung angekündigt. Da in der Anforderung Matlab gewünscht wird und die Produkte noch keine MathML Implementierung aufweisen, wird speziell für diese Arbeit ein eigener Translator geschrieben.

### **4.1 Konzept**

Das Programm muss aus dem Web aufgerufen werden können. Es muss aber auch seinen Output wieder ins Web bringen können. Es gibt die beiden Möglichkeiten, das Programm auf dem Client oder auf dem Server auszuführen. Um das Programm auf dem Client laufen zu lassen, muss ein Plug-in oder ein JavaApplet programmiert werden. Dieses müsste dann vom Benutzer auf seinen Rechner geladen und ausgeführt werden, was die Performance für den User beeinträchtigt. Den Translator also auf dem Client-Rechner zu haben ist nicht effizient. Viel eleganter ist es, das Übersetzungsprogramm auf dem Server laufen zu lassen. Alle gängigen Webserver haben eine Schnittstelle, welche die Kommunikation zwischen ihm, dem HTTP Webserver, und einem externen Programm ermöglichen. Diese Pforte heisst Common Gateway

Interface (CGI). Somit kann, mit der Hilfe von CGI, vom Browser auf der Client-Seite über den Webserver ein Programm auf der Server-Seite aufgerufen werden. Die populärste CGI-Programmiersprache ist das mächtige Perl. Es kann auch ein C++ Programm aufgerufen werden. Perl ist aber dank seiner besonders guten Verarbeitung von textuellen Daten sehr gut für die Aufgabe eines Translators geeignet.

Perl ist eine Interpreter-Sprache. Das Programm muss vor seiner Ausführung nicht kompiliert werden. Ein auf dem System installierter Perl Interpreter [27] interpretiert bei einem Aufruf den Code und führt in aus. Der Quellcode eines Programms ist immer zugänglich. Perl ist eine frei verfügbare Sprache. Der Perl Interpreter wie auch zusätzliche Module sind kostenlos zu haben. Es gibt eine umfangreiche Modulsammlung [26], wo man sich ein gewünschtes Modul holen und in sein eigenes Programm einfügen kann. Als Arbeitsumgebung dient ein einfacher Texteditor.

#### **4.2 Das Perl-Programm (MML2Matlab.pl)**

MathML ist eine XML-Notation. Um an die strukturierten Daten zu kommen, müssen alle Tags des MathML-Baums mit ihren Informationen eingelesen werden. Es wird ein sogenannter Parser benötigt. Um auf die mit dem Parser eingelesenen Daten zugreifen zu können, wird der MathML-String in ein Document Object Model (DOM) [35] überführt. Das DOM wurde von W3C definiert. Es ermöglicht, im Vergleich zu anderen Zugriffsmodellen wie SAX (The Simple API for XML) [38], einen ungeordneten, zufälligen Zugriff auf die Daten. Diese Fähigkeit benötigt das vollständige Einlesen des XML-Dokuments, was beim Parsen von grossen Dokumenten zeit- und arbeitspeicherintensiv ist. Da die MathML-Strings relativ kurz sind, fallen diese Performance dämpfenden Eigenschaften von DOM nicht ins Gewicht. Es wird für den Perl Translator das populäre XML::DOM Modul [29] eingesetzt. XML::DOM baut eine

DOM Level 1 Dokumenten Struktur auf. Es existiert im Moment leider noch kein Modul das DOM Level 2 unterstützen würde. Das DOM Level 2 wurde unter anderem speziell für den Umgang mit MathML erweitert. Das XML::DOM Modul selber basiert auf dem XML::Parser Modul. Dieses Modul ist eine Schnittstelle zu James Clark's Parser expat [37]. Zuunterst in dieser Modulhierarchie steht also der berühmte, in C geschriebene, XML Parser von James Clark, auf dem die beiden Perl-Module aufbauen.

Mit der Hilfe des DOM wird die baumförmige Struktur von MathML durch einen rekursiven Programmaufbau relativ leicht vollständig durchlaufen und vorweg übersetzt. MathML Content Markup besteht vorwiegend aus dem `<apply>` Konstrukt. In MathML 1.0 wird für Relationen noch das spezielle `<reln>` Tag benutzt, dieses wird bei MathML 2.0 auch durch das `<apply>` Tag ersetzt. Es wird bei jedem Knoten sein Typ untersucht, und dann auf eine spezifische Funktion verwiesen, welche diesen Knoten Typen (apply, reln oder mrow etc.) nach den offiziellen MathML Spezifikationen [40] richtig behandelt und matlabkompatibel übersetzt. Diese Funktion ruft, je nach erreichtem Knoten, sich selber oder eine andere spezifische Funktion auf. Es bildet sich eine Rekursion über mehrere verschiedene Funktionen. Das Resultat ist ein String, der die Formel in der Matlab Notation beschreibt.

Um die Informationen vom Browser (Programmaufruf) zu lesen und den Output des Programm wieder dem Browser zu senden, wird das standard CGI.pm Module verwendet. Dieses Modul übernimmt die ganze Kodierung und Dekodierung der zu sendenden Daten.

In den Daten, welche das Programm nach der Übersetzung dem Browser zurückschickt, sind alle Informationen für den weiteren Ablauf enthalten. Es muss dem Browser, damit dieser die Informationen richtig interpretieren kann, ein HTML-Rahmen (HTML-Header, Body, etc.) geschickt werden. In diesen Daten ist eine JavaScript-Section enthalten, die alle nötigen Manipulationen auf dem Client steuert. Es wird mit der

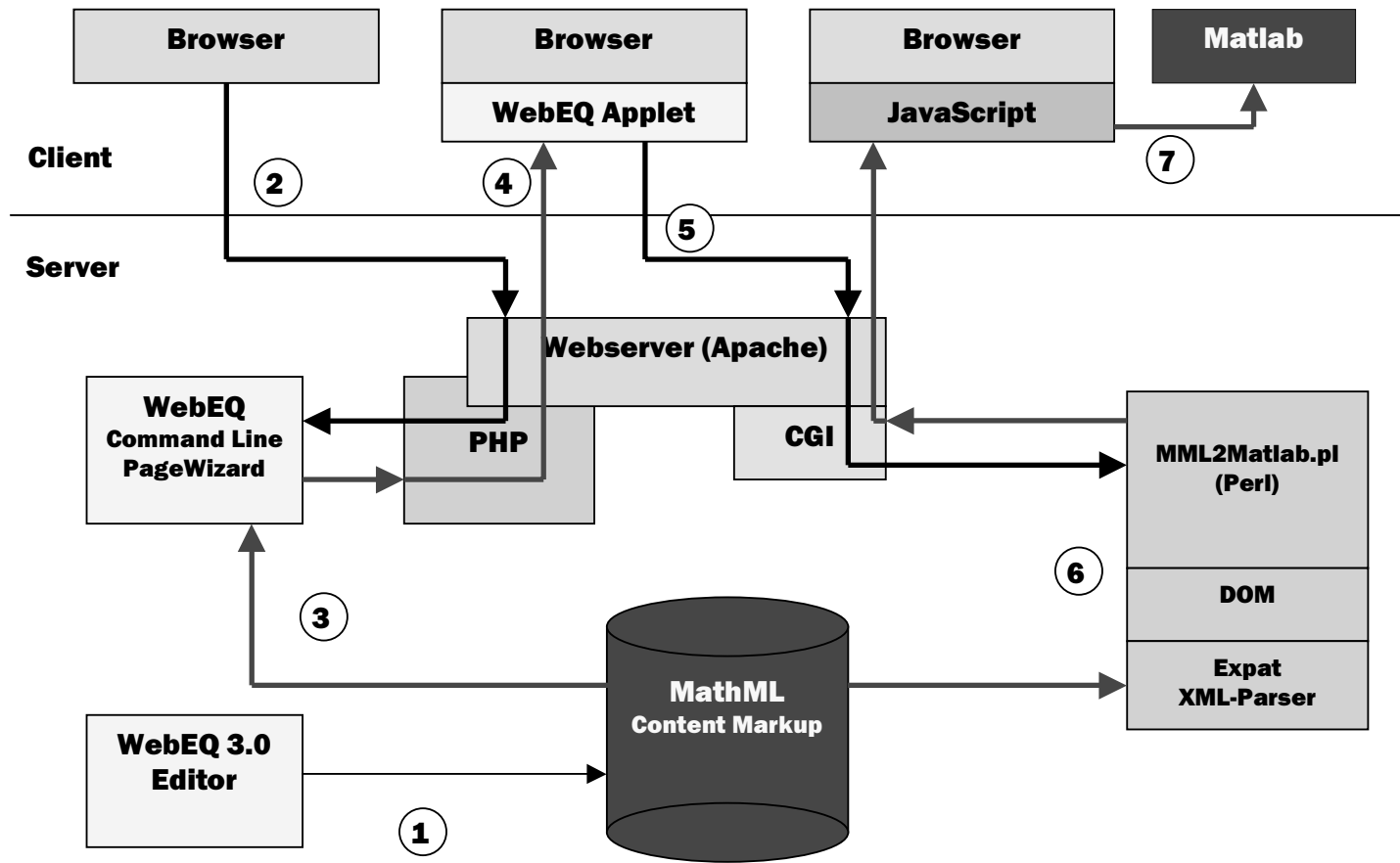
Hilfe von ActiveX das Matlab auf dem Rechner des Benutzer gestartet, und das Resultat der Translation in den Zwischenspeicher gelegt. Da der elegante ActiveX Aufruf von Matlab auf dem Client aus Sicherheitsgründen Probleme hervorrufen kann, ist noch eine alternative Matlab-Client Methode in den Output integriert worden. Es wird dabei der resultierende Matlab-String auf dem Server als m-Matlabfile gespeichert. Das JavaScript referenziert diese Datei und öffnet auf dem Client automatisch den Matlab Editor. So kann man zwischen diesen beiden Methoden wählen.

