Universität Wuppertal

Studiengang Informations-Technologie

Bachelor Thesis

Automatische Dokumentation einer umfangreichen Software-Klassen-Bibliothek

abgegeben von

Mike Hinz
Matrikel-Nr. 428326

Datum der Abgabe: 4. Juni 2007

Betreuer: Prof. Dr. W. Krämer, Dr. W. Hofschuster

BERGISCHE UNIVERSITÄT
WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
Telefon (0202) 439-3060

FACHBEREICH C
MATHEMATIK

Prof. Dr. Walter Krämer

Bachelor Thesis

KANDIDAT:	Mike Hinz
MATRIKELNUMMER:	428326
BACHELOR-STUDIENGANG:	Informationstechnologie
STUDIENRICHTUNG:	Information Science
BETREUER:	Prof. Dr. W. Krämer / Dr. W. Hofschuster

THEMA: Automatische Dokumentation einer umfangreichen Software-Klassenbibliothek

AUFGABENSTELLUNG:

Inhalt dieser Bachelor-Thesis ist die Evaluierung und Realisierung einer automatischen API-Dokumentation für eine umfangreiche Software-Klassenbibliothek. Konkret wird dies am Beispiel der Klassenbibliothek C-XSC durchgeführt. Es sollen mögliche Varianten für eine automatisch erstellte Dokumentation analysiert und bewertet werden. Anhand des Resultates dieser Analyse wird die Vorgehensweise am konkreten Beispiel C-XSC erläutert werden.

Wuppertal, 01.03.2007 — Prof. Dr. W. Krämer

ERSTGUTACHTER:	Prof. Dr. W. Krämer
ZWEITGUTACHTER:	Dr. W. Hofschuster

Prüfungsamt	Kennziffer:	07 [THI] 5, 7 Krä
	Ausgabedatum:	06. 03. 2007
	Abgabedatum und Signum:	06. 06. 2007

Zusammenfassung

Mit dieser Ausarbeitung wird beschrieben, wie eine bestehende umfangreiche Klassen-Bibliothek, der eine aktuelle Dokumentation fehlt, so dokumentiert werden kann, dass Dokumentation zukünftig kaum Mehrarbeit für die Entwickler der Bibliothek bedeutet und trotzdem beständig aktuell gehalten werden kann.

Da die manuelle Erstellung einer Dokumentation ein sehr zeit- und kostenaufwendiger Prozess sein kann, wird ein Ansatz verfolgt, der die automatische Generierung der Dokumentation zum Ziel hat. Bei diesem Ansatz wird die Dokumentation auch näher an den Entwickler der Bibliothek gebracht, weil die Dokumentation aus zusätzlich im Quelltext angebrachten Kommentaren besteht.

Da der zeitliche Rahmen dieser Thesis beschränkt ist, wurde einerseits der Umfang der Dokumentation beschränkt und andererseits wurden nicht alle Gestaltungsmöglichkeiten genutzt und demonstriert.

- Die Umfangsbeschränkung ergab sich durch die Zielgruppe der zu erstellenden Dokumentation (Nutzer der Bibliothek C-XSC, ohne das Ziel der Weiterentwicklung der Bibliothek). So war die Dokumentation von privaten Methoden, oder Methoden, die ein Nutzer nicht verwendet beziehungsweise nicht verwenden soll, nicht nötig. Entwickler, die die Bibliothek ausbauen möchten, benötigen diese Informationen allerdings, um die internen Zusammenhänge verstehen und nutzen zu können.
- Bei den Gestaltungsmöglichkeiten wurde in der Regel mit den vorgegebenen Standard-Einstellungen gearbeitet. Besonders betrifft das beispielsweise die Gestaltung des Layouts der Dokumentation. Bei näherer Beschäftigung mit diesem Themenkomplex könnte leicht eine separate Ausarbeitung dazu erstellt werden, die das Für und Wider bestimmter Gestaltungsmöglichkeiten gegeneinander abwägt.

Bedeutendster Vertreter dieser Art der Dokumentation ist die Programmiersprache Java und deren Dokumentations-Werkzeug JavaDoc. Da hier aber im konkreten Fall eine C++ Bibliothek dokumentiert werden soll, wird zu Beginn eine Auswahl unter anderen frei verfügbaren Anwendungen getroffen, die die Programmiersprache C++ beherrschen.

Anhand des dabei ausgewählten Werkzeugs und der Klassen-Bibliothek C-XSC wird schließlich schrittweise die Herangehensweise erläutert und verdeutlicht („Wo fange ich an?“, „Wie teile ich die Arbeit ein?“, „Was muss enthalten sein?“ etc.). Dazu werden aufgetretene Probleme aufgeführt und Lösungen oder Lösungsansätze genannt.

Abstract

The aim of this thesis is to describe, how to document a huge class-library, which has actually no documentation, so that in the future the documentation has no big impact on the developers and can be actualized fast and quickly.

The manual creation of a documentation is a time- and cost-consuming process. Here we follow the approach to generate the documentation automatically. By doing it this way, documenting the library is approaching the developers, because they just have to insert additional comments in the source-code.

Because of the timing of this thesis, the size and complexity of the documentation has been limited and not all possibilities for the arrangement of the documentation have been used and demonstrated.

- The limitation of size and complexity is explained by the targeting group of the documentation (normal users of the C-XSC library, without the aim to further improve the library itself). So the documentation of private methods or methods the user does not or shall not use is not necessary. Developers, who want to improve the library, need this kind of information, to understand and use the internal connections of the library.
- For the arrangement of the documentation in most parts the standard settings have been used. This concerns especially the layout of the documentation. With more time it would have been possible to write another draft just figuring out what, why and in which manner all possible options could have been used.

The most well known representation of this kind of documentation is the programming language Java and the included tool JavaDoc. Because of the use of a C++ class library, another one has to be selected from other freely available tools, which is able to handle the programming language C++.

On the basis of the selected tool and the class-library C-XSC the procedures are gradually explained and demonstrated (e.g. “Where do I start?”, “How to handle the work?”, “What has to be included?”, etc.). Therefore the appearing problems are documented and solutions or solutions approaches are named.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und Zitate kenntlich gemacht habe.

Engelskirchen, den 4. Juni 2007

Inhaltsverzeichnis

Danksagung

Mein Interesse an Technik wurde schon in frühen Jahren geweckt. Seither habe ich meinem Lebenslauf stets in diese Richtung gelenkt und meine privaten Interessen zu einem großen Teil mit meinen beruflichen Interessen kombiniert. Mit dem Angebot, mir ein Studium zu ermöglichen, gab mir mein Ausbildungsbetrieb nach meiner erfolgreichen Ausbildung, eine schon fast aus der Perspektive verlorene Möglichkeit wieder, dieses Ziel weiter zu verfolgen.

Mit dem Abschluss meines Studiums rückt nun erneut ein großer Meilenstein in meine Reichweite, wie es schon bei meiner Abschlussprüfung als Fachinformatiker der Fall war. Ich hoffe, dass ich mit dem Passieren dieses Meilensteins eine weitere Hürde auf dem Weg in die Zukunft nehmen werde.

Diese Bachelor-Thesis ist daher meiner Familie, meinen Freunden und meinen Kollegen gewidmet, dich mich während meines Studiums zahlreich und tatkräftig unterstützt haben. Ohne diese Unterstützung wäre ich nicht so weit gekommen!

Mike Hinz

1 Einleitung

Das Ziel dieser Bachelor-Thesis ist die Erläuterung der Vorgehensweise eine umfangreiche Klassen-Bibliothek mit einer Dokumentations-Anwendung automatisch aus dem Quelltext der Klassen-Bibliothek dokumentieren zu lassen. Es soll also eine vollständige Dokumentation einer Klassen-Bibliothek automatisch erstellt werden.

Am Beispiel eines Programmes, das unter Zuhilfenahme der Klassenbibliothek C-XSC das exakte Ergebnis einer Division in Intervall-Schreibweise darstellt, soll gezeigt werden, warum eine Dokumentation einer Klassen-Bibliothek notwendig ist.

```
#include "interval.hpp"  // predefined interval arithmetic
#include <iostream>
using namespace cxsc;
using namespace std;

int main()
{
  interval a, b;            // Standard intervals
  a = 1.0;                  // a   = [1.0,1.0]
  b = 3.0;                  // b   = [3.0,3.0]
  cout << "a/b = " << a/b << endl;
  return 0;
}
```

Hier wird ein neuer Datentyp und überladenene Standard-Methoden verwendet, die C++ von Haus aus zunächst nicht beherrscht. All diese Funktionalität stammt aus der Klassen-Bibliothek C-XSC.

Ohne Dokumentation kann ein geübter Entwickler dieses einfache Beispiel noch verstehen. Bei einem komplexeren Programm ist dies ohne hinreichende Vorkenntnisse auch einem geübten Entwickler nicht mehr möglich:

```
static real MaxNorm ( const rmatrix& M )
{
  int          i, j;
  real         Max, Tmp;
  dotprecision Accu;

  Max = 0.0;
  for (i = Lb(M,1); i <= Ub(M,1); i++) {
    Accu = 0.0;
    for (j = Lb(M,2); j <= Ub(M,2); j++)
      Accu += abs(M[i][j]);
    Tmp = rnd(Accu,RND_UP);
    if (Tmp > Max) Max = Tmp;
  }
  return Max;
} // MaxNorm
```

Um einen solchen Programmcode zu verstehen, muss einem Entwickler deshalb eine Dokumentation der Klassen-Bibliothek zur Verfügung stehen, die ihm ermöglicht den Quelltext und die genutzten Klassen, Methoden und Funktionen zu analysieren und zu verstehen.

Im konkreten hier bearbeiteten Fall wird also eine Dokumentation der Klassen-Bibliothek C-XSC, der beigefügten Toolbox und der Beispielprogramme, die die Verwendung der Bibliothek demonstrieren, für Anwender der Bibliothek erstellt werden. Ausdrücklich wird zunächst die Dokumentation nur für Anwender der Bibliothek und nicht für Programmierer, die die Bibliothek erweitern wollen, erstellt, um den geplanten Umfang dieser Bachelor-Thesis nicht zu überschreiten.

Hierzu werden zunächst im ersten Abschnitt der Ausarbeitung mögliche Software-Kandidaten für die Dokumentationserstellung ausgewählt und deren Funktionen und Beschränkungen miteinander verglichen. Aus diesem Vergleich wird die für diesen Zweck und für den in dieser Arbeit konkret bearbeiteten Fall (die Klassen-Bibliothek C-XSC) am besten geeignete Anwendung ermittelt.

Im Anschluss wird die Herangehensweise an die Dokumentation beschrieben. Wo ist der beste Ausgangspunkt für eine Dokumentation? Welche Regeln sollten beachtet werden? Wie soll oder muss der Quelltext dokumentiert werden? Müssen weitere Dokumente erzeugt werden?

Im Abschnitt Zusammenfassung und Ausblick wird das Ergebnis noch einmal kurz und prägnant zusammengefasst und ein Ausblick für weitere Schritte gegeben.

2 Vergleich der Dokumentations-Werkzeuge

Diese Ausarbeitung soll einen Weg beschreiben, wie die Dokumentation für eine umfangreiche Klassen-Bibliothek mithilfe eines Zusatzprogrammes elegant automatisch erstellt werden kann. Um dieses Ziel zu erreichen, muss zuvor unter der Vielzahl der zu diesem Zweck zur Verfügung stehenden Software eine Auswahl getroffen werden, um anschließend eine begründete Entscheidung für das im konkreten Fall am besten geeignete Programm zu treffen.

Bei der Auswahl des zu verwendenden Programms wurden allerdings von Beginn an zwei Einschränkungen festgelegt:

- Das Programm muss unter einer Lizenz verfügbar sein, durch die die Verfügbarkeit und Nutzbarkeit auch für die Zukunft im universitären Umfeld sichergestellt ist. Idealerweise wäre dies dieselbe Lizenz, unter der auch das zu dokumentierende Software-Produkt steht.
- Das Programm muss für die Programmiersprache C++, in der die Bibliothek C-XSC geschrieben ist, geeignet sein. Aus diesem Grunde wurde z. B. das sehr bekannte Werkzeug JavaDoc nicht in die Auswahl aufgenommen.

2.1 Auswahl eines Dokumentations-Werkzeug

2.1.1 Doc++

Doc++[1] ist ein älteres Programm, das vorwiegend für die Dokumentation von C++ und Java-Quelltext entwickelt wurde. Genau wie die nachfolgend genannten Anwendungen erstellt es eine Dokumentation basierend auf in den Quelltext eingefügten speziell formatierten Kommentaren. Doc++ wird von verschiedenen Projekten, die auf der Homepage[2] genannt werden, verwendet. Leider wird dort kein bekannteres Projekt genannt, das einem größeren fachfremden Publikum bekannt sein dürfte.

2.1.2 Doxygen

Doxygen[3] ist ein in der Open-Source-Szene für viele Projekte verwendetes Dokumentations-Werkzeug. In seiner ersten Version basierte es zum Teil auf Quellcode des Programmes Doc++. Diese Teile sind aber in aktuelleren Versionen komplett neu- oder umgeschrieben worden. Als „prominente“ Beispiele, die die jeweilige API-Dokumentation mithilfe von Doxygen automatisch erstellen lassen, sind unter anderem die Projekte Asterisk[4], D-Bus[5] und ScummVM[6] zu nennen. Weitere Projekte, die Doxygen verwenden, werden wie bei Doc++ auf der Projekthomepage[7] gelistet.

1 Projekthomepage http://docpp.sourceforge.net/ (Stand: 15.03.2007)
2 http://docpp.sourceforge.net/projects.html (Stand: 12.05.2007)
3 Projekthomepage http://www.stack.nl/~dimitri/doxygen/ (Stand: 15.03.2007)
4 Projekthomepage http://www.asterisk.org/ (Stand: 15.03.2007)
5 Projekthomepage http://www.freedesktop.org/wiki/Software/dbus (Stand: 15.03.2007)
6 Projekthomepage http://scummvm.org/ (Stand: 15.03.2007)
7 http://www.stack.nl/~dimitri/doxygen/projects.html (Stand: 12.05.2007)

Doxygen bietet einen entsprechend hohen Funktionsumfang, der später noch im Rahmen eines Vergleich der ausgewählten Anwendungen näher erläutert wird.

2.1.3 HeaderDoc

HeaderDoc[8] ist eine Anwendung, die von der Firma Apple als Open Source zur Verfügung gestellt wird. Der Schwerpunkt des Einsatzes dieses Programms liegt im Rahmen der „Mac OS X Developer Tools“, und damit natürlich auf der Mac OS-Plattform. Es dient wie auch die anderen Anwendungen dazu, aus einem bestehenden Quelltext eines Softwareprogrammes automatisiert eine zugehörige Dokumentation zu erstellen.

