Hochschule Wismar
University of Technology, Business and Design

Diplomarbeit

Entwicklung einer automatisierten Software zur Darstellung des Inhalts einer meteorologischen Datenbank

Rachholz, Dirk

geb. am 28. 09. 1979 in Kühlungsborn

Studiengang Multimediatechnik

Hochschule Wismar - University of Technology, Business and Design

Fachbereich Elektrotechnik und Informatik

Betreuer, Einrichtung:	Prof. Dr. S. Pawletta, FH Wismar
	Dr. U. Berger, Leibniz-Institut für Atmosphärenphysik e.V., Kühlungsborn
	Dr. G. Baumgarten, Leibniz-Institut für Atmosphärenphysik e.V., Kühlungsborn
Abgabetermin:	03. 10. 2006

Aufgabenstellung

Entwicklung einer automatisierten Software zur Darstellung des Inhalts einer meteorologischen Datenbank

Die Datenbank enthält binäre Dateien der meteorologischen Grundgrößen Temperatur, Wind, Dichte, Druck, Wasserdampf und Ozon, die global, dreidimensional und zeitabhängig definiert sind.

- **Einzelbilderstellung:**
  Es sollen Skripte programmiert werden, die beliebige Dateien der meteorologischen Datenbank in einer standardisierten Form visualisieren. Diese Bilddateien sollen in einer vordefinierten Verzeichnisstruktur abgelegt werden. Die Einzelbilderstellung soll verschiedenste Darstellungen umfassen, die an die wissenschaftlichen Fragestellungen angepasst sind.

- **Filmerstellung:**
  Zu ausgewählten Einzelbildformaten sollen Filme erstellt werden. Zum Beispiel wird eine Filmversion angestrebt, die ähnlich zum Strömungsfilm des ARD Wetterberichts aufgebaut ist. Des Weiteren soll ein Anwenderprogramm entwickelt werden, das die Animation von zweidimensionalen atmosphärischen Gezeiten mit beliebig gewählten Eigenschaften erlaubt.

Die Diplomarbeit wird in Kooperation mit dem Leibniz-Institut für Atmosphärenphysik Kühlungsborn durchgeführt.

Tag der Ausgabe: 03.07.2006
Tag der Abgabe : 03.10.2006

Autorenreferat

In der Optikabteilung des Instituts für Atmosphärenphysik in Kühlungsborn (IAP), wird seit 2005 das atmosphärische Modell LIMA (Leibniz-Institut Middle Atmosphere Model) benutzt, um Prozesse in der Atmosphäre besser erforschen zu können. In der vorliegenden Arbeit wurden Methoden zur automatisierten Visualisierung der LIMA-Ergebnisdaten für verschiedene Problemstellungen untersucht und zum Teil entwickelt. Die Untersuchungen wurden unter Berücksichtigung der dem IAP zur Verfügung stehenden Rechnersysteme und Software durchgeführt. Weiterhin wurde eine Software entwickelt die Gezeitenwellen anhand von Eingabeparametern simuliert und visualisiert.

Abstract

The optic department of the Institut of Atmospheric Physics of Kühlungsborn (IAP) utilises a newly developed atmospheric model called LIMA (Leibniz-Institut Middle Atmosphere Model), in order to investigate physical processes of the atmosphere. In the following diploma work, I investigate different methods in which computational requirements of data visualizations were performed on computer facilities available at the IAP. Furthermore, a software simulating tidal waves was developed which allows a visualisation of the waves in an interactive way.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AVI	**A**udio**V**ideo**I**nterleave (Videoformat)
BMP	Windows Device Independent **B**it**M**a**P** (Grafikformat)
CMYK	**C**yan **M**agenta **Y**ellow **K**ey (Farbmodell)
ECMWF	**E**uropean **C**enter for **M**edium range **W**eather **F**orecast
DMF	**D**ata **M**igration **F**ilesystem
EPS	**E**ncapsulated **P**ost **S**cript (Grafikformat)
GCM	**G**eneral **C**irculation **M**odel
GUI	**G**raphical **U**ser **I**nterface
IAP	Leibniz-**I**nstitut für **A**tmosphären**P**hysik e.V. Kühlungsborn
IDL	**I**nteractive **D**ata **L**anguage (Research Systems, Inc.)
JPEG	**J**oint **P**hotographic **E**xpert **G**roup (Grafikformat)
LIMA	**L**eibniz-**I**nstitut **M**iddle **A**tmosphere Model
LZ77	**L**empel-**Z**if-19**77** Kompressionsverfahren
LZW	**L**empel-**Z**if-**W**elch Kompressionsverfahren
MOV	QuickTime **MOV**ie (Videoformat)
MPEG	**M**oving **P**ictures **E**xpert **G**roup (Videoformat)
NLC	**N**octi**L**ucent **C**loud (leuchtende Nachtwolken)
PC	**P**ersonal **C**omputer
PMSE	**P**olar **M**esosphere **S**ummer **E**cho
PMWE	**P**olar **M**esosphere **W**inter **E**cho
PNG	**P**ortable **N**etwork **G**raphics (Grafikformat)
PPM	**P**ortable **P**ixel **M**ap (Grafikformat)
PS	**P**ost **S**cript (Grafikformat)
RGB	**R**ot **G**rün **B**lau (Farbmodell)
RISC	**R**educed **I**nstruction **S**et **C**omputer (Prozessorarchitektur)

RM	**R**eal **M**edia (Videoformat)
RLE	**R**un **L**ength **E**ncoding Kompressionsverfahren
SRF	**S**un **R**aster **F**ile (Grafikformat)
TIFF	**T**agged **I**mage **F**ile **F**ormat (Grafikformat)
UV	**U**ltra-**V**iolett

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die grafische Datenverarbeitung befasst sich mit der computergestützten Visualisierung numerischer Daten beliebiger Objekte (z.B. Wahlergebnisse, Umweltstatistiken, Simulationsergebnisse, etc.). Visualisierung beinhaltet also die Transformation von Daten oder Informationen in Bilder oder Filme. Dabei werden in der Regel umfangreiche numerische Datenbestände mit Hilfe grafischer Darstellungsmethoden (z.B. Bilder, Diagramme, Zeichnungen) so visualisiert, dass die in den Daten enthaltenen Informationen für den wissenschaftlichen Betrachter in verständliche und interpretierbare Darstellungen umgesetzt werden. Die Umwandlung der numerischen Datenbestände erfolgt dabei mit Programmen, welche die Eingangsdaten einlesen, für die gewünschte Darstellung bearbeiten und ausgeben, ohne dass der Anwender über die genauen Prozesse der Visualisierung informiert sein muss. Bei komplexen Themen wird dabei auf Programmiersprachen mit Grafikfunktionen zurückgegriffen, mit denen individuelle Grafiken erzeugt werden können. Dazu bedarf es sowohl seitens der Softwareentwicklung als auch seitens der Hardware oft einen großen Aufwand. In dieses Gebiet fällt die Aufgabenstellung der vorliegenden Diplomarbeit, ein Simulationswerkzeug mit grafischer Ausgabe zu erstellen, das Problemstellungen der Thematik der Atmosphärenphysik veranschaulicht. Weiterhin soll für vorhandene numerische Simulationsergebnisse die Möglichkeiten der automatisierten Visualisierung in Form von Einzelbildern und Filmen untersucht und angewendet werden. Die Arbeit soll die Forschungen auf diesem Gebiet unterstützen, in dem effiziente Softwarelösungen zur Visualisierung von Simulationsergebnissen gefunden werden. Die Berechnung der Simulation erfolgt auf Großrechnern. Ihre Ergebnisse nehmen

ein sehr großes Datenvolumen ein und können deshalb nur begrenzt direkt auf einem Standard PC verarbeitet werden, sodass Möglichkeiten gefunden werden sollen, die Visualisierungen automatisiert auf den vorhandenen Großrechnern zu erstellen. Die Arbeit bezieht sich zum Teil auf Erfahrungen einer vorausgegangenen Belegarbeit [1], die am IAP Kühlungsborn angefertigt wurde.

Kapitel 2

Grundlagen

Im folgenden Kapitel werden Themen und Begriffe näher beleuchtet, die zum besseren Verständnis der vorliegenden Arbeit beitragen sollen.

2.1 Arbeitsschwerpunkte des IAP

Das IAP arbeitet auf dem Gebiet der Atmosphärenphysik, wobei der Schwerpunkt bei der Erforschung der Atmosphäre zwischen 10 und 100 km liegt [2]. Hierbei werden die Mesosphäre und die dynamischen Wechselwirkungen zwischen den verschiedenen Schichten der Atmosphäre besonders berücksichtigt. Ferner wird untersucht, ob es in der oberen Atmosphäre zu langfristigen Veränderungen kommt und ob diese u. U. zur frühzeitigen Warnung von Klimaänderungen genutzt werden können. Am IAP werden drei Schwerpunkte bearbeitet:

- Erforschung der Mesosphäre:
  Die Mesosphäre wird in verschiedenen geographischen Breiten experimentell mit Hilfe von Lidars, Radars und Höhenforschungsraketen untersucht, wobei der Schwerpunkt auf der thermischen und dynamischen Struktur der Mesopausenregion liegt. Darüber hinaus werden Modellrechnungen unterschiedlicher Komplexität zum tieferen Verständnis der Phänomene von NLC, PMSE und PMWE, durchgeführt.

- Kopplung der atmosphärischen Schichten:
  Das Forschungsgebiet der Wechselwirkung von Troposphäre, Stratosphäre und

Mesosphäre dient einem verbesserten Verständnis der Atmosphäre. Atmosphärische Wellen findet man auf sehr unterschiedlichen räumlichen und zeitlichen Skalen. Sie sind das zentrale Element der dynamischen Kopplung der einzelnen Schichten. Um Anregungsprozesse von Wellen im Einzelnen zu verstehen, werden Ergebnisse von Messungen und von zeitlich und räumlich hoch aufgelösten Modellen kombiniert.

- Trends in der mittleren Atmosphäre:
  Die Untersuchungen langfristiger Änderungen der Atmosphäre erfolgen sowohl aus grundlagenwissenschaftlichem als auch aus umweltpolitischem Interesse. Dazu werden die am IAP durchgeführten langzeitigen Beobachtungsreihen sowie Temperaturmessungen in der polaren Mesosphäre, die nun schon seit fast zehn Jahren vorgenommen werden, im Hinblick auf Trends in der oberen Atmosphäre herangezogen.

2.2 Grundlagen der Atmosphäre

Um das allgemeine Verständnis für die Problemstellungen, die am IAP untersucht werden, zu erleichtern, wird ein kurzer Überblick über den Aufbau der Atmosphäre gegeben.

Die Erdatmosphäre unterteilt sich nach unterschiedlichen physikalischen und stofflichen Gesichtspunkten in Sphären und Schichten. Als gebräuchliche Einteilung wird der mittlere vertikale Temperaturverlauf verwendet, um die Atmosphäre in Höhenbereiche der Troposphäre, Stratosphäre, Mesosphäre, Thermosphäre und Exosphäre zu gliedern (Abb. 2.1). Die Grenze zwischen zwei Sphären wird Pause genannt. So liegt die Tropopause oberhalb der Troposphäre, die Stratopause oberhalb der Stratosphäre usw. [3]. Die Atmosphäre besteht aus einem Gasgemisch, das bei 0 % relativer Luftfeuchtigkeit volumenanteilig aus 78,08 % Stickstoff, 20,95 % Sauerstoff, 0,93 % Argon, 0,00005 % Wasserstoff und zu 0,00245 % aus anderen Edelgasen zusammengesetzt ist. Dieses Mischungsverhältnis gilt bis in eine Höhe von etwa 100 km, wobei 80 % der Luftmasse in der Troposphäre vorhanden sind. In ihr findet das allgemeine Wettergeschehen statt.

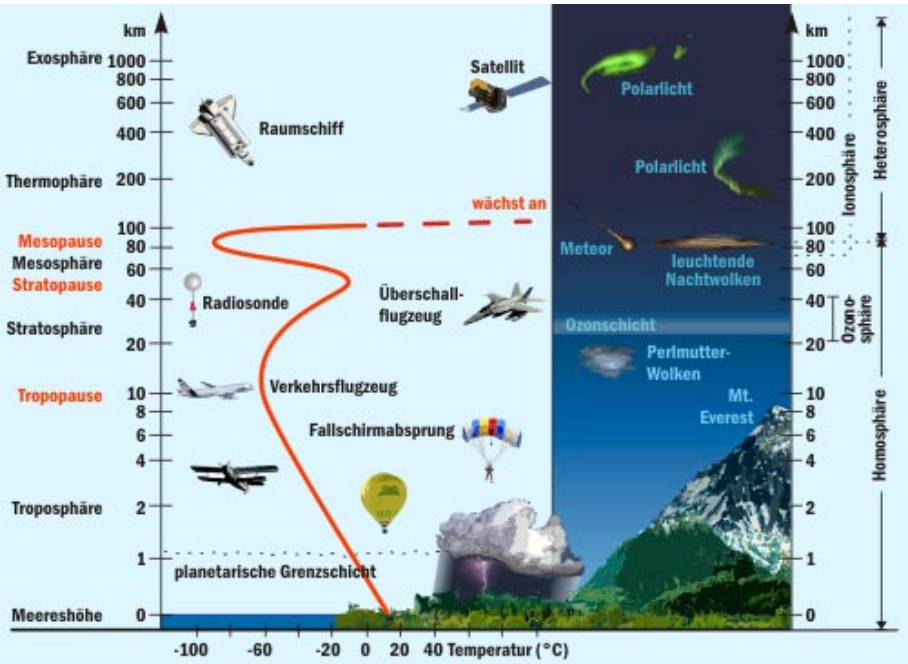

Abbildung 2.1: Thermische Schichtung der Atmosphäre [4]

Als relative Luftfeuchtigkeit ist der Wasserdampfgehalt der Luft in Prozent angegeben. Bei einer relativen Luftfeuchtigkeit von 100 % ist die Luft völlig mit Wasserdampf gesättigt, der dann wiederum bis 4 % des Gesamtluftvolumen einnehmen kann. Überschüssiger Wasserdampf kondensiert zu Tröpfchen bzw. sublimiert zu Eiskristallen [5]. Dabei wird die Luftfeuchtigkeit vor allem durch die Verfügbarkeit von Wasser, die Temperatur und den Grad der Durchmischung der Atmosphäre beeinflusst, wobei höhere Lufttemperaturen die Luft befähigen mehr Wasserdampf aufzunehmen. Das vertikale Temperaturprofil entsteht durch die Absorption des Sonnenlichts, durch die Atmosphäre und ihrem Untergrund, sowie die Rückstrahlung im infraroten Spektralbereich (Abb. 2.2). Die spektrale Energieverteilung der Sonnenstrahlung außerhalb der Erdatmosphäre, dargestellt im oberen Teilbild (Kurve a), umfasst den Wellenlängenbereich bis etwa 3,5 μm ($1\mu\text{m} = 10^{-6}$ m = 1 millionstel m) oder 3500 nm (1 nm = 10^{-9} m = 1 milliardstel m). Den höchsten Energiefluss erhalten wir am Erdboden um 500 nm im sichtbaren Spektralbereich, dargestellt im oberen Teilbild (Kurve b), der von etwa 400 nm (violett) bis 750 nm (rot) reicht. Die gesamte ultraviolette Strahlung bis etwa 175 nm Wellenlänge wird oberhalb der Mesopause, die in etwa 90 km Höhe liegt, absorbiert. Dies führt zur Ionisierung der atmosphärischen Bestandteile und zur Aufheizung der Hochatmosphäre. Oberhalb

der

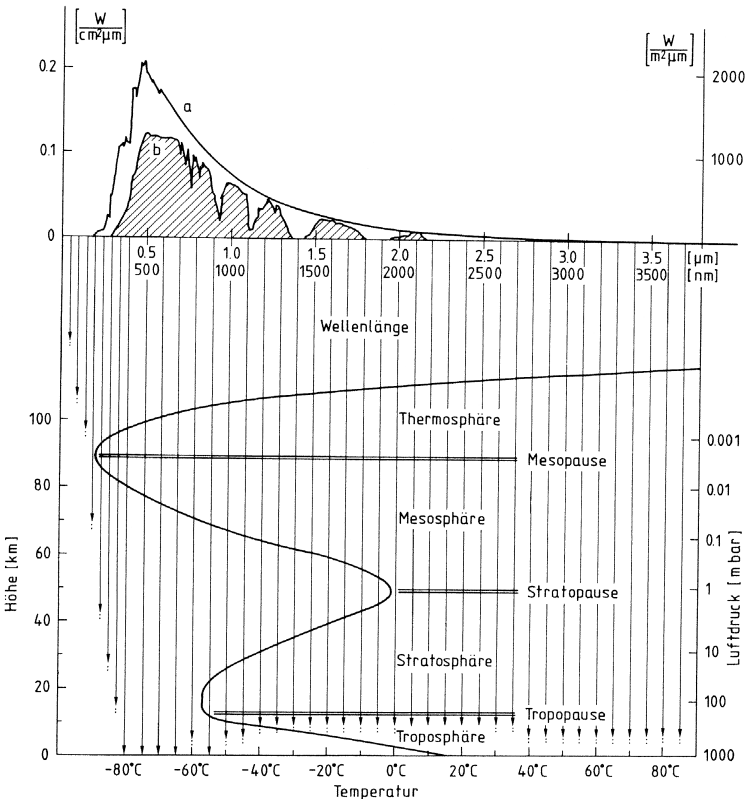

Abbildung 2.2: Spektrale Verteilung der Sonnenstrahlung (oberes Teilbild) *a* außerhalb der Erdatmosphäre, *b* am Erdboden. Temperaturverteilung und Stockwerkeinteilung der Atmosphäre (unteres Teilbild). Die warme Schicht mit einem Temperaturmaximum im Stratopausenniveau ist eine Folge der Strahlungsabsorption durch Ozon. Die senkrechten Pfeile deuten schematisch an, wie tief Sonnenstrahlung der betreffenden Wellenlängen in die Atmosphäre eindringt [5].

Mesopause steigt die Temperatur bis auf etwa 1700 ℃ in 500 km Höhe an. Die UV-Strahlung mit Wellenlängen zwischen 175 und 200 nm wird vollständig in der Mesosphäre (etwa 50-90 km Höhe), diejenige zwischen 200 und 245 nm in der Stratosphäre (etwa 15-50 km Höhe) durch Sauerstoff-Moleküle (O_2) absorbiert. Daraus resultiert eine Ozonschicht, die nun ihrerseits UV-Strahlung zwischen 200 und 340 nm sowie geringfügig auch im sichtbaren Spektralbereich um 600 nm absorbiert. Die Folge ist eine Aufheizung der Stratosphäre und Mesosphäre, wobei die Stratopause in etwa 50 km Höhe das Temperaturmaximum markiert, das ungefähr im

Bereich der Temperaturen der Erdoberfläche liegt. Die Troposphäre und die Erdoberfläche erhalten von der Sonne nur Strahlung im Wellenlängenbereich oberhalb 290 nm, wobei der UV-Anteil zwischen 290 und 340 nm auf Grund der Absorption in der Stratosphäre geschwächt ist. Der längerwellige Spektralbereich oberhalb 800 nm wird größtenteils durch Wasserdampf und Kohlendioxid in der Troposphäre absorbiert. Der überwiegende Anteil der einfallenden Sonnenstrahlung zwischen 400 und 800 nm dringt bis zur Erdoberfläche durch. Diese wird dadurch erwärmt und gibt die Wärme an die Atmosphäre ab. Dies ist der Grund für die Temperaturabnahme mit zunehmender Höhe bis zur Tropopause, die in den Tropen, wo die solaren Energieflüsse am größten sind, im Mittel bei etwa 18 km, in mittleren Breiten zwischen 10 und 15 km und in der Polarregion in nur etwa 8 km Höhe liegt. Durch die unterschiedlich großen Einstrahlungsmengen der Sonnenradiation beim Tag und Nachtwechsel (solare Gezeiten), werden atmosphärische Gezeiten, die sich über den gesamten Höhenbereich der mittleren Erdatmosphäre erstrecken, überwiegend durch thermische Prozesse angeregt. Sie gehören insbesondere in der oberen Mesosphäre und unteren Thermosphäre zu den wichtigsten dynamischen Prozessen. Diese Gezeiten sind großskalige, d.h. globale Wellen mit ausgeprägter Horizontal- und Vertikalstruktur im Wind, in der Temperatur, im Druck und in der Teilchendichte. Die Periodendauern sind harmonische Komponenten eines solaren Tages, wobei die 24 Stunden- und 12 Stunden-Wellen am stärksten ausgeprägt sind. So angeregte Gezeitenwellen wandern mit der relativen Sonnenbewegung nach Westen [6]. In der Atmosphäre sind auch Wellenstrukturen vorhanden, die sich entgegengesetzt zu den solaren Gezeiten bewegen, deren Ursachen und Folgen hier aber nicht weiter betrachtet werden sollen.

2.3 Allgemeines zum LIMA-Modell

Die Daten, die in der Aufgabenstellung dieser Arbeit visualisiert werden sollen, sind Ergebnisse des Atmosphärischen Modells LIMA (Leibniz-Institut Middle Atmosphere). Im Jahr 2005 wurde das neue Zirkulationsmodell namens LIMA am IAP fertig gestellt. Es beschreibt die wichtigsten physikalischen Prozesse in der mittleren und oberen Atmosphäre, bestehend aus Dynamik, Strahlung, Chemie und Trans-

port. LIMA beinhaltet im Vergleich zum vorhergehenden Modell zwei fundamentale Neuerungen; den Einsatz einer hochaufgelösten sphärischen Dreiecksgitterstruktur und den Einsatz von Datenassimilationstechniken für eine reale Beschreibung der Troposphäre und unteren Stratosphäre anhand von ECMWF (European Center for Medium range Weather Forecast) -Daten[1].

Der Ansatz zur Benutzung eines Dreiecksgitters auf Ikosaederbasis wurde erstmalig am Deutschen Wetterdienst eingesetzt, modifiziert und erfolgreich für die Modellierung der Atmosphäre bis zur Thermosphäre (135 km) umgesetzt. LIMA besitzt als erstes General Circulation Model (GCM)[2] der mittleren und oberen Atmosphäre eine sphärische Dreiecksgitterstruktur. Abbildung 2.3 zeigt eine Realisation des Gitters mit 6812 Gitterpunkten und einer Kantenlänge von cirka 270 km. Zwischen zwei Breitenkreisen startet eine neue Reihe von Punkten, jeweils versetzt um eine halbe Maschenweite, worüber sich eine Dreiecksstruktur bilden lässt. Der entscheidende Vorteil eines Dreiecksgitters gegenüber klassischen Längen/Breiten Gittern liegt darin, dass die räumliche Auflösung überall konstant ist. Dies ist für die Beschreibung von kleinskaligen Prozessen, wie z. B. Wellen sehr wichtig.

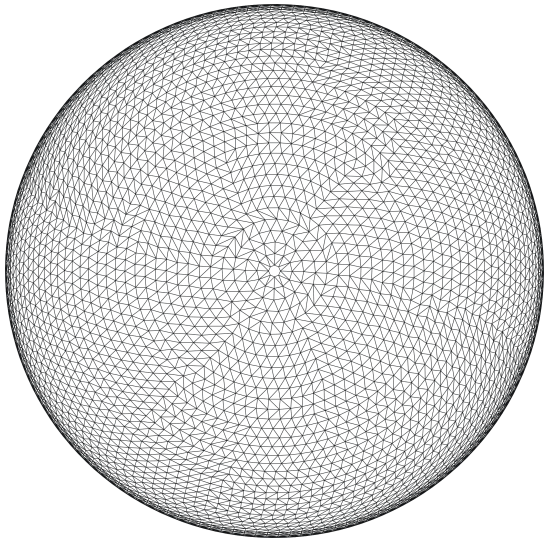

Abbildung 2.3: LIMA Dreiecksgitter mit 270 km Kantenlänge

[1] Atmosphärische Daten die täglich von ECMWF gemessen und berechnet werden.

[2] Bezeichnung für atmosphärische Modelle, die die allgemeinen Zirkulationsprozesse der Atmosphäre berücksichtigen.

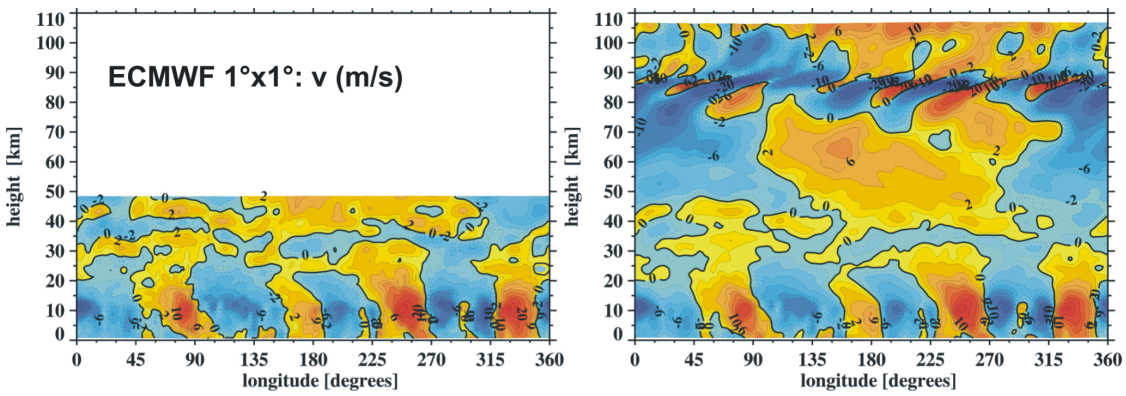

Abbildung 2.4: Vertikalschnitte vom 10. Juli 2005 (00:00 UT) für die Breite Kühlungsborn (54N) des Meridionalwind (m/s) mit ECMWF Daten (links) und LIMA Modelldaten (rechts) [2].

Die aktuelle operationelle LIMA-Version basiert auf einer Maschenweite von 110 km und besitzt 41804 Gitterpunkte. Des weiteren besitzt LIMA eine vertikale Diskretisierung von 118 Punkten (Δz=1.1km) vom Boden bis in die untere Thermosphäre. Der zweite Schwerpunkt bei den Neuentwicklungen für LIMA basiert auf der Verwendung von Datenassimilationstechniken. Die zu assimilierende Datenbank besteht aus ECMWF-Dateien, die ab dem Jahr 1980 vorliegen. Im Detail erfolgt die Assimilation in LIMA wie folgt: Jeder ECMWF-Datensatz liegt global in einem horizontalen Spektralraster von 1°$\times$ 1° auf 21 Druckniveaus (0-45 km Höhe) jeweils alle sechs Stunden vor. In LIMA werden diese Daten auf das horizontale Dreiecksgitter in einem Höhenbereich von 0-40 km interpoliert (Abb 2.4). Parallel zu allen in LIMA ablaufenden physikalischen Prozessen in 0-40 km Höhe werden die ECMWF-Größen Horizontalwind, Temperatur und Bodengeopotential mittels einer 'nudging' Technik angepasst, wobei die ECMWF-Daten innerhalb eines sechs Stunden Zeitintervalls linear interpoliert werden.

2.4 Grafik

Zu dem sehr umfangreichen Gebiet der Computergrafik sollen hier grundsätzliche Konzepte vorgestellt werden. Dabei geht es um die elementaren Objekte, mit denen Grafiken erstellt werden können: *Pixel* (Bildpunkte) und *Vektoren* (Linien). Die folgenden Angaben über Bild- und Grafikformate sind aus [7], [8] und [9] entnommen.

