Thomas Orgis

Lagrangesche Traceradvektion in einem globalen gekoppelten Klimamodell

Diplomarbeit im Studiengang Physik an der Universität Potsdam

21. Mai 2007

Φ

Ein Physiker hat Daten oder macht sich welche.

Dann sieht er sie sich an.
Dann denkt er etwas nach
Dann redet er darüber.

Wenn er Glück hat, hören ihm andere zu.

Vielleicht schreibt er seine Erkenntnisse nieder, so dass andere mit größerem Δt bzw. Δs sie verfolgen können.

Ich bin ein Physiker. Ich bin mir nicht sicher was die Erkenntnis betrifft, jedoch werde ich nun etwas niederschreiben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltexte

1 Einleitung

Das Thema dieser Arbeit ist die Traceradvektion im Windfeld eines globalen gekoppelten Klimamodells. Wesentlich dabei ist das Globale: Der Raum ist die Erdatmosphäre.

In erster Näherung ist dies der Raum zwischen zwei konzentrischen Kugelschalen, wobei der Abstand zwischen den Schalen klein ist im Vergleich zu deren Durchmesser. Simulationen des Klimamodells liefern multivariate[1] Zeitreihen verschiedener Größen (dreidimensionaler Wind, Geopotential, Temperatur, chemische Konzentrationen, ...) auf einem Raster im üblichen geographischen Gradnetz, was bezogen auf die Erdoberfläche ein nicht-kartesisches Netz bedeutet. Desweiteren ist die vertikale Richtung in Form von Duckkoordinaten gegeben, deren Bezug zu Ortskoordinaten (in Metern) räumlich und zeitlich variiert.

Mein Hauptaugenmerk liegt in den Transporteigenschaften des räumlich und zeitlich hoch variablen dreidimensionalen Windfeldes. Mittel dieser Untersuchung ist der numerisch berechnete Transport von virtuellen passiven Tracerteilchen bzw. den repräsentierten Luftmassen. Wegen der komplexen Struktur der Felder (zeitlich variablel, große Gradienten bei grober Auflösung, langer Zeitraum) ist die Einzeltrajektorie numerisch sehr unzuverlässig. Sie hängt empfindlich von Parametern und Implementation des Algorithmus sowie verwendeten Compileroptionen / Rechengenauigkeit ab. Deshalb folgt der Ansatz, eine große Menge dieser Trajektorien im Ensemble zu betrachten und so zu sinnvollen Aussagen zu gelangen.

Die Bewegung von großen Ensembles soll über statistische Auswertung bzw. Analyse von aus Zeitreihen abgeleiteten Dichten und Mischungsverhältnisse Transporteigenschaften mit wesentlichen Barrieren und Austauschpunkten liefern.

Die hier vorgestellte Arbeit umfaßt die ersten Schritte der Transportanalyse, den grundlegenden Ansatz, der aus den modellierten Windfeldern den dreidimensionalen Transport von passiven Tracerensembles berechnet. Dabei ist die Methodik im Prinzip nicht neu. Die numerische Integration von Trajektorien in gegebenen Windfeldern ist nicht neu.

Neu ist vieleicht das Ausmaß und die Ignoranz des Vorhabens. Ich möchte globalen Transport in Länge, Breite und Höhe berechnen, allein gestützt auf die Windfelder. Es gibt keine Rücksicht auf Erhaltung fluiddynamischer Größen (wie in [Stohl1998] beschrieben) oder gar generelle Beschränkung auf als getrennt betrachtete Luftschichten. Die weitere Physik sowie die Problematik des unterschiedlichen Charakters der horizontalen und vertikalen Winde wird bewußt beiseite gestellt. Ich untersuche ein einheitliches globales Schema zum Transport in der Atmosphäre, welches auch an den Polen funktioniert. Es geht um die Frage, in wie weit solch ein "blinder"[2] Ansatz mit heute verfügbarer Rechentechnik durch die Betrachtung großer Mengen von individuellen Trajektorien zu sinnvollen Aussagen führt.

[1] Im konkret verwendeten Gitter sind dies Zeitreihen von Vektoren mit $48 \cdot 96 \cdot 23 = 105984$ Dimensionen, bzw. beim Wind aus drei Komponenten 317952.

[2] ☺ Blind im nichtbiologischen Sinne von "unvoreingenommen", "ohne weiteres Vorwissen" – bzw. *Augen-zu-und-durch*-blind ... nicht *gegen-die-Wand*-blind oder *den-Wald-vor-Bäumen-nicht*-blind.

Ich werde meine Methodik zur Berechnung der Bewegung großer Partikelensembles vorstellen, zusammen mit dem entworfenen Rechnerprogrammpaket unter dem Arbeitstitel PEP-Tracer (verfügbar unter [PEP-Tracer]). Beides stellt einen in mancherlei Hinsicht (noch) primitiven Ansatz mit heutigen Werkzeugen dar und eine wesentliche Eigenschaft der Programme ist die Ausbaufähigkeit[3]. Die "heutigen Werkzeuge" sind leistungsstarke Rechner – vernetzt zu Clustern – in großer Anzahl und mit viel Speicherplatz[4] sowie objektorientierte Programmierung, um neben dem Wachstum der Rechenleistung auch das Wachstum der Programme geordnet zu ermöglichen.

Nach Vorstellung der Methodik im Detail werde ich erste Langzeitläufe im Hinblick auf eine Konsistenzprobe gegenüber dem GCM[5] auswerten. Diese besteht in der Erhaltung der vom Modell bestimmten Masseverteilung (durch Druck, Temperatur, Feuchte und Volumen) beim numerischen Transport von globalen Startpositionierungen über die Zeit.

Ich verwende hauptsächlich drei Sprachen:

1. Deutsch im Haupttext.

2. Mathematische Syntax und Symbole in einfachen Formeln sowie formalen Darstellungen von Algorithmen.

3. ANSI/ISO[6] C++ für Beispiele konkreter Implementationen (Auszüge aus dem dieser Arbeit zugrundeliegendem Programmpaket). Eingebettete Kommentare sind in englisch, wie auch fast alle Symbolbezeichnungen. Die C++ Standard Template Library wird ebenfalls verwendet.

Es wird vorkommen, dass ich ein und denselben Sachverhalt in allen drei Weisen darstelle, was hoffentlich weniger als unnötige Redundanz denn als die tiefere Einsicht fördernde Betrachtung aus verschiedenen Blickwinkeln erscheint.

1.1 Symbole und Syntax

Nicht alle von mir verwendeten mathematischen Schreibweisen sind durch etablierte Konventionen offensichtlich. Daher listet Tab. 1.1 einige syntaktische Elemente auf, die vielleicht der Klarstellung bedürfen.

In Anlehnung an den trinären Operator in C++ nutze ich bei der mathematischen Darstellung von Programmlogik folgende Symbolik für bedingte Anweisungen oder auch nur Ausdrücke als Teil von Anweisungen:

[3] Was ich als *positive* Eigenschaft verstanden wissen will.

[4] Nach heutigem Maßstab Arbeitsspeicher von mehreren GiB und Festplattenspeicher im TiB – Bereich. Ich gehe davon aus, dass diese Angaben in wenigen Jahren weniger beeindrucken als denn amüsieren.

[5] Global Climate/Circulation Model; das globale Klimamodell

[6] ISO/IEC 14882:1998 bzw.14882:2003, mitunter auch ANSI C++ genannt; Wesentlich ist hierbei der Grundsatz, nicht einen bestimmten Compiler und seine Erweiterungen, sondern den plattformübergreifenden Standard zu betrachten.

$[a;b]$	geschlossenes Intervall von a zu b
$[a;b)$	Intervall von a zu b, offen bei b
$(a;b]$	Intervall von a zu b, offen bei a
$(a;b)$	offenes Intervall zwischen a und b
$\{a;b\}$	Menge mit Elementen a und b
$\text{ganz}\,(x)$	Ganzzahliger Teil von x ($\text{ganz}\,(3,14) = 3$; $\text{ganz}\,(-3,14) = -3$)
$\text{rund}\,(x)$	Zur Ganzzahl gerundeter Wert von x ($\text{rund}\,(3,14) = 3$; $\text{rund}\,(3,5) = 4$)
$i,\ j\ k,\ n$	ganze Zahlen, meist als Indizes; ohne weitere Angabe gilt $i,j,k,n \in \mathbb{Z}$
$a := b$	Weise a den Wert b zu. Diese Schreibweise findet Verwendung, wenn der Wert von a durch weitere Zuweisungen geändert werden kann und so ein $a = b$ mathematisch nicht korrekt wäre. Es ist eine Art temporäres "=".

Tabelle 1.1: Vereinbarungen zur mathematischen Syntax

([Bedingung] **?** [Wenn-ja-Ausdruck] **:** [Wenn-nein-Ausdruck])

[Bedingung]	**?**	[Wenn-ja-Ausdruck 1] [Wenn-ja-Ausdruck 2] ...
	:	[Wenn-nein-Ausdruck 1] [Wenn-nein-Ausdruck 2] ...

Die erste Variante wird platzsparend in andere Ausdrücke eingebettet. Sie ist so zu lesen, dass an ihrer Stelle [Wenn-ja-Ausdruck] steht, wenn [Bedingung] zutrifft, und [Wenn-nein-Ausdruck], wenn [Bedingung] nicht zutrifft. Die zweite Variante steht für die Ausführung des einen oder anderen Anweisungsblocks, abhängig von [Bedingung]. Für die Bedingungen selbst kommt übliche boolsche Syntax wie in $(x = 0 \wedge y = 0) \vee z = 0$ ("wenn x und y gleich Null oder z gleich Null") zum Einsatz.

Ich verwende grundsätzlich das SI-Einheitensystem. Im Gesamtkontext dieser Arbeit haben einige mathematische Symbole durchgängige Bedeutung. Tab. 1.2 zeigt diese in einer Übersicht.

Die eingebundenen Programmtextbeispiele benutzen einige gemeinsame, nicht dem C++ Standard entspringende Symbole, die hier kurz beschrieben werden sollen. Der Datentyp **pep_t** steht für den verwendeten Fließkommazahlentyp, ist praktisch also identisch mit **double** oder **float**.

Wichtige benannte Konstanten sind in Tab. 1.3 zu finden. Diese Konstanten sind mehrheitlich redundant, helfen aber bei der Abstraktion[7] und zeigen in einer **for**-Schleife sofort, ob sie sich mit Koordinaten oder Geschwindigkeiten beschäftigt.

[7] ☺ sowie möglichst einfacher Umstellung der Programme auf 17 oder 42 Dimensionen

λ, β		geographische Länge (λ) und Breite (β), Grad oder Bogenmaß
h		globale Höhenkoordinate (Meter)
p		globale Druckkoordinate (Pascal) – meist genuzt an Stelle einer Höhenkoordinate
T		absolute Temperatur (Kelvin)
t		Zeit
R		Radius der idealen Erdkugel
$r_\mathrm{b}(\beta)$ $R\cos\beta$	=	Radius eines Breitenkreises bei β ($r_\mathrm{b}(0) = R$; $r_\mathrm{b}(\frac{\pi}{2}) = 0$)
$\vec{r} = \begin{pmatrix}\lambda\\ \beta\\ p\end{pmatrix}$		Ortsvektor / Koordinaten eines Partikels im globalen System
$\vec{x} = \begin{pmatrix}x\\ y\\ z\end{pmatrix}$		Ortsvektor / Koordinaten eines Partikels im lokalen System
$(x, y, z)_\mathrm{lok}$		explizite Kennzeichnung von Koordinaten im lokalen System (wenn aus Kontext nicht klar)
$\dot{\vec{r}} = \frac{\mathrm{d}}{\mathrm{d}t}\vec{r}$		Geschwindigkeitsvektor im globalen System
$\dot{\vec{x}} = \frac{\mathrm{d}}{\mathrm{d}t}\vec{x}$		Geschwindigkeitsvektor im lokalen System
o, a, e		Partitionsparameter für Länge, Breite, Höhe (in dieser Reihenfolge, von *l**o**ngitude*, *l**a**titude*, *l**e**vel*)
η		Massenerhaltungsgrad
γ		Gravitationskonstante
g_0		Erdbeschleuningung auf Meereshöhe
M		Erdmasse
m		allgemeine Masse

Tabelle 1.2: Symbole mit fester Bedeutung

Konstante	Wert	Bedeutung
`DIMS`	4	Anzahl der Raumzeit-Dimensionen (3 Raum + 1 Zeit)
`SPACEDIMS`	3	Anzahl der Raumdimensionen
`LON`	0	Index der Länge
`LAT`	1	Index der Breite
`LEV`	2	Index der Höhe / des Druckes
`TIME`	3	Index der Zeit
`SPEEDS`	3	Anzahl der Geschwindigkeitsrichtungen, prinzipiell identisch mit `SPACEDIMS`
`U`	0	Index von u
`V`	1	Index von v
`W`	2	Index von ω
`X`	0	Index von x oder $\dot{x}$
`Y`	1	Index von y oder $\dot{y}$
`Z`	2	Index von z oder $\dot{z}$

Tabelle 1.3: Benannte Konstanten
zur erhöhten Lesbarkeit der Quelltexte

Definition	Bedeutung
typedef pep_t place[DIMS];	Koordinaten in Raum und Zeit, $(\vec{x}, t)$, $(\vec{r}, t)$
typedef size_t dataindex[DIMS];	Indizes im Koordinatengitter
typedef pep_t wind[SPEEDS];	Wind... $\dot{\vec{x}}$, $\dot{\vec{r}}$
typedef pep_t spaceplace[SPACEDIMS];	Koordinaten ohne Zeitm $\vec{x}$, $\vec{r}$

Tabelle 1.4: Vektorielle Datentypen
für die Koordinaten und Winde:

Darüber hinaus vereinfacht die Definition von einigen vektoriellen Datentypen (siehe Tab. 1.4) die Arbeit mit Winden und Orten im dreidimensionalem Raum bzw. in der vierdimensionalen Raumzeit. Zu diesen Datentypen werden auch entsprechende vektorielle Operatoren verwendet. Tab. 1.5 zeigt diese.

In Bezug auf Winkelangaben, meist zu den geographischen Koordinaten λ und β, werde ich manchmal zur Anschaulichkeit von °N oder °O sprechen, aber in der Mathematik fast ausschließlich mit dem Bogenmaß erbeiten. Im Text ist eine Angabe von 90°O gleichbedeutend mit $\lambda = \frac{\pi}{2}$. Ebenso meint $\beta \in \left[-\frac{\pi}{2}; \frac{\pi}{2}\right]$ einfach, dass der Breitengrad zwischen Süd- und Nordpol liegt.

1.2 Verwendete Rechnersysteme

Berechnungen wurden auf einer Reihe von Systemen mit unterschiedlicher Konfiguration durchgeführt. Sowohl Hardware als auch Software haben Einfluss auf erzielte Ergebnisse und auch vor allem auf die benötigte Zeit, um diese Ergebnisse zu erzielen. Ich werde im

Signatur	Bedeutung
`null_dataindex(n)`; `null_place(n)`; `null_space(n)`; `null_wind(n)`	Null-Setzen von Indizes, Orten in Raumzeit und Raum, Winden: $\vec{n} = \vec{0}$
`copy_dataindex(a, b)`; `copy_place(a, b)`; `copy_space(a, b)`; `copy_wind(a, b)`	Kopie von Indizes, Orten in Raumzeit und Raum, Winden: $\vec{b} = \vec{a}$
`add_space(a, b, c)`; `add_flatspace(a, b, c)`; `add_place(a, b, c)`; `add_wind(a, b, c)`	Addition von Orten in Raumzeit, Raum, horizontalem Raum (λ, β oder x, y), Winden: $\vec{c} = \vec{a} + \vec{b}$
`sub_place(a, b, c)`; `sub_space(a, b, c)`; `sub_flatspace(a, b, c)`; `sub_wind(a, b, c)`	Subtraktion von Orten in Raumzeit, Raum, horizontalem Raum (λ, β oder x, y), Winden: $\vec{c} = \vec{a} - \vec{b}$
`sadd_space(a, s, c)`	Addition von Skalar in jeder Komponente: $\vec{c} = \vec{a} + s \cdot \vec{1}$
`ssub_space(a, s, c)`	Subtraktion von Skalar in jeder Komponente: $\vec{c} = \vec{a} - s \cdot \vec{1}$
`smul_space(a, s, c)`	Multiplikation mit Skalar: $\vec{c} = s \cdot \vec{a}$
`smul_flatspace(a, s, c)`	wie oben, nur in den ersten zwei Dimensionen (λ, β oder x, y)
`sdiv_space(a, s, c)`	Division durch Skalar: $\vec{c} = \frac{1}{s}\vec{a}$
`sdiv_flatspace(a, s, c)`	wie oben, nur in den ersten zwei Dimensionen (λ, β oder x, y)

Tabelle 1.5: Vektorielle Operatoren als Funktionen:

Text auf in Tab. 1.6 genannten Systeme durch den Namen verweisen, anstatt jedesmal die Konfiguration aufzulisten. Allen Rechnersystemen ist gemein, dass sie im Wesentlichen mit freier Software betrieben werden.

- **Adams**: Compaq XP1000 AlphaStation mit einem DEC 21264A (EV67) 64 Bit Prozessor, 667MHz und 1GiB PC100 RAM (zwei Kanäle zu 256 Bit, mit 4MiB noch immer größter L2 Cache in dieser Liste). Software: SourceMage GNU/Linux mit Kernel 2.6.17.1 und gcc-4.1.1 .
- **Neuling**: Laptop IBM ThinkPad X31, 32 Bit Pentium-M Prozessor der ersten Generation (Banias, 1,4GHz, 1MiB Cache), Intel 855PM Chipsatz mit 512MiB DDR266 RAM. Software: SourceMage GNU/Linux mit Kernel 2.6.20.7 und gcc-4.1.2
- **Atlas**: Opteron 2210 (zwei Kerne mit 1,8GHz), Broadcom HT1000 Chipsatz, 2GiB DDR2 RAM. Software: openSUSE 10.2 GNU/Linux (x86-64), Kernel 2.6.18.2-34 und gcc-4.1.2 20061115. Hauptfunktion ist die Bereitstelliung von cs. 3,7TiB Festplattenplatz auf 15 SATAII - Festplatten im RAID Verbund per NFS an den Grotrian-Cluster
- **Grotrian-Cluster**: Verbund (Gigabit-Ethernet) aus neun Doppelprozessor Opteron Systemen (AMD Opteron 248, AMD Chipsatz, 2GiB DDR RAM) und 6 Systemen mit jeweils zwei Doppelkernprozessoren (AMD Opteron 275, nVidia Chipsatz, 2GiB DDR RAM); zusammen 42 mit 2,2GHz getaktete Prozessorkerne. Software: SuSE Linux 10.0, Kernel 2.6.13-15, gcc-4.0.2 20050901.
- **Grotrian**: Kontrollknoten des Clusters mit zwei AMD Opteron 248 Prozessoren und 2GiB DDR RAM, ebenfalls SuSE Linux 10.0 .

Tabelle 1.6: Rechnersysteme

2 Datenquelle: ECHO-GiSP GCM

2.1 Modell

Im Rahmen des Pole-Equator-Pole Projektes (siehe [PEP]) ist Meine Arbeit eng verbunden mit der Entwicklung des ECHO-GiSP[1] GCM[2] (Abb. 2.1) durch Sascha Brand am Alfred-Wegener-Institut Potsdam (Artikel zur Veröffentlichung eingereicht, außerdem auf auf der EGU General Assembly präsentiert: [Brand2007]). Zu bestehenden Komponenten (ECHO-G) kommt dabei ein Modul für die Rückkopplung mit stratosphärischer Chemie (diese berechnet von MECCA/iSP) hinzu. Neben generellen Aussagen zum Tracertransport im GCM spielt hier auch die Frage nach dem Einfluss der hinzugefügten Chemie-Rückwirkung auf diesen eine Rolle.

Einige Modellparameter sind:

- 39 Modell-Druckebenen, inklusive Stratosphäre (Abstände logarithmisch)
- 40 variable Tracer (d.h. chemische Konstituenten), zum Beispiel $H_2O, NO_2, OH, HCHO$
- 117 chemische Gleichungen für die Stratosphäre
- Zeitschritte
  - Atmosphäre: 15min
  - Ozean: 2h
  - Kopplung zwischen Atmosphäre und Ozean: 1d

Im Klimamodell finded im Chemiemodul Semi-Lagrangescher Transport statt. Dabei werden die Dichten auf Gitterpunkten einen Zeitschritt weit mit dem Wind integriert und dann die verschobenen Dichten wieder auf das ursprüngliche Gitter zurückinterpoliert. Um die Kontinuität zu erhalten, wird das Resultat mit einem Korrekturfaktor versehen — sonst ginge schon allein durch die Interpolation Masse verloren (oder käme sogar hinzu).

Sascha Brand hat mir Daten aus zwei Modellläufen zur Verfügung gestellt: Einem interaktiven Lauf mit der Rückkopplung der Chemie und einen Referenzlauf ohne diese. Besagte Rückkopplung findet über die Beeinflussung der Energieeinstrahlung (Absorption/Reflektion in verschiedenen Luftschichten abhängig von den chemischen Konstituenten) in den dynamischen Kern statt.

[1] (**ECH**AM & H**O**PE-**G** **i**ncluding **S**tratosphere by AWI Research Unit **P**otsdam)

[2] Global Circulation/Climate Model, ein globales Modell für die Atmosphäre; hier speziell ein AOGCM, d.h. Atmosphäre gekoppelt mit Ozean und Meereis

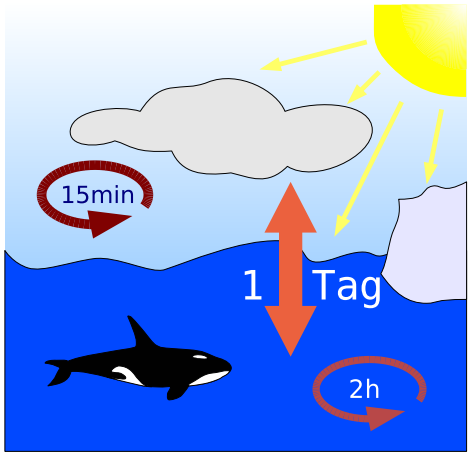


Abbildung 2.1: Stark vereinfachtes Schema des ECHO-GiSP AOGCM
Atmosphärendynamik wird mit einem Zeitschritt von 15 Minuten berechnet, Ozeandynamik mit 2 Stunden Zeitschritt, während die Kopplung zwischen beiden in Schritten von einem Tag erfolgt. Die Extraktion der mir zur Verfügung gestellten Daten finded halbtäglich statt.

2.2 Daten

Mein Thema ist nun nicht das Modell selbst, sondern die Daten die es produziert. Zuerst natürlich die Windfelder, aber es sind für mich auch andere Daten von Interesse. Allen Daten, die mich aus ECHO-GiSP erreichen, ist das Format gemeinsam. Aus dem spektralen Modell werden die Felder in einem horizontalen Gitter aus

- 96 Längengraden mit fester Schrittweise 3.75°O (entspricht 417km am Äquator) sowie
- 48 Breitengraden mit variabler Schrittweite, die sich um $3,71$°N bewegt, auf
- 23 Druckebenen (in Pascal, Pa) interpoliert. Die Extraktion von Feldern erfolgt mit
- $0,5$ Tagen zeitlicher Auflösung.

Die Windfelder sind an jedem Gitterpunkt gegeben als

- Westwindkomponente (in Richtung Osten) u in Metern pro Sekunde (m/s)
- Südwindkomponente (in Richtung Norden) v in Metern pro Sekunde (m/s)
- Vertikalwindkomponente ω in Pascal pro Sekunde (Pa/s).

Der vorläufig finale Datensatz für mich besteht aus jeweils 95 Jahren eines Referenzlaufes und eines Laufes mit interaktiver Chemie. Eine technische Herausvorderung ist die effiziente Arbeit mit der beachtlichen Datenmenge: Eine Zeitreihe des multivariaten Windfeldes über 95 Jahre benötigt 83GiB Speicher.

Der Zugriff auf die Daten wird durch die Klasse `data` abstrahiert (gekürzte Definition in Quelltext 2.1). Hinter dieser Klasse verstecken sich zum einen potentiell viele (hunderte,

tausende) Dateien, die zusammen einen Datensatz ergeben, und zum anderen wird auch die konkrete Struktur des Koordinatengitters und die Liste der enthaltenen Variablen gekapselt. Nirgendwo in meinen Programmen wird die Gitterauflösung in Raum oder Zeit festgeschrie-

Quelltext 2.1: Wesentliche Elemente der allgemeinen Datenschnittstelle

```
1  class data
2  {
3    public:
4    vector<string> varnames;
5    vector<string> dimnames;
6    size_t vars; size_t dims; // convenience shortcuts to .size()
7    pep_t timeunit; //seconds per time unit in file (p.ex. days in
         file: 86400)
8    pep_grid grid;
9    bool ready;
10
11   data();
12   virtual ~data(){ destruct("data"); };
13   // placeholders to be implemented in special interface
14   virtual bool init(){ return false; }; // to-be useful
15   virtual bool init(char** filenames, const int count){ return
         false; }; // to-be useful
16
17   // generic data access via index
18   virtual void get_vector(dataindex index, pep_t *vals) = 0;
19   // coordinate access
20   virtual void get_vector(const place p, pep_t *w);
21
22   // omitted here: utility functions for finding a point in the
         grid, etc.
23 };
```

ben, diese Information kommt aus den selbstbeschreibenden Datensätzen im offenen und plattformübergreifenden NetCDF – Format ([NetCDF]). Ohne ein solches Format wäre der Entwurf eines flexiblen und portablen Programmpaketes zur Datenanalyse sehr fragwürdig.

Durch das gemeinsame Datenformat ist es für mich relativ einfach, meine Programme auf die Arbeit mit zusätzlichen bzw. anderen Daten umzustellen. So habe ich zuerst nur die dreidimensionalen Winddaten benötigt und meine Programme entsprechend gestaltet, konnte jedoch schnell Anpassungen vornehmen, um für eine Mitarbeiterin die ECHO-GiSP Daten von Geopotential und chemischen Konzentrationen zugänglich zu machen (siehe [Chandra2007]). Inzwischen existiert eine Reihe von speziellen Klassen, die über die `data` – Schnittstelle[3] genutzt werden können: `echog_pressure`, welche die den ECHO-GiSP – Daten gemeinsame Gitterdimensionen definiert, `echog_pressure_cached`, welche einen für den quasi-zufälligen

[3] durch die C++ – Eigenschaft der Polymorphie

Zugriff auf Windfelder unverzichtbaren[4] Datenpuffer im Arbeitsspeicher hinzufügt, `egp_wind` als Spezialisierung auf die Windkomponenten u, v und ω, `egp_geopoth` zum Zugriff auf das Geopotential inklusive einer speziellen Methode zur Umrechnung in Höhenmeter und `egp_density`, welche Temperatur und Feuchte Daten verarbeitet und die Dichte berechnet[5].

[4] Ohne diesen ist der Rechner nur mit Suchen auf der Festplatte beschäftigt und der Prozessor hat praktisch nichts zu tun außer zu warten.

[5] Die speziellen Methoden sind natürlich nicht über die allgemeine Schnittstelle ansprechbar, die anderen nicht-speziellen Methoden natürlich schon.

3 Vom Windfeld zum bewegten Ensemble

Der zentrale Baustein der Lagrangeschen Betrachtung des Transportes ist die einzelne Trajektorie, die Bewegung eines einzelnen Partikels. Es geht also um die Berechnung der Trajektorie des Orstvektors

$$\vec{x}(t)$$

aus dem gegebenen instationären Geschwindigkeitsfeld $\dot{\vec{r}}$

$$\dot{\vec{r}} \;=\; f(\vec{r},t);\; \vec{r}(t_0)=\vec{r}_0 \tag{3.1}$$

Dies ist ein klassisches Anfangswertproblem erster Ordnung. In analytischer Betrachtung mit gegebener Funktion $\dot{\vec{r}}(\vec{r},t)$ ist die Lösung prinzipiell klar[1]. Hier zeige ich ein einfaches (unphysikalisches) Beispiel mit $\vec{r}=(x);\; f(\vec{r},t)=(3x+1)t$ in dimensionsloser Formulierung:

$$\begin{aligned} \dot{x}=\frac{\mathrm{d}}{\mathrm{d}t}x &= (3x+1)t \\ \overset{x\geq=0}{\Leftrightarrow} \frac{1}{3x+1}\mathrm{d}x &= t\mathrm{d}t \\ \Leftrightarrow \int_0^x \frac{1}{3x+1}\mathrm{d}x &= \frac{1}{2}t^2 \end{aligned} \tag{3.2}$$

Unter Verwendung der Substitution

$$\begin{aligned} y &= 3x+1 \\ \frac{\mathrm{d}y}{\mathrm{d}x} &= 3 \Rightarrow \mathrm{d}x=\frac{\mathrm{d}y}{3} \end{aligned}$$

lässt sich das Integral wie folgt lösen:

$$\begin{aligned} \int_1^{3x+1} \frac{1}{3}\cdot\frac{1}{y}\mathrm{d}y &= \frac{1}{2}t^2 \\ \Leftrightarrow \frac{1}{3}\ln y|_1^{3x+1}=\frac{1}{3}\ln(3x+1) &= \frac{1}{2}t^2 \\ \Leftrightarrow \ln(3x+1) &= \frac{3}{2}t^2 \\ 3x+1 &= e^{\frac{3}{2}t^2} \\ \Rightarrow x(t) &= \frac{e^{\frac{3}{2}t^2}-1}{3} \end{aligned}$$

[1] ☺ sofern die Differentialgleichung einfach zu integrieren ist...

Dies erfüllt die Differentialgleichung Gl. 3.2:

$$\begin{aligned}
\Rightarrow \frac{\mathrm{d}}{\mathrm{d}t}x &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{e^{\frac{3}{2}t^2}-1}{3}\right) \\
&= 2\frac{3}{2}t\frac{e^{\frac{3}{2}t^2}}{3} = t\cdot 3\frac{e^{\frac{3}{2}t^2}}{3} \\
&= t\cdot\left(3\frac{e^{\frac{3}{2}t^2}-1}{3}+3\frac{1}{3}\right) \\
&= (3x+1)t
\end{aligned}$$

Nun habe ich hier keine einfache Differentialgleichung zu behandeln, sondern ein Klimamodell, dass mir zu diskreten Zeitpunkten Werte der Funktion $f(\vec{x},t)$ aus Gl. 3.2 in Form der Windkomponenten u, v und ω auf diskreten Punkten eines Gitters in λ, β und p vorgibt.

Das Anliegen bleibt: Ich möchte aus gegebenem $\dot{\vec{r}} = f(\vec{r},t)$ die Trajektorie $\vec{r}(t)$ errechnen. Nur führt hier nicht eine analytische Betrachtung, sondern die numerische Integration im gegebenen diskreten Windfeld (Indizes o,a,e für Dimensionen Länge, Breite, Höhe und i für Zeit)

$$\dot{\vec{x}} = \begin{pmatrix} u_{o,i} \\ v_{a,i} \\ \omega_{e,i} \end{pmatrix};\ o,a,e,i \in Z^+ \tag{3.3}$$

zum Ziel. Eine Grundfrage stellt sich in der Ebene, wie man die in m/s gegebenen Geschwindigkeiten mit den Gradnetz-Koordinaten (λ,β) in Einklang bringt. Die vertikale Geschwindigkeit ist mir in Pa/s gegeben, was prinzipiell zu den Druckebenen in Pa passt. Wenn man eine Kugelgeometrie unter (lokaler) Vernachlässigung der Höhenvariationen der Druckebenen ansetzt, so ist diese Druckgeschwindigkeit auch als rechtwinklig zu den streng horizontal definierten Winden anzusehen. Diese Eigenschaft ist für den Einsatz eines Integrationsschemas wie Runge-Kutta in mehreren Dimensionen vorausgesetzt[2]. Es bleibt die Frage der horizontalen Koordinaten mit den dazugehörigen Winden. Man muss sich zwangsläufig entscheiden, zwischen

1. entweder Rechnung im Gradnetz, dazu lokale Umrechnung von m/s in °O/s und °N/s **oder**
2. Rechnung im lokalen metrischen Raum, dazu Umrechnung der °O und °N Koordinaten in m sowie Transformation der Windrichtungen in gewählter Projektion

Die erste Variante ist zweifelsohne die einfachste - man skaliert lediglich die Beträge der Windkomponenten und hat nicht mit weiterer Geometrie zu kämpfen. Ein Nachteil, neben systematischen Fehlern, ist die Situation an den Polen. Direkt am Pol ist in (λ,β)-Koordinaten der Wind überhaupt nicht definiert. Es gibt nur noch die eine entartete Richtung zum Gegenpol (am Nordpol kann man lediglich eine Südrichtung als Synonym für "beliebig" wählen und am Südpol umgekehrt) und Osten oder Westen sind schlicht nicht definiert. Es gibt die Praxis, an den Polen einfach künstlich Werte zu definieren bzw. in den

[2] Ohne Orthogonalität der Dimensionen wären die Koordinatenänderungen der Integrations-Unterschritte nicht unabhängig; eine Bewegung in x Richtung würde auch p ändern.

Längengraden fortzuschreiben, um dann im Gradnetz rechnen zu können. Ich möchte dieses Problem direkt angehen und mit sauber definierten Winden und Richtungen arbeiten. Daher werde ich letztendlich die zweite Methode mit der Rechnung in einem lokalen karthesischen Koordinatensystem (mit den ebenen Koordinaten in Metern) verfolgen, zuvor aber auch die einfachere Rechnung im Gradnetz ansprechen.

Bevor ich näher auf die Transportrechnungen in ihrer Ganzheit eingehe, möchte ich die allgemeinen Algorithmen für Interpolation und Integration vorstellen, die bisher in meiner Arbeit Verwendung finden[3].