Für genauere Informationen über das Programm selber, ist ein Manual im Anhang A.4 und der Quellcode im Anhang B.2 abgedruckt.

## **5 Realisierung**

Die Lösung der beiden Hauptprobleme, das Darstellen von MathML in einem Browser und das Übersetzen von MathML nach Matlab-Code, ist in den beiden Kapiteln 3 und 4 besprochen worden. In diesem Kapitel wird anhand eines Übersichtsplans - Bild 5.1 - die ganze Realisierung beleuchtet und die einzelnen Punkte diskutiert. Es werden, um eine ganzheitliche funktionierende Lösung zu erhalten, einige Hilfsmittel und Tricks benötigt.

Bild 5.1: Überblick über die vollständige Realisierung.



1

### *MathML generieren*

Um mit einer Formel arbeiten zu können, muss sie erst einmal geschrieben werden. Dies kann, wie im Bild 5.2 dargestellt, von Hand mit einem einfachen Texteditor geschehen. Das MathML ist mit ein bisschen Übung schnell erlernt und einfach anzuwenden. Das MathML kann aber auch von einem Formeleditor automatisch generiert werden. Der Editor muss Content Markup MathML 1.0 oder 2.0 ausgeben können. WebEQ beinhaltet einen visuellen Formel Editor. Das Content Markup 2.0 wird aber erst in der Version 3.0 unterstützt. Diese Version existiert bereits als Betaversion. Es ist auch denkbar, dass eine neue Version des bekannten Formel Programms „Math Type“ von Design Science, die WebEQ übernommen haben, bald Content Markup produzieren kann. Ob ein Editor benutzt wird, um das MathML zu generieren, oder ob man die Formel von Hand notiert, ist irrelevant. Der Syntax muss der offiziellen MathML Spezifikation [40] genügen. Wegen des Arbeitsaufwandes wird vom Übersetzungsprogramm nicht der gesamte MathML-Tag-Satz unterstützt. Mehr Informationen über die MathML Unterstützung des Perl Translator werden im Anhang A.4 aufgezeigt.

Die MathML Notation wird als .mml-File im ASCII-Format gespeichert. Das File bekommt einen eindeutigen Namen, damit später auf die richtige Formel zugegriffen werden kann. Die Namengebung ist frei wählbar, muss aber vom Publisher richtig im Webdokument referenziert werden. Die MathML-Dateien werden an einem zentralen Ort in einem speziellen Formelverzeichnis, einem sogenannten Formelpool, gespeichert.



anzuweisenden Formel mitgegeben. Nähere Informationen über die Konfiguration und den Einsatz von PHP sind im Anhang A.2 und A.3 zu finden. Im Anhang B.1 ist das PHP-Skript abgedruckt.

### 3

#### *MathML dem WebEQ Java Applet als Parameter übergeben*

Der WebEQ Command Line Page Wizard liest das spezifische MathML-File ein und bereitet es so auf, dass das MathML als WebEQ Formel Viewer Java Applet im Browser angezeigt werden kann. Der Wizard bestimmt die Grösse des Appletfensters, die von der Formel und ihrer Schriftgrösse abhängt. Das MathML wird neben einigen Konfigurationsinformationen als Parameter übergeben. Es wird dem Applet ein Hyperlink hinterlegt, der beim Anklicken der Formel das Perl Übersetzungsprogramm aufruft. Dem Link ist die Formel-ID angehängt, damit der Translator später das richtige MathML-File übersetzt.

### 4

#### *Formel Rendering im Browser*

Jede einzelne Formel wird durch ein Java Applet im Webbrowser dargestellt. Im Applet können onmouseover-Effekte definiert werden. Im Bild 5.3 ist eine solche Implementierung dargestellt. Das Applet wird, falls es nicht auf dem Client vorinstalliert ist, beim ersten Gebrauch geladen. Ist es während einer Session einmal vom Server geholt worden, so kann es für diese Zeit lokal vom Rechner ausgeführt werden, ohne jedesmal neu ein Update vom Server herunterladen zu müssen. Die Dateigrösse des Applets wurde von 343KB auf 274KB gebracht. Das Applet kann aber auch auf dem lokalen Rechner fest installiert werden. Ein downloaden des Math Viewers ist dann nicht mehr nötig. Mehr Details sind im Anhang A.1 zu finden.

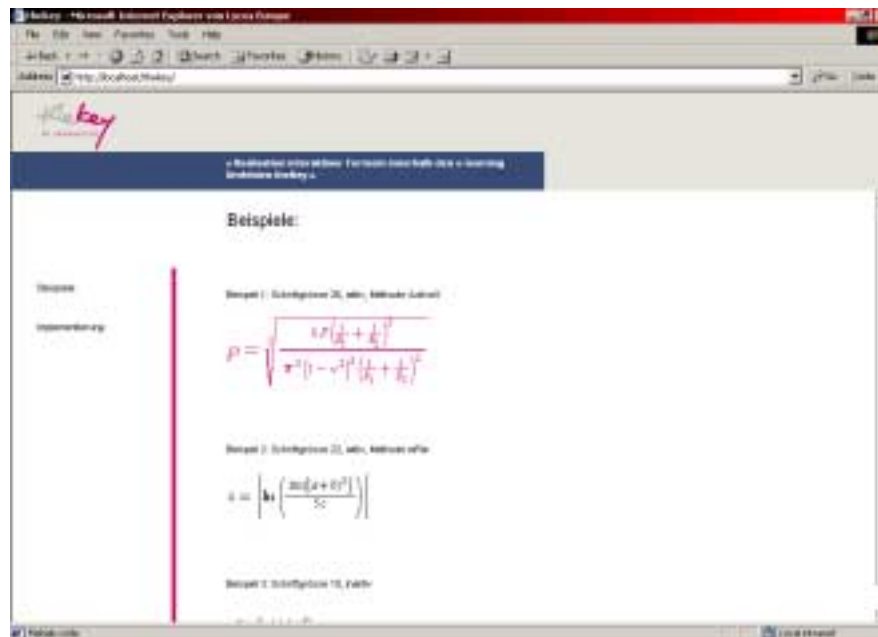


Bild 5.3: Formeln in einem Webbrowser, gerendert mit dem WebEQ Java Applet. Onmouseover-Effekt ist erkennbar.

5

#### *Aufruf des Perl Translators*

Dem Applet ist der Hyperlink, um den Perl Translator aufzurufen, hinterlegt worden. Durch das Anklicken der Formel aus dem Browser wird über das CGI das Perl Programm aufgerufen. Es wird dabei der Name der zu öffnenden MathML-Datei als Parameter mitgegeben.