2.4.1 Pixelgrafik

Bei Pixelgrafiken (auch Rastergrafik) besteht das Bild aus einzelnen Bildpunkten, den Pixeln. Auf Computerbildschirmen können grundsätzlich nur Pixelgrafiken dargestellt werden. Jedes Pixel besteht bei Farbbildschirmen aus drei Elementen, welche die drei Grundfarben Rot, Grün und Blau darstellen. Für den Speicherbedarf einer Pixelgrafik sind drei Faktoren entscheidend: die Anzahl der Bildpunkte in vertikaler Richtung, ihre Anzahl in horizontaler Richtung und die Farbe der einzelnen Pixel. Eine Pixelgrafik, die ein Bild mit einer Auflösung von 800×600 Bildpunkten füllt und aus 256 verschiedenen Farben besteht, benötigt zum Speichern $800 \times 600 \times 1 = 480\,000$ Byte. In diesem Fall wird die Farbe eines jeden Pixels genau mit einem Byte gespeichert, denn ein Byte kann $2^8 = 256$ verschiedene Werte annehmen. Die Pixelgrafik hat bei der Verarbeitung mit dem Computer den Vorteil, dass sie sehr schnell auf dem Bildschirm dargestellt werden kann. Damit ist es beispielsweise möglich, Bilder in so schneller Folge anzuzeigen, dass der Betrachter eine Filmsequenz sieht. Die große Anzahl von Pixeln in einem Bild lässt vor allem bei Farbgrafiken eine hohe Informationsdichte zu. Das entscheidende Problem bei Pixelgrafiken ist der Qualitätsverlust bei Skalierungen. Bei der Vergrößerung von schrägen oder runden Linien entsteht ein Treppeneffekt der durch die gegebene Rasterung der Pixelgrafik verursacht wird. Bei Verkleinerung können Linien unterschiedlich dick erscheinen.

Grafikformate

Als Grafikformat wird eine festgelegte Vorschrift bezeichnet, nach der eine Grafik gezeichnet werden kann. Die Zahl der existierenden Grafikformate ist relativ groß, da nahezu jedes Grafikverarbeitungsprogramm in einem eigenen Format speichert. Etwa ein Dutzend Grafikformate haben sich jedoch zu einem Standard entwickelt, sodass nachfolgend nur relevante Formate kurz betrachtet werden.

BMP-Format (Bitmap)

BMP ist das hauseigene Bitmap- Grafikformat von Windows. Es ist ein sehr einfaches Standardformat für Bilder mit RGB Farbsystem, das von jedem Windowsfähigen Grafikprogramm gelesen werden kann. In den gängigen Formaten von 1 Bit (Schwarzweiß) und 24 Bit (Echtfarben) Farbtiefe werden BMP Grafiken unkompri-

miert abgespeichert. Bei anderen Farbtiefen ist eine RLE-Kompression[3] möglich. Umkomprimierte BMP-Dateien sind sehr groß, haben dafür aber keine Kompressionsartefakte und keine Kompatibilitätsprobleme. BMP unterstützt Farbtiefen von 1, 4, 8 und maximal 24 Bit. Es arbeitet nur mit dem RGB-Farbmodell.

GIF-Format (Graphics Iterchange Format)

GIF ist ein im Internet weit verbreitetes Format für Bitmap-Grafiken. Bei der Farbtiefe werden 1, 4 und 8 Bit unterstützt. Die mit maximal 256 Farben begrenzte Farbpalette ist eine Indexpalette, deren Farben frei wählbar sind. GIF verwendet zur Kompression das verlustfreie LZW-Verfahren[4], das bis vor kurzem noch patentrechtlich geschützt war. Enthält eine Grafik nicht mehr als 256 unterschiedliche Farben und sind diese über größere Flächen gleichmäßig verteilt, erzielt man mit GIF kleinere Dateien als mit JPEG, die zudem keinen Qualitätsverlust durch Kompressionsartefakte haben.

JPEG-Format (Joint Photographic Experts Group, JPG)

JPEG ist das am meisten genutzte Bildformat, speziell für digitale Fotos. Dieses Format beinhaltet eine datenverlustbehaftete Komprimierung und verwendet dazu Fourier-Transformationen. Die Kompressionsrate ist variabel. Die verlustbehaftete Komprimierung äußert sich im Bild, in dem Artefakte von 8×8 Pixeln auftauchen. Es unterstützt Graustufenbilder mit 8 Bit und RGB-Bilder mit 24 Bit Farbtiefe.

PNG-Format (Portable Network Grafics)

PNG wurde als Alternative zum früher patentrechtlich geschützten GIF-Format und für die Verwendung im Internet entwickelt. Trotz seiner Vielseitigkeit hat es aber nur eine geringe Verbreitung im Internet gefunden. Die Unterstützung des Formates durch Grafikverarbeitungsprogramme ist jedoch mit einigen Einschränkungen fast

[3] Run Length Encoding - Zusammenfassen von mehrfach hintereinander Auftretenden gleichen Zeichen in einem Datenstrom.

[4] Nach den Erfindern Lempel-Ziv-Welch. LZW enwickelt während der Kodierung ein Wörterbuch, das die bereits kodierten Zeichen aufnimmt. In den kodierten Daten sind lediglich die Indizes der Einträge in diesem Wörterbuch enthalten. Die ersten 256 Einträge werden mit den einzelnen Zeichen vorbesetzt. Alle nachfolgenden Einträge repräsentieren längere Zeichenketten.

vollständig gegeben. Bei den Farbtiefen sind 1, 8, 16, 24 und 48 Bit-RGB möglich. Die hochauflösenden Varianten mit 16 Bit Graustufen und 48 Bit Farben werden jedoch von vielen Grafikverarbeitungsprogrammen nicht unterstützt. Es kann ein Transparenzkanal (Alphakanal) abgespeichert werden. Die Kompression erfolgt mit dem verlustfreien Algorithmus LZ77[5], der bei 8 Bit Indexfarben in etwa an die Effizienz der GIF-Kompression heranreicht. Die maximale Bildgröße ist auf 30.000 × 30.000 Pixel beschränkt.

TIFF-Format (Tagged Image File Format, TIF)

TIFF ist ein Bitmapformat, das neben EPS (Encapsulated PostScript) in der Druckindustrie bevorzugt wird. Ab der Version 6 erlaubt es, Bilder in allen gängigen Farbtiefen (1-48 Bit) inklusive zusätzlichen Alphakanälen abzuspeichern. Es werden die Farbmodelle RGB und CMYK (Cyan Magenta Yellow Key (Black)) unterstützt. TIFF-Grafiken können unkomprimiert abgespeichert werden oder unter Verwendung verschiedener verlustfreier Kompressionsverfahren (z.B. LZW).

2.4.2 Vektorgrafik

Bei den Vektorgrafiken werden nicht die einzelnen Bildpunkte aufgezeichnet, sondern Bildelemente beschrieben. Diese genau definierten Bildelemente werden als Grafikobjekte oder einfach nur als Objekte Bezeichnet. Man spricht dabei von objektorientierter Grafik. Grafikobjekte sind Linien, Flächen, Texte oder der Bildhintergrund. Für jedes Objekt werden die Attribute wie Farbe, Dicke, Muster etc. aufgeführt. Die Objekte besitzen aber auch nicht sichtbare Attribute wie z.B. die Zugehörigkeit zu einer Objektgruppe, die Lage im Bild (Vordergrund, Ebene oder Hintergrund) oder auch die Zuweisung einer Materialeigenschaft oder Positionsnummer bei CAD-Zeichnungen. Eine Vektorgrafik besteht also aus einer Folge von Zeichenanweisungen, die in einer festdefinierten Sprache formuliert sind. Darin gibt es Kommandos, mit denen die Objekte gezeichnet und ihre Eigenschaften bestimmt werden können. Diese Sprachen werden als Seitenbeschreibungssprachen bezeichnet. Die Ausgabe bezieht sich auf ein Koordinatensystem, mit dem jeder beliebige Punkt auf einer

[5]LZ77 ist ein wörterbuchbasiertes Verfahren, das anstelle der Originaldaten, Rückgriffe auf Sequenzen aus dem bisherigen Inhalt kodiert.

Seite angesprochen werden kann. Der Begriff Vektor kommt daher, dass dabei Linien mit exakten Koordinaten (Anfangs- und Endpunkt) eine große Rolle spielen. Die wichtigste Seitenbeschreibungssprache ist PostScript (PS). Nahezu jedes Grafikprogramm, das Vektorgrafiken verarbeitet, kann seine Grafiken in der Sprache PostScript ausgeben. Weiter können fast alle hochwertigen Drucker damit angesprochen werden. Ein Vorteil von Vektorgrafiken liegt darin, dass sie ohne Qualitätsverlust skaliert oder gedreht werden können. Dazu ist lediglich eine Umrechnung der Koordinaten nötig. Außerdem kann die Datei, in der die Anweisungen zur Ausgabe einer Vektorgrafik gespeichert sind, relativ klein gehalten werden, da die Grafikbefehle sehr umfangreich sind. So wird das Füllen eines Polygons mit einem Muster erreicht, in dem man z.B. bei PostScript die Anweisung *fill* benutzt. Die Dateien können aber auch unbrauchbar groß werden, wenn sehr viele komplexe Objekte beschrieben werden, die sich möglicherweise sogar gegenseitig verdecken, also nicht sichtbar sind. Bei der Ausgabe auf den Bildschirm oder den Drucker wird die Vektorgrafik in eine Pixelgrafik umgerechnet, die genau für das jeweilige Ausgabegerät optimiert ist. Lediglich ein Stiftplotter kann Vektorgrafiken als solche ausgeben. Die Umrechnung einer einzelnen Vektorgrafik in eine Pixelgrafik kann einen je nach Komplexität der Grafik hohen Rechenaufwand erfordern. Daher sind sie nicht für Animationen oder Filmsequenzen geeignet.

2.5 Farbe

Im folgenden Abschnitt wird auf grundlegende technische und physiologische Aspekte der Farbgebung von Grafiken eingegangen. Sie ist eine essentielle Thematik der Visualisierung, die wesentlichen Einfluss auf rechentechnische Größen sowie die subjektive Deutung des Betrachters hat.

2.5.1 Farbwahrnehmung

Farbe wird definiert als diejenige Empfindung, die es uns ermöglicht, Objekte voneinander zu unterscheiden, die auf Grund ihrer Textur nur schwer unterscheidbar sind [10]. Farben sind rein subjektive Empfindungen, die unser Gehirn bestimmten Wellenlängen des sichtbaren elektromagnetischen Lichtspektrums zuordnet. Licht

wird vom Gehirn also erst intern, nach Absorption durch die Netzhaut, mit einer bestimmten Farbe assoziiert. Licht verschiedener Wellenlängen interpretiert unser Gehirn als Mischfarben, die sich allgemein nach der technischen Realisierung des additiven Farbmodell auf nur drei Grundfarben reduzieren lassen, Rot, Grün und Blau. Die Lichtabsorption der Netzhaut durch die drei verschiedenen Zapfenarten hat mit der Farbmischung, wie sie das technische RGB-Modell beschreibt, allerdings nur sehr wenig zu tun. Die Helligkeitswahrnehmung ist beim Menschen viel ausgeprägter als die der Farbe. Da die allgemeine individuelle Farbwahrnehmung von Mensch zu Mensch sehr variiert, ist es nötig zu wissen, wie Grafiken farblich entworfen werden können. Ein farblicher Entwurf kann so gestaltet werden, dass sogar Menschen mit teilweiser Farbblindheit einer Grafik den vollen Informationsgehalt entnehmen können [11].

2.5.2 Farbtiefe

Die Farbtiefe bezeichnet die Anzahl der Bits, die für die Speicherung der Farbe eines Pixels verwendet werden. Bei der Farbdarstellung mit einem Bit sind nur zwei Farben möglich: in der Regel Schwarz und Weiß. Werden für jeden Bildpunkt vier Bit verwendet, so können damit $2^4 = 16$ verschiedene Farben kodiert werden. Für einfache Sachverhalte in Grafiken sind im allgemeinen acht Bit für jeden Bildpunkt, womit 256 verschiedene Farben dargestellt werden können, ausreichend. Für Abbildungen wie z.B. Fotografien, die deutlich bessere Farbdarstellungen benötigen, werden Farbtiefen von 15 Bit (32 768 Farben, Hicolor) bzw. 16 Bit (65 536 Farben, Directcolor) verwendet. Als professioneller Standard hat sich inzwischen unter der Bezeichnung Truecolor eine Farbtiefe von 24 Bit etabliert. Damit können weit mehr Farbnuancen dargestellt werden, ca. 16,7 Millionen, als das menschliche Auge zu unterscheiden in der Lage ist. Es werden so viele Farbeinteilungen benötigt, um wirklich fließende Farbübergänge erzeugen zu können. Die Farbtiefe bestimmt, wie in Abschnitt 2.4.1 schon erwähnt, zusammen mit der gewählten Auflösung des Bildes den Speicherbedarf in Bytes. Der Speicherbedarf ist nicht nur für die Aufnahmefähigkeit von Massenspeichermedien, sondern auch für die Darstellung am Bildschirm wichtig, denn um ein Bild vollständig anzuzeigen, benötigt ein Rechner ausreichend Grafikspeicher. Ein leistungsfähiges Grafiksystem (Grafikkarte und

Bildgröße	Farbtiefe	Farbanzahl	Speicherbedarf
640x480	1 Bit	2	38 KB
640x480	4 Bit	16	150 KB VGA
640x480	24 Bit	16.7 Mio	900 KB
800x600	8 Bit	256	469 KB Super VGA
800x600	24 Bit	16.7 Mio	1.4 MB
1024x768	24 Bit	16.7 Mio	2.3 MB
1280x1024	24 Bit	16.7 Mio	3.8 MB
1600x1200	24 Bit	16.7 Mio	5.6 MB
3072x2048	24 Bit	16.7 Mio	18.4 MB Photo-CD
6144x4096	24 Bit	16.7 Mio	73.7 MB Pro Photo-CD

Tabelle 2.1: Speicherbedarf in Abhängigkeit von Bildgröße und Farbtiefe für unkomprimierte Pixelgrafiken [10].

Bildschirm) ist für viele Grafikanwendungen wie z.B. Bildbearbeitung, eine unabdingbare Voraussetzung. Der Speicherbedarf in Abhängigkeit verschiedener Kombinationen von Bildgröße und Farbtiefe ist in Tabelle 2.1 veranschaulicht.

2.5.3 RGB-Modell

Ein Farbmodell ist ein Verfahren zur Darstellung bzw. Beschreibung von Farben. Das RGB oder auch additive Farbmodell wird immer dann verwendet, wenn Bilder von selbstleuchtenden Medien dargestellt werden sollen. Die Farben auf einem Bildschirm entstehen dadurch, dass jedes Pixel aus einer Kombination der drei Grundfarben Rot, Grün und Blau besteht. Denn die Grundfarben des RGB-Modells wirken additiv und entstehen durch Ausstrahlung von „farbigem“ Licht (bzw. einer bestimmten Wellenlänge). Das RGB-Modell arbeitet normalerweise mit dem Truecolorstandard. So kann jeder der drei Punkte eines Pixels mit einer Intensität zwischen 0 (leuchtet gar nicht) und 255 (maximale Leuchtkraft) angesteuert werden. Überlagert sich rotes, grünes und blaues Licht mit maximaler Leuchtkraft, erhält man Weiß. Leuchten die drei Punkte eines Pixels gar nicht, erhält man Schwarz.

Abbildung 2.5: Additives RGB-Farbmodell und seine Mischfarben [26].

2.5.4 CMYK-Modell

Für Farbausdrucke auf nicht selbstleuchtenden Oberflächen wie Papier benötigt man das CMYK-Farbmodell. CMYK steht für die Komplementärfarben **C**yan (Hellblau), **M**agenta (Purpur), **Y**ellow (Gelb) und zusätzlich Blac**k** bzw. **K**ey (Schwarz). Hier wirken die Farben subtraktiv, sie absorbieren einfallendes Licht und reflektieren nur die Anteile, die ihrer Farbwahrnehmung entsprechen. Eine blaue Oberflächenfarbe absorbiert also alles Licht außer den Blauanteilen. Die damit erzielte subtraktive Farbmischung führt in der Praxis zu einigen Problemen. Mischt man die drei Farben Hellblau, Purpur und Gelb im gleichen Verhältnis, so sollte nach dem Konzept ein reines Grau herauskommen. Tatsächlich erhält man einen bräunlichen Farbton. Erst nach einer Korrektur, durch Erhöhung des hellblauen Anteils, wird diese Abweichung wieder ausgeglichen. Ein deckendes Schwarz ist damit jedoch auch nicht zu erzielen. Aus diesem Grund wird in diesem Farbmodell mit der vierten Farbe, dem Schwarz gearbeitet, mit dem sich der Kontrast deutlich erhöht. Durch den zusätzlichen Farbkanal erhöht sich auch die Datentiefe von 24 Bit auf 32 Bit (4×8 Bit je Bildpunkt). Das hat zur Folge, dass von RGB nach CMYK umgewandelte Bilder 33 % mehr Speicherplatz benötigen. Trotz dieser höheren Datentiefe ist aber der Farbraum (die Gesamtzahl aller darstellbaren Farben) kleiner als bei RGB. Das heißt, dass beim herkömmlichen Druck mit CMYK nicht alle Farben, die am Bildschirm sichtbar sind, hundertprozentig dargestellt werden können. Die Umsetzung

Abbildung 2.6: Subtraktives CMYK-Farbmodell und seine Mischfarben [26].

2.5.5 Probleme durch Grafikausgabemedien

vom RGB-Schema in das CMYK-Modell wird als Farbseparation bezeichnet. Ein Problem ist, dass die verschiedenen Grafikausgabemedien, Bildschirmausgabe und Druckerausgabe wie oben schon erwähnt, zwei unterschiedliche Farbmodelle benutzen, die nicht eins zu eins übersetzt werden können. Das äußert sich in der Praxis so, dass z.B. bei den RGB-Farbwerten, die ein Scanner liefert, am Bildschirm fast immer eine gewisse Farbabweichung vom Original auftritt. Auch wenn man ein Bild ausdruckt, gibt es deutliche Abweichungen zwischen Bildschirm- und Druckfarben. Um dieses Problem zu umgehen, werden im professionellen Bildverarbeitungsbereich meist genormte Farben verwendet. Eine große Verbreitung hat das Pantonesystem zur Farbauswahl. Anhand der dabei verwendeten Farbmustertabellen lässt sich das Ergebnis einer Farbauswahl sehr genau vorhersagen [9]. Es besteht aus über 800 Referenzfarben, deren exakte Position im RGB- und CMYK-Modell definiert ist. Diese Farbinkonsistenz zwischen den Medien durch die unterschiedlichen Farbmodelle werden zusätzlich durch die unterschiedlichen technischen Realisierungen der einzelnen Ausgabemedien verstärkt. Das heißt, dass ein und dasselbe Bild an zwei verschiedenen Rechnern farbliche Unterschiede aufweisen kann. Das kann direkt an dem jeweiligen Bildschirmen liegen, z.B. am technischen Zustand oder an der Konstruktionsweise des jeweiligen Herstellers. Weiter kann die Grafikkarte und ihre individuellen Einstellungen, sogar die Qualität der Signalübertragung von der

Grafikkarte zum Bildschirm die farbliche Darstellung beeinflussen. Dasselbe gilt für die Drucker, hier kann die chemische Zusammensetzung der einzelnen Farben eine gravierende Rolle spielen. Durch all diese Faktoren ist es sehr schwierig eine Grafik farblich so zu gestalten, dass sie ihre Informationen immer in gleicher Qualität preisgibt. Diese Probleme müssen insbesondere in dieser Arbeit beachtet werden, da die erstellten Bilder für Präsentationen durch selbstleuchtende Ausgabemedien sowie Printmedien genutzt werden sollen.

Kapitel 3

Theoretische Vorbetrachtungen

3.1 Automatisierte Ergebnisvisualisierung

3.1.1 LIMA Ergebnisdaten

Die Simulation ist mit Fortran90 implementiert worden und wird zur Zeit auf einem Computingserver ausgeführt. Von dort aus speichert sie jede sechs Stunden eines simulierten Tages ein Ergebnisfile auf einem Fileserver. Jedes Datenfile ist 112 MB groß und wird binär im unformatierten Fortranformat abgelegt. Die LIMA-Version, die hier betrachtet werden soll, simuliert wie schon erwähnt, das Wettergeschehen global auf einem Dreiecksgitter mit 110 km Maschenweite am Erdboden und einer vertikalen Auflösung von 118 Schichten auf jeweils gleichem Druckniveau mit einem Höhenabstand von ungefähr 1.1 km zueinander. Auf jedem Punkt dieses Systems werden die Größen Zonalwind[m/s], Meridionalwind [m/s], Temperatur [K], geometrische Höhe [m], Dichte [g/cm^3] und Vertikalwind für jeden Zeitschritt berechnet. Das heißt, dass jede dieser Einheiten in einem Feld der Größe Gitterpunktanzahl (41802) $\times$ Anzahl der Höhenbereiche (118) vorliegen. Diese Größen werden durch physikalische und chemische Prozesse in der Simulation verändert. Die Dateinamen werden in der Form YYYYMMDD.HH.bin erstellt, wobei hier die Platzhalter für Zahlen von 0 bis 9 stehen, Y für Jahr, M für Monat, D für Tag und H für Stundenzeitangaben. Die Dateien werden in der zugehörigen Katalogstruktur immer für einen Monat gespeichert: ..LIMA/LIMA-DATA/global/YYYY/MM. Zu jedem Modellgitter gibt es ein Koordinatenfile in dem die Ortskoordinaten für jeden Gitterpunkt

des Dreiecksgitters festgehalten sind.

3.1.2 Visualisierung

Diese Daten sollen nun mit der Programmiersprache IDL von Creaso © [12], die in der Version 6.3 vorliegt, visualisiert werden. Zur Auswahl stehen verschiedenste Visualisierungsmöglichkeiten, die durch die Bedürfnisse der betroffenen Zielgruppe und die jeweilige verwendete Software begrenzt werden. Die Komplexität der Visualisierungsart muss dem Zweck und den Kenntnissen der Zielgruppe angepasst sein. Die Bildgröße sollte so gewählt werden, dass alle Informationen gut erkennbar sind und die Bilder eventuell für weitere Zwecke, z.B. Power Point Präsentationen oder Filme weiterverarbeitet werden können. Die Farbgebung der Bilder sollte eindeutig sein, d.h., dass die verschiedenen darzustellenden Informationen mit unterschiedlich erkennbaren Farben zu visualisieren sind. Bei der Farbgebung ist zu beachten, dass bei mehreren Bildern, die den gleichen Sachverhalt zeigen, aber einen unterschiedlichen Wertebereich aufweisen, zu klären ist, ob hier ein zeitlicher Vergleich in einer Abfolge der Bilder anzustellen ist. Wenn dies zutrifft, muss im Vorhinein der Wertebereich des darzustellenden Sachverhaltes festgelegt werden. Da er für alle Bilder auf einer Farbtabelle fest abgebildet wird, um in einer sequentiellen zeitlichen Abfolge, z.B. einem Film, Unterschiede korrekt erkennbar zu machen. Soll wiederum der Wertebereich jedes Bildes genauestens abgebildet werden, so muss die Farbtabelle bei jedem Bild individuell dem Wertebereich angepasst werden, um Veränderungen im Bild so deutlich wie möglich darzustellen. Eine Auswertung in einem Film wird dadurch jedoch unmöglich, da Unterschiede zwischen den Bildern nicht mehr nachvollziehbar sind. Ist der darzustellende Wertebereich in positive und negative Werte aufgeteilt, so muss auch die Farbtabelle an der dargestellten Null deutlich geteilt sein. Gängige Visualisierungsvarianten in der Atmosphärenphysik sind z.B. einfache zweidimensionale globale (Abb. 3.1a) oder polarstereographische Abbildungen, um globale Datensätze darzustellen. Bei den polarstereografischen Projektionen ist einer der beiden Pole in der Mitte des Bildes zentriert. Die sichtbare Globushälfte besitzt bei der zweidimensionalen Variante keine Krümmung (Abb. 3.1b). Diese Variante wird jedoch meist nur mit Fokussierung auf einen Pol angewendet wie in

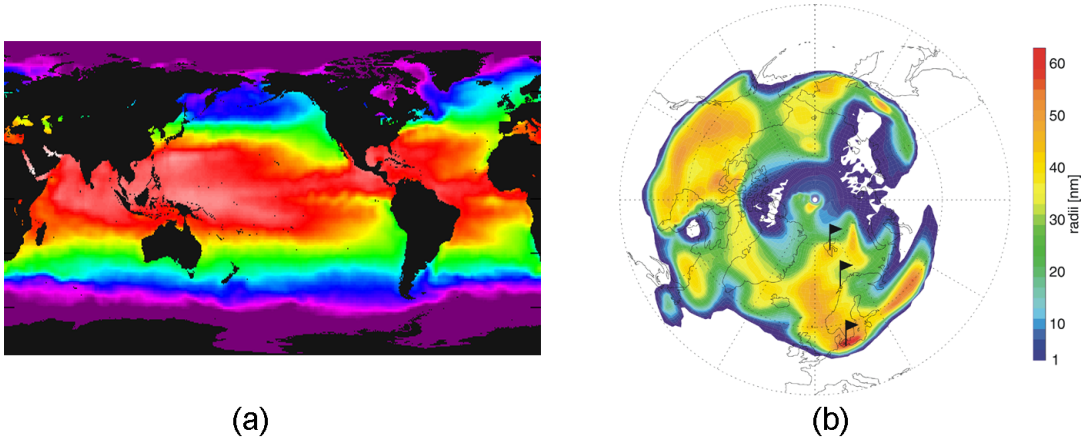

Abbildung 3.1: 2D Abbildungsbeispiele, (a) einfache globale 2D Projektion von Wasseroberflächentemperaturen, (b) polarstereografische 2D Projektion für Eisradiuswerteeiner Eiswolke [2]

der Abbildung auch dargestellt wird, weil die Positionskoordinaten zum Äquator hin immer mehr verzerrt werden und so ein sehr unwirkliches Gesamtbild entsteht. Die Fokussierung wird im Allgemeinen angewandt um die Betrachtung auf einen wesentlichen Bereich zu begrenzen. Weitere Varianten sind Sichten auf ebene Landschaftsausschnitte oder horizontale und vertikale Höhenschnitte (Abb. 2.4). Die Daten auf diesen Abbildungen sind entweder auf Flächen glatt gerechnet (interpoliert) um einen Gesamteindruck der zu betrachtenden Daten zu bekommen oder direkt geplotet. Bei der letzteren Methode wird jeder einzelne Wert mit seinem Farbwert mittels einem Zeichen oder Punkt auf dem Bild positioniert. So kann man z.B. bei Simulations- oder Messergebnissen überprüfen ob sich einzelne Werte außerhalb des erwarteten Wertebereichsrahmen befinden. Bei dreidimensionalen Darstellungen besitzen die Daten oder der Hintergrund der Abbildungen Körperformen. Bei den polarstereografischen Plots wird dann zum Beispiel die Erdkrümmung berücksichtigt (Abb. 3.2). Diese Art der Erdballvisualisierung bietet sich auch an um andere Teile des Globus zu zentrieren. Bei den Landschaftsausschnitten erhält die Landschaft dann ein Profil und die Daten eine dreidimensionale Körperform (z.B. eine Wolkenform), wie man es auch aus den Wetterberichten kennt. Bei dreidimensionalen stereografischen Direktplotdarstellungen werden die diskreten Daten mittels

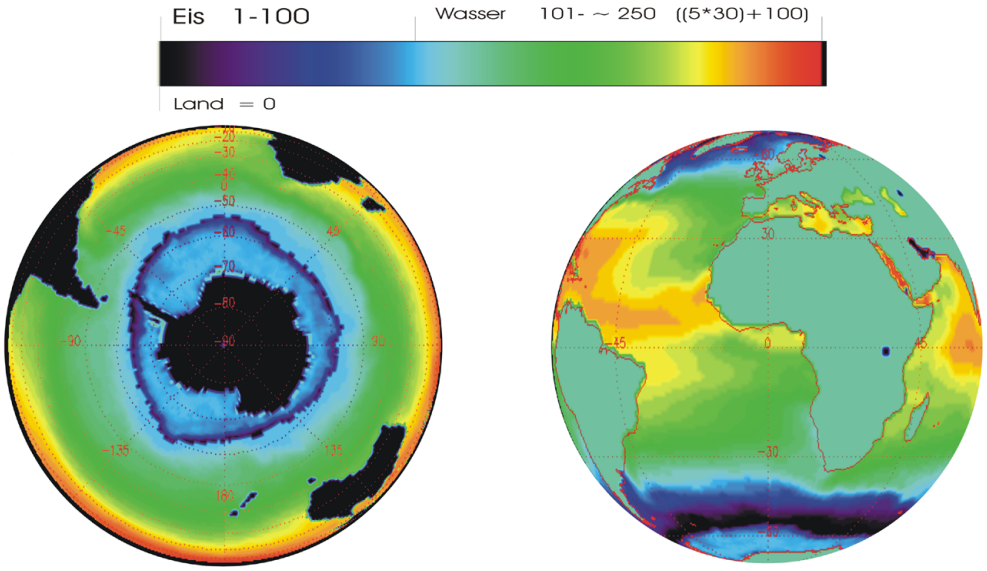

Abbildung 3.2: 3D Abbildungsbeispiele für globale Projektionen von Wasseroberflächentemperaturen

Polygone realisiert. In einigen Fällen ist es nötig, mehrere unterschiedliche Informationen in einer Grafik darzustellen, um Zusammenhänge zu erfassen. Dies kann z.B. mit Hilfe von Farbkodierungen der einzelnen Informationen realisiert werden. Das linke Bild in der Abbildung 3.2 zeigt so ein Beispiel. In ihm werden Wasseroberflächentemperaturen dargestellt die aus Satellitenmessungen resultieren. Da Wasser unter 0 ℃ seinen Aggregatzustand ändert, wird in dem Bild ab diesem Temperaturpunkt die Dicke der Eisschicht über dem Meeresspiegel dargestellt. Landmassen werden Schwarz dargestellt. Der Farbbalken über den beiden Bildern in Abbildung 3.2 zeigt die farbliche Kodierung für diesen Fall. Der erste Wert der Farbtabelle ist für die Landmassen reserviert, die Werte 1-100 für eine Eisschichtdicke zwischen 1-100 m und die Werte 101-250 für die Wasseroberflächentemperatur zwischen 0 und 30 ℃ . Weiter können mit grafischen Objekten eine Vielzahl von unterschiedlichen Informationen dargestellt werden, wie z.B Kontinentenumrahmungen (Abb. 3.2 rechts), Gradnetze, Positionsangaben (Abb. 3.1b) und andere deckungsgleiche Informationen. Mit deckungsgleich ist gemeint, dass verschiedene Daten, die z.B. in der gleichen Auflösung und auf den gleichen Koordinaten vorhanden sind, gleichzeitig dargestellt werden können. Ein gutes Beispiel dafür sind Strömungsbilder (Abb. 3.3), in denen gleichzeitig in einem Bild Windstärke und Richtungen zusammen mit

Temperaturen dargestellt werden können, die in dieser Arbeit auch noch näher betrachtet werden sollen. Dabei wird die Temperatur im Hintergrund farblich kodiert und der Wind mit diversen Symbolen dargestellt. In welchem Format die einzelnen Bilder abgespeichert werden, hängt davon ab, wie sie archiviert und weiterverarbeitet sollen. In dieser Arbeit werden Pixelgrafikformate und Vektorgrafikformate erstellt. Pixelgrafikformate bieten sich an, da viele Einzelbilder gespeichert werden sollen, die sehr komplexe Farbverläufe besitzen. Außerdem soll die Möglichkeit bestehen, im Nachhinein aus den Einzelbildern einen Film zu erstellen. Da die gängigen Filmformate Pixelgrafiken als Grundlage verwenden, ist dies ein weiterer Punkt sich für die Pixelformate zu entscheiden. Vektorgrafiken werden durch die fließenden Farbübergänge von z.B. Contourplots sehr komplex und belegen dadurch sehr viel Speicherplatz.