2.2 Benötigter Funktionsumfang

Um die ausgewählten Produkte miteinander zu vergleichen, wurden zunächst die Kriterien, anhand der die Programme miteinander verglichen werden sollen, festgelegt:

- Unterstützte Programmiersprachen
- Ausgabeformate
- Dokumentation der Anwendung
- Entwicklungsstand
- Unterstützung für mathematische Formeln
- Grafik-Unterstützung
- Verfügbarkeit
- Lizenz

Der wichtigste Punkt, der auch schon bei der Auswahl der Programme berücksichtigt wurde, ist die Eignung für die Programmiersprache C++. Wäre ein Programm nicht in der Lage, einen solchen Quelltext zu analysieren, wäre es nicht möglich, die Dokumentation der Klassen-Bibliothek C-XSC mit dem entsprechenden Programm zu erstellen.

Ein weiterer Punkt ist das Ausgabeformat, in dem die Dokumentation generiert wird. Mindestanforderung ist die Ausgabe im HTML-Format, um diese leicht zusammen mit der Klassenbibliothek ausliefern bzw. im Internet online zur Verfügung stellen zu können. Jedes weitere Ausgabeformat, das von einer Anwendung unterstützt wird, bedeutet weitere Flexibilität und stellt einen Mehrwert des Produktes dar.

Um die jeweilige Anwendung auch entsprechend ihren Fähigkeiten verwenden zu können, ist natürlich auch eine aktuelle und umfassende Dokumentation von Nöten. Je umfangreicher und vielfältiger (z. B. leicht auffindbare Howtos oder Dokumentationen Dritter, Community-Foren, Mailing-Listen, etc.) diese ausfiel, desto positiver wurde dies bei der endgültigen Entscheidung berücksichtigt.

Für die Entscheidung ebenso zu berücksichtigen ist der aktuelle Entwicklungsstand bzw. ob das Programm momentan noch weiterentwickelt wird. Ist seit dem letzten stabilen Release

8 Projekthomepage http://developer.apple.com/opensource/tools/headerdoc.html (Stand: 15.03.2007)

schon einen längere Frist vergangen, lässt dies vermuten, das auch in Zukunft mit keiner neuen Version zu rechnen ist. Somit ist die Verwendung des Programmes nicht mehr anzuraten, weil in Zukunft neue Anforderungen an das Programm gestellt werden könnten, die dann nicht mehr in das Produkt integriert werden (können).

Da im Fall der Klassen-Bibliothek C-XSC auch die Nutzung und Anzeige von mathematischen Formeln in der Dokumentation zwingend notwendig ist, sollte jedes Produkt einen Mechanismus bieten, der diese Verwendung ermöglicht.

Nicht zwingend erforderlich, aber doch sinnvoll ist die Einbindung vorhandener Grafiken, also durch manuellen Eingriff des Dokumentations-Autors, beziehungsweise die automatische Erstellung von Grafiken (z. B. UML-Vererbungs-Diagramme), die in der Dokumentation Verwendung finden können.

Da C-XSC Plattform-übergreifend verfügbar ist, sollte die Dokumentationserstellung natürlich auch Plattform-übergreifend möglich sein. Dies wurde unter dem Punkt „Verfügbarkeit“ bei der Entscheidungsfindung berücksichtigt.

Unter dem Punkt Lizenz wurde die jeweils zugrunde liegende Lizenz, unter der das Produkt verwendet werden kann, aufgelistet. Bei diesem Punkt wurde, wie schon in der Einleitung erwähnt, darauf wertgelegt dass eine kostenfreie Verwendung erlaubt ist. Da C-XSC unter der GNU General Public License[9] lizenziert wird, ist es bei diesem Punkt besonders hervorzuheben, wenn die Dokumentationsanwendung dieselbe Lizenz verwendet, da somit jegliche Lizenz-Problematiken, die den gemeinsamen Einsatz verhindern könnten, von Beginn an ausgeschlossen sind.

2.3 Auswahl der am besten geeigneten Anwendung

Zur Auswahl des besten Programms wurde Tabelle 1 anhand der Angaben auf der jeweiligen Projekthomepage angefertigt, in der die maßgeblichen Faktoren bzw. Funktionen und deren Unterstützung durch das jeweilige Programm aufgeführt und in Beziehung zueinander gesetzt werden.

9 Lizenztext: http://www.gnu.org/copyleft/gpl.html (Stand: 26.04.2007)

Kriterium / Funktion	Doc++	Doxygen	HeaderDoc
Unterstützte Programmiersprachen	C, C++, IDL, Java	C, C++, Java, Objective-C, Python, IDL, (teilweise PHP, C#, D)	C, C++, Objective-C, Java, Javascript, Pascal, PHP, Perl, Shell-Skripte
Ausgabeformate	HTML, Latex	HTML, Latex, Man Pages, RTF, XML (indirekt: Compressed HTML, Postscript, PDF)	HTML, XML
Dokumentation der Anwendung	Gut	Gut	Gut
Entwicklungsstand[10]	22.12.2002	04.04.2007	07.11.2006
Unterstützung für mathematische Formeln	+	+	–
Grafik-Unterstützung (manuell / automatisch)	+ / +	+ / +	– / –
Verfügbarkeit (Lt. Projekthomepage)	Linux, Unix, Windows	Linux, Unix, Windows, MacOS	Linux, Unix, MacOS
Lizenz	GNU General Public License	GNU General Public License	Apple Public Source License[11]

Tabelle 1: Entscheidungstabelle

Aus Tabelle 1 ergibt sich, dass insbesonders von der Verwendung des Produktes HeaderDoc in der aktuellen Version abzuraten ist, da es keine Unterstützung für die Eingabe und Anzeige mathematischer Formeln sowie die Einbindung und automatische Erstellung von Grafiken bietet.

Gegen die Verwendung des Programmes Doc++ spricht vorrangig das letzte Release-Datum, das im Jahr 2002 liegt. Es ist zu erwarten, dass dieses Produkt nicht mehr gepflegt wird. Dadurch ist die Zukunftsfähigkeit der Software fraglich und von einer Nutzung zum jetzigen Zeitpunkt abzuraten, obwohl es die geforderten notwendigen Funktionen zur Dokumentation von C-XSC anbieten kann.

Wie aus Tabelle 1 ersichtlich, spricht also alles für die Nutzung des Programmes Doxygen, da dieses alle benötigten Funktionen und darüber hinaus weitere komfortable Funktionen bietet, wie z. B. eine größere Anzahl Ausgabe-Formate als bei den Vergleichsprodukten. Zudem spricht die große Verbreitung auch bei größeren Software-Projekten für die Verwendung von Doxygen, da es offensichtlich die meisten Erfordernisse professioneller Dokumentation, wie sie bei großen Software-Projekten nötig ist, erfüllen kann.

10 Stand: 26.04.2007

11 Lizenztext: http://www.opensource.apple.com/apsl/ (Stand: 26.04.2007)

3 Dokumentation der Klassen-Bibliothek

3.1 Warum C-XSC?

C-XSC ist eine Klassen-Bibliothek, die unter anderem an der Universität Karlsruhe und der Universität Wuppertal entwickelt wurde bzw. weiterentwickelt wird. Die Bibliothek wird im wissenschaftlichen Umfeld bei der Berechnung von numerischen Fragestellungen genutzt, um verifizierte Resultate erzielen zu können, bei denen sichergestellt ist, dass das Ergebnis der Berechnung zumindest den Ergebnisbereich eindeutig und verifiziert wiedergibt.

Die Wahl für die Dokumentation ist aus mehreren Gründen auf diese Klassen-Bibliothek gefallen.

Zunächst wird sie von Mitarbeitern der Universität Wuppertal (weiter-)entwickelt. Dieser Fakt unterstützt die Erstellung der Dokumentation, da bei Fragen zum Quelltext der Bibliothek, diese direkt von (Mit-)Entwicklern der Bibliothek beantwortet werden können. Weite Teile der bisher veröffentlichten Dokumentationen sind schon einige Jahre alt und deshalb teilweise überholt. Allerdings wurden auch diese für die Erstellung der Dokumentation im Rahmen dieser Ausarbeitung herangezogen. Insbesonders wurden die Bücher [1] und [2], sowie der Artikel [3] genutzt, um die Funktion, den Aufbau und die Nutzung der Klassen-Bibliothek zu verstehen.

Ein weiterer Punkt ist die Entstehung des Quelltextes. Die Bibliothek hat eine längere Historie, es handelt sich also um „gewachsenen“ Quelltext. Über einen langen Zeitraum haben viele Menschen an der Erstellung des Codes mitgewirkt. Viele mit jeweils einem eigenen Stil, wie sie Programmcode schreiben und kommentieren. Dadurch ergeben sich praktisch von selbst Schwierigkeiten bei der Dokumentation und Verwaltung des Quelltextes. Es gibt z. B. keine eindeutigen Richtlinien, wie der Code dokumentiert werden muss. An vielen Stellen fehlen zudem Kommentare oder der Quelltext ist nicht einheitlich dokumentiert (z. B. Kommentare teilweise in Deutsch, teilweise in Englisch). Diese Herausforderungen stellen sich auch bei vielen anderen ähnlich entstandenen Klassen-Bibliotheken, sodass C-XSC hier als ein Beispiel für andere Projekte dieser Art dienen kann.

Es ist nun unter anderem ein Ziel dieser Ausarbeitung, den Entwicklern eindeutige Richtlinien für die Dokumentation der Bibliothek an die Hand zu geben, um auch die zukünftige Wartbarkeit und Erweiterbarkeit des Quelltextes sicherzustellen.

3.2 Aufbau und Struktur von C-XSC

3.2.1 Aufbau

Die Klassen-Bibliothek C-XSC baut auf zwei – in Abbildung 1 dargestellte – grundlegende Teile auf, die sich hauptsächlich aus der gewachsenen Struktur der Bibliothek ergeben haben. Das RTS-Modul wurde 1990 bis 1991 für die Überführung der Pascal-XSC Bibliothek in C-

Code geschaffen[12]. Die FI-Lib[13] ist eine eigenständige Klassen-Bibliothek, die von der C-XSC Bibliothek genutzt wird, um präzise und schnelle Rechenfunktionen zu implementieren. Der Benutzer kommt mit diesen Klassen-Bibliotheken allerdings nicht in Berührung, da alle Aufrufe durch Wrapper-Klassen bzw. -Methoden realisiert sind.

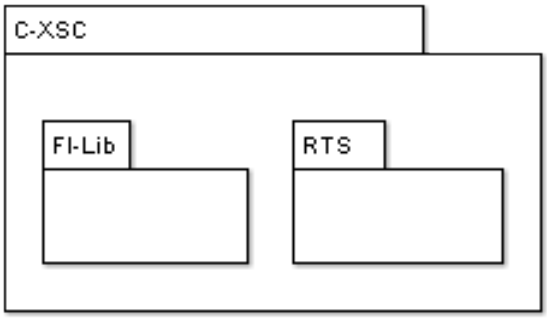


Abbildung 1: Aufbau C-XSC

Zudem besteht C-XSC aus mehreren weiteren Teilen, deren Zahl in künftigen Versionen weiter ausgebaut werden soll. Zur Zeit (C-XSC V2.2.1) umfasst sie folgende Teile:

- Die Klassen-Bibliothek

  Sie stellt die eigentlichen Funktionen von C-XSC, also die Datentypen, die Rechenoperatoren, Zugriffsmethoden, usw., zur Verfügung.

- Die C++ Toolbox

  In der C++ Toolbox sind schon für vielfältige Problemstellungen fertige Lösungen vorbereitet, die ein Anwender für ein an dieser Stelle schon behandeltes Problem nur übernehmen und mit den eigenen konkreten Werten befüllen muss. Dadurch ist es möglich, viele übliche und wiederkehrende Problemstellungen mit getesteten Algorithmen ohne größeren Zeitaufwand zu bewältigen.

- Beispielprogramme

  Die Beispielprogramme dienen als Einleitung oder Tutorial, um für (Neu-)Einsteiger die Klassen-Bibliothek zu erläutern bzw. ihnen die Verwendung der zur Verfügung stehenden Mechanismen näher zu bringen.

3.2.2 Struktur

Die Klassen-Bibliothek C-XSC stellt die Mittel bereit, um als Programmierer wissenschaftliche Problemstellungen in einem mathematischen Algorithmus zu implementieren und somit zumindest näherungsweise numerisch lösen zu können.

12 http://www.math.uni-wuppertal.de/wrswt/xsc/history.html (Stand: 28.02.2007)
13 Projekthomepage http://www.math.uni-wuppertal.de/wrswt/software/filib.html (Stand: 28.02.2007)

C-XSC stellt zu diesem Zweck die folgenden 4 Grunddatentypen zur Verfügung:

- Real (`real`) – Darstellung reeller Zahlen gemäß beschränkter Fließ-Komma-Darstellung nach IEEE

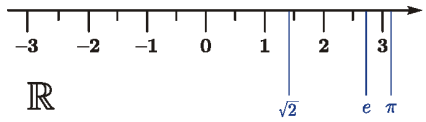


Abbildung 2: Darstellung reeller Zahlen auf dem Zahlenstrahl[14]

- Interval (`interval`) – Intervalle mit einer unteren und oberen reellen Grenze

  Mathematisch ausgedrückt: $[a,b]=\{x\in\mathbb{R} \mid a\le x\le b\}$ wobei die Zahlen a und b Elemente des beschränkten Zahlenbereiches des `real`-Datentyps sind.

- Complex (`complex`) – Darstellung von Zahlen in der komplexen Ebene durch Zerlegung der Zahl in einen Real- und einen Imaginär-Teil, die jeweils Elemente des beschränkten Zahlenbereiches des `real`-Datentyps sind.

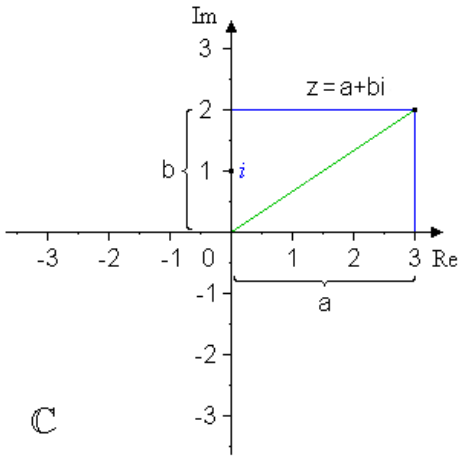


Abbildung 3: Darstellung einer komplexen Zahl in der komplexen Ebene[15]

- Complex Interval (`cinterval`) – Komplexe Intervalle mit einer unteren und oberen komplexen Grenze gemaß des Datentyps `complex`.

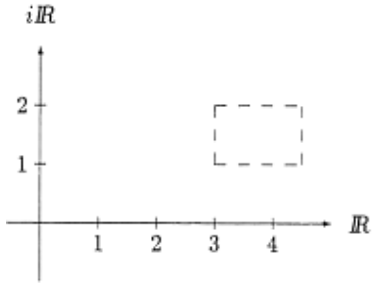


Abbildung 4: Darstellung des komplexen Intervalls [3,4.5]+[1,2]i[16]

14 Quelle: http://de.wikipedia.org/wiki/Bild:Real_number_line.svg (Stand: 14.05.2007)
15 Quelle: http://de.wikipedia.org/wiki/Bild:Gaussebene_Koordinatendarstellung.png (Stand: 14.05.2007) mit Modifikationen durch den Autor
16 Quelle: [2] Seite 77, Abbildung 3.2

Jeder dieser Datentypen kann auch in Vektoren und Matrizen beliebiger Dimensionen über die ebenfalls zur Verfügung stehenden entsprechenden Vektor- und Matrix-Klassen genutzt werden. Als besondere Eigenschaft ermöglicht C-XSC auch die dynamische Dimensions-Anpassung von Vektoren und Matrizen. Zu diesem Zweck müssen keine neue Variablen in den gewünschten neuen Dimensionen definiert und die Inhalte in die neuen Variablen kopiert werden, wie es z. B. bei normalen Arrays unter C++ der Fall wäre.