6

#### *Übersetzung mit dem Perl Programm*

Der Perl Translator MML2Matlab.pl empfängt mit dem Aufruf den Parameter mit der Formelidentität. Es wird, wie es im Kapitel 4 erklärt ist, mit der Hilfe des XML::DOM Modules der gewünschte MathML-String in ein DOM überführt und zu einem Matlab-Terminal-String übersetzt.

Das Resultat wird nun Browserkompatibel an den Client geschickt. Der Output enthält dabei ein JavaScript, um mit ActiveX auf Matlab zugreifen zu können.

7

#### *Öffnen von Matlab auf dem Client*

Vom Browser aus mit dem Client-Rechner zu kommunizieren, ist sehr schwierig. Es ist aus Sicherheitsgründen nicht erlaubt, dass durch einen Webbrowser auf das System zugegriffen werden kann, obwohl solche Vorrichtungen vorhanden wären. Auf dem Windows Betriebssystem gibt es ActiveX. Diese Technik ermöglicht unter anderem das Ausführen von speziellen Kommandos auf einem Client Rechner, auch von einem Browser aus. Das ActiveX ist, da es im alltäglichen Umgang mit dem World Wide Web eine Sicherheitslücke darstellen würde, abgeschaltet. Falls der User einem ActiveX Zertifikat vertraut, so kann er eine Ausführung erlauben. In unserem Fall ist dieses Vertrauen da. ActiveX kann in einer Browserumgebung mit JavaScript realisiert werden. Das JavaScript kann nicht erkennen, wo das Mathematikprogramm auf dem Rechner installiert ist. Um das Matlab auf dem Client erfolgreich öffnen zu können, muss Matlab der System Variable PATH hinzugefügt werden. Dann kann Matlab von jedem Ort des Systems mit dem Aufruf der ausführenden Datei geöffnet werden. Da das Skript vom Perl Translator an den Browser übergeben wird, ist es im Anhang B.2 am Ende des Quelltextes von MML2Matlab.pl zu finden.

Der Matlab-String selber wird mit JavaScript ins Clipboard kopiert. Ist das Matlab Terminal offen, so kann die Formel einfach hinter dem Prompt eingefügt werden. Im Bild 5.4 das Matlab Terminal mit eingefügter Formel zu sehen.

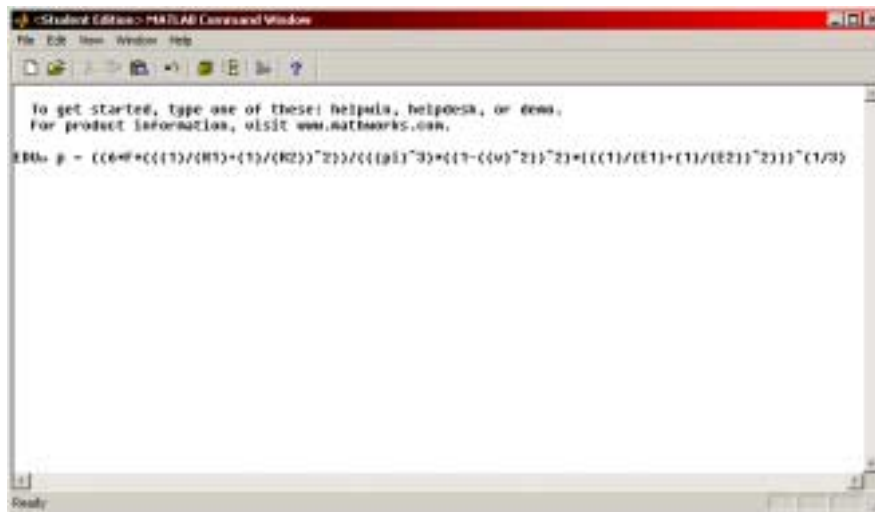


Bild 5.4: Matlab Terminal mit übersetzter Formel hinter dem Prompt.

Da der Gebrauch der ActiveX-Methode zum Aufruf von Matlab mit einigen Schwierigkeiten verbunden ist, gibt es noch eine alternative Methode. Diese Methode arbeitet auf allen möglichen Plattformen. Dabei wird der Matlab-String als Matlab-File auf dem Server gespeichert. Im Browser wird mit JavaScript auf dieses File referenziert. Das System des Client erkennt das m-File als Matlab-Datei und fragt den User, ob das File geöffnet oder gespeichert werden soll. Wird öffnen gewählt, so startet der Matlab Editor. Der vom Perl Translator generierte Matlab-String erscheint dann, wie im Bild 5.5 abgebildet, im Editor. Diese Methode ist nicht so elegant, wie die ActiveX Methode, dafür unproblematischer. Der Nachteil dieser zweiten Methode liegt darin, dass der eigentliche Matlab Kernel beim Öffnen des Matlab Editors nicht gestartet wird. Will man mit der Formel rechnen, so muss zuerst das Matlab Terminal geöffnet werden.

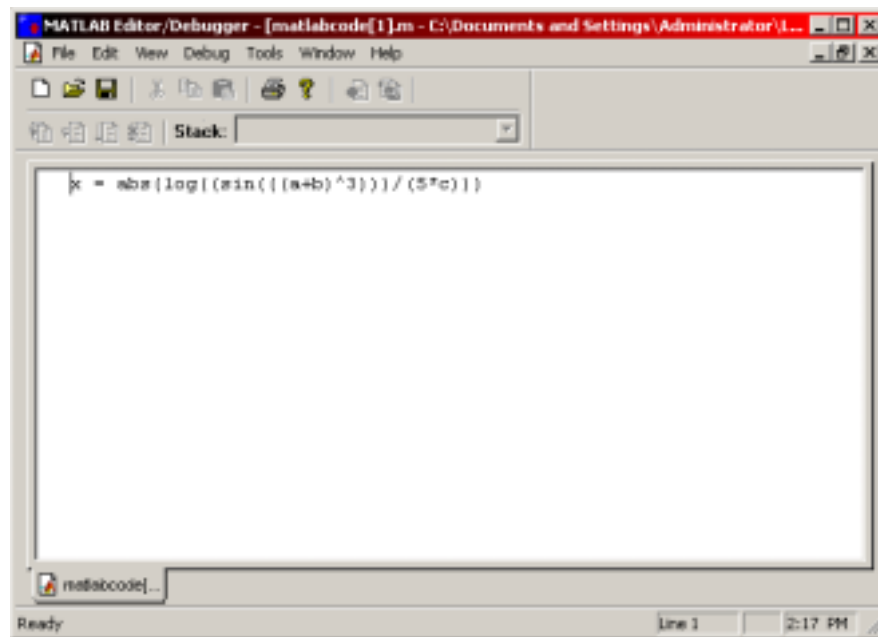


Bild 5.5: Matlab Editor mit geöffnet m-File, das die übersetzte Formel enthält.

## **6 Schlussfolgerungen und Ausblick**

Die Aufgabe konnte erfolgreich gelöst werden. Die Grundanforderungen sind erfüllt. Diese Arbeit steht aber mehr als ausbaufähige Basislösung denn als Endlösung. Die gewählten Standards wie MathML und WebEQ werden auch in Zukunft eine wichtige Rolle spielen. Man darf aber nicht auf dem jetzigen Stand stehen bleiben. Das Thema dynamische mathematische Objekte im WWW zu implementieren ist brisant. Mit der Etablierung von MathML vor rund drei Jahren und der Publikation von MathML 2.0 Anfang dieses Jahres sind viele neue Projekte in diese Richtung gestartet worden. Es sind während dieser Arbeit neue Produkte, Tools und Lösungen entstanden und auf den Markt gekommen, welche zu Beginn, bei den Recherchen zu dieser Arbeit, noch nicht existierten. Dieses Kapitel beinhaltet daher eine speziell wichtige Rolle. Das Thema, Mathematik interaktiv im WWW zu publizieren, hat sich noch längst nicht erschöpft. Die Softwarefirmen von Maple, Mathematica und Mathcad [43] werben für die neuen Versionen ihrer Produkte, die in den letzten Wochen raus gekommen sind, mit ihrer World Wide Web Tauglichkeit und einer vollständigen MathML Unterstützung. Es ist wichtig, dass der Markt im Auge behalten wird. Neue Web-Mathematik-Tools, die für das e-learning Umfeld thekey in Frage kommen, sollten auf ihre Tauglichkeit geprüft und eventuell eingesetzt werden.