3.1 Lineare Interpolation in N kartesischen Dimensionen

Die einfachste Methode, zwischen Stützpunkten eines *kartesischen* Gitters Werte zu interpolieren, ist die lineare Interpolation in einer Gitterzelle[4]. Genauer:

Gesucht ist der linear interpolierte Wert einer Variable y, welche von Koordinaten $x_n;\ n \in [1; N]$ abhängt: $y(\vec{x});\ \vec{x} \in \mathbb{R}^N$. Gegeben sind Werte von y an den Eckpunkten eines den Punkt $\vec{x}$ einschließenden Hyperquaders in $\mathbb{R}^N$, folgend als Gitterzelle bezeichnet. Die Gitterzelle ist definiert durch Intervalle in jeder der N Dimensionen, welche die entsprechende Komponente von $\vec{x}$ eingrenzen: $x_{n,0} \leq x_n \leq x_{n,1}$.

In der Literatur finde ich zum Thema der mehrdimensionalen linearen Interpolation meist eine Darstellung der Variante für zwei Dimensionen mit dem lapidaren Hinweis, dass es in mehr Dimensionen analog sei, oder zumindest mit der Skizzierung der Idee der Rekursion von eindimensionalen linearen Interpolationen. Die Algorithmen für bilineare und trilineare (also 2D und 3D) Interpolation sind durchaus verbreitet. Was ich aber bisher nicht finden konnte, war eine konkrete Formel und einen dazugehörigen Algorithmus für die N-lineare Interpolation. Ich lese oft, dass die Erweiterung auf beliebige Dimensionen offensichtlich sei, trotzdem – oder gerade deswegen? – konnte ich dazu nichts Konkretes finden. Zwar benötige ich maximal die Interpolation in vier Dimensionen, jedoch finde ich es angebracht, diesen Spezialfall aus einer allgemeinen Regel abzuleiten Da ich diese “offensichtliche” Regel nirgends finden kann, muss ich sie vorher herleiten. Die Schwierigkeit liegt dabei weniger im Konzept[5] als in der Formulierung. Diese ist aber möglich und mit den passendem Hilfsmittel auch recht zwanglos, wie ich in den folgenden Seiten zeigen werde.

3.1.1 Eine allgemeine Rekusionsformel

Das Konzept der linearen Interpolation ist sehr einfach. Was die Betrachtung in beliebigen Dimensionen verkompliziert, ist die Indizierung der Ecken der Gitterzelle. Der Koordinatenvektor eines jeden Stützpunktes besteht als Ecke der Gitterzelle aus einer Kombination von unteren und oberen Intervallgrenzen in den einzelnen Dimensionen. Die x_n-Koordinate einer Ecke ist entweder gleich $x_{n,0}$ oder $x_{n,1}$. Eine konkrete Ecke wird also indiziert durch eine Reihe von N Festlegungen auf die Untergrenze (0) oder die Obergrenze (1) des Intervalls in der betreffenden Dimension. Eine nach Dimensionen hierarchisch strukturierte Ordnung der

[3] Es ist ein Ziel des Programmentwurfs, die Algorithmen stets austauschbar zu halten. Verschiedene Ansätze sollen noch hinzugefügt und evaluiert werden können.

[4] im Gegensatz zum allgemeinen Interpolationsproblem, welches eine Funktion für das *gesamte* Gitter sucht

[5] ☺ welches ja nunmal offensichtlich ist...

Stützpunkte ergibt sich auf natürliche Weise, wenn diese Festlegungen auf obere oder untere Intervallgrenze in der Dimension n als n-te Stelle einer Binärzahl interpretiert werden. Um mit dieser Zahl als ganzes sowie auch mit den einzelnen Binärstellen zu operieren, nutze ich den Begriff des Vektors im Binärzahl-Vektorraum $\mathbb{B}^k;\ \mathbb{B} = \{0;1\}$. Jedes Element $\vec{b}$ dieses Vektorraums entspricht der Binärkodierung einer Zahl[6] $b \in \left[0; 2^k - 1\right] \subset \mathbb{Z}$:

$$b = \sum_{n=1}^{k} b_n 2^{n-1} \tag{3.4}$$

Für $k = 3$ ist zum Beispiel $\vec{b} = (0;1;1)$ der Vektor zur Dezimalzahl $b = 6$, entsprechend der Darstellung dieser als Binärzahl[7]: 110.

Diese Zuordnung ist nichts anderes als die Darstellung einer Zahl in dem einen oder anderen Zahlsystem und somit in beide Richtungen eindeutig. Mit der Definition von $\vec{b} \in \mathbb{B}^k$ kann ich alle Eckpunkte der k-dimensionalen Gitterzelle durch einfache Iteration in b und den direkt verbundenen Indexvektor $\vec{b}$ erfassen. Die k-dimensionalen Koordinatenvektoren $\vec{x}^{(k)}$ werden mit der einfachen Zahl b indiziert, was über die Beziehung zu $\vec{b}$ die Indizes für alle Komponenten $x_n^{(k)} = x_{n,b_n}$ (Erinnerung: $x_{n,0}$ und $x_{n,1}$ sind die gegebenen Intervallgrenzen) beinhaltet:

$$\vec{x}_b^{(k)} = (x_{1,b_1}, \ldots, x_{k,b_k}) \tag{3.5}$$

Ich habe hier k von der Dimensionszahl N getrennt, um die in den N-dimensionalen Raum eingebetete k-kimensionale interpolierte Gitterzelle repräsentieren zu können, wie sie als Zwischenstufe der rekursiven Interpolation erscheint. Der komplette Koordinatenvektor $\vec{x}_b$ eines Eckpunktes der k-Dimensionalen Gitterzelle im N-dimensionalen Raum ist beschrieben durch

$$k \in [0;N]:\ \vec{x}_b = (\vec{x}_b^{(k)}, x_{k+1}, \ldots, x_N) = (x_{1,b_1}, \ldots, x_{k,b_k}, x_{k+1}, \ldots, x_N) \tag{3.6}$$

(Komponenten von $\vec{x}_b^{(k)}$ werden in $\vec{x}_b$ eingefügt).

Eine Bemerkung zur Hierarchie der Indizes b soll helfen, den folgenden Rekursionsmechanismus zu verstehen. Zwischen den Koordinaten und Indizes aufeinanderfolgender Rekursionsstufen besteht die Beziehung

$$\begin{aligned} \left(\vec{x}_b^{(k-1)}, x_{k,\mathbf{0}}\right) &= x_b^{(k)} \\ \left(\vec{x}_b^{(k-1)}, x_{k,\mathbf{1}}\right) &= x_{b+2^{k-1}}^{(k)} \end{aligned} \tag{3.7}$$

Das wird klar, wenn ich den Indexvektor betrachte. Für $x_b^{(k-1)}$ ist dies $\vec{b} = (b_1, \ldots, b_{k-1})$, der Indexwert $\sum_{n=1}^{k-1} b_n 2^{n-1} = b$. Wird $\vec{b}$ um eine Dimension erweitert und $b_k = 0$ angefügt, so ergibt sich $\vec{b'} = (b_1, \ldots, b_{k-1}, 0)$ und der Indexwert ändert sich nicht: $b' = b + 0 \cdot 2^{k-1} = b$. Wird dagegen eine 1 angefügt, so ergibt sich für $\vec{b''} = (b_1, \ldots, b_{k-1}, 1)$ der Indexwert $b'' = b + 1 \cdot 2^{k-1}$. Die Erweiterung des Indexvektors bedingt eine Verdopplung der genutzten Indizes von 2^{k-1} auf $2 \cdot 2^{k-1} = 2^k$. Alle Indexvektoren, denen eine 0 angefügt wurde, haben

[6] b ist eine gültige Norm des Vektors $\vec{b}$.

[7] In der Binärzahl steht die erste Stelle (niederwertigstes Bit) *rechts* und im Vektor dagegen *links*, gemäß der üblichen jeweiligen Konvention

einen Indexwert $b' \in \left[0; 2^{k-1}-1\right]$, alle, die um eine 1 erweitert wurden, belegen die obere Hälfte des erweiterten Interalls: $b'' \in \left[2^{k-1}, 2^k-1\right]$.

Mit dieser Indizierung ist nun eine klare allgemeine Formulierung der meist verbal bzw. nur in zwei Dimensionen dargestellten linearen Interpolation in einer Gitterzelle in N Dimensionen möglich:

$$\begin{aligned}
\forall n \in [1;N]: \qquad & d_{n,0} = \frac{x_{n,1}-x_n}{x_{n,1}-x_{n,0}};\ d_{n,1} = 1-d_{n,0} \\
\forall k \in [1;N]: \qquad & \forall b \in \left[0; 2^{k-1}-1\right]: \\
y_b^{(k-1)}(\vec{x}_b^{(k-1)}, x_k, \ldots, x_N) \quad = \quad & d_{k,0} \cdot y_b^{(k)}\left(\vec{x}_b^{(k-1)}, x_{k,\mathbf{0}}, x_{k+1}, \ldots, x_N\right) \\
& + d_{k,1} \cdot y_{b+2^{(k-1)}}^{(k)}\left(\vec{x}_b^{(k-1)}, x_{k,\mathbf{1}}, x_{k+1}, \ldots, x_N\right)
\end{aligned} \tag{3.8}$$

Entsprechend der Lage von x_n zwischen $x_{n,0}$ und $x_{n,1}$ sind die Gewichte $d_{n,0}$ und $d_{n,1}$ festgelegt – genau wie im eindimensionalen Fall. Die Rekursion führt die Interpolation in der Dimension n auf die von N bis zur Dimension $n+1$ davor interpolierten Werte zurück. So werden die auf 2^N Stützstellen gegebenen $y_b^{(N)}$ in N Rekursionsstufen auf das gewünschte $y_0^{(0)}$ interpoliert. Insbesondere sind die erste und letzte Rekursionsstufe gegeben durch

$$\begin{aligned}
y(x_1, \ldots, x_N) \quad &= \quad y_0^{(0)}(x_1, \ldots, x_N) \\
&= \quad d_{1,0} \cdot y_0^{(1)}\left(x_{1,\mathbf{0}}, x_2, \ldots, x_N\right) \\
&\quad + d_{1,1} \cdot y_1^{(1)}\left(x_{1,\mathbf{1}}, x_2, \ldots, x_N\right) \\
\forall b \in \left[0; 2^{N-1}-1\right]: & \\
y_b^{(N-1)}\left(\vec{x}_b^{(N-1)}, x_N\right) \quad &= \quad d_{k,0} \cdot y_b^{(N)}\left(\vec{x}_b^{(N-1)}, x_{N,\mathbf{0}}\right) \\
&\quad + d_{k,1} \cdot y_{b+2^{N-1}}^{(N)}\left(\vec{x}_b^{(N-1)}, x_{N,\mathbf{1}}\right) \\
\forall b \in \left[0; 2^N-1\right]: \ y_b^{(N)} \quad &= \quad y_b^{(N)}\left(\vec{x}_b^{(N)}\right)
\end{aligned} \tag{3.9}$$

Aus Gl. 3.9 und Gl. 3.9 kann sehr direkt eine Funktion in C++ erstellt werden, welche in Quelltext 3.1 zu sehen ist. Entsprechend meiner Anwendung berechnet diese Funktion nicht lediglich einen einzelnen y Wert, sondern einen Vektor $\vec{y}$ von mehreren Variablen. Diese sind unabhängig voneinander behandelt und beeinflussen den Grundalgorithmus nicht.

3.1.2 Umkehr der Rekursion

Ich muss gestehen, dass die rekursive Umsetzung in Quelltext 3.1 zwar didaktisch die erste Variante ist, allerdings von mir zuletzt in Betracht gezogen wurde. Natürlicher erschien mir da die umgekehrte Arbeitsweise: Nicht per Rekursion von oben nach unten, sondern iterativ von unten nach oben. Generell empfiehlt sich die Umformulierung eines rekursiven Algorithmus zu einer Iteration, wobei aber im Einzelfall nicht garantiert ist, dass dies in der Anwendung auf einem Rechner tatsächlich effizienter ist.

Nach kurzer Überlegung ist eine solche Formulierung zur iterativen Berechnung auch zu finden. Die $d_{n,i}$ nach Gl. 3.8 bleiben gegeben, Wenn man sich Gl. 3.9 ansieht, kann man

Quelltext 3.1: N-dimensionale lineare Interpolation durch Rekursion

```
1  void int_linear_recursive( size_t dims, size_t vars,
2                             pep_t *here, pep_t *corners,
3                             pep_t *cornval, pep_t *result,
4                             size_t k = 1, size_t b = 0 )
5  {
6    // work on indices b and b+2^(k-1)
7    size_t b1 = b + ((size_t)1<<(k-1))*vars;
8    xdebug("k=%zu;  b=%zu;  b1=%zu", k, b, b1);
9    // calculate weigths
10   pep_t d0 =
11     (corners[dims+k-1] != corners[k-1])
12     ? ((corners[dims+k-1] - here[k-1]) / (corners[dims+k-1] -
         corners[k-1]))
13     : 1;
14   pep_t d1 = 1 - d0;
15
16   if(k < dims)
17   {
18     pep_t b1result[vars];
19     // recursion to get y_b and y_(b+2^(k-1))
20     int_linear_recursive(dims, vars, here, corners, cornval,
         result, k+1, b);
21     int_linear_recursive(dims, vars, here, corners, cornval,
         b1result, k+1, b1);
22     // ...and the final weighted sum
23     for(size_t v = 0; v < vars; ++v)
24     result[v] = d0*result[v] + d1*b1result[v];
25   }
26   else
27   {
28     // bottom case: k=dims
29     for(size_t v = 0; v < vars; ++v)
30     result[v] = d0*cornval[b+v] + d1*cornval[b1+v];
31   }
32 }
```

anstatt einer rekursiven Vorgehensweise genauso gut die iterative ablesen! Ich durchlaufe eine Schleife über Dimensionen von hohen k her und berechne in jeder Stufe den Satz von $y_b^{(k-1)}$ aus den $y_b^{(k)}$ bzw. $y_{b+2^{(k-1)}}^{(k)}$. Die Formel ist dieselbe, jedoch das Programm in Quelltext 3.2 kommt ohne Rekursion aus; lediglich **for** - Schleifen.

Quelltext 3.2: N-dimensionale lineare Interpolation durch Iteration

```
1  void int_linear_reverse(size_t dims, size_t vars,
2                          pep_t *here, pep_t *corners[2],
3                          pep_t *cornval, pep_t *result  )
4  {
5    // eliminating every dimension
6    for(size_t d = 0; d < dims; ++d)
7    {
8      // weights
9      pep_t s0 =
10       (corners[1][d] != corners[0][d])
11       ? ( (corners[1][d] - here[d])
12           / (corners[1][d] - corners[0][d]) )
13       : 1;
14     pep_t s1 = 1 - s0;
15     // the 2^(dims-d) points left after eliminating d dimensions
16     for(size_t corn=0; corn < ((size_t)1<<dims)>>d; corn+=2)
17     {
18       // we compute the values for p0 and p1 together into the
             place of p0
19       // next iteration uses these values (when in doubt, think of
             memory layout)
20       size_t p0_offset = (corn<<d)*vars;
21       size_t p1_offset = ((corn+1)<<d)*vars;
22       // now interpolating in dimension d (from behind, actually)
             for every variable
23       for(size_t v = 0; v < vars; ++v)
24       cornval[p1_offset+v] = s1*cornval[p1_offset+v]
25                            + s0*cornval[p0_offset+v];
26     }
27   }
28
29   if(result != NULL)
30   memcpy(result, cornval, sizeof(pep_t)*vars);
31 }
```

Praktisch wird bei *festgesetztem* N weder Quelltext 3.1, noch Quelltext 3.2 zum Einsatz kommen. Zur effizienten der für mich konkret benötigten Interpolation in maximal vier Dimensionen leite ich daher die explizite Berechnung entsprechend der einmal komplett durchgeführten Rekursion nach Gl. 3.9 ab. Die Betrachtung der resultierenden Formel inspiriert

eine wesentlich direktere Formulierung der N-dimensionalen Interpolation, welche die kleine Formel- und Algorithmensammlung an dieser Stelle vervollständigen wird.

3.1.3 Explizite Darstellung in vier Dimensionen

Ich habe es bei den ECHO-GiSP Daten mit den vier Dimensionen λ (Länge), β (Breite), p (Druck/Höhe) und t (Zeit) zu tun. In vier Dimensionen ist es durchaus noch machbar, die Rekursion komplett auszuführen – und wenn es länglich erscheint, so ist es zumindest einprägsam. Anstatt der anonymen Numerierung in x_n verwende ich die konkreten Symbole der Dimensionen; gesucht ist $y(t, p, \beta, \lambda)$ in der zwischen $(t_0, p_0, \beta_0, \lambda_0)$ und $(t_1, p_1, \beta_1, \lambda_1)$ aufgespannten Gitterzelle[8].

Die Gewichte sind bestimmt nach Gl. 3.8:

$$\begin{aligned} d_{\lambda,0} &= \frac{\lambda_1 - \lambda}{\lambda_1 - \lambda_0};\ d_{\lambda,1} = 1 - d_{\lambda,0} \\ d_{\beta,0} &= \frac{\beta_1 - \beta}{\beta_1 - \beta_0};\ d_{\beta,1} = 1 - d_{\beta,0} \\ d_{p,0} &= \frac{p_1 - p}{p_1 - p_0};\ d_{p,1} = 1 - d_{p,0} \\ d_{t,0} &= \frac{t_1 - t}{t_1 - t_0};\ d_{t,1} = 1 - d_{t,0} \end{aligned} \tag{3.10}$$

Schritt um Schritt wird nun Dimension um Dimension "weginterpoliert". Die Rolle von $\vec{b}$ bzw b füllen in den verschiedenen Rekursionsstufen die Zahlen i, j, k.

$$\begin{aligned} y_0^{(0)}(t, p, \beta, \lambda) &= d_{t,0} \cdot y_0^{(1)}(t_0, p, \beta, \lambda) \\ &\quad + d_{t,1} \cdot y_1^{(1)}(t_1, p, \beta, \lambda) \\ y_i^{(1)}(t_{i_1}, p, \beta, \lambda) &= d_{p,0} \cdot y_i^{(2)}(t_{i_1}, p_0, \beta, \lambda) \\ &\quad + d_{p,1} \cdot y_{i+2}^{(2)}(t_{i_1}, p_1, \beta, \lambda) \\ y_j^{(2)}(t_{i_1}, p_{j_2}, \beta, \lambda) &= d_{\beta,0} \cdot y_j^{(3)}(t_{i_1}, p_{j_2}, \beta_0, \lambda) \\ &\quad + d_{\beta,1} \cdot y_{j+4}^{(3)}(t_{i_1}, p_{j_2}, \beta_1, \lambda) \\ y_k^{(3)}(t_{i_1}, p_{j_2}, \beta_{k_3}, \lambda) &= d_{\lambda,0} \cdot y_k^{(4)}(t_{i_1}, p_{j_2}, \beta_{k_3}, \lambda_0) \\ &\quad + d_{\lambda,1} \cdot y_{k+8}^{(4)}(t_{i_1}, p_{j_2}, \beta_{k_3}, \lambda_1) \end{aligned} \tag{3.11}$$

[8] also: $x_1 \to t;\ x_2 \to p;\ x_3 \to \beta;\ x_4 \to \lambda$

Nach gegenseitigem Einsetzen und Ausmultiplizieren stehen alle $2^4 = 16$ Terme fest.

$$\begin{aligned} y(t,p,\beta,\lambda) \;=\; & d_{t,0}d_{p,0}d_{\beta,0}d_{\lambda,0}y(t_0,p_0,\beta_0,\lambda_0) \\ & +d_{t,0}d_{p,0}d_{\beta,0}d_{\lambda,1}y(t_0,p_0,\beta_0,\lambda_1) \\ & +d_{t,0}d_{p,0}d_{\beta,1}d_{\lambda,0}y(t_0,p_0,\beta_1,\lambda_0) \\ & +d_{t,0}d_{p,0}d_{\beta,1}d_{\lambda,1}y(t_0,p_0,\beta_1,\lambda_1) \\ & +d_{t,0}d_{p,1}d_{\beta,0}d_{\lambda,0}y(t_0,p_1,\beta_0,\lambda_0) \\ & +d_{t,0}d_{p,1}d_{\beta,0}d_{\lambda,1}y(t_0,p_1,\beta_0,\lambda_1) \\ & +d_{t,0}d_{p,1}d_{\beta,1}d_{\lambda,0}y(t_0,p_1,\beta_1,\lambda_0) \\ & +d_{t,0}d_{p,1}d_{\beta,1}d_{\lambda,1}y(t_0,p_1,\beta_1,\lambda_1) \\ & +d_{t,1}d_{p,0}d_{\beta,0}d_{\lambda,0}y(t_1,p_0,\beta_0,\lambda_0) \\ & +d_{t,1}d_{p,0}d_{\beta,0}d_{\lambda,1}y(t_1,p_0,\beta_0,\lambda_1) \\ & +d_{t,1}d_{p,0}d_{\beta,1}d_{\lambda,0}y(t_1,p_0,\beta_1,\lambda_0) \\ & +d_{t,1}d_{p,0}d_{\beta,1}d_{\lambda,1}y(t_1,p_0,\beta_1,\lambda_1) \\ & +d_{t,1}d_{p,1}d_{\beta,0}d_{\lambda,0}y(t_1,p_1,\beta_0,\lambda_0) \\ & +d_{t,1}d_{p,1}d_{\beta,0}d_{\lambda,1}y(t_1,p_1,\beta_0,\lambda_1) \\ & +d_{t,1}d_{p,1}d_{\beta,1}d_{\lambda,0}y(t_1,p_1,\beta_1,\lambda_0) \\ & +d_{t,1}d_{p,1}d_{\beta,1}d_{\lambda,1}y(t_1,p_1,\beta_1,\lambda_1) \end{aligned} \tag{3.12}$$

Im C++ Programm (Quelltext 3.3) dazu wird die Summe in vier kurzen Schleifen zusammengefasst – es ist dem Compiler überlassen, diese zur maximalen Effizienz auszurollen[9].

3.1.4 Verallgemeinerung auf einfache Summe in N Dimensionen

Die Betrachtung von Gl. 3.12 und vor allem der komprimierten Schleife im Programm dazu führt zu dem Gedanken, dass auch die allgemeine Interpolation in N Dimensionen direkt als einfache Summe darstellbar ist. Dem ist so! Mit dem Werkzeug des Binärvektors kann die lineare Interpolation in N Dimensionen sehr kompakt in einer Summe über alle möglichen Beträge b und somit den direkt zugeordneten $\vec{b} \in \mathbb{B}^N$ – also den Belegungen der Indizes b_n mit 0 oder 1 – ausgedrückt werden:

$$y(\vec{x}) \;=\; \sum_{b=0}^{2^N-1} \left(\prod_{n=1}^{N} d_{n,b_n} \right) \cdot y\left(\vec{x}_{\vec{b}}\right) \tag{3.13}$$

Diese Summe erfasst alle 2^N Stützpunkte der Gitterzelle und gewichtet sie mit dem passenden Produkt der $d_{n,i}$. Theoretisch wird hier das gleiche Ergebnis wie in Gl. 3.9 errechnet – die Formel entspricht der aufgelösten Rekursion, wie für vier Dimensionen gezeigt. Praktisch sind Unterschiede durch beschränkte Rechengenauigkeit und die geringere Anzahl von Rechenoperationen der einfachen Summe gegeben. Durch Induktion läßt sich die theoretische Äquivalenz zu Gl. 3.9 allgemein beweisen:

Ich gehe davon aus, dass die Interpolation von $y(x'_1, \ldots, x'_N)$ durch die einfache Summe gegeben ist (wie für $N = 4$ gezeigt). Jede Interpolation über N' Dimensionen kann betrachtet

[9] ...was hier wohl angemessen ist, jedoch kann exzessives Auflösen von Schleifendurch Vergrößerung des Programms auch negative Auswirkung haben.

Quelltext 3.3: konkretes Programm zur linearen Interpolation in vier Dimensionen

```
1  pep_t interpolate_4d(pep_t grid[DIMS][GRIDSIZE],
2    size_t index[2][DIMS], pep_t position[DIMS], pep_t val
       [2][2][2][2])
3  {
4    pep_t m = 0;    // the value at point of interest
5    pep_t d[DIMS][2]; // d and (1-d) for 4 coordiantes
6
7    for(int i = 0; i < DIMS; ++i)
8    {
9      d[i][0] = (grid[i][ index[1][i] ] - position[i])
10               / (grid[i][ index[1][i] ] - grid[i][ index[0][i] ]);
11     d[i][1] = 1-d[i][0];
12   }
13
14   for(int t = 0; t < 2; ++t)
15   for(int p = 0; p < 2; ++p)
16   for(int beta = 0; beta < 2; ++beta)
17   for(int lambda = 0; lambda < 2; ++lambda)
18   m += d[0][t] * d[1][p] * d[2][beta] * d[3][lambda]
19        * val[t][p][beta][lambda]
20
21   return m;
22 }
```

werden als unvollständige Interpolation im $(N = N' + 1)$ - dimensionalen Raum. Die Anwendung vom finalen Schritt aus Gl. 3.9 wird zeigen, dass das Resultat wiederum als einfache Summe repräsentiert werden kann. Um Gl. 3.9 ansetzen zu können, füge ich zur Einbettung in den N - dimensionalen Raum die zusätzliche Intervallgrenze $x_{1,i}$ als neue erste Koordinate ein und verschiebe die Indizes der folgenden $x_n = x'_{n-1}$, so dass nun die zwei interpolierten Werte für die Ober- und Untergrenze des Intervalls $[x_{1,0}; x_{1,1}]$ der ersten Dimension erfasst werden:

$$i \in \{0; 1\} : \; y_i(x'_1, \ldots, x'_N) = y(x_{1,i}, x'_1, \ldots, x'_N) = y(x_{1,i}, x_2, \ldots, x_N)$$

Die direkte Formel für y muss leicht abgeändert werden, da eine neue Koordinate und somit ein neuer Index vorhanden ist. Dieser erste Index ist aber fest auf 0 oder 1 gesetzt, während die bisherigen Indizes um eine Stelle verschoben sind. Es wird immernoch über $b \in \left[0; 2^{N'} - 1\right]$ Elemente summiert, nur wird der tatsächliche Arbeitsindex modifiziert: $b(i) = 2b' + i$. Die Multiplikation mit 2 verschiebt die Indizes (Bits) um eine Stelle, die Addition von i legt den neu geschaffenen ersten[10] Index fest auf $i = 0$ oder $i = 1$.

$$b(i) = 2b' + i : \; y(x_{1,i}, x_2, \ldots, x_N) \quad = \quad \sum_{b'=0}^{2^{N-1}-1} \left(\prod_{n=2}^{N} d_{n,(b(i))_n} \right) \cdot y(\vec{x}_{\vec{b}(i)}) \tag{3.14}$$

Nun kann ich diese Form von Gl. 3.13 mit der finalen Stufe der Rekursion aus Gl. 3.9 vereinen:

$$\begin{aligned} y(x_1, \ldots, x_N) \quad = \quad & d_{1,0} \cdot \sum_{b'=0}^{2^{N-1}-1} \left(\prod_{n=2}^{N} d_{n,(b(0))_n} \right) \cdot y(\vec{x}_{\vec{b}(0)}) \qquad (3.15) \\ & + d_{1,1} \cdot \sum_{b'=0}^{2^{N-1}-1} \left(\prod_{n=2}^{N} d_{n,(b(1))_n} \right) \cdot y(\vec{x}_{\vec{b}(1)}) \\ \overset{\substack{(b(0))_1=0 \\ (b(1))_1=1}}{=} \quad & \sum_{b'=0}^{2^{N-1}-1} \left(\prod_{n=1}^{N} d_{n,(b(0))_n} \right) \cdot y(\vec{x}_{\vec{b}(0)}) \qquad (3.16) \\ & + \sum_{b'=0}^{2^{N-1}-1} \left(\prod_{n=1}^{N} d_{n,(b(1))_n} \right) \cdot y(\vec{x}_{\vec{b}(1)}) \end{aligned}$$

Die globalen Faktoren $d_{1,0}$ und $d_{1,1}$ gehen in das Produkt innerhalb der Summe ein, wodurch dieses mit $(b(i))_1 = i$ um den ersten Index erweitert werden kann.

Was noch zur Identifikation mit Gl. 3.13 fehlt, ist die Vereinigung der beiden Summen. Der Blick auf die Binärindizes zeigt, dass dies faktisch schon gegeben ist: Zusammen decken beide Teilsummen durch $b' = 2^{N-1} - 1 \Rightarrow b(1) = 2(2^{N-1} - 1) + 1 = 2^N - 1$ alle $b(i) \in \left[0; 2^N - 1\right]$ ab. Die erste Summe beinhaltet dabei alle geraden $b(i)$ (da die niederwertigste Binärstelle

[10] das erste bzw. das *nullte* Bit - entsprechend der Wertigkeit $2^0 = 1$

gleich Null ist) und die zweite alle ungeraden. Der abgeleitete Index $b(i)$ kann somit einem direkten b in der vereinigten Summe über $b \in \left[0; 2^N - 1\right]$ weichen, womit gezeigt ist, dass

$$\begin{aligned} i \in \{0;1\} : \; y(x_{1,i}, x_2, \ldots, x_N) &= \sum_{b'=0}^{2^{N-1}-1} \left(\prod_{n=2}^{N} d_{n,(b(i))_n} \right) \cdot y(\vec{x}_{\vec{b}(i)}) \\ \Rightarrow y(x_1, x_2, \ldots, x_N) &= \sum_{b=0}^{2^N-1} \left(\prod_{n=1}^{N} d_{n,b_n} \right) \cdot y\left(\vec{x}_{\vec{b}}\right) \end{aligned} \tag{3.17}$$

Also: Aus der Gültigkeit von Gl. 3.13 für $N-1$ Dimensionen folgt mit Gl. 3.9 als Spezialfall von Gl. 3.9 die Gültigkeit von Gl. 3.13 für N Dimensionen. Ich habe die direkte Formel ja aus der aufgelösten Rekursion für vier Dimensionen abgeleitet, somit ist bewiesen, dass sie auch für $N > 4$ gültig ist. Prinzipiell fehlt hier der Bereich $N < 4$ – aber schon ein kurzer Blick auf $N = 1$ zeigt die Äquivalenz zur Rekursion in diesem Fall.

Die soeben gesicherte Formel mit der expliziten Verwendung der Komponenten von $\vec{b}$ kann mit Hilfe der Bitoperatoren von C++ in Quelltext 3.4 umgesetzt werden. Die Kombination von Bitshift nach links und rechts[11] sowie bitweisem "und" ermöglicht den Zugriff auf einzelne Bits der Binärrepräsentation einer Ganzzahl. ((`corn` & (1<<`d`))>>`d`) ergibt 1, wenn Bit d (von rechts) gesetzt ist und 0, wenn nicht.

3.1.5 Effizienzvergleich

Man mag die Frage stellen, wozu drei verschiedene Algorithmen für ein und dieselbe (doch eigentlich simple) Rechnung notwendig seien. Eine Antwort liefert eine Untersuchung der benötigten Laufzeit für eine Interpolation auf meinem in Abhängigkeit von Dimensionalität und Variablenanzahl bis jeweils 20. Abb. 3.1 zeigt ein durchaus gemischtes Bild: Jede Variante hat ihren Bereich, in dem die die Aufgabe am schnellsten bearbeitet. So ist Quelltext 3.2 auf Neuling im Bereich um 10 bis 16 Dimensionen und niedrigeren Variablenanzahlen dominant, während bei höheren Werten beider Parameter diese Variante den beiden anderen deutlich unterlegen ist. Diese Relationen hängen allerdings empfindlich vom Rechnersystem ab – besonders die Speicherarchitektur entscheidet, ob des um Rechenzeit des Prozessors oder um Wartezeit auf den Speicher geht. Daher ist die wesentliche Aussage der Messungen, dass die Frage nach dem generell effizientesten Algorithmus nicht klar beantwortet ist. Am Rand des untersuchten Parameterbereiches scheint die einfache Summe zu hohen Dimensionen und Variablenanzahlen hin zu gewinnen, jedoch kann such das mit weiter skalierten Parametern oder Wechsel des Rechnersystems ändern.