Außerdem gibt es für jeden Datentypen einen zugehörigen DotPrecision- und einen Langzahl-Datentyp. In dem DotPrecision-Datentyp wird das exakt berechnete Skalarprodukt gespeichert, das mit maximal einer Rundung in einen anderen Datentyp umgewandelt werden kann. Die Langzahl-Datentypen hingegen können einen wesentlich erweiterten aber trotzdem begrenzten Zahlenbereich im Vergleich zu den gemäß IEEE implementierten Standard-Datentypen erfassen.

3.3 Vorgehensweise

3.3.1 Überblick über die Bibliothek verschaffen

Zunächst muss man sich einen Überblick über die zu dokumentierende Bibliothek verschaffen. Im Fall von C-XSC wurde zu diesem Zweck die vorhandene, aber veraltete und unvollständige Dokumentation bzw. Literatur genutzt, um zunächst ein Verständnis für den Aufbau und die Funktionsweise der Bibliothek zu entwickeln. Anschließend wurden die vorhandenen Dokumentationskommentare innerhalb des Quelltextes analysiert und ebenso für die weiteren Schritte berücksichtigt. Zusätzlich wurden alle Quellen im Laufe der Bearbeitung mit einem Entwickler besprochen, um offene Fragen und Verständnisprobleme zu klären, sowie Rückmeldungen zur Dokumentation zu erhalten.

3.3.2 Einarbeitung in das Dokumentations-Werkzeug

Der nächste Schritt ist, sich zunächst grundlegend mit dem Dokumentationsprogramm der Wahl näher auseinanderzusetzen. Aus diesem Grunde wurde das Programm Doxygen zunächst ohne genauere Vorgaben, also mit den vorgegebenen Standard-Einstellungen, auf die Klassen-Bibliothek angewendet. Um diese Standard-Einstellungen zu erhalten, ist es zunächst notwendig, eine Konfigurations-Datei für Doxygen zu erstellen. Da die Standard-Einstellungen verwendet werden sollen, muss die Erstellung durch das Programm Doxygen selbst geschehen, das hierfür den Kommandozeilen-Parameter `-g` gefolgt von dem gewünschten Dateinamen der Konfigurations-Datei anbietet. Der Befehl lautet also:

```
doxygen -g doxyfile
```

In der erstellten Konfiguration wird jede mögliche Option mit einem Standardwert belegt und durch einen kurzen Kommentar beschrieben. Diese Hinweise und die Dokumentation [4] ermöglichen später die schnelle Modifikation der einzelnen Optionen.

Da später die Option einer zusätzlichen Konfigurationsdatei gegeben sein soll, um eine andere Dokumentation (z. B. eine Entwickler-Dokumentation) zu erstellen, wurde die Konfigurationsdatei `src-doxyfile` benannt und im Unterverzeichnis `src/` abgelegt.

Im Anschluß wurde eine erste Fassung der Dokumentation wie im Abschnitt „Generieren der Dokumentation“ erläutert erstellt.

Das Resultat – die automatisch erzeugte Dokumentation – wurde dann mit den eigenen Erwartungen verglichen. Nun werden Modifikationen an der Standard-Konfiguration vorgenommen, um sie weitestgehend an die eigenen Vorstellungen anzupassen. Die vorgenommenen Einstellungen stehen zu Beginn der Dokumentation noch unter Vorbehalt. Da im weiteren Verlauf noch Änderungen und strukturelle Anpassungen zu erwarten sind, müssen diese auch in der Konfigurationsdatei berücksichtigt werden können.

Weil bei Verwendung der Standard-Einstellungen die Dokumentation noch einen beachtlichen Umfang erreichte, wurden ein paar konkrete Maßnahmen ergriffen, um den Umfang der Dokumentation einzugrenzen:

- Zunächst wurde über die Option `FILE_PATTERNS` in der Konfigurationsdatei festgelegt, dass nur Dateien mit den Endungen `*.hpp`, `*.cpp`, und `*.inl` für die weitere Verwendung vorgesehen sind. Nur in diesen Dateien steht im Fall der Bibliothek C-XSC relevanter Quelltext. Bei der Dokumentation von anderen Projekten müssen deren Quelltext-Dateien an dieser Stelle natürlich entsprechend berücksichtigt werden.
- Viele Quelltext-Dateien wurden von der Dokumentationserstellung ausgeschlossen, da sie für die zu erstellende Benutzer-Dokumentation irrelevant sind (z. B. Test-Dateien (Dateinamen: `test*.*`), die für Funktionstests verwendet werden).
- In der Konfigurationsdatei wurde festgelegt, dass private Variablen und Methoden nicht dokumentiert werden müssen. Auf diese hat der Anwender keinen Zugriff, somit brauchen sie in der Dokumentation auch nicht erscheinen.

3.3.3 Einstieg in die Dokumentierung finden

Der nächste Schritt besteht darin, den passenden Einstieg in die Dokumentation der Bibliothek zu finden. Im Falle von C-XSC wurde mit der Klasse `real` begonnen, auf die die meisten Elemente der Bibliothek angewiesen sind und aufbauen. Die Klasse `real` stellt den grundlegenden Daten-Typ für C-XSC dar. Außerdem ist diese Klasse noch recht einfach beschaffen, so dass die ersten „Gehversuche“, um die Dokumentations-Kommentare einzufügen, schnell erledigt, nachvollzogen und kontrolliert werden können.

In der Folge wurden die einzelnen Klassen in Rücksprache mit einem Entwickler, entsprechend ihres „Schwierigkeitsgrades“ bzw. Umfanges eingeteilt und der Reihe nach bearbeitet. So wurden zuerst die skalaren Grunddatentypen Real, Complex, Interval, Complex Interval und der Datentyp DotPrecision in seinen Ausprägungen für die Grunddatentypen dokumentiert. Danach wurden die Vektor-Klassen derselben Datentypen, gefolgt von den Matrix- und schließlich den Langzahl-Klassen bearbeitet.

Die während der Bearbeitung festgelegten Regeln, Richtlinien und Notationen wurden in der später erläuterten Dokumentations-Richtlinie festgehalten, sodass die Dokumentation einem einheitlichen Stil folgt.

3.3.4 Kommunikation mit den Entwicklern

Bei regelmäßig vereinbarten Treffen mit einem Entwickler wurde der jeweilige Stand der Dokumentation und der Bibliothek miteinander synchronisiert. Auf diesem Wege wurde trotz der parallelen Weiterentwicklung der Bibliothek während der Erstellung der Dokumentation sichergestellt, dass der aktuelle Dokumentationsstand mit in den aktuellen Entwicklungsstand der Bibliothek einfloss, durch alle Entwickler wahrgenommen wurde und so das Feedback der Entwickler noch während der Bearbeitungszeit einen qualitativen Einfluss auf die Dokumentation hatte.

3.3.5 Nutzung vorhandener Ressourcen

Um nicht die komplette Dokumentation von Grund auf neu erstellen zu müssen, wird zudem – wo dies sinnvoll ist – die alte teilweise vorhandene Dokumentation in die neue Dokumentation übernommen und wenn notwendig entsprechend angepasst. Konkret geschah dies z. B. bei einigen ausführlichen Klassen- und Methodenbeschreibungen oder Grafiken, die aus [1] und [2] übernommen wurden. Beispielhaft sei dies an der Beschreibung der Klasse Real hier aufgezeigt:

```
//! The Scalar Type real
/*!
The arithmetic of C-XSC is based on the IEEE standard for binary floating-
point arithmetic. Data of the type real consist of all floating-point
numbers and special values specified by the standard for floating-point
numbers of double mantissa length. Therefore, a number of the type real is
a 64-bit value, and the base \f$ b \f$ of the floating-point system \f$ R =
R(b , l , e_min, e_max)\mbox{ is }2 \f$.

The C-XSC data type real differs only in some special aspects such as error
handling from the C type double if used on a IEEE standard conforming
arithmetic. If the C compiler on the host computer is not standard
conforming, the data type real uses its own IEEE software arithmetic.
Hence, the introduction of a new data type is necessary for the portability
of C-XSC.
(...)
*/
class real
{
(...)
}
```

Die ausführliche Beschreibung wurde in diesem Fall aus [2] Seite 70 übernommen und stellt sich in der Dokumentation wie in Abbildung 5 dar.

Detailed Description

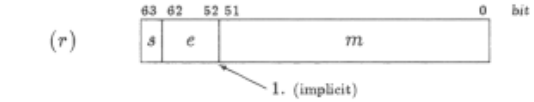
The Scalar Type **real**.

The arithmetic of C-XSC is based on the IEEE standard for binary floating-point arithmetic. Data of the type **real** consist of all floating-point numbers and special values specified by the standard for floating-point numbers of double mantissa length. Therefore, a number of the type **real** is a 64-bit value, and the base b of the floating-point system $R = R(b, l, e_min, e_max)$ is 2.

The C-XSC data type **real** differs only in some special aspects such as error handling from the C type double if used on a IEEE standard conforming arithmetic. If the C compiler on the host computer is not standard conforming, the data type **real** uses its own IEEE software arithmetic. Hence, the introduction of a new data type is necessary for the portability of C-XSC.

Structure

A sketch of the **real** floating-point data format is given in the figure below.



$$r = \begin{cases} (-1)^s \cdot 1.m \cdot 2^{e-1023} & 1 \le e \le 2046 \quad \text{normalized} \\ (-1)^s \cdot 0.m \cdot 2^{-1022} & e = 0, m \neq 0 \quad \text{denormalized} \\ (-1)^s \cdot 0 & e = 0, m = 0 \quad \text{signed zero} \\ (-1)^s \cdot \infty & e = 2047, m = 0 \quad \text{signed infinity} \\ \text{NaN} & e = 2047, m \neq 0 \quad \text{not a number} \end{cases}$$

The real Floating-Point Format

Abbildung 5: Detaillierte Beschreibung der Klasse Real

Bei einigen der mathematischen Funktionen wurden auch Erläuterungen aus der freien Internet-Enzyklopädie Wikipedia[17] übernommen. Trotz genereller Vorbehalte wegen möglicherweise falscher Angaben in Artikeln wurde die Wikipedia in dem vorliegendem Fall als Quelle berücksichtigt, weil sie in mathematischen und technischen Bereichen sehr gut und in der Regel auch fehlerfrei aufgestellt ist. Zudem wirkt die Rückkopplung zu den Entwicklern an dieser Stelle natürlich als Sicherheitsnetz beziehungsweise korrigierender Faktor, sollte diese zusätzliche Dokumentation wider Erwarten doch Fehler enthalten.

Als Beispiel für diese Verwendung sei hier die (komplementäre) Fehler-Funktion genannt:

17 http://www.wikipedia.com

```
/*!
\param arg The value for which to compute the value of the error function
\return The computed result of the error function

In mathematics, the error function (also called the Gauss error function)
is a non-elementary function which occurs in probability, statistics and
partial differential equations.

When the results of a series of measurements are described by a normal
distribution with standard deviation s and expected value 0, then
\f$ \mbox{erf} \left( \frac{a}{\sigma \sqrt{2}} \right) \f$ is the
probability that the error of a single measurement lies between \f$ -a \f$
and \f$ +a \f$ .

The error and complementary error functions occur, for example, in
solutions of the heat equation when boundary conditions are given by the
Heaviside step function.
*/
inline real erf(const real & arg)    { return q_erf(*(double*)&arg);  }
```

In der generierten Dokumentation stellt sich diese Kommentierung wie in Abbildung 6 dar.

real cxsc::erf (const real & *arg*) [inline]

The Gauss error function $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

Parameters:

arg The value for which to compute the value of the error function

Returns:

The computed result of the error function

In mathematics, the error function (also called the Gauss error function) is a non-elementary function which occurs in probability, statistics and partial differential equations.

When the results of a series of measurements are described by a normal distribution with standard deviation s and expected value 0, then $\mathrm{erf}\left(\frac{a}{\sigma\sqrt{2}}\right)$ is the probability that the error of a single measurement lies between $-a$ and $+a$.

The error and complementary error functions occur, for example, in solutions of the heat equation when boundary conditions are given by the Heaviside step function.

Definition at line **118** of file **rmath.inl**.

Abbildung 6: Dokumentation der Fehler-Funktion

3.3.6 Zusätzliche Dokumentation einbinden

Außer den Klassen, Typen, Funktionen und Methoden die eine Klassenbibliothek wie C-XSC bereitstellt, gehören zu einer professionellen Dokumentation auch weitergehende Informationen. Dies können z. B. Tutorials, Hintergrundwissen, Beispiele, Internetlinks und vieles Anderes sein.

Im Fall von C-XSC wurde eine Einleitung beziehungsweise Überblick zu C-XSC selbst, Erläuterungen zur „C++ Toolbox for Verified Computing“ und Erläuterungen zu ein paar einfachen Beispiel-Programmen beispielhaft in die Dokumentation aufgenommen.

Zu diesem Zweck wurde die neue Datei `docu.hpp` angelegt, in der sämtliche Zusatzdokumentation aufgenommen wird. Neue Seiten werden in eigenen Kommentar-Blöcken in diese Datei eingebunden. In der Dokumentation tauchen sie dann unter dem Punkt „Related Pages“ auf. Zusätzlich ist es Aufgabe der Autoren der jeweiligen Zusatzdokumentation sicherzustellen, dass ihre Dokumente auch von anderer Stelle der Dokumentation aus erreichbar sind. Dazu können gezielt Referenzen zu den neuen Bestandteilen in der bestehenden Dokumentation eingebunden werden.

Ein „Highlight“ bei der zusätzlichen Dokumentation ist die Aufnahme von Quelltexten an beliebigen Stellen in der Dokumentation. Zum Einen kann die gesamte Quelltext-Datei eingebunden, zum Anderen können auch einzelne Zeilen in die Dokumentation aufgenommen werden. Bei beiden Arten werden dabei die Anweisungen einem automatischen Syntax-Highlighting unterworfen, sodass die Struktur des Quelltextes hervorgehoben und das Verständnis für den Leser gesteigert wird.

Die genauer spezifizierten Anweisungen und Vorgehensweisen sind natürlich auch in der Dokumentations-Richtlinie aufgeführt.