In den nächsten Kapiteln wird bezüglich einiger wichtiger Fragen in die Zukunft geschaut. Viele Informationen sind dem halbjährlich erscheinenden «Math on the Web» Status Report [6] entnommen.

## **6.1 MathML im WWW**

Für diese Arbeit ist MathML Version 1.0 gewählt worden, da WebEQ 2.5 nur MathML 1.0 unterstützt. Das Ziel wird sein, für die gesamte Implementierung MathML 2.0 zu verwenden. WebEQ 3.0 wird die neuste MathML Version vollständig unterstützen. WebEQ 3.0 wird bald als Alpha Version erhältlich sein und bietet zusätzlich neue mächtige Scripting Möglichkeiten für den Umgang mit dem DOM.

Für die fernere Zukunft ist MathML 3.0 beim W3C schon in Bearbeitung.

Design Science arbeitet an einer Software Komponente (MathPlayer), die Microsoft's Internet Explorer die Darstellung von MathML ermöglichen soll. Der MathPlayer soll im Herbst 2001 gratis erhältlich sein.

Mozilla wird MathML in der neuesten Version unterstützen. Somit werden die neuen Netscape Browser MathML selber rendern können.

Eine neue, sehr aufregende Art, um dynamische Mathematik im Webbrowser darzustellen, ist der neue vom W3C empfohlene SVG (Scalable Vector Graphics) [44] Standard. SVG ist eine XML-basierte Markup Language die, ähnlich wie PostScript, 2D Grafiken beschreibt. SVG wird durch ein Browser Plug-in gerendert, und kann on the fly in einer Webseite durch JavaScript modifiziert werden. Der neue Standard wird in Zukunft dynamische, mathematische Graphen ermöglichen.

## **6.2 Ausbau von MML2Matlab.pl**

MML2Matlab.pl kann im technischen Bereich verbessert und in seinen Möglichkeiten erweitert werden.

### **6.2.1 Technische Verbesserungen**

Für den Perl Translator MM2Matlab.pl ist ein DOM Modul eingesetzt worden. Dieses Modul unterstützt aber erst DOM Version 1.0. Es sollte, sobald ein Perl Modul für DOM 2.0 erhältlich ist, dieses eingesetzt werden. DOM 2.0 bietet für den Umgang mit MathML einige neue Eigenschaften und Möglichkeiten.

Wegen des riesigen Aufwandes unterstützt MML2Matlab.pl nur einen begrenzten MathML Tag Satz. Es werden erst die wichtigsten Operationen und Symbole erkannt und nach Matlab übersetzt. Das Programm muss in der Zukunft noch weiter ausgebaut werden, damit alle Operationen unterstützt werden.

### **6.2.2 Funktionale Erweiterungsmöglichkeiten**

Das Programm könnte umgeschrieben werden, um weitere Mathematikprogramme, als nur Matlab, zu unterstützen. Während dem Entstehen dieser Arbeit publizierten gerade drei Mathematik-Programmersteller ihre neue Versionen. Maple 7, Mathematica 4.1 und Mathcad 2001 können neu MathML importieren. Es ist nun relativ einfach, auch die Benutzer der anderen Mathematik Tools zu bedienen. Es ist nicht nötig einen neuen Übersetzungsalgorithmus zu programmieren, da die erwähnten Anwendungen pures MathML lesen können. Es muss eine Routine geschrieben werden, die den User nach seinem bevorzugten Mathematik Werkzeug fragt und entsprechend handelt. So können noch mehr User angesprochen werden, da jeder mit dem Tool arbeiten kann, das er am besten beherrscht.

Das Programm könnte die fehlenden Parameter in einer Formel erkennen und diese durch ein Formular vom Benutzer abfragen lassen. Diese Parameter würden dann vom Programm selbstständig in die Formel eingefügt. Der User müsste im Matlab nur noch auf das Resultat warten.

Es kann dem Autor die Möglichkeit gegeben werden, dass er Beispielparameter und zusätzliche Matlab Befehle mitgeben kann. Nach dem das Matlab auf dem Client geöffnet wäre, könnte durch ein Enter eine ganze Reihe von vordefinierten Befehlen ausgeführt werden. Es könnten so zum Beispiel Graphen geplottet werden. Dieses zusätzliche Feature ist mehr für Benutzer, die Matlab Laien sind, gedacht.

Die ganzen Berechnungen eines mathematischen Ausdrucks könnte mit der Hilfe eine Mathematik Moduls direkt mit dem Perl-Programm vollzogen werden. Dieser Schritt geht dann aber in eine Richtung die andere Tools (siehe Kapitel 6.3) schon gehen.

### **6.3 Alternative Konzepte**

In diesem Kapitel werden alternative Lösungen aufgelistet, mit denen Mathematik dynamisch und interaktiv ins World Wide Web gebracht werden kann. Es werden dabei die Anforderungen der Aufgabenstellung bewusst ignoriert.

Es ist zu beachten, dass folgende Lösungen zur Bereitstellung von interaktiven Beispielen, zusätzlich zum Publizieren des Online-Scripts, einen grossen Arbeitsaufwand benötigen.

#### **6.3.1 Java Applets**

Um einzelne komplexe Probleme (z.B. Mohrscher Spannungskreis - siehe Bild 6.2 -, Röscher Diagramm, etc) interaktiv dem Lernenden zu vermitteln, eignen sich Java Applets. Solche Applets erlauben eine komplexe Programmierung im Hintergrund und sie lassen sich in Echtzeit im Browser bedienen. Die Veränderung des Resultats kann beim Modifizieren eines Parameters live mitverfolgt werden.

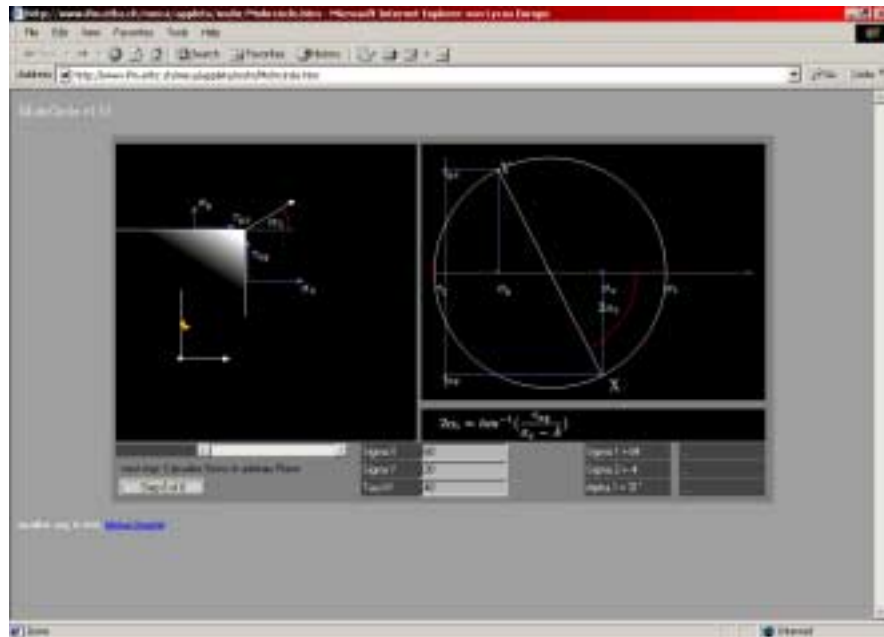


Bild 6.1: Mohrscher Spannungskreis erklärt mit einem interaktiven Java Applet [45].

### 6.3.2 webMathematica

webMathematica [46] ist ein neues von Wolfram Research entwickeltes Tool, mit dem die Möglichkeiten von Mathematica auch im Web genutzt werden können. Es wird dafür ein Mathematica-Server eingesetzt. Dieser besitzt alle Eigenschaften des echten Mathematica Kernels und lässt sich durch ein hinterlegtes Skript steuern. Kommt vom Browser eine Anfrage auf eine Berechnung, so wird ein speziell für jedes Beispiel geschriebenes Skript aufgerufen und ausgeführt. Der Client sendet dem Mathematik Server dabei die benötigten Parameter. Der Mathematica-Server kann die Resultate in Form von MathML, TeX oder einem Mathematica Notebook ausgeben. Visualisierungen sind auch möglich. Dazu werden Bilder generiert, die dem Browser gesendet werden (siehe Bild 6.2).