Für alle hier gezeigten Methoden gilt: Die Komplexität der linearen Interpolation unter Verwendung aller Ecken der N - dimensionalen Gitterzelle ist bestimmt durch 2^N, der Anzahl der Eckpunkte. Der Rechenaufwand steigt exponentiell mit der Zahl der Dimensionen, daran kann kein Algorithmus etwas ändern, der alle Ecken einbezieht. Eine Möglichkeit der linearen Interpolation mit deutlich verringerter Komplexität liegt in der Auswahl einer Untermenge von Eckpunkten der Gitterzelle – wie in [Rovatti1998] dargestellt. Dort wird der den Interpolationspunkt einschließende Simplex aus $N + 1$ Eckpunkten bestimmt und dann nur mit diesen die Interpolation durchgeführt. Gegenüber den 2^N Eckpunkten der kompletten

[11] links: in Richtung höherwertiger bits; rechts: in Richtung niederwertiger Bits

Quelltext 3.4: N-dimensionale lineare Interpolation mit einfacher Summe

```
1  void int_linear_direct(size_t dims, size_t vars, pep_t *here,
       pep_t *corners[2], pep_t *cornval, pep_t *result)
2  {
3    pep_t d[dims][2];   // weight factors for dims directions
4    for(size_t dim = 0; dim < dims; ++dim)
5    {
6      d[dim][0] =
7        (corners[1][dim] != corners[0][dim])
8        ? ( (corners[1][dim] - here[dim])
9            / (corners[1][dim] - corners[0][dim]) )
10       : 1;
11     d[dim][1] = 1 - s[dim][0];
12   }
13
14   for(size_t i=0; i != vars; ++i) result[i] = 0; // init
15
16   // sum up the parts, we have 2^dims corners
17   for(size_t corn=0; corn < ((size_t)1)<<dims; ++corn)
18   {
19     // compute weight of corner point
20     pep_t weight = 1;
21     for(size_t dim = 0; dim < dims; ++dim)
22     weight *= d[dim][(corn & (((size_t)1)<<dim))>>dim];
23
24     // now add the weighted values
25     for(size_t v=0; v < vars; ++v)
26     result[v] += cornval[corn*vars+v]*weight;
27   }
28 }
```

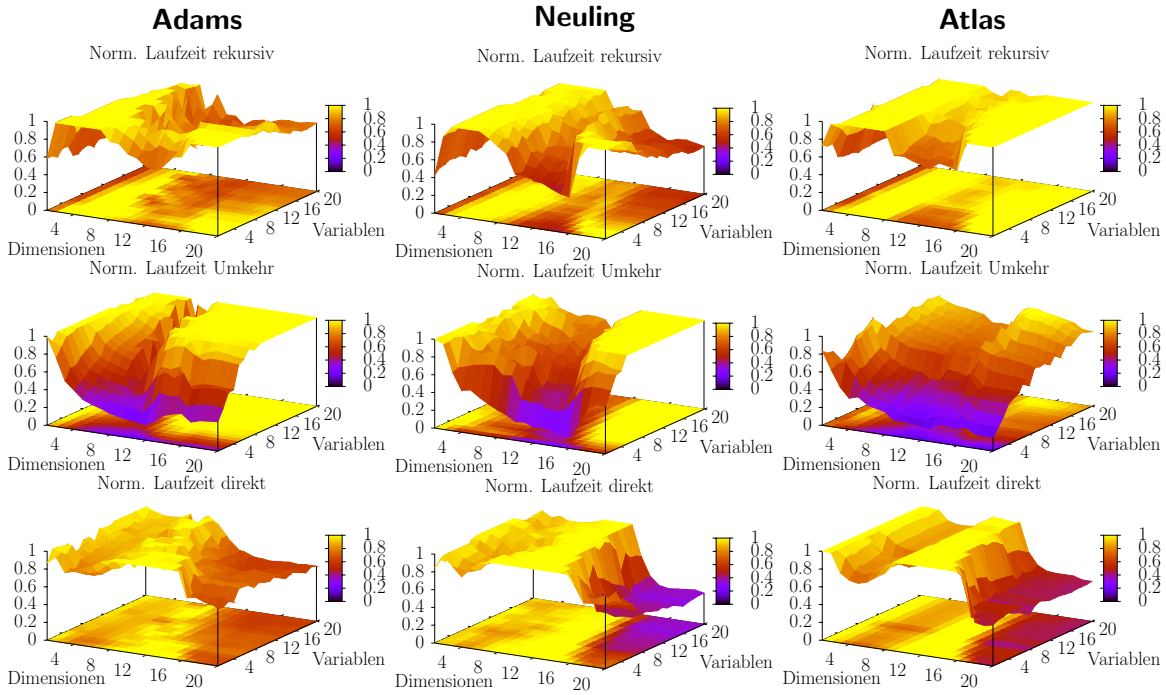


Abbildung 3.1: Verhalten der relativen Laufzeit der linearen Interpolation von bis zu 20 Variablen in bis zu 20 Dimensionen mit den drei verschiedenen Ansätzen auf drei verschiedenen Rechnern. Der dargestellte Wert ist die auf das Maximum des Rechners für die jeweiligen Parameter unter allen drei Läufen normierte Laufzeit. Ein Wert von 1 bedeutet, dass die Variante die längste Zeit brauchte, ein Wert unter 1 zeigt, wieviel schneller die betreffende Variante im Vergleich zur langsamsten war. Auch wenn die Algorithmen auf jedem Rechner eine gewisse Charakteristik bezüglich der Parameter bewahren, ist z.B. deutlich, dass Atlas den speicherintensiven Umkehralgorithmus durchweg besser verarbeitet als die Rekursion, während die beiden anderen Rechner (mit langsameren Speicher) für hohe Dimensionen und Variablenzahlen die Rekursion effizienter verarbeiten. Ein wesentlicher Faktor ist immer die unterschiedliche Balance zwischen Speicherzugriff und -transfer und der eigentlichen Prozessorleistung.

Gitterzelle ist das in der Tat eine deutliche Einsparung, aber eben genau weil weniger Daten in die Interpolation einbezogen werden.

Letztendlich kommt für meine Anwendung hier mit linearer Interpolation in zwei oder vier Dimensionen eine festgeschriebene Variante wie Quelltext 3.3 zum Einsatz. Es schadet aber sicherlich nicht, sich des Hintergrundes bewußt zu sein – sowie die N - dimensionale lineare Interpolation einmal gründlicher als meist üblich behandelt zu haben.

3.2 Interpolation über abstandsgewichtete Summe / Shepard-Interpolation

Die lineare Interpolation bildet eine Summe über die Werte auf umgebenden Gitterpunkten, gewichtet über das Produkt der Abstände in den einzelnen Dimensionen. Das funktioniert in einem kartesischen Gitter mit recktwinkligen Zellen, nicht aber wenn die Stützpunkte ungeordnet sind oder auch nur vom rechtwinkligen Gitter abweichen. Betrachtet man die (λ, β) Gitterpunkte des Gradnetzes nicht im abstrakten Koordinatenraum, sondern direkt auf der Erdoberfläche, so sind die Gitterzellen nicht mehr rechtwinklig.

Ein einfaches Verfahren zur Interpolation in beliebigen Gittern stellte Shepard vor knapp 40 Jahren in [Shepard1968] vor. Er formulierte es für zwei Dimensionen, jedoch ist die Zahl der Dimensionen für den Grundgedanken nicht wichtig, welcher auch als "abstandsgewichtete Summe" (distance weighted sum) der umliegenden bzw. aller Gitterpunkte bekannt ist. Ich ziehe hier nur eine lokale Shepard-Interpolation in Betracht: Ich beschränke die Summe von vornherein auf die umliegenden Punkte der Gitterzelle. Diese Zelle ist zwar nicht mehr kartesisch, doch aber in (λ, β) regelmäßig, so dass die einzubeziehenden Punkte schnell gefunden sind. Mit der Abstandsfunktion $d(\vec{x}, \vec{x}_i)$ ist die Formel nach Shepard gegeben durch

$$y(\vec{x}) = \frac{\sum_{i=1}^{n} \frac{1}{d(\vec{x},\vec{x}_i)^k} y(\vec{x}_i)}{\sum_{i=1}^{n} \frac{1}{d(\vec{x},\vec{x}_i)^k}} \tag{3.18}$$

Die Wahl des Exponenten k steht frei, aber schon in [Shepard1968] ist $k = 2$ als optimaler, zumal auch im kartesischen Raum[12] mit minimalem Aufwand versehender Wert empfohlen: So ist einfach $d(\vec{x}, \vec{x}_i)^2 = \left(\sqrt{(\vec{x} - \vec{x}_i)^2}\right)^2 = (\vec{x} - \vec{x}_i)^2$ mit der euklidischen Metrik. Eine aus das Wesentliche reduzierte Fassung der verwendeten Shepard-Interpolation (in der Klasse `world`) is in Quelltext 3.5 gezeigt.

3.3 Runge-Kutta-Integration vierter Stufe (RK4)

Das etablierte Runke-Kutta Verfahren in vierter Ordnung findet man schematisch in einschlägiger Literatur, üblicherweise für die einzelne Koordinate $x(t)$. Das Schema bleibt auch

[12] das lokal kartesische Koordinatensystem, in dem die im globalen System als kartesische gesehene Gitterzelle nicht mehr kartesisch ist

Quelltext 3.5: Prototyp der angewandten Shepard-Interpolation

```
1  size_t n;              // input: number of grid cell points
2  coordinates *system; // input: local system
3  place np[n];           // input: grid cell places
4  wind  nw[n];           // input: grid cell winds
5  wind w;                // output: resulting local wind
6
7  wind lw[n];  // local grid cell winds
8  pep_t weightsum = 0;
9  size_t i;
10
11 for(i = 0; i < n; ++i) // weights, sum of
12 {
13   // global -> local coordinates
14   system->lwind(np[i], nw[i] ,lw[i]);
15   system->lplace(np[i],workplace);
16   // lmetric_2d = 2D (squared) distance from system base
17   if( (weight[i] = system->lmetric_2d(workplace)) != 0 )
18   weightsum += (weight[i] = 1/weight[i]);
19   else break; // hit a grid point
20 }
21
22 if(i < n) // unlikely exact hit of a point
23 copy_wind(lw[i],w);
24 else // sum winds with normalized weights
25 {
26   for(i = 0; i < n; ++i) weight[i] /= weightsum;
27
28   for(size_t speed = 0; speed < SPEEDS; ++speed)
29   {
30     w[speed] = 0;
31     for(i = 0; i < n; ++i)
32     w[speed] += weight[i] * lw[i][speed];
33   }
34 }
```

für höhere (kartesische!) Dimensionen prinzipiell unverändert und kann hier direkt angewandt werden, indem man es auf Vektoren verallgemeinert. Aus $\vec{x}$ zum Zeitpunkt t_0 wird dann $\vec{x}$ zum Zeitpunkt $t_0 + \Delta t$ berechnet:

$$\begin{aligned}
\vec{x}_0 &= \vec{x}(t_0) \\
\vec{k}_1 &= \Delta t \cdot \dot{\vec{x}}(\vec{x}_0, t_0) \\
\vec{k}_2 &= \Delta t \cdot \dot{\vec{x}}(\vec{x}_0 + \frac{1}{2}\vec{k}_1, t_0 + \frac{1}{2}\Delta t) \\
\vec{k}_3 &= \Delta t \cdot \dot{\vec{x}}(\vec{x}_0 + \frac{1}{2}\vec{k}_2, t_0 + \frac{1}{2}\Delta t) \\
\vec{k}_4 &= \Delta t \cdot \dot{\vec{x}}(\vec{x}_0 + \vec{k}_3, t_0 + \Delta t) \\
\Rightarrow \vec{x}(t_0 + \Delta t) &= \vec{x}_0 + \frac{1}{6}\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right)
\end{aligned} \tag{3.19}$$

Für weitere Details zu diesem Standardverfahrens verweise ich auf [Bronstein2000], wo die wesentlichen Sachverhalte zusammengefasst sind.

3.4 3D Integration im Gradnetz

Die erste Rechenmethode arbeitet komplett im geographischen Gradnetz und blendet im Wesentlichen die Krümmung der Dimensionen aus. Werte für $\dot{\vec{x}}$ zwischen den Gitterpunkten werden durch lineare Interpolation in allen vier Dimensionen des Gradnetzes errechnet. Die aus ECHO-GiSP extrahierten horizontalen Winde sind aber in m/s angegeben. Um die Integration im Gradnetz durchzuführen, müssen v und u von m/s lokal in °/s umgerechnet werden:

$$u'(\lambda, \beta, p) = u(\lambda, \beta, p) \cdot \frac{180°}{\pi R \cdot cos(\beta)} \tag{3.20}$$

$$v'(\lambda, \beta, p) = v(\lambda, \beta, p) \cdot \frac{180°}{\pi R} \tag{3.21}$$

Zusammen mit dem unveränderten vertikalen Wind ω in der lokal angepassten Geschwindigkeit ergibt sich

$$\dot{\vec{x}}'(\vec{r}) = \dot{\vec{x}}'(\lambda, \beta, p) = \begin{pmatrix} u' \\ v' \\ \omega \end{pmatrix} \tag{3.22}$$

In 3.19 wird nun anstatt $\dot{\vec{r}}$ das angepasste $\dot{\vec{x}}'$ verwendet.

Das ist dann auch schon die ganze Rechnung! Wahrlich einfach, jedoch durchaus problematisch.

Mein erster Prototyp[13] nutzt dieses Verfahren, beschränkt auf horizontale Bewegung entsprechend den ersten Testdaten. Unabhängig von dieser Einschränkung zeigt sich ein fundamentales Problem in unmittelbarer Polnähe: Ohne Spezialbehandlung werden Partikel am Pol

[13] in Perl unter Nutzung von PDL zum Einlesen der NetCDF Daten und vektorieller Rechnung, lediglich 370 Programmzeilen

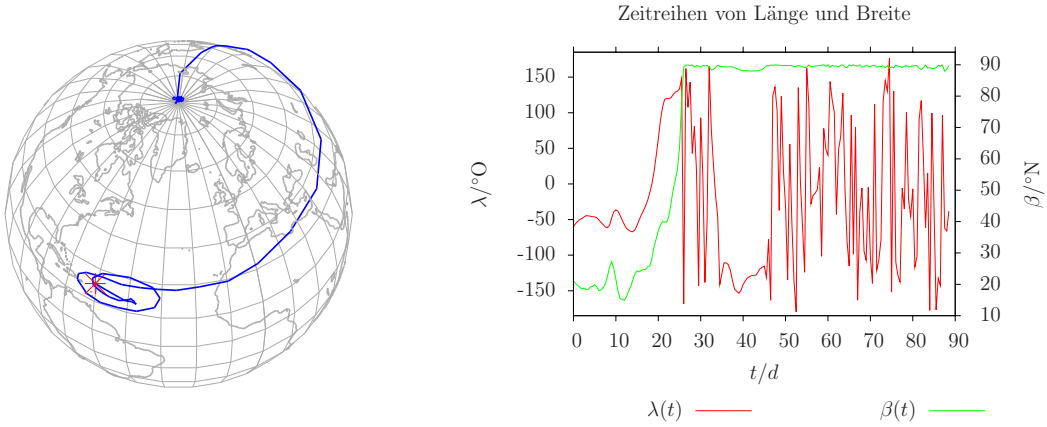


Abbildung 3.2: Die Polfalle bei Gradnetz-Integration
Ein frühes Ergebnis mit Gradnetz-Integration ohne vertikalen Transport (Bewegung auf einer Druckebene): Das Partikel bleibt am Nordpol gefangen, anstatt auf die andere Seite zu gelangen.

eingefangen, siehe Abb. 3.2. Aus Abb. 3.3 und Abb. 3.4 kann abgelesen werden, dass gerade am Nordpol die Stagation von Partikeln auf einer Stelle angesichts der Windverhältnisse höchst unrealistisch ist.

3.5 Die Erde ist rund

In der dem Prototyp folgenden Umsetzung wurde die bilineare Interpolation in λ, β duch die Transformation in ein lokales Koordinatensystem mit flexiblerer abstandsgewichteter Summe ersetzt. Dieser Ansatz des lokalen Koordinatensystems folgt als eine Behandlung der Tatsache, dass die Erde rund ist, während man ein Integrationsverfahren anwendet, welches für kartesischen Raum konzipiert ist.

Ein mehrdimensionales Runke-Kutta Verfahren (wie auch andere übliche Verfahren) geht davon aus, dass die verschiedenen Dimensionen kartesisch sind, die Koordinatenachsen also rechtwinklich zueinander stehen. Tatsächlich ist das im Länge-Breite-Höhe[14] System *lokal* der Fall, aber schon die Eckpunkte von Gitterzellen haben abweichendes Norden und Osten bzw. die einzelnen Integrationsschritte sind nicht eindeutig, wenn man entsprechend den in m/s gegebenen Geschwindigkeiten sich in m auf der Oberfläche bewegt.

Wenn ich abweichendes Norden und Osten sage, dann sollte ich das wohl besser erklären. Natürlich sind die Himmelsrichtungen klar definiert[15], jedoch ist in der gekrümmten Geometrie die Frage der Beziehung von Richtungen an verschiedenen Orten nicht so einfach. Befinde ich mich auf der Erdoberfläche, so erscheint sie erst einmal flach. Für nicht zu große Entfernungen wähne ich mich in einem kartesischen Koordinatensystem mit *einem* Norden und rechtwinklig dazu *einem* Osten. Der Eindruck trügt natürlich – wir wissen, dass die Erde rund ist und das Koordinatensystem, dass die Längen- und Breitengrade aufbauen,

[14] mit Länge-Breite-Druck ist es dank des nichttrivialen Druckgradienten noch interessanter
[15] Das stimmt bis auf die Kleinigkeit, dass an den Polen keine Richtung außer der zum Gegenpol definiert ist, welche wiederum in der Ebene völlig entartet ist.

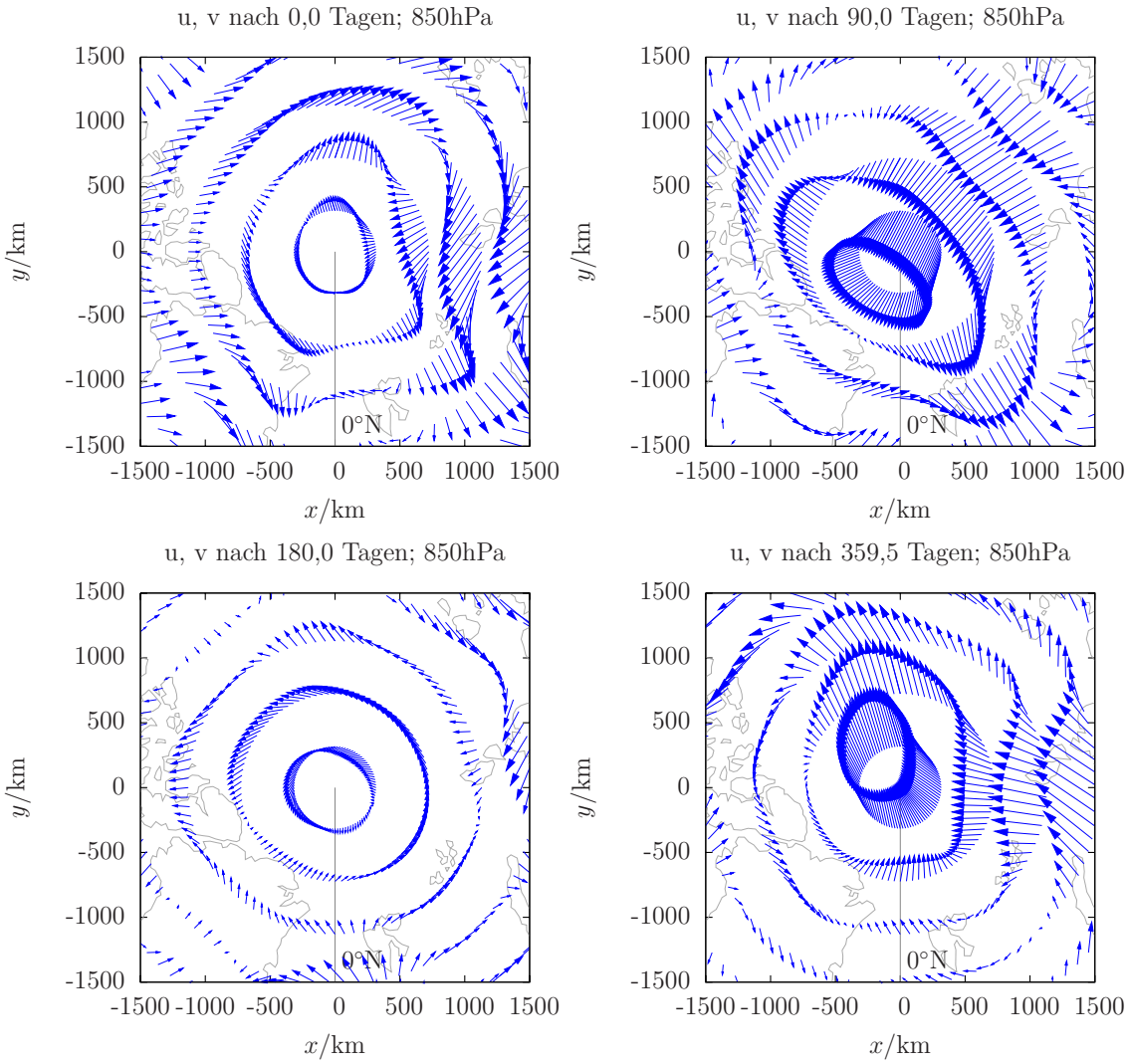


Abbildung 3.3: Nordpolarwind in 850hPa (orthographisch) von interaktivem Lauf
Gerade über den Pol weht heftiger Wind, nicht nur hinein – hier eine Darstellung aus dem interaktivem Lauf des Jahres 1965. In der Mitte des Bildes entsprechen 30 Kilometer Länge des gezeichneten Windvektors 1m/s Windgeschwindigkeit.

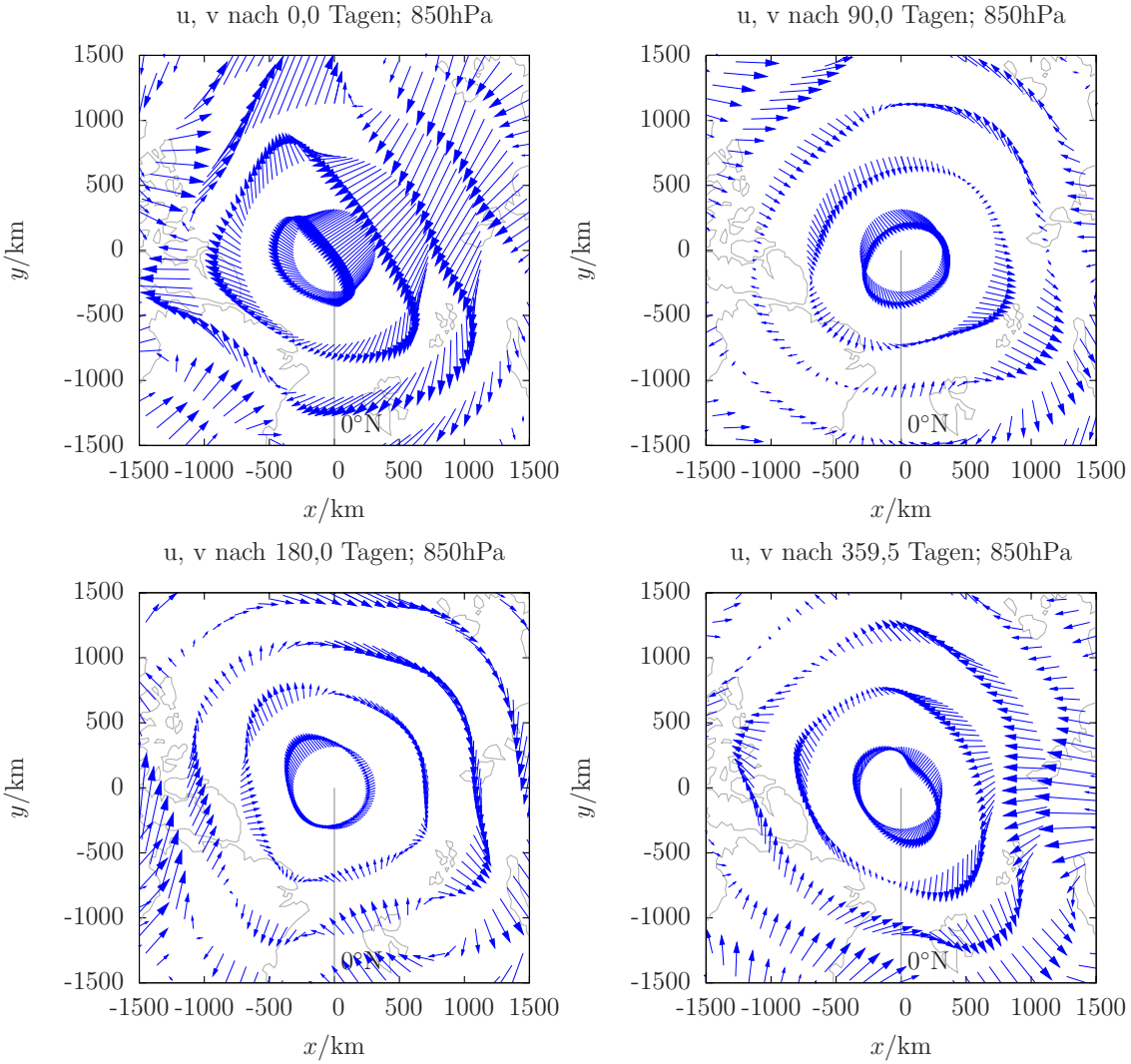


Abbildung 3.4: Nordpolarwind in 850hPa (orthographisch) von Referenz-Lauf
Gerade über den Pol weht heftiger Wind, nicht nur hinein – hier eine Darstellung aus dem Referenz-Lauf des Jahres 1965. In der Mitte des Bildes entsprechen 30 Kilometer Länge des gezeichneten Windvektors 1m/s Windgeschwindigkeit.

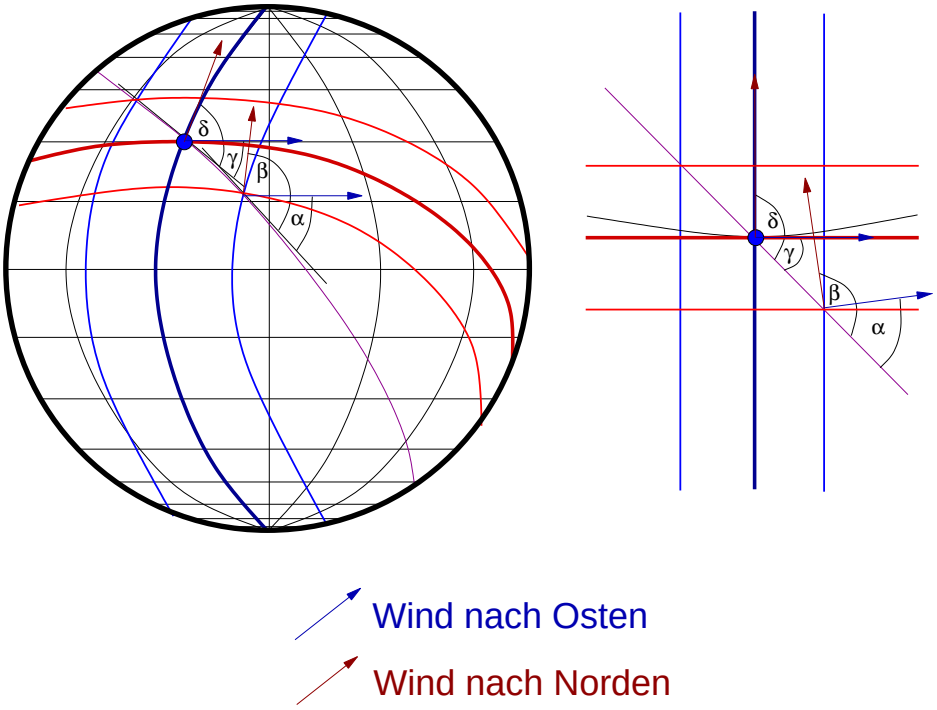


Abbildung 3.5: lokales quasi-kartesisches Koordinatensystem in globaler Perspektive
Ein Koordinatensystem mit Großkreisen als Hauptachsen und den Koordinaten auf dazu parallelen Kleinkreisen, welches der lokalen Sicht einer fortgeschriebenen Nord- und Ostrichtung entspricht, zeigt einen Begriff der veränderlichen Bedeutung der Richtungen der Geokoordinaten. Links die Ansicht auf der Erdkugel, rechts eine Darstellung im individuellen kartesischen System.

nicht kartesisch. Jedoch hat diese lokale Sicht und die damit verbundene Trennung der Himmelsrichtungen ihren Sinn.

Ein (Gedanken)Experiment diene als Beispiel: Ich habe zwei hinreichend voneinander entfernte Punkte auf der Erdoberfläche, die nicht beide auf dem Äquator oder auf einem gemeinsamen Längengrad liegen, einen Startpunkt und einen Zielpunkt. Am Startpunkt richte ich einen Kompass[16] zum Zielpunkt hin aus, notiere mir die Position der nach Norden zeigenden Nadel und bewege mich dann zum Zielpunkt. Dort angekommen, richte ich den Kompass zum Startpunkt hin aus, drehe ihn um 180 Grad, so dass er nun nach meinem Begriff[17] dieselbe Orientierung aufweist, und notiere wiederum die Position der nach Norden zeigenden Nadel.

Die Position der Nadel am Zielpunkt unterscheidet sich von der am Startpunkt notierten! Weshalb? Nun, ich habe mich in meiner “flachen” Welt bewegt, ohne Rücksicht auf das Gradnetz. “Geradeaus” bedeutet, dass ich meinen Weg zum Zielpunkt auf einem Großkreis beschritten habe – und dass ich auch entlang dieses Großkreises den Kompass ausrichte. Hätte ich einen Kurs (z.B. streng nach Osten) festgelegt und wäre diesem mit Hilfe des

[16] gemeint ist ein idealer Kompass, der tatsächlich immer zum geographischen Nordpol zeigt, bzw. am jeweiligen Standort korrekt auf die Deklination eingestellt wird
[17] als Ameise auf der Erdoberfläche, die nicht weiss, was Norden eigentlich auf der Kugel bedeuten soll

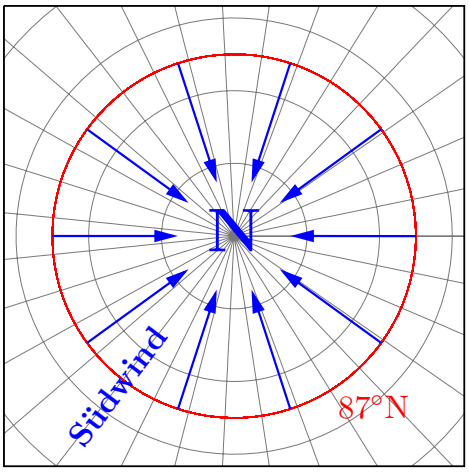


Abbildung 3.6: Norden ist Norden?
Alle fiktiven Südwinde wehen nach Norden in dieser orthographischen Sicht auf den Nordpol... und sie zeigen klar in unterschiedliche Richtungen!

Kompasses gefolgt, wäre ich nicht wirklich geradeaus gegangen, sonder hätte den Pol umkreist[18].

Wenn man sich Abb. 3.5 anschaut, sollte der Unterschied in der Nordrichtung auffallen. Am Zielpunkt zeigt die Richtung, welche am Startpunkt als Norden festgelegt wurde, nicht mehr nach Norden – und ebenso die Richtung Osten (welche hier wie dort orthogonal zum lokalen Norden ist).

Es geht auch weniger subtil, wie Abb. 3.6 zeigt: Schaut man aus dem Weltall auf die nahe Umgebung des Nordpols, so ist mehr als offensichtlich, dass ein Wind nach Norden aus sehr unterschiedlicher Richtung wehen kann.

Ebenso ist es einleuchtend, dass, wenn eine beliebige Anzahl Menschen an beliebigen Orten in Richtung Norden starten, sich schließlich alle am Nordpol treffen. In ebener Geometrie würden sich die Wege von unterschiedlichen Startpunkten, wenn die gewählten Richtungen parallel zueinander sind, niemals kreuzen. Andersherum können zwei Menschen auf dem selben Breitengrad jeweils in Richtung des anderen nach Osten und Westen starten. Würden beide ihre ursprüngliche Richtung beibehalten ohne anhand eines Kompanten fortwährend zu “korrigeren” und sich so auf Großkreisen bewegen, könnten sie sich mit viel Glück an den zwei Kreuzungspunkten ihrer Großkreise treffen, wenn sie ihre Geschwindigkeiten gut aufeinander absestimmt haben, auf keinen Fall ist ein (Wieder)Sehen garantiert.

3.6 Die lokale Transformation

Die Interpolation und Integration im Gradnetz kommt für mich als finale Lösung nicht in Frage, daher liegt das Hauptaugenmerk auf einer Rechnung der Integrationsschritte in einem lokalen Koordinatensystem. Die erste Inkarnation dieses Systems stellt eine orthographische Projektion um die momentane Position des betrachteten Partikels dar (skizziert in Abb. 3.7).

[18] was sich lediglich am Äquator nicht ausschließt

Zwar ist dies auf globaler Skale weder flächen- noch winkeltreu, kommt dem in lokaler Umgebung aber nahe. Zudem werden Kreise um die Startposition erhalten, also sind Verzerrungen der Projektion gleichmäßig in jede Richtung.

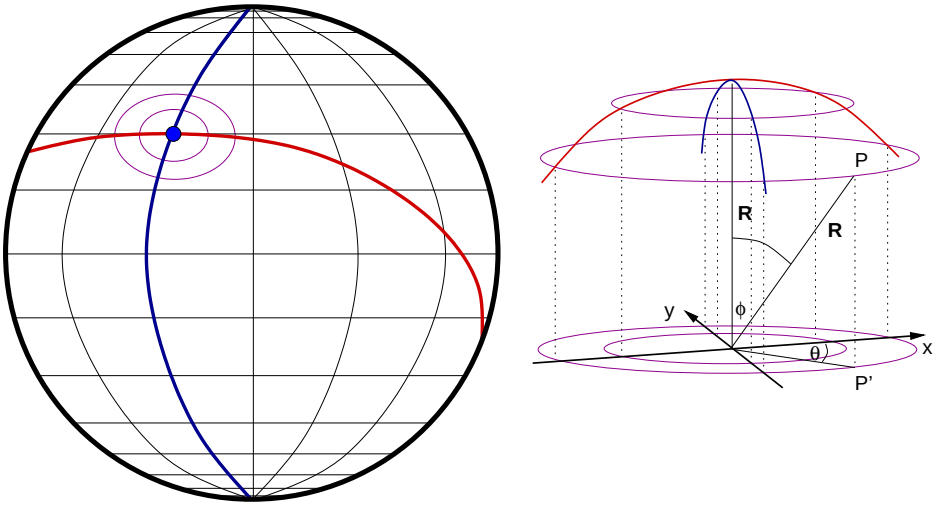


Abbildung 3.7: Skizze der lokalen orthographischen Projektion

Es folgt die Betrachtung dieser Transformation auf der Oberfläche — die vertikale Druckkoordinate wird ohne Änderung übernommen. Ich möchte einen Punkt (λ, β) in den Koordinaten (x, y) der orthographischen Projektion auf einen Bezugspunkt $P = (\lambda_p, \beta_p)$ ausdrücken.