3.3.7 Generieren der Dokumentation

Die Dokumentation kann schließlich mit dem Aufruf des Programms Doxygen und der Angabe der Konfigurationsdatei erstellt werden (s. Abbildung 7). Hier lautet der Aufruf im Verzeichnis `src/` also:

```
doxygen src-doxyfile
```

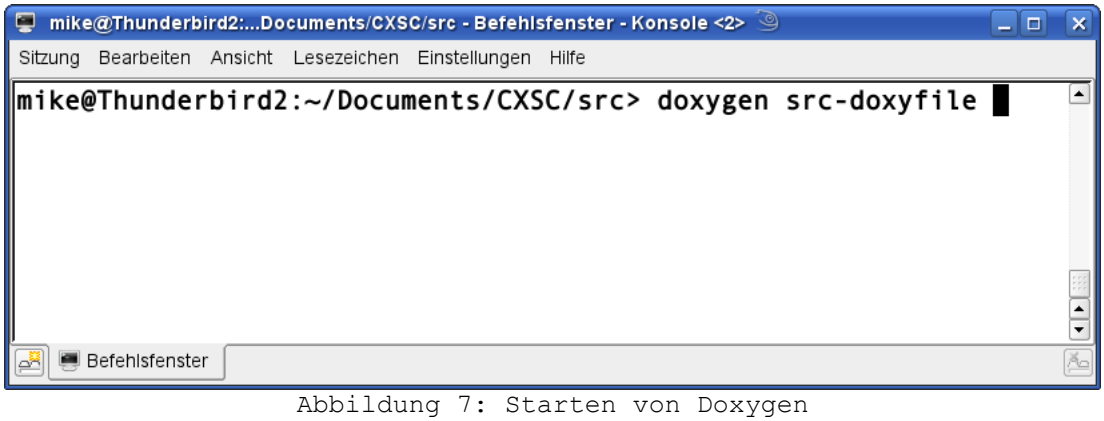


Abbildung 7: Starten von Doxygen

Doxygen beginnt dann mit dem Einlesen der Quellen und generiert die Dokumentation in dem Ziel-Verzeichnis, das in der Konfigurationsdatei angegeben wurde (hier im konkreten Fall das Verzeichnis `docu/apidoc/` innerhalb der Verzeichnishierarchie von C-XSC). Sollten Fehler bei der Generierung der Dokumentation auftreten, werden diese gesammelt und in der Datei `doxywarn.txt` im Hauptverzeichnis der C-XSC-Verzeichnishierarchie abgelegt. Typische

Ausgaben, die während der Erstellung der Dokumentation auf der Konsole ausgegeben werden, sind in Abbildung 8 dargestellt.

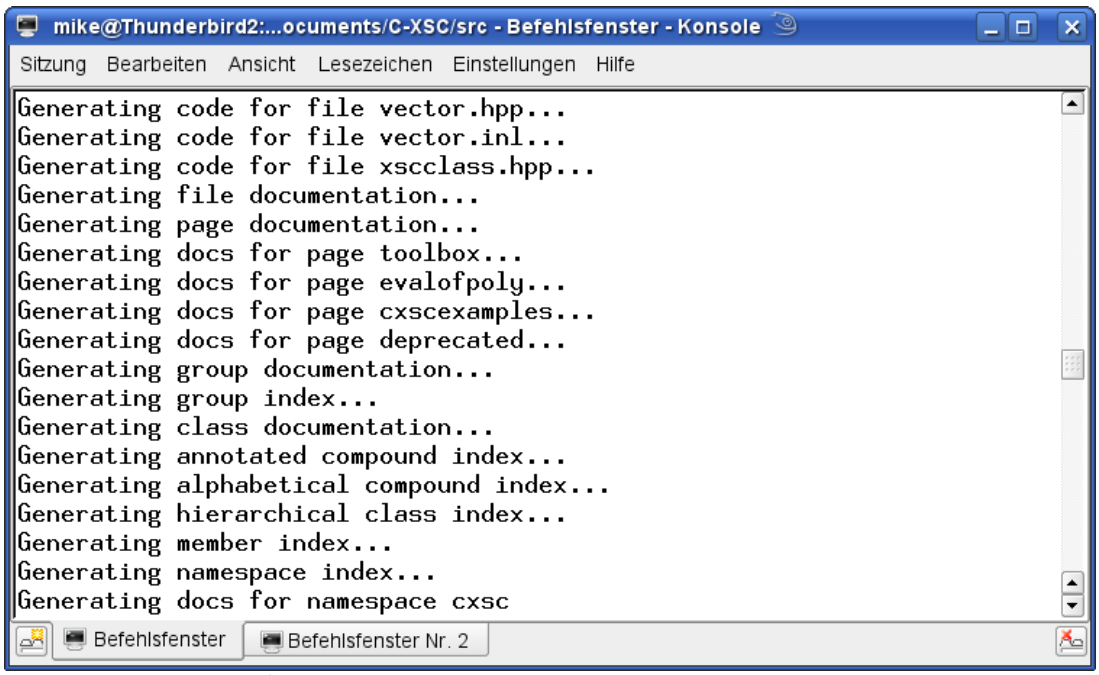


Abbildung 8: Konsolen-Ausgaben von Doxygen

Für die Erstellung der Dokumentation wurde hier Doxygen 1.5.2 unter SuSE Linux 10.2 verwendet. Für die automatische Erstellung der Grafiken wird zusätzlich das Programm-Paket GraphViz[18] benötigt, das hier in der Version 2.6-44 genutzt wurde. Für die Erstellung der Formel-Grafiken wird zudem ein einsatzbereites LaTeX-System benötigt, wie es gewöhnlich unter jedem Linux-System standardmäßig verfügbar ist.

3.4 Dokumentations-Richtlinie

Der Zweck einer Dokumentations-Richtlinie ist zum einen die Qualitätssicherung der zu erstellenden Dokumentation und zum anderen die Gewährleistung einer einheitlichen Fortschreibung der Dokumentation während der Weiterentwicklung der Klassen-Bibliothek.

Dadurch soll sichergestellt werden, dass Anwendern der Bibliothek immer eine aktuelle Dokumentation zur Verfügung steht. Natürlich muss diese auch so formuliert werden, dass sie auch von einem ungeübtem Anwender genutzt werden kann, um sich in die Klassenbibliothek einarbeiten zu können.

Zudem wird durch die automatische Generierung der Aufwand der Dokumentierung für die Entwickler verringert, da sie nun nur noch die neuen oder veränderten Stellen innerhalb der Klassenbibliothek im Quelltext gemäß der Richtlinie dokumentieren müssen und keine externen Dokumente für die Dokumentation pflegen müssen.

18 Projekthomepage http://www.graphviz.org/ (Stand: 12.05.2007)

3.4.1 Inhalt

In der Dokumentations-Richtlinie wurden die während der Erstellung der Dokumentation aufgestellten Regeln und Richtlinien gesammelt und zusammengefasst, um den Entwicklern der Klassen-Bibliothek eine Orientierung bei der weiteren beziehungsweise zukünftigen Dokumentation von C-XSC geben zu können. Die Richtlinie ist zu diesem Zweck in vier Bereiche aufgeteilt:

- Regeln

  In diesem Abschnitt werden die grundsätzlichen Dokumentationsregeln benannt, erklärt und anhand von einem Beispiel in der praktischen Verwendung gezeigt.

- Zusätzliche Dokumentation

  Besondere Regeln und Vorgehensweisen, wie zusätzliche Dokumentation, die nicht direkt mit bestimmten Klassen und Methoden der Klassenbibliothek in Verbindung zu bringen ist, verfasst werden soll, werden in diesem Abschnitt erläutert. Zu einer solchen zusätzlichen Dokumentation gehören zum Beispiel die Ausführungen zu den Beispiel-Programmen, die mit der Klassenbibliothek ausgeliefert werden.

- Dokumentation von Quelltext-Dateien

  Um Quelltexte in die Dokumentation aufnehmen und diese besonders dokumentieren zu können, wurde dieser besondere Abschnitt in die Richtlinie aufgenommen. Hier wird erklärt, wie die Quelltext-Dateien eingebunden werden und wie sie unter Umständen Zeile für Zeile dokumentiert werden können.

- Erstellen der Dokumentation

  Im letzten Abschnitt der Richtlinie ist schließlich kurz erklärt, wie die automatische Dokumentationserstellung überhaupt gestartet werden kann und wo sich das Resultat schließlich befindet.

3.4.2 Regeln

3.4.2.1 Grundsätzliches

- Grundsätzlich werden alle Kommentare in separaten Zeilen eingefügt!
- Jede für den Anwender der Klassenbibliothek nutzbare Methode muss zumindest mit einer Kurzbeschreibung erläutert sein. Bei trivialen Methoden (z. B. Ein-/Ausgabe-Operatoren) reichen diese Kurzbeschreibungen aus, so dass keine zusätzliche ausführlichere Dokumentation erstellt werden muss.
- Der Quelltext wird auf mehrere Dateien aufgeteilt:
  - Die Deklarationen befinden sich immer in der jeweiligen Header-Datei (`*.hpp`).
  - Der eigentliche Quelltext der Methoden befindet sich in der jeweiligen Code-Datei (`*.cpp`).

- Der Quelltext von Inline-Methoden befindet sich abweichend von der oberen Aussage in der jeweiligen Inline-Datei (`*.inl`).

3.4.2.2 Klassen-Beschreibungen

In den Header-Dateien (`*.hpp`) der Klassenbibliothek wird oberhalb des Klassennamens je ein Dokumentations-Kommentar mit einer kurzen und mit einer ausführlichen Beschreibung der Klasse eingefügt. Die Kurzbeschreibung wird durch den Ausdruck `//!` eingeleitet, dem der Text der Kurzbeschreibung folgt. Die ausführliche Beschreibung beginnt mit den Zeichen `/*!` und endet mit den Zeichen `*/`

Beispiel:

```
//!The Scalar Type real
/*!
The arithmetic of C-XSC is based on the IEEE standard for binary floating-
point arithmetic. Data of the type real consist of all floating-point
numbers and special values specified by the standard for floating-point
numbers of double mantissa length. Therefore, a number of the type real is
a 64-bit value, and the base b of the floating-point system R = R(b, l,
e_min, e_max) is 2.

The C-XSC data type real differs only in some special aspects such as error
handling from the C type double if used on a IEEE standard conforming
arithmetic. If the C compiler on the host computer is not standard
conforming, the data type real uses its own IEEE software arithmetic.
Hence, the introduction of a new data type is necessary for the portability
of C-XSC.
*/
class real
{
...
}
```

3.4.2.3 Kurzbeschreibungen

Die in den Header-Dateien deklarierten Methoden werden mit einem einzeiligen Dokumentationskommentar dokumentiert, der mit den Zeichen `//!` beginnt. Dieser Kommentar stellt eine Kurzbeschreibung der Aufgabe der jeweiligen Methode dar.

Beispiel:

```
//! Constructor of class real
real(void) throw () { }
```

3.4.2.4 Detaillierte Beschreibungen

Ausführliche Kommentare mit genaueren Beschreibungen, Beispielen etc. werden in der Regel in den Code-Dateien (`*.cpp`) bzw. in den Dateien eingefügt, in denen der eigentliche Programm-Code der Methode steht. Der Kommentar beginnt mit dem Zeichen `/*!` und endet mit den Zeichen `*/`. Dieser Kommentar kann und soll sich über mehrere Zeilen erstrecken. Bei

Methoden die diesen umfangreicheren Kommentar benötigen, sollen unbedingt folgende Elemente mitkommentiert werden:

- Die Parameter
- Die Rückgabewerte

Der vorhandene Kurzkommentar wird dabei immer in die detaillierte Methodenbeschreibung übernommen. Es ist also nicht notwendig, diesen Teil erneut zu verfassen.

Beispiel:

```
/*!
\param sign The sign of the number
\param expo The exponent of the number
\param manthigh The high byte of the mantissa
\param mantlow The low byte of the mantissa
\return The IEEE 64-bit floating point number
*/
const real& MakeHexReal(int sign, unsigned int expo, a_btyp manthigh,
a_btyp mantlow)
{
...
}
```

Sollte die Funktion hingegen veraltet sein, so reicht die Markierung als `\deprecated` mit Angabe der neuen vorgesehenen Funktion sowie ggfs. eines Verweises auf die entsprechende neue Funktion mit dem Ausdruck `\sa`.

Beispiel:

```
/*!
\deprecated use standard constructors for typecasting

\sa cinterval(const real & a)
*/
inline cinterval _interval(const real & a) throw()
{
...
}
```

3.4.2.5 Absätze

Absätze werden in den Kommentaren durch einfaches Einfügen einer Leerzeile zwischen den Absätzen gebildet. Durch diese Vorgehensweise wird sichergestellt, dass der Text eines Absatzes nicht gesammelt in einer Zeile geschrieben werden muss und somit auch im Quelltextes lesbar hinterlegt werden kann.

Beispiel:

```
//!The Scalar Type real
/*!
The arithmetic of C-XSC is based on the IEEE standard for binary floating-
point arithmetic. Data of the type real consist of all floating-point
numbers and special values specified by the standard for floating-point
numbers of double mantissa length. Therefore, a number of the type real is
a 64-bit value, and the base b of the floating-point system R = R(b, l,
e_min, e_max) is 2.

The C-XSC data type real differs only in some special aspects such as error
handling from the C type double if used on a IEEE standard conforming
arithmetic. If the C compiler on the host computer is not standard
conforming, the data type real uses its own IEEE software arithmetic.
Hence, the introduction of a new data type is necessary for the portability
of C-XSC.
*/
class real
{
...
}
```

3.4.2.6 Grafiken

Wenn Bilder innerhalb der Dokumentation verwendet werden sollen, werden diese im PNG-Format im Verzeichnis `docu/images/` abgelegt. Diese können dann im Quellcode mit dem Ausdruck `\image` eingebunden werden. Zusätzlich muss dazu immer auf das `\image` folgend der Ausdruck `html` angeben werden, dem widerum der Dateiname und die Benennung der Grafik jeweils in Anführungszeichen folgen.

Beispiel:

```
/*!
...
\image html “cinterval.png” “Complex Interval [3.0, 4.5] + [1.0, 2.0]i”
...
*/
```

3.4.2.7 Formeln

Sollen mathematische Formeln in die Dokumentation aufgenommen werden, so bedarf es der besonderen LaTeX-Notation. Genauere Erläuterungen zu dieser Notation sind in einschlägiger Literatur als auch z. B. in [5] und [6] zu finden. In den Kommentaren werden diese Abschnitte mit einem `\f$` eingeleitet und beendet, wenn die Formel innerhalb eines Fließtextes erscheinen soll.

Beispiel:

```
//! Calculates \f$ \sin(x) \f$
inline real sin (const real&) throw();
```

Soll die Formel hingegen zentriert in einer separaten Zeile angezeigt werden, so wird der entsprechende Abschnitt mit dem Ausdruck `\f[` eingeleitet und mit dem Ausdruck `\f]` beendet.

Beispiel:

```
//!The Scalar Type interval
/*!
The data type interval is used to represent intervals over the real
floating-point numbers:

\f[
\left[ a \right] = \left[ \underline a , \overline a \right] := \left\{ x
\in R | \underline a \le x \le \overline a \right\} \in R
\f]

i.e. \f$ \left[ a \right]\f$ represents the set of all real numbers
enclosed within the bounds \f$ \underline a , \overline a \in R \f$ .
...
*/
class interval
{
...
}
```

3.4.2.8 Querverweise

Sollen Querverweise zu anderen Methoden in die Dokumentation eingefügt werden, geschieht dies über den Ausdruck `\sa` im entsprechenden ausführlichen Dokumentations-Block für die Methode, in der dieser Verweis erscheinen soll.