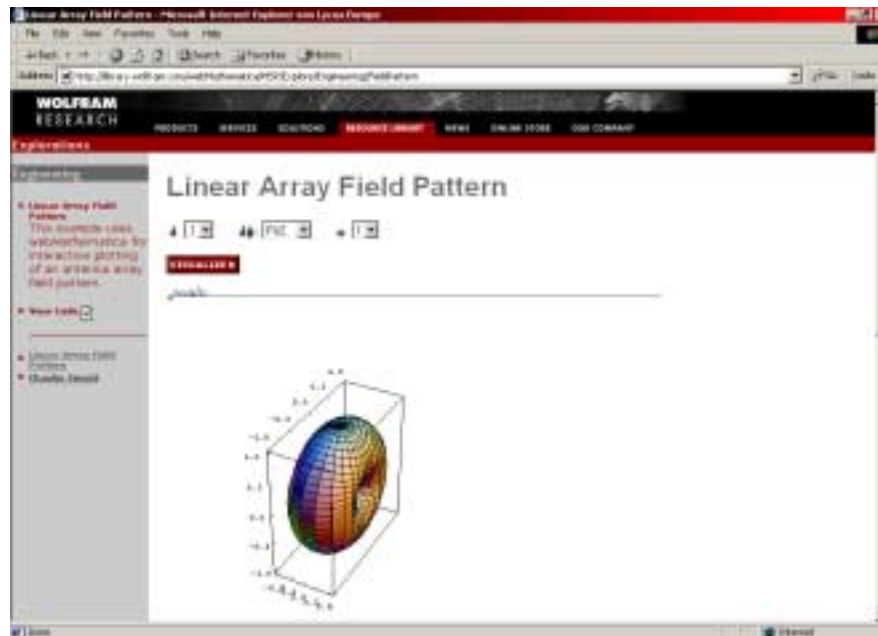


Bild 6.2: webMathematica Beispiel mit einer vom Mathematica Server generierter Grafik.

## **A Manual**

Dieses Kapitel dient dazu, die im Rahmen dieser Semesterarbeit entwickelte Lösung durch Drittpersonen erfolgreich umsetzen zu können.

### **A.1 WebEQ**

Die gesamte Produktfamilie installiert sich beim Ausführen der Setup Datei von allein. Angaben über eine automatische Installation des Math Viewer Applets, findet man im WebEQ Manual.

Das Math Viewer Applet kann im Applet-Header durch den Parameter ARCHIVE referenziert werden. Damit muss der User das Java Programm nicht auf seinem Rechner lokal installieren. Das Applet muss dazu als JAR-File zur Verfügung gestellt werden. Das JAR Format ist eine ZIP-Komprimierung der im Applet enthaltenen verschiedenen Klassen (CLASS-Files). Durch die Dekomprimierung des JAR-Applets, lassen sich nicht benötigte Files löschen. Die Filegrösse kann so von 343KB auf 247KB verringert werden.

Das Applet wird am besten an einem Punkt auf dem Server installiert, auf den von allen Webseiten einfach zugegriffen werden kann.

### **A.2 Apache Konfiguration für PHP und CGI**

Falls Apache noch nicht auf dem Server als Webserver installiert ist, so kann man ihn gratis unter [32] beziehen. Apache ist eigentlich ein Unix Programm, ist aber auch für Windows erhältlich. Wird auf dem Server ein anderer Webserver (IIS) verwendet ist das kein Problem, da alle hier eingesetzten Tools und Standards auch auf andern Webservern und Plattformen problemlos funktionieren.

PHP, wie auch das CGI sind im Apache einfach einzurichten. Das PHP-Paket kann unter [33] gratis heruntergeladen werden. Der PHP Interpreter wird auf Windows selbstständig installiert. Es müssen im httpd.conf File von Apache folgende Zeilen eingefügt werden. Das Konfigurationsfile ist Dank Kommentaren selbsterklärend und schnell richtig ergänzt.

```
<IfModule mod_alias.c>

    ScriptAlias /cgi-bin/ /semesterarbeit/cgi-bin/
    ScriptAlias /php4/ /php/
    <Directory /semesterarbeit/cgi-bin/>
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>

</IfModule>

<IfModule mod_mime.c>

    AddType application/x-httpd-php .php
    AddType application/x-httpd-php .php4
    AddType application/x-httpd-php-source .phps
    AddType application/x-httpd-php4 .php

    Action application/x-httpd-php4 "/php4/php.exe"

    AddHandler cgi-script cgi pl

</IfModule>
```

### **A.3 PHP Scripting, Formel ins Web einfügen**

Hier wird gezeigt, wie eine Formel ins thekey Umfeld eingefügt werden kann.

Nachdem PHP auf dem Webserver installiert ist, müssen aus allen HTML-Webdokumenten PHP-Dokumente gemacht werden.

- Die HTML Files, die Formeln enthalten, müssen neu mit der Endung PHP versehen werden.
- Das PHP File muss die folgende Zeile enthalten:  
`<?php require ('formel_handler.inc'); ?>`  
Dabei ist zu beachten, dass der richtige Pfad zum `formel_handler.inc` angegeben wird.

Soll nun im Dokument eine Formel dargestellt werden, so muss folgende Zeile an die gewünschte Stelle der PHP-Datei eingefügt werden:

```
<?php show_formel('formel_id',16,'activ'); ?>
```

Es sind drei Parameter zu setzen:

- der eindeutige Name der zu publizierenden Formel
- die Schriftgrösse
- der Status, [ activ | inactiv ]

Im Bild A.1 wird eine exemplarische PHP Datei gezeigt



```
formel.php - notepad
File Edit Format Help

<?php require ('formel_handler.inc'); ?>

<html>
<head>
<title>Beispiele</title>
<link rel="stylesheet" href="thekay/thekay.css" type="text/css">
</HEAD>
<body bgcolor="white">
<p class="texttitle"><br>Implementierung:<br><br><br><br></p>
<p class="textnormal">hier soll nun eine Formel in die thekey-Umgebung eingefügt werden.</p>

<p><?php show_formel('presentation',16,'aktiv'); ?></p>

</body>
</html>
```

Bild A.1: PHP Datei mit eingefügter Formel.

Das File `formel_handler.inc` muss richtig Konfiguriert werden.

- `$pfad_webeqapplet="/webeqapplet.jar";`  
Der Pfad zum Math Viewer Applet auf dem Server
- `$formel_pfad = "/formeln/" . $formel_id . ".mml";`  
Pfad zum Formeln Pool
- `$statusline_kommentar = "Oeffnet Matlab mit dieser Formel";`  
Kommentar in der Status Zeile des Browser, erscheint bei onmouseover der Formel
- `$mocolor = "#cc0066";`  
Farbe bei onmouseover
- `$pfad_cgiscript = "/cgi-bin/mml2matlab.pl";`  
Pfad zum Perl Translator

Der Quelltext des Including ist im Anhang B.1 abgedruckt.

#### A.4 MML2Matlab.pl

Der Perl Translator muss richtig konfiguriert werden. Die Konfigurationsarea ist gut ersichtlich und befindet sich in den ersten Zeilen des Programms.

Die Konfiguration ist durch die Kommentare selbsterklärend. Der Quelltext von MML2Matlab.pl ist im Anhang B.2 abgedruckt.

Das Programm unterstützt im momentanen Stadium die in Tabelle A.1 aufgelisteten Operationen. MML2Matlab.pl soll in Zukunft erweitert werden.

Addition, Subtraktion, Division, Multiplikation	+, -, /, *
Betrag	abs
Logarithmen	log, log10
Wurzel, Exponent, Inverse	sqrt, ^(), ^-1
Trigonometrische Funktionen	sin, cos, cot, tan, arcsin, arccos, arccot, arctan
hyperbolische Funktionen	sinh, cosh, coth, tanh, archsinh, archcosh, arccoth, arctanh
Relationen	=, <, >, <=, >=, ~=

Tabelle A.1: Liste der momentan von MML2Matlab.pl unterstützten Operationen.