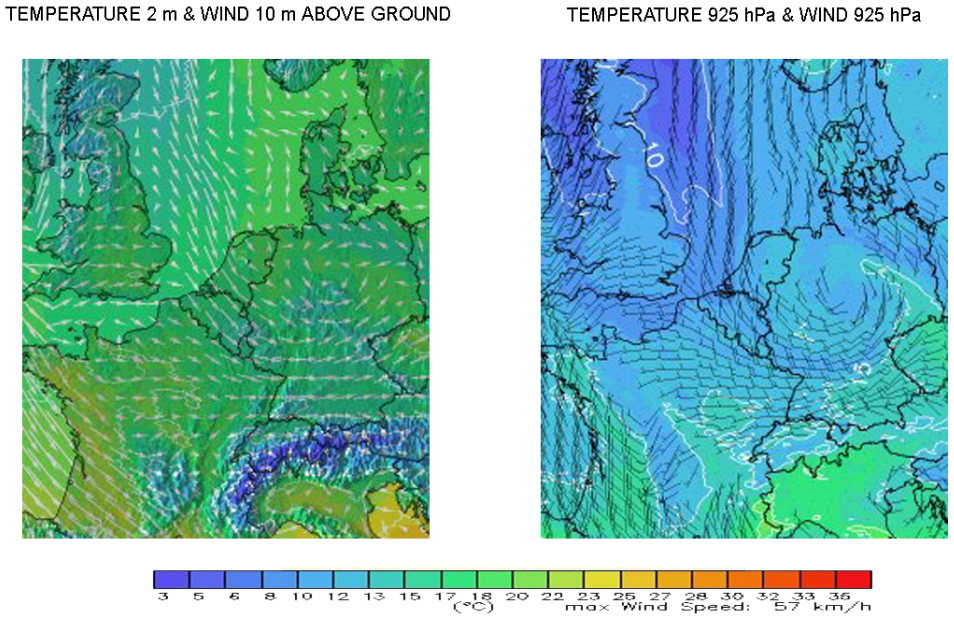

Abbildung 3.3: Abbildungsbeispiele für Temperatur und Wind. Links mit farblich kodierten Temperaturen auf 3D Reliefkarte im Hintergrund und weißen Windpfeilen im Vordergrund, die mit ihrer Länge die Windstärke symbolisieren. Rechts mit farblich kodierten Temperaturen mit Isothermen im Hintergrund und schwarzen Windfähnchen, deren Symbole am Schaft (Fieder, Dreiecke) die Windgeschwindigkeit angeben [13].

Für Präsentationszwecke von Einzelbildern, besonders in der gedruckten Form, sind Vektorgrafiken aber wichtig, da der Informationsgehalt des Bildes bei jeder Größenskalierung exakt erhalten bleibt und die meisten Drucker diese Formate sehr genau abbilden können. Die gebräuchlichsten Formate sind hier PS (PostScript) und EPS (Encapsulated PostScript), die auch standardmäßig mit IDL erstellt werden können. Um Speicherplatz zu sparen, sollen nur vereinzelt ausgewählte Vektorgrafikdateien erstellt werden. Die Bildgröße der Pixelgrafikformate sollte so gewählt werden, dass die gewünschten Informationen gut erkennbar sind, aber sich die Speicherplatzbelegung im vertretbaren Rahmen befindet. Weiter sollten die Bilder so aufgelöst sein, dass sie der Auflösung des Filmformates im Nachhinein entsprechen. Gebräuchliche Pixelgrafikformate sind BMP, PNG, GIF, TIFF und JPEG. JPEG (File Interchange Format) mit der Dateiendung '.jpg' ist ein sehr weit verbreiteter Standard mit einer variablen verlustbehafteten Komprimierung. In IDL können JPEG Bilder mit einem Qualitätsfaktor zwischen 1 und 100 erstellt werden. PNG(Portable Network Graphics) wurde speziell für Webanwendungen entwickelt und vereint Vorteile von GIF und JPEG in sich. Dieses Bildformat ist hier favoritisiert. Die Erfahrungen mit diesem Format zeigen, dass mit den verschiedensten Programmen kompatibel ist und sich zusätzlich mit seiner verlustfreien Komprimierung sehr zur Weiterverarbeitung eignet. Die Formate BMP, GIF und TIFF besitzen nicht die günstigen Kompressionseigenschaften bzw Kompatibilität wie die Formate JPEG oder PNG, sollten jedoch als Option erstellbar sein, um eine größtmögliche Flexibilität zu gewährleisten. Bei der Filmerstellung ist zu beachten, in welchem Rahmen der Film letztendlich präsentiert wird. Sollte er z.B. in Verbindung mit einer PowerPoint - Präsentation vorgeführt werden, müssen die von PowerPoint unterstützten Videoformate beachtet werden. Soll der Film mit einem Overheadprojektor als Vollbild oder in einem Ausschnitt auf einer Webseite präsentiert werden, müssen die Bildauflösungen beachtet werden, um Verzerrungen durch nachträgliche Vergrößerung oder Verkleinerung zu verhindern. Wenn der Film auf einem fremden Rechner vorgeführt werden soll, ist sicher zu stellen, ob der verwendete Rechner auch über den 'Codec' verfügt, mit dem der Film erstellt wurde. Ein Videoformat kann mit unterschiedlichen Codecs erstellt werden. Es gibt neben den ohnehin zahlreichen Formaten - etwa AVI, MPEG, Quicktime MOV und Realplayer (etwa RM) - oft eine Vielzahl an Unterfor-

maten, die mit anderen Komprimierungsverfahren erstellt werden. Das gilt vor allem für die Formate MPEG und AVI. Diese Unterformate werden durch die jeweiligen Codecs bestimmt. Bei fast allen Videoformaten kann die Wiedergabequalität beeinflusst werden, wenn es das verwendete Videobearbeitungsprogramm zulässt. Welches Bildformat in Verbindung mit welchem Videoformat und Codec ein befriedigendes Ergebnis in Bildqualität und Ergebnisdateigröße liefert, muss durch Ausprobieren verschiedener Varianten festgestellt werden. Mit IDL können MPEG Filme erstellt werden. Die Qualität dieser Filme ist aber leider sehr schlecht, sodass eine Variante umgesetzt werden muss, mit der Filme nachträglich aus Einzelbildern erstellt werden können. Dafür steht einmal VideoMach von Gromada © [14] zur Verfügung, welches ein sehr vielfältiges Audio/Videoerstellungs- und Konvertierungswerkzeug ist, das alle allgemein gebräuchlichen Bild-, Audio- und Videoformate verarbeiten kann. Dieses Programm ist allerdings nur über seine GUI (Graphical User Interface) zu bedienen. Weitere Software ist Convert von Imagemagick © [15] und MJPEGTOOLS [16]. Dies sind sehr vielfältige Programme die aber beide aus der Kommandozeile heraus bedient werden und damit aus Skripten heraus ausgeführt werden können.

3.1.3 Datenarchivierungs- und Netzwerkstruktur

Dem IAP stehen verschiedene Computingserver und ein zentraler Fileserver zur Verfügung, um große Datenmengen verarbeiten zu können. Da das angestrebte Visualisierungsprogramm in dieser Rechnerstruktur arbeiten soll, ist es wichtig, die Arbeitsweise dieses Systems zu kennen. Das IAP ist in zwei Gebäude aufgeteilt, dem Hauptgebäude mit den Büros der Wissenschaftler und dem Rechenzentrum in dem die Großrechentechnik konzentriert ist. Die beiden Gebäude sind mit vier mal 4 Gb/s Leitungen verbunden. Alle Daten, die zu groß für Desktop PC's sind oder öffentlich im Rahmen des IAP's verfügbar sein sollen, werden auf dem zentralen Fileserver gespeichert. Sein allgemeiner Name ist DMF, der von der Bezeichnung des Betriebsystems des Fileservers stammt. Der Fileserver besteht aus einem Origin® 350 Steuerrechner von SGI® und einer Scalar i2000 Magnetbandbibliothek von Adic. Die Bandbibliothek besitzt eine Bandrobotik mit 100 TB und einen Plattencache mit 4 TB Speicherkapazität. Der Steuerrechner von SGI® besitzt acht R16000 Prozessoren mit je 700 MHz. Sein Betriebssystem mit dem Namen Data Migration

Facility (DMF) verwaltet die Daten auf der Bandbibliothek. Die Computingserver, die in dieser Arbeit berücksichtigt werden, sind zwei HP Server, die APOLLO und die HYDRA. Die Apollo besitzt acht RISC PA8700 Single-Core Prozessoren mit je 750 MHz, 8GB Hauptspeicher und 430 GB Festplattenspeicher im Raid-verband. Das Betriebsystem ist HPUX 11.0. Die HYDRA besitzt acht tanium-2 Prozessoren, 8 GB Hauptspeicher und das Betriebssystem HPUX 11.23. Diese Rechner haben eine direkte 1 Gb/s Verbindung mit dem DMF. Die einzelnen Benutzer können sich per remote-login auf den Computingservern einloggen. Von dort aus können sie auf den DMF zugreifen, der als gemounteter Ordner verfügbar ist. Da einige Programme auf HPUX Betriebssystemen nicht lauffähig sind, wurden einige Desktop PC's als Linuxserver eingerichtet. Auf den DMF kann auch direkt vom PC als Netzlaufwerk zugegriffen werden. Das gesamte Netzwerk (Abb 3.4) des IAPs verfügt über eine Bandbreite von 1 Gb/s und ist mit Switchen segmentiert. Die Zugriffszeiten auf die Daten können auf dem DMF sehr unterschiedlich sein, je nachdem ob sie auf den Festplatten oder auf den Bändern vorliegen.

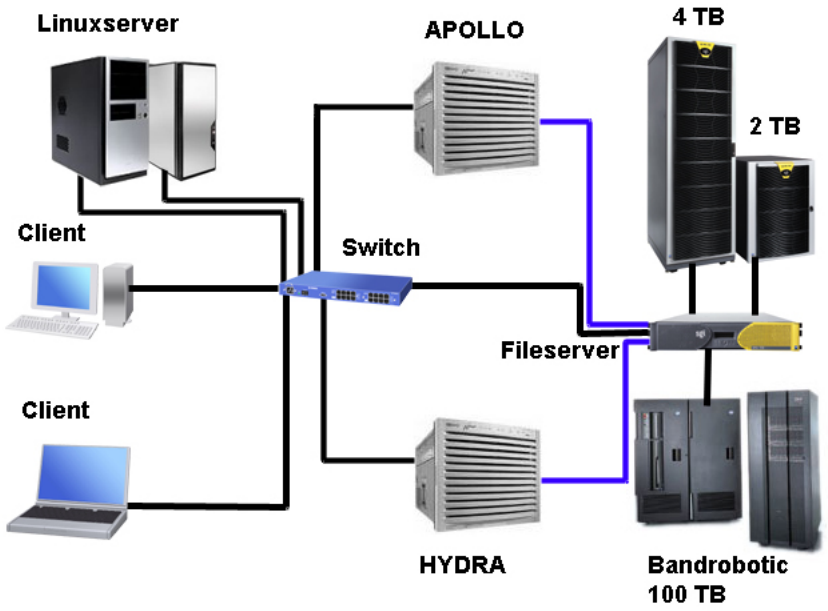

Abbildung 3.4: Schematische Darstellung des IAP Netzwerkes das einem Client- Server-Modell mit zentralem Fileserver zugrunde liegt. Dargestellt sind unter anderem die Computingserver APOLLO, HYDRA und die Linuxserver. Die Direktanbindung von APOLLO und HYDRA an den DMF ist blau dargestellt.

Wo die Daten zurzeit des Zugriffs liegen, hängt von ihrer hierarchischen Einstufung ab, die der DMF vornimmt. Alle Daten, die vom DMF übertragen werden, liegen im sogenannten „Plattencache“. Sind die Daten nicht im „Plattencache“ vorhanden, müssen sie erst vom Bandlaufwerk gelesen werden. Da der DMF eine Vielzahl von Bändern besitzt, wird jedes einzelne Band von einem Roboterarm aus seinem Lagerschacht geholt und in ein Bandlesegerät eingelegt. Wenn die angeforderten Daten ausgelesen sind, wird das Band wieder in seinen Schacht zurückgelegt. Da ein Magnetband nicht wie eine CD oder Festplatte durchsucht werden kann indem vom Lesekopf einfach der entsprechende Sektor oder die entsprechende Spur angesteuert wird, muss das entsprechende Band eventuell bis zum letzten Eintrag durchgespult werden. Hierbei kann es nach Herstellerangaben bis zu einigen Minuten dauern, bis die Daten verfügbar sind. Dies ist aber nicht der Regelfall, sondern eher die Ausnahme. Wenn die Daten auf dem „Plattencache“ vorliegen, sollten quasi keine Verzögerung bei einem Datenzugriff von einem PC aus vorliegen. Der Hersteller gibt die Verzögerungszeit mit unter einer Sekunde an; der Benutzer am PC nutzt das Netzlaufwerk in diesem Fall als wäre es direkt auf seiner Festplatte. Wenn nun auf Daten zugegriffen wird, die auf den Bändern vorliegen, gibt es eine Verzögerung von im Durchschnitt 10 - 40 s, je nachdem wo sich die Daten auf dem Band befinden. Anders verhält sich dies, wenn mehrere Programme parallel auf den unterschiedlichen Computingservern laufen, die jeweils eine sehr große Menge an Dateien sequentiell verarbeiten und sie genau so anfordern. Das bedeutet, dass ein einzelnes Programm sequentiell eine Datei lädt, sie verarbeitet und danach meist noch eine Datei schreibt und dann dasselbe erneut mehrere tausendmal durchführt. Wenn nun jede einzulesende Datei auf Band gespeichert ist, wird bei jedem Lesevorgang einmal die Bandrobotik betätigt wie oben beschrieben. Wenn dies nun mehrere solcher Programme gleichzeitig versuchen, bricht der Datenverkehr irgendwann zusammen, weil die Warteschlange zu lang wird und somit auch die Wartezeit. Die hierarchische Einstufung erfolgt mittels zwei Faktoren. Erstens, wie lange ist der letzte Aufruf der Daten auf dem DMF her und zweitens durch die Größe der einzelnen Daten. Generell ist der DMF so konfiguriert das wenn Daten einmal aufgerufen wurden, sie 48 Stunden auf dem „Plattencache“ vorliegen. Wenn der „Plattencache“ durch viele Zugriffe volläuft, müssen Dateien ausgelagert werden; hier stehen dann die kleinsten

Dateien in der Hierarchie ganz oben, sodass die größten als erstes wieder auf die Bänder ausgelagert werden. Beobachtungen haben gezeigt, dass die meisten Programme in den frühen Morgenstunden mit der Fehlermeldung *connection timed out* abstürzen. Dies liegt daran, dass zu dieser Zeit sämtliche Backup- Programme im IAP gestartet werden. Solche Abstürze durch zu lange Wartezeiten können durch Errorhandler vermieden werden, die den aufgetretenen Fehlercode behandeln, indem sie so lange warten, bis die erforderliche Datei verfügbar ist, und durch eine intelligente Datenverwaltung gemanaged werden. Der Hintergrund für die intelligente Datenverwaltung ist, dass Datenmengen zusammenhängend vom DMF auf den Bändern abgelegt werden. So sollten die erforderlichen Daten in den „Plattencache“ (Online machen) geladen werden, bevor sie von einem Programm ausgelesen werden müssen. Dies sollte in einer größeren Dateianzahl geschehen um Bandzugriffe für jede einzelne Datei zu verhindern. Für diese Problemstellung gibt es eigene Befehle und Zustände der Dateien auf dem DMF, die in den Tabellen 3.1 und 3.2 veranschaulicht werden. Leider sind diese Befehle auf dem DMF des IAP's Nutzerrechtlich beschränkt. Im Klartext heißt das, dass nur der Nutzer, der die Dateien angelegt hat, diese Befehle nutzen kann. Dies soll in naher Zukunft durch ein öffentliches Verzeichnis auf dem DMF, in dem alle Nutzer die gleichen Rechte besitzen, umgangen werden.

Zustand	Bedeutung
REG	Daten auf Disk
OFL	Daten auf Band
DUL	Daten auf Band und Disk
MIG	Daten werden von Disk auf Band kopiert
UNM	Daten werden von Band auf Disk kopiert
PAR	Daten beim Kopieren von Band auf Disk, teilweise schon auf Disk

Tabelle 3.1: Übersicht der möglichen Dateizustände auf dem DMF.

Befehl	Auswirkung
dmls	Statusabfrage
dmget	Dateien werden vom Zustand REG → DUL gebracht
dmgetdir	Ordner werden vom Zustand OFL → DUL gebracht
dmput	Dateien werden vom Zustand REG → DUL gebracht
dmputr	Dateien werden vom Zustand REG → OFL gebracht
dmputdir	Ordner werden vom Zustand REG → DUL gebracht
dmputrdir	Ordner werden vom Zustand REG → OFL gebracht

Tabelle 3.2: Übersicht DMF Befehle.

3.1.4 Möglichkeiten der Automatisierung

Die Visualisierung der LIMA-Daten soll möglichst automatisiert von statten gehen. Aufgaben, die von Computern im Allgemeinen automatisch ausgeführt werden sollen, sind Arbeiten wie z.B. Überwachungsaufgaben, wiederkehrende Aufgaben die in bestimmten Zeitintervallen ausgeführt werden sollen oder rechenintensive Prozesse die über eine lange Zeit verarbeitet werden, ohne dass das Beisein einer Person erforderlich ist. Für solche Aufgaben hat sich das sogenannte „Scripting“ etabliert. Scripting ist das Schreiben eines Programms mit Hilfe einer Skriptsprache. Das Programm wird in diesem Zusammenhang Skript genannt. Kriterien, die Skriptsprachen von anderen Sprachen unterscheiden [17] sind, das eine Skriptsprache dem Ad-hoc-Gebrauch dient, das sie interpretiert wird ohne das eine Kompilierung notwendig ist und sie einfach zu erlernen und zu verwenden ist. Beispiele für Skriptsprachen sind z.B. Perl, PHP, AppleScript, JavaScript und natürlich auch die Unix-Shellsprachen wie sh und csh. Skriptsprachen kommt in Zusammenhang mit komponentenorientierter Softwareentwicklung oft die Rolle zu, als Verbindung (so genannter Glue Code) zwischen Komponenten zu fungieren [18]. Eine Anwendung wird automatisierbar genannt, wenn es möglich ist, sie durch Programmcode zu steuern und damit bestimmte Prozesse automatisch ausführt. Die LIMA Simulation läuft bereits automatisiert auf den Computingservern. Die Visualisierung soll auf den gleichen Rechnern lauffähig sein. Da die Computingserver alle mit UNIX Betriebssystemen laufen, bieten sich die Unix-Shellsprachen für die Lösung der Automatisierungs-

aufgabe an. IDL besitzt zum Programmieren eine Umgebung mit einer GUI. In ihr können programmierte Anwendungen kompiliert und ausgeführt werden. Unter UNIX wird diese Umgebung mit dem Kommando *idlde* aufgerufen. Das ist allerdings nur möglich wenn der User unter Windows eine Verbindung benutzt, die einen X-Server zur Verfügung stellt. Im IAP wird dafür die Software EXCEED von Humingbird © verwendet, mit deren Hilfe grafische Ausgaben der benutzten Programme auf dem remote-PC dargestellt werden. Um Programme auch ohne eine X-Windows Schnittstelle ausführen zu können, stellt IDL eine Kommandozeilenumgebung bereit. Sie wird mit dem Kommando *idl* aufgerufen. Diese Umgebung kann genauso benutzt werden wie die Kommandoeingabezeile in der IDL GUI. So wird beispielsweise ein Programm kompiliert und ausgeführt wenn man seinen Namen eingibt. Weiter ist es möglich für IDL ein Batchfile zu schreiben, das als Startupfile gesetzt werden kann. In ihm können Eigenschaften der IDL Laufzeitumgebung gesetzt werden, die beim nächsten Start von IDL berücksichtigt werden. So können automatisch Programme unmittelbar nachdem Start von IDL aufgerufen und ausgeführt werden. Das setzen des Startupfiles lässt sich nun wieder sehr gut mit einem Shellskript realisieren. Eine Automatisierung im Visualisierungsprogramm sollte die Aufgabenstellungen Filenamengenerierung, Visualisierung, laden/ speichern von Daten und Fehlermeldungenbehandlung abdecken. Für einen Gesamtüberblick der erzeugten Einzelbilder soll nachträglich ein Film erstellt werden. Dies sollte im selben Skribt so eingebunden werden, das nach der Generierung einer gewünschten Anzahl von Bildern ein Film erstellt wird. Für diese Aufgabe müssen wieder Programme verwendet werden, die aus der Kommandozeile bedient werden können.

3.1.5 Strömungsfilm

Die Vorlage für den angestrebten Strömungsfilm gibt eine Version die zurzeit beim Wetterbericht der ARD gezeigt wird (Abb. 3.5). Erstellt wird dieser Film von der Uni Basel, veröffentlicht durch die Website Meteoblue [13]. Die gewünschte Strömungsvisualisierung soll gleich des Vorlagefilms mit Temperaturen unterlegt sein. Eine Reliefkarte ist nicht vonnöten, da die jeweilige Visualisierung von Temperatur und Wind auf unterschiedlichen Höhenniveaus erfolgen soll. Die graphische Darstellungsform soll wie die der vorhergehend beschriebenen

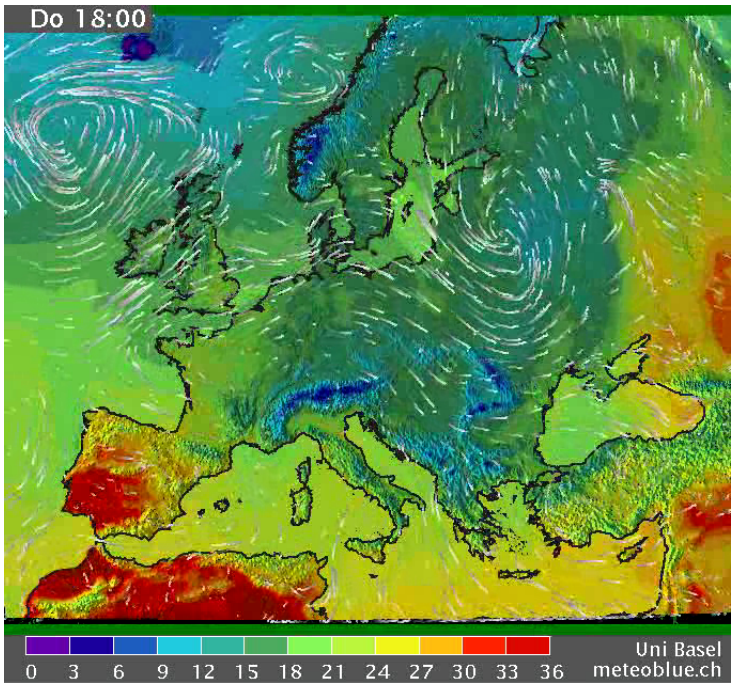

Abbildung 3.5: Beispielbild aus einem Strömungsfilm der Universität Basel der die Vorlage des gewünschten Strömungsfilm ist [13].