Vor der eigentlichen Projektion überführe ich das geographische Koordinatensystem $(\lambda, \beta$ der Erdoberfläche (als ideale Kugeloberfläche) in ein kartesisches (x, y, z), welches die z-Achse vom Erdmittelpunkt durch P legt. Die z-Koordinate hat ihre Bedeutung in der Erfüllung der Kugelgeometrie und wird auch für die Rücktransformation genutzt, wobei die eigentliche Projektion nichts anderes ist als das Ersetzten von z durch die Druckkoordinate. Die Dicke der Atmosphäre wird hierbei gegenüber dem Erdradius vernachlässigt[19].

3.6.1 Äquivalente Dartellung im gedrehten kartesischen System

Der Übergang eines Punktes auf der Erdoberfläche in das an P ausgerichtete kartesische System erfolgt in drei anschaulichen Stufen. Beginnend mit den Ausgangskoordinaten (λ, β) des zu übertragenden Punktes sowie den Windrichtungen $\vec{e}_u$ und $\vec{e}_v$ im geographischen System, werden die in der Stufe i geltenden (nicht zwingend geänderten) Zwischenwerte mit (λ_i, β_i) bzw.$\vec{x}_i = (x_i, y_i, z_i)$ und die Windrichtungen mit $\vec{e}_{u,i}$ und $\vec{e}_{v,i}$ bezeichnet. Am Ende der Darstellung im gedrehten kartesischen System stehen also

$$\vec{x}_3 = \begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} ; \quad \vec{e}_{u,3} = \begin{pmatrix} u_{x,3} \\ u_{y,3} \\ u_{z,3} \end{pmatrix} ; \quad \vec{e}_{v,3} = \begin{pmatrix} v_{x,3} \\ v_{y,3} \\ v_{z,3} \end{pmatrix}$$

[19] Eine genauere Projektion als die orthographische mag auch die Höhe über dem Erdboden einbeziehen; hier erscheint mir das aber nicht sinnvoll.

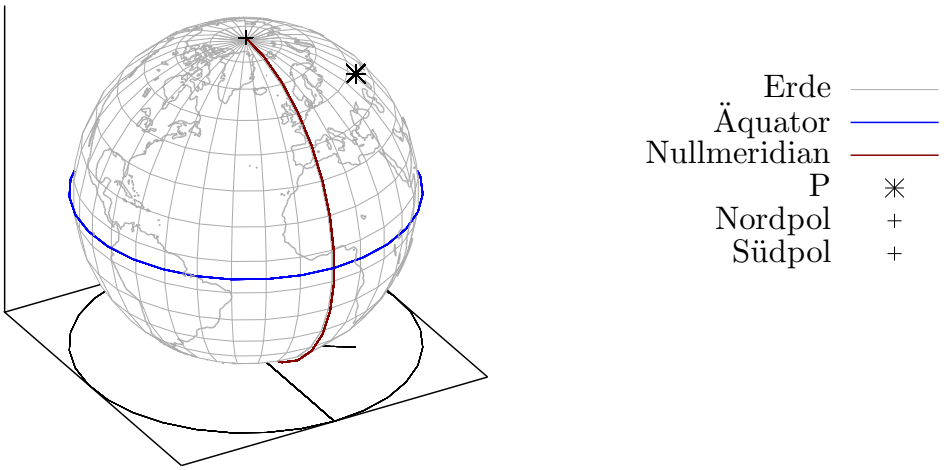


Abbildung 3.8: Lokale Transformation, Stufe 0
Es ist die Ausgangssituation im geographischen System mit dem beliebigen Punkt P sowie Hervorhebung des Nullmeridians und des Äquators gezeigt.

Die Stufen selbst sind eine Drehung (Ausrichtung der folgend definierten $x-Achse$) in Stufe 1, die Umwandlung in ein kartesisches System in Stufe 2 und letztendlich die Drehung dieses kartesischen Systems um die x-Achse, um die z-Achse durch P zu führen.

Stufe 1: Drehung um Polachse, neuer Nullmeridian

Die erste Drehung erfolgt recht einfach um die Polachse und sorgt dafür, dass die Drehung im letzten Schritt um eine Koordinatenachse erfolgt (x-Achse). Es sei

$$\begin{aligned} \lambda_1(\lambda,\beta) &= \lambda - \lambda_p - \frac{\pi}{2} \\ \beta_1(\lambda,\beta) &= \beta \end{aligned} \tag{3.23}$$

Die Windrichtungen sind davon nicht zusätzlich betroffen: $\vec{e}_{u,1} = \vec{e}_u$; $\vec{e}_{v,1} = \vec{e}_v$. Im System der Stufe 1 hat P die Kugelkoordinaten $(-\frac{\pi}{2}, \beta_p, R)$.

Stufe 2: Definition eines kartesischen Systems für die Kugel

Vor der zweiten Drehung rechne ich die Kugelkoordinaten um in ein dreidimensionales kartesisches System (siehe Abb. 3.9).

$$\begin{aligned} x_2(\lambda_1,\beta_1) &= R \cdot \cos\beta_1 \cdot \cos\lambda_1 \\ y_2(\lambda_1,\beta_1) &= R \cdot \cos\beta_1 \cdot \sin\lambda_1 \\ z_2(\lambda_1,\beta_1) &= R \cdot \sin\beta_1 \end{aligned} \tag{3.24}$$

Der Ursprung ist der Erdmittelpunkt, die positive x_2-Achse verläuft in der Äquatorialebene durch den Längengrad 0, die positive y_2-Achse durch den Längengrad $\frac{\pi}{2}$ (90°) und die positive z_2-Achse durch den Nordpol.

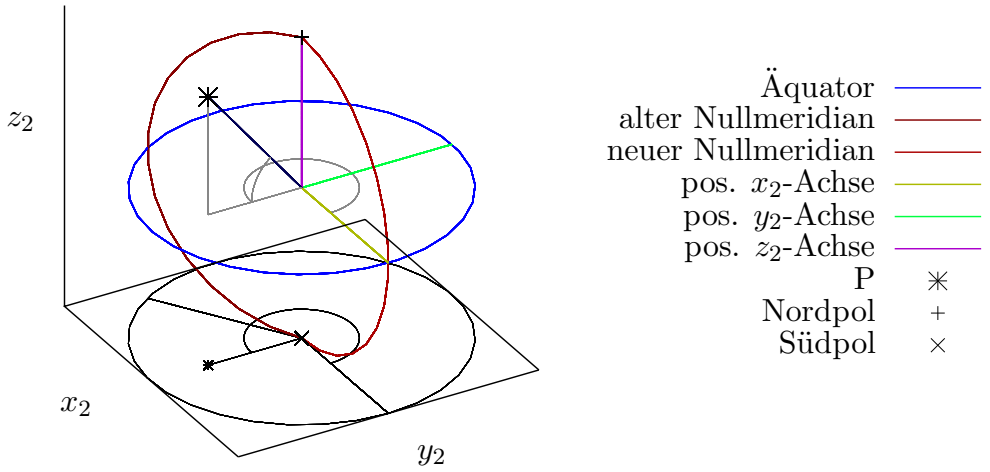


Abbildung 3.9: Lokale Transformation, Stufe 1 und 2
Hier sind die erste Drehung (Stufe 1) und die Definition von kartesischen Koordinaten (Stufe 2) zusammengefasst.

Hier erfolgt auch der Blick auf die Winde; Nord- und Ost-Richtung müssen ortsabhängig auf die kartesischen Achsen projiziert werden. Sprich, ich brauche $\vec{e}_{u,1}(\lambda_1, \beta_1)$ und $\vec{e}_{v,1}(\lambda_1, \beta_1)$ im kartesischen System, um sie im nächsten Schritt einfach mit zu transformieren.

Die Ostrichtung $\vec{e}_u$ zeigt immer parallel zum Äquator, also ist die z_2-Komponente gleich Null, da die z_2-Achse mit der Polachse zusammenfällt. Folglich ist die Ostrichtung im kartesischen System der Stufe 2 gegeben durch

$$\vec{e}_{u,2}(\lambda_1, \beta_1) = \begin{pmatrix} -\sin\lambda_1 \\ \cos\lambda_1 \\ 0 \end{pmatrix} \tag{3.25}$$

Dies ist ein normierter Einheitsvektor: $|\vec{e}_{u,2}| = \sqrt{sin^2\lambda_1 + \cos^2\lambda_1} = \sqrt{1} = 1$

Die Nordrichtung auf der Kugel ist nicht ganz so einfach. hier gibt es Komponenten zu allen Achsen, jedoch kann man sich dem schrittweise nähern. Betrachtet man den Nord-Einheitsvektor...

$$\begin{aligned} e_{v,z,2} &= \cos\beta_1 \\ e_{v,xy,2} &= -\sin\beta_1 \end{aligned}$$

($e_{v,xy,2} < 0$ meint zum Ursprung zeigend)

Die xy-Komponente verteilt sich nun anhand λ_1:

$$\begin{aligned} e_{v,x,2} &= \cos\lambda_1 e_{v,xy,2} = -\cos\lambda_1 \sin\beta_1 \\ e_{v,y,2} &= \sin\lambda_1 e_{v,xy,2} = -\sin\lambda_1 \sin\beta_1 \end{aligned}$$

Somit als Ganzes:

$$\vec{e}_{v,2}(\lambda_1, \beta_1) = \begin{pmatrix} -\cos\lambda_1 \sin\beta_1 \\ -\sin\lambda_1 \sin\beta_1 \\ \cos\beta_1 \end{pmatrix} \tag{3.26}$$

Ich prüfe wiederum die Normiertheit dieses Einheitsvektors:

$$\begin{aligned} |\vec{e}_{v,2}| &= \cos^2\lambda_1 \sin^2\beta_1 + \sin^2\lambda_1 \sin^2\beta_1 + \cos^2\beta_1 \\ &= (1-\sin^2\lambda_1)\sin^2\beta_1 + \sin^2\lambda_1 \sin^2\beta_1 + \cos^2\beta_1 \\ &= \sin^2\beta_1 + \cos^2\beta_1 \\ &= 1 \end{aligned}$$

Die Einheitsvektoren für die Windrichtungen sind bestimmt; der horizontale Wind weht nach

$$\vec{v}(\lambda_1,\beta_1) = u\cdot\vec{e}_{u,2}(\lambda_1,\beta_1) + v\cdot\vec{e}_{v,2}(\lambda_1,\beta_1) \tag{3.27}$$

Stufe 3: Drehung um x-Achse

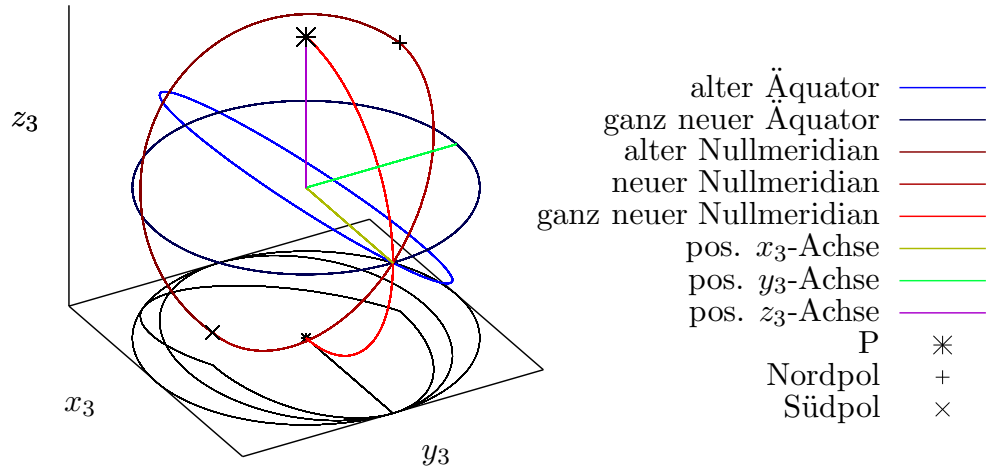


Abbildung 3.10: Lokale Transformation, Stufe 3
Die letzte Drehung um die x_2-Achse (identisch mit x_3-Achse) führt zur Stufe 3. Der "ganz neue" Nullmeridian ist zur Orientierung eingezeichnet — eigentlich existiert er im System der Stufe 3 nicht, da dieses lediglich in kartesischen Koordinaten gegeben ist.

Zuletzt muss das System zu P hin geneigt werden, also um den Winkel $\frac{\pi}{2}-\beta_p$ um die x-Achse gedreht werden. Allgemein bekannt ist die Drehmatrix M_α im $\mathbb{R}^3$ für die Drehung um den Winkel α um die x-Achse gegen den Uhrzeigersinn und die inverse $D_{-\alpha}$ für die Drehung im Uhrzeigersinn:

$$D_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{3.28}$$

$$D_{-\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-\alpha) & -\sin(-\alpha) \\ 0 & \sin(-\alpha) & \cos(-\alpha) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix} \tag{3.29}$$

Ich möchte das Koordinaten*system* vom Nordpol zu P hin drehen – *gegen* den Uhrzeigersinn. Das bedeutet, dass ich die *Koordinaten mit* den Uhrzeigersinn drehe. Der Betrag des Winkels

ist die Breitendifferenz vom *Nordpol* zu P, $\alpha = \frac{\pi}{2} - \beta_p$. Zur Vereinfachung der Drehmatrix sind folgende Identitäten hilfreich:

$$\begin{aligned} \cos\left(\frac{\pi}{2} - \gamma\right) &= \sin\gamma \\ \sin\left(\frac{\pi}{2} - \gamma\right) &= \cos\gamma \end{aligned} \tag{3.30}$$

Die Drehmatrix für die Stufe 3 ist somit $D_{-(\pi 2 - \beta_p)}$:

$$\begin{aligned} D_3 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\frac{\pi}{2} - \beta_p) & \sin(\frac{\pi}{2} - \beta_p) \\ 0 & -\sin(\frac{\pi}{2} - \beta_p) & \cos(\frac{\pi}{2} - \beta_p) \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sin\beta_p & \cos\beta_p \\ 0 & -\cos\beta_p & \sin\beta_p \end{pmatrix} \end{aligned} \tag{3.31}$$

Zusammen ergeben diese Stufen die Transformationen der globalen Koordinaten (λ, β) (auf der Erdoberfläche) in die lokalen Koordinaten (x, y, z) mit dem Erdmittelpunkt als Ursprung.

$$\begin{aligned} \vec{x}_3 &= \begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} = D_3 \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sin\beta_p & \cos\beta_p \\ 0 & -\cos\beta_p & \sin\beta_p \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} \\ &= \begin{pmatrix} x_2 \\ y_2\sin\beta_p + z_2\cos\beta_p \\ -y_2\cos\beta_p + z_2\sin\beta_p \end{pmatrix} = \begin{pmatrix} R\cdot\cos\beta_1\cdot\cos\lambda_1 \\ R\cdot\cos\beta_1\cdot\sin\lambda_1\sin\beta_p + R\cdot\sin\beta_1\cos\beta_p \\ -R\cdot\cos\beta_1\cdot\sin\lambda_1\cos\beta_p + R\cdot\sin\beta_1\sin\beta_p \end{pmatrix} \\ &= R\begin{pmatrix} \cos\beta\cdot\cos(\lambda - \lambda_p - \frac{\pi}{2}) \\ \cos\beta\cdot\sin(\lambda - \lambda_p - \frac{\pi}{2})\sin\beta_p + \sin\beta\cos\beta_p \\ -\cos\beta\cdot\sin(\lambda - \lambda_p - \frac{\pi}{2})\cos\beta_p + \sin\beta\sin\beta_p \end{pmatrix} \end{aligned} \tag{3.32}$$

$$\begin{aligned} \vec{e}_{u,3} &= D_3\vec{e}_{u,2} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sin\beta_p & \cos\beta_p \\ 0 & -\cos\beta_p & \sin\beta_p \end{pmatrix} \begin{pmatrix} -\sin\lambda_1 \\ \cos\lambda_1 \\ 0 \end{pmatrix} = \begin{pmatrix} -\sin\lambda_1 \\ \sin\beta_p\cos\lambda_1 \\ -\cos\beta_p\cos\lambda_1 \end{pmatrix} \\ &= \begin{pmatrix} -\sin(\lambda - \lambda_p - \frac{\pi}{2}) \\ \sin\beta_p\cos(\lambda - \lambda_p - \frac{\pi}{2}) \\ -\cos\beta_p\cos(\lambda - \lambda_p - \frac{\pi}{2}) \end{pmatrix} \end{aligned} \tag{3.33}$$

$$\begin{aligned} \vec{e}_{v,3} &= D_3\vec{e}_{v,2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sin\beta_p & \cos\beta_p \\ 0 & -\cos\beta_p & \sin\beta_p \end{pmatrix} \begin{pmatrix} -\cos\lambda_1\sin\beta \\ -\sin\lambda_1\sin\beta \\ \cos\beta \end{pmatrix} \\ &= \begin{pmatrix} -\cos\lambda_1\sin\beta \\ -\sin\beta_p\sin\lambda_1\sin\beta + \cos\beta_p\cos\beta \\ \cos\beta_p\sin\lambda_1\sin\beta + \sin\beta_p\cos\beta \end{pmatrix} \\ &= \begin{pmatrix} -\cos(\lambda - \lambda_p - \frac{\pi}{2})\sin\beta \\ -\sin\beta_p\sin(\lambda - \lambda_p - \frac{\pi}{2})\sin\beta + \cos\beta_p\cos\beta \\ \cos\beta_p\sin(\lambda - \lambda_p - \frac{\pi}{2})\sin\beta + \sin\beta_p\cos\beta \end{pmatrix} \end{aligned} \tag{3.34}$$

Ich erachte es für sinnvoll, hier einen Moment zu verweilen. In solch eine Herleitung kann sich leicht ein Fehler eingeschlichen haben; ein Sinus anstatt eines Kosinus oder auch einfach ein Plus anstatt eines Minus[20].

Test: Kugelgeometrie

Bisher wurden lediglich die Koordinaten eines Punktes auf der (idealen) Erdkugel transformiert, ohne die Projektion selbst auszuführen. Somit muss $\vec{x}_3$ den Betrag R haben, wenn die Rechnung bisher korrekt verlief. Ich teste dies:

$$
\begin{aligned}
|\vec{x}_3| &= \sqrt{x_3^2+y_3^2+z_3^2} = R\cdot\left(\left(\frac{x_3}{R}\right)^2+\left(\frac{y_3}{R}\right)^2+\left(\frac{z_3}{R}\right)^2\right)^{-\frac{1}{2}} \\
\Rightarrow \left(\frac{|\vec{x}_3|}{R}\right)^2 &= \Big(\cos\beta\cdot\cos(\lambda-\lambda_p-\frac{\pi}{2})\Big)^2 \\
&\quad +\Big(\cos\beta\cdot\sin(\lambda-\lambda_p-\frac{\pi}{2})\sin\beta_p+\sin\beta\cos\beta_p\Big)^2 \\
&\quad +\Big(-\cos\beta\cdot\sin(\lambda-\lambda_p-\frac{\pi}{2})\cos\beta_p+\sin\beta\sin\beta_p\Big)^2 \\
&= \cos^2\beta\underbrace{\Big(\cos^2(\lambda-\lambda_p-\frac{\pi}{2})+\sin^2(\lambda-\lambda_p-\frac{\pi}{2})(\sin^2\beta_p+\cos^2\beta_p)\Big)}_{=1} \\
&\quad +2\cos\beta\cdot\sin(\lambda-\lambda_p-\frac{\pi}{2})\sin\beta_p\sin\beta\cos\beta_p \\
&\quad -2\cos\beta\cdot\sin(\lambda-\lambda_p-\frac{\pi}{2})\cos\beta_p\sin\beta\sin\beta_p \\
&\quad +\sin^2\beta(\sin^2\beta_p+\cos^2\beta_p) \\
&= \cos^2\beta+\sin^2\beta = 1
\end{aligned}
$$

Wie wir sehen, ist dies tatsächlich der Fall. Ich werde noch weitere spezielle Testfälle bei P und dessen Gegenpol $P' = (\lambda_p+\pi, -\beta_p)$ betrachten. Dieser Gegenpol liegt zwar außerhalb des gültigen Bereiches für die später folgende orthographische Projektion, muss aber von der bisher durchgeführten global gültigen Transformation korrekt behandelt werden.

Tests bei P und Gegenpol P'

Ich betrachte die Transformation von Ort und Wind bei P mit

$$
\begin{aligned}
&\lambda = \lambda_p;\ \beta = \beta_p \\
&\lambda_1 = \lambda_p - \lambda_p - \tfrac{\pi}{2} = -\tfrac{\pi}{2}
\end{aligned}
$$

sowie den Gegenpol P' mit

$$
\begin{aligned}
&\lambda = \lambda_p+\pi;\ \beta = -\beta_p \\
&\lambda_1 = \lambda_p+\pi - \lambda_p - \tfrac{\pi}{2} = \tfrac{\pi}{2}
\end{aligned}
$$

[20] ☺ Der menschliche Intellekt ist einfach nicht gut bei harten Ja/Nein Fragen, oft verschätzt man sich leicht und landet bei Nein, wo man doch Ja meint. Die menschliche Antwort auf "Ja oder Nein?" kann eigentlich nur "Vielleicht?" lauten.

Die zu erwarteten Werte für die $\vec{x}_3$-Koordinaten der beiden Punkte sowie der entsprechenden Windkomponenten sind ohne explizite Rechnung aus der Überlegung klar. Weiterhin sorgt $\lambda_1 = \pm\frac{\pi}{2}$ für schnelle Auflösung der Winkelfunkionen durch $\cos\left(\pm\frac{\pi}{2}\right) = 0$ und $\sin\left(\pm\frac{\pi}{2}\right) = \pm 1$.

Ort von P Für die Transformation des Ortes von P erwarte ich die $\vec{x}_3$-Koordinaten $(0, 0, R)$:

$$\begin{aligned}
\vec{x}_3(\lambda_p, \beta_p) &= R \begin{pmatrix} \cos\beta_p \cos\left(-\frac{\pi}{2}\right) \\ \cos\beta_p \sin\left(-\frac{\pi}{2}\right) \sin\beta_p + \sin\beta_p \cos\beta_p \\ -\cos\beta_p \sin\left(-\frac{\pi}{2}\right) \cos\beta_p + \sin\beta_p \sin\beta_p \end{pmatrix} \\
&= R \begin{pmatrix} 0 \\ -\cos\beta_p \sin\beta_p + \sin\beta_p \cos\beta_p \\ \cos\beta_p \cos\beta_p + \sin\beta_p \sin\beta_p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ R \end{pmatrix}
\end{aligned}$$

Südwind bei P Die Nordrichtung am Ort P sollte in die y-Komponente des transformierten Vektors übergehen.

$$\begin{aligned}
\Rightarrow \vec{e}_{v,3}(\lambda_p, \beta_p) &= \begin{pmatrix} -\cos\left(-\frac{\pi}{2}\right) \sin\beta_1 \\ -\sin\beta_p \sin\left(-\frac{\pi}{2}\right) \sin\beta_p + \cos\beta_p \cos\beta_p \\ \cos\beta_p \sin\left(-\frac{\pi}{2}\right) \sin\beta_p + \sin\beta_p \cos\beta_p \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ \sin\beta_p \sin\beta_p + \cos\beta_p \cos\beta_p \\ -\cos\beta_p \sin\beta_p + \sin\beta_p \cos\beta_p \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}
\end{aligned}$$

Westwind bei P Die Ostrichtung am Ort P sollte in die x-Komponente des transformierten Vektors übergehen.

$$\vec{e}_{u,3}(\lambda_p, \beta_p) = \begin{pmatrix} -\sin\left(-\frac{\pi}{2}\right) \\ \sin\beta_p \cos\left(-\frac{\pi}{2}\right) \\ -\cos\beta_p \cos\left(-\frac{\pi}{2}\right) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Ort von P' Für die Transformation des Ortes von P' erwarte ich die Koordinaten von P mit invertiertem z_3:

$$\begin{aligned}
\vec{x}_3(\lambda_p + \pi, -\beta_p) &= R \begin{pmatrix} \cos(-\beta_p) \cos\frac{\pi}{2} \\ \cos(-\beta_p) \sin\frac{\pi}{2} \sin\beta_p + \sin(-\beta_p) \cos\beta_p \\ -\cos(-\beta_p) \sin\frac{\pi}{2} \cos\beta_p + \sin(-\beta_p) \sin\beta_p \end{pmatrix} \\
&= R \begin{pmatrix} 0 \\ \cos\beta_p \sin\beta_p - \sin\beta_p \cos\beta_p \\ -\cos\beta_p \cos\beta_p - \sin\beta_p \sin\beta_p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -R \end{pmatrix}
\end{aligned}$$

Südwind bei P' Die Nordrichtung am Ort P' sollte wie auch die bei P in die y-Komponente des transformierten Vektors übergehen.

$$\begin{aligned}\vec{e}_{v,3}(\lambda_p+\pi,-\beta_p) &= \begin{pmatrix} -\cos\frac{\pi}{2}\sin(-\beta_p) \\ -\sin\beta_p\sin\frac{\pi}{2}\sin(-\beta_p)+\cos\beta_p\cos(-\beta_p) \\ \cos\beta_p\sin\frac{\pi}{2}\sin(-\beta_p)+\sin\beta_p\cos(-\beta_p)\end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \sin\beta_p\sin\beta_p+\cos\beta_p\cos\beta_p \\ -\cos\beta_p\sin\beta_p+\sin\beta_p\cos\beta_p\end{pmatrix} = \begin{pmatrix}0\\1\\0\end{pmatrix}\end{aligned}$$

Westwind bei P' Die Ostrichtung am Ort P' sollte in die x-Komponente des transformierten Vektors übergehen, allerdings negativ, da Osten auf der anderen Seite der Erdkugel genau die entgegengesetzte Richtung bedeutet.

$$\vec{e}_{u,3}(\lambda_p+\pi,-\beta_p) = \begin{pmatrix} -\sin\frac{\pi}{2} \\ \sin\beta_p\cos\frac{\pi}{2} \\ -\cos\beta_p\cos\frac{\pi}{2}\end{pmatrix} = \begin{pmatrix}-1\\0\\0\end{pmatrix}$$

3.6.2 Rücktransformation in das geographische System

Wenn der Integrationsschritt im lokalen kartesischen System stattfindet, muss danach der neu errechnete Ort wieder in das globale geographische System übertragen werden. Dazu verfolge ich die Schritte zurück, beginned mit der Drehung von Schritt 3. Ich muss um den Winkel $\frac{\pi}{2}-\beta_p$ zurückdrehen, was gleichbedeutend mit der Invertierung von D_3 ist. Die entsprechende Matrix für $\alpha=\frac{\pi}{2}-\beta_p$ ist in Gl. 3.28 zu finden.

$$\begin{aligned}\vec{x}_2 &= D^{-1}\vec{x}_3 = \begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\frac{\pi}{2}-\beta_p) & -\sin(\frac{\pi}{2}-\beta_p)\\ 0 & \sin(\frac{\pi}{2}-\beta_p) & \cos(\frac{\pi}{2}-\beta_p)\end{pmatrix}\vec{x}_3 \\ &= \begin{pmatrix}1 & 0 & 0\\ 0 & \sin\beta_p & -\cos\beta_p\\ 0 & \cos\beta_p & \sin\beta_p\end{pmatrix}\vec{x}_3 = \begin{pmatrix} x_3 \\ y_3\sin\beta_p - z_3\cos\beta_p \\ y_3\cos\beta_p + z_3\sin\beta_p\end{pmatrix}\end{aligned} \tag{3.35}$$

Nach Anwendung der Rückdrehung muss Stufe 2 rückgängig gemacht werden: Die Umrechnung von kartesischen in geographische Kugelkoordinaten. Das System Gl. 3.24 muss nach λ_1 und β_1 aufgelöst werden:

$$\begin{aligned} G1: &\quad \frac{x_2}{R} = \cos\beta_1\cdot\cos\lambda_1 \\ G2: &\quad \frac{y_2}{R} = \cos\beta_1\cdot\sin\lambda_1 \\ G3: &\quad \frac{z_2}{R} = \sin\beta_1\end{aligned} \tag{3.36}$$

Die Lösung erfolgt über den Tangens; ich suche

$$\begin{aligned}\tan\lambda_1 &= \frac{\sin\lambda_1}{\cos\lambda_1} \\ \tan\beta_1 &= \frac{\sin\beta_1}{\cos\beta_1}\end{aligned}$$

was für λ_1 schnell erfolgreich ist:

$$G2/G1: \qquad \frac{\sin\lambda_1}{\cos\lambda_1} = \frac{y_2}{x_2}$$

Für β_1 muss etwas mehr geschehen:

$$\begin{aligned}
G1^2: &\qquad \frac{x_2^2}{R^2} = \cos^2\beta_1 \cos^2\lambda_1 \\
G2^2: &\qquad \frac{y_2^2}{R^2} = \cos^2\beta_1 \sin^2\lambda_1 \\
G1^2 + G2^2: &\qquad \frac{1}{R^2}(x_2^2 + y_2^2) = \cos^2\beta_1 \underbrace{(\sin^2\lambda_1 + \cos^2\lambda_1)}_{=1} = \cos^2\beta_1 \\
\frac{G3}{\sqrt{G1^2 + G2^2}}: &\qquad \frac{z_2}{\sqrt{x_2^2 + y_2^2}} = \frac{\sin\beta_1}{\cos\beta_1}
\end{aligned}$$

So sind beide Tangens-Werte gefunden und die Winkel sind bestimmt durch:

$$\begin{aligned}
\tan\lambda_1 = \frac{\sin\lambda_1}{\cos\lambda_1} &= \frac{y_2}{x_2} \\
\tan\beta_1 = \frac{\sin\beta_1}{\cos\beta_1} &= \frac{z_2}{\sqrt{x_2^2 + y_2^2}}
\end{aligned} \tag{3.37}$$

In der Literatur (siehe [Bronstein2000]) findet man die Umrechnung von kartesischen in "mathematische" Kugelkoordinaten im Gegensatz zu geographischen:

$$\begin{aligned}
\tan\lambda &= \frac{y}{x} \\
\tan\beta &= \frac{\sqrt{x^2 + y^2}}{z}
\end{aligned} \tag{3.38}$$

Diese ist mit der hier hergeleiteten identisch, wenn man die unterschiedliche Definition des Breitenwinkels β einbezieht und Gl. 3.30 nutzt:

$$\begin{aligned}
&\beta_{\text{geo}} = \frac{\pi}{2} - \beta_{\text{math}} \\
\Rightarrow \qquad &\sin\beta_{\text{geo}} = \sin\left(\frac{\pi}{2} - \beta_{\text{math}}\right) = \cos\beta_{\text{math}} \\
&\cos\beta_{\text{geo}} = \cos\left(\frac{\pi}{2} - \beta_{\text{math}}\right) = \sin\beta_{\text{math}} \\
\Rightarrow \qquad &\tan\beta_{\text{geo}} = \frac{\sin\beta_{\text{geo}}}{\cos\beta_{\text{geo}}} = \frac{\cos\beta_{\text{math}}}{\sin\beta_{\text{math}}} = (\tan\beta_{\text{math}})^{-1}
\end{aligned}$$

Die Tangensumkehr Die arctan Funktion wird also aus den kartesischen Koordinaten wieder Winkel hervorbringen. Grundsätzlich ist diese Funktion nur eindeutig, wenn man sie auf ein bestimmtes Intervall beschränkt. Da ich hier letztendlich auf die Arbeit auf elektronischen Rechnern ziele, konzentriere ich mich nicht lediglich auf theoretische (auch durchaus sinnvolle) Konvention, sondern frage die Maschine direkt. Das ergibt[21]:

[21] `rechne` ist ein Perl-Programm, welches die übergebene Formel auswertet und das Ergebnis ausgibt. Letztendlich nutzt es dieselben C-Bibliotheksfunktionen wie mein Trajektorienprogramm.

```
shell$ rechne 'atan(0)'
0
shell$ rechne 'atan(1000)'
1.56979632712823
shell$ rechne 'atan(-1000)'
-1.56979632712823
```

Das praktisch gewählte Intervall ist also

$$\arctan \quad : \quad (-\infty;\infty) \rightarrow \left[-\frac{\pi}{2};\frac{\pi}{2}\right] \tag{3.39}$$

Dies ist auch exakt das, was der ISO C90 Standard[22] spezifiziert[23]. Ich bekomme somit aus

$$\lambda_1 = \arctan\frac{y_2}{x_2} \tag{3.40}$$

$$\beta_1 = \arctan\frac{z_2}{\sqrt{x_2^2+y_2^2}} \tag{3.41}$$

direkt Werte für β_1 im gewünschten Intervall ($\beta_1 < 0$ bei $z < 0$ und $\beta_1 > 0$ bei $z > 0$) und für λ_1 muss ich zusätzlich prüfen, ob $x_2 < 0$, um zu entscheiden ob der erhaltene Wert aus $\left[-\frac{\pi}{2};\frac{\pi}{2}\right]$ um π auf die gegenüberliegende Halbkugel verschoben werden soll.

Die Problemstellen: Zwar kann man arctan für $\pm\infty$ nach $\pm\frac{\pi}{2}$ sowie hier $\frac{0}{0} = 0$ definieren und bekäme so auch für grenzwertige $\vec{x}_2$ sinnvolle Werte, jedoch ist es mit Blick auf die Programmierung eines Rechners angebracht, die problematischen Randfälle gesondert zu betrachten, Gl. 3.41 wird problematisch, wenn x_2 und y_2 gleichzeitig gleich Null sind. β_1 ist aber dann klar $\frac{\pi}{2}$, wenn z_2 größer Null, und $-\frac{\pi}{2}$, wenn z_2 kleiner Null. Gl. 3.40 swird problematisch, wenn x_2 gleich Null. Da ist λ gleich $\frac{\pi}{2}$ für y_2 größer Null und $-\frac{\pi}{2}$ für y kleiner Null. Wenn x_2 und y_2 beide gleich Null sind, dann ist der Wert von λ_1 irrelevant.