Um im WebEQ Math Viewer einen horizontalen Bruchstrich zu generieren muss das Presentation Markup-Tag `<mfrac>` verwendet werden. Das Perl Programm erkennt `<mfrac>` und übersetzt dieses richtig zu einer Division („/“).

## A.5 System Struktur

Die Anordnung der Verzeichnisse und Dateien auf dem Server müssen den vorgegebenen Verhältnissen angepasst sein. Es wird hier eine mögliche aber rein exemplarische Struktur wiedergegeben.

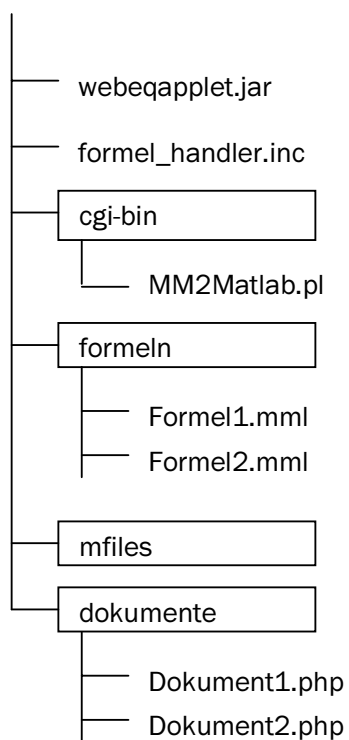


Bild A.2: Beispiel einer Verzeichnisstruktur auf dem Server.

## B Source Code

### B.1 Formel\_handler.inc

```
<?php

function show_formel($formel_id, $schriftgroesse, $activity)
{

    // Configuration #####

    $pfad_webeqapplet = "/webeqapplet.jar";
    $formel_pfad = "/formeln/" . $formel_id . ".mml";
    $statusline_kommentar = "Oeffnet Matlab mit dieser Formel";
    $mocolor = "#cc0066";
    $pfad_cgiscript = "/cgi-bin/mml2matlab.pl";

    // End Configuration #####

    // dem MathML-Code mit dem WebEQ Command-line Page Wizard einen Applet-Mantel verpassen

    $appletcode = `java webeq.wizard.clwizard -delims MathML -parser MathML -size $schriftgroesse
    -bg #ffffff -controls false -outtype Applets $formel_pfad`;

    // action hinzufügen
    // 1. pfad zum perlscript
    // 2. highlightcolor onmouseover
    // 3. statusline report

    if ($activity == 'activ')
    {
```

```
$appletcode = str_replace("<math>","<math><maction actiontype='href' other='" . $pfad_cgiscrypt .  
"?formelnr=" . $formel_id . "'><maction actiontype='fghilicht' other='" . $mocolor . "'><maction action-  
type='statusline' other='" . $statusline_kommentar . "'>",$appletcode);
```

```
$appletcode = str_replace("</math>","</maction></maction></maction></math>",$appletcode);  
}
```

```
# code ins html-env schreiben
```

```
echo $appletcode;
```

```
}  
?>
```

## B.2 MML2Matlab.pl

```
#!/c:/perl/bin/perl"

$| = 1;

#####
# Semesterarbeit SS01 #
# ETH Zürich - Zentrum für Produkt-Entwicklung #
# # #
# Realisation interaktiver Formeln innerhalb des e-learning Umfeldes thekey #
# # #
# Silvan Schmid, sschmid@student.ethz.ch #
# # #
# # #
# Dieses Program parsed ein MathML-File und liest es in ein DOM ein. Das MathML #
# wird rekursiv durchlaufen und dabei nach Matlab-Code übersetzt. #
# # #
# Standard: W3C, MathML 1.01, Content Markup #
# # #
#####

#####
# Konfiguration #####

# Pfad zum Verzeichnis mit den Content-MathML Files
my $formeln_verzeichnis = "./formeln";

# Methode zum Matlab-Aufruf definieren
#
# Die default Methode speichert den generierten Matlab-Code in ein .m file ab und referenziert
```

```

# dieses. Die mfile Methode funktioniert, sofern Matlab auf dem Client installiert ist, immer.
# Es wird aber nur der Matlab-Editor geöffnet.
#
# Die ActiveX-Methode öffnet das Matlab-Terminal, verlangt aber, dass Matlab im System PATH
# eingetragen ist. Der Matlab-Code wird in den Zwischenspeicher gelegt und muss mit "Ctrl + v"
# in den Terminal gepastet werden.

my $methode_matlabaufruf = 1;          # 0 für mfile-Methode
                                       # 1 für ActiveX-Methode

# falls Matlab-Aufruf mit mFile
my $mfile_verzeichnis = "./mfiles";

# Ende Konfiguration #####
#####

# Module laden #####

use CGI;          # Formulardaten Handler
use XML::DOM;     # DOM-Parser Modul, aufgebaut auf XML::Parser und Expat

# Formulardaten einlesen #####

my $query = CGI::new();          # lesen mit Hilfe vom CGI.pm Modul

# formel ID
my $nr = $query->param("formelnr");

```

```

# Formel holen und lesen #####

my $formel_pfad = "$formeln_verzeichnis/$nr.mml";

open (DATA,"< $formel_pfad") || print "Kann die Formel nicht finden.";
while(<DATA>)
{
    s/\s//g;                # Whitespacezeichen weg
    s/\&(,+);/$1/g;        # alle undefined entities für den parser akzeptable machen &pi; -> pi etc.
    chomp;                 # \n's weg
    $mathmlstring .= $_;   # alle strings aneinandereihen
}
close(DATA);

# Main #####

# Parse MathML-String and translate it into a DOM (Document Object Model)

my $parser = new XML::DOM::Parser;
my $mstring = $parser->parse($mathmlstring) or die "unable to parse document";

# Unsere MathML root soll nur ein "Child" haben - einen <math>-Node

my $root = $mstring->getFirstChild;
print "Bad MathML-String! MathML muss einen single Node \"math\" haben.<br>Die Uebersetzung ist
wahrscheinlich fehlerhaft." unless $root->getTagName eq 'math';

# Initialaufruf

```

```
&mrowhandler($root);
```

```
sub mrowhandler #####  
{  
  my $root = $_[0];  
  my $nodes = $root->getChildNodes;  
  my $n = $nodes->getLength;  
  
  for (my $i = 0; $i < $n; $i++)  
  {  
    my $kid = $nodes->item($i);  
  
    SWITCH1:  
    {  
      # mrow ist eigentlich ein Presentation-Markup-Tag welches aber gut mit content-Markup gemischt  
      # werden kann  
      # dient hier eigentlich nur der Blockbildung und führt uns eine Rekursion weiter  
      ($kid->getTagName eq 'mrow') && do { &mrowhandler($kid); last SWITCH1; };  
  
      # apply wie auch die reln (welche im MathML 2 durch apply ersetzt wurden) werden hier gleich behandelt.  
      # reln hat nur drei nodes, sind es mehr, so wird der fehler nicht entdeckt  
      (($kid->getTagName eq 'apply') || ($kid->getTagName eq 'reln')) && do { &applyhandler($kid); last  
      SWITCH1; };  
  
      # mfrac ist ein Presentations-Markup-Tag (Division mit horizontalem Bruchstrich, Quotient)  
      # wurde in den Content-Markup-Tag-Pool aufgenommen, da es sehr oft benutzt wird und nicht darauf  
      # verzichtet werden kann  
      ($kid->getTagName eq 'mfrac') && do { &mfrachandler($kid); last SWITCH1; };  
    }  
  }  
}
```

```

sub applyhandler #####
{
my $root = $_[0];
my $operator = $root->getFirstChild->getTagName;

my $operator_vor = "";
my $operator_mitte = "";
my $operator_zweitletzter = "";
my $operator_nach = "";
my $start_for = 1;

# Operation zu apply abfragen
SWITCH:
{
    ($operator eq 'eq')    && do { $operator_mitte = "="; last SWITCH; };

    # diese operatoren könne mehrmals hintereinander auftreten, es können mehrer geschwister-nodes
    folgen
    ($operator eq 'plus')  && do { $operator_mitte = "+"; last SWITCH; };
    ($operator eq 'minus') && do { $operator_mitte = "-"; last SWITCH; };
    ($operator eq 'divide') && do { $operator_mitte = "/"; last SWITCH; };
    ($operator eq 'times') && do { $operator_mitte = "*"; last SWITCH; };

    # diese operatoren sollten nur von einem geschwister-node gefolgt werden
    ($operator eq 'power') && do { $operator_vor = "("; $operator_zweitletzter = "^"; $operator_nach = ")";
    last SWITCH; };

    ($operator eq 'abs')   && do { $operator_vor = "abs("; $operator_nach = ")"; last SWITCH; };

    ($operator eq 'ln')    && do { $operator_vor = "log("; $operator_nach = ")"; last SWITCH; };
    ($operator eq 'log')   && do { $operator_vor = "log10("; $operator_nach = ")"; last SWITCH; };

    ($operator eq 'root') && do {
        $operator_vor = "(";
        $operator_nach = ")^(1/2)";
    }
}