Beispiel:

```
/*!
\deprecated use standard constructors for typecasting

\sa cinterval(const real & a)
*/
inline cinterval _interval(const real & a) throw()
{
...
}
```

3.4.2.9 Abschnitte

Soll in einem ausführlichen Kommentar-Block einzelne Abschnitte gebildet werden, so geschieht dies mit dem Ausdruck `\section` dem ein eindeutiger(!) Bezeichner sowie der Titel des Abschnitts folgt.

Alle Bezeichner sollten immer nur aus Kleinbuchstaben bestehen, da es sonst zu Fehlern bei der Dokumentationserstellung kommen kann!

Beispiel:

```
/*!
...
\section structure Structure
...
*/
```

3.4.2.10 Referenzen

Soll in einem Kommentar-Block eine Referenz zu einem anderen Abschnitt eingefügt werden, wird der Ausdruck `\ref` gefolgt von dem eindeutigen Bezeichner des entsprechenden Abschnitts eingefügt. In der generierten Dokumentation wird an dessen Stelle ein Link mit dem Titel des entsprechenden Abschnitts eingefügt.

Beispiel:

```
/*!
...
The \ref toolbox is the C++ edition of the <em>Numerical Toolbox for
Verified Computing</em>.
...
*/
```

3.4.2.11 Hervorhebungen

Sollen bestimmte Wörter oder Sätze in der Dokumentation hervorgehoben werden, so werden diese mit dem Ausdruck `<em>` eingeleitet und mit dem Ausdruck `</em>` beendet, äquivalent zu Auszeichnung in HTML-Notation.

Beispiel:

```
/*!
...
The \ref toolbox is the C++ edition of the <em>Numerical Toolbox for
Verified Computing</em>.
...
*/
```

3.4.3 Zusätzliche Dokumentation

Außer der regulären Dokumentation, die konkrete Klassen, Methoden etc. betrifft, kann auch zusätzliche ausführliche Dokumentation auf separaten Seiten erstellt werden. Die entsprechenden Dokumentationskommentare werden in der Datei `docu.hpp` vorgenommen, in der sämtliche Zusatzdokumentation gesammelt wird.

Die Datei `docu.hpp` ist dabei nur als Platzhalter-Datei zu verstehen. Sie findet in der Klassenbibliothek selbst keine Anwendung. Das bedeutet, in dieser Datei befindet sich kein regulärer Code der Klassenbibliothek C-XSC.

3.4.3.1 Neue Seite einfügen

Für jede neue separate Seite, wird in der Datei `docu.hpp` ein neuer Dokumentations-Kommentar gefolgt von dem Ausdruck `\page`, dem widerum ein eindeutiger Bezeichner sowie der Titel der Seite folgt, eingefügt. Es ist dabei darauf zu achten, dass jede neue Seite auch entsprechend über einen Link auf einer oder mehreren anderen Seiten verfügbar gemacht wird (s. Abschnitt Referenzen).

Beispiel:

```
/*!
\page toolbox C++ Toolbox for Verified Computing
...
*/
```

3.4.3.2 Inhaltsverzeichnis

Soll wie im folgenden Abschnitt „Abschnitte" beschrieben, eine Seite in mehrere Abschnitte eingeteilt werden, so muss zu Beginn der Seite ein Inhaltsverzeichnis der Seitenabschnitte eingefügt werden. Zu diesem Zweck wird für jeden Abschnitt der Seite eine neue Zeile beginnend mit einen Bindestrich gefolgt von dem Ausdruck `\ref` und dem jeweiligen Abschnittsbezeichner eingefügt.

Beispiel:

```
/*!
\page toolbox C++ Toolbox for Verified Computing

- \ref toolbox_sec_resultverification
- \ref toolbox_sec_history
...
*/
```

3.4.3.3 Abschnitte

Sollen einzelne Abschnitte gebildet werden, so geschieht dies mit dem Ausdruck `\section` dem ein eindeutiger Bezeichner sowie der Titel des Abschnitts folgt. Der Bezeichner wird in der Regel so gewählt, daß er folgendem Muster entspricht:

`BezeichnerDerSeite_sec_BezeichnerDesAbschnittes`

Alle Bezeichner sollten immer nur aus Kleinbuchstaben bestehen, da es sonst zu Fehlern bei der Dokumentationserstellung kommen kann!

Beispiel:

```
/*!
\page toolbox C++ Toolbox for Verified Computing

...

\section toolbox_sec_resultverification Why Numerical Result Verification?

...
*/
```

3.4.4 Dokumentation von Quelltext-Dateien

In der Dokumentation können auch separate Quelltext-Dateien eingebunden und dokumentiert werden, um z. B. Beispiele für die Anwendung von C-XSC zu geben. Es kann dabei der Quelltext insgesamt, aber auch einzelne Zeilen bzw. Abschnitte dokumentiert werden.

3.4.4.1 Gesamten Quelltext einbinden

Um den gesamten Code einer Quelltext-Datei einzubinden, kann der Ausdruck `\includelineno` gefolgt von dem Dateinamen der einzubindenden Datei genutzt werden. Die Datei muss dabei in einem in der Doxygen-Konfigurationsdatei angegebenen Verzeichnis vorhanden sein. Hierfür vorgesehen sind zur Zeit die Verzeichnisse `examples/` und `CToolbox/`. Die Datei wird dann in der Dokumentation komplett mit Zeilennummern vor jeder Zeile eingefügt.

Beispiel:

```
/*!
...
\section cxscexamples_sec_ex1 Example 1 - An Introduction

We will start with the following simple program showing you how to use
intervals, compute one operation and print out the result:

\includelineno example.cpp

...
*/
```

3.4.4.2 Einzelne Zeilen und Abschnitte einbinden

Nachdem eine Quelltext-Datei eingebunden wurde, können nachfolgend einzelne Zeilen und Abschnitte der Datei im Detail eingebunden und dokumentiert werden. Zu diesem Zweck wird der Ausdruck `\skipline` bzw. `\skipline` und `\until` verwendet. Auf die Ausdrücke folgend muss jeweils ein eindeutiger Ausdruck des Quelltextes angegeben werden, bis zu dem die Zeilen übersprungen bzw. eingebunden werden sollen. Die Zeilen werden dabei von oben nach unten abgearbeitet, so dass in der Regel eine sequentielle Beschreibung des

Quelltextes möglich ist. Wichtig hierbei zu wissen ist, dass jeweils die zuletzt eingebundene Quelltext-Datei für die Einbindung der einzelnen Zeilen genutzt wird.

Beispiel:

```
/*!
...
\includelineno example.cpp

Let's start investigating the interesting lines. With the line

\skipline interval.hpp

you include the functionality of the interval class of C-XSC in your
program. After that you have to inform the compiler about the namespace
cxsc - where all classes and methods of C-XSC are stored in - to use C-XSC
without fully qualified identifiers.

\skipline using namespace cxsc

Then you declare your variables and assign adequate values in the following
two lines.

\skipline interval a
\until b =

Finally you want to print out the result for your desired computation.

\skipline cout

So it's just that easy to use C-XSC.
...
*/
```

3.4.5 Erstellen der Dokumentation

Die Dokumentation kann durch den Aufruf `doxygen src-doxyfile` im Unterverzeichnis `src/` der Klassen-Bibliothek generiert werden.

Die Dokumentation ist anschließend im Verzeichnis `docu/apidoc/` zu finden.

Um Inkonsistenzen aufgrund einer schon vorhandenen Dokumentation in diesem Verzeichnis zu vermeiden, sollte dieses Verzeichnis vor der Dokumentations-Generierung gelöscht werden. Dadurch werden bei der erneuten Erstellung der Dokumentation sämtliche automatisch generierten Grafiken neu erstellt.

3.5 Probleme und Lösungsansätze

Natürlich sind der während der Erstellung der Dokumentation auch Probleme aufgetreten. Auf die wichtigsten Probleme soll an dieser Stelle noch einmal eingegangen werden.

3.5.1 Quelltext-Aufteilung

Schon bei der Dokumentation der ersten Klasse, wurde klar, dass die Dokumentation der gesamten Bibliothek schwieriger als zunächst gedacht würde, da der Quelltext einer Klasse sich nicht nur auf eine bzw. zwei Dateien (Header- und Code-Datei) beschränkt sondern unter Umständen auf mehrere Dateien aufgespalten sein kann (z. B. zusätzliche Datei für die Implementierung von Inline-Funktionen, Auslagerung der mathematischen Funktionen in separate Dateien). Zusätzlich erschwert wurde die Analyse dadurch, dass die Aufteilung trotz alledem nicht konsequent eingehalten wurde und somit doch Teile, die systematisch in die eine Datei gehören in einer anderen auftauchen.

Beispielhaft sei dies hier an einem Auszug aus der Datei `real.hpp` gezeigt, in der eigentlich nur die Deklarationen von Klassen und Methoden definiert sein sollen. Trotzdem tauchen an dieser Stelle Quelltexte der deklarierten Methoden auf, die eigentlich in der separat angelegten Datei `real.inl` zu finden sein sollen:

```
(...)
//! Returns the greater value of two real values
inline real max(const real & a, const real & b) { return a>b?a:b; }
//! Returns the smaller value of two real values
inline real min(const real & a, const real & b) { return a<b?a:b; }
//! Returns the greater value of two real values (for Compatibility)
inline real Max(const real & a, const real & b) { return a>b?a:b; }
(...)
```

Durch ausdrückliche Vorgaben in der Dokumentations-Richtlinie wurde versucht, dieses Problem zu lösen. Es wurde festgelegt, dass in den Header-Dateien nur die Deklarationen und in den Code-Dateien (`*.cpp` und `*.inl`) der zugehörige Quelltext eingebunden wird.

Wo diese Fehler aktuell allerdings schon vorhanden sind, wurden entsprechende Hinweise an die Entwickler weitergegeben, damit die Quelltexte entsprechend angepasst bzw. überarbeitet werden können. Die „Fehler“ wurden nicht direkt behoben, da Änderungen am Quelltext nur von den Entwicklern selbst vorgenommen werden dürfen, weil diese eventuell auftretende Wechselwirkungen mit anderen Quelltext-Teilen besser einschätzen und auch beheben können.

Zudem wurde in diesem Zusammenhang für die Dokumentation festgelegt, dass in den Header-Dateien in der Regel die Klassenbeschreibung samt der Kurzbeschreibungen zu den einzelnen Methoden aufgeführt sein soll. Die ausführlicheren Methodenbeschreibungen hingegen finden sich in den jeweiligen Quelltext-Dateien, die die Implementation der zu beschreibenden Methode beinhaltet.

3.5.2 Mathematische Formeln

Ein weiteres Problem – das von Beginn an zu erwarten war, da C-XSC eine Klassen-Bibliothek mit mathematischen Zielen ist – ist die Formulierung von Formeln in den Dokumentationskommentaren. Doxygen bietet die Notation in LaTeX-Schreibweise an, sodass eine Einarbeitung in diese Notation notwendig wurde. Hierzu wurden Beispiele aus der Doxygen-

Dokumentation [4], als auch die Beschreibungen und Anleitungen aus den Dokumenten [5] und [6] genutzt.

Von Beginn an bietet Doxygen zwei verschiedene Arten an, wie Formeln in der Dokumentation eingebunden werden können. Als Erstes können sie einfach fließend in den Text eingefügt werden:

```
//! The Gauss error function \f$ \mbox{erf}(x) = \frac{2}{\sqrt{\pi}} \int
\limits_0^x e^{-t^2} dt \f$
inline real erf    (const real&);
```

Merkmal sind hierbei die Beginn- und Ende-Kennzeichnung des mathematischen Ausdrucks mit den Zeichen `\f$`. Zwischen diesen beiden Markierungen können nun beliebig Formelausdrücke in LaTeX-Schreibweise eingefügt werden, die später von Doxygen beziehungsweise LaTeX in eine Grafik umgesetzt werden, die wiederum an dieser Stelle der Dokumentation eingefügt wird. Bildlich stellt sich dies wie in Abbildung 9 dar.

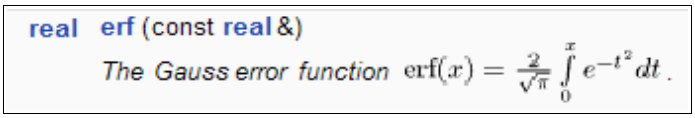


Abbildung 9: Im Fließtext integrierte Formel

Die zweite Art eine mathematische Formel einzubinden unterscheidet sich prinzipiell nicht von der Vorherigen. Es werden lediglich andere Beginn- (`\f[`) und Ende-Kennzeichnungen (`\f]`) verwendet:

```
//!The Scalar Type interval
/*!
The data type interval is used to represent intervals over the real
floating-point numbers:

\f[
\left[ a \right] = \left[ \underline a , \overline a \right] := \left\{ x
\in R | \underline a \le x \le \overline a \right\} \in R
\f]

(...)
*/
```

Merkmal dieser Notation ist, dass hierbei die Formel nicht mehr in den fließenden Text eingebunden wird, sondern zentriert in einer neuen Zeile, sodass sie bewusst gegenüber dem sonstigen Text hervorgehoben wird, wie in Abbildung 10 dargestellt.

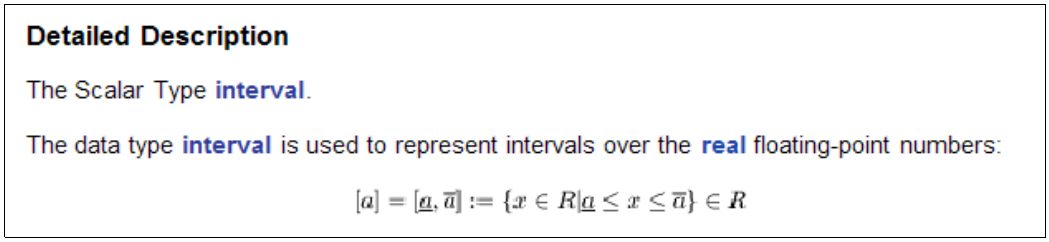


Abbildung 10: Hervorgehobene mathematische Formel

3.5.3 Überfrachtung

Zu Beginn dieser Ausarbeitung fiel auch auf, dass die einfache Inkludierung aller Quelltext-Dateien der Bibliothek zu einem weiteren Problem führte: Die Dokumentation wurde in diesem Stadium so umfangreich, dass eine sinnvolle Arbeit mit der Dokumentation nicht möglich war. Zum Beispiel wurden viele Methoden, auf die der Anwender der Bibliothek im Regelfall gar keinen Zugriff hat, in die Dokumentation aufgenommen. Ebenso wurden auch Klassen und Methoden eingebunden, die für die Anwendung der Bibliothek gar keine Notwendigkeit darstellen, wie z. B. Test-Klassen und -Methoden, um die ordnungsgemäße Funktion der Bibliothek zu testen.