Wenn wir dann wieder im System der Stufe 1 mit λ_1 und θ_1 sind, dann braucht es nur noch

$$\begin{aligned} \beta &= \beta_1 \\ \lambda &= \lambda_1 + \lambda_p + \frac{\pi}{2} \end{aligned} \tag{3.42}$$

zu den normalen geographischen Koordinaten.

[22] ISO/IEC 9899-1990, Programming Languages - C

[23] Man sollte sich weder blind auf den Standard, noch blind auf auf eine spezielle Implementation verlassen, um böse Überraschungen zu vermeiden.

Ein Algorithmus für die komplettee Rücktransformation des Ortes von $\vec{x}_3$ zu (λ, β) unter Beachtung der Randfälle ist in Gl. 3.43 dargestellt:

$$
\begin{aligned}
&x_2 = x_3 \\
&y_2 = y_3 \sin\beta_p - z_3 \cos\beta_p \\
&z_2 = y_3 \cos\beta_p + z_3 \sin\beta_p
\end{aligned}
$$

$$
\mathbf{x_2 = 0} \left\| \begin{array}{ll} & \lambda_1 = \left(y_2 > 0 \;\mathbf{?}\; \frac{\pi}{2} \;\mathbf{:}\; -\frac{\pi}{2}\right) \\ \mathbf{?} & \mathbf{y_2 = 0} \left\| \begin{array}{ll} \mathbf{?} & \beta_1 = \left(z_2 > 0 \;\mathbf{?}\; \frac{\pi}{2} \;\mathbf{:}\; -\frac{\pi}{2}\right) \\ \mathbf{:} & \beta_1 = \arctan\frac{z_2}{\sqrt{x_2^2+y_2^2}} = \arctan\frac{z_2}{y_2} \end{array} \right. \\ \hline \mathbf{:} & \begin{array}{l} \lambda_1 = \arctan\frac{y_2}{x_2} + (x_2 < 0 \;\mathbf{?}\; \pi \;\mathbf{:}\; 0) \\ \beta_1 = \arctan\frac{z_2}{\sqrt{x_2^2+y_2^2}} \end{array} \end{array} \right. \tag{3.43}
$$

$$
\begin{aligned}
&\lambda = \lambda_1 + \lambda_p + \tfrac{\pi}{2} \\
&\beta = \beta_1
\end{aligned}
$$

Wenn $x_2 = 0$, dann ist die Länge auf $\frac{\pi}{2}$ für positives y_2 und $-\frac{\pi}{2}$ für negatives y_2 festgelegt. Ist ebenfalls $y_2 = 0$, so befinden wir uns an einem der Pole, wo die Länge beliebig ist und die Breite $\frac{\pi}{2}$ für positive z_2 (Nordpol) und $-\frac{\pi}{2}$ für negative z (Südpol).

Auch wenn IEEE Fließkommarechnung gewisse Handhabung von Teilung durch 0 unterstützt, so ist es doch sicherer, die offensichtlichen Fälle explizit zu handhaben. Speziell würde $0/0$ zu NaN führen, was hier nicht dienlich ist.

Zudem kommt noch die korrekte Bestimmung von λ_1 ins Spiel – für $x_2 < 0$ ist der Gegenwinkel auf der anderen Halbkugel gemeint.

3.6.3 Orthographische Projektion, Transformation hin und zurück

Die eigentliche orthographische Projektion ist nun das simple Ignorieren der z_3 Koordinate (Projektion aus dem Unendlichen auf die x_3y_3-Ebene), die geometrische Betrachtung der lokalen Bewegung in einer xy-Tangentialebene um das Zentrum des Interesses P. Anstatt z_3 wird die Druckkoordinate der für die Transformation ja als flach betrachteten Atmosphäre genutzt.

Die Arbeitskoordinaten sind

$$
\vec{x} \;=\; \begin{pmatrix} x \\ y \\ p \end{pmatrix} \tag{3.44}
$$

mit dem Druck p als dritte Koordinate. Dazu kommt dann die Geschwindigkeit

$$
\dot{\vec{x}} \;=\; \begin{pmatrix} ue_{u,x} + ve_{v,x} \\ ue_{u,y} + ve_{v,y} \\ w \end{pmatrix} \tag{3.45}
$$

Bei einem Integrationsschritt kommen dann neue Koordinaten x, y und p heraus, wovon p bei der Rückübertragung ins globale System einfach übernommen wird und x, y zu λ, β

führen. Indirekt wird dafür ein z benötigt; welches aus der simplen Bedingung folgt, dass wir auf einer Kugel mit Radius R bleiben.

$$\begin{aligned} z^2 &= R^2 - (x^2 + y^2) \\ z &= +\sqrt{(R^2 - (x^2 + y^2))} \end{aligned} \tag{3.46}$$

Dieses kommt anstatt des "echten" z_3 zum Einsatz. Hinter der Festlegung auf die positive Wurzel verbirgt sich die Beschränkung der Betrachtung auf die Halbkugel um P, d.h. nur Punkte, die im Umkreis von einem Viertel Erdumfang um P liegen. Da es schließlich um ein *lokal* gültiges Koordinatensystem geht, stellt das kein Problem dar – ich werde keine Rechnung in Betracht ziehen, die in einem lokalen Integrationsschritt Bewegungen um tausende Kilometer beinhaltet.

Abschließend die Transformationen für den Basispunkt $P(\lambda_p, \beta_p)$ und die Koordinaten $\vec{r} = (\lambda, \beta, p)$ im Erdsystem sowie $\vec{x} = (x, y, p)$ im Lokalsystem.

Ort – hin Es erfolgt eine Drehung um die Polachse, wobei die Winkel λ_1 und β_1 entstehen ($\beta_1 = \beta$). Danach erfolgt die Umdeutung der Welt in kartesische Koordinaten: $\vec{x}_2$. Letzter Schritt ist die Drehung des Systems mit diesen um den Winkel $\frac{\pi}{2} - \beta_p$: $\vec{x}_3$. Schlussendlich haben wir die lokalen Koordinaten $\vec{x}$, welche aus den ersten zwei Komponenten von $\vec{x}_3$ und der unveränderten Druckkoordinate p bestehen.

$$\vec{x} = \begin{pmatrix} R\cos\beta \cdot \cos(\lambda - \lambda_p - \frac{\pi}{2}) \\ R\left(\cos\beta \cdot \sin(\lambda - \lambda_p - \frac{\pi}{2})\sin\beta_p + \sin\beta\cos\beta_p\right) \\ p \end{pmatrix} \tag{3.47}$$

Ort – zurück Hier ist zu beachten, dass anstatt z nun die Kugelbedingung in Form von Gl. 3.46 eintritt

$$\begin{aligned} \lambda &= \arctan\left(\frac{y\sin\beta_p - \sqrt{(R^2 - (x^2 + y^2))}\cos\beta_p}{x}\right) + \lambda_p + \frac{\pi}{2} \\ &\quad + (x < 0 \;\mathbf{?}\; \pi \;\mathbf{:}\; 0) \end{aligned} \tag{3.48}$$

$$\beta = \arctan\left(\frac{y\cos\beta_p + \sqrt{(R^2 - (x^2 + y^2))}\sin\beta_p}{\sqrt{(x)^2 + \left(y\sin\beta_p - \sqrt{(R^2 - (x^2 + y^2))}\cos\beta_p\right)^2}}\right) \tag{3.49}$$

Wind Auch hier lassen wir den horizontalen Wind (global: u und v) nicht in z- bzw. p-Richtung wehen, der lokale 3D Wind besteht aus Projektionen von u und v und der unveränderten vertikalen Komponente ω.

$$\begin{aligned}
\vec{e}_u &= \begin{pmatrix} -\sin(\lambda - \lambda_p - \frac{\pi}{2}) \\ \sin\beta_p \cos(\lambda - \lambda_p - \frac{\pi}{2}) \\ 0 \end{pmatrix} \\
\vec{e}_v &= \begin{pmatrix} -\cos(\lambda - \lambda_p - \frac{\pi}{2}) \sin\beta \\ -\sin\beta_p \sin(\lambda - \lambda_p - \frac{\pi}{2}) \sin\beta + \cos\beta_p \cos\beta \\ 0 \end{pmatrix} \\
\vec{e}_w &= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
\dot{\vec{x}} &= u\vec{e}_u + v\vec{e}_v + w\vec{e}_w \qquad (3.50) \\
&= \begin{pmatrix} -u\sin(\lambda - \lambda_p - \frac{\pi}{2}) - v\cos(\lambda - \lambda_p - \frac{\pi}{2}) \sin\beta \\ u\sin\beta_p \cos(\lambda - \lambda_p - \frac{\pi}{2}) + v\left(-\sin\beta_p \sin(\lambda - \lambda_p - \frac{\pi}{2}) \sin\beta + \cos\beta_p \cos\beta\right) \\ w \end{pmatrix}
\end{aligned}$$

3.7 3D Integration mit lokalem System

Also, ich habe Gl. 3.47 bis Gl. 3.50, um in einem lokalen System um Punkt $P_0 = (\lambda_0, \beta_0, p_0)$ herum zu arbeiten. Was mache ich jetzt damit? Ich möchte in einem Integrationsschritt von P_0 nach P_1 (Zeitschritt Δt) kommen. Der grundlegende Algorithmus gliedert sich in drei Stufen:

1. Wechsel in das kartesische lokale System: $P_0 = (\lambda_0, \beta_0, p_0) \rightarrow (0, 0, p_0)_{\text{lok}}$
2. Integrationsschritt zu $(x_1, y_1, p_1)_{\text{lok}} = (\Delta x, \Delta y, p + \Delta p)_{\text{lok}}$
3. Wechsel in das globale System: $(x_1, y_1, p_1)_{\text{lok}} \rightarrow (\lambda_1, \beta_1, p_1) = P_1$

Im Laufe der Integration werde ich interpolierte Geschwindigkeiten an verschiedenen Punkten $Q = (x, y, p)_{\text{lok}}$ im lokalen System benötigen. Dazu brauche ich $Q = (\lambda(x, y),\ \beta(x, y), p)$ im globalen System, um die umliegenden Gitterpunkte[24] $G_n(\lambda_n, \beta_n, p_n)$ im Koordinatensystem der Wind-Datenquelle auf einfache Weise zu bestimmen sowie die Winde $\dot{\vec{r}}_n$ selbst auszulesen. Im lokalen System sind dies dann $G_n = (x_n(\lambda_n, \beta_n), y_n(\lambda_n, \beta_n))_{\text{lok}}$ sowie $\dot{\vec{x}}_n$. Durch die Koordinatentransformation sind zumindest die horizontalen Gitterzellen nich mehr rechtwinklig und in Polnähe ist es sogar eine Kreisanordnung. So kann die einfache lineare Interpolation aus Gl. 3.13 nur in der Höhe und Zeit angewendet werden. In der xy-Ebene nimmt ihren Platz die entfernungsgewichtete Summe aus Gl. 3.18 ein, welche mit beliebiger Punktanordnung umgeht[25].

Die Funktion `ensemble_rk4::single_step()` in Quelltext 3.6 vereinigt diese Bausteine und sorgt für Bewegung (von simulierten Luftpaketen). Durch die Abstraktion des Koordinatensystems kann sich hinter der angeblichen Transformation in ein lokales Koordinatensystem

[24] üblicherweise die 16 Eckpunkte der Raumzeit-Gitterzelle, and den Polen alle umliegenden Punkte mit entsprechend nördlichstem oder südlichsten Breitengrad

[25] Eine vorgesehene Verbesserung ist die Einbeziehung der Nähe der Gitterpunkte zueinander in das Gewicht.

auch die Identität auf dem globalen Geographischen System verbergen - so dass diese Funktion genauso für die Integration im Gradnetz ohne lokales kartesisches System eingesetzt werden kann.

Diese ist wiederum eingebettet in eine Klasse `ensemble_rk4`, welche ein Teilensemble von Partikeln zusammen repräsentiert. Mit der in Abs. 2.2 eingeführten Klasse `egp_wind` (hinter der Abstraktion der `data` Schnittstelle), `ensemble_rk4` (hinter der `ensemble` Schnittstelle), der lokalen Koordinatentransformation in `flat_pressure` (`coordinates`) und `world` (Standardimplementation der `world` Schnittstelle) als Verbindung von Daten, Koordinaten und Interpolation reichen wenige Programmzeilen, um ein großskaliges numerisches Tracerexperiment durchzuführen (Quelltext 3.7). Eine Bemerkung zur Großskaligkeit: Richtig skalieren kann man durch Unterteilung des Gesamtensembles und so serielles (wegen Arbeitsspeicherbegrenzung, Handhabung in Dateien) oder paralleles (Nutzung eines Clusters) Abarbeiten. Von mir genutzte Blockgrößen liegen zwischen 1000 und 100000 Partikeln; Geschwindigkeitsoptimum hängt maßgeblich von Parametern wie Prozessor-Cachegröße ab.

Quelltext 3.6: Einzelner RK4 Integrationsschritt

```
1  void ensemble_rk4::single_step(place pos)
2  {
3    place workplace;  // scratch variable for place (x)
4    place localstart; // start place in local system
5    spaceplace k[4];  // the RK4 coefficients
6    spaceplace half;  // scratch var for k/2
7    home->system->set_base(pos); // move local system
8    home->system->lplace(pos, localstart); // get local coordinates
9    // k1 = h*f(x0, y0)
10   if(home->get_lwind(localstart,k[0])) // get local wind
11   {
12     smul_space(k[0], timestep, k[0]);
13     // k2 = h*f(x0+h/2, y0+k1/2)
14     sdiv_space(k[0], 2, half);
15     add_space(localstart, half, workplace)
16     workplace[TIME] = localstart[TIME]+timestep/2;
17     if(home->get_lwind(workplace, k[1]))
18     {
19       smul_space(k[1], timestep, k[1]);
20       // k3 = h*f(x0+h/2, y0+k2/2)
21       sdiv_space(k[1], 2, half);
22       add_space(localstart, half, workplace);
23       // still: workplace[TIME] = localstart[TIME]+timestep/2;
24       if(home->get_lwind(workplace, k[2]))
25       {
26         smul_space(k[2], timestep, k[2]);
27         // k4 = h*f(x0+h, y0+k3)
28         add_space(localstart, k[2], workplace);
29         workplace[TIME] = localstart[TIME]+timestep;
30         if(home->get_lwind(workplace, k[3]))
31         {
32           smul_space(k[3], timestep, k[3]);
33           // x1 = x0 + (k0 + 2k1 + 2 k2 + k3)/6
34           for(size_t i = 0; i < SPACEDIMS; ++i)
35           workplace[i] = localstart[i]
36                        + (k[0][i] + k[1][i]*2 + k[2][i]*2 + k[3][i
                             ])/6;
37
38           home->system->gplace(workplace, pos);
39 } } } } }
```

Quelltext 3.7: Prototyp eines Tracerexperimentes

```
1  // parameters
2  pep_t begin   = START_TIME;
3  pep_t end     = END_TIME;
4  pep_t step    = TIMESTEP;
5  long nsteps   = (long) ((end-begin)/step);
6  pep_t members = NUMBER_OF_PARTICLES;
7  // compute/set the starting positions somehow
8  spaceplace *start = GET_START_POS(NUMBER_OF_PARTICLES);
9  // data source for ECHO-GiSP NetCDF files
10 data *pepper = new egp_wind; // could be some different source
       ...
11 pepper->init(datafiles, number_of_files);
12 world *w = new world; // the default world with lokal
       coordinates
13 coordinates *s = new flat_pressure; // the coordinate
       transformations
14 // connect the world with coordinates and data source
15 w->set_system(s);
16 w->set_source(pepper);
17 // create the ensemble and place it into our world
18 ensemble *p = new ensemble_rk4;
19 p->set_home(w);
20 p->set_timestep(step);
21 p->set_starttime(begin)
22 p->set_members(members);
23 p->beam(start); // set the starting positions
24 do
25 {
26   p->step(); // advance the ensemble one timestep
27   do_something_with(p->position);
28 }
29 while(p->steps < nsteps);
```

4 Globale Ensembles

4.1 Startpositionierungen

Nachdem geklärt ist, *wie* die Bewegung funktioniert, folgt die Frage, *was* bewegt werden soll. Diese Frage ist bei den betrachteten passiven Tracern reduziert auf die Startposition zu einem gegebenem Zeitpunkt[1]. Für ein Ensemble: viele Startpositionen in einem vorgegebenen Bereich. Wenn ich quantitative Untersuchungen der Durchmischung von Luftmassen unterschiedlicher Herkunft oder der Verteilung von Luft aus einem beschränkten Startgebiet durchführen will, muss ich den Partikeln (Subensembles) Gewichte / repräsentierte Luftmasse zuordnen oder dafür sorgen, dass alle die gleiche Masse repräsentieren und so Zählungen direkt aussagekräftig sind.

In erster Näherung kann man die Luftdichte in einer Druckschicht als konstant betrachten, keinesfalls aber den vertikalen Dichtegradienten (barometrische Höhenformel!) ignorieren.

Als größtmöglichen Ramen betrachte ich folgend Ensembles über ganze Luftschichten oder gar über die ganze (mit Daten versehene) Atmosphäre verteilt. Spätere Untersuchungen werden höchstwahrscheinlich in lokalerem Rahmen durchgeführt (begrenzte Startgebiete), z.B. um höhere Auflösung und verlässlichere Massenvergleiche zu erzielen. Die einmal erarbeitete globale Sicht liefert eine Basis, die gegebenenfalls vereinfacht werden kann. Zudem führe ich tatsächlich zuerst globale Läufe mit über die gesamte Atmosphäre verteilten Ensembles im Rahmen der Speicherplatzgrenzen durch, um globales Transportverhalten zu bewerten; als grundlegende Verifikation der Funktionalität der Methode. Die globale Entwicklung der Massenverteilung durch den modellierten Transport ermöglicht auch einen Vergleich mit der Dichte, die das ECHO-GiSP Modell über zusätzliche Variablen zur Verfügung stellt. Idealerweise sollten das Klimamodell und die Lagrangesche Transportrechnung bei der Verteilung von Luftmasse übereinstimmen – in der Realität bedeutet das, dass sie in einem vertretbaren Verhältnis stehen sollten, das eine gewisse Zeit Bestand hat.

4.1.1 Globale Startpositionierung in der Ebene (dh)

In der Horizontalen ist die Gleichverteilung von Startpositionen auf der Fläche ein sinnvoller Ansatz. Unter Vernachlässigung der Dichtevariation in der Ebene ist damit die Gleichwertigkeit der Partikel sichergestellt. Die Zahlenverhältnisse der Partikel in einem Gebiet spiegeln die Verhältnisse von Luftmassen wider. Auch wenn die Dichtevariation einbezogen wird, stellt die gleichmäßige Festlegung von Startpositionen entsprechend einer mittleren Dichte eine gute Ausgangslage dar.

[1] Der Begriff des Tracers bezieht somit auf eine spezifische Luftmasse, die zum Startzeitpunkt am Ort (bzw. in dessen Umgebung, siehe Abs. 5.2

Eine einfache deterministische Positionierung in der Ebene ergibt sich aus der Festlegung von Startpositionen auf Breitenkreisen, wobei die Anzahl der Positionen mit dem Umfang des Breitenkreises skaliert. Der eindeutig bezeichnende Parameter dieser Positionierung ist die gerade Zahl h, die die Anzahl der Startpositionen auf dem Äquator vorgibt. Ich verwende für die ebene Startpositionierung zum Parameter h das Kürzel dh. So meint "d180" die Positionierung mit 180 Positionen auf dem Äquator und "d1000" die mit deren 1000. Zwei Formeln für β und λ bestimmen die einzelnen Positionen:

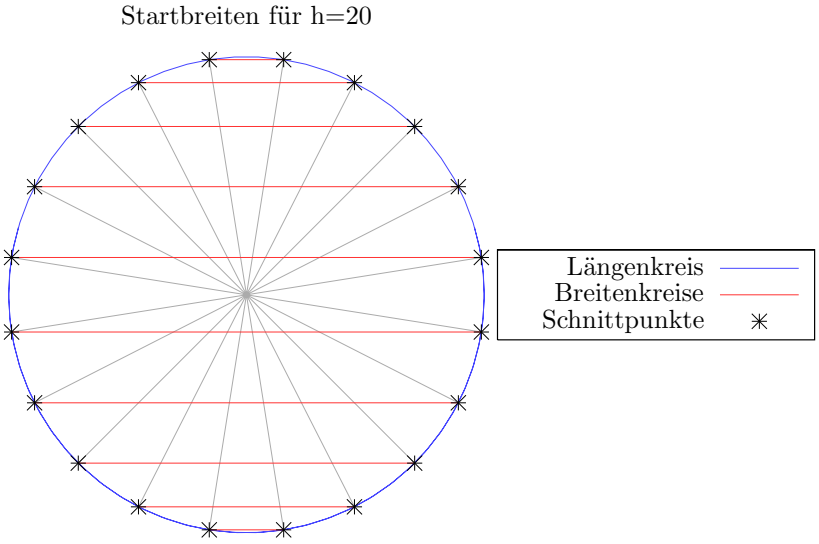


Abbildung 4.1: Gleichmäßige Teilung der Breiten

$$\begin{aligned} \beta_a &= \pi\left[\left(a+\frac{1}{2}\right)\frac{2}{h}-\frac{1}{2}\right];\ a\in\left[0;\frac{h}{2}-1\right] \\ \lambda_{a,o} &= o\cdot\frac{2\pi}{\text{ganz}\,(h\cdot cos\beta_a)}-\pi;\ o\in[0;\text{ganz}\,(h\cos\beta_a)-1] \end{aligned} \tag{4.1}$$

($\lambda\in[-\pi;\pi)$ bzw. $\lambda\in[-180°;180°)$, h gerade Zahl)

Ich verteile hier $\frac{h}{2}$ Breitenkreise im Abstand von $\frac{2\pi}{h}$ voneinander, im Intervall

$$\left[-\frac{\pi}{2}+\frac{\pi}{2h};\frac{\pi}{2}-\frac{\pi}{2h}\right]$$

Das bedeutet, dass die Verlängerung eines Meridians zum Vollkreis mit diesen Breitenkreisen h Schnittpunkte im gleichen Abstand $R\frac{2\pi}{h}$ (in Metern) voneinander hat – auch über die Pole hinweg, wie Abb. 4.1 verdeutlicht. Auf diesen Breitenkreisen werden dann Positionen λ_o vergeben, im Abstand von

$$r_\text{b}(\beta_a)\frac{2\pi}{\text{ganz}\,(h\cdot cos\beta_a)}=R\cos\beta_a\frac{2\pi}{\text{ganz}\,(h\cdot cos\beta_a)}$$

zueinander. Für große h, wo $\text{ganz}\,(h\cdot cos\beta_a)\approx h\cdot cos\beta_a$, ist dieser Abstand der Startpositionen gleich dem in Breitenrichtung:

$$R\cos\beta_a\frac{2\pi}{\text{ganz}\,(h\cdot cos\beta_a)}\overset{h>>1}{\approx}R\cos\beta_a\frac{2\pi}{h\cdot cos\beta_a}=R\frac{2\pi}{h} \tag{4.2}$$

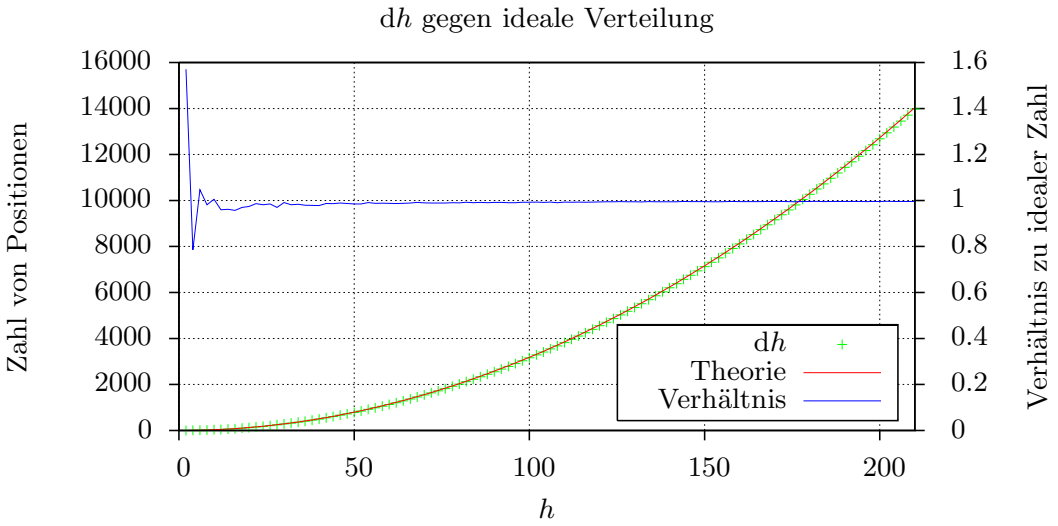


Abbildung 4.2: Vergleich zwischen Zahl der Positionen von dh mit $\frac{h^2}{\pi}$

Diese Positionierung gewährleistet also für nicht zu kleine h auf dem gesamten Globus Startpositionen mit gleichem Abstand – in Metern – zum nächsten Nachbarn. Anders gesagt: Die Positionen sind auf der Kugeloberfläche gleichmäßig verteilt und ich starte mit einer weitgehend konstanten Flächendichte von Partikeln.

Der Wert von h ist dabei ein direktes Maß für die Flächendichte. Er ist definiert als die eindimensionale Dichte von Startpositionen auf dem Äquator, also h Positionen auf einem Erdumfang, was die eindimensionale Dichte $h\,(2\pi R)^{-1}$ bewirkt. Die Positionierung ist so konstruiert, dass in Breitenrichtung dieselbe eindimensionale Dichte herrscht, so dass die Flächendichte für hinreichend große h durch $h^2\,(2\pi R)^{-2}$ gegeben ist. Eine einfacher Test der Gleichmäßigkeit der dh Positionierung ergibt sich durch den Vergleich der tatsächlich auf der Oberfläche verteilten Partikel mit dem erwarteten Wert bei konstanter Dichte: $h^2\,(2\pi R)^{-2} \cdot 4\pi R^2 = h^2\pi^{-1}$. Abb. 4.2 zeigt angemessene Übereinstimmung[2].

4.1.2 Startpositionierung in gesamter Atmosphäre (dh-v)

Vertikal müsste man entsprechend dem exponentiellem Abfall der Luftdichte mit der Höhe ebenfalls die Dichte der Startpositionen exponentiell abnehmen lassen. Dies ist aber nicht praktikabel – weil man entweder in den höheren Luftschichten gar keine oder zu wenige Partikel hat, um nennenswert die Dynamik zu erschließen, oder in den unteren Schichten einfach zu viele Partikel berechnen müsste.

Mein praktizierter Ansatz für eine globale Startpositionierung basiert auf den vorgegebenen Druckebenen aus ECHO-GiSP und gemäß einem Parameter v generierten Zwischenebenen. dh-v meint die Verteilung von v Startebenen der Positionierung dh für jede der n_p Druckebenen aus den Modelldaten (jeodch nicht über die oberste hinaus):

$$p_{n_p,1} = p_{n_p};\ \forall k \in [1, n_p - 1]:\ \forall i \in [1; v]:\ p_{k,i} = p_k + (i-1)\frac{p_{k+1} - p_k}{v}$$

[2]Genaue Übereinstimmung ist nicht möglich, da ich keine *Bruchteile* sondern nur ganze Positionen setze.

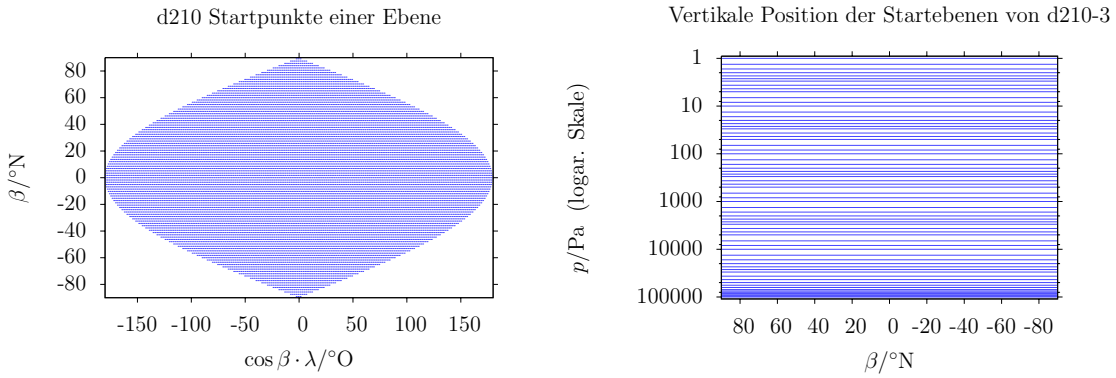


Abbildung 4.3: Blick auf horizontale und vertikale Positionierung von d210-3
Je nach vorliegender Druckqualität sollte die horizontale Startpositionierung links in Form von vielen gleichmäßig verteilten Punkten oder als eine Farbfläche erscheinen.

Diese Platzierung ist enstanden als schnelle Erweiterung der einfachsten Legens einer Startebene auf jeder gegebenen Modellebene. In Zukunft wird sie wahrscheinlich von einer durchgängig logarithmischen Platzierung der Startbenen ersetzt werden.

Die Gesamtpositionierung aus $n_p \cdot v$ Ebenen nenne ich dh-v. Im nächsten Kapitel komme ich zur Auswertung von d210-3 und d1000-3 Läufen, also jeweils 3 Startebenen pro Modellebene und 210 Partikel pro Großkreis bzw. 1000 Partikel pro Großkreis. Einen Eindruck von d210-3 soll Abb. 4.3 geben.

4.2 10 Tage Transport

Jetzt ist geklärt *wovon wie was* transportiert werden soll. *Wovon:* Ich habe den Zugriff auf die Windfelder aus ECHO-GiSP über die Schnittstelle von `egp_wind`. *Wie:* Dazu wurde eine Methode zur Integration von Trajektorien entwickelt, kumuliert im Programm `ensemble_run` , welches im Wesentlichen dem in Quelltext 3.7 gezeigten Ablauf folgt. *Was:* Mit dh-v ist eine Systematik zum Erstellen von globalen und lokalen Partikelensembles mit homogener horizontaler Dichte in einer Druckschicht[3] gegeben.

Zur Demonstration, zeige ich nun Ansichten von einigen Testläufen[4], die ich auf meinem Laptop Neuling erstelle — just, während ich dies hier schreibe. Für die gewählten Teilchenzahlen (bis 8744) und die Anzahl von Integrationsintervallen (200 bei Schrittlänge von 0,05 Tagen und zehn Tagen Gesamtdauer) ist das durchaus praktikabel, die Rechenzeit beträgt dabei wenige Minuten[5].

Zuerst lenke ich den Blick auf die Entwicklung von zwei globalen d160-Positionierungen auf jeweils einer Druckschicht in der Troposphäre (850hPa, Abb. 4.4) und einer in der Stratosphäre (10hPa, Abb. 4.5). Die Schnappschüsse in horizontaler Sicht zeigen zumindest ansatzweise die unterschiedliche Charakteristik der Dynamik in diesen Luftschichten. Die Bewegung

[3] Das lässt außer Acht, dass die Luftdichte in einer Druckschicht nicht wirklich konstant ist — im nächsten Kapitel komme ich darauf zurück.
[4] Die Rechnungen erfolgen für die ersten Tage des Modelljahres 1965, mit Winddaten vom interaktiven Lauf.
[5] O-Ton `ensemble_run`: “Statistics: 8110 trajectories of 200 steps in 157 seconds (0.0193588s per trajectory, 10331.2ss/s)”, also unter drei Minuten für 8110 Trajektiorien zu 200 Zeitschritten.

aus 10hPa ist geprägt vom polaren Vortex, während aus 850hPa eher kleinskalige Mischung erkennbar ist. Allerdings ist bei beiden die zonale Ausprägung der Winde und somit der Äquator als Transportbarriere zumindest in der Zeitskale bis zu zehn Tagen überdeutlich.

Dem konvektiven Auftrieb am Äquator widmet sich Abb. 4.6, wo ein Auschnitt aus einem d400 Ensemble in 850hPa zwischen $-10°$N und $10°$N Breite transportiert wird. Es ist deutlich zu sehen, wie Partikel zur Tropopause hochgetrieben werden.

Vorerst abgeschlossen werden die Einsichten durch zwei in der Horizontalen lokale, sich aber über die Höhe von 1000hPa bis zu 0.01 erstreckende Ensembles, jeweils im Umkreis von $5°$N um den Nord- bzw Südpol in Abb. 4.7 und Abb. 4.8. In den Bildern wird die Einmischung von polarer Luft in niedrigere Breiten ohne Unterscheidung nach Herkunftshöhe gezeigt. Die unterschiedlichen Situationen auf der Nord- und der Südseite sind zu erkennen. Vom Südpol aus erfolgt die Verteilung in Form eines weitgehend symmetrischen Wirbels, während die Asymmetrie im Norden sich von Anfang an klar zeigt.

Dies sind einige qualitative Eindrücke, die hauptsächlich belegen, dass die von mir umgesetzte Lagrangesche Advektion so weit funktioniert. Es werden bekannte Transportstrukturen der Atmosphäre reproduziert — ich bin also auf einem richtigen Weg.