Simulationsdatenvisualisierung stereographisch erfolgen. Diese Visualisierungsweise der UNI Basel wurde ausgewählt, da sie derzeitig die beste Methode ist, diese beiden zweidimensionalen Größen gleichzeitig mit einer sehr hohen Informationsvermittlung darzustellen. Die Information, die dieser Visualisierung entnommen werden kann, wird so klar dargestellt, dass dieser Film sogar beim Wetterbericht für die breite Öffentlichkeit gezeigt wird. Die gleichzeitige Darstellung der meteorologischen Größen ergibt einen Informationsgewinn, der den Zusammenhang zwischen Wind und Temperatur verdeutlicht. Der Vorteil des Filmes liegt in der Darstellung der Strömung. Genauer in der Anzahl der Objekte, die den Wind darstellen, und wie sie sich verhalten. Durch die begrenzte Anzahl wird eine Überfüllung des Bildes mit Strömungslinien vermieden. Die Objekte, welche die Strömung darstellen, sind Linien. Standardmäßige Windpfeile besitzen eine feste Koordinate, die am Ende des Pfeils liegt. Der Pfeil zeigt für diesen einen festen Punkt in die Windrichtung und gibt die Windstärke mit seiner Länge an (Abb. 3.3). Bei einer Windrichtungsänderung dreht er sich um diesen Punkt. Diese Vektorpfeile zeigen also an, wie sich ein Partikel im Punkt weiterbewegen würde, wenn es sich in der Strömung bewegt. Der

Betrag des Vektors ist die Momentangeschwindigkeit am aktuellen Punkt. Eine weitere Möglichkeit sind Strömungspfeile, die den Weg des Windes beschreiben (Abb 3.6 a). Diese Wegbahnen nennt man Trajektorien [19]. Vorstellen kann man sich das so, dass die Bahn eines Luftpartikels in einem bestimmten Zeitraum dargestellt ist. Genauer gemeint ist damit die Verbindungslinie aller Orte, die das Partikel in einer Strömung im Laufe der Zeit erreicht [20]. Die Strömungspfeile besitzen als „Kopf" einen Pfeil um Anfang und Ende zu unterscheiden. Weiterhin zeigt der Pfeilkopf die letzte Position des Partikels an, dessen Weg beschrieben ist. Wie in Abbildung 3.6 a zu sehen ist, wirkt dieses Standbild an einigen stellen, an denen viele Pfeile übereinander gezeichnet sind schon sehr überladen. Wenn man mit dieser Methode eine Animation über einen Tag oder eine Woche erstellen würde, ist ein weißes Bild zu erwarten. Außerdem ist nicht zu vollziehen, an welcher Stelle des Bildes der Wind nun schneller oder langsamer ist. Die Methode der Trajektorienvisualisierung wird in dem Beispielfilm der UNI Basel auch verwendet, wie Abbildung 3.6 c zeigt. Mit dem Problem des übermäßigen Überlagerns der einzelnen Strömungslinien wird in dem Film so umgangen, dass das Weggedächtnis jedes einzelnen dargestellten Partikels nur eine bestimmte kurze Zeit anhält. Bei starker Vergrößerung (in Abb. 3.6 b) wird deutlich, dass die Linien nach kurzer Strecke enden. Weiter ist in Abbildung 3.6 b zu sehen, dass die Linien zum Ende hin die Farbe des Hintergrundes annehmen. Dies deutet auf einen Transparenzeffekt hin. Durch den offensichtlichen Sachverhalt, dass das Weggedächtnis jeder Linie dieselbe Zeitdauer anhält, wird ein relativer Geschwindigkeitseffekt der Linien zueinander erzeugt. Die Richtung der Strömung ist im Film durch die Bewegung der Linien offensichtlich, sodass Pfeile als Richtungsweiser überflüssig sind. Dies wird aber durch den Transparenzeffekt zum Ende der Linie noch verstärkt.

(a) (b) (c)

Abbildung 3.6: Bild 3.6 a zeigt Strömungspfeile in Kombination mit Trajektorien als Objekte für die Winddarstellung. Die Bilder 3.6 b und c verdeutlichen die Eigenschaften Trajektorienverwendung, zeitlich beschränktes Weggedächtnis und Transparenz zum Ende der Linien die Luftströmung im angestrebten Beispielfilm darstellen [13].

3.2 Gezeitenwellen-Analysetool

Die Erforschung atmosphärischer Wellen ist ein sehr komplexes Thema der Atmosphärenphysik, mit dem sich das IAP befasst. Interaktionen von einfachen harmonischen Wellen sind nur schwer vorstellbar, aber mit dem heutigen technischen Stand gut berechenbar. Aus diesem Grund wurde die Aufgabe zur Realisierung einer Software mit IDL gestellt, die diese Problemstellung simuliert und visualisiert. Ziel der Entwicklung einer solchen Software ist es, dem Benutzer zu ermöglichen, bei der Visualisierung mit den Daten interagieren, um sie besser zu verstehen und nachzuvollziehen.

3.2.1 Graphical User Interface

Das Akronym GUI bedeutet übersetzt Grafische Benutzerschnittstelle. Sie bezeichnet eine Softwarekomponente, die einem Benutzer die Anwendung eines Programms über grafische Elemente ermöglicht. Anforderungen an ein Programm mit einer GUI sind Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Fehlertoleranz und Erwartungskonformität [21]. Diese Entwicklungsrichtlinien der DIN-Norm für interaktive Systeme (DIN 66234), sind festgelegt worden, um eine allgemeine Anwendbarkeit solcher Systeme zu gewährleisten. Die Architektur eines solchen grafisch interaktiv bedienbaren Systems wird durch das Seeheim Schichtenmodell verdeutlicht (Abb. 3.7). Um GUIs mit IDL einzubinden, ist die Möglichkeit

der Verwendung von so genannten Widgets vorhanden. Widgets bezeichnen im allgemeinen die Interaktions- oder Steuerelemente in grafischen Benutzeroberflächen. Ein Nachteil von IDL ist, dass Programme die ausschließlich über eine GUI gesteuert werden, nur gestartet werden können in dem der Quellcode vorher in der Entwicklungsumgebung kompiliert und ausgeführt wird. Das bedeutet für den Anwender, dass auf dem Rechner, auf dem das Programm ausgeführt werden soll, eine Version von IDL installiert sein muss.

Abbildung 3.7: Seeheim Schichtenmodell [22].

3.2.2 Anforderungen an das Programm

Das Programm soll nicht aus dem Quellcode heraus bedient werden. Das heißt, dass es eine GUI benötigt. Es sollen 4 Wellen darstellbar sein, deren Eigenschaften separat eingestellt werden können. Die Welleneigenschaften Amplitude, Periodendauer, Wellenanzahl auf einem Beitenkreis und die Laufrichtung sollen durch Eingabeparameter festgelegt werden. Da die Software für Atmosphärenphysiker vorgesehen ist, die Gezeitenwellen betrachten möchten, die vom Tageszyklus abhängig sind, ist hier eine polarstereografische Projektion die beste Visualisierungsmöglichkeit. So können die Ausläufer einer solchen Welle, die longituginal[1] verläuft, auf der Tag und der Nachtseite gleichzeitig betrachtet werden. Die Stärke der Wellenamplitude sollte eindeutig farblich kodiert sein, sodass keine weiteren Informationen zum Bild nötig sind. Zusätzlich soll die polarstereografische Projektion mit Kontinentenum-

[1] entlang der Breitenringe von Ost nach West oder von West nach Ost

randungen und einem Gradnetz unterlegt werden. Die Bedienbarkeit muss auf den Anwender, der sich mit Gezeitenwellen beschäftigt, abgestimmt sein. Hinweise und Bezeichnungen im Programm müssen so gewählt werden, dass sie von der fokussierten Anwendergruppe schnell und einfach verstanden werden. Um ein zufriedenstellendes Ergebnis zu erzielen, muss bei der Ideenfindung und im Entwicklungsprozess des Programms ständig mit den potentiellen Anwendern Rücksprache gehalten werden. Die Grundidee für die Anwendbarkeit dieses Programms ist, dass nachdem der Anwender seine Einstellungen getätigt hat, er recht schnell eine Animation visualisieren kann. Dies setzt voraus, dass die Rechenzeit für die Erstellung der Animation so kurz wie möglich gehalten wird. Je größer und komplexer eine Animation jedoch gestaltet wird, desto mehr Rechenaufwand verursacht sie wiederum. Hier muss ein Konsens zwischen Qualität und Informationsgehalt der Animation für den Betrachter auf der einen Seite und vertretbarer Rechenzeit auf der anderen Seite gefunden werden. Die Animation soll in Einzelfällen auch direkt als separater Film gespeichert werden können. Außerdem sollen die Eingabeparameter nicht nur über die GUI eingegeben werden können, sondern auch aus externen Dateien geladen oder die vorhandenen Eingabeparameter in Dateien gespeichert werden.

Kapitel 4

Realisierung der Problemstellungen

Im folgenden Kapitel wird die praktische Umsetzung der Aufgabenstellungen beschrieben. Die Erläuterung der programmiertechnischen Umsetzung wird teilweise anhand von exemplarisch gewähltem Quellcode erfolgen. Zunächst wird auf die automatisierte Ergebnisvisualisierung eingegangen, dann auf den Strömungsfilm und zuletzt auf das Gezeitenwellenprogramm.

4.1 Automatisierte Ergebnisvisualisierung

Das aktuelle Programm der automatisierten Ergebnisvisualisierung liegt als IDL-Programm *hori110lesen7_6apollo.pro* vor. Es ist für die UNIX-Umgebung auf dem APOLLO Computingserver optimiert und wird dort mit der Versionsnummer 7.6 bereits genutzt. Um es über Skripte steuern zu können, wurden Eingabeparameter festgelegt, die dem Programm bei seinem Start übergeben werden. Die Menge der Ergebnisfiles des Limamodells beläuft sich auf 1460 pro simuliertes Jahr. In ihnen sind jeweils sechs verschiedene Daten auf 118 verschiedenen Höhenniveaus vorhanden, die alle für sich visualisiert werden können. Da dies eine Menge von Bildern ergibt, die einzeln von Hand nicht mehr ausgewertet werden kann, ist während der Implementierung des Programms entschieden worden, dass es möglich sein soll, aus den Bildern nachträglich einen Film zu erstellen. Das setzt voraus, dass alle Bilder einer Größe in einem bestimmten Zeitraum die gleichen Wertebereichsgrenzen

besitzen. Um diese Grenzen zu bestimmen, wurde ein zusätzliches Programm *werte_ermitteln.pro* geschrieben, das die Wertebereichsgrenzen in Dateien bereitstellt.

4.1.1 Abstraktion

Für einen Gesamtüberblick werden die Programmfunktionen zunächst grob dargestellt. Später werden dann die einzelnen Prozeduren und Funktionen für sich betrachtet. Eine Abstraktion wird im Datenflussschema in Abbildung 4.1 dargestellt. Das Programm besteht aus dem Hauptprogramm und jeweils einem Unterprogramm für jede zu visualisierende Größe.

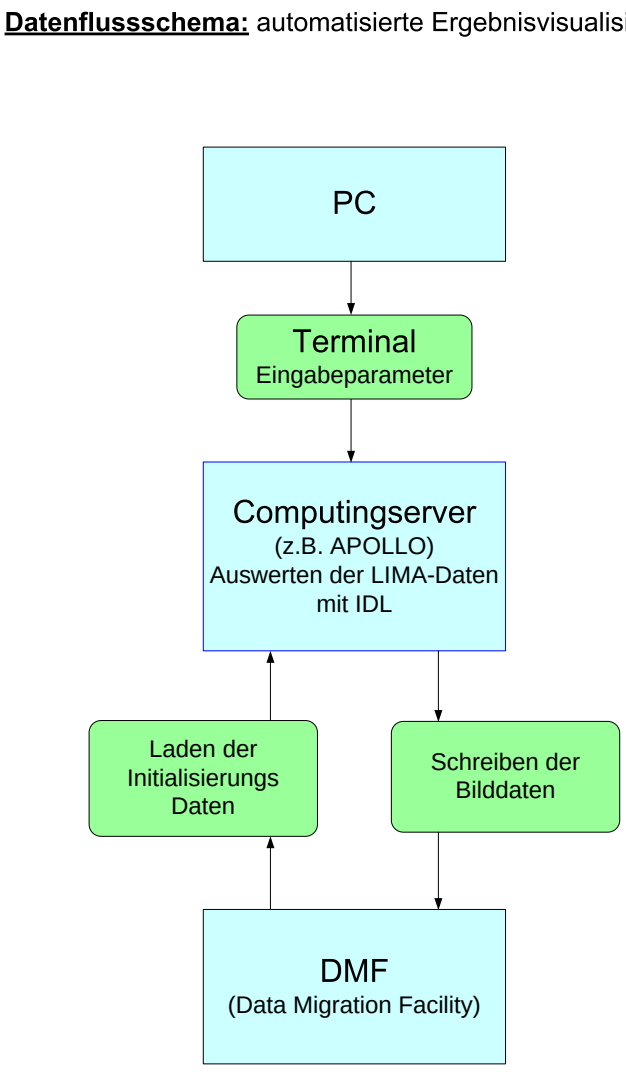

Abbildung 4.1: Datenflussschema der automatisierten Ergebnisvisualisierung.

Eine Schleife im Hauptprogramm lädt alle benötigten Daten und ruft die Unterprogramme für die gewünschten Visualisierungen auf. Dieser Programmaufbau wurde gewählt, da die einzelnen Größen der LIMA-Ergebnisdaten zum Teil eine individuelle Verarbeitung voraussetzen, um zu dem gewünschten Visualisierungsergebnis zu gelangen.

4.1.2 Vorbereitung: Wertebereiche ermitteln

Zur Optimierung der Darstellung muss zunächst der Wertebereich der darzustellenden Daten ermittelt werden. Das Programm *werte_ermitteln.pro* ist mit IDL und für den APOLLO Computingserver umgesetzt worden. Es liest die Werte aus den LIMA-Datenfiles ein, filtert die jeweils größten und kleinsten Werte für jede Größe des LIMA-Modells und für jedes Höhenniveau aus. Die endgültigen Werte werden in der Form einer Tabelle in ein ASCII-File geschrieben und gelten für ein komplettes Jahr. Das Auslesen der einzelnen Dateien erfolgt ähnlich wie im Visualisierungsprogramm in einer Schleife, in der auch die Dateinamen generiert werden. Nachdem eine Datei ausgelesen ist, wird in einer weiteren Schleife, die alle 118 Höhenniveaus durchläuft, der maximale und minimale Wert für jede Größe mit den Filterfunktionen *MAX* und *MIN* ermittelt. Dieser wird dann in einem *maxminarry* festgehalten. Bei jedem erneute Auslesen einer Datei werden die Daten im *maxminarry* überschrieben wenn die aktuellen Werte größer bzw. kleiner sind als die vorhandenen. Am Ende des Programms wird das Feld in eine Datei geschrieben. Die Werte in dem Maxminfile sind so angeordnet, dass es möglich ist sie von Hand zu überprüfen. Um mit dieser Methode das exakte Maximum bzw. Minimum über den Zeitraum eines Jahres zu ermitteln müsste jedes einzelne LIMA-Datenfile ausgelesen werden. Da das aber einen sehr großen Rechenaufwand darstellt, wurden statistisch für das gesamte Jahr jeweils die vier Datenfiles des ersten Tages, jeden Monats gewählt. Durch zukünftige Erfahrungen mit den vorliegenden Ergebnissen kann dieses Verfahren der Wertebereichsermittlung noch verfeinert werden. Das bedeutet, dass die Zeitpunkte der Stichproben für die Wertebereichseingrenzung verändert und eventuell auch mehr Zeitpunkte gewählt werden müssen.

4.1.3 Farbpaletten

In den Prozeduren werden zu allererst die Farben für den Plot festgelegt. Dafür wurden zwei Möglichkeiten umgesetzt. Eine ist, eine von IDL bereitgestellte und vordefinierte Farbtabelle zu laden. Die andere Möglichkeit ist, eine individuelle Farbtabelle zu definieren. Bei der ersten Variante wird die gewünschte Farbtabelle wie im folgenden Quellcodebeispiel in Zeile 8 geladen, die einzelnen Farbvektoren der Tabelle in Zeile 11 ausgelesen und sie so angepasst, dass die Farben Schwarz und Weiß zur Verfügung stehen. Durch *TVLCT* werden die veränderten Farbvektoren als aktuelle Farbtabelle im laufenden Programm verwendet.

```
8  loadct,39
9  ;----------------------------Farben wei{\ss} und schwarz festlegen
10 ;Farbvektoren der geladenen Farbtabelle auslesen
11 COMMON COLORS, R_orig, G_orig, B_orig, R_curr, G_curr, B_curr
12 ;print,R_orig, G_orig, B_orig, R_curr, G_curr, B_curr
13
14 R_curr[1] = 0        ;Hintergrundfarbe definieren
15 G_curr[1] = 0
16 B_curr[1] = 0
17 R_curr[0] = 255      ;Schriftfarbe  definieren
18 G_curr[0] = 255
19 B_curr[0] = 255
20
21 TVLCT,R_curr, G_curr, B_curr ;geladene Farbtabelle mit geaenderten Werten
       redefinieren
```

Die von IDL standardmäßig verwendete Farbtabelle ist eine RGB-Farbtabelle mit 256 möglichen Farben, die in Abschnitt 2.5.3 schon beschrieben wird. Jeder Vektor kann 256 ganzzahlige Werte aufnehmen, die im Bereich von 0 bis 255 liegen sollten. Wenn ein Wert in der Farbtabelle über 255 liegt, teilt die IDL Laufzeitumgebung ihn durch 255 und interpretiert den ganzzahligen Rest als Farbwert. IDL stellt intern 41 verschiedene, vordefinierte Farbtabellen zur Verfügung. Diese sind nicht für jede Visualisierung geeignet. Die durch Tests ermittelten Farbtabellen, die für dieses Programm in Frage kommen, sind Nummer 26, 33 und 39 für die Darstellung von Temperatur und geografischer Höhe, sowie Nummer 11, 12, 17, 18 und 37 für die Darstellung von Dichte, Zonalwind und Meridionalwind. Für die Darstellung des Vertikalwindes ist beispielsweise keine der vordefinierten Farbtabellen geeignet. Für diese Problemstellung wird nachfolgend erörtert, wie in IDL eine individuelle Farbtabelle erstellt werden kann.

Um eine optimale Visualisierung des Vertikalwindes zu erreichen, wurden die Bedingungen der zu erstellenden Farbtabelle festgelegt. Der Vertikalwind ist in negative und positive Werte aufgeteilt. Die Wertebereichsgrenzen sind in beide Richtungen ungefähr gleich groß. Das bedeutet, dass die Farbtabelle genau in der Mitte Farblich getrennt sein muss. Weiter ist bei der Visualisierung eine deutliche Markierung des Nullbereiches gewünscht. Die Implementierung der Lösung sieht so aus, dass drei Felder R, G und B deklariert werden. Deren Elemente besitzen die Werte ihrer eigenen Indexierung. So wird es möglich, die Elemente mit der Filterfunktion *WHERE* anzusprechen. Im folgenden Quellcodebeispiel wird die Definierung des Vektors für den grünen Farbanteil in der Farbtabelle gezeigt.

```
;GRUEN-------------------------------------------------------------------------------


G1 = G
G2 = G
G3 = G

; erste GRUEN funktion (anstieg)
     ....................................................

;alle werte in G1 um den 75. - 125. index null setzen
index = Where ((G1 LT 75) OR (G1 GT 125),count)
if count GT 0 THEN G1[index] = 0

index = Where ((G1 GE 74) AND (G1 LT 126),count)
if count GT 0 THEN G1[index]= fix(255.* sin(float(index - 75.)/32.)) ;

index = WHERE(G1 LT 0, count)  ;negative werte null setzen
if count GT 0 Then G1[index] = 0

; zweite GRUEN funktion (maximum)
     ....................................................

;alle werte in G2 um dem 125. - 175. index null setzen
index = Where ((G2 LT 125) OR (G2 GT 199),count)
if count GT 0 THEN G2[index] = 0

index = Where ((G2 GE 124) AND (G2 LT 200),count)
if count GT 0 THEN G2[index]= 255

index = WHERE(G2 LT 0, count)   ;negative werte null setzen
if count GT 0 Then G2[index] = 0

```

```
2224 ; dritte GRUEN funktion (abfallen)
          ....................................................
2225
2226 ;alle werte in G1 um dem 175. - 225. index null setzen
2227 index = Where ((G3 LT 200) OR (G3 GT 250),count)
2228 if count GT 0 THEN G3[index] = 0
2229
2230 index = Where ((G3 GE 199) AND (G3 LT 251),count)
2231 if count GT 0 THEN G3[index]= fix(255.* sin(float(index + 51.)/32.)) ;- 60
2232
2233 index = WHERE(G3 LT 0, count)   ;negative werte null setzen
2234 if count GT 0 Then G3[index] = 0
2235
2236 G = G1 + G2 + G3
```

Bevor die Farbvektoren mit Farbwerten gefüllt werden, muss klar sein, wie sich ihre grundlegende Mischung verhält, die in Abschnitt 2.5.3 betrachtet wird. Eine geeignete Farbmischung wird empirisch ermittelt. Die besten Ergebnisse für Farbübergänge wurden bei verschiedenen Versuchen mit Elementen homogener Schwingungen erziehlt. Konkret bedeutet das, dass die Werteflanken und auch ganze Kurven in einem Farbkanal mit Sinusfunktionen realisiert werden. Eine komplette Kurve kann dabei aus mehreren Teilstücken bestehen, wenn sie unterschiedliche Werteverläufe vollzieht. Wenn der Anstieg einer Kurve z.B. steiler ist als ihr Abstieg, wird für jede Kurvenphase eine andere Funktion definiert. Bei dem vorangegangenem Quellcodebeispiel ist die Einteilung in drei Kurvenabschnitte notwendig, wie in Abbildung 4.2 nachvollzogen werden kann. Das sind der Anstieg, der lineare Verlauf und der Kurvenabstieg. Dazu werden in den Zeilen 2196 bis 2198 drei Felder mit dem zuvor deklarierten G Feld initialisiert. Anschließend werden die jeweiligen Felder mit Werten der separaten Kurvenabschnitte gefüllt. Mit Hilfe der *WHERE*-Funktion werden wie in Zeile 2203 zunächst die benötigten Feldeinträge mit den Indexwerten herausgefiltert. Das geschieht, indem durch die Filterfunktion ein weiteres Indexfeld (count) der selben Länge angelegt wird. Für die Werte in dem G1 Feld, die der Bedingung der *WHERE*-Funktion entsprechen, werden die Einträge in dem zusätzlichen Indexfeld eins gesetzt, die anderen null. In der If-Abfrage nach aufrufen der *WHERE*-Funktion werden die „indexierten“ Feldeinträge so behandelt, wie es die folgende Anweisung in z.B. Zeile 2216 vorgibt. Wie der aufsteigende Kurvenabschnitt definiert wird, ist in Zeile 2207 zu sehen. Der Wert *255* gibt das Maximum

der Amplitude an, der Wert *-75* verschiebt den Beginn der Sinusfunktion bis zu dem Eintrag, an dem sie Starten soll. Der Wert 32 gibt an, wie oft die Differenz des darzustellenden Bereichs durch $\frac{\pi}{2}$ teilbar ist. So wird die „viertel“ Sinusfunktion auf diesen Bereich gestreckt. Nach der Wertezuweisung anhand der Sinusfunktion erfolgt eine Sicherheitsabfrage in den Zeilen 2209 und 2210, um eventuell entstandene Negativwerte zu verhindern. Diese Funktionsabfolge wird für jedes Teilstück der Kurve durchlaufen. Nachdem die Kurve komplett ist, werden die Felder in Zeile 2236 mit den jeweiligen Teilstücken miteinander addiert und in dem Feld für den grünen Farbanteil der Farbpalette festgehalten. Das geschieht wiederum für jeden einzelnen Farbkanal Rot, Grün und Blau. Die drei Farbvektoren werden dann durch *TVLCT* als aktuelle Farbtabelle im laufenden Programm zur Verfügung gestellt. Der Verlauf der einzelnen Funktionen der Farbtabelle, die für die Visualisierung des Vertikalwinds definiert wurde, sind in Abbildung 4.2 verdeutlicht. Zusätzlich ist analog zum Funktionsgraphen die damit erstellte Farbtabelle zu sehen.

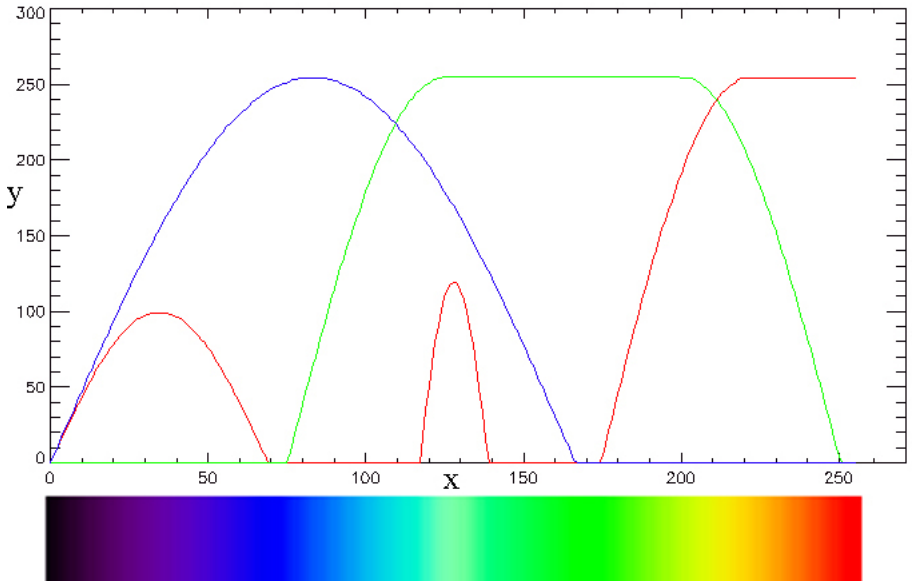

Abbildung 4.2: Farbbalken für die Vertikalwindvisualisierung. Oben: Funktionsverläufe für die drei Farbvektoren R, G und B. Die X-Achse stellt den Farbtabellenindex dar und Y die Intensität der jeweiligen Grundfarbe. Unten: Der Farbbalken, der aus den Funktionen resultiert

4.1.4 Hauptprogramm

Das gesamte Programm besteht aus dem Hauptprogramm *hori110lesen7_6apollo.pro* und den Prozeduren *plot_zonalwind, plot_meridionalwind, plot_temperatur, plot_vertikalgeschw, plot_hoehe* und *plot_dichte* die, die einzelnen Simulationsergebnisgrößen visualisieren. Im Hauptprogramm werden die benötigten Daten wie z.B. das Koordinatenfile, das Wertebereichsfile und das eigentliche Datenfile vom DMF gelesen. Diese werden dann formatiert und Feldern zugewiesen die den entsprechenden Prozeduren beim Aufruf übergeben werden. Dies geschieht in einer Schleife, deren Länge abhängig von der Anzahl der zu Visualisierenden Datenfiles ist (Abb. 4.3). Das Hauptprogramm erhält beim Start auf der Apollo Eingabeparameter. Diese Parameter beeinflussen welche Größen visualisiert werden sollen, in welchem Zeitraum, wie sie visualisiert werden sollen, und legen die Pfade für die Ein- und Ausgabedaten fest.

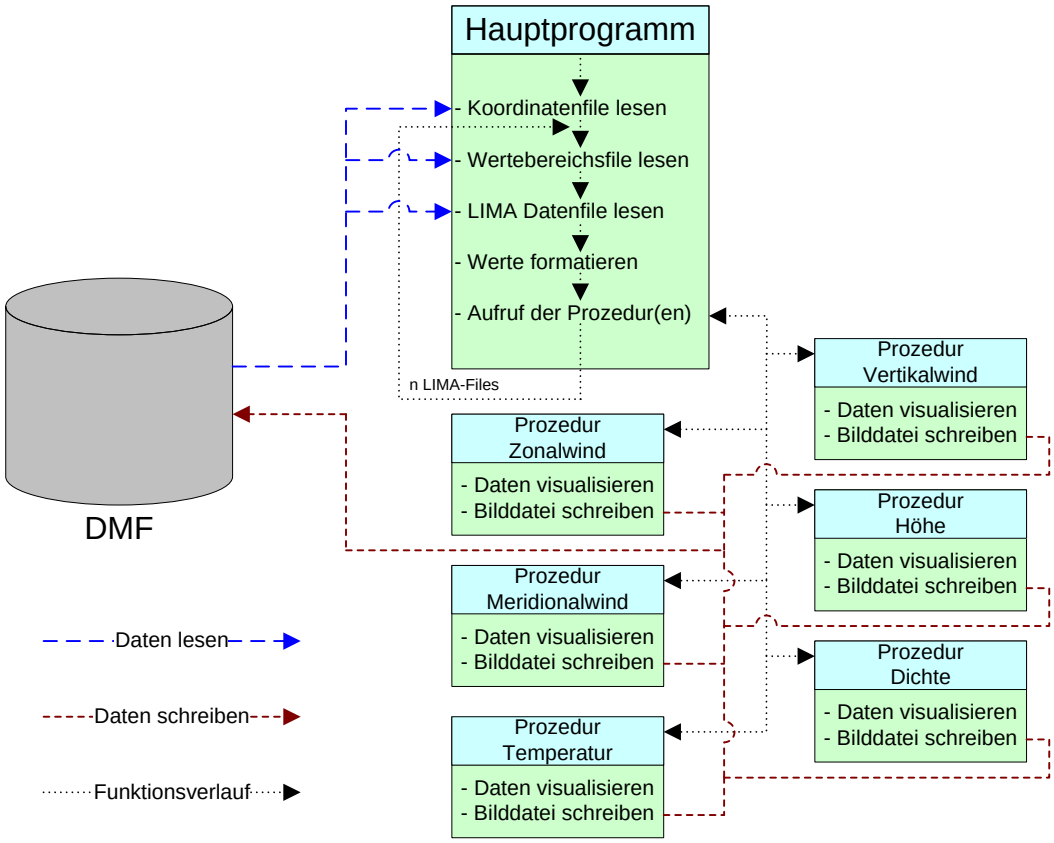

Abbildung 4.3: Funktionsschema der automatisierten Ergebnisvisualisierung.