```

```

    if ($root->getFirstChild->getNextSibling->getTagName eq 'degree') {
        # es wird nur folgender syntax berücksichtigt:
        # <root/>
        # <degree><cn>Nummer</cn></degree>
        $operator_nach = ")^(1/" . $root->getFirstChild->getNextSibling->getFirstChild->getFirstChild->getData . ")";
        $start_for = 2;
    }
    last SWITCH;
};

```

```

($operator eq 'exp') && do { $operator_vor = "exp("; $operator_nach = ")"; last SWITCH; };
($operator eq 'inverse') && do { $operator_vor = "("; $operator_nach = ")^(-1)"; last SWITCH; };

```

```

($operator eq 'sin') && do { $operator_vor = "sin("; $operator_nach = ")"; last SWITCH; };
($operator eq 'cos') && do { $operator_vor = "cos("; $operator_nach = ")"; last SWITCH; };
($operator eq 'cot') && do { $operator_vor = "cot("; $operator_nach = ")"; last SWITCH; };
($operator eq 'tan') && do { $operator_vor = "tan("; $operator_nach = ")"; last SWITCH; };
($operator eq 'sinh') && do { $operator_vor = "sinh("; $operator_nach = ")"; last SWITCH; };
($operator eq 'cosh') && do { $operator_vor = "cosh("; $operator_nach = ")"; last SWITCH; };
($operator eq 'tanh') && do { $operator_vor = "tanh("; $operator_nach = ")"; last SWITCH; };
($operator eq 'coth') && do { $operator_vor = "coth("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arcsin') && do { $operator_vor = "aasin("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arccos') && do { $operator_vor = "acos("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arccot') && do { $operator_vor = "acot("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arctan') && do { $operator_vor = "atan("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arcsinh') && do { $operator_vor = "asinh("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arccosh') && do { $operator_vor = "acosh("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arctanh') && do { $operator_vor = "atanh("; $operator_nach = ")"; last SWITCH; };
($operator eq 'arccoath') && do { $operator_vor = "acoath("; $operator_nach = ")"; last SWITCH; };

```

# Relations (reln), welche aber eigentlich genau gleich wie die apply's angewendet werden (reln bei MathML #2.0 nicht mehr im Gebrauch)

```

($operator eq 'lt') && do { $operator_mitte = "<"; last SWITCH; };
($operator eq 'gt') && do { $operator_mitte = ">"; last SWITCH; };
($operator eq 'leq') && do { $operator_mitte = "<="; last SWITCH; };
($operator eq 'geq') && do { $operator_mitte = ">="; last SWITCH; };
($operator eq 'neq') && do { $operator_mitte = "~="; last SWITCH; };
print "undefined operator";

```

```
}
```

```
my $nodes = $root->getChildNodes;
```

```
my $n = $nodes->getLength;
```

```
for (my $i = $start_for; $i < $n; $i++)
```

```
{
```

```
    my $kid = $nodes->item($i);
```

```
    # wird nur geschrieben, wenn sicher ist, dass es vor dem ersten kind steht
```

```
    # einige operatoren können parameter aufweisen, was eine verschiebung des startpunktes bewirken  
    kann, daher $start_for-1
```

```
    $matlabcode .= $operator_vor if ($kid->getPreviousSibling == $nodes->item($start_for - 1));
```

```
    SWITCH2:
```

```
    {
```

```
        (($kid->getTagName eq 'cn') || ($kid->getTagName eq 'ci')) && do { $matlabcode .=  
            &tokenhandler($kid->getFirstChild); last SWITCH2; };
```

```
        ($kid->getTagName eq 'apply') && do { &applyhandler($kid); last SWITCH2; };
```

```
        ($kid->getTagName eq 'mrow') && do { &mrowhandler($kid); last SWITCH2; };
```

```
        ($kid->getTagName eq 'mfrac') && do { &mfrachandler($kid); last SWITCH2; };
```

```
    }
```

```
    # operator mitte, nur wenn noch ein nächster Sibling existiert
```

```
    $matlabcode .= $operator_mitte if $kid->getNextSibling;
```

```
    # operator zweitletzter, nur wenn nächster Nachbar gleich letzter Nachbar
```

```
    $matlabcode .= $operator_zweitletzter if ($kid->getNextSibling == $kid->getParentNode->getLastChild);
```

```
    # operator nach, nur wenn kein node mehr folgt
```

```

    $matlabcode .= $operator_nach unless $kid->getNextSibling;
}
}

```

```

sub mfracandler #####
{
my $root = $_[0];
my $nodes = $root->getChildNodes;

for (my $i = 0; $i < 2; $i++)
{
    my $kid = $nodes->item($i);

    $matlabcode .= "(";
    if ($kid->getTagName eq 'mrow')
    {
        &mrowhandler($kid);
    }
    elsif ($kid->getTagName eq 'apply')
    {
        &applyhandler($kid);
    }
    elsif (($kid->getTagName eq 'cn') || ($kid->getTagName eq 'ci'))
    {
        $matlabcode .= &tokenhandler($kid->getFirstChild);
    }
    else
    {
    }
    $matlabcode .= ";";
    $matlabcode .= "/" if ($kid == $kid->getParentNode->getFirstChild);
}
}

```

```
}
```

```
}
```

```
sub tokenhandler #####
```

```
{
```

```
my $token = $_[0];
```

```
my $token_content;
```

```
if ($token->getNodeTypes == 3)
```

```
{
```

```
    # falls der token aus einem textnode besteht, so soll dieser direkt ausgegeben werden
```

```
    $token_content = $token->getData;
```

```
}
```

```
else
```

```
{
```

```
    if ($token->getTagName eq 'msub')
```

```
    {
```

```
        # hier wird das tiefgestellte Zeichen dem Parent-zeichen auf gleicher Höhe angehängt
```

```
        my $nodes = $token->getChildNodes;
```

```
        $token_content = $nodes->item(0)->getFirstChild->getData;
```

```
        $token_content .= $nodes->item(1)->getFirstChild->getData;
```

```
    }
```

```
    else
```

```
    {
```

```
        $token_content = $token->getData;
```

```
    }
```

```
}
```

```
SWITCH5:
```

```
{
```

```
    ($token_content eq '&pi;') && do { $token_content_return = "pi"; last SWITCH5; };
```

```
    $token_content_return = $token_content;
```

```
}
```

```
return $token_content_return;
```

```
}
```

```

# HTML und Matlab-Handling #####

print "Content-Type: text/html\n\n";          # header für den browser

# Methode des Matlab-Aufruf abklären
# Default ist die mfile Methode

if ($methode_matlabaufruf == 1)
{

# ActiveX-Methode #####

print <<EOHTML1;
<html>
<head>

<script>
function runProg(command)
{
    if (document.layers && navigator.javaEnabled())
    {
        alert("Please click the 'Grant' button on the next screen to allow " + command + " to run on your
computer");

        window._command = command;
        window.oldOnError = window.onerror;
        window.onerror = function (err)
        {
            if (err.indexOf ("User didn't grant") != -1)
            {
                alert('command execution of ' + window._command + ' disallowed by user.');
```

```

        }
        else return false;
    }
    netscape.security.PrivilegeManager.enablePrivilege('UniversalExecAccess');
    java.lang.Runtime.getRuntime().exec(command);
    window.onerror = window.oldOnError;
}
else if (document.all)
{
    window.oldOnError = window.onerror;
    window._command = command;
    window.onerror = function (err)
    {
        if (err.indexOf('utomation') != -1)
        {
            alert('command execution of ' + window._command + ' disallowed by user.');
```

return true;

```
        }
        else return false;
    }
    var wsh = new ActiveXObject('WScript.Shell');
    if (wsh)
    wsh.Run(command);
    window.onerror = window.oldOnError;
}
history.back();
}

function copytoclipboard()
{
    var where=document.formname.matlabcodeforclipboard;
    var copytext=where.createTextRange();
    copytext.execCommand('Copy');
}

</script>

</head>