Dieser Fehler wurde durch einen Datei-Filter in der Konfigurationsdatei der Dokumentationsanwendung Doxygen behoben, sodass die entsprechenden Dateien von der automatischen Dokumentierung ausgeschlossen wurden. Dadurch finden sie sich natürlich nicht mehr in der Dokumentation wieder.

3.5.4 Fehlende Kommentare in erstellter Dokumentation

Trotz aller Sorgfalt während der Dokumentation von C-XSC fehlen in der finalen Version der erstellten Dokumentation zu einigen Methoden die Kurzkommentare. Dies betrifft allerdings nur den Bereich „Namespace Reference“. Werden die jeweiligen Methoden in den Quellen betrachtet, wurden die entsprechenden Kommentare eingefügt. Aus welchem Grunde Doxygen die Kommentare nicht in die Dokumentation übernimmt, konnte der Autor im Rahmen dieser Ausarbeitung trotz Kontakt mit der Benutzer-Mailing-Liste[19] nicht herausfinden. Es besteht allerdings die Vermutung, dass die betroffenen Methoden nicht klar beziehungsweise eindeutig deklariert sind, sodass Doxygen eventuell keine eindeutige Zuordnung der Kommentare zu der entsprechenden Methode für die Dokumentation vornehmen konnte.

3.5.5 Template-Methoden in Dokumentation

Ein noch nicht gelöstes Problem ist das Auftauchen von Template-Methoden in der Dokumentation. Template-Methoden bilden eine Art Schablone für später konkret ausgebildete Methoden.

Leider konnte nicht festgestellt werden, welche Datei oder Dateien dafür verantwortlich sind, dass teilweise auch Template-Methoden in der Dokumentation erscheinen. Die vermutliche Ursache ist, dass durch den häufigen Wechsel zwischen Public- und Private-Bereichen in den

19 An-/Abmeldung: https://lists.sourceforge.net/lists/listinfo/doxygen-users (Stand: 12.05.2007)
Archiv:http://sourceforge.net/mailarchive/forum.php?forum_name=doxygen-users (Stand: 12.05.2007)

Klassendefinitionen im Quelltext auch die eigentlich dem Private-Bereich zuzuordnenden Template-Methoden in einem oder mehreren Public-Bereichen erscheinen und dadurch auch von Doxygen erfasst und dokumentiert werden. An dieser Stelle sollten daher die Entwickler ansetzen und den Quelltext überarbeiten und ihn systematisch in einen Public- und einen Private-Bereiche pro Klassendefinition aufteilen, sodass diese Fehlerquelle ausgeschlossen werden kann.

3.5.6 Nicht gelöste Problemmeldungen

In der Datei `doxygen_warnings.txt` werden die Fehler, die während eines Durchlaufs von Doxygen passieren oder auffallen, gesammelt und gespeichert. Nach Fertigstellung der Dokumentation sind noch eine Reihe von Meldungen dort enthalten, die sich mit der Duplizität von Kommentaren beschäftigen. Vielfach hat der Autor Methoden doppelt kommentiert, weil sie in verschiedenen Header-Dateien mehrfach vorkommen. An dieser Stelle müssen nun allerdings die Entwickler den Quelltext überarbeiten oder korrigieren, da der Autor Änderungen beziehungsweise Löschungen von doppelten Deklarationen im Quelltext nicht vornehmen durfte. Diese Meldungen sollten also nach einer Überarbeitung der Klassen-Bibliothek in Zukunft ausgeräumt sein.

Weitere Fehlermeldungen gibt es noch bezüglich Problemen bei der automatischen Diagramm-Erstellung mithilfe des Werkzeugs `dot`[20], das von Doxygen für diesen Zweck genutzt wird. Auch hier kann der Autor keine Lösung anbieten, da das Erforschen der Zusammenhänge zwischen den beiden Werkzeugen den Zeitrahmen dieser Bachelor-Thesis überstrapaziert hätte. Da dieses Problem auch „nur“ drei Diagramme (Anzahl laut Fehlermeldungen) betrifft, hatte dieser Fehler auch keine hohe Priorität bei der Erstellung der Dokumentation erlangt.

20 Enthalten im Programmpaket GraphViz http://www.graphviz.org/ (Stand: 12.05.2007)

4 Zusammenfassung und Ausblick

Entsprechend der im Rahmen dieser Bachelor-Thesis beschriebenen Vorgehensweise ist eine aktuelle Dokumentation für die Klassen-Bibliothek C-XSC entstanden. Zusätzlich wurde eine Dokumentations-Richtlinie für neue Quelltext-Teile angefertigt, an der sich die Entwickler für die Dokumentation dieser neuen Bestandteile orientieren können.

Ein Resultat aus dieser Ausarbeitung für künftige Projekte ist, dass sich Entwickler schon so frühzeitig wie möglich um so essenzielle Dinge wie die Erstellung einer Dokumentation kümmern sollten. So wird die Notwendigkeit einer nachträglichen kompletten Überarbeitung oder Neuerstellung der Dokumentation stark verringert, weil beständig während der Entwicklung auch die Dokumentation aktualisiert und erweitert wird.

Die hier beschriebene Dokumentation innerhalb des Quelltextes erfordert zudem keine besonderen Kenntnisse bei der Erstellung der Dokumentation (z. B. Layout, HTML, Latex, etc.), da genau dieser Teil der Dokumentations-Erstellung in den Aufgabenbereich der Dokumentations-Anwendung verschoben wird und dort in der Regel auch einfacher und zentraler zu gestalten ist. Zudem werden die Entwickler in Zukunft durch die erstellte Dokumentations-Richtlinie in dieser Hinsicht unterstützt, da aus den dort definierten Richtlinien schon ein standardisiertes Grundgerüst für neue Dokumentationsinhalte vorgegeben ist.

Ein weiterer Punkt, der bei einer anstehenden Überarbeitung der Dokumentation angegangen werden sollte, ist die Inkludierung der Dokumentations-Erstellung in das Makefile der Klassenbibliothek. Dadurch würde immer die für den entsprechenden Compilerlauf aktuelle Dokumentation mit erstellt werden und den Entwicklern in Zukunft ein weiterer Arbeitsschritt erspart bleiben. Außerdem wird die Qualität der gesamten Klassen-Bibliothek durch die aktuelle Dokumentation gesteigert.

Um die Dokumentation in Zukunft noch attraktiver zu gestalten, wäre auch die Überarbeitung beziehungsweise Anpassung der Layout-Vorlagen der Dokumentation angebracht. Im Rahmen dieser Ausarbeitung wurden die standardmäßigen Voreinstellungen, die das Programm Doxygen mitbringt, genutzt. Diese sind aber sehr flexibel an neue Anforderungen anpassbar (durch Verwendung von z. B. Cascading Stylesheets). Dadurch ist es möglich, der Dokumentation ein individuelles Äußeres zu geben, sodass sie sich von anderen mit Doxygen erstellten Dokumentationen abheben kann. Weil die Inhalte aber auch das Äußere der Dokumentation bei der Auswahl geeigneter Werkzeuge und Hilfsmittel durch potenzielle Anwender eine Rolle spielen, könnte sich C-XSC auf diese Weise auch Vorteile in der Verbreitung und Verwendung innerhalb anderer Projekte verschaffen.

5 Anhang

I Doxygen-Konfigurations-Datei

Nachfolgend die erstellte Konfigurationsdatei für das Programm Doxygen. Bis auf ein paar Ausnahmen zur Demonstration der Erläuterungskommentare einiger besonderer Optionen, wurden diese aus Längengründen entfernt.

```
# Doxyfile 1.5.2

# This file describes the settings to be used by the documentation system
# doxygen (www.doxygen.org) for a project
#
# All text after a hash (#) is considered a comment and will be ignored
# The format is:
#       TAG = value [value, ...]
# For lists items can also be appended using:
#       TAG += value [value, ...]
# Values that contain spaces should be placed between quotes (" ")

#---------------------------------------------------------------------------
# Project related configuration options
#---------------------------------------------------------------------------

# This tag specifies the encoding used for all characters in the config file that
# follow. The default is UTF-8 which is also the encoding used for all text before
# the first occurrence of this tag. Doxygen uses libiconv (or the iconv built into
# libc) for the transcoding. See http://www.gnu.org/software/libiconv for the list of
# possible encodings.

DOXYFILE_ENCODING      = UTF-8

# The PROJECT_NAME tag is a single word (or a sequence of words surrounded
# by quotes) that should identify the project.

PROJECT_NAME           = "C-XSC - A C++ Class Library for Extended Scientific Computing"

# The PROJECT_NUMBER tag can be used to enter a project or revision number.
# This could be handy for archiving the generated documentation or
# if some version control system is used.

PROJECT_NUMBER         = 2.2.1

# The OUTPUT_DIRECTORY tag is used to specify the (relative or absolute)
# base path where the generated documentation will be put.
# If a relative path is entered, it will be relative to the location
# where doxygen was started. If left blank the current directory will be used.

OUTPUT_DIRECTORY       = ../docu/apidoc/
CREATE_SUBDIRS         = NO
OUTPUT_LANGUAGE        = English
BRIEF_MEMBER_DESC      = YES
REPEAT_BRIEF           = YES
ABBREVIATE_BRIEF       =
ALWAYS_DETAILED_SEC    = NO
INLINE_INHERITED_MEMB  = NO
FULL_PATH_NAMES        = NO
STRIP_FROM_PATH        =
STRIP_FROM_INC_PATH    =
SHORT_NAMES            = NO
JAVADOC_AUTOBRIEF      = NO
MULTILINE_CPP_IS_BRIEF = NO
DETAILS_AT_TOP         = NO
INHERIT_DOCS           = YES
SEPARATE_MEMBER_PAGES  = NO
TAB_SIZE               = 8
```

```
ALIASES                =
OPTIMIZE_OUTPUT_FOR_C  = NO
OPTIMIZE_OUTPUT_JAVA   = NO
BUILTIN_STL_SUPPORT    = NO
CPP_CLI_SUPPORT        = NO
DISTRIBUTE_GROUP_DOC   = YES
SUBGROUPING            = YES

#---------------------------------------------------------------------------
# Build related configuration options
#---------------------------------------------------------------------------

# If the EXTRACT_ALL tag is set to YES doxygen will assume all entities in
# documentation are documented, even if no documentation was available.
# Private class members and static file members will be hidden unless
# the EXTRACT_PRIVATE and EXTRACT_STATIC tags are set to YES

EXTRACT_ALL            = NO

# If the EXTRACT_PRIVATE tag is set to YES all private members of a class
# will be included in the documentation.

EXTRACT_PRIVATE        = NO

# If the EXTRACT_STATIC tag is set to YES all static members of a file
# will be included in the documentation.

EXTRACT_STATIC         = NO

# If the EXTRACT_LOCAL_CLASSES tag is set to YES classes (and structs)
# defined locally in source files will be included in the documentation.
# If set to NO only classes defined in header files are included.

EXTRACT_LOCAL_CLASSES  = NO

# This flag is only useful for Objective-C code. When set to YES local
# methods, which are defined in the implementation section but not in
# the interface are included in the documentation.
# If set to NO (the default) only methods in the interface are included.

EXTRACT_LOCAL_METHODS  = NO

# If the HIDE_UNDOC_MEMBERS tag is set to YES, Doxygen will hide all
# undocumented members of documented classes, files or namespaces.
# If set to NO (the default) these members will be included in the
# various overviews, but no documentation section is generated.
# This option has no effect if EXTRACT_ALL is enabled.

HIDE_UNDOC_MEMBERS     = YES

# If the HIDE_UNDOC_CLASSES tag is set to YES, Doxygen will hide all
# undocumented classes that are normally visible in the class hierarchy.
# If set to NO (the default) these classes will be included in the various
# overviews. This option has no effect if EXTRACT_ALL is enabled.

HIDE_UNDOC_CLASSES     = YES
HIDE_FRIEND_COMPOUNDS  = NO
HIDE_IN_BODY_DOCS      = NO
INTERNAL_DOCS          = NO
CASE_SENSE_NAMES       = NO
HIDE_SCOPE_NAMES       = NO
SHOW_INCLUDE_FILES     = YES
INLINE_INFO            = YES
SORT_MEMBER_DOCS       = YES
SORT_BRIEF_DOCS        = YES
SORT_BY_SCOPE_NAME     = NO
GENERATE_TODOLIST      = YES
```

```
GENERATE_TESTLIST      = YES
GENERATE_BUGLIST       = YES
GENERATE_DEPRECATEDLIST= YES
ENABLED_SECTIONS       =
MAX_INITIALIZER_LINES  = 40
SHOW_USED_FILES        = YES
SHOW_DIRECTORIES       = NO
FILE_VERSION_FILTER    =

#---------------------------------------------------------------------------
# configuration options related to warning and progress messages
#---------------------------------------------------------------------------

QUIET                  = NO
WARNINGS               = YES
WARN_IF_UNDOCUMENTED   = NO
WARN_IF_DOC_ERROR      = YES
WARN_NO_PARAMDOC       = YES
WARN_FORMAT            = "$file:$line: $text"
WARN_LOGFILE           = ../doxygen_warnings.txt

#---------------------------------------------------------------------------
# configuration options related to the input files
#---------------------------------------------------------------------------

# The INPUT tag can be used to specify the files and/or directories that contain
# documented source files. You may enter file names like "myfile.cpp" or
# directories like "/usr/src/myproject". Separate the files or directories
# with spaces.

INPUT                  = ./
INPUT_ENCODING         = UTF-8

# If the value of the INPUT tag contains directories, you can use the
# FILE_PATTERNS tag to specify one or more wildcard pattern (like *.cpp
# and *.h) to filter out the source-files in the directories. If left
# blank the following patterns are tested:
# *.c *.cc *.cxx *.cpp *.c++ *.java *.ii *.ixx *.ipp *.i++ *.inl *.h *.hh *.hxx
# *.hpp *.h++ *.idl *.odl *.cs *.php *.php3 *.inc *.m *.mm *.py

FILE_PATTERNS          = *.hpp \
                         *.cpp \
                         *.inl
RECURSIVE              = NO
EXCLUDE                =
EXCLUDE_SYMLINKS       = NO

# If the value of the INPUT tag contains directories, you can use the
# EXCLUDE_PATTERNS tag to specify one or more wildcard patterns to exclude
# certain files from those directories. Note that the wildcards are matched
# against the file with absolute path, so to exclude all test directories
# for example use the pattern */test/*

EXCLUDE_PATTERNS       = except.* \
                         *test* \
                         ioflags.* \
                         cxscmatr.* \
                         cxscvect.* \
                         matrix.* \
                         old*
EXCLUDE_SYMBOLS        =

# The EXAMPLE_PATH tag can be used to specify one or more files or
# directories that contain example code fragments that are included (see
# the \include command).

EXAMPLE_PATH           = ../examples/ \
```

```
                         ../CToolbox/
EXAMPLE_PATTERNS       =
EXAMPLE_RECURSIVE      = YES

# The IMAGE_PATH tag can be used to specify one or more files or
# directories that contain image that are included in the documentation (see
# the \image command).

IMAGE_PATH             = ../docu/images/
INPUT_FILTER           =
FILTER_PATTERNS        =
FILTER_SOURCE_FILES    = NO

#---------------------------------------------------------------------------
# configuration options related to source browsing
#---------------------------------------------------------------------------

SOURCE_BROWSER         = YES
INLINE_SOURCES         = NO
STRIP_CODE_COMMENTS    = YES
REFERENCED_BY_RELATION = YES
REFERENCES_RELATION    = NO
REFERENCES_LINK_SOURCE = YES
USE_HTAGS              = NO
VERBATIM_HEADERS       = YES

#---------------------------------------------------------------------------
# configuration options related to the alphabetical class index
#---------------------------------------------------------------------------

ALPHABETICAL_INDEX     = YES
COLS_IN_ALPHA_INDEX    = 5
IGNORE_PREFIX          =

#---------------------------------------------------------------------------
# configuration options related to the HTML output
#---------------------------------------------------------------------------

# If the GENERATE_HTML tag is set to YES (the default) Doxygen will
# generate HTML output.

GENERATE_HTML          = YES
HTML_OUTPUT            = html
HTML_FILE_EXTENSION    = .html
HTML_HEADER            =
HTML_FOOTER            =
HTML_STYLESHEET        =
HTML_ALIGN_MEMBERS     = YES
GENERATE_HTMLHELP      = NO
CHM_FILE               =
HHC_LOCATION           =
GENERATE_CHI           = NO
BINARY_TOC             = NO
TOC_EXPAND             = NO
DISABLE_INDEX          = NO
ENUM_VALUES_PER_LINE   = 4
GENERATE_TREEVIEW      = YES
TREEVIEW_WIDTH         = 250

#---------------------------------------------------------------------------
# configuration options related to the LaTeX output
#---------------------------------------------------------------------------

# If the GENERATE_LATEX tag is set to YES (the default) Doxygen will
# generate Latex output.

GENERATE_LATEX         = NO
```

```
LATEX_OUTPUT           = latex
LATEX_CMD_NAME         = latex
MAKEINDEX_CMD_NAME     = makeindex
COMPACT_LATEX          = NO
PAPER_TYPE             = a4wide
EXTRA_PACKAGES         =
LATEX_HEADER           =
PDF_HYPERLINKS         = YES
USE_PDFLATEX           = YES
LATEX_BATCHMODE        = NO
LATEX_HIDE_INDICES     = NO

#---------------------------------------------------------------------------
# configuration options related to the RTF output
#---------------------------------------------------------------------------

GENERATE_RTF           = NO
RTF_OUTPUT             = rtf
COMPACT_RTF            = NO
RTF_HYPERLINKS         = NO
RTF_STYLESHEET_FILE    =
RTF_EXTENSIONS_FILE    =

#---------------------------------------------------------------------------
# configuration options related to the man page output
#---------------------------------------------------------------------------

GENERATE_MAN           = NO
MAN_OUTPUT             = man
MAN_EXTENSION          = .3
MAN_LINKS              = NO

#---------------------------------------------------------------------------
# configuration options related to the XML output
#---------------------------------------------------------------------------

GENERATE_XML           = NO
XML_OUTPUT             = xml
XML_SCHEMA             =
XML_DTD                =
XML_PROGRAMLISTING     = YES

#---------------------------------------------------------------------------
# configuration options for the AutoGen Definitions output
#---------------------------------------------------------------------------

GENERATE_AUTOGEN_DEF   = NO

#---------------------------------------------------------------------------
# configuration options related to the Perl module output
#---------------------------------------------------------------------------

GENERATE_PERLMOD       = NO
PERLMOD_LATEX          = NO
PERLMOD_PRETTY         = YES
PERLMOD_MAKEVAR_PREFIX =

#---------------------------------------------------------------------------
# Configuration options related to the preprocessor
#---------------------------------------------------------------------------

ENABLE_PREPROCESSING   = YES
MACRO_EXPANSION        = YES
EXPAND_ONLY_PREDEF     = NO
SEARCH_INCLUDES        = YES
INCLUDE_PATH           =
INCLUDE_FILE_PATTERNS  =
```

```
PREDEFINED             =
EXPAND_AS_DEFINED      =
SKIP_FUNCTION_MACROS   = YES

#---------------------------------------------------------------------------
# Configuration::additions related to external references
#---------------------------------------------------------------------------

TAGFILES               =
GENERATE_TAGFILE       =
ALLEXTERNALS           = NO
EXTERNAL_GROUPS        = YES
PERL_PATH              = /usr/bin/perl

#---------------------------------------------------------------------------
# Configuration options related to the dot tool
#---------------------------------------------------------------------------

CLASS_DIAGRAMS         = YES
MSCGEN_PATH            =
HIDE_UNDOC_RELATIONS   = YES
HAVE_DOT               = YES
CLASS_GRAPH            = YES
COLLABORATION_GRAPH    = YES
GROUP_GRAPHS           = YES
UML_LOOK               = YES
TEMPLATE_RELATIONS     = NO
INCLUDE_GRAPH          = NO
INCLUDED_BY_GRAPH      = NO
CALL_GRAPH             = NO
CALLER_GRAPH           = NO
GRAPHICAL_HIERARCHY    = NO
DIRECTORY_GRAPH        = NO
DOT_IMAGE_FORMAT       = png
DOT_PATH               =
DOTFILE_DIRS           =
DOT_GRAPH_MAX_NODES    = 50
DOT_TRANSPARENT        = NO
DOT_MULTI_TARGETS      = YES
GENERATE_LEGEND        = YES
DOT_CLEANUP            = YES

#---------------------------------------------------------------------------
# Configuration::additions related to the search engine
#---------------------------------------------------------------------------

SEARCHENGINE           = NO
```