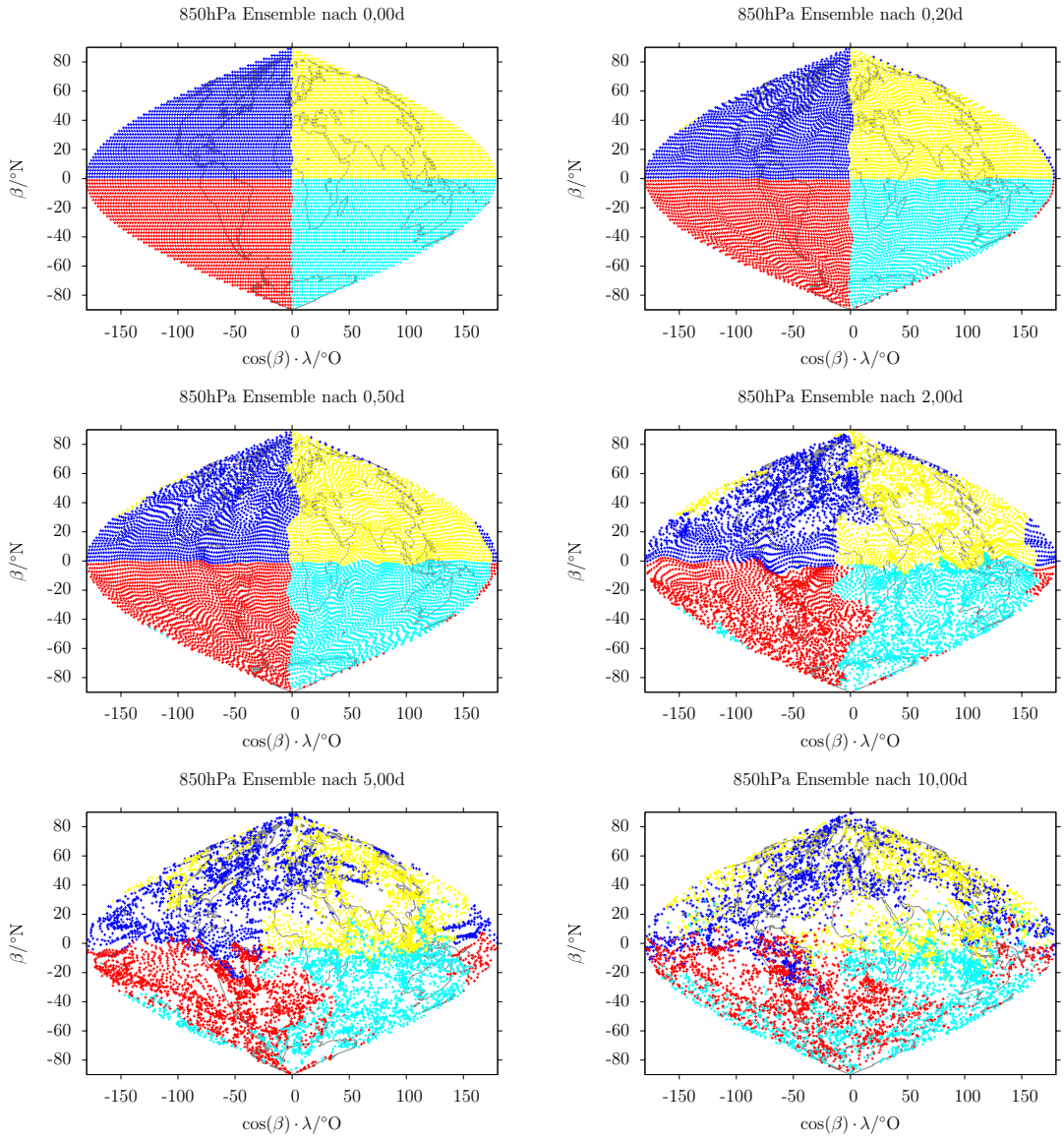


Abbildung 4.4: 10 Tage Transport von d160 in 850hPa
Jedes Partikel des Ensembles (8110 insgesamt) ist durch ein farbiges Symbol gekennzeichnet. Die Farbe des Symbols bezeugt den Herkunftsquadranten (Nordwest, Nordost, Südwest und Südost) des Partikels. Diese farbliche Aufteilung ermöglicht eine einfache aber wirkungsvolle Darstellung der Mischungsdynamik in Nord-Süd und Ost-West Richtung.
Beachte: Die Angabe 850hPa bezieht sich nur auf die *Start*ebene des Ensembles — der Transport erfolgt in drei Dimensionen; somit ist dies hier keine Mischung von Partikeln *in* 850hPa, sondern *aus* 850hPa.

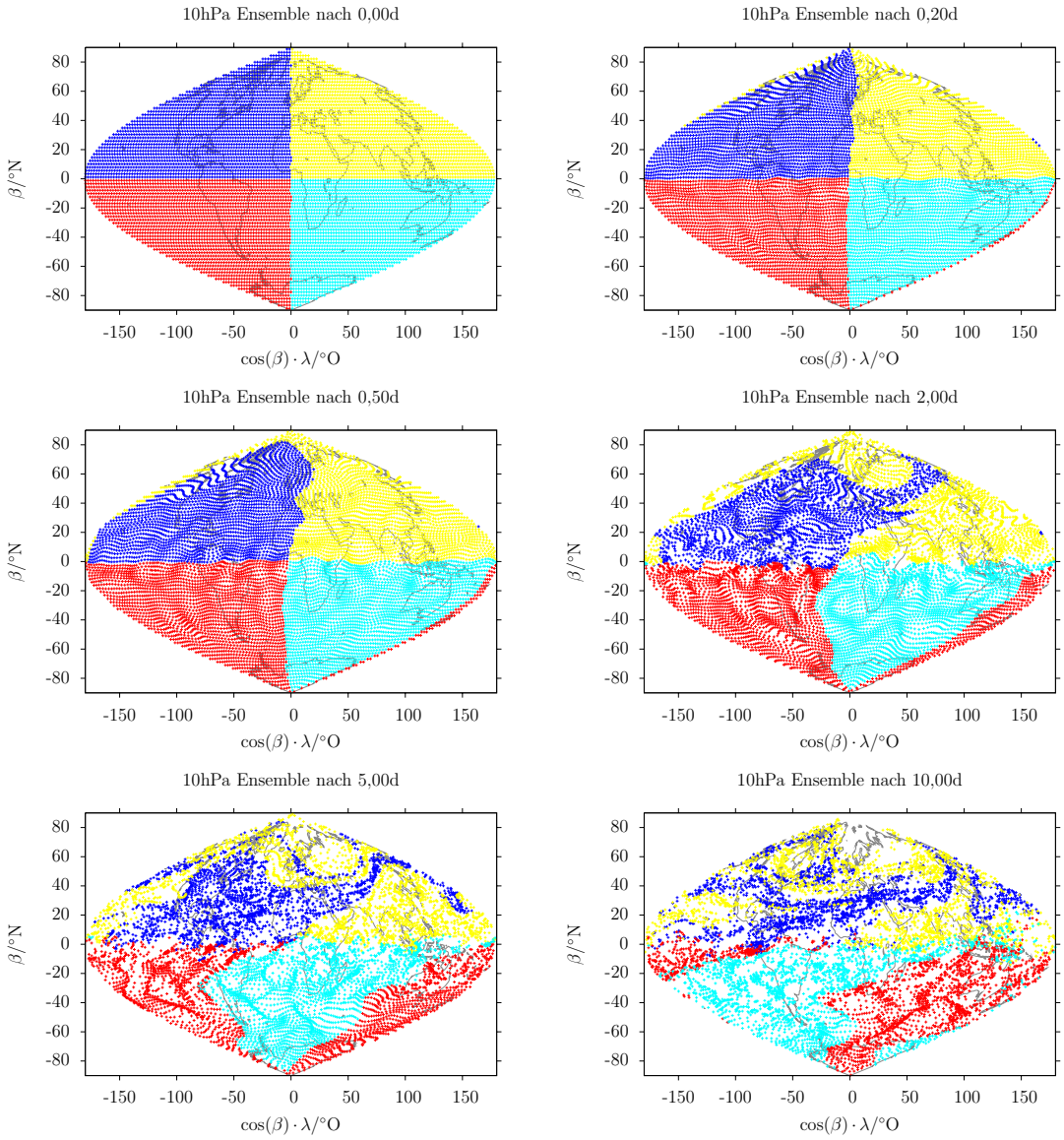


Abbildung 4.5: 10 Tage Transport von d160 in 10hPa
Jedes Partikel des Ensembles (8110 insgesamt) ist durch ein farbiges Symbol gekennzeichnet. Die Farbe des Symbols bezeugt den Herkunftsquadranten (Nordwest, Nordost, Südwest und Südost) des Partikels. Diese farbliche Aufteilung ermöglicht eine einfache aber wirkungsvolle Darstellung der Mischungsdynamik in Nord-Süd und Ost-West Richtung.
Beachte: Die Angabe 10hPa bezieht sich nur auf die *Start*ebene des Ensembles — der Transport erfolgt in drei Dimensionen; somit ist dies hier keine Mischung von Partikeln *in* 10hPa, sondern *aus* 10hPa. Die nächste Abbildung, Abb. 4.6, belegt äußerst klar das Vorhandensein von vertikalem Transport.

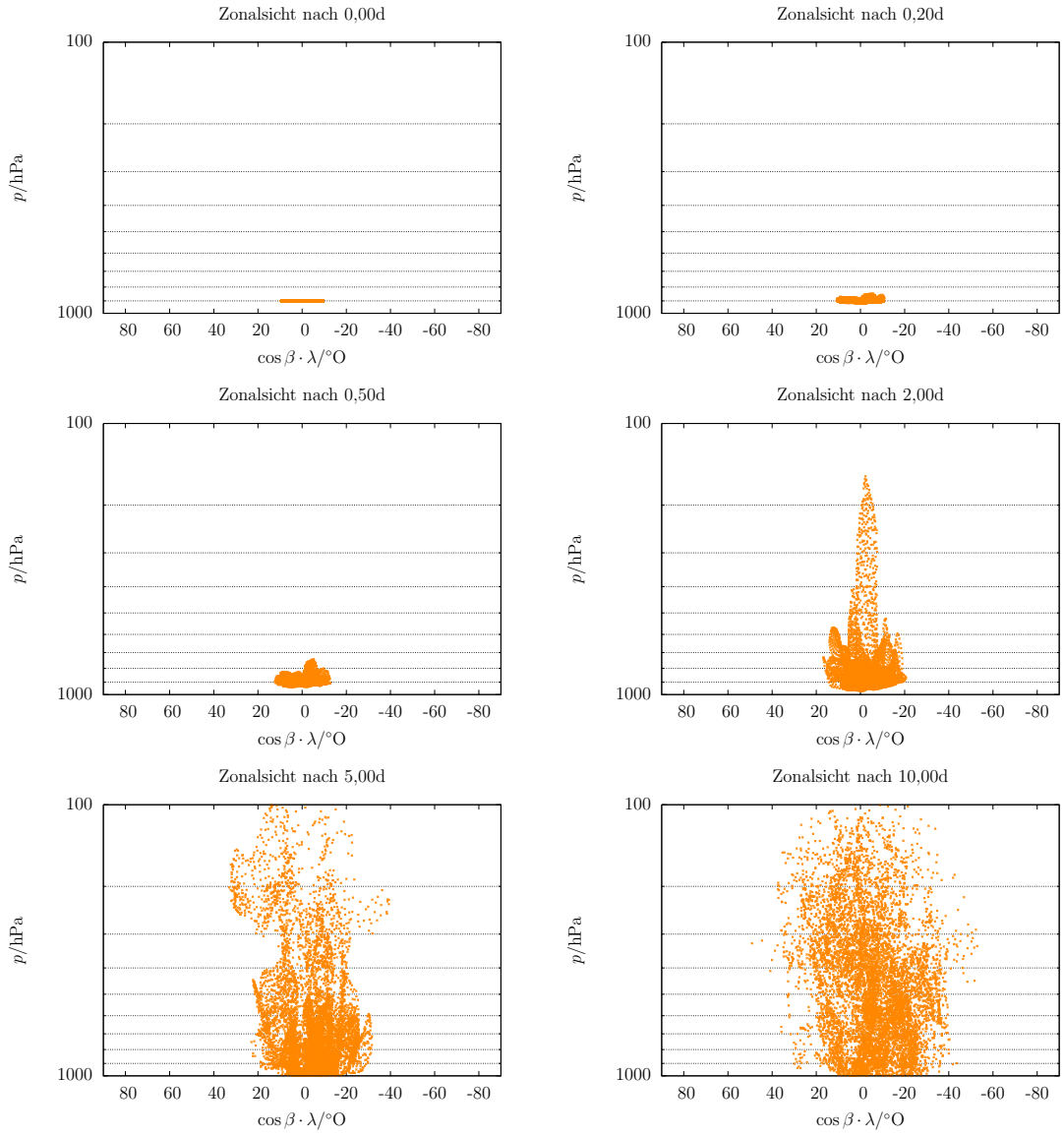


Abbildung 4.6: Konvektiver Auftrieb in den Tropen
Das Ensemble aus 8744 Partikeln startet bei $p = 850\text{hPa}$ und $\beta \in [-10; 10]$ (über alle Längen verteilt) und unterliegt dem konvektiven Auftrieb der zentralen Tropen.

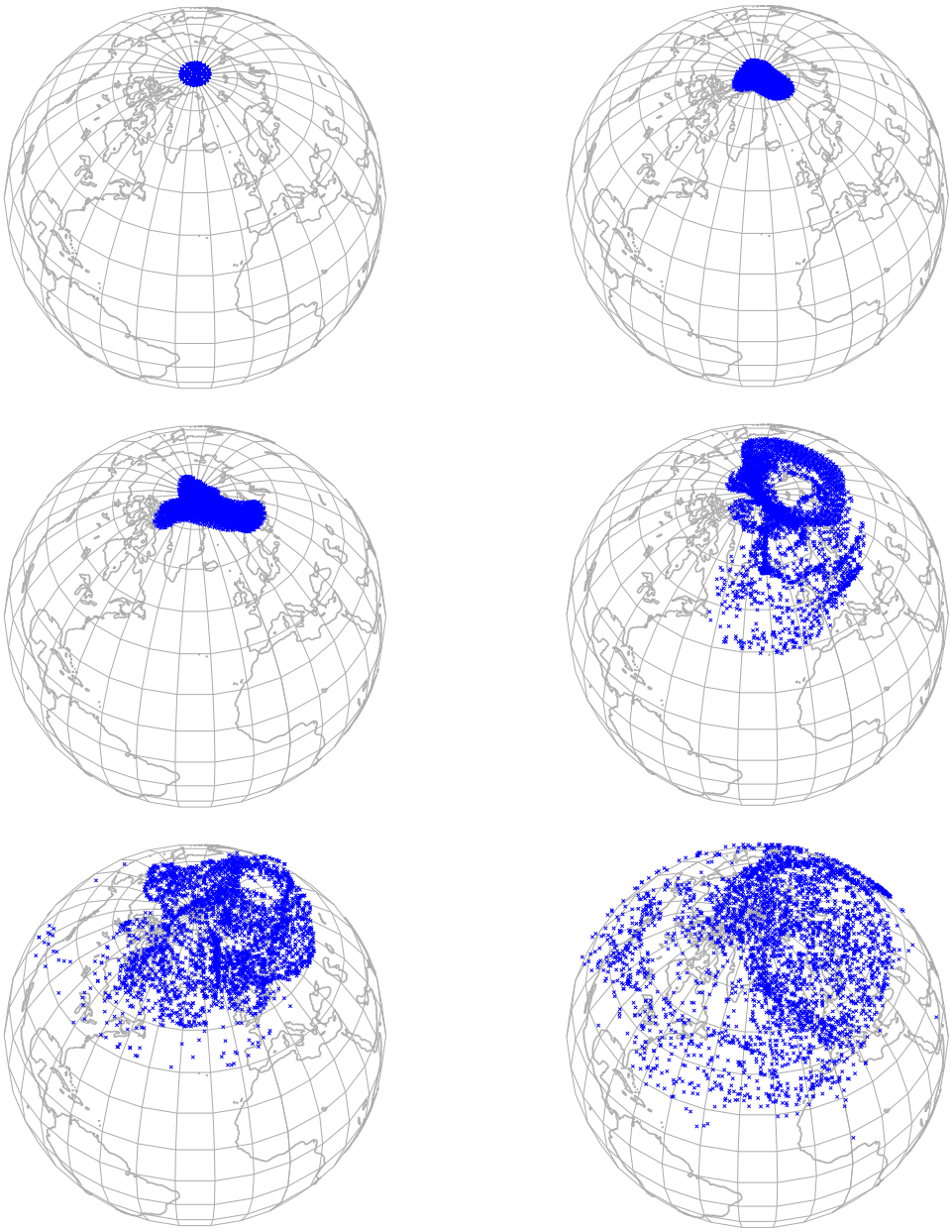


Abbildung 4.7: Verteilung vom Nordpol in 10 Tagen
3216 Partikel aus allen Schichten des Klimamodells erfahren Verbreitung aus der nächsten Umgebung des Nordpols. Die zeitliche Reihenfolge ist wie zuvor, von links nach rechts, von oben nach unten: Anfang, nach 0,2, 0,5, 2, 5 und 10 Tagen

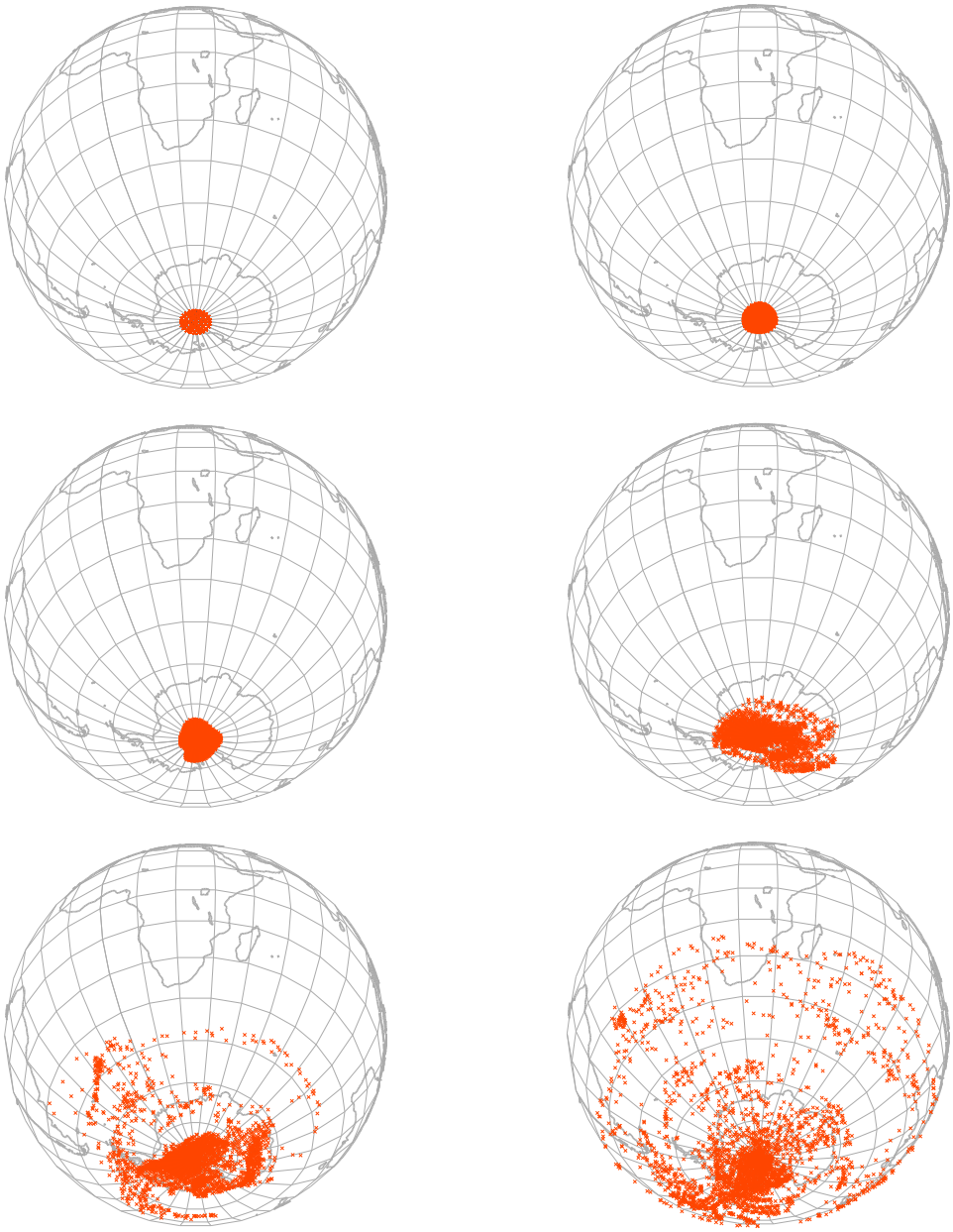


Abbildung 4.8: Verteilung vom Südpol in 10 Tagen
3216 Partikel aus allen Schichten des Klimamodells erfahren Verbreitung aus der nächsten Umgebung des Südpols. Die zeitliche Reihenfolge ist wie zuvor, von links nach rechts, von oben nach unten: Anfang, nach 0,2, 0,5, 2, 5 und 10 Tagen

5 Massenkonsistenz mit dem Klimamodell

5.1 Erste globale Läufe bis zu 80 Jahren

Das langfristige Ziel meiner Arbeit ist die Analyse von langfristigen Transportstrukturen — Barrieren, Mischungsraten, Fernkonnektion. Ein notwendiger Schritt in diese Richtung sind komplette globale Rechnungen mit niedriger (machbarer) Auflösung in Raum und Zeit. Bevor Detailfragen bearbeitet werden, will ich das globale Bild geklärt wissen. Ich habe aus den mir verfügbaren Jahren den 80 Jahre umfassenden Zeitraum 1965 bis 2044 für dieses globale, langzeitliche Bild gewählt. Die ersten 15 Jahre (1950 bis 1964) lasse ich als transitive Phase[1] des Modells außen vor. Es gibt nun insgesamt vier globale Transportläufe, bestehend aus

- 80 Jahre mit d210-3 (ca. $1 \cdot 10^6$ Partikel) mit Zeitschritt $0,1$d und Speicherung der Positionen jeden Monat[2]
- 1 Jahr mit d1000-3 (ca. $21 \cdot 10^6$ Partikel), ebenfalls mit Zeitschritt $0,1$d, allerdings mit Speicherung jeden Tag

jeweils einmal getrieben von den Windfeldern des Referenzlaufes von ECHO-GiSP und einmal von denen des interaktiv-gekoppelten Laufes. Dies sind wirklich recht grobe Übersichtsläufe als Anfang, so habe ich vor, zukünftige Rechnungen auch mit kleineren Zeitschritten durchzuführen — ein zehntel Tag ist noch recht grob.

Die Läufe mit d1000-3 wurden erst durch die Anschaffung des RAID-Servers Atlas möglich, welcher den nötigen Speicherplatz für ein Jahr d1000-3 mit täglicher Speicherung zur Verfügung stellt. Dieser beläuft sich für einen Lauf auf 356GiB. Tatsächlich wirkt diese Zahl nicht mehr so erschreckend wie noch vor wenigen Jahren. Ein Zeichen der Zeit ist auch, dass wir diese Datenmengen heutzutage auf relativ preisgünstigen und schnellen RAID-Verbunden von ATA-Festplatten nutzen können; es besteht nicht mehr der Zwang zu einem aufwändigen und auch wesentlich langsameren Bandarchivsystem[3].

Sind die Ensembles berechnet, stellt die Frage einer sinnvollen Visualisierung. Der erste Ansatz sind Abbildungen der Punkte in einer Projektion wie in Abb. 4.5 oder in der zonalen Sicht wie Abb. 4.6. Die Färbung von Punkten nach ihrer Herkunft kann einen Eindruck von der Durchmischung liefern. Allerdings ist das gerade in der bedeutsamen zonalen Sicht (vertikaler und in Nord-Süd-Richtung stattfindender Transport) sehr problematisch. Zwei Aspekte verbieten quantitative Aussagen:

[1] der Zeitraum vom Start, den das Modell sozusagen zum "Einschwingen" benötigt, bis es einen verwertbaren Zustand erreicht

[2] also alle 30 Tage, entsprechend den 360 Modelltagen pro Jahr

[3] Dieses mag als Langzeitarchiv benötigt werden, aber bei der mittelfristigen Speicherung der Daten, vor allem während mit ihnen gearbeitet wird, sind Festplatten im Vorteil.

- Partikel aus unterschiedlichen Herkunftsgebieten, besonders aus unterschiedlichen Höhen, repräsentieren unterschiedliche Luftmassen. Ein Punkt ist aber ein Punkt — da ist kein Spielraum für relative Gewichtung. Ohne einen Ausgleich erscheinen die oberen Luftschichten schnell entvölkert, da viele — leichte — Partikel in die tieferen Schichten vordringen und wenige — aber schwere — Partikel von dort in die höheren.
- Wenn alle Punkte auf einem Breitengrad einfach aufsummiert werden, entsteht der Eindruck, dass Luftmasse von den Polen zum Äquator hin konzentriert wird. Besser ist eine Ansicht, die korrekt über die Längengrade mittelt, um die Dichte wiederzuspiegeln. Für die selbe Massendichte müssen sich in den Tropen mehr Luftpartikel aufhalten als an den Polen, da sie auch ein größeres Volumen zu füllen haben. Sinnvoll ist letzlich die Darstellung einer Dichte und nicht die der absoluten Massenkonzentration.

Eine historische Darstellung (Abb. 5.1), die ich ähnlich vor einem Jahr bei einem Arbeitstreffen als erstes Ergebnis der dreidimensionalen Integration präsentierte, zeigt beide Probleme klar: Die Atmosphäre ballt sich über den Tropen zusammen.

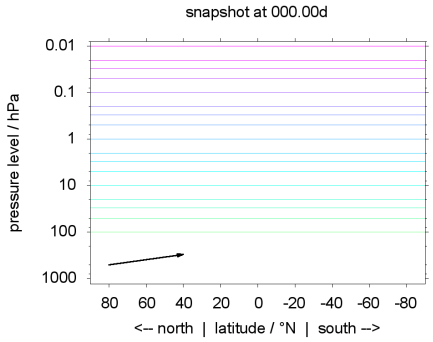
Ich sehe, dass für eine sinnvolle Darstellung etwas mehr erforderlich ist, als ohne Bearbeitung viele bunte Punkte auf einer Bildfläche zu verteilen. Schlussendlich möchte ich aber nicht nur Darstellungen anfertigen, sondern die Daten quantitativ *auswerten*. Ich berechne die große Anzahl Trajektorien, um aus ihr statistisch Maße zu errechnen. Das Maß, welches gerade vordergründig fehlt, ist das Gewicht eines Partikels, eines Teilensembles — die repräsentierte Luftmasse. Wenn ich Partikeln bzw. Partikelgruppen Luftmassen zugewiesen habe, dann kann ich erst aussagekräftige Zahlen zu Mischunsprozessen errechnen. Die Berechnung der repräsentierten Luftmasse und der damit ermöglichte Konsistenzvergleich mir dem Klimamodell sind die Themen, womit ich diese Diplomarbeit schließen werde. Dabei werde ich hier nicht weiter auf den 80 Jahre umfassenden d210-3 – Lauf eingehen und mich stattdessen auf die bessere Auflösung des d1000-3 – Laufes konzentrieren. Beide Läufe sind allerdings in [Orgis2007] eingegangen.

5.2 Quantifizierung: Partition & Gewicht

Wenn ich Mischungsverhältnisse oder Transportraten aus den vielen Einzeltrajektorien berechnen will, benötige ich eine Definiton von Start- und Zielzonen[4]. Ich muss die Atmosphäre unterteilen. In den definierten Teilgebieten kann ich dann aus der Anzahl von Partikeln, die dort hintransportiert wurden, Maße berechnen. Ich kann Fragen nach der Verbreitung von Luft aus einem gewissen Startgebiet in einer bestimmten Zeit behandeln, oder auch die Umkehrfrage: Aus welchen Startgebiet kommt welcher Anteil der Luftmasse in einem Zielgebiet?

Um die von meinen Partikeln repräsentierten Luftmassen aus verschiedenen Zonen überhaupt quantitativ vergleichen zu können, muss ich ihnen ein Gewicht zuordnen. Allerdings reicht selbst eine (praktisch auch nicht vorhandene) genaue Dichte am Startpunkt nicht aus, da es nicht praktikabel ist, die Startpositionen anhand der Luftdichte in der Vertikalen zu verteilen. Wie schon zuvor angemerkt und später in Abb. 5.7 verbildlicht, verhindert der exponentielle Dichtegradient in dem Fall eine hinreichende Abdeckung der Atmosphäre in den oberen

[4]Ich verwende hier einen allgemeinen Begriff der Zone als ein beliebiges Teilgebiet der Atmosphäre in 3D bzw. in 2D der Erdoberfläche, also nicht beschränkt auf festgelegte Gebiete wie subtropische oder polare (Klima)Zone.



Die Partikel wurden auf den ausgezeichneten Ebenen gestartet; die Färbung eines Punkes verrrät die Herkunftshöhe.

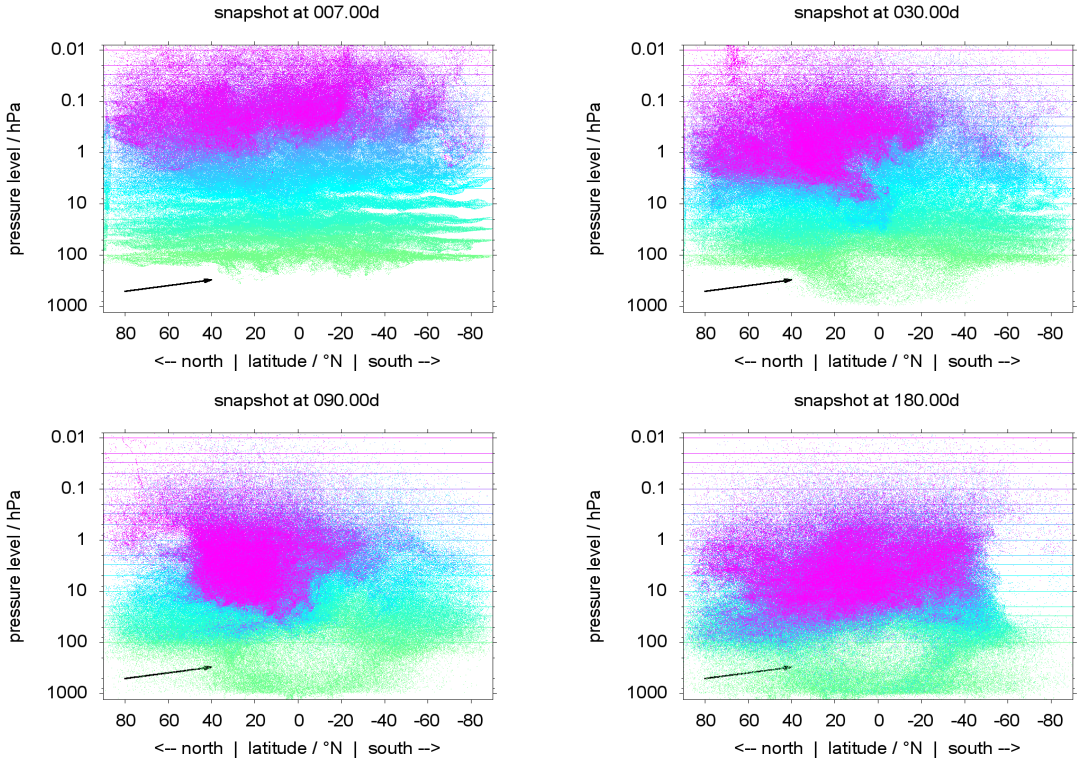


Abbildung 5.1: Historischer Blick auf 180 Tage eines frühen 3D Transportlaufes
...mit Hinweis auf stratosphärische Intrusion (Pfeil).
Jedes Partikel im Ensemble ist als ein Punkt dargestellt, ohne irgend welche Unterschiede in der repräsentierten Masse zu beachten. Desweiteren sind in dieser zonalen Sicht alle Partikel auf einer Breite zusammen dargestellt. Naturgemäß bedingt das die visuelle Konzentration zum Äquator — zusammen mit der Kontentration auf untere Luftschichten wegen dem fehlenden Gewichtsausgleich.

Schichten bzw. erfordert zu viele Partikel in den unteren Schichten. Einen praktikablen Ansatz stellt die Zuordnung eines Gewichtes durch Aufteilung der in einer Zone enthaltenen Masse auf die enthaltenen Startpositionen dar.

Ich investiere etwas Zeit in die Frage nach der zu verwendenten Aufteilung der Atmosphäre. Interessant ist dabei hauptsächlich die Kugeloberfläche[5] mit den inhomogenen Koordinaten. Vertikal ist einer nach dem Druck logarithmischen Teilung in Schichten pragmatisch wenig entgegen zu setzen, horizontal jedoch ist die Sache nicht so klar, wenn man ein paar Bedingungen in Betracht zieht.

Die grundlegende Bedingung für mich ist, die Betrachtung von Masse/Dichte und Mischung von der systematische Verengung der Koordinaten zu den Polen hin zu trennen. Wenn ich einfach Zellen eines Gitters in (λ, β) mit festen Gitterpunktabständen $\Delta\lambda$ und $\Delta\beta$ zur Einteilung der Oberfläche nutze (siehe Abb. 5.2), ist die Fläche einer Zelle in Polnähe viel kleiner als die einer in Äquatornähe[6] Dies ist problematisch, weil dies einen *systematischen* Unterschied in der Unsicherheit der Rechnung von gezählten Partikeln im Gebiet zu Luftmasse bedeutet. Eine aus den Partikelensemble berechnete Luftdichte in einer polnahen Gitterzelle beruht auf einer wesentlich kleineren Zahl von Partikeln in dem kleineren Gebiet. Es nicht sinnvoll, mit solch unterschiedlicher Basis einen statistischen Vergleich (wie Berechnung einer Massendichte aufgrund der Partikelzahl) anzustrengen. Das Sprichwort prangert den Vergleich von Äpfeln mit Birnen an — der Vergleich der Dichte aus verschiedenen Zellen des einfachen Gitters geht mehr in die Richtung, Äpfel mit Erbsen zu vergleichen!