Im Einzelnen sind diese Parameter:

- *Auswahl der zu visualisierenden Daten*
  Zur Auswahl stehen die Größen des LIMA-Modells, Zonalwind, Meridionalwind, Temperatur, Vertikalwind, geometrische Höhe, Dichte im Einzelnen und die Möglichkeit alle Größen zu visualisieren. Weiter kann gewählt werden auf welchem Höhenniveau die Größen betrachtet werden sollen.

- *Art der Visualisierung*
  Es besteht die Möglichkeit, die Daten getrennt als Konturplot und als Direktplot oder als Kombination auszugeben. Für den Konturplot ist es möglich, ihn mit oder ohne Kontinentenumrandungen zu visualisieren. Weiter steht zur Auswahl, diese Visualisierungsvarianten als Nord- oder Südpolarstereographische, oder beide Projektionen zu wählen.

- *Art der Ausgabe*
  Gewählt werden kann zwischen Vektorgrafikdatenformat, Bitmapdatenformat und einer Bildschirmausgabe durch einen X-Server.

- *Zeitraum der Visualisierung*
  Es kann der Zeitraum gewählt werden, für den die Größen Visualisiert werden sollen. Das ist durch festlegen von Anfangs- und Endedatum möglich.

- *Festlegen der Pfade für Ein- und Ausgabedaten*
  Die Pfade werden direkt als Strings angegeben. Für die Simulationsergebnisdateien, die Dateien mit den Wertebereichsgrenzen (s. Abschnitt 4.1.2), die PostScript-Bilder und die Bitmap-Bilder wird nur der Ordnerpfad angegeben. Für die Datei mit den Koordinateninformationen (koordfile) muss der der komplette Pfad mit Dateinamen und Endung angegeben werden.

Im Folgenden wird ein IDL Beispiel für ein Batchfile gegeben in dem die Eingabeparameter festgelegt werden.

```
1
2 SET_PLOT,'Z'
3
4 hemis = 2;          -> Suedhemisphaere        =1
5 ;                   -> Nordhemisphaere        =2
```

```
;                   -> beide Hemisphaeren    =3
nvar = 3;           -> Zonalwind (OW)        =1
;                   -> Meridionalwind (NS)   =2
;                   -> Temperatur            =3
;                   -> Vertikalwind          =4
;                   -> Hoehe in km           =5
;                   -> Dichte                =6
;                   -> alle moeglichen Plots =7
kakt = 2;           -> Hoehe (auf gleichem Druckniveau)
;                      k=36 (=3 mbar), k= 84 Mesopause,
;                      k= 75 nlc Hoehe, 1. Hoehe (=3 mbar)
;                      aus ecmwf stimmt mit LIMA k=36
;                      ueberein!
scr = 3 ;           -> 1 fuer Bildschirm (nicht im Batchbetrieb!),
;                      2 fuer ps ,3 fuer BMP
plotart = 1 ;       -> 1 fuer gefuellten Konturplot,
;                      2 fuer Direktplot mit Polygonen,
;                      3 fuer beides
cnt = 2;            -> Konturplot mit Kontinenten cnt=2, ohne cnt=1

b_jear = 2006       ;Anfangsdatum
b_mon  = 1
b_day  = 1
e_jear = 2006       ;Enddatum
e_mon  = 8
e_day  = 15

;kompletten Ladepfad fuer das koordfile angeben
fnkoord = String('~/bin/lima/limplo/anfcommon110.dat')

;Ladepfad fuer die Datenfiles angeben
fndata = String('~/dmf/LIMA/LIMA-DATA/global/')

;Ladepfad fuer die Maxminfiles angeben
fnmaxmin = String('~/bin/lima/limplo/MAXMIN_WERTE')

;Speicherpfad fuer die PS-Bilder
fnps = String('~/dmf/plots/wdir/ps/')

;Speicherpfad fuer die Bitmap-Bilder
fnbitm = String('~/dmf/plots/2006/NPS/T_002/')

hori110lesen7_6apollo,hemis,nvar,kakt,scr,plotart,cnt,b_jear,b_mon,b_day,$
    e_jear,e_mon,e_day,fnkoord,fndata,fnmaxmin,fnps,fnbitm

retall
exit
```

Im Batchfilebeispiel Zeile 2 ist zu sehen, dass die IDL Displayvariable auf den

Z-Buffer gesetzt wird. Das ist nötig, da IDL beim Start standardmäßig eine X-Windowsschnittstelle anspricht. Da diese Schnittstelle unter UNIX im Batchbetrieb nicht vorhanden ist, würde das Programm nach der ersten Grafikoperation abbrechen. Nachdem das Programm gestartet ist, wird überprüft, ob alle nötigen Eingabeparameter übergeben wurden. Wurden keine Parameter übergeben oder fehlen teilweise Parameter in der Übergabe, werden vordefinierte Standardwerte benutzt. Nach dieser Abfrage erfolgt eine weitere, die überprüft, ob die übergebenen Werte im vorgegebenen Wertebereich liegen. Ist das nicht der Fall, so wird eine Fehlermeldung mit dem Hinweis auf die entsprechende Variable und den Wertebereich ausgegeben und das Programm beendet. Danach erfolgt die Deklaration der für das Hauptprogramm benötigten Felder und Variablen. Hier werden zunächst Variablen mit konstanten Werten deklariert, die oft gebraucht werden, so dass z.B. bei den späteren Felddeklarationen nur noch die entsprechenden Variablennamen verwendet werden. Diese Werte sind die Gitterpunktanzahl des vom LIMA-Modells benutzten 110er Dreiecksgitter, die Hälfte der Gitterpunktanzahl für die Hemisphärendarstellung, die Anzahl der Höhenniveaus und die Anzahl der Breitenringe auf dem Globus in Eingradabschnitten. Weiterhin werden hier die Anzahlen der Tage in den Monaten eines normalen und eines Schaltjahrs in einem Feld festgelegt, wie im nachfolgenden Quellcodebeispiel gezeigt wird. Die Zeilennummerierungen sind im Hauptprogramm höher als in den Unterprogrammen weil in IDL die letzte Prozedur in einer Quellcodedatei die Hauptprozedur sein sollte. Der Compiler übersetzt ein Programm bei seinem ersten Aufruf sequentiell bis zur Hauptprozedur, die logisch als Filename des Programms vergeben wurde. Eventuell nachfolgende Routinen werden ignoriert [23].

```
4689 igitnb_v = 41804L           ; Gesamtgitterpunktanzahl des Modells
4690 igitnb_v2= igitnb_v / 2     ; Gesamtgitterpunktanzahl einer Hemisphaere
4691 kgit     = 118              ; Anzahl der Hoehen (Auswahl -> kakt)
4692 nb       = 180              ; Anzahl der Breitenringe (Globus)
4693
4694 ;Anzahl der tage je Monat im Schaltjahr o. Nichtschaltjahr
4695 daycount = [[31,28,31,30,31,30,31,31,30,31,30,31],$
4696             [31,29,31,30,31,30,31,31,30,31,30,31]]
```

Nach den Feld und Variablendeklarationen wird das Koordinatenfile eingelesen. Dateioperationen ähneln in IDL sehr denen in Fortran. Das bedeutet, dass man eine Datei zum Lesen oder Schreiben erst öffnen muss, nachdem man die Datei gelesen oder geschrieben hat, muss man diese dann wieder schließen. Hinter dem jeweiligem

Befehl der Dateioperation muss eine Unitnummer angegeben werden. Diese wird für Ein- und Ausgaberoutinen benötigt, um sie bei mehreren geöffneten Dateien, korrekt zuordnen zu können. Der Pfad und Dateiname wird durch die Stringvariable *koordfile* angegeben. In ihr ist die Pfadangabe festgehalten die dem Programm übergeben wurde. Komfortabel ist in IDL, dass mit Schlüsselwörtern (Keywords) auf Dateiformate zugegriffen werden kann, die keinen Standard darstellen. Das LIMA-Modell beispielsweise ist in Fortran implementiert und legt die Ergebnisdateien in einem eigenen Binär-Format ab. Mit dem Keyword *F77_UNFORMATTED* können diese Dateien ausgelesen werden. Da das Programm teilweise unter Windows entwickelt wurde, ist zu erwähnen, dass das HP-UX Betriebssystem unter dem die Daten angelegt wurden, ein anderes Byteformat verwendet als unter Windows, nämlich das Big Endian Format. Wenn das Programm von Windows aus gestartet wird, ist zu beachten, dass das Keyword *SWAP_ENDIAN* benutzt werden muss, um die Daten lesen zu können. In dem Koordinatenfile sind außer den Daten, die in dem Programm benötigt werden, noch weitere für dieses Programm aber unrelevante Daten enthalten. Das folgende Quellcodebeispiel zeigt, wie das Koordinatenfile ausgelesen wird. Die für dieses Programm relevanten Daten sind *igit_v* für die Anzahl der Gitterpunkte auf jedem Breitenring, *v_kordi_x* und *v_kordi_y* für die Koordinaten jedes einzelnen Gitterpunktes. Die nicht relevanten Daten müssen jedoch mit ausgelesen werden, um an die relevanten Daten zu kommen. Das liegt an der Anordnung der Daten im LIMA-Fileformat. Beim Schreiben wurden die unterschiedlichen Daten aus diversen Feldern eines Programms hintereinander weg in das Koordinatenfile geschrieben. Die Daten liegen in der Datei wie in einem einzigen Feld vor, das aus den Daten vieler Felder unterschiedlicher Daten besteht. Um die Datei auslesen zu können, muss die Reihenfolge, die Anzahl und die jeweilige Größe der Felder bekannt sein.

```
4780 ;->/SWAP_ENDIAN fuer einlesen unter WINDOWSsystemen
4781 OPENR, 12, koordfile, /F77_UNFORMATTED;, /SWAP_ENDIAN
4782 READU, 12, dy,dx,phi_v,cosfi,cosf2
4783 READU, 12, igit_v,ianf_v,iend_v,winkelweite_v,v_kordi_x,v_kordi_y
4784 READU, 12, iv_p2,iv_p4,jv_vp2,wjv_vp2,jv_vp4,wjv_vp4
4785 CLOSE, 12
```

Das Koordinatenfile muss nur einmal gelesen werden, da die Gitterpunkte ihre Position nicht verändern. Die Visualisierung der LIMA-Daten findet in Prozeduren

statt. Diese Prozeduren werden in einer Schleife aufgerufen, die für jedes LIMA-Datenfile einmal durchlaufen wird. Wie oft diese Schleife durchlaufen wird, ist wiederum durch die Zeitgrenzen festgelegt, die dem Programm übergeben werden. Eine Indexvariable leitet die Beendigung der Schleife ein, wenn sie einen bestimmten Wert annimmt. Dieser Wert wird gesetzt, wenn das letzte mögliche LIMA-File des Endedatums ausgelesen und visualisiert worden ist. In der Schleife werden zunächst die Pfade und Filenamen der LIMA-Files und der Maxminfiles generiert. Der Pfad für die Lima-Files wird bei der Übergabe nur bis zu einem bestimmten Ordnerverzeichnis angegeben. Ab diesem Ordnerverzeichnis sind die Unterordner und Filenamen durch Datumsangaben strukturiert. Die Pfadangabe, die im aktuellen Programm übergeben wird, ist ab dem Homeverzeichnis des Users */dmf/LIMA/LIMA-DATA/global/*. Die vollständige Pfadangabe für ein LIMA-File ist aber */dmf/LIMA/LIMA-DATA/global/2005/01/20050101.00.bin*. Zu Beginn der Schleife wird überprüft, ob das aktuell auszulesende Jahr ein Schaltjahr ist. Schaltjahre treten alle vier Jahre einmal auf und alle 400 Jahre bei Jahrhundertüberschreitungen. Insbesondere auf das Jahr 2000 traf beides zu. Die LIMA-Files liegen zurzeit in einem Zeitraum von 1997 bis 2006 vor, wobei das ausserordentliche Jahrhundertschaltjahr 2000 hier vernachlässigt werden konnte. Die Daten für die Anzahl der Tage in den Monaten eines Schaltjahres und Nichtschaltjahres sind in einem zweidimensionalen Feld festgelegt und werden je nach eintreffen des Zustandes von dort bezogen. Im folgenden Abschnitt der Schleife werden die Pfad und Filenamen generiert. Dazu werden die Datumsangaben in Strings umgewandelt, wie im folgenden Quellcodebeispiel gezeigt wird.

```
4824 ;Einzulesender Datenfile des Modells
4825
4826 loadstring11 = string (fix(jahr))
4827 loadstring12 = string (monat)
4828 loadstring13 = string (tag)
4829 loadstring14 = string (tageszeit)
4830 nullstring   = ('0')
4831
4832 loadstring11 = strtrim(loadstring11,2)
4833 loadstring12 = strtrim(loadstring12,2)
4834 loadstring13 = strtrim(loadstring13,2)
4835 loadstring14 = strtrim(loadstring14,2)
```

Danach wird durch eine Abfrage, wenn die Tagesnummerierung kleiner Zehn ist,

der *nullstring* vor der Nummerierung eingefügt und die unterschiedlichen Strings, die dann mit unterschiedlichen Stringoperationen zum gesamten Ladepfad zusammengefügt. Hier wird auch gleich der Teil der Plotbeschriftung erzeugt, der später im Bild den jeweiligen Plot über die Datumsanzeige identifizierbar macht. Weiter erfolgen Abfragen, die Teile der Datumswerte um Eins erweitern oder sie auf den jeweiligen Anfangswert zurücksetzen. Das heißt, dass wenn z.B. das aktuelle Datum den letzten LIMA-File eines Jahres darstellt, muss der Jahreswert für den nächsten Schleifendurchlauf um Eins erhöht werden. Der Monatswert und der Tageswert müssen wiederum auf Eins und der Tageszeitwert auf Null gesetzt werden. An einem simulierten Tag sind vier Files im Abstand von sechs Stunden erstellt worden, sodass die Tageszeit die Werte 00, 06, 12 und 18 annehmen kann. Nachdem die Generierung der LIMA-Datenfiles und die Bearbeitung der Datumswerte für den nächsten Schleifendurchlauf abgeschlossen sind, werden die Filenamen der Maxminfiles generiert. Der Ladepfad der Maxminfiles wird wie schon bei den LIMA-Datenfiles nur bis zu einem bestimmten Ordnerverzeichnis übergeben. Die Files sind in jeweils einem zusätzlichen Unterorder gespeichert, der die Jahreszahl darstellt, in dem das Maxminfile gültig ist. Der vollständige Pfad für ein Maxminfile ist z.B. */bin/-lima/limplo/MAXMIN_WERTE/2005/lima_maxmin_2005.dat*. Zurzeit ist in einem Jahresordner ein File vorhanden, das die Wertebereichsgrenzen der zu visualisierenden Größen beinhaltet. In diesem Zustand erscheint die Ordnerstruktur jedoch nicht sinnvoll. Dies ändert sich, wenn mehrere Files angelegt werden. Das wird notwendig, wenn z.B. LIMA-Filegrößen Jahreszeitabhängig gravierende Unterschiede in ihren Wertebereichsgrenzen aufweisen. Ein Beispiel dafür wäre, wenn visualisierte Temperaturunterschiede im gewählten Jahr im Winter viel größer sind, als im Sommer. Dann wären die Sommerbilder u.U. mit nur einer einzigen Farbe ausgefüllt, da durch den großen Wertebereich die geringen Temperaturunterschiede farblich nicht mehr aufgelöst werden können. Im Folgenden wird das Maxminfiles und das LIMA-Datenfile ausgelesen.

Das Maxminfile wird im selben Format eingelesen, wie es geschrieben wurde. Maximalwert und Minimalwert für jede LIMA-Größe mal 118 Höhen. Weil die Wertebereichsgrenzen für einen Abschnitt von einem Jahr immer gleich bleiben, wird das Maxminfile nur einmal beim ersten Durchlauf der Schleife ausgelesen und wenn eine

Jahresgrenze überschritten ist. Für das LIMA-Datenfile gilt beim Auslesen dasselbe wie für das Koordinatenfile. Der Parameter, der hier mit ausgelesen wird aber für das Programm unrelevant ist, ist *ncom*. Das ist eine Variable vom Typ Long. Sie ist ein Zeitschrittzähler der als Kontrollwert für den Fortschritt der Simulation dient. Das folgende Quelltextbeispiel zeigt die Leseroutinen, mit denen das Maxminfile und das LIMA-Datenfile ausgelesen werden.

```
4973 ; Einlesen der Daten
4974 ; ==================
4975 ;MAXMINFILE----------------------sollte nur einmal pro Jahr ausgelesen werden
4976
4977 if vorf_jahr NE akt_jahr THEN BEGIN
4978 maxmin_err = 1          ;falls leseerror auftritt -> verursacht durch diese routine
          ?
4979
4980 OPENR, 14, maxmin_file  ; ->/SWAP_ENDIAN fuer einlesen unter WINDOWSsystemen
4981 READF, 14, r_max_u,r_min_u,r_max_v,r_min_v,r_max_w,r_min_w,r_max_t,r_min_t,
          r_max_zgeo,r_min_zgeo
4982 CLOSE, 14
4983
4984 maxmin_err = 0   ;wenn diese routine den leseerror verursacht hat wird maxmin_err
          nicht null gesetzt
4985
4986 endif
4987
4988 ;LIMA DATEN---------------------
4989 datafile = loadpath
4990
4991 OPENR, 11, datafile, /F77_UNFORMATTED;->/SWAP_ENDIAN fuer einlesen unter
          WINDOWSsystemen
4992 READU, 11, ncom
4993 READU, 11, un1, vn1, tn1
4994 READU, 11, zgeo
4995 READU, 11, dichtecgs
4996 READU, 11, wn1
4997 CLOSE, 11
```

Da beim Lesen vom DMF immer wieder Probleme auftauchen, wie in Abschnitt 3.1.3 schon beschrieben wurde, ist es notwendig diese Probleme der langen Ladezeiten, die durch den DMF verursacht werden zu behandeln, sodass ein Programmabsturz vermieden wird. Dazu wurde vor den beiden Leseroutinen ein Errorhandler implementiert. Im vorangegangenen Quellcodebeispiel ist zu sehen, dass zur Unterstützung der Errorbehandlungsroutine die Variable *maxmin_err* eingefügt worden ist. Sie soll im Errorhandler helfen, die Leseroutine zu identifizieren, in der der Fehler verursacht

wurde. Der Befehl *CATCH* macht es in IDL möglich, einen Errorhandler zu schreiben, der aufgerufen wird, sobald ein Errorcode von IDL ausgelöst wird. Der Error-Status wird in IDL zu Beginn des Programms mit Null initialisiert. Um einen Error zu behandeln, ist die Anweisung *CATCH,error_status* notwendig. Wenn ein Fehler auftritt, wird im IDL Calling-Stack nach *CATCH* gesucht. Ist *CATCH* vorhanden, werden die auf *CATCH* folgenden Statements ausgeführt. Mit *CATCH,/CANCEL* wird der aktuelle CATCH-Mechanismus aufgehoben[24]. Im nachfolgendem Quellcodebeispiel ist der Errorhandler zu sehen, der die Lesefehler, die vom DMF verursacht werden behandeln soll.

```
4929 ;Errorhandling
4930 errindex = 0            ;Kontrollindex
4931 CATCH,error_status      ;Error abfangen
4932
4933 ;bekannte error codes : Unix(-247, -258); Windows(-263, -250)
4934
4935 if error_status ne 0 THEN BEGIN ;Errorhandler
4936
4937 ;spezielle Fehlercodes abfragen und behandeln
4938 if (error_status eq -247) or (error_status eq -258) THEN BEGIN
4939     PRINT, 'Error index: ', Error_status
4940     PRINT, 'Error message:', !ERR_STRING
4941
4942     errtxt = 'Wiederholt'
4943
4944     ;Handle the error
4945     if maxmin_err EQ 1 THEN BEGIN  ;welche Leseroutine hat der error verursacht?
4946     CLOSE,14      ;open Anweisung schliessen um wiederholen zu koennen (maxminfile)
4947     ENDIF ELSE BEGIN
4948     CLOSE,11       ;(limafile)
4949     ENDELSE
4950
4951     WAIT,30
4952     errindex ++
4953     if errindex gt 2 Then Begin
4954     errindex = 0
4955     exit
4956     endif
4957
4958     goto,WIEDERHOLEN
4959
4960  ENDIF
4961
4962     CATCH, /CANCEL
4963  ENDIF
```

Um nur spezielle Fehler zu behandeln, müssen die Errorcodes bekannt sein, die IDL beim entsprechenden Fehler auslöst. Dafür wurden die auftretenden Errorcodes und Errormessages notiert. Da keine Referenz der IDL Errorcodes zur Verfügung stand, wurde eine Liste mit allen Codes und Messages angefertigt. Dazu wurde ein kurzes Programm geschrieben, das alle möglichen Errorcodes mit deren zugehörigen Messages wiedergibt. So können die auftretenden Errorcodes mit der Liste verglichen und deren Ursachen erörtert werden. Zu beachten ist, dass sich die Errorcodes unter UNIX von denen unter Windows unterscheiden. Auch die Anzahl der in IDL verfügbaren Errorcodes unterscheidet sich unter UNIX von der unter Windows. Die Errorcodes werden immer als negative Zahlen zurückgegeben. Die aufgetretenen Fehler, die vom DMF verursacht worden sind, werden in Zeile 4933 des vorangegangenen Quellcodebeispiel angeführt. Ihre zugehörigen Errormessages sind *Error opening file* und *Error encountered reading from file*. Um zu verhindern, dass beim normalen Programmdurchlauf das Programm mit den Zeilen, des Errorhandlers in Berührung kommt, wurde dieser Teil mit einer if-Abfrage von Zeile 4935 bis 4963 sozusagen ausgeklammert. Wenn das Programm im Normalbetrieb beispielsweise auf das Statement *CATCH,/CANCEL* stößt, ohne dass vorher *CATCH* durch einen Error aufgerufen wurde, beendet es sich an dieser Stelle. Als nächstes werden dann in Zeile 4938 die konkreten Errorcodes abgefragt. Diese Stelle im Quellcode muss Betriebssystemspezifisch angepasst werden. Wenn der Errorcode nicht dem der Abfrage entspricht, wird das Programm wieder bei *CATCH,/CANCEL* beendet. Entspricht der Errorcode dem der Abfrage, werden zunächst der Errorcode und die Errormessage ausgegeben, um im Nachhinein im angelegten Logfile nachvollziehbar zu machen, wann und wie oft dieser Fehler aufgetreten ist. Weiter wird die Kontrollvariable *errtxt* mit dem String „Wiederholt“ deklariert, die den String „keine Wiederholung“ besitzt wenn das Programm den Errorhandler nicht durchläuft. Die auftretenden Errors können erst entstehen, nachdem der Befehl *OPENR* aufgerufen ist. Auch wenn der Error auftritt, während diese Anweisung ausgeführt wird, d.h. sie im Fehlerfall nicht zu Ende ausgeführt werden kann, ist eine Fileunit mit der in der Anweisung angegebenen Unitnummer geöffnet worden. Vor dem wiederholten Ausführen der gleichen Anweisung mit der gleichen Unitnummer muss diese Unit erst geschlossen werden. Wird das nicht getan, wird ein erneuter Error ausgelöst, der eine andere

Errornummer besitzt als die, die in dem Errorhandler behandelt wird und so das Programm beendet. Mit der Variable *maxmin_err* wird in Zeile 4945 die Leseroutine identifiziert, in der der Fehler ausgelöst wurde. So kann die korrekte Fileunitnummer geschlossen werden. Die nächste Anweisung *WAIT,30* veranlasst das Programm, an dieser Stelle 30 Sekunden zu warten, bevor es die *OPEN* Anweisung wiederholt. In der nächsten if-Abfrage ist durch die Indexvariable *errindex* festgelegt, wie oft der Errorhandler durchlaufen werden darf, bevor das Programm endgültig beendet wird. Die Begrenzung der Errorhandlerdurchläufe ist notwendig, um eine Endlosschleife zu verhindern. Mit der Anweisung *GOTO* wird aus dem Errorhandler zu der Stelle im Programm gesprungen, von der aus das Programm normal weiter laufen soll; in diesem Fall erneut direkt vor die *OPENR* Anweisungen. Nachdem die Files eingelesen worden sind, werden die Daten aus den Maxminfiles und den LIMA-Datenfiles auf das gewählte Höhenniveau reduziert und in für jede Größe separate Felder geschrieben. Diese werden dann in die Übergabestruktur für die Prozeduren integriert. So kann jede Prozedur nur die für sie relevanten Daten ohne weitere Feldoperationen der Struktur entnehmen. In einer letzten Routine, vor den Prozeduraufrufen, werden die geometrischen Höhen des gewählten Druckniveaus an drei geografischen Stellen ausgelesen. Diese liegen entlang des Greenwich Meridians auf dem Äquator, einer mittleren Breite und auf dem Nordpol. Aus ihnen wird der Maximal und Minimalwert ermittelt und zu einem String umgewandelt, um ihn den Prozeduren zu übergeben. Diese Werte geben in der Plotbeschriftung die geometrischen Höhenbereichsgrenzen des gewählten Druckniveaus an. Die Übergabestruktur wird mit allen Werten und Feldern gefüllt, die für alle Prozeduren benötigt werden. Sie beinhaltet Parameter, die dem Hauptprogramm beim Aufruf übergeben wurden, aber erst in den Prozeduren ausgewertet werden, wie z.B. die Parameter für die Art der Visualisierung oder Art der Ausgabe. Weiter sind Konstanten, die für die Visualisierung benötigt werden, integriert und die Felder, in denen die Wertebereichsgrenzen und die LIMA-Daten für die jeweilige Größe enthalten sind. Das Aufrufen der Prozeduren erfolgt in einer CASE-Anweisung, die die Auswahl der zu visualisierenden Daten berücksichtigt. In den CASE-Blöcken werden die Abarbeitungen der Auswahlen für Konturplots oder Direktplots und für die Hemisphärenansichten gesteuert. Ein Beispiel eines CASE-Blocks für die zu visualisierende Größe Temperatur wird in dem

folgenden Quellcodebeispiel gezeigt.

```
2: BEGIN
    if plotart eq 2 then begin ;beide plotarten ausfuehren----------------------
        uebergabestruktur.plotart  =0
        if mhemisphere eq 2 then begin  ;beide hemisphaeren plotten.............
            uebergabestruktur.mhemisphere  =0
            plot_temperatur,uebergabestruktur
            uebergabestruktur.mhemisphere  =1
            plot_temperatur,uebergabestruktur
        endif else begin;.....................................................
            plot_temperatur,uebergabestruktur
        endelse
        uebergabestruktur.plotart  =1
        if mhemisphere eq 2 then begin  ;beide hemisphaeren plotten.............
            uebergabestruktur.mhemisphere  =0
            plot_temperatur,uebergabestruktur
            uebergabestruktur.mhemisphere  =1
            plot_temperatur,uebergabestruktur
        endif else begin;.....................................................
            plot_temperatur,uebergabestruktur
        endelse
    endif else begin;-----------------------------------------------------------
        if mhemisphere eq 2 then begin  ;beide hemisphaeren plotten.............
            uebergabestruktur.mhemisphere  =0
            plot_temperatur,uebergabestruktur
            uebergabestruktur.mhemisphere  =1
            plot_temperatur,uebergabestruktur
        endif else begin;.....................................................
            plot_temperatur,uebergabestruktur
        endelse
    endelse
   END
```

Für jedes zu erstellende Bild muss die Prozedur einmal aufgerufen werden. Wenn z.B. Nord und Südhemisphäre dargestellt werden sollen, wird die Prozedur zwei Mal mit verschiedenen Hemisphärenparametern aufgerufen. Die Parameter der Übergabestruktur werden direkt im CASE-Block verändert. Durch die Plotauswahlmöglichkeiten sind die Prozeduraufrufe noch mal geschachtelt, sodass sie insgesamt vier Mal in einem CASE-Block aufgerufen werden können.