II Auszug aus erstellter Online-Dokumentation

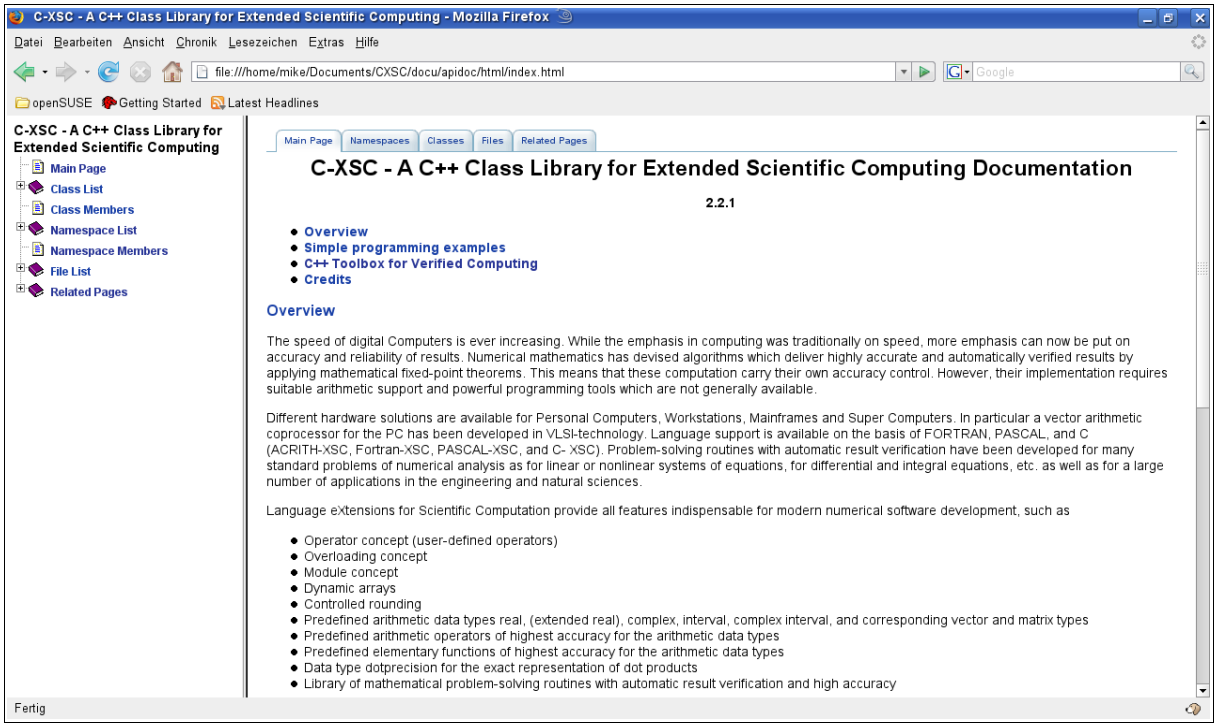


Abbildung 11: Startseite der Dokumentation

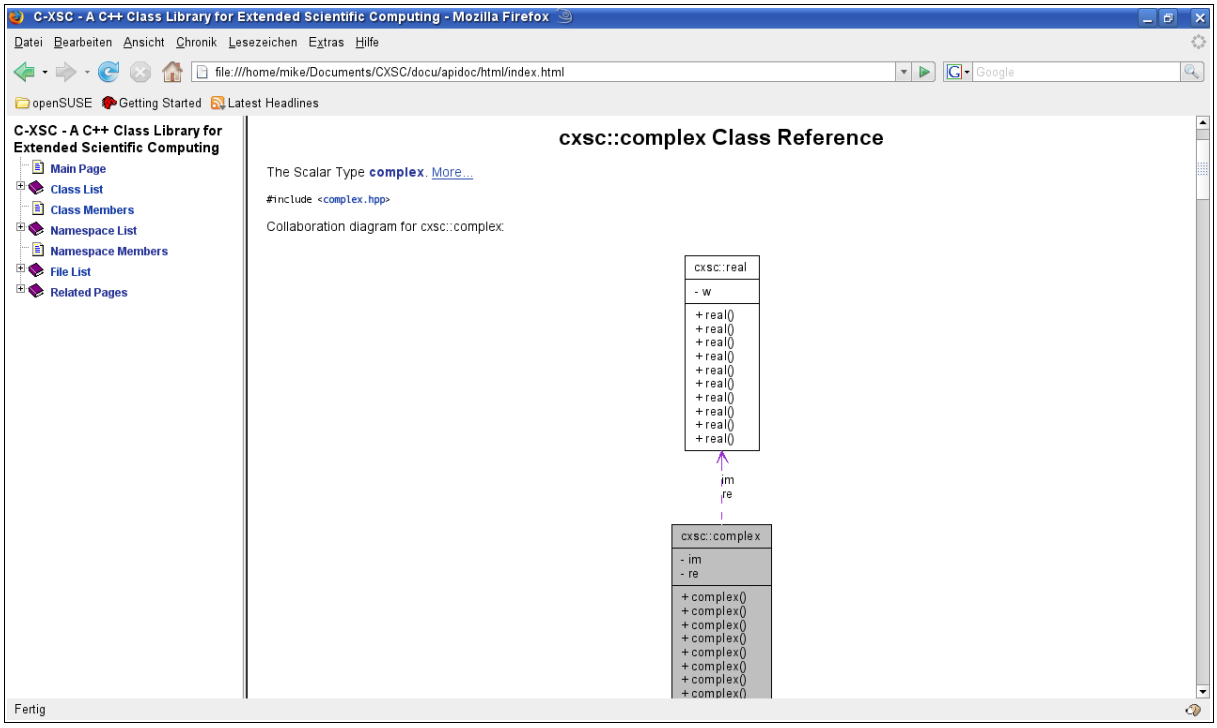


Abbildung 12: Beginn einer Klassenbeschreibung

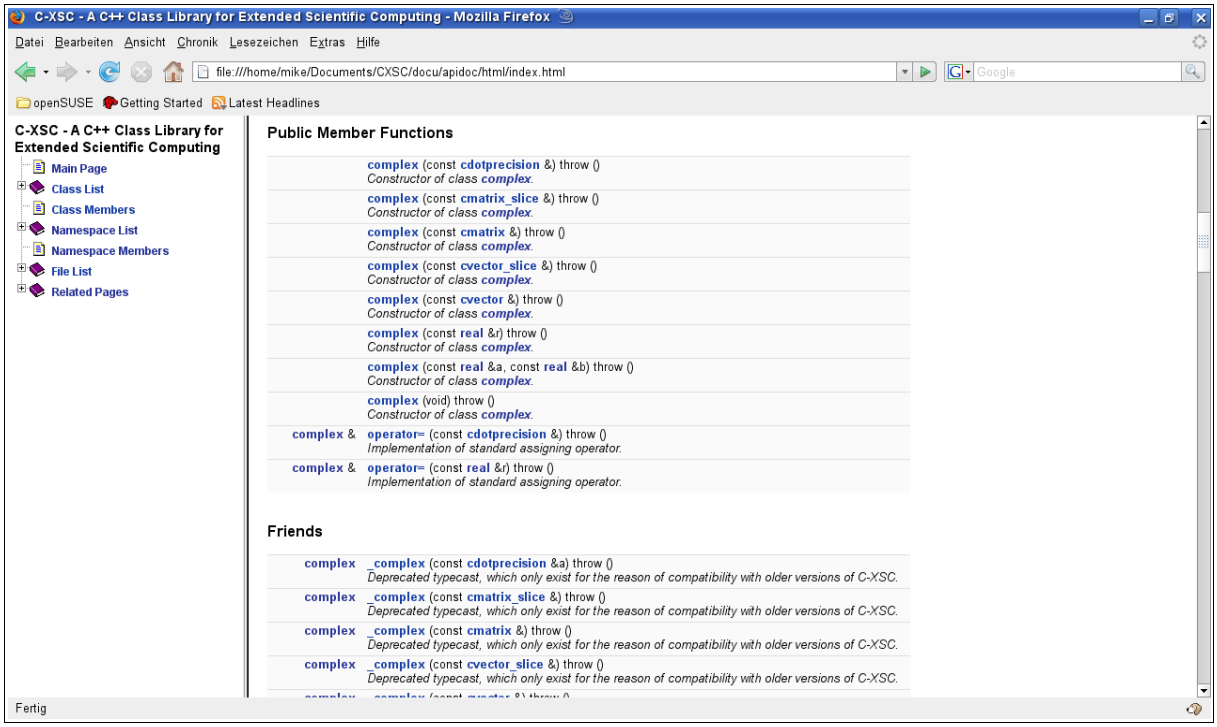


Abbildung 13: Kurzbeschreibungen von Konstruktoren und Methoden

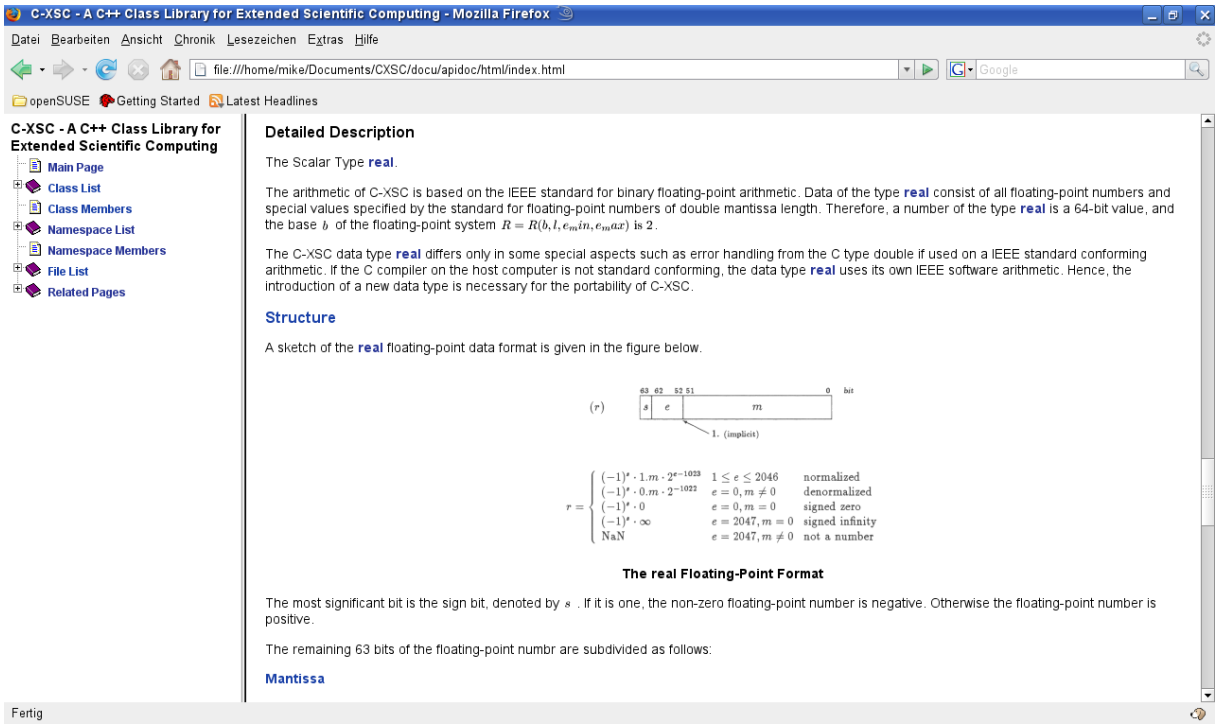


Abbildung 14: Detaillierte Klassenbeschreibung

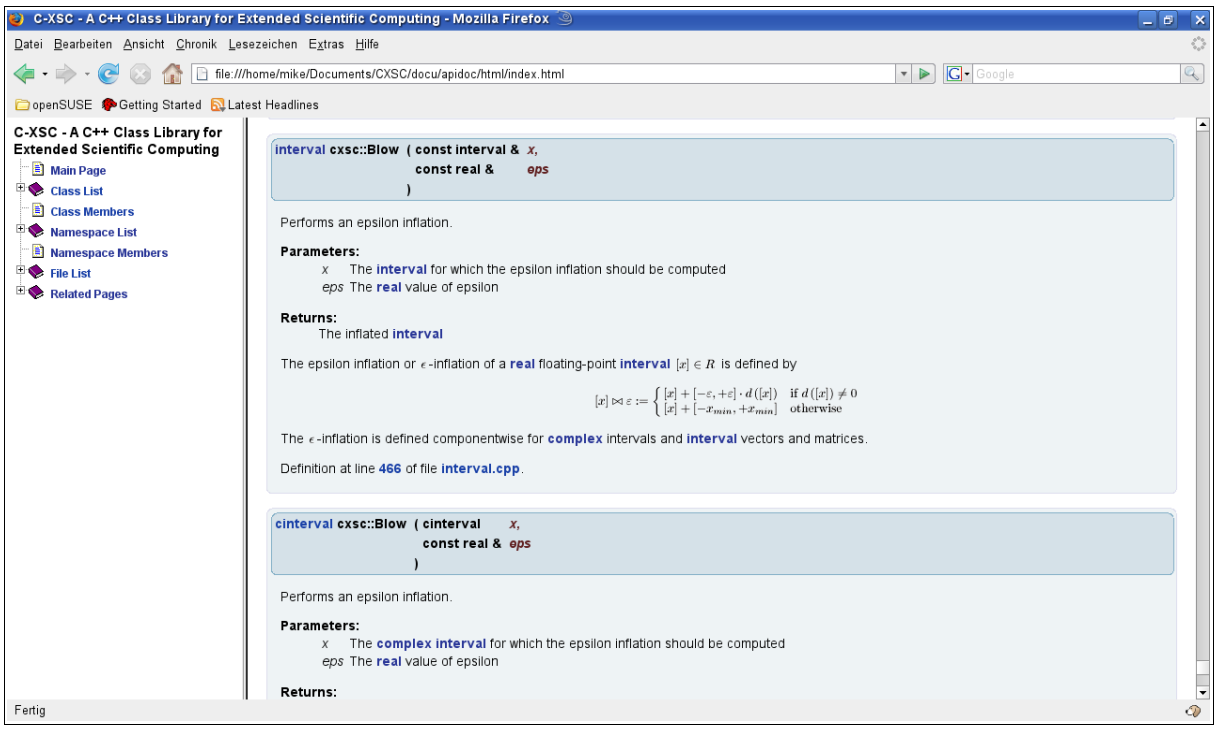


Abbildung 15: Detaillierte Methodenbeschreibung

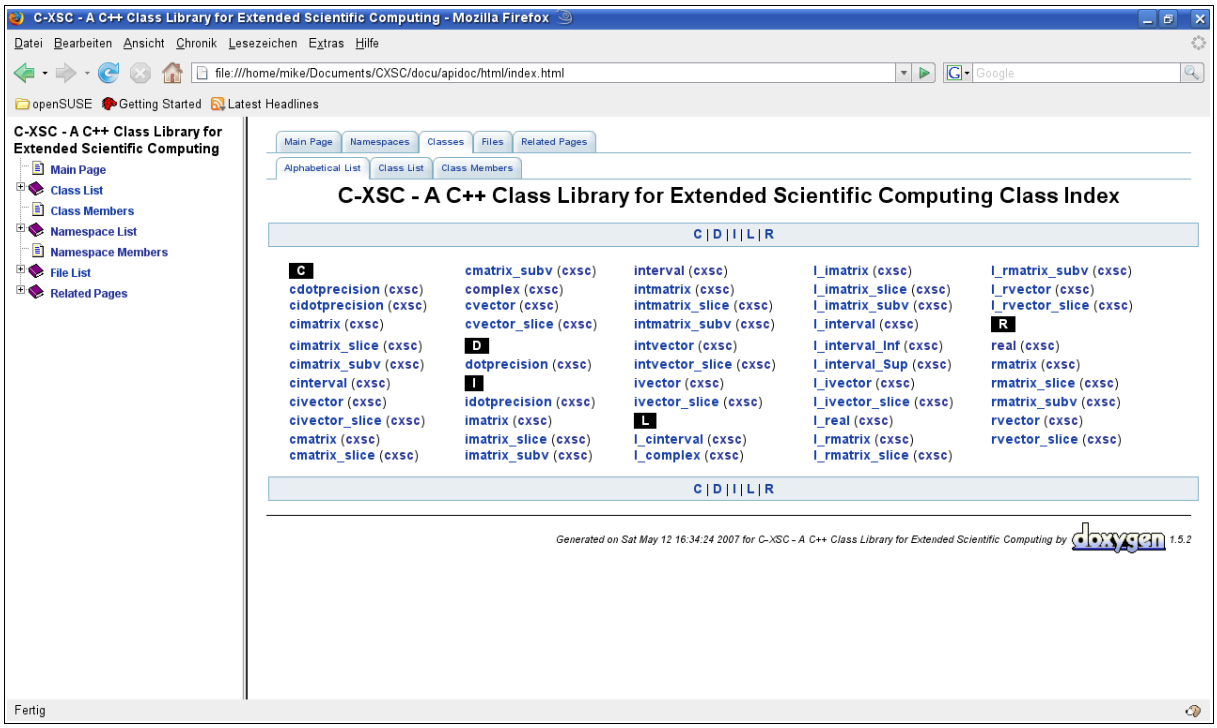


Abbildung 16: Übersicht über alle Klassen

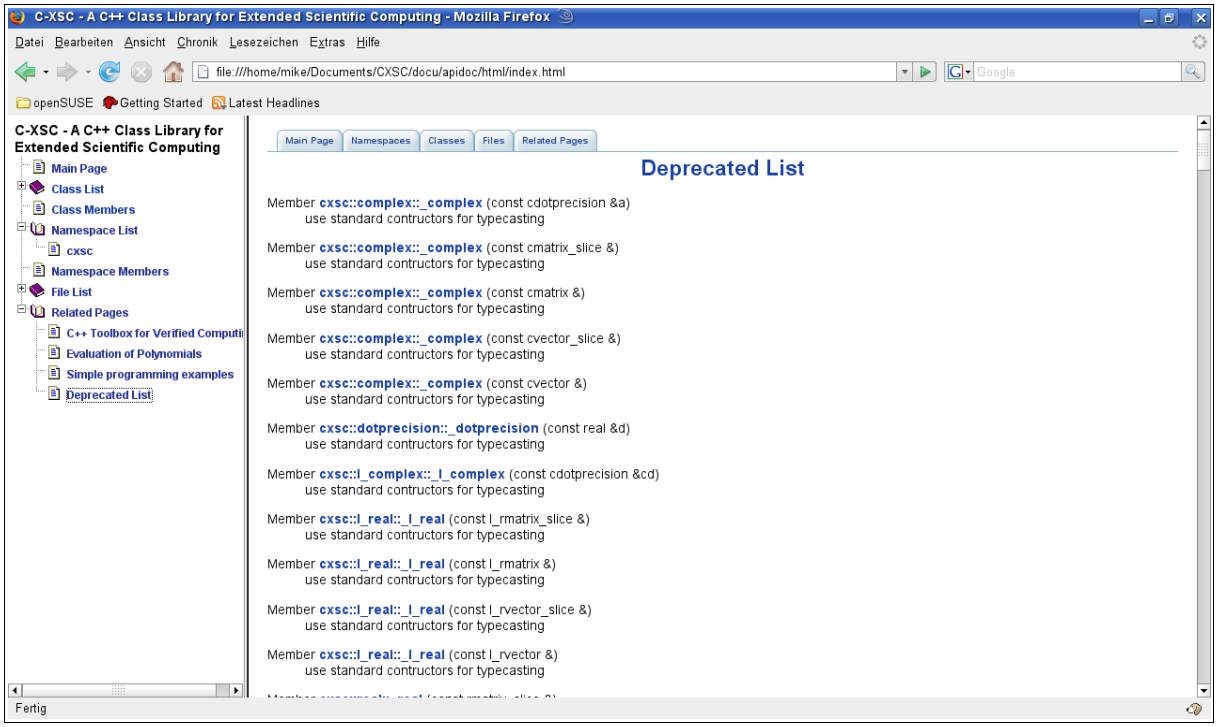


Abbildung 17: Auflistung aller veralteten Methoden

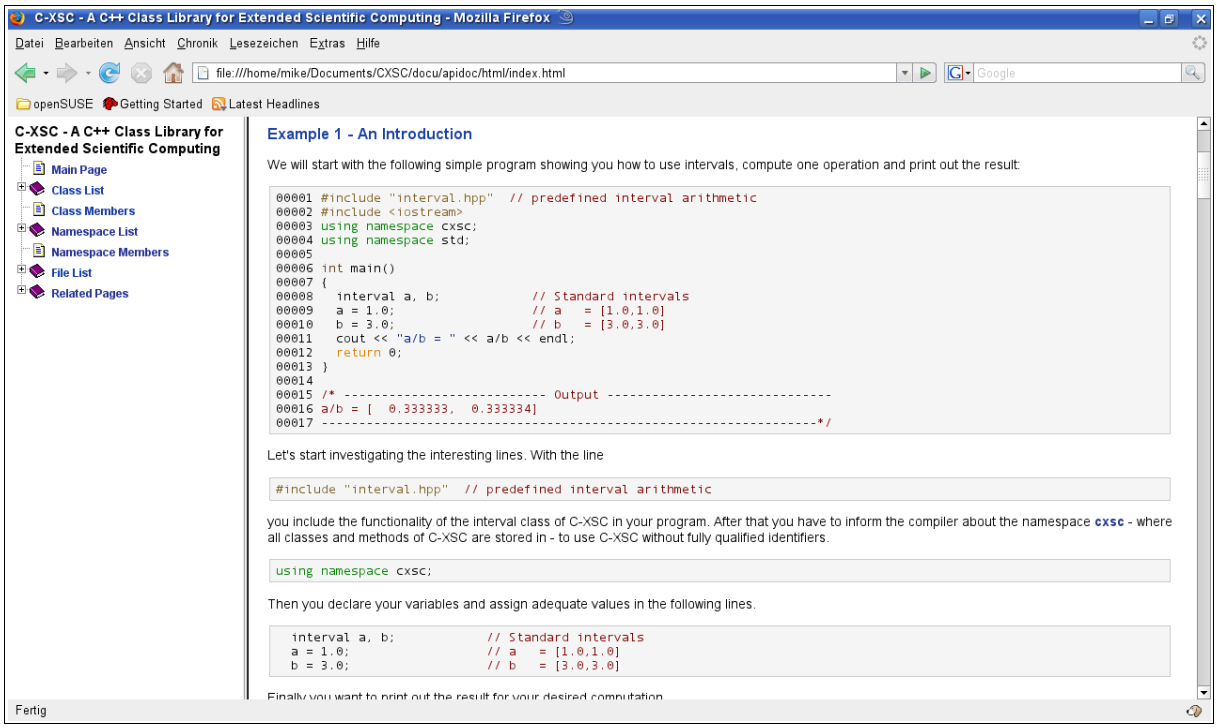


Abbildung 18: Quelltext-Dokumentierung

III Literaturverzeichnis

[1] R. HAMMER, M. HOCKS,U. KULISCH, D. RATZ, *C++ Toolbox for Verified Computing*, 1995, Springer-Verlag Berlin Heidelberg 1995