Der Unterschied zwischen Äpfeln und Erbsen liegt hauptsächlich in der Größe. Birnen dagegen sind anders geformt und schmecken auch anders[7] als Äpfel, sind aber von der Größe her vergleichbar und somit ein deutlicher Forschritt, verglichen mit den Erbsen. Ich kann die underschiedlich geformten und sehr unterschiedlich großen Zonen durch eine flexiblere Aufteilung mit immernoch unterschiedlich geformten, aber dafür gleich großen Zonen ersetzen. Die Formunterschiede vernachlässigend, kann ich folgend beruhigt Äpfel mit Birnen vergleichen, die so unterschiedlich ja gar nicht sind.

Das Ziel ist also eine Aufteilung der Erdoberfläche in gleich große Zonen. Ich werde nun zwei Ansätze dazu präsentieren, von denen letztlich der zweite von mir als Optimum verwendet wird. Nachdem die Aufteilung der Atmosphäre in Zonen aus der Oberflächenpartition und logarithmischer Aufteilung der (Druck)Höhe etabliert ist, wende ich mich der Frage der Zuteilung einer Luftmasse an diese Zonen zu.

5.2.1 Einbettbare Äquipartition

Eine naheliegende Möglichkeit, die Aufteilung in der Fläche gleichmäßig zu gestalten, besteht in der Variation der Breitenteilung. Anstatt des festen Intervalls $\Delta\beta$ wird der Breitenbereich der Teilung variiert — zu den Polen hin wird der von einer Zone erfasste Breitenbereich größer, um die Fläche auszugleichen.

[5] Ja, *idealisiert* als Kugeloberfläche.

[6] Entsprechend der Abnahme des Breitenkreisumfangs mit $\cos\beta$ nimmt die Oberfläche zwischen zwei Breitenkreisen ab. Bei $\Delta\beta = 10°\mathrm{N}$ ist die Fläche einer Zelle mit $\beta \in [0°\mathrm{N}; 10°\mathrm{N}]$ $11,4$ mal so groß wie die einer Zelle mit $\beta \in [80°\mathrm{N}; 90°\mathrm{N}]$.

[7] ☺ Besser? Das mag ich hier nicht entscheiden...

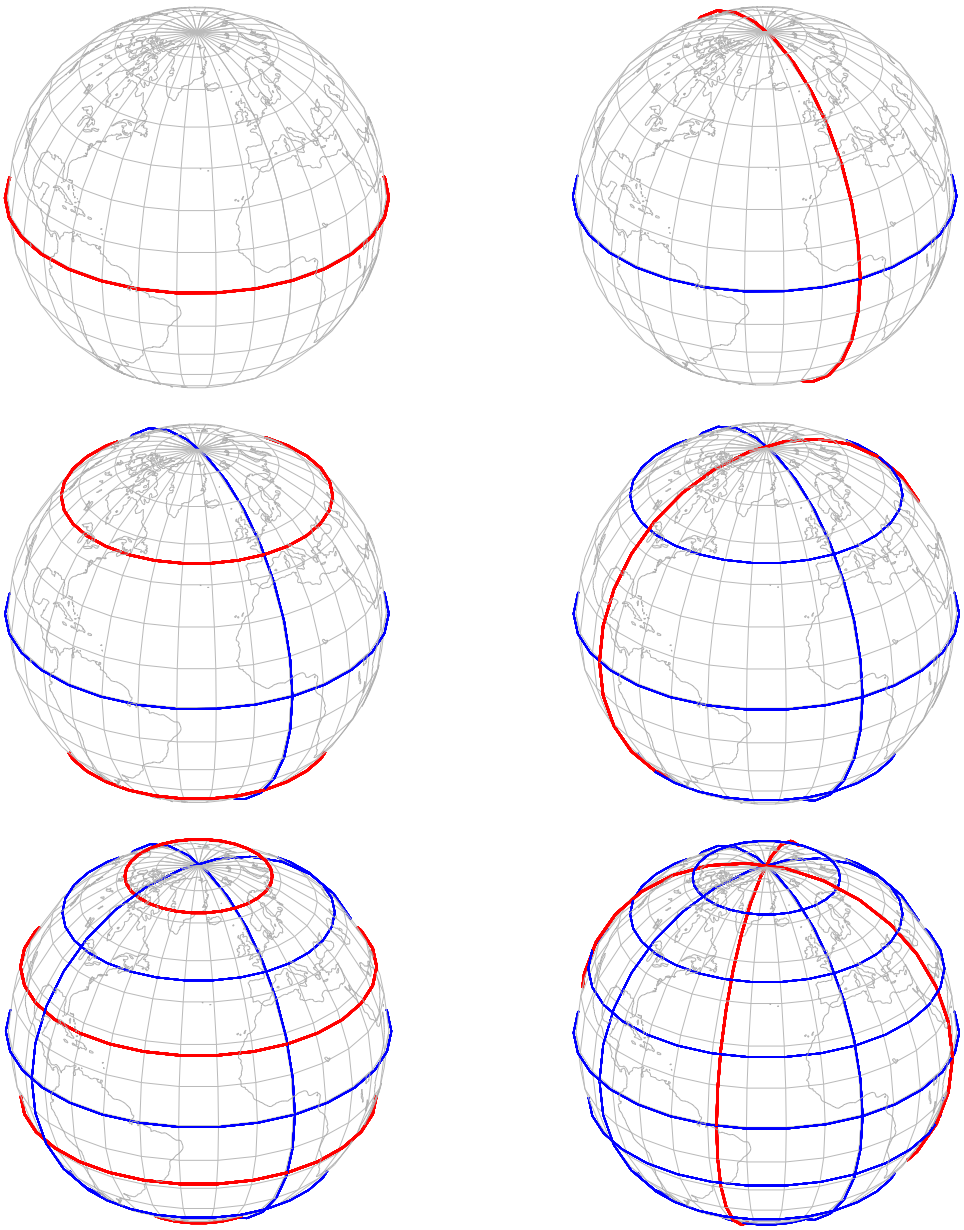


Abbildung 5.2: Die Aufteilung der Erdoberfläche mit festem Gitter in ungleiche Gebiete ...mit Anzahlverdopplung zu 2, 4, 8, 16, 32 und 64 Gebieten. Teilzonen an den Polen sind erheblich kleiner als in Äquatornähe (ab dem dritten Bild).

Auf der Kugel ist die Fläche einer Zone mit $\lambda \in [\lambda_0; \lambda_1]$ und $\beta \in [\beta_0; \beta_1]$ gegeben durch

$$\begin{aligned} A_{\text{real}} &= \int_{\beta_0}^{\beta_1} (\lambda_1 - \lambda_0) R \cos\beta \cdot R \mathrm{d}\beta \\ &= (\lambda_1 - \lambda_0) R^2 \sin\beta \Big|_{\beta_0}^{\beta_1} \end{aligned} \tag{5.1}$$

R ist eine globale Konstante und $(\lambda_1 - \lambda_0)$ ist bei festem Intervall in der Länge, wie ich es hier betrachte, für alle Zonen gleich. Somit kann ich auf die Fläche einer Halbkugel (bzw. eines Segmentes dieser zwischen λ_0 und λ_1) normieren[8] und einfach

$$A(\beta_0, \beta_1) = \sin\beta_1 - \sin\beta_0 \tag{5.2}$$

betrachten. Die Fläche folgt also einfach dem Verlauf des Sinus zwischen β_0 und β_1.

Nach dieser Erkenntnis ist die Aufteilung in Breitenbereiche mit gleicher Fläche auf ein Festes Inkrement im Sinus zurück zu führen. Wenn ich den Bereich zwischen B_0 und B_1 ($B_0 < B_1$) in a Breitenbereiche teilen möchte, sind die Grenzen $\beta_i;\ i \in [0; a]$ gegenen durch

$$\beta_i = \operatorname{asin}\left(\sin B_0 + \frac{i}{a}(\sin B_1 - \sin B_0)\right) \tag{5.3}$$

insbesondere ergibt sich für die gesamte Kugel mit $B_0 = -\frac{\pi}{2}$ und $B_1 = \frac{\pi}{2}$

$$\beta_i = \operatorname{asin}\left(-1 + 2\frac{i}{a}\right) \tag{5.4}$$

Die Längenteilung des Bereiches von Λ_0 zu Λ_1 in o Teile ist direkt mit festem Intervall gegeben:

$$\lambda_i = \Lambda_0 + \frac{i}{o}(\Lambda_1 - \Lambda_0) \tag{5.5}$$

bzw. wiederum für die ganze Kugel von 0 bis 2π:

$$\lambda_i = 2\frac{i}{o}\pi \tag{5.6}$$

Dieses Gitter definert insgesamt $a \cdot o$ Zonen auf der Kugeloberfläche, jeweils mit dem Flächeninhalt

$$A_{\text{real}} = \frac{1}{o}(2\pi - 0) \cdot R^2 \cdot \frac{1}{a}\left(\sin\frac{\pi}{2} - \sin\left(-\frac{\pi}{2}\right)\right) = \frac{4\pi}{oa}R^2$$

Ein Aspekt, den diese Partition vom einfachen Gitter mit konstantem $\Delta\beta$ erbt, ist die Einbettbarkeit höherer Auflösungen: Wenn o oder a auf ein jeweiliges Vielfaches erhöht werden, so sind alle bisher definierten Grenzen β_i und λ_i weiterhin in Gebrauch und es werden lediglich zwischen ihnen zusätzliche hinzugefügt. Diese Eigenschaft wird in Abb. 5.3 genutzt, um iterativ die Teilung mit $o = a = 8$ aufzubauen.

Die Eigenschaft der gleichen Fläche pro Zone wird erfüllt, jedoch hat die Anpassung allein über die Breitenbereiche den Nachteil, dass eben diese zu den Pole hin weiter werden. In der wichtigen zonalen Darstellung der gesamten Atmosphäre ($p - \beta$ - Plot, Mittelung über λ) bedeutet das einen Verlust and Auflösung und folglich Information zu den Polen. Die Pole sind wichtig — zumal meine Arbeit ja im Rahmen des Pole-Equator-Pole - Projektes steht!

[8] Normierte Gesamtoberfläche der Kugel ist gleich 2.

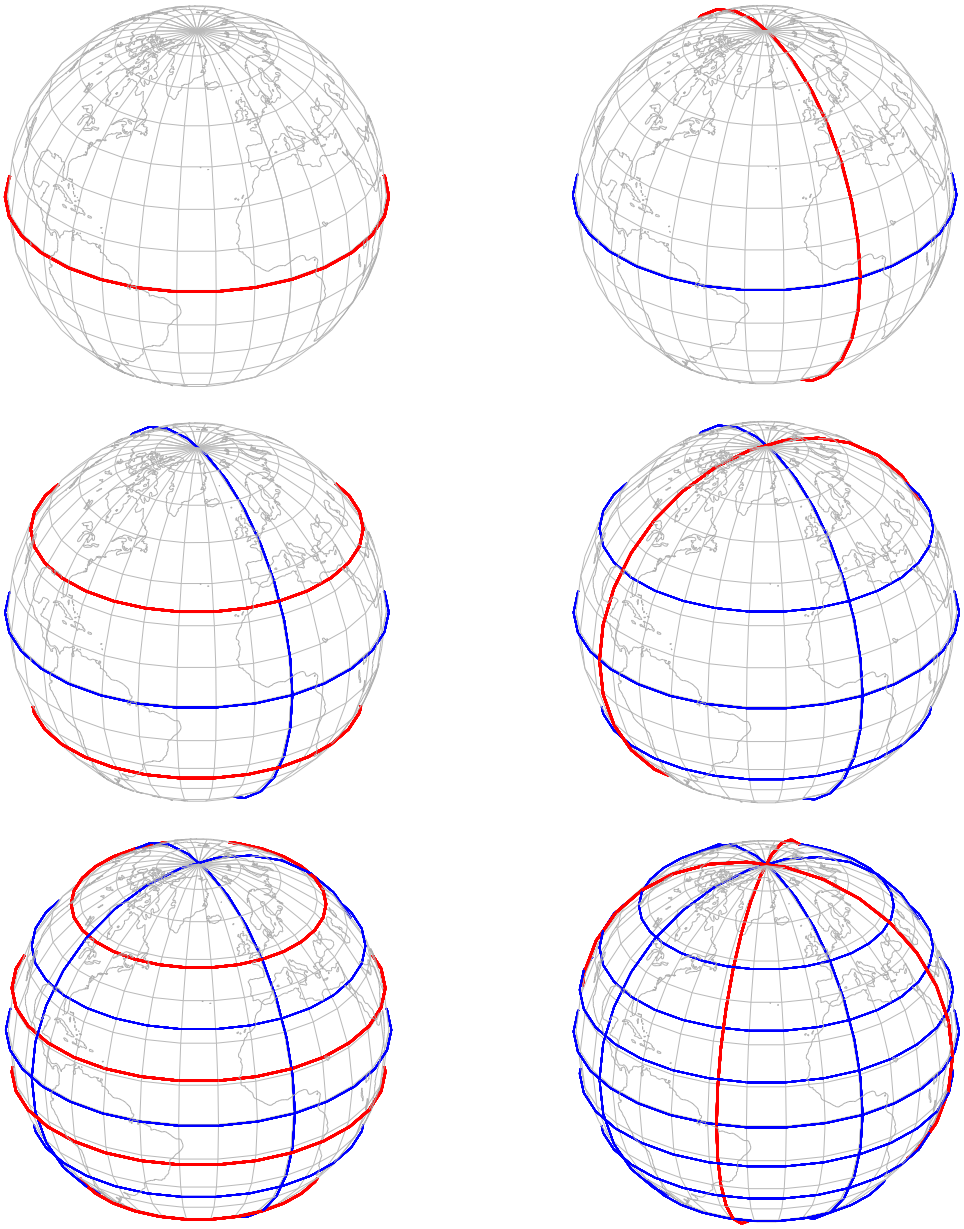


Abbildung 5.3: Die erste Aufteilung der Erdoberfläche in flächengleiche Gebiete ...mit Anzahlverdopplung zu 2, 4, 8, 16, 32 und 64 Gebieten. Flächengleichheit wird erkauft durch die gröbere Breitenauflösung an den Polen.

5.2.2 Ausgeglichene Äquipartition (Zo-a)

Die Anpassung der Zonengrundfläche über die Breite allein stellt mich nicht zufrieden. Der nächste Schritt ist die Einbeziehung der Längenteilung — dergestalt, dass ich die einfache Struktur der homogenen Längenteilung und damit die des globalen Gitters überhaupt aufgebe. Zur Variation der Breitengrenzen kommt nun eine variable Anzahl Längenteilungen (also Einzelzonen) für den individuellen Breitenbereich hinzu. Zu den Polen hin werden weniger Zonen in die Breitenbereiche gelegt und somit die gleiche Fläche pro Zone mit geringerer Breitenausdehnung ermöglicht. Im der zonalen Darstellung bleibt so eine an den Polen höhere, insgesamt homogenere Auflösung in Breitenrichtung.

Ich gebe dieser Partition der Fläche die Bezeichnung Zo-a entsprechend der Anzahl der Breitenbereiche a und der maximalen Zahl (im dem Äquator nachsten Breitenbereich) o an Längenbereichen. Verbleibende Variable ist die mögliche Eingrenzung des Längen- bzw Breitenbereiches (Grenzen $B_{0/1}$ und $\Lambda_{0/1}$). Wenn diese Grenzen nicht erwähnt werden, dann ist die Partition der gesamten Oberfläche gemeint (also $B_{0/1} = \pm\frac{\pi}{2}$ und $\Lambda_0 = 0 \wedge \Lambda_1 = 2\pi$). Im (implizit) gegebenen Gesamtbereich bestimmen die Parameter o und a eine Partition der Oberfläche eindeutig. Im Detaul besteht die Partition aus den Breitengrenzen $\beta_i \; \forall i \in [0; a]$ und den Längengrenzen $\lambda_{i,j} \; \forall i \in [0; a) \wedge j \in [1; z_i]$ für jede einzelne Breitenzone.

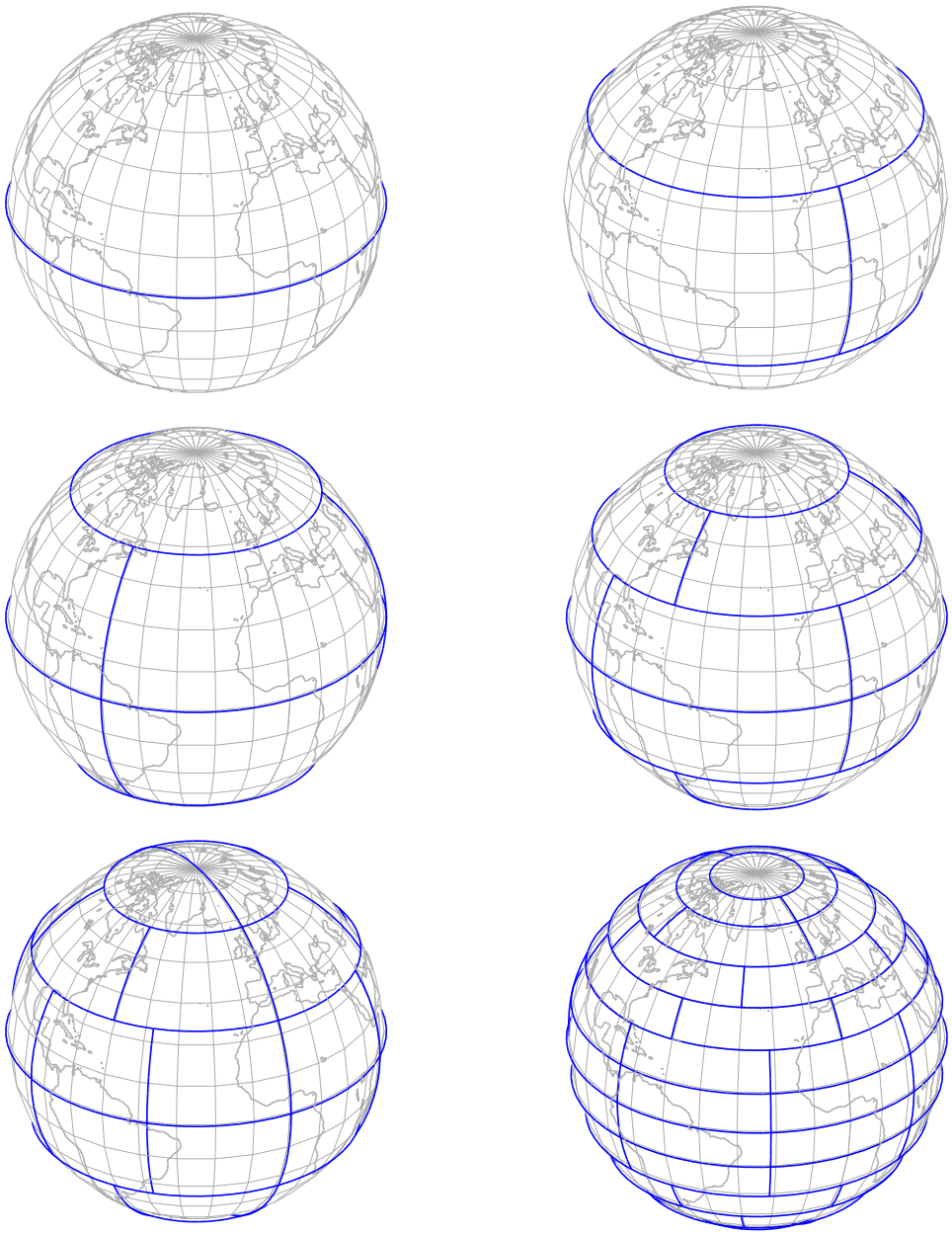


Abbildung 5.4: Ausgeglichenere Aufteilungen der Erdoberfläche
...zu 2, 4, 8, 16, 32 und 64 Gebieten. Die gewählten Parameter (von links nach rechts, oben nach unten) sind jeweils (o, a): $(1, 2)$, $(2, 3)$, $(3, 4)$, $(4, 6)$, $(8, 6)$, $(7, 14)$. Diese Aufteilung kombiniert gleiche Flächeninhalte mit nahezu gleichförmiger Breitenteilung, unter Verzicht auf die globale Teilung in Längenintervalle.

Der Algorithmus für Teilung Z*o*-*a*

1. Es existiert der vorgegebene Bereich: $\beta_0 = B_0 < \beta_a = B_1$ und $\Lambda_0 <= \lambda_{i,k} <= \Lambda_1$
2. Ausgangspunkt ist die gewünschte ideal gleichförmige Einteilung der Breiten.

$$\Delta\beta = \frac{\beta_a - \beta_0}{a};\ \ \forall i \in [1; a) :\ \beta_i := \beta_0 + i \cdot \Delta\beta$$

3. Aus dieser wird relative[9] Richtfläche A^* für eine einzelne Zonenfläche aus der Teilung des größten Breitenabschnittes bestimmt.

$$A^* := \frac{1}{o} \max \{(\sin \beta_{i+1} - \sin \beta_i) : i \in [0; a)\}$$

   Beachte: Die globale Einschränkung der Längen spielt hier keine Rolle, es geht um relative Gewichtung der Breiten.

4. Initialisierung der Gesamtzonenzahl: $Z := 0$
5. Mittels A^* wird die Anzahl z_i der Zonen in Breitenbereich i bestimmt, die Gesamtzonenanzahl Z iterativ aufsummiert:

$$\begin{array}{rrcl} \forall i \in [0; a) : & L_i & := & Z \\ & z_i & := & \text{rund}\left(\frac{1}{A^*}(\sin \beta_{i+1} - \sin \beta_i)\right) \\ z_i = 0\ ?\ & z_i & := & 1 \\ & Z & := & Z + z_i \end{array}$$

6. Die tatsächlichen relativen Einzelzonenfläche ergibt sich aus Z:

$$A^\dagger = \frac{\sin \beta_a - \sin \beta_0}{Z}$$

7. Mit dieser und den z_i (L_i sind aufsummierte z_i) werden die Breitengrenzen final festgelegt[10].

$$\forall i \in [1; a) : \beta_i := \text{asin}(\sin \beta_0 + L_i \cdot A^\dagger)$$

8. Zuletzt erfolgt die Längenteilung für jeden nun fixierten Breitenbereich. Die Ränder ergeben sich theoretisch aus der Berechnung und man könnte sich die beiden Einzelzuweisungen sparen. Hier wird der mögliche Fehler von vier unötigen Rechnenoperationen gespart und die Ecken der Welt exakt festgehalten.

$$\begin{array}{rrcl} \forall i \in [0; a) : & \Delta\lambda & := & \frac{\Lambda_1 - \Lambda_0}{z_i} \\ & \lambda_{i,0} & := & \Lambda_0 \\ & \lambda_{i,z_i} & := & \Lambda_1 \\ \forall j \in [1; z_i) : & & & \lambda_{i,j} := \Lambda_0 + j \cdot \Delta\lambda \end{array}$$

Das wird recht nah im Programm (Fragment in Quelltext 5.1) umgesetzt – mit der Ausnahme, das das Programm erst in β die Sinus-Werte speichert und bei der finalen Belegung diese durch Winkel ersetzt.

[9] relativ zum eingeschränken Längenbereich; $A^*/A^*_{\text{real}} = 2\pi/(\Lambda_1 - \Lambda_0)$

[10] Die hier stattfindene Verschiebung von der Idealteilung wird durch hinreichendes o minimiert.

Quelltext 5.1: Zo-*a* Partition

```
1   pep_t step = (B1-B0)/a; // D beta
2   #define latsin latb
3   #define latarea(i) (latsin[i+1]-latsin[i])
4   latsin[0] = sin(B0); latsin[a] = sin(B1);
5   for(size_t i = 1; i < a; ++i) latsin[i] = sin(B0 + i*step);
6   pep_t onezone = 0;
7   for(size_t i = 0; i < a; ++i) // find biggest ideal region
8   {
9     pep_t area = latarea(i);
10    if(area > onezone) onezone = area;
11  }
12  onezone /= o;       // A^*
13  zones_per_lev = 0; // Z
14  for(size_t i = 0; i < a; ++i)
15  {
16    latindex[i] = zones_per_lev; // index of first zone of band
17    // z_i: how many zones of about standard size to put in this
          latitude band
18    zones_per_lat[i] = (size_t) rint(latarea(i)/onezone);
19    if(zones_per_lat[i] == 0) zones_per_lat[i] = 1; // > 0 !
20    zones_per_lev += zones_per_lat[i];
21  }
22  // Now it's the real deal - A^dagger
23  onezone = (latsin[a]-latsin[0])/zones_per_lev;
24  // place the latitudes area-even
25  // actually switching to latitudes here
26  for(size_t i = 1; i < a; ++i)
27  latb[i] = asin(latsin[0] + latindex[i]*onezone);
28  #undef latarea
29  #undef latsin
30  latb[0] = B0; latb[a] = B1;
31  for(size_t i  = 0; i < a; ++i) // longitudes at last
32  {
33    lonb[i] = new pep_t[zones_per_lat[i]+1];
34    pep_t step = (L1-L0)/zones_per_lat[i];
35    lonb[i][0] = L0; lonb[i][zones_per_lat[i]] = L1;
36    for(size_t j = 1; j < zones_per_lat[i]; ++j)
37    lonb[i][j] = L0 + j*step; // lambda_i,j
38  }
```

5.2.3 Globale Partition in 3D (Z*o*-*a*-*e*)

Zur globalen Auswertung kombiniere ich eine Partition nach Z*o*-*a* mit der logarithmischen Teilung des gesamten vertikalen Bereichs in e Schichten. Diese Gesamtpartition in ausgeglichene Zonen wird dann mit Z*o*-*a*-*e* bezeichnet. Alternativ kann das Programm auch eine vorgegebene Liste von Grenzen in der Druckkoordinate verwenden, um z.B. die Tropopause[11] besser zu erfassen.

Beispiele der globalen Partition sind in Abb. 5.5 dargestellt.

5.2.4 Pauschale Gewichtung mit Standardatmosphäre

Mein erster Versuch, die Partikelzahlen vergleichbar zu rechnen, bestand in der Anwendung der US-Standardatmosphäre 1976[12], dargestellt in Tab. 5.1, auf die Druckebenen.

Mit Hilfe der Dichtetabelle kann ich durch Interpolation und Integration der Dichte in einem Druckbereich $p \in [p_a; p_b]$ diesem eine Pseudomasse zuordnen. "Pseudo", da ich ohne Verwendung der Raumausdehnung in m aus den kg/m^3 nicht kg errechne, sondern nur eine Skalierungsgröße mit der Einheit Pakg/m^3. Der abstraktere Begriff "Gewicht" scheint mir folglich angemessen. Die Berechnung des Gewichtes $g(p_a, p_b)$ einer Luftschicht erfolgt nach folgendem Schema (graphisch in Abb. 5.6):

- Gegeben: $p_a > p_b$ (Beginn und Ende der Schicht, Druck nach oben abfallend).
- Extrahiere (auch mit linearer Inter-/Extrapolation) aus der Atmosphärentabelle eine Untertabelle (p_i, ρ_i) mit n Einträgen, so dass $p_1 = p_a$ und $p_n = p_b$.
- Gewicht wird errechntet durch mittlere Dichte in den Teilintervallen:

$$g(p_a, p_b) = \sum_{i=2}^{n} \frac{1}{2}(\rho_{i-1} + \rho_i) \cdot (p_{i-1} - p_i) \tag{5.7}$$

Mit diesem Pauschalgewicht kann ich an dieser Stelle genauer demonstrieren, weshalb die Verteilung von gleich schweren (die gleiche Luftmasse repräsentierenden) Partikeln in der gesamten Atmosphäre unpraktikabel ist. In Abb. 5.7 ist zu sehen, wie gut die Stratosphäre mit zweihundert Schichten in der gesamten Atmosphäre abgedeckt ist — nämlich so gut wie gar nicht. Es wäre eine *sehr hohe* Zahl von Schichten und folglich eine *extrem* hohe Zahl von Partikel notwendig.

Die Zuteilung eines pauschalen Gewichtes an eine Luftschicht (Druckbereich) ist eine einfache Möglichkeit, den Zonen der dreidimensionalen Partition (basierend auf Z*o*-*a*) ein Gewicht zu geben. Per Konstruktion sind alle Zonen in ihrer Grundfläche gleich[13]. So ist ein relatives Gewicht von Zonen unterschiedlicher Schichten allein bestimmt durch die untere und obere Grenzen ihrer Schichten in der Druckkoordinate. Insbesondere haben Zonen in einer Druckschicht das gleiche Gewicht, was die Betrachtung von horizontalem Massenaustausch sehr einfach gestaltet.

[11] Grenze/Übergang zwischen den Schichten der Troposphäre und der Stratposphäre

[12] Diese Wahl ist nicht sehr bedeutsam. Es stellte sich heraus, dass dieser Ansatz problematisch genug ist, um die Frage der Wahl einer der verfügbaren Normatmosphärentabellen zu nivellieren.

[13] Erinnerung: Die Dicke der Atmosphäre wird gegenüber dem Erdradius vernachlässigt.

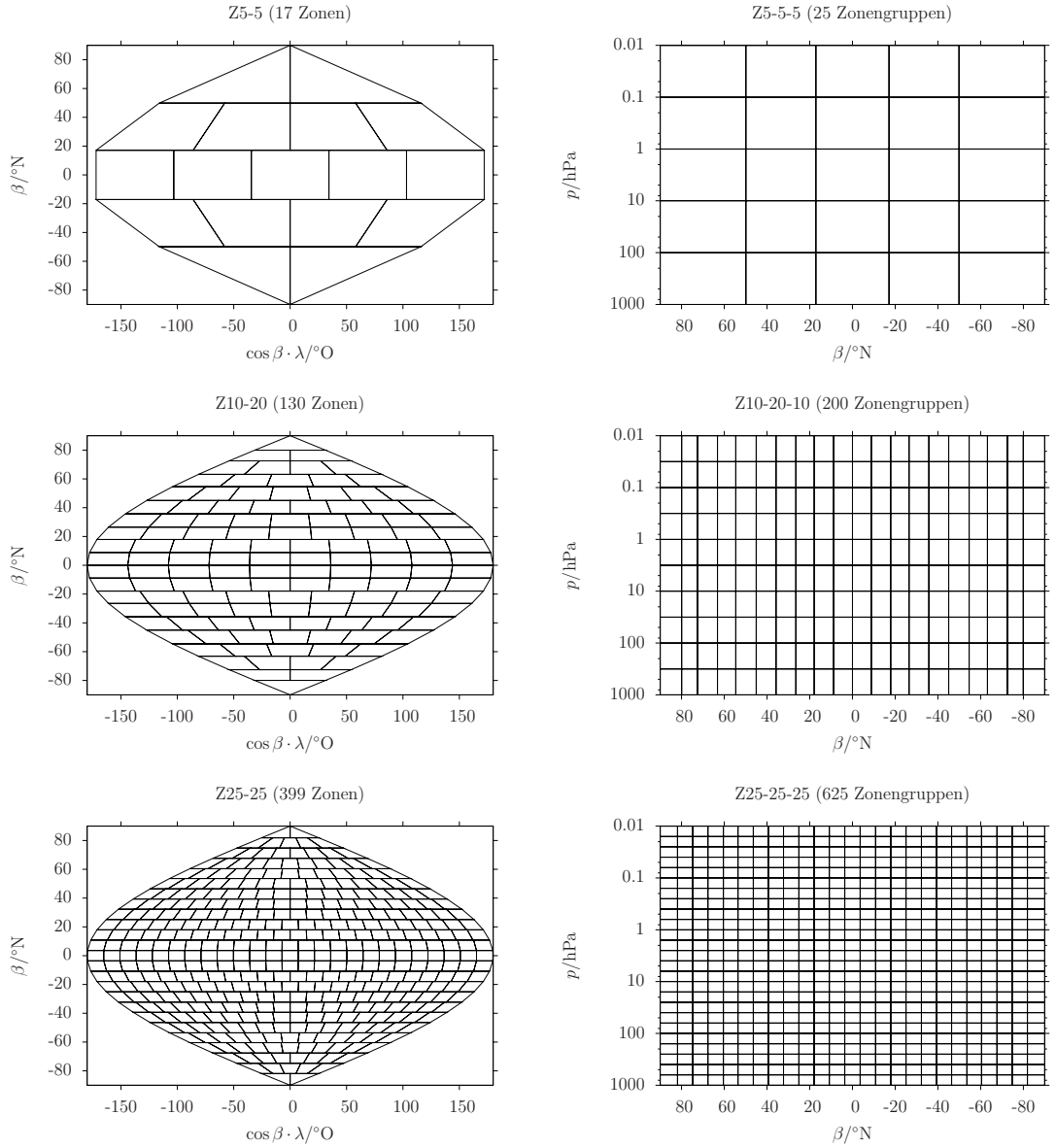


Abbildung 5.5: Drei Beispiele für Z*o*-*a* bzw. Z*o*-*a*-*e*

geopot. Höhe h/m	geometr. Höhe z/m	Temperatur T/°C	Luftdruck p/Pa	Dichte $\rho/\frac{\text{kg}}{\text{m}^3}$
0	0	15	$1,0132 \cdot 10^{+05}$	$1,2250 \cdot 10^{+00}$
11000	11019	$-56,5$	$2,2632 \cdot 10^{+04}$	$3,6393 \cdot 10^{-01}$
20000	20063	$-56,5$	$5,4749 \cdot 10^{+03}$	$8,8037 \cdot 10^{-02}$
32000	32162	$-44,5$	$8,6802 \cdot 10^{+02}$	$1,3225 \cdot 10^{-02}$
47000	47350	$-2,5$	$1,1091 \cdot 10^{+02}$	$1,4276 \cdot 10^{-03}$
51000	51413	$-2,5$	$6,6939 \cdot 10^{+01}$	$8,6163 \cdot 10^{-04}$
71000	71802	$-58,5$	$3,9564 \cdot 10^{+00}$	$6,4212 \cdot 10^{-05}$
84852	86000	$-86,2$	$3,7340 \cdot 10^{-01}$	$6,9582 \cdot 10^{-06}$

Tabelle 5.1: Standardatmosphäre 1976 mit Dichte
Abgeleitet aus [WikiNorm].