4.1.5 Prozeduren

Die Prozeduren bestehen aus zwei verschiedenen Plotroutinen. Die erste Plotroutine erstellt einen Konturplot, die zweite einen Direktplot. Bevor die Plotroutinen die Grafiken erstellen können, müssen die Plotumgebungen konfiguriert und die LIMA-Daten für den Plot zum Teil angepasst werden.

Plotspezifisches

Nachdem die Farbtabelle definiert ist, die Übergabestruktur ausgelesen wurde und alle benötigten variablen und Felder deklariert sind, werden plotspezifische Parameter ausgewertet und festgelegt. Dazu gehört, dass die Positionsangaben für den Plot und den Farbbalken im Bild je in einem Feld festgelegt werden. Weiter gehört dazu, dass die Plotbeschriftung individuell für jeden Plot generiert wird und damit auch gleichzeitig der Speicherpfad des Bildes. Der derzeitige Dateiname einer Bilddatei ist z.B. *b_t_n_c2005072906*. Dabei identifiziert das erste Zeichen die Bildart, „*b*“ für Bitmap und „*ps*“ für PostScript. Das zweite Zeichen identifiziert die in dem Bild visualisierte LIMA-Größe, „*t*“ Temperatur, „*u*“ Zonalwind, „*v*“ Meridionalwind, „*h*“ geometrische Höhe, „*d*“ Dichte und „*w*“ den Vertikalwind. Das dritte Zeichen identifiziert die Hemisphärenansicht, „*n*“ die Nordhalbkugel und „*s*“ die Südhalbkugel. Das vierte Zeichen identifiziert die Plotart, hier steht „*c*“ für Konturplot und „*d*“ für Direktplot. Die Zahlenfolge bezeichnet das Datum entsprechend des LIMA-Datenfiles. „*2005*“ stellt den Platzhalter für das Jahr dar, „*07*“ den Monat, „*29*“ den Tag und „*06*“ die Uhrzeit. Da eine Änderung der Katalogstruktur für die Bilder beschlossen wurde, muss die derzeitige Routine noch auf das neue Format angepasst werden. Ein Dateiname sieht dann im Detail z. B. so aus *L.NPS.T.075.20050729.06*. Das „*L*“ steht hier für LIMA, „*NPS*“ bezeichnet die Nordhemisphäre, „*SPS*“ an dieser Stelle die Südhemisphäre. „*T*“ stellt den Platzhalter da, der die visualisierte LIMA-Größe identifiziert. An der Stelle von „*075*“ wird der Index der Modellhöhe bezeichnet, auf dem die LIMA-Größe visualisiert wurde. Die darauf folgende Zahlenfolge kennzeichnet wieder das Datum, wie oben bereits beschrieben.
Nachdem die Plot- und Dateibezeichnungen generiert sind, folgt die Konfiguration des Ausgabegerätes (Outputdevice). Dazu wird abgefragt welche Ausgabeart dem Programm übergeben wurde. Die Bildschirmausgabe erfolgt unter einem UNIX-

System im X-Device, unter Windows im WIN-Device. Die Deviceeinstellung wird am Anfang des Hauptprogramms aus der Systemvariablen *D.NAME* ausgelesen, in einer Variable festgehalten und den Prozeduren übergeben, um sie nachdem Schreiben der Datei wieder zurücksetzen zu können. Wenn das Ausgabebild in ein PostScriptfile geschrieben werden soll, wird mit der Anweisung *SET_PLOT,'PS'* das PostScriptdevice gewählt. In diesem Fall müssen vor dem eigentlichen Plotvorgang PostScript spezifische Einstellungen mit dem Befehl *DEVICE* getätigt werden, wie z.B. festlegen des Speicherpfad, Größe der Zeichenfläche, Schriftgröße, Farbtiefe u.s.w. . Beim Erstellen eines PostScriptes wird die Bilddatei erst angelegt und während des Plotvorgangs die Bilddaten in die Datei geschrieben. Nachdem Plotvorgang wird mit der Anweisung *Device,CLOSE* die Ausgabe auf das Ausgabegerät beendet und die PostScript-Datei geschlossen. Wenn eine Pixelgrafikdatei erstellt werden soll, wird das Outputdevice wie in Zeile 2409 des folgenden Quellcodebeispiels auf den Z-Buffer gesetzt. Durch Verwendung dieses von IDL bereitgestellten Pseudodevice können komplexe dreidimensionale Darstellungen für Bitmapgrafiken definiert werden. Verdeckte Linien und Flächen werden nicht angezeigt. Der Z-Buffer wird von IDL auch bei der Bildschirmausgabe verwendet, aber nur für die Bildberechnungen, nicht als Ausgabegerät. Durch die Wahl des Z-Buffers als Ausgabegerät wird die Bildschirmausgabe ausgeschaltet, sodass das Bitmapbild lediglich im Speicher vorhanden ist. Durch *COPY* wird die aktuelle Farbtabelle auch im Z-Buffer angewendet.

```
2397 if scr eq 1 then begin          ;scr =1 -> PS erstellen
2398
2399 SET_PLOT,'PS'
2400 DEVICE, FILENAME = savestring_ps,/ENCAPSULATED, $
2401     xsize=22.,ysize=17.,font_size=12,/color, $
2402     BITS_PER_PIXEL=8, /PORTRAIT,/bold, $
2403     xoffset=1.905,yoffset=27
2404
2405
2406 endif else begin               ;scr =! 1 -> Bildschirmausgabe & Pixelgrafikdateien
2407 if scr gt 1 then begin         ;Pixelgrafikdateien
2408
2409 SET_PLOT,'Z',/COPY             ;wenn SET_PLOT,'Z' aktiv ist gibts keine
      Bildschirmausgabe
2410                                ;(Bild wird in den Z-Buffer geschrieben)
2411                                ;## optional (the default size is 640 by 480):
2412 device , SET_RESOLUTION = [1024,768]
2413 endif else begin              ;Bildschirmausgabe
```

```
2414 WINDOW, 0, XSIZE=1024, YSIZE=768
2415 endelse
2416 endelse
```

Die Auflösung des Ausgabebildes wird mit der Anweisung *SET_RESOLUTION* in Zeile 2412 festgelegt. Nachdem die vollständige Bitmapgrafik erstellt worden ist, wird am Ende der Prozeduren mit der Anweisung *WRITE_IMAGE,savestring,'PNG', TVRD(),R_curr, G_curr,B_curr* das im Speicher vorliegende Bild in ein Dateiformat geschrieben. Die durch diese Anweisung zur Verfügung gestellten Pixelgrafikformate sind PNG wie angegeben, BMP, JPEG, PPM, SRF und TIFF. Wenn eine Bildschirmausgabe erfolgen soll, wird keine Ausgabegeräteveränderung vorgenommen da IDL das Bildschirmausgabegerät als Standardausgabegerät anspricht. Als verbleibender Parameter wird die Auflösung des Ausgabefensters mit dem Befehl *WINDOW* festgelegt. Die Bildschirmausgabe ist im Batchbetrieb unter UNIX nicht vorhanden. Für eine Bildschirmausgabe durch die Apollo muss eine Verbindung über einen X-Server mit dem Computingserver bestehen. Sie ist nur für die Programmweiterentwicklung notwendig, um nicht für jede Bildüberprüfung eine Datei erstellen zu müssen. Die Möglichkeit der Bildschirmausgabe ist im Batchfile mit angegeben, um zu erklären, wofür dieser Wert vergeben ist. Nachdem die Auswahl der Ausgabeart in der Prozedur behandelt ist, werden die tatsächlichen Bereichsgrenzen der zu visualisierenden LIMA-Größe festgestellt und mit denen vom Maxminfile für diesen Bereich verglichen. Überschreitet eine der tatsächlichen Bereichsgrenzen der LIMA-Werte die des Maxminfiles, wird eine entsprechende Warnmeldung ausgegeben. Zusätzlich werden die Bereichsgrenzen ausgegeben. So kann nachträglich im Logfile nachvollzogen werden, wie hoch die Wertebereichsüberschreitung in welchem LIMA-File ist. Da einige Größen des LIMA-Modells in einem Wertebereich weit unter Eins liegen, folgt eine Routine, die abfragt, bis zu welcher Zehnerpotenz die Werte unter Eins liegen. Dabei wird ein Faktor festgelegt, der die größten Werte auf eine Zehnerpotenz über Zehn vergrößert und die Zahl der Zehnerpotenz um die die Werte vergrößert werden, um dies später in der Plotbeschriftung angeben zu können. Mit der nächsten Routine werden die obere und untere Wertebereichsgrenzen auf die nächst kleinere / größere Zehnerstelle erweitert. Damit werden glatte Wertebereichsgrenzen für die Farbskala geschaffen und ein zusätzlicher Werteraum um Wertebereichsgrenzüberschreitungen der Lima-Daten bis zu einem bestimmten Grad trotzdem korrekt darstellen zu

können. Weiterhin wird eine Anpassung der Farbskalabeschriftung vorgenommen sodass, wenn mehr als 15 Werteangaben an der Farbskala gemacht, werden nur noch alle 20 Einheiten eine Werteangabe gemacht wird, nicht jede Zehnte. Dadurch wird eine zunehmende Unübersichtlichkeit der Farbskala durch große Wertebereiche verhindert. Nachdem die Plotvorbereitenden Maßnahmen beendet sind, beginnen die Plotroutinen.

Konturplot

Bei den Konturplots trat ein Problem in Verbindung mit den Kontinentenumrandungen auf. Wenn Daten visualisiert wurden, die alle in einem positiven Wertebereich lagen, waren die Kontinente nicht zu sehen. Bei der Visualisierung von Datensätzen mit negativen und positiven Werten waren die Kontinentenumrandungen nur auf den Flächen die negative Wertebereiche darstellten zu sehen (Abb. 4.4 a). Das gilt zusätzlich nur für Pixelgrafiken, in den PostScriptbildern sind die Kontinentenumrandungen immer vollständig vorhanden. Trotz intensiven Nachforschungen in zur Verfügung stehender Fachliteratur und dem Internet zu diesem Problem, wurden keine Informationen gefunden. Letztendlich wurde das Problem gelöst, indem alle Werte, in dem zu visualisierendem Feld einfach

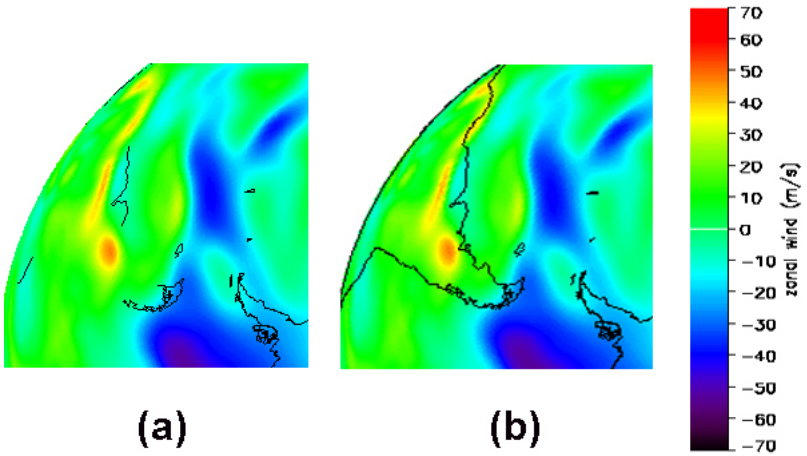

Abbildung 4.4: Fehlerhafte Kontinentendarstellung im Konturplot in (a) und korrekte Darstellung in (b).

um den maximalen positiven Wert vermindert wurden. Damit wurde erreicht, dass alle Werte im gleichen Abstand in den negativen Bereich verschoben wurden und so die vollständigen Kontinentenumrandungen in den Pixelgrafiken sichtbar sind (Abb. 4.4 b). Diese Methode ist bis zum Ende der Bearbeitungszeit dieser Diplomarbeit in anwendung geblieben und bestand bis zu diesem Zeitpunkt auch jeden Testlauf. Zwei Fakten für den Ansatz einer Ursachenforschung könnten sein, dass der Z-Bufferalgorithmus für Pixelgrafiken berechnet, was in einer dreidimensionalen Grafik zu sehen ist und perspektivisch verdeckt wird, und dass das Problem nur bei Pixelgrafikdarstellungen auftauchte. Bevor die Daten mit der eigentlichen Funktion visualisiert werden können, muss ein dreidimensionaler Darstellungsraum mit der Funktion *MAP_SET* wie im folgenden Quellcodebeispiel initialisiert werden.

```
2609 MAP_SET, hem,0, /ORTHOGRAPHIC, /NOBORDER, COLOR = 0, $
2610                       POSITION = plotfenster, /ISOTROPIC
```

Mit *MAP_SET* sind mehr als zehn verschiedene Kartenprojektionen möglich, darunter auch die in diesem Programm benötigte stereographische Projektion. Die ersten beiden Parameter in der Anweisung definieren den darzustellenden Mittelpunkt der Projektion in Breitengrad und Längengrad. Die hier verwendete Variable *hem* realisiert die beiden gewünschten Projektionen. Für die Nordpolarstereographische Projektion enthält sie den Wert 90, für die Südprojektion -90. Das Schlüsselwort *ORTHOGRAPHIC* veranlasst die *MAP_SET*-Funktion die Stereographische Projektion zur Verfügung zu stellen. Durch das Schlüsselwort *NOBORDER* wird das Zeichnen eines Rahmens um den Plot verhindert. Der *COLOR* übergebene Wert indiziert die Farbe in der aktuellen Farbtabelle, mit der die *MAP_SET*-Elemente dargestellt werden. *POSITION* wird das Feld übergeben in dem die Positionsparameter für den Plot im Bild festgelegt wurden. Mit *ISOTROPIC* wird festgelegt, dass der Darstellungsraum in alle Koordinatensystemrichtungen die gleiche Ausdehnug besitzt. Die IDL-Funktion, mit der die LIMA-Daten visualisiert werden, ist die *CONTOUR*-Funktion. Ursprünglich ist sie entwickelt worden, um Konturlinienplots zu erstellen, wie man sie von z.B. Höhenlinien auf Landkarten her kennt. Sie besitzt das Schlüsselwort *CELL_FILL* durch das es erst interessant wird, Oberflächen mit Feldern unterschiedlicher Wertigkeit darzustellen. Durch dieses Schlüsselwort werden die Zwischenräume der Konturlinien ausgefüllt. Die Eigenschaft dieser Funktion,

dass sie Übergänge zwischen den Werten durch Interpolation berechnet, kann sich nachteilig auf den Informationsgehalt des Bildes auswirken. So kann es passieren, dass einzelne Datenwerte mit abweichenden Werten, die im gesamten Bild nur einen kleinen Punkt darstellen, den Umgebungswerten angepasst, glatt gerechnet werden. Dieser Effekt kann durch eine hohe Auflösung des darzustellenden Wertebereichs eingeschränkt, aber nicht ausgeschaltet werden. Dazu kann die Anzahl der Konturlinien in dem zu visualisierenden Wertebereich mit dem Schlüsselwort *LEVELS* festgelegt werden. Wenn die Auflösung fein genug festgelegt wird, scheint die mit dieser Methode visualisierte Grafik aus stufenlosen Farbübergängen zu bestehen. Diese Anzahl entspricht in den Plots der Anzahl der Farben, in der Farbtabelle die für die Visualisierung der LIMA-Daten festgelegt wurden. Das sind nach Abzug der Farben Schwarz und Weiß 253. Mit den Schlüsselwörtern *MAX_VALUE* und *MIN_VALUE* wird der Wertebereich festgelegt, auf den die Funktion die Anzahl der Konturlinien aufteilen soll. In dem nachfolgenden Quellcodebeispiel wird gezeigt wie die *CONTOUR*-Funktion in den Prozeduren aufgerufen wird. *arr_plot2* ist das Feld in dem die zu visualisierenden LIMA-Daten enthalten sind. Die Felder *v_kordi_x* und *v_kordi_y* enthalten die Positionsangaben in Breiten- und Längengrad für jeden Wert in *arr_pot2*. Das Schlüsselwort *OVERPLOT* bewirkt, dass das vorher mit *MAP_SET* erstellte Bild überschrieben wird, aber die stereographische Plotumgebung, die mit *MAP_SET* erstellt wurde, nicht verworfen wird. Durch das Schlüsselwort *IRREGULAR* werden die Werte in den Koordinatenfeldern von der *CONTOUR*-Funktion korrekt eingesetzt. Mit dem Grafikschlüsselwort *noerase* wird verhindert, dass das gesamte Bild gelöscht wird, wenn die Kontinente mit der darauf folgenden *MAP_SET*-Anweisung über den Konturplot gezeichnet werden. Durch das Schlüsselwort *CONT* werden die Kontinente mit der *MAP_SET*-Funktion gezeichnet. Mit dem Schlüsselwort *GRID* wird bewirkt, dass ein Gradgitternetz, mit *HORIZON* eine Horizontlinie gezeichnet wird.

```
2659
2660 CONTOUR, arr_plot2, v_kordi_x, v_kordi_y, C_COLORS = INDGEN(ncolors), $
2661              MAX_VALUE = maxval2, MIN_VALUE = minval2,LEVELS = levels,/OVERPLOT,$
2662              /CELL_FILL, /IRREGULAR ,/noerase;ACHTUNG! farbanzahl beachten
2663
2664 if cnt eq 1 then begin
2665        MAP_SET,hem,0, /ORTHOGRAPHIC, /NOBORDER,/CONT, /HORIZON,/ISOTROPIC, $
2666              COLOR = 1, POSITION = plotfenster,/NOERASE ;,/GRID
```

```
2667     endif
```

Anschließend wird der Farbbalken in ein zweites Plotfenster neben dem eigentlichen Plot gezeichnet. Die Umgebung für den Farbbalken ist ein Plotgraph, dessen Bezeichnungen und Linien einfach mit der Farbe Weiss gezeichnet werden, sodass sie vor dem Hintergrund nicht sichtbar sind. Dieser Graph wird mit der *PLOT*-Funktion, wie es das folgende Quellcodebeispiel zeigt, realisiert.

```
2689     ; den Raum fuer den Farbbalken aufspannen
2690
2691     PLOT, yy1, xx2, XSTYLE=4, YSTYLE=1, XRANGE=yy2, YRANGE=xx1, $
2692         COLOR=0,  /NODATA, /NOERASE, /ISOTROPIC, $
2693         CHARSIZE=0.1, CHARTHICK=0.1, POSITION = farbfenster, XTICKS = fbteiler
```

Der Farbbalken wird danach in einer Schleife mit Polygonen in die Plotumgebung gezeichnet. Die Beschriftung des Farbbalkens mit den Werten erfolgt mit der *AXIS*-Funktion, mit der separate Achsen Dargestellt werden können. In Diesem fall die Y-Achse. Der Konturplot wird mit der Bildbeschriftung fertig gestellt, die mit der *XYOUTS*-Funktion realisiert wird wie sie unten zu sehen ist.

```
2733 ;plotbeschriftung
2734
2735 XYOUTS,[0.05,0.05,0.05],[0.94,0.9,0.86],[title,heightstring,plotname],$
2736                                  /NORM,CHARS=1.5,CHARTHICK=2, color=1
```

Sie ist in der oberen linken Ecke des Bildes platziert. Ganz oben ist, wie in Abblidung 4.5 gezeigt wird, die visualisierte LIMA-Größe aufgeführt. Darunter die geometrische Höhe des Modellniveaus, in dem sich die LIMA-Größen befinden, und als drittes der Zeitpunkt der Momentaufnahme der gleichzeitig auch den Dateinamen der LIMA-Datei darstellt.

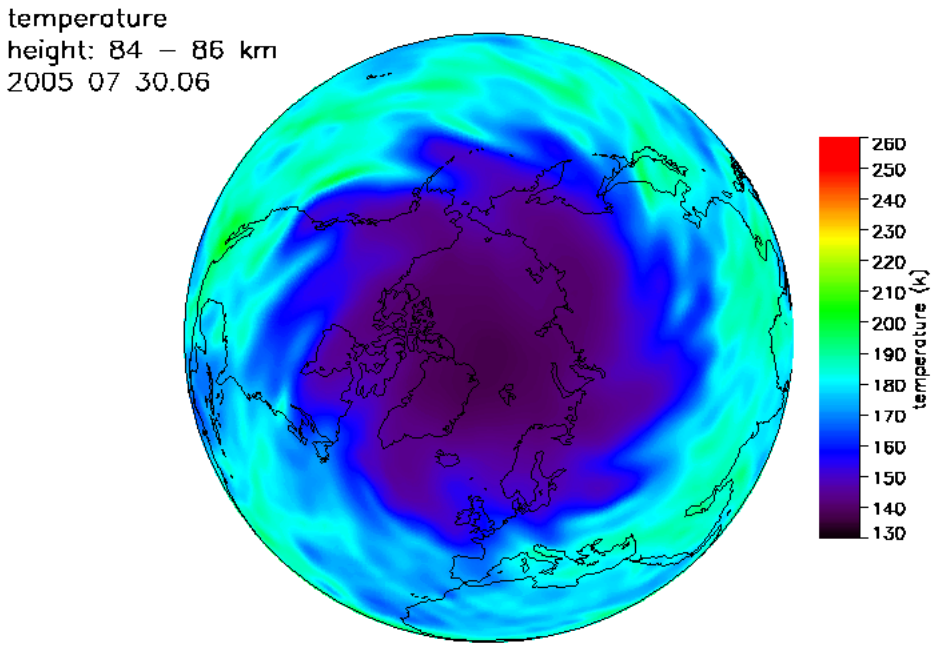

Abbildung 4.5: Beispiel für einen Temperatur Konturplot mit Kontinenten.

Direktplot

Diese Variante der LIMA-Datenvisualisierung wurde gewünscht um fehlerhafte Werte, so genannte „Ausreißer“in den LIMA-Modelldaten lokalisieren zu können. Die Konturdatenplots geben ein harmonisches Gesamtbild, das durch interpolation der Informationen erreicht wird. Beim Direktdatenplot wird jeder einzelne Punkt des LIMA-Modells als diskreter Wert mit seiner entsprechenden Farbe gezeichnet. Das wurde realisiert, in dem eine Kugel erstellt wurde deren Oberfläche mit viereckigen Polygonen bedeckt ist. Die Vierecke sind zentral auf den LIMA-Gitterpunkten positioniert. Ihre Ecken liegen jeweils mittig zwischen dem benachbarten Punkt auf derselben Punktreihe und mittig zwischen den unterschiedlichen Punktreihen. Die Kugelkoordinaten der Polygoneckpunkte werden anhand der Vorschrift wie sie in Abbildung 4.6 a angegeben ist berechnet. Dadurch, das die Eckpunktkoordinaten mit Winkelangaben berechnet werden sind die Polygone in ihrer Flächenform der Kugeloberfläche angepasst. Das äußert sich in dem die obere Kannte (näher zum Pol) des Polygons kürzer ist als die untere wie in Abbildung 4.6 b zu sehen ist.

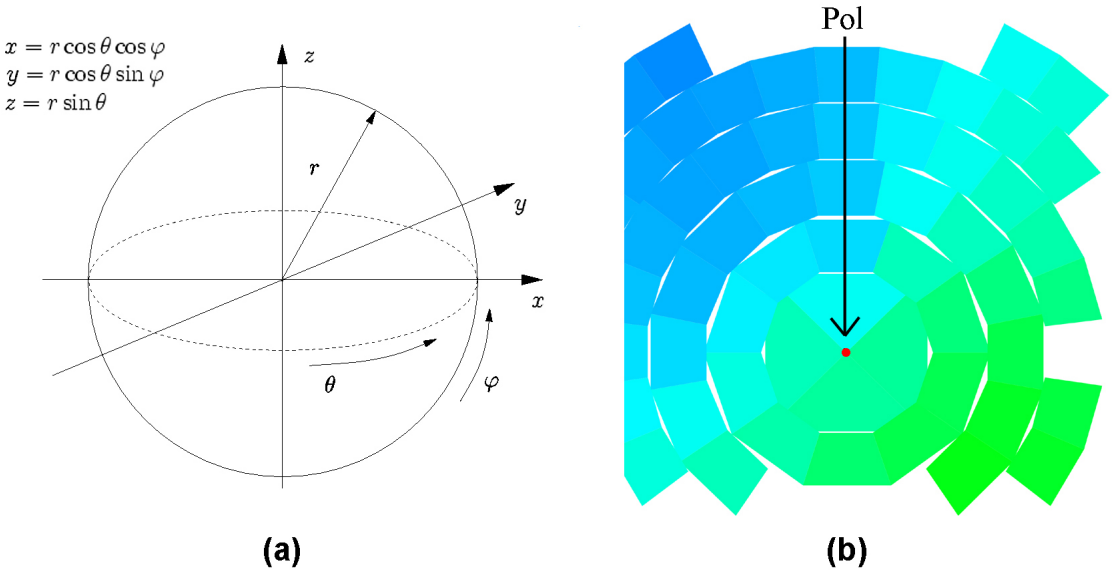

Abbildung 4.6: In a wird das Prinzip der Umrechnung von Kugelkoordinaten auf kartesische Koordinaten veranschaulicht. In b ist ein stark vergrößerter Ausschnitt einer Polarstereographischen Darstellung von LIMA-Daten mit der Polygon-Methode gezeigt.

Die Punkte die auf der letzten Reihe an den Polen liegen, werden mit Dreiecken dargestellt, die bis zum Pol reichen. Durch die Zeichenreihenfolge werden die zu großen Ecken von den nachfolgend gezeichnetten Polygonen verdeckt. Auch die Größe der Vierecke ist so gewählt das sich die Kannten der Polygone geringfügig überlappen, um sichtbare Lücken zu vermeiden. Die dreidimensionale Zeichenumgebung für die Polygone wird von einer leeren *SURFACE*-Funktion bereit gestellt, wie sie im folgenden Quellcodebeispiel gezeigt wird.

```
2755
2756 Surface,Dist(50), XRange=[-1,1], YRange=[-1,1], ZRange=[-1,1],xtitle='X',$
2757    ytitle='Y',ztitle='Z',AZ=270,AX=hem,/NoData,/Save,/normal,POS=posit,COLOR = 0;
```

Mit dem ersten Parameter wird ein leeres Feld übergeben. In *XRange*, *YRange* und *ZRange* wird das Koordinatensystem ausgerichtet. Der Mittelpunkt der Kugel liegt wenn man einen Radius von Eins benutzt bei der Umrechnung von Polar- auf kartesische Koordinaten, auf der Koordinate 0,0,0. Der Wertebereich für alle Koordinaten zwischen 1 und -1 festgelegt. Mit *AZ* und *AX* wird der Blickwinkel auf den Zeichenraum gesteuert. Wie bei den Konturplots gibt die Variable *hem* die darzustellende

Hemisphäre an, *AZ* muss der Wert 270 Grad zugewiesen werden damit der Plot dieselbe Ausrichtung hat wie der Konturplot. *POS* wird das Feld mit den Positionsparametern für den Plot im Bild übergeben. Weil sich bei dieser Routine die Positionen des Plots zwischen süd- und nordstereographischer Ansicht unterscheiden müssen hier jeweils unterschiedliche Positionsangaben gemacht werden, um einheitliche Bilder zu erstellen. Mit dem Schlüsselwort *NoData* wird festgelegt das mit der Funktion keine Daten ins aktuelle Koordinatensystem geplottet werden sollen und mit *SAVE* bleibt der erstellte Zeichenraum für nachfolgende Zeichenfunktionen erhalten. In zwei ineinander liegenden Schleifen für die Anzahl der Breitenringe und der jeweiligen Anzahl darauf liegender Pukte werden die Polygone gezeichnet. Das geschieht mit der *POLYFILL*-Funktion, wie nachfolgend zu sehen ist.

```
2889
2890        POLYFILL,xpoly,ypoly,zpoly,/T3D,Color=colorpoly,/Data
```

Die drei Variablen *xpoly*, *ypoly* und *zpoly* enthalten die zuvor ermittelten X, Y und Z Koordinaten der Eckpunkte des Polygons. Mit dem Schlüsselwort *T3D* ist es möglich die *POLYFILL*-Funktion in der 3D Transformationsmatrix, die durch die *SURFACE*-Funktion erstellt wurde anzuwenden. *COLOR* wird der Farbwert für das Polygon übergeben und *Data* veranlasst das im aktuellen Koordinatensystem gezeichnet wird. Der Farbbalken und die Plotbeschriftung erfolgt in der Direktplotroutine genauso wie in der Konturplotroutine. Das Ergebnisbild ist vom Layout her genauso gestaltet wie das der Konturplotroutine, wie in der Abbildung 4.7 veranschaulicht wird. Im Direktdatenplot sind keine Kontinentenumrandungen verfügbar, weil die Priorität hier in der Sichtbarkeit möglichst aller visualisierten Datenpunkte liegt. Wie das Direktplotbeispiel zeigt sind in der Originalansicht kaum Lücken zwischen den einzelnen Polygonen zu erkennen. Um optische Unterschiede zwischen den Visualisierungsarten zu veranschaulichen sind in den beiden Beispielbildern genau dieselben Daten visualisiert worden.