[2] R. KLATTE, U. KULISCH, A. WIETHOFF, C. LAWO, M. RAUSCH, *C-XSC - A C++ Class Library for Extended Scientific Computing*, 1993, Springer-Verlag Berlin Heidelberg 1993

[3] W. HOFSCHUSTER, W. KRÄMER, *C-XSC 2.0: A C++ Library for Extended Scientific Computing* aus *Numerical Software with Result Verification*, Lecture Notes in Computer Science Nummer 2991/2004, Springer-Verlag Berlin Heidelberg 2004, Seiten 15 - 35

[4] D. VAN HEESCH, *Doxygen Manual*, `http://www.stack.nl/~dimitri/doxygen/manual.html` (Stand: 28.02.2007)

[5] M. JÜRGENS, *LATEX — eine Einführung und ein bisschen mehr*, `ftp://ftp.fernuni-hagen.de/pub/pdf/urz-broschueren/broschueren/a0260003.pdf` (Stand: 12.03.2007)

[6] M. JÜRGENS, *LATEX - Fortgeschrittene Anwendungen*, `ftp://ftp.fernuni-hagen.de/pub/pdf/urz-broschueren/broschueren/a0279510.pdf` (Stand: 12.03.2007)

IV Abbildungsverzeichnis

V Tabellenverzeichnis