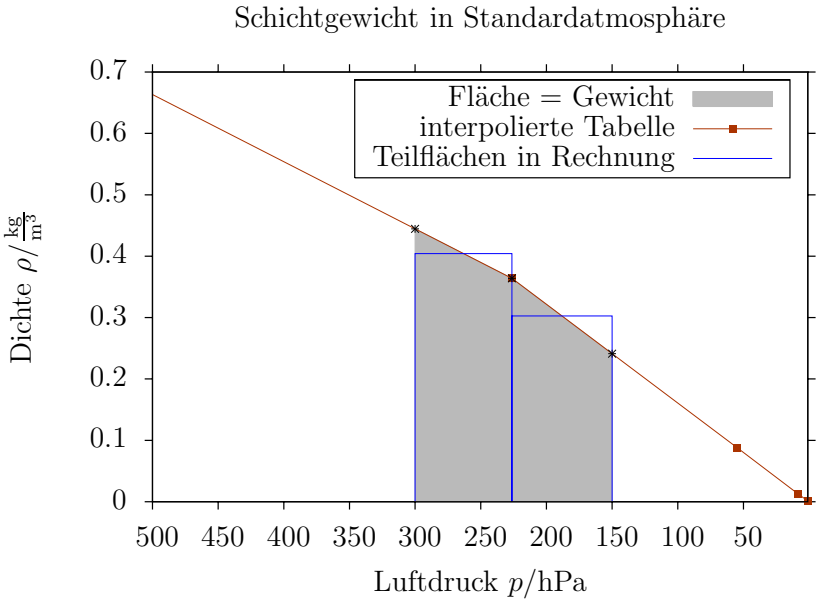


Abbildung 5.6: Berechung des Gewichtes einer Luftschicht
von 300hPa bis 150hPa in der Dichtetabelle aus der Standardatmosphäre. Es ist die einfachste Integration über Mittelwerte in durch Tabellenpunkte bestimmten Teilintervallen.

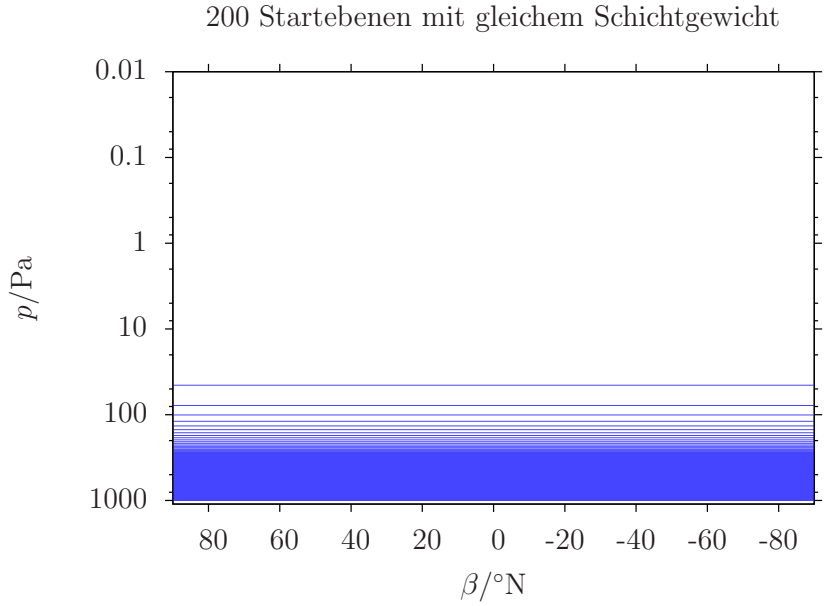


Abbildung 5.7: 200 gleich-gewichtete Schichten nach Standardatmosphäre
Es verwundert wenig: Die Luftmasse konzentriert sich am Boden. Das ist realistisch, jedoch bleibt bei solchen Startpositionen kaum Information über Stratosphärendynamik.

Wie später in der Auswertung der globalen Berechnungen noch zeigen wird, ist jedoch diese pauschale Gewichtung zur Transportanalyse ungeeignet. Dabei ist es eigentlich nebensächlich, dass die Tabelle viel zu wenige Datenpunkte enthält, um die Modellatmosphäre genügen genau zu beschreiben. Das Grundproblem ist, dass die Dynamik der Luftmassenverteilung in der Analyse von Transportberechnungen nicht ignoriert werden darf. Die Massendichte im Modell zeigt räumliche[14] und zeitliche Variabilität.

5.2.5 Dynamische Dichte aus dem Klimamodell

Ich bin nicht auf eine Tabelle wie Tab. 5.1 angewiesen, um den transportierten Partikeln Massen zuzuordnen. Die Windfelder kommen schließlich aus einem kompletten Klimamodell! Zwar ist die Luftdichte keine der von den Modellläufen ausgegebenen Variablen, aber sie lässt sich ableiten. Verfügbar sind Felder von Temperatur $T(\lambda, \beta, p)$ und relativer Feuchte $s(\lambda, \beta, p)$. Mit der Gaskonstanten für trockene Luft $R = 287,1\text{J/kgK}$ kann ich daraus die Dichte ρ berechnen:

$$\begin{aligned} p &= \rho RT(1 + 0.61s) \\ \Leftrightarrow \rho &= \frac{p}{RT(1 + 0.61s)} \end{aligned} \tag{5.8}$$

Aus dieser Formel erhalte ich zu jedem Zeitpunkt das Feld der Dichte $\rho(\lambda, \beta, o)$. In Abb. 5.8 ist zu sehen, dass das logarithmische Profil erwartungsgemäß das prägende Muster bleibt, jedoch dieses Feld horizontale sowie zeitliche Variationen zeigt. Es ist keineswegs statisch. Dies gilt allerdings auch für den Bezug der Druckkoordinaten zur Höhe in Metern, welchen

[14] auch Längen- und Breitengrad, nicht nur in der Höhe

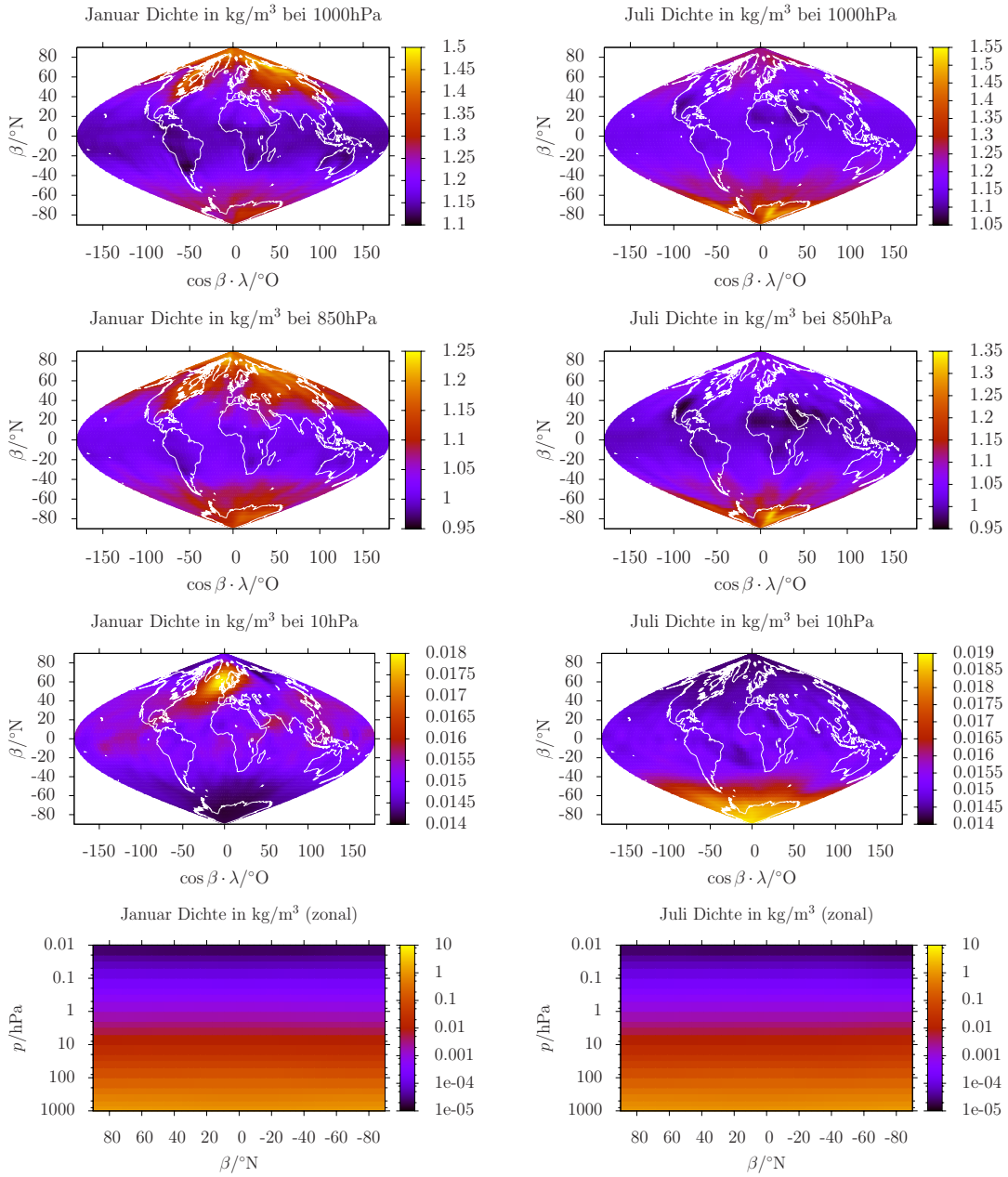


Abbildung 5.8: Luftdichte aus ECHO-GiSP im Januar und Juli 1965
Auch wenn das Dichteprofil in der Zonalansicht glatt erscheint und nur der Gradient in der Höhe auffällt, ist die horizontale und zeitliche Dichtevariabilität gegeben.

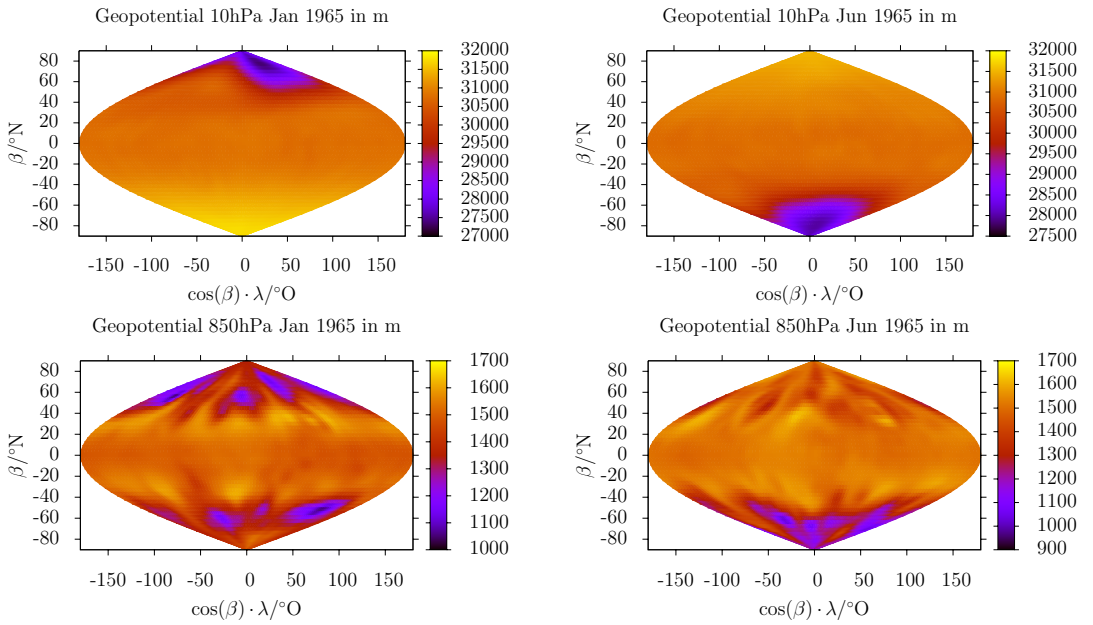


Abbildung 5.9: Geopotentielle Höhe von 850hPa und 10hPa im Januar und Juni 1965
Die Höhe der Druckebenen ist weder räumlich noch zeitlich konstant.

ich für die Anwendung dieser Dichte auf Zonen aus der Partition benötige, siehe Abb. 5.9. Die vertikalen Ausdehnung einer Zone in m zusammen mit der Grundfläche in m^2 ermöglicht die Verrechnung mit der Dichte in kg/m^3 zu einer Masse — keine Pseudomasse, sondern eine konkrete Zahl in kg.

Ich berechne die konkrete Höhe $h(\lambda, \beta, p)$ eines Punktes mit Druckkoordinate über das ebenfalls vom Klimamodell bereitgestellte Feld des Geopotentials[15] $h_p(\lambda, \beta, p)$. Das Geopotential ist angegeben in geopotentiellen Metern, welche zur Eliminierung des Höhenverlaufes der Erdbeschleunigung $g(h)$ definiert sind durch:

$$\begin{aligned} g(h)\mathrm{d}h &= g_0 \mathrm{d}h_p \\ \Rightarrow \Delta E_{pot} = \int_0^h mg(h)\mathrm{d}h &= mg_0 \cdot h \end{aligned} \tag{5.9}$$

eine direkte Umrechnung von geopotentiellen und "echten" Metern ergibt sich mit der Definition von $g(h)$ repektive $g_0 = g(0)$ durch das Verhältnis $g(h)/g_0$

$$\begin{aligned} g(h) &= \gamma \frac{M}{(R+h)^2} \\ \Rightarrow \frac{g(h)}{g_0} &= \frac{\gamma \frac{M}{(R+h)^2}}{\gamma \frac{M}{R^2}} = \left(\frac{R}{R+h}\right)^2 \end{aligned}$$

[15] Man beachte bitte, dass es hier um zeitabhängige Größen geht, auch wenn ich den Parameter t nicht explizit mitführe. Die Rechnung wird für einen speziellen Zeitpunkt behandelt.

in der Differentialgleichung zu h_p:

$$\begin{aligned} \mathrm{d}h_p &= \frac{g(h)}{g_0}\mathrm{d}h = \left(\frac{R}{R+h}\right)^2 \mathrm{d}h \\ \Leftrightarrow h_p &= \int_0^h \left(\frac{R}{R+h}\right)^2 \mathrm{d}h = -\frac{R^2}{R+h}\Big|_0^h \\ &= -\frac{R^2}{R+h} + R = \frac{-R^2 + R(R+h)}{R+h} = \frac{-R^2 + R^2 + Rh}{R+h} \\ &= \frac{Rh}{R+h} \end{aligned}$$

Die Umrechnung zwischen h_p und h ist folglich [16]

$$h_p = \frac{Rh}{R+h} \Leftrightarrow h \quad \Leftrightarrow \quad h = \frac{Rh_p}{R-h_p} \tag{5.10}$$

Mit den Feldern $\rho(\lambda, \beta, p)$ und $h(\lambda, \beta, p)$ habe ich die Daten, mit denen einer dreidimensionalen Partitionszone eine fundierte Masse zugewiesen werden kann. Die Methode **pep_t** `zone_mass::get_mass(`**place** `&bottom`, **place** `&top`) kombiniert zur Berechnung der Luftmasse in der durch `bottom` und `top` spezifizierten Zone (Intervallgrenzen für λ, β, p sowie ein bestimmter Zeitpunkt t) die Dichte aus `egp_density::get_density(dataindex index)` und die Höheninformation aus `egp_geopoth::get_height(dataindex index)` sowie die durch die Partition für alle Zonen festgelegte Grundfläche auf folgende Weise:

1. Sie bestimmt die Intervalle im Gitter der Quelldaten, die `top` und `bottom` komplett einschließen. Dies bedeutet die Auswahl aller Gitterpunkte im Inneren der Zone sowie angrenzender Punkte, wenn die Zonengrenze sich zwischen diesen und dem eingeschlossenen Gitterpunkt befindet.

2. Es wird eine mittlere Dichte aus den Werten von `egp_density::get_density(dataindex index)` an diesen Punkten berechnet. Die Mittelung erfolgt linear durch eine gewichtete und anschließend normierte Summe über alle Punkte, wobei das Gewicht durch $\cos\beta$ bestimmt wird — entsprechend der Zunahme der Dichte von Gitterpunken in höheren Breiten. In der Höhe (noch durch Druckkoordinate) gibt es keine spezielle Wichtung, da zum einen die lineare Mittelung mit logarithmisch verteilten Druckebenen schon das exponentielle Element der Dichte beinhaltet und zum anderen ich lediglich 23 Druckebenen in den Daten gegeben habe; es wird praktisch also fast immer nur über zwei eingrenzende Ebenen summiert, nicht über eine größere Anzahl, wo eine etwaige spezielle Wichtung angebracht sein könnte.

3. Aus `egp_geopoth::get_height(dataindex index)` (und lineare Interpolation zu p_{top} und p_{bottom}) wird auf analoge Weise eine mittlere Höhe durch Summierung über alle (λ_i, β_i) errechnet.

4. Diese mittlere Höhe wird mit der gespeicherten Grundfläche einer Zone zum zugeordneten Volumen multipliziert und nach Multiplikation mit der gemittelten Dichte ist das Ergebnis eine Masse für die angegebene Zone.

[16] Beruhigung misstrauischer Gemüter beim Anblick von $(R-h_p)^{-1}$: $h_p = R$ ist nur möglich als Grenzwert $\lim_{h\to\infty} \frac{Rh}{R+h} = R \lim_{h\to\infty} \frac{1}{\frac{R}{h}+1} = R$

Eine Probe dieses Verfahrens ist der Vergleich der Summe aller Zonenmassen mit einem erwarteten Wert für die Gesamtmasse der Atmosphäre. [NCAR-Masse] nennt als mittleren Gesamtmasse $5,1480 \cdot 10^{18}$kg mit Schwankungen im Bereich von $1 \cdot 10^{15} kg$ (veränderlicher Gehalt von Wasserdampf). Mit Partition Z20-20-30 erreiche ich $5,197 \cdot 10^{18}$kg, ohne Horizontale Teilung liegt Z1-1-30 bei $5,206 \cdot 10^{18}$kg. Mit nur 10 Schichten weicht Z1-1-10 etwas mehr nach oben ab, diese Teilung führt zu $5,729 \cdot 10^{18}$kg. Zwar liegen die Werte ausherhalb der von [NCAR-Masse] angegebenen Schwankungsbreite, jedoch stimmt klar die Größenordnung. Dazu kommt die Frage, wie genau denn überhaupt die von ECHO-GiSP implizierte Atmosphärenmasse mit dem Literaturwert überein zu stimmen hat. Es steht mehr die Reproduktion von globalen Zirkulationsmustern als die möglichst genaue Reproduktion der Gesamtmasse im Vordergrund.

Auch in Bezug auf die Konsistenz des Massentransportes ist die Berechnung der in Zonen aufgeteilten Luftmasse mittels Temperatur, Feuchte und Geopotential zumindest ein wichtiger Fortschritt gegenüber der pauschalen Gewichtung aus dem vorigen Abschnitt. Es zeigt sich, dass die Bezugnahme auf Modelldaten absolut notwendig ist, wenn man Transport und Mischung zwischen verschiedenen Luftschichten betrachten will. Der Vergleich des Massenerhaltungsgrades wird es belegen — nachdem ich diesen definiert habe.

5.3 Massenerhaltungsgrad, Konsistenz mit dem Klimamodell

Den Massenerhaltungsgrad definiere ich in der Modellatmosphäre, die (wie z.B. in U-Abs. 5.2.3 dargestellt) in Z Zonen mit entsprechendem Volumen V_i der Zone $i \in [1; Z]$ aufgeteilt ist:

$$V = \sum_{i=1}^{Z} V_i \tag{5.11}$$

Man beachte, dass die Volumina V_i der Zonen sich unterscheiden; sogar zeitlich variieren. Deswegen und vor allem wegen der starken Unterschiedlichkeit der Luftdichte [17] in diesen Zonen repräsentieren Skalare beim Lagrangeschen Teilchentransport in jeder Zone eine andere Luftmasse — auch, wenn in jeder Zone die gleiche Anzahl Partikel startet.

Das Tracerexperiment mit der genannten Einteilung der Atmosphäre besteht im Start mit einer festgelegten Verteilung von Partikeln auf die Zonen und dem Festhalten der Anzahlen von Partikeln aus Zone j in Zone i.

$$\text{Anzahl von } j\text{-Partikeln in Zone } i: \qquad n_{ij}(t) \tag{5.12}$$

Für weitere Betrachtungen und als Basis für Vergleiche wird die Luftmasse $\mu_i(\vec{P}(t))$ einer Zone benötigt. Diese wird abgeleitet aus den zeitabhängigen physikalischen Parametern $\vec{P}$ des Klimamodells (Temperatur, Feuchte, Zellengröße durch Geopotential der Druckflächen).

$$\text{Modellmasse} \qquad \mu_i(\vec{P}(t)) \tag{5.13}$$

[17] Hauptsächlich durch den vertikalen Gradienten, aber auch weden horizontaler Variationen

Ein Partikel aus der Zone j repräsentiert einen Bruchteil der Masse, den die Zone zum Zeitpunkt t_0 hatte, $\mu_j(\vec{P}(t_0))/n_{jj}(t_0)$. Daraus erhält man die *repräsentierte Masse* aller Partikel aus Zone j in Zone i:

$$\text{repräsentierte Masse } m_{ij}(t) \quad = \quad n_{ij} \cdot \frac{\mu_j(\vec{P}(t_0))}{n_{jj}(t_0)} \tag{5.14}$$

Summiert man diese Teilmassen für jede Quellzone j auf, so ergibt das die *Transportmasse*, d.h. die Masse, die die Zone i durch die Transportberechnung nun beherbergt.

$$\text{Transportmasse } M_i(t) \quad = \quad \sum_{j=1}^{Z} m_{ij}(t) \tag{5.15}$$

Das ist die eine Seite. Die andere Seite ist die Masse $\mu_i(\vec{P}(t))$, die Zelle i laut den Modelldaten zum Zeitpunkt t haben sollte. Indealerweise sollten diese Massen übereinstimmen. Wenn das der Fall ist, dann ist meine durch die Windfelder aus dem Klimamodell getriebene Transportberechnung mit dem Gesamtzustand des Modells konsistent (was die Masseverteilung angeht). Das wäre eine Bestätigung für meine Methodik, aber auch für das Klimamodell selbst. Schließlich sollte der generierte Wind tatsächlich die Dynamik der Luftmassen widerspiegeln. Nun, das Maß zur Untersuchung dieser Konsistenz ist der

$$\text{Massenerhaltungsgrad } \eta(t) \quad = \quad \frac{M_i(t)}{\mu_i(\vec{P}(t))} \tag{5.16}$$

Dieser ist zum Zeitpunkt t_0 definitionsgemäß gleich 1 und wird in Verlauf des numerischen Transportexperiments mehr oder minder stark variieren. Der Vergleich zwischen dem Modelllauf mit interaktiver Chemie und der Referenz ist hier interessant, da die generierten Winde nur beim interaktiven Lauf eine Rückkopplung in das Klimamodell (über das Chemiemodul) aufweise. ECHO-GiSP berechnet Semi-Lagrangeschen Transport der Dichten der verschiedenen chemischen Konstituenten und die geänderte Verteilung der Chemie-Dichten nimmt durch Änderung der Energieeinstrahlung (Absorption/Reflektion von Sonnenlicht) Einfluss auf die Modelldynamik. Beim Referenzlauf werden die Winde zwar auch im Chemiemodul genutzt, aber es findet keine Rückkopplung zur Kerndynamik statt.

Die ersten Betrachtungen zum Massenerhaltungsgrad führte ich durch, nachdem ich die Zuordnung von Gewichten mittels der groben Tabelle der Standardatmosphäre eingebunden hatte. Es waren diese Betrachtungen, die mir deutlich mitteilten, dass dieser Ansatz zum Gewicht (zur Luftmasse) viel zu grob ist. In Abb. 5.10 ist das eindrucksvoll nachzuvollziehen — die Zahlen sind gerade in den oberen Luftschichten so verschieden von der idealen Eins, dass sie Nachbarzonen überschreiben! In den untersten vier Schichten könnte man noch über die Anwendung dieser Gewichtung nachdenken (und überlegen, ob das Verändern anderer Parameter, die die Genauigkeit der Rechnung beeinflussen, zum Ziel führen würde), aber darüber hinaus sind die Werte völlig inakzeptabel. Eine Dichtetabelle mit mehr Einträgen als Tab. 5.1 mag ein Stück helfen, jedoch ist der dynamische Ansatz mit Daten aus dem Modell mit sicherheit besser als jede statische Tabelle.

Die Massenerhaltung unter Verwendung der Modelldaten in Abb. 5.11 zeichnet ein weitaus angenehmeres Bild. Zwar sind die Werte von η nicht frei von Abweichungen, jedoch bleiben diese in der einstelligen Größenordnung und können als Diskussionsgrundlage für reale Probleme meiner Methode oder auch der Modelldaten dienen.

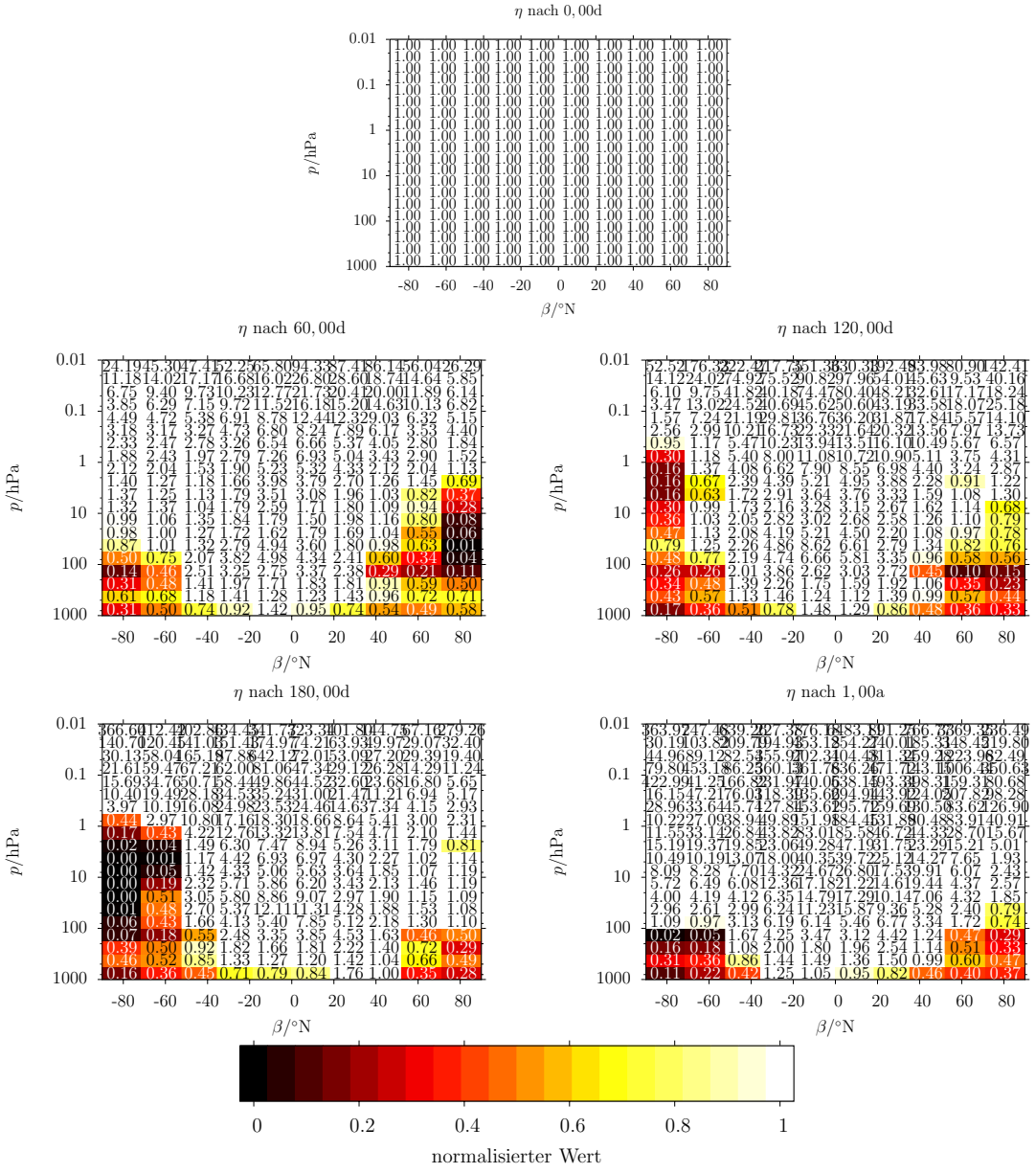


Abbildung 5.10: Massenerhaltungsgrad d1000-3, Standardatmosphäre
Die erste — erschreckende — Variante des Massenvergleichs nach U-Abs. 5.2.4 für das vom d1000-3 Lauf abgedeckte Jahr 1965 des interaktiven ECHO-GiSP Laufes. Die Werte von η in den jeweiligen Zonen (bzw. das Mittel über alle Zonen in einem Breitengrad/Druck-Bereich, die in dieser Darstellung "hintereinander" liegen) sind duch die Farbkodierung und die eingetragene Zahl gegeben. Die Farbkodierung ist dabei nur aussagekräftig von 0 bis 1.

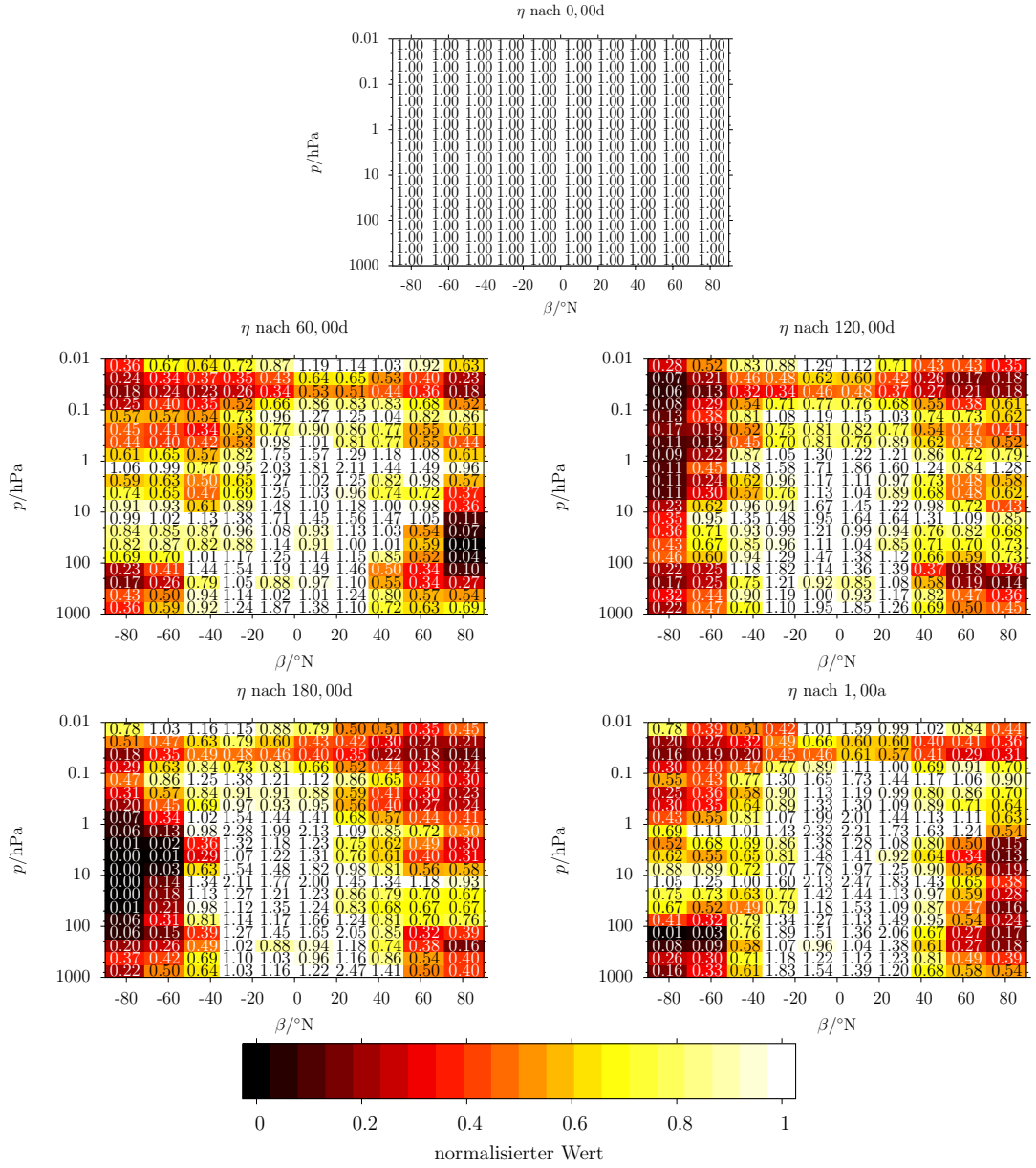


Abbildung 5.11: Massenerhaltungsgrad d1000-3, dynamisches Modellgewicht
Aus dem selbem Trajektorienensembe berechnet wie Abb. 5.10, zeigt diese hier nach einbeziehung der Dichtedaten aus dem Modell ein wesentlich ausgeglicheneres Bild. Allerdings ist das Bild nichtungetrübt: Besonders an den Polen scheint es ein Problem zu geben, die Masse zu halten. Es muss in Zukunft geklärt werden, inwiefern das an der Trajektorienberechnung liegt und auch mit ihrer Verbesserung verbessert werden kann, oder eine (Teil-) Ursache in dem Klimamodell hat.