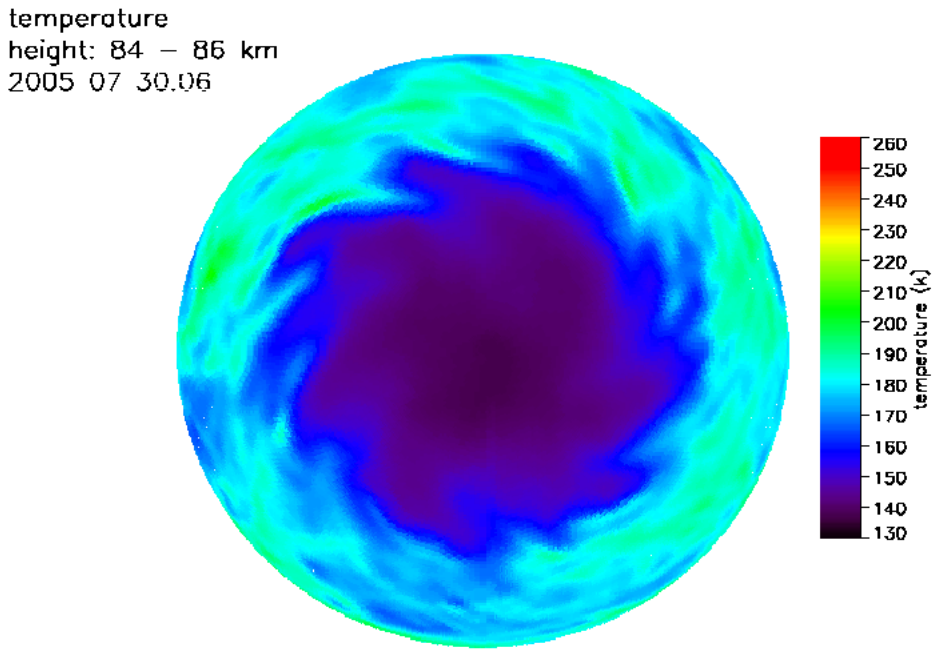

Abbildung 4.7: Beispiel für einen Temperatur Direktplot

4.1.6 Automatisierung

LIMA-Ergebnisvisualisierung

Eine Grundautomatisierung der Ergebnisvisualisierung wird durch verschiedene Eigenschaften des Programms gegeben. Diese sind: die Möglichkeit der Parameterübergabe beim Aufruf des Programms, die dynamische Pfad und Dateinamengenerierung, die eigenständige Fehlerbehandlung und das unabhängige Laufzeitverhalten, von z.b. Ausgabegeräten. Die Automatisierung auf den Computingservern APOLLO oder HYDRA wird durch den Einsatz von Shell-Skripten realisiert. Ein solches C-Shell-Skript für die APOLLO ist nachfolgend aufgeführt.

```
1 #!/bin/csh
2
3 source /opt/local/rsi/idl_6.3/bin/idl_setup
4 setenv IDL_STARTUP hl_batchf_e2
5 idl -nw
```

In der ersten Zeile wird angegeben von welchem Interpreter das Skript bearbeitet werden soll. Mit der dritten Zeile wird die Umgebung für das Programm eingerich-

tet, in dem der Pfad für die IDL-Laufzeitumgebung angegeben wird. Danach wird in Zeile vier das Batchfile, das am Anfang von Abschnitt 4.1.3 aufgelistet ist, als *IDL_STARTUP*-Datei festgelegt. Dadurch wird bewirkt, dass direkt nachdem Start der IDL-Umgebung das Batchfile abgearbeitet wird, in dem das eigentliche Programm mit den nötigen Parametern aufgerufen wird. Mit der Anweisung *idl -nw* in Zeile fünf wird IDL im Kommandozeilenmodus aufgerufen [25].

Filmerstellung

Zusätzlich sollen diese großen Mengen von Einzelbildern in die effektiv auswertbare Form eines Filmes umgewandelt werden. Um diese Problemstellung auch möglichst automatisiert zu realisieren stehen zwei Batch-fähige Programme zur Verfügung. Das sind, wie in Abschnitt 3.1.2 schon eingeführt wurde, Convert und MJPEGTOOLS. Leider sind zurzeit beide Programme nur eingeschränkt oder gar nicht auf den im IAP vorhandenen Computingservern mit HP-UX Betriebssystemen lauffähig. Da in naher Zukunft ein neuer Computingserver mit LINUX Betriebssystem angeschafft werden soll und diverse PC's mit LINUX Betriebssystemen als Computingserver schon vorhanden sind, wurde die Filmerstellung auf einen dieser PC's verlagert. Da beide Programme Vor- und Nachteile in verschiedenen Funktionen aufweisen wurde eine Lösung durch Kombination der beiden Programme beschlossen. Dazu wurde von Herrn Dr. Baumgarten eine Abfolge von Skripten implementiert, die durch Kombination der beiden Programme einen MPEG-Film erstellen. Die LIMA-Daten liegen in einem Abstand von sechs Stunden vor, was für die Darstellung von z.B. Vertikalwind oder Zonalwind in der Höhe der Mesopause zu grob ist, sodass in einer schnellen Animation der Bilder nur schwer Informationen entnommen werden können. Bei einer langsamen Animation dieser Bilder können dem Film besser seine Informationen entnommen werden aber durch das starke Springen der Bilder wird das Ergebnis wieder unbefriedigend. Hier wird nun eine *Morphing*-Funktion von Convert eingesetzt um ein interpoliertes Zwischenbild zu erzeugen, das den Sprungeffekt aus dem mit MJPEGTOOLS erstellten Film entfernt. Die Originalbilder werden in JPEG-Bilder umgewandelt und mit den Zwischenbildern in ein Temporäres Verzeichnis geschrieben. Mit MJPEGTOOLS werden diese Bilder mit sehr viel geringerem Hauptspeicherbedarf, als bei Convert dies tut, dann zu einem MPEG-Film

zusammengefasst. Für diese Art der Filmerstellung ist auch eine Version ohne interpolierte Zwischenbilder vorhanden, weil beim Morphen Farbveränderungen in den Bildern auftreten und nicht alle Bildfolgen einen so starken Strukturwechsel aufweisen, dass sie Zwischenbilder benötigen würden. Die ursprünliche Bildauflösung der Einzelbilder musste im IDL Visualisierungsprogramm von 800 × 600 Bildpunkten auf 1024 × 768 angehoben werden, weil MJPEGTOOLS bei der Behandlung der Bilder mit exakt 800 × 600 Pixeln abstürzt. Die gewünschte zukünftige Weiterentwicklung der Automatisierung sollte so aussehen, dass das Ergebnisvisualisierungsprogramm parallel zur LIMA-Simulation läuft, sodass es selbstständig jede neu erstellte LIMA-Ergebnisdatei in den gewünschten Formen visualisiert. Sobald die Bilder für einen kompletten Monat fertig gestellt sind, sollen sie dann durch Aufrufen der Filmerstellungsskripte zu einem Film zusammengefasst werden.

4.2 Strömungsfilm

Im Rahmen dieser Arbeit wurde die Möglichkeit untersucht aus LIMA-Daten einen Strömungsfilm zu erstellen (s. Abschnitt 3.1.5). Nach dieser Vorgabe wurde im Internet recherchiert um eine Lösung zu Identifizieren. Anhand einer detailierten Studie, des durch Meteoblue zur Verfügung gestellten Materials, wurde der folgende Lösungsansatz erarbeitet. Die Umsetzung war jedoch im Rahmen dieser Arbeit nicht mehr möglich. Für die Darstellung des Windes werden die Parameter Windrichtung und Windgeschwindigkeit benötigt. Sie werden aus den meteorologischen Größen Meridionalwind (v) und Zonalwind (u) berechnet. Wie die Richtung des Windes an einem Punkt ermittelt wird, ist in Abbildung 4.8 veranschaulicht. Der Richtungsvektor in dem jeweiligen Quadranten ergibt sich aus den Gleichungen der dazu gehörigen Winkel in Abbildung 4.8. Die Winkelgleichungen ergeben sich wiederum aus den Wertezustandsbedingungen von (v) und (u). Die Windstärke, in Meter je Sekunde, errechnet sich aus dem Betrag des horizontalen Windvektors mit der Formel $\sqrt{(u^2 + v^2)}$. Diese Parameter müssen für jeden Punkt des LIMA-Modells berechnet werden. Da die zeitliche Auflösung der LIMA-Ergebnisdaten für die Erstellung eines flüssigen Strömungsfilmes zu gering ist, müssen zwischen den einzelnen LIMA-Datenzeitpunkten interpolierte Datensätze erstellt werden. Um die

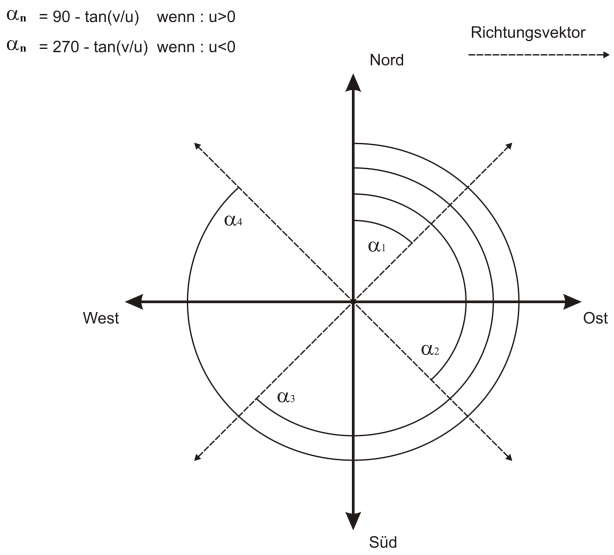

Abbildung 4.8: Prinzip der Windrichtungsberechnung mit u und v.

Strömungslinien in diesen interpolierten Datensätzen nicht nur in die festgelegte Anfangswindrichtung laufen zu lassen, muss eine Wichtung an den Strömungslinienpositionen zwischen den LIMA-Gitterpunken stattfinden. Das bewirkt, dass die Stömungslinien in den interpolierten Datensätzen ihre Richtung in Abhängigkeit der Wertigkeit und des Abstandes zu den sie umgebenden Gitterpunkten ändern. Angenommen, es würden zwischen den einzelnen LIMA-Ergebnisdateien fünf Datensätze interpoliert werden, also zu jeder simulierten Stunde einen, könnte sich die Bewegung einer Strömungslinie in Abhängigkeit von vier Gitterpunkten wie in Abbildung 4.9 verhalten. Um den Effekt der wandernden Strömungslinie zu realisieren, die für kurze Zeit ihren zurückgelegten Weg anzeigt, müssen für eine feste Anzahl die vergangenen Positionen festgehalten werden. Die einzelnen Teile einer Strömungslinie könnten z.B. mit Polygonen oder Linienfunktionen realisiert werden.

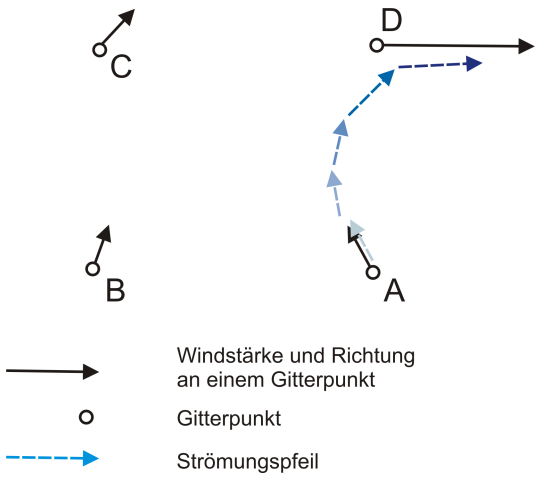

Abbildung 4.9: Bewegung einer Stömungslinie in Abhängigkeit von vier Gitterpunkten.

4.3 Gezeitenanalysetool

Im laufenden Entwicklungsprozess wurde in enger Zusammenarbeit mit den Anwendern eine Software entwickelt, die es ermöglicht, verschiedene Kombinationen von Gezeitenwellen in der Atmosphäre darzustellen. Das Programm liegt zum Ende des Bearbeitungszeitraumes in einer lauffähigen Version vor, die die ursprünglichen Anforderungen erfüllt. Die in diesem Entwicklungsprozess entstandene grafische Benutzerschnittstelle ist in Abbildung 4.10 zu sehen. Sie ist mit den Dimensionen 430 Pixel Breite und 700 Pixel Höhe so gewählt, dass sie an einem Bildschirm mit einer Auflösung von 800 × 600 Pixeln vollständig dargestellt werden kann. Die GUI wurde so entworfen, dass die Abarbeitung der Steuerelemente von oben nach unten verläuft, um eine grundlegende intuitive Bedienung zu ermöglichen. Das Layout für die Parametereingabe wurde in Form einer Tabelle realisiert. Die Zeile unter dem Hinweis *Parameter festlegen* erläutert das Verhalten der Wellen bei Parametrisierung in den Eingabefeldern für die Wellenzahl. Es können die Interaktionen von bis zu vier Wellen gleichzeitig dargestellt werden. In der Spalte *Wellen* werden die Zeilen für die Parametereingabe den einzelnen Wellen zugeordnet. Für die einzelnen Wellen können die Parameter Amplitudenstärke, die Dauer einer Wellenperiode in Stunden und die Wellenanzahl auf einer Ausdehnung von 360 Längengraden festgelegt werden. Ganz oben links auf der GUI, wurde ein Pulldownmenü implementiert, mit dem Eingabeparameter für die Wellensimulation in einer Textdatei gespeichert oder aus einer gelesen werden können. Mit den Radiobuttons *Animation* und *Bilder-*

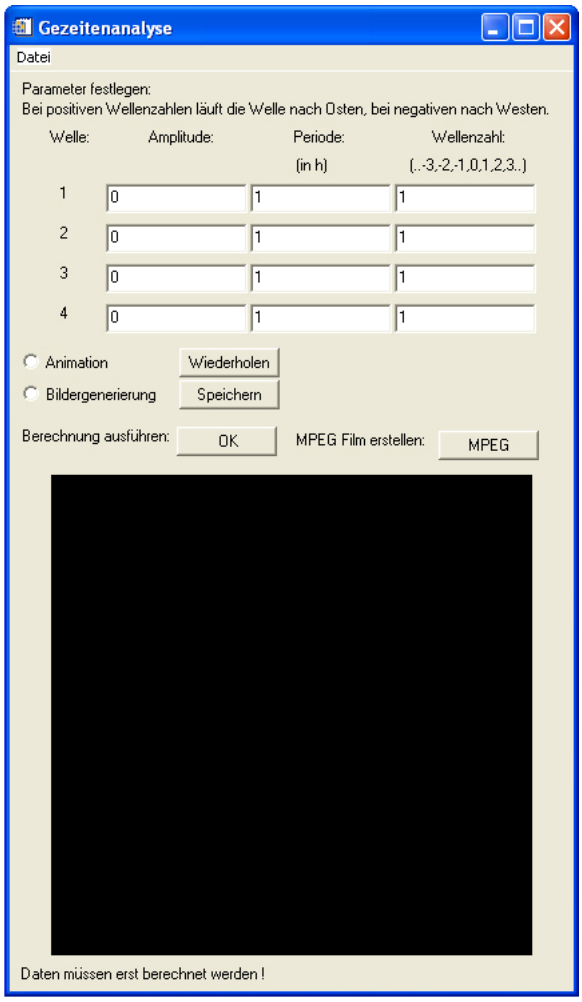

Abbildung 4.10: GUI der Gezeitenanalysesoftware.

generierung können die primären Ausgabevarianten der Wellensimulation gewählt werden. Der Zeitraum, der in den beiden Varianten dargestellt wird, ist 24 Stunden. Durch Wählen des *Animation*-Radiobuttons wird eine Animation im Grafikfenster auf der GUI gezeigt. Die Animation besteht dabei nur aus 36 Bildern um die Rechenzeit möglichst kurz zu halten. Bei Auswahl des *Bildergenerierung*-Radiobuttons wird die Simulation in Form von 360 Bitmapbildern im PNG-Format gespeichert. Diese müssen nachträglich mit einer externen Software zu einer Animation zusammengefügt werden. Durch die höhere Anzahl der Bilder wird die im Nachhinein erstellte Simulation viel flüssiger, ist durch die Erstellung der vielen Bilder aber auch rechenintensiver. Für den Fall, dass keine externe Software zur Verfügung steht aber trotzdem ein Film erstellt werden soll, wurde eine dritte, sekundäre Variante realisiert. Mit Betätigung des *MPEG*-Buttons wird sofort ein MPEG-Film der

gewünschten Simulation von dem Programm erstellt. Der Film wird dabei direkt mit einer IDL-Routine erzeugt und besteht aus 360 Einzelbildern. Die Rechenzeiten für diese Variante sind allerdings noch länger als bei der Bildergenerierung. Außerdem ist die Qualität des Filmes, der mit IDL erstellt wird, nur „ausreichend“, da er deutliche Artefakte durch die verlustbehaftete Komprimierung aufweist. Zur Auswahl der Bildergenerierung gehört der *Speichern*-Button. Bei dessen Betätigung öffnet sich ein Popupfenster, in dem der Speicherpfad für die Bilder gewählt werden kann. Wenn eine der beiden Primärvarianten gewählt und spezifische Einstellungen getätigt sind, wird mit dem *OK*-Button die entsprechende Berechnung der Wellensimulation ausgeführt. Mit dem *Wiederholen*-Button kann, wenn der *Animation*-Radiobutton gewählt wurde, die vorherige Animation ohne erneute Berechnung betrachtet werden. Unter dem Zeichenfeld ist eine Textzeile platziert, die Hinweise ausgibt, wie z.B. den Fortschritt der aktuellen Aktion oder Fehlermeldungen, wenn ein Eingabefehler vom Benutzer gemacht wurde. Das Funktionsschema der Gezeitenanalysesoftware und die Einordnung der Programmelemente in die Schichten des Seeheimmodells (Abb. 3.7) ist in Abbildung 4.11 verdeutlicht. Im Hauptprogramm wird zunächst die grafische Benutzerschnittstelle erzeugt. Dies geschieht mit den von IDL bereitgestellten *WIDGET*-Funktionen [27]. Das Layout der GUI wird durch die hierarchische Anordnung von *WIDGET_BASE*-Funktionen bestimmt. Sie enthalten die eigentlichen Widgets-Funktionen, die dem Benutzer das Interagieren mit dem Programm ermöglichen. Die für die GUI verwendeten Widget-Funktionen sind: *WIDGET_LABEL* für Beschriftungen und Hinweise, *WIDGET_TEXT* für die Parametereingabefelder, *WIDGET_BUTTON* für diverse Knöpfe und das Pulldownmenü, sowie eine *WIDGET_DRAW*-Funktion für das Zeichenfeld. Durch die Anweisung *WIDGET_CONTROL,totalbase,/REALIZE* wird das Widget als Oberfläche gestartet. „*totalbase*“ stellt in der Anweisung den Namen des in der Hierarchie ganz unten liegendem Widget dar, auf dem alle anderen platziert sind [28]. Nachdem die GUI erzeugt ist, wird eine Struktur erstellt, die alle benötigten Steuerparameter beinhaltet, die nicht von der Struktur des Event-Handlers berücksichtigt werden. Weiterhin wird ein Zeiger auf die Struktur erstellt, um sie von den Funktionen im Event-Handler auslesen oder beschreiben zu können. Durch die Anweisung *XMANAGER,'gezeiten_gui',totalbase* wird der XMANAGER gestartet. Durch die Angabe

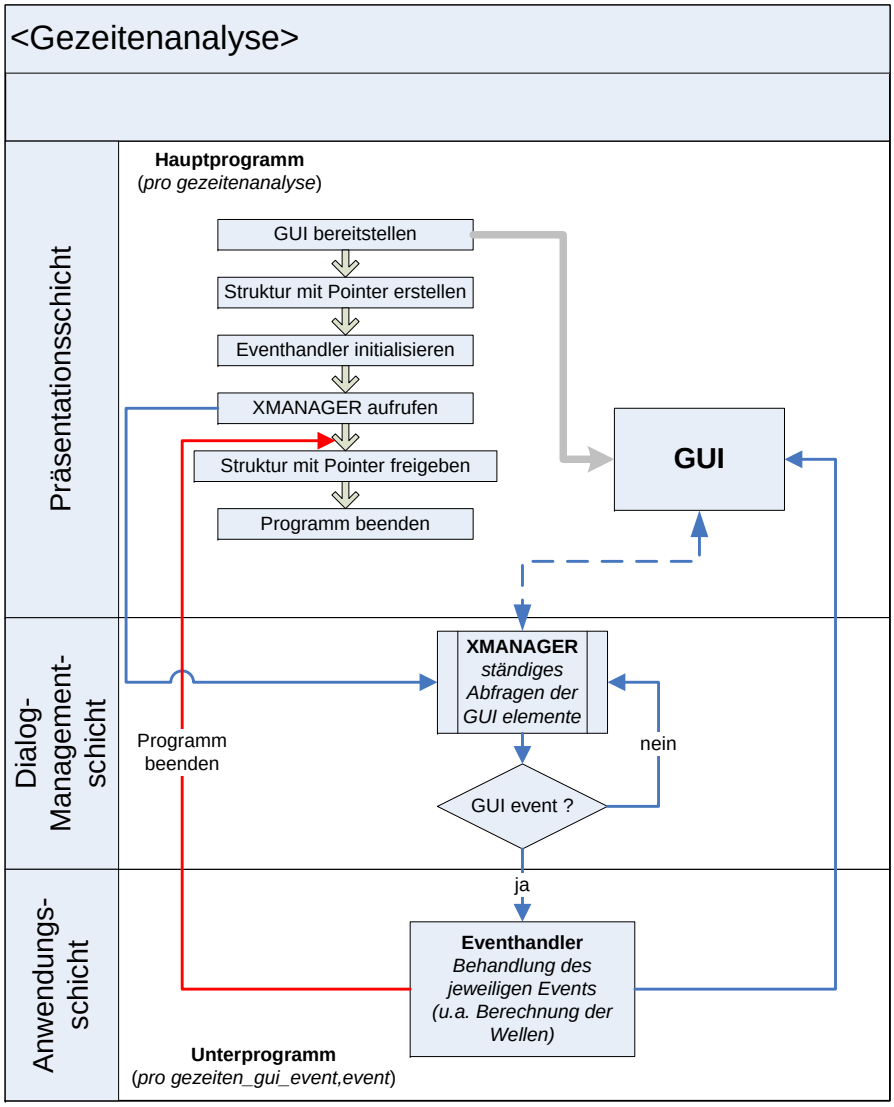

Abbildung 4.11: Funktionsschema des Gezeitenanalyseprogramms unter Berücksichtigung des Seeheim Schichtenmodells.

von *'gezeiten_gui'* wird der Name der Event-Handler Routine festgelegt, der dann *'gezeiten_gui_event'* lauten muss um vom XMANAGER aufgerufen zu werden. Der XMANAGER ist ein IDL-Programm, das ständig abfragt, welche Elemente auf der Oberfläche betätigt werden. Wenn ein Element betätigt wurde, wird der Name, der das Widget eindeutig identifiziert, an den XMANAGER übergeben. Diese Information wird in der so genannten Event-Struktur abgespeichert. Die Event-Handler-Routine wird vom XMANAGER aufgerufen, sobald eine Eingabe auf der

GUI getätigt wurde. Der Name des Widgets, das betätigt wurde, wird mit *WIDGET_INFO* ausgelesen. Dieser Name steht dann bei einer CASE-Anweisung vor dem Doppelpunkt und kann entsprechend verarbeitet werden. Mit *WIDGET_INFO* ist es auch möglich die Eingabewerte der Textfelder auszulesen. In der Behandlungsroutine des *OK*-Buttons findet die Berechnung und Visualisierung der Wellen statt. Zur Berechnung der einzelnen Bilder, sind Felder der Dimensionen 360 × 360 gewählt. So wird in der Ost- West- Einteilung mit 360 Längengraden und in der Nord- Süd- Einteilung mit 180 ($\frac{360}{2}$) Breitengraden gerechnet. Die Dauer der Simulation beläuft sich auf 24 Stunden. Die zeitliche Auflösung ist für die direkte Animation mit 36 Zeitschritten (je 40 min.), für die Bildergenerierung und den MPEG-Film mit 360 Zeitschritten (je 4 min.) festgelegt. Um eine flüssige Animation in der GUI zu gewährleisten, werden erst alle Bilder berechnet und im Nachhinein hintereinander ausgegeben. Daraus ergeben sich Felder der Größen 360 × 360 × 36 und 360 × 360 × 360. Die Werte der Wellen werden separat berechnet, nachträglich miteinander addiert und mit einem Wichtungsfaktor multipliziert. Das Prinzip der Visualisierung ist in den Abbildungen 4.12 erläutert. In den Bildern 4.12 a, b und c sind immer dieselben Werte derselben Sinusfunktion zu sehen. Bild a zeigt einen

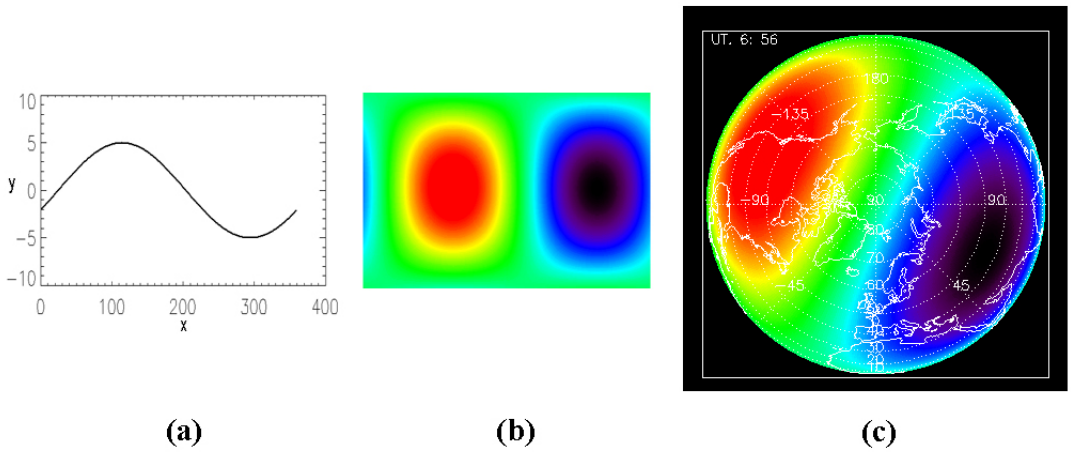

Abbildung 4.12: Sinuswelle in verschiedenen Darstellungen. Bild a zeigt eine Schwingung mit einer Periodendauer von X = 360 und einem Amplitudenausschlag von Y = 5 bis -5. In b ist dieselbe Schwingung in 3 Dimension en dargestellt X = 360 Y = 90 und Z = -5 bis 5 mittels Farbkodierung dargestellt, mit zusätzlicher vertikalen positiven Halbwelle. In c ist das Feld aus b als polarstereographische Projektion zu sehen.