```

```

<body onload="copytoclipboard(); runProg('matlab.exe');">
<form name="formname">
<input type="hidden" name="matlabcodeforclipboard" value="$matlabcode">
</form>
Das Matlab wird automatisch geöffnet, sofern Sie Matlab in Ihrem System PATH eingetragen haben.<br>
Weiter müssen Sie die Sicherheitseinstellungen Ihres Browser soweit minimieren, dass Sie ActiveX-Programm-
Aufrufe erlauben.<br>
<br>
Der Matlab-Code ist ins Clipboard gespeichert worden. Sie können Sie mit einem einfachen "Ctrl + v" oder mit Edit -
Paste ins Matlab-Terminal einfügen.<br>
<br>
<a href="#" onclick="history.back()">Hier gehts zurück...</a>
</body>

</html>

```

EOHTML1

```

}
else
{

```

# mfile-Methode

#####

# Matlab-Code in ein file speichern

```

open (DATA2, "> $mfile_verzeichnis/matlabcode.m") or print "Kann outputfile nicht öffnen!";
print DATA2 $matlabcode;
close (DATA2);

```

# HTML

```
print <<EOHTML;
<html>
<head>
<title>Matlab-Code</title>
    <script language=javascript>
    function openmatlab()
    {
    location.href="$mfile_verzeichnis/matlabcode.m";
    }
    </script>
</head>
<body bgcolor=white onLoad="openmatlab()">
Falls Sie Matlab auf Ihrem System installiert haben, wird das generierte <a
href="$mfile_verzeichnis/matlabcode.m">Matlab-File</a> automatisch geöffnet.<br>
<br>
<a href="" onclick="history.back()">Hier gehts zurück</a>.
</body>
</html>
EOHTML
}
```

## **C Referenzen**

- [1] World Wide Web Consortium, Mathematical Markup Language Home Page, <http://www.w3c.org/Math>
  
- [2] World Wide Web Consortium, Extensible Markup Language Home Page, <http://www.w3c.org/XML>
  
- [3] The OpenMath Society Home Page, <http://www.openmath.org>
  
- [4] Ha Quang Le, "Client-Server Communication Standards for Mathematical Computation", A thesis presented to the University of Waterloo, 1999
  
- [5] Ha Le, "Communication-Oriented Representation of Mathematical Objects", University of Waterloo, 2000
  
- [6] Robert Miner, Paul Topping, Design Science Focus: Standards-based Math on the Web, January 2001  
Focus: Distance Learning, Juli 2001  
<http://www.dessci.com/webmath/status/>
  
- [7] Richard Fateman, "More Versatile Scientific Documents", Univ. of California, Berkley, ICDAR 97
  
- [8] MathML International Conference 2000 Home Page, <http://www.mathmlconference.org/>
  
- [9] TeX Users Group Home Page, <http://www.tug.org>

- [10] IBM techexplorer Hypermedia Browser Product Page,  
<http://www.software.ibm.com/network/techexplorer>
  
- [11] WebEQ Home Page, <http://www.dessci.com/webmath/>
  
- [12] Amaya W3C's Editor/Browser Home Page,  
<http://www.w3c.org/Amaya/>
  
- [13] IceSoft Browser Home Page,  
[http://www.windriver.com/products/html/icebrowser\\_ds.html](http://www.windriver.com/products/html/icebrowser_ds.html)
  
- [14] EZMath Home Page,  
<http://www.w3.org/People/Raggett/EzMath/>
  
- [15] Maple Server OpenMath MathML VRML,  
<http://daisy.uwaterloo.ca/~hqle/MS/MS.html>
  
- [16] Martin Kraus, LiveGraphics3D Homepage,  
<http://wwwvis.informatik.uni-stuttgart.de/~kraus/LiveGraphics3D/index.html>
  
- [17] LiveMath, Webified Computer Algebra and Graphing,  
<http://www.livemath.com>
  
- [18] Waterloo Maple Homepage, <http://www.maplesoft.com>
  
- [19] The Organic Mathematics Home Page,  
<http://www.cecm.sfu.ca/projects/OMP/>
  
- [20] Christian Gut, Hypergeometric functions on the web - a learning program, diploma thesis in computer science, 1999, ETH Zürich

- [21] MathML Content2Presentation Transformation (MathMLc2p)  
Home Page,  
<http://www.inrialpes.fr/opera/people/Emmanuel.Pietriga/mathmlc2p.htm>
  
- [22] Wolfram Research Inc., Mathematica, Home Page,  
<http://www.wolfram.com>
  
- [23] Farid Hajji, Perl: Einführung Anwendungen Referenz, 2.Auflage,  
2000 Addison-Wesley
  
- [24] Thomas Theis, PHP4 Galileo Computing, Webserver-  
Programmierung für Einsteiger, Galileo Press GmbH, Bonn 2000
  
- [25] Ellen Siever, Stephen Spainhour & Nathan Patwardhan, Perl in a  
Nutshell, O'Reilly, deutsche Ausgabe 1. Auflage 2000
  
- [26] Comprehensive Perl Archive Network, <http://cpan.perl.org>
  
- [27] ActiveState Home Page, <http://www.activestate.com/>
  
- [28] W3Schools Online Web Tutorials, <http://www.w3schools.com/>
  
- [29] PerlMonth, <http://www.perlmonth.com>  
Eric Bohlman, Parsing XML, Part 1, PerlMonth – Issue #4  
Eric Bohlman, Parsing XML, Part 2, PerlMonth – Issue #5  
Eric Bohlman, The DOMinant Technique for Parsing XML,  
PerlMonth- Issue #6

- [30] Clark Cooper, Using The Perl XML::Parser Module, XML.com,  
<http://www.xml.com/pub/a/98/09/xml-perl.html>
  
- [31] Ingo Macherius, Michel Rodriguez, Ways to Rome: Processing XML with Perl, Version 2.0, 2000
  
- [32] The Apache Software Foundation, Home Page,  
<http://www.apache.org>
  
- [33] PHP: Hypertext Preprocessor, Home Page, <http://www.php.net>
  
- [34] libxml-eno Home Page, <http://jonas.liljegren.org/perl/libxml/>
  
- [35] Document Object Model (DOM), Home Page,  
<http://www.w3.org/DOM/>
  
- [36] Jonathan Robie, Texcel Research, What is the Document Object Model?, September 2000, <http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/introduction.html>
  
- [37] James Clark's Home Page, expat - XML Parser Toolkit  
<http://www.jclark.com/xml/expat.html>
  
- [38] Megginson Technologies, The Simple API for XML,  
<http://www.megginson.com/SAX/index.html>
  
- [39] David Hunter, Beginning XML, 2000 Wrox Press Ltd.
  
- [40] Mathematical Markup Language (MathML), Spezifikationen,  
<http://www.w3.org/TR/MathML2/>
  
- [41] WebCT Home Page, <http://www.webct.com>

- [42] The MathWorks – Matlab Home Page,  
<http://www.mathworks.com/products/matlab/>
- [43] Mathcad Home Page, <http://www.mathcad.com>
- [44] W3C Scalable Vector Graphics (SVG),  
<http://www.w3.org/Graphics/SVG/Overview.htm8>
- [45] IMES ETH Zürich, Center of Mechanics, MECA und andere Ap-  
plets, <http://www.ifm.ethz.ch/meca/>
- [46] Wolfram Research, webMathematica, Product Page,  
<http://www.wolfram.com/products/webmathematica/>