Graph, mit den Werten der Funktion aus einem eindimensionalen Feld der Größe 360. Im Bild b wurde dieses Feld 180 mal untereinander, aneinander gereiht, sodass ein zweidimensionales Feld der Größe 360 × 180 entstanden ist. Durch eine farbliche Darstellung der verschiedenen Wertigkeiten entsteht eine dritte Dimension. Alle Vektoren dieses Feldes besitzen dieselben Werte der im Bild horizontal verlaufenden Sinusfunktion. Dieses Feld wird mit einem Vektor entlang der Y-Achse multipliziert, der die Werte einer positiven Sinushalbwelle besitzt. So entsteht eine vertikale Wichtung des Feldes. In Bild c ist dieses Feld mit einer IDL-Funktion in eine Polarstereographische Projektion transformiert worden. Der obere Rand von Bild b ist zu einem Punkt umgerechnet worden und repräsentiert den Nordpol. Der untere Rand stellt den Äquator dar. Die Schleifen, in denen die Werte berechnet werden, sind im folgendem Quellcodebeispiel zu sehen.

```
466 for izeit = 1,360,step do begin
467
468 for j = 180,360 do begin  ;y-achse
469  gewicht=arrw(j-1)
470  for i = 1,360 do begin   ;x-achse
471
472 ;Sinusfunktion----------------------------------------------------------
473
474 arr1(i-1,j-1) = ampli[0] * sin(-2. * !PI * float(izeit) / period[0] + 2. * $
475                                                 !PI * float(i) / horil[0])
476 arr2(i-1,j-1) = ampli[1] * sin(-2. * !PI * float(izeit) / period[1] + 2. * $
477                                                 !PI * float(i) / horil[1])
478 arr3(i-1,j-1) = ampli[2] * sin(-2. * !PI * float(izeit) / period[2] + 2. * $
479                                                 !PI * float(i) / horil[2])
480 arr4(i-1,j-1) = ampli[3] * sin(-2. * !PI * float(izeit) / period[3] + 2. * $
481                                                 !PI * float(i) / horil[3])
482
483 arr(i-1,j-1,index)=gewicht*(arr1(i-1,j-1)+arr2(i-1,j-1)+arr3(i-1,j-1)+ $
484                                                 arr4(i-1,j-1))
485
486 end
487 end
488 index++
489 end
```

Die äußere Schleife stellt die Zeitschritte der einzelnen Bilder dar. In den beiden inneren Schleifen werden die Felder für die Einzelbilder erstellt. Da die Werte auf der Seite vom Äquator bis zum Südpol nicht zu sehen sind, werden sie auch nicht berechnet, um Rechenzeit zu sparen. In *ampli* sind die eingegebenen Amplitudenwerte der

einzelnen Wellen vorhanden, in *period* die Periode $\times \frac{360}{24}$ und in *horil*, $\frac{360}{Wellenanzahl}$. In der Zeile 483 werden die Werte der einzelnen Wellen miteinander addiert und mit dem Gewichtungsfaktor für den jeweiligen Y-Bereich multipliziert. Der daraus resultierende Wert wird im Feld für die Visualisierung festgehalten. In Abbildung 4.13 wird der Effekt der Überlagerung verschiedener Schwingungen veranschaulicht. In Bild a sind die drei unterschiedlichen Schwingungen einzeln für sich dargestellt und in b als Superposition zusammengefasst. Nachdem das Feld für die Visualisierung erstellt ist, wird der Maximal- und Minimalwert ermittelt, sodass der Wertebereich für die farbliche Darstellung definiert werden kann. Danach folgen, durch if-Abfragen getrennt, die Visualisierungsroutinen. In der Routine für die GUI-Animation wird mit der *MAP_IMAGE*-Funktion das jeweilige Feld in die polarstereographische Projektion transformiert und an das Zeichenfeld auf der GUI mit der *TV*-Funktion übergeben. Die Feldtransformationen geschehen genauso in der Routine für die Einzelbildgenerierung, in der dann die Bilder mit der *WRITE_IMAGE*-Funktion im PNG-Format gespeichert werden. Auch wenn die Felder effektiv nur zur Hälfte mit Werten gefüllt sind, muss die nicht visualisierte Hälfte als Dimensionsumfang

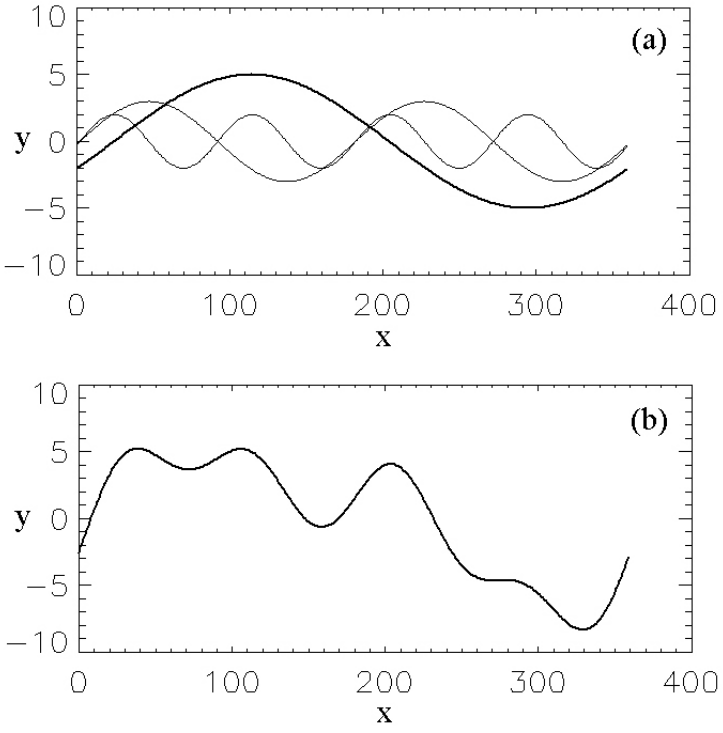

Abbildung 4.13: Überlagerung von drei harmonischen Wellen.

erhalten bleiben, weil die *MAP_IMAGE*-Funktion die Felder auf die gesamte Kugel streckt. Ein Endresultat ist in Abbildung 4.12 c zu sehen. Im Bild wurden zusätzlich Kontinentenumrandungen und ein Gradnetz eingefügt, um eine geographische Orientierung zu ermöglichen. Für eine zeitliche Orientierung wurde in der oberen linken Ecke eine universelle Zeitangabe eingefügt. Wenn der *MPEG*-Button betätigt wird, wird ein Dialogfenster geöffnet in dem der Speicherpfad und der Dateiname festgelegt werden kann. Sobald der Speichern-Button in dem Dialogfenster betätigt wurde öffnet sich ein Popupfenster über der GUI, das den Benutzer darüber informiert, dass der Film zurzeit erstellt wird. Während dieser Zeit ist die GUI inaktiv und es können keine Eingaben getätigt werden. Die Statuszeile unter dem Zeichenfeld gibt während dessen Auskunft über den Fortschritt der Aktion. Die Werteberechnung und Bilderstellung erfolgt hier nachdem selben Schema wie für die Bildgenerierung beschrieben wurde. Der MPEG-Film wird erstellt, indem durch die *MPEG_PUT*-Funktion jedes einzelne Bild in dem Film eingefügt wird. Für die nachträgliche Filmerstellung aus den Einzelbildern der Bilderstellungsroutine bietet sich die Software VideoMach an. Mit ihr kann in kürzester Zeit ein Film mit höchster Qualität und vielen Filmformaten aus den sequentiell nummerierten Bildern erstellt werden. Die Funktion des Programms wurde während der Entwicklungsphase durch immer neue Ideen und Wünsche immer weiter entwickelt. So konnten, bis zum Schluss der Bearbeitungszeit in der vorliegenden Diplomarbeit, nicht alle Ideen umgesetzt werden. Ein weiterer Vorschlag für eine Erweiterung des Programms ist z.B., dass die GUI-Fläche auf den gesamten Bildschirm vergrößert werden kann und optional weitere Ausgabefenster geöffnet werden können, in denen vertikale Wellenprofile vorher definierter geographischer Punkte angezeigt werden.

Kapitel 5

Test und Fazit

In diesem Kapitel werden Testergebnisse der erstellten Programme aufgeführt und ausgewertet.

5.1 Automatisierte Ergebnisvisualisierung

Die automatisierte Ergebnisvisualisierung wurde auf Programmlaufstabilität, auf Fehler in den unterschiedlichen Plotvarianten und auf Wertebereichsüberschreitungen getestet. Diese Test konnten nur exemplarisch durchgeführt werden, da eine ausführliche Testung aller möglichen Programmauswahlen den Rahmen dieser Arbeit überschritten hätten. Die Anzahl der verschiedenen Auswahlmöglichkeiten, zwischen denen der Benutzer wählen kann, belaufen sich auf insgesamt 126. Diese ergeben sich aus zwei möglichen Bildformaten, drei Projektionsauswahlen, die Auswahl von drei Plotarten und sieben Möglichkeiten die verschiedenen LIMA-Größen zu visualisieren. Aus einem LIMA-Datenergebnisfile können mit dem Programm 2832 verschiedene Bilder erzeugt werden. Diese ergeben sich aus den sechs LIMA-Größen, zwei Projektionsarten, zwei Visualisierungsarten, und 118 Modellhöhen. Der Testlauf auf Fehler in den verschiedenen Plotvarianten beschränkte sich auf die beiden Bildformate in Verbindung mit kombinierter Projektionsart und kombinierter Plotart. Das heißt, dass die beiden möglichen Projektionsarten und die beiden möglichen Plotarten in einem Programmlauf visualisiert wurden. Bei diesem Test sind keinerlei Fehler festgestellt worden. Die Tests auf Programmstabilität und Wertebereichsüberschreitungen sind in einem Testlauf durchgeführt worden, da für beide

Tests eine große Anzahl verschiedener LIMA-Datenfiles verarbeitet werden müssen. Für diese Tests sind exemplarisch eine LIMA-Größe und eine Modellhöhe gewählt worden. Der Zeitraum für die gewählten LIMA-Daten belief sich auf ein Jahr, also 1440 LIMA-Files. Von vier Läufen wurde einer mit der Fehlermeldung „*Error occurred at: WRITE_IMAGE 123*“ unterbrochen. Der Fehlercode, mit der dazugehörigen Fehlermeldung, lässt darauf schließen, dass das Programm beim Schreiben einer Bilddatei auf den DMF unterbrochen wurde. Der Errorhandler, in dem Programm der automatisierten Ergebnisvisualisierung, sollte für diesen Fehlercode erweitert werden. Im aktuellen Programm behandelt er nur Fehler, die beim Lesen einer Datei auftauchen. Der aufgetretene Fehler ist ungewöhnlich, da im Allgemeinen Fehler beim Lesen vom DMF auftauchen, die aus den Speicherbandoperationen des DMF resultieren. Beim Schreiben auf den DMF entfällt jedoch dieser Vorgang. Die Laufzeit der drei anderen Programmdurchläufe beliefen sich auf 18 h 30 min, 20 h 30 min und 21 h 30 min. Das Programm benötigt während der Laufzeit auf der APOLLO 145 MB Hauptspeicher, wie im Auszug der Prozesstabelle in Abbildung 5.1 gezeigt wird. Die unterschiedliche Laufzeit, resultiert wahrscheinlich aus der unterschiedlichen Auslastung des DMF. Die Testläufe, die für die Auswertung der Wertebereichsüberschreitungen herangezogen wurden, haben LIMA-Größen in drei unterschiedlichen Modellhöhen visualisiert. In den Programmläufen für die Modellhöhe 2 und der LIMA-Größe Temperatur, sowie in der Modellhöhe 75, für den Vertikalwind sind keine Wertebereichsüberschreitungen aufgetreten. In der Modellhöhe 42, für die Temperatur sind Wertebereichsüberschreitungen aufgetreten, die die Werteskala nach oben und nach unten für große Flächen im Bild überschreiten (Abb. 5.2 a). Anhand der Bilder, mit denen die Zeitpunkte der Wertebereichsüberschreitungen bestimmt werden können und den ermittelten Wertebereichsgrenzen der einzelnen

CPU	TTY	PID	USERNAME	PRI	NI	SIZE	RES	STATE	TIME	%WCPU	%CPU	COMMAND
5	?	7261	oa	152	20	27344K	1088K	run	31373:24	100.17	100.00	idl
3	?	20486	oa	241	20	67216K	62336K	run	116:05	100.16	99.98	main2.exe
6	?	4189	oa	241	20	175M	171M	run	5483:11	99.96	99.78	lic1.exe
7	?	29965	oa	241	20	1441M	1191M	run	5516:32	99.32	99.15	lima9.exe
4	?	15405	oa	241	20	62480K	57664K	run	2913:07	98.82	98.65	main.out
0	?	26636	oa	152	20	60184K	37012K	run	7:48	96.54	96.37	idl
2	pts/13	19921	kd	154	20	3004K	440K	run	38:28	61.11	61.00	ftp
1	?	7190	root	152	20	384K	0K	run	104:12	29.88	29.83	nfskt
1	pts/16	26818	oa030	152	20	145M	122M	run	0:00	2.02	1.65	idl
1	?	40	root	152	20	1856K	0K	run	95:41	1.18	1.18	vxf
1	?	722	root	154	20	32K	32K	sleep	1081:14	0.84	0.84	bi

Abbildung 5.1: Laufendes Programm in der APOLLO-Prozesstabelle.

LIMA-Files, die im Logfile festgehalten sind, kann nachträglich eine Anpassung der Wertebereichsfestlegung erörtert werden. Im Gegensatz dazu, werden im Programmlauf der Modellhöhe 42, über fünf Monate die ausgelesenen Werte durch die Farbskala zu schlecht aufgelöst, sodass die Plots für diese Zeit quasi mit einer Farbe ausgefüllt sind, wie in Abbildung 5.2 b zu sehen ist. Die PostScript-Dateien der Konturplots nehmen ein Datenvolumen von, je nach Komplexität des Bildes, ca. 5 bis 15 MB ein, die Direktplots ca. 1,4 MB. Bei den PNG-Dateien sind die Direktplots nur geringfügig kleiner als die Konturplots. Das benötigte Datenvolumen für ein Bild beläuft sich auf 18 bis 85 KB. Ein Bilderordner, der 1440 PNG-Dateien für die Visualisierung einer LIMA-Größe eines Jahres auf einer Modellhöhe beinhaltet, ist ungefähr 100 MB groß. Die Erstellung eines Films auf dem LINUX-Computingserver aus Bildern eines Jahres, ist nach ungefähr 45 min. abgeschlossen. Der daraus resultierende Film ist, je nachdem ob er mit interpolierten Zwischenbildern erstellt wurde, ca. 85 bis 130 MB groß und ohne zusätzliche Zwischenbilder ca. 32 MB bis 38 MB groß. Die Entwicklung des Programms für die automatisierte Ergebnisvisualisierung ist zum größten Teil abgeschlossen. Für eine endgültige Version, sollten die Wertebereichsgrenzen und die Errorhandler-Routine den Ergebnissen der Tests angepasst werden. Weiterhin sollte das Programm auf die neue Ordnerstruktur, wie sie im Abschnitt 4.1.5 besprochen wurde, eingestellt werden. Die Komplexität der Automatisierung

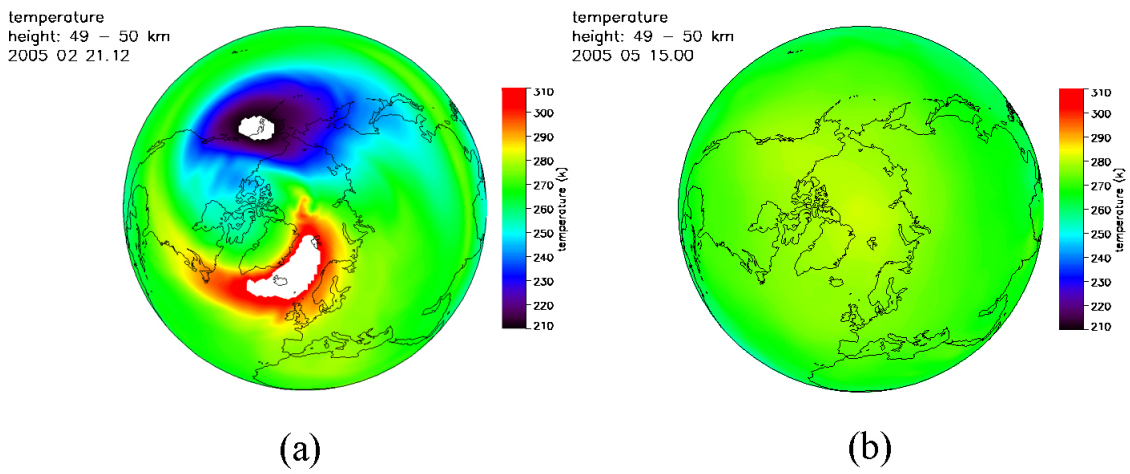

Abbildung 5.2: Plot mit Wertebereichsüberschreitungen in a und zu groß aufgelöstem Wertebereich in b.

nimmt durch den Umstand, dass mit IDL keine verwendbaren Filme erstellt werden können, sehr zu. Da Creaso © für IDL unter Anderem mit dem Argument der Filmerstellung wirbt, bleibt zu erwarten, dass sich auf diesem Gebiet in naher Zukunft etwas verändert.

5.2 Gezeitenanalysetool

Das Gezeitenanalysetool wurde auf zwei unterschiedlichen PC-Systemen unter Windows getestet. Bei der Laufstabilität, sowie bei Darstellung der Werte in den Bildern, traten keine Fehler auf. Die Rechenzeiten der Simulationsläufe auf den unterschiedlichen PCs sind in Tabelle 5.1 aufgeführt. Zur Laufzeit benötigt die IDL-Anwendung, die die Berechnungen ausführt, für die Animation in der GUI 70 MB Hauptspeicher, für die Bildergenerierung und den MPEG-Film 240 MB Hauptspeicher. Die erstellten Einzelbilder sind je ca. 40 KB bis 55 KB groß, alle 360 Bilder nehmen zusammen ein Datenvolumen von ca. 14 bis 20 MB ein. Die Filme, die nachträglich aus den Bildern mit VideoMach in maximaler Qualität erstellt wurden, sind ca. 9,5 MB groß, die mit der IDL-MPEG Routine erstellt werden ca. 1,7 MB. Das Gezeitenanalysetool ist, so weit es die Anforderungen der Aufgabenstellung angeht, fertig gestellt. Die gewünschten Erweiterungen die in Abschnitt 4.3 besprochen wurden sollten jedoch noch umgesetzt werden. Die Berechnungszeiten der einzelnen Wiedergabeaktionen, dauern auf älteren Rechnern verhältnismäßig lang,

	PC 1	PC 2
Konfiguration	P4/1,6GHz/2GB RAM/XP	P4/3,2GHz/1GB RAM/XP
Animation	45 s.	18 s.
Wiederholung	1 s. <	1 s. <
Bildergenerierung	9 min.	3 min. 56 s.
MPEG-Film	11 min. 55 s.	4 min. 44 s.

Tabelle 5.1: Übersicht Gezeitenanalysetooltests auf den unterschiedlichen PC-Systemen. Die Zeile *Konfiguration* gibt die Eigenschaften des PCs wie folgt an: Prozessortyp / Prozessortakt / Hauptspeicher / Betriebssystem. Die darunter liegenden Zeilen geben die Berechnungszeiten der jeweiligen Aktionen an.

im Gegensatz zu aktuellen Rechnerkonfigurationen. Für Rechnersysteme mit weniger als 256 MB Hauptspeicher, erhöht sich die Rechenzeit bei der Bildergenerierung und der MPEG-Filmerstellung noch einmal durch den hohen Hauptspeicherverbrauch. Für aktuelle Rechnersysteme stellt die Rechenzeit aber kein Problem dar. Eine Einschränkung, die für alle Programme gilt, die in IDL implementiert werden, ist, dass sie nur in einer IDL-Umgebung gestartet werden können. Da der Erwerb von IDL relativ teuer ist, ist die Verbreitung dieser Software sehr eingeschränkt.

Kapitel 6

Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurden Methoden untersucht und umgesetzt, LIMA-Modellergebnisdaten automatisiert auf vorhandenen Computingservern auszuwerten und zu visualisieren. Dafür wurde ein Programm entwickelt, dass die verschiedenen LIMA-Daten durch Eingabeparameter gesteuert, in der gewünschten Form visualisiert und die daraus resultierenden Bilddateien in einem vordefinierten Verzeichnis ablegt. Der Computingserver, auf dem das Programm läuft, besitzt ein UNIX-Betriebssystem, sodass die Automatisierung des Programms durch UNIX-Skripte unterstützt werden konnte. Aus der großen Menge von Bildern werden mit zwei Konsolensoftwaretools Filme erstellt. Dieser Vorgang wurde wieder mit Hilfe von Skripten automatisiert. Außerdem konnten die für die Filmerstellung nützlichen Eigenschaften dieser beiden Programme mit den Skripten kombiniert werden. Dieser Vorgang musste allerdings auf einen LINUX-Server verlagert werden, da die Programme zur Filmerstellung auf dem Betriebsystem des Computingserver nicht korrekt bzw. gar nicht ausführbar sind. Da in naher Zukunft im IAP die Anschaffung eines neuen Computingservers mit LINUX-Betriebssystem geplant ist, können die beiden Arbeitsschritte der Bilderstellung und Filmerstellung durch Erweiterung der Skripte kombiniert werden. So kann der Grad der Automatisierung der derzeitigen Lösung noch erhöht werden. Um eine vollständige Automatisierung dieser Aufgaben zu erreichen, müssen das Programm der Ergebnisvisualisierung und die Skripte dahingehend erweitert werden, dass sie z.B. eigenständig detektieren, wenn ein neues LIMA-Ergebnisfile erstellt wurde. Weiterhin sollten sie in der Lage sein eigenständig eine Verzeichnisstruktur für die fortlaufend erstellten Bilder anzulegen.

Es wurde in dieser Arbeit weiterhin untersucht, wie aus den LIMA-Ergebnisdaten ein Strömungsfilm nach den Vorgaben eines Beispielfilms der Universität Basel erzeugt werden kann. Durch Nachforschungen im Internet und in Fachliteratur, wurde für die Umsetzung eine Grundlage geschaffen. Außerdem wurde eine Software zur Simulation von Gezeitenwellen erstellt, die es ermöglicht, verschiedene Kombinationen von Gezeitenwellen in der Atmosphäre darzustellen. Sie wird über eine grafische Benutzerschnittstelle bedient und ist in der Lage die Interaktion von vier Wellen zu simulieren. Die daraus resultierende Animation kann wahlweise auf der GUI wiedergegeben werden, als Einzelbilder für die externe Filmerstellung abgespeichert werden oder direkt als MPEG-Film gespeichert werden. Die Software sollte durch die Ideen und Wünsche der Anwender weiterentwickelt werden. Die Programme, die in der Bearbeitungszeit der vorliegenden Diplomarbeit erstellt wurden, liegen in Versionen vor, die die an sie gestellten Anforderungen weitestgehend erfüllen. Ihre Entwicklungen sind aber noch nicht in der Endversion, da noch Erweiterungsmöglichkeiten vorhanden sind, die die Anwendung der Programme noch verbessern können.

Kapitel 7

Danksagung

Ich danke Herrn Prof. Dr. Lübken für die Aufnahme in das Institut. Herrn Dr. Berger und Herrn Dr. Baumgarten danke für die Aufnahme in die Arbeitsgruppe und für ihre ständige Diskussionsbereitschaft sowie die vielen wertvollen Anregungen und Hinweise bei der Erstellung dieser Arbeit. Außerdem möchte ich mich noch bei allen nicht namentlich genannten Mitarbeitern des Instituts bedanken, die mich durch ihr freundliches Entgegenkommen und ihrer Hilfe bei dieser Arbeit unterstützt haben.

Literaturverzeichnis

[1] *D.Rachholz : Untersuchungen zur automatisierten Filmerstellung einer Eiswolke; Hochschule Wismar, IAP-Kühlungsborn, Belegarbeit*, 2005

[2] *Prof. Dr. F. -J. Lübken, Prof. Dr. G. Schmitz, Dr. J. Bremer : Institutsbericht 2004/2005, IAP-Kühlungsborn* , 2005

[3] *G.Sonnemann : Ozon - Natürliche Schwankungen und antropogene Einflüsse. Akademie Verlag* , 1992

[4] *www.kowoma.de*, 2006

[5] *P Fabian : Atmosphäre und Umwelt, 4. Auflage. Springer-Verlag*, 1992

[6] *Jens Oberheide : Messung und Modellierung von Gezeitenwellen in der mittleren Erdatmosphäre: Ergebnisse des Crista-Experiments, Bergische Universität, Dissertation*, 2000

[7] *A. Voss : Das Große PC & Internet Lexikon., Data Becker Verlag*, 2004

[8] *M. Precht, N. Meier, D. Tremel : EDV - Grundwissen. Eine Einführung in die Theorie und Praxis der modernen EDV, 4. Auflage. Addisson Wesley Verlag*, 2004

[9] *P. Winkler : M + T Computerlexikon. Markt + Technik Verlag*, 2003

[10] *K. R. Gegenfurtner : Farbwahrnehmung. www.allpsych.uni-giessen.de/karl/ - teach/farbe.html*, 2006

[11] *A. Light, P. J. Bartlein : The End of the Rainbow? Color Schemes for Improved Data Graphics. Eos, Vol. 85, No. 40, 5. October* 2004

[12] *www.creaso.com*, 2006

[13] *Meteoblue - Numerical Weather Prediction; www.unibas.ch/geo/mcr/3d/ - meteo/index.dt.htm*, 2006

[14] *www.gromada.com/videomach.html*, 2006

[15] *www.imagemagick.org*, 2006

[16] *mjpeg.sourceforge.net/*, 2006

[17] *R. Fischbach : Schwierige Abgrenzungen. Von den Job Control Languages bis Perl und Python. In iX 12/99. S. 60*, 1999

[18] *H. Schwichtenberg : Windows Scripting. Automatisierte Systemadministration, 3. Auflage . Addison-Wesley Verlag*, 2003

[19] *M. Bender M. Brill : Computergrafik. Ein anwendungsorientiertes Lehrbuch. Carl Hanser Verlag*, 2003

[20] *D. Etling : Theoretische Meteorologie. Eine Einführung, Vieweg Verlag*, 1996

[21] *R. v. Ammon : Ungeschickt. Erfahrungen mit grafischen Benutzerschnittstellen. In iX 06/95. S. 120*, 1995

[22] *S. Keller : Grafische Oberflächen; www.erde.fbe.fh-weingarten.de/keller/ - Downloads/grabo/*, 2000

[23] *M. Busch; R. Bauer; H. Heer; M. Wagener: Praxisbezogene IDL Programmierung. Forschungszentrum Jülich : Zentralbibliothek Jülich*, 2002

[24] *Fanning Software Consulting; David Fanning, Ph.D.; Fort Collins USA ; Coyote's Guide to IDL Programming : http://www.dfanning.com*, 2004

[25] *RSI; Using IDL version 6.0*, 2003

[26] *www.wendler-im-netz.de/informatik/studies/met.html*, 2006

[27] *L. E. Gumley : Practical IDL Programming. Creating effective data analysis and visualization applications. Academic Press*, 2002

[28] *N. Hahn : Überblick IDL Programmierung. www.tu-darmstadt.de/hrz/ - software/grafik/idl.tud*, 2006

Hiermit bestätige ich, die vorliegende Arbeit selbstständig und nur unter Zuhilfenahme der angegebenen Quellen und Hilfsmittel verfasst zu haben.

Wismar, den 2. Oktober 2006

(Dirk Rachholz)

Anhang

- Thesen
- Programmdokumentation: Automatisierte Ergebnisvisualisierung
- Programmdokumentation: Gezeitenanalysetool
- CD mit Programmen und Ergebnissen

Thesen

- IDL bietet nur eine mangelhafte Software für die Erstellung von Filmen an.
- Schnelle Visualisierungen von großen Datenmengen sollten sich zur Fehlererkennung in Modellen eignen.
- Eine Gute Visualisierung sollte den Informationsgehalt einer Datenbank schnell und verständlich auch für „fachfremde“ Betrachter darstellen.
- Eine gute Visualisierung ist bei meteorologischen Themen anscheinend schwer umzusetzen.
- Farbumgebungen wirken jeweils anders in wechselnden Medien.
- Diverse Visualisierungstechniken sollten vermehrt im Unterricht eingesetzt werden, um mathematische Sachverhalte verständlich zu machen.
  Nach dem Motto: „Ein Bild sagt mehr als tausend Worte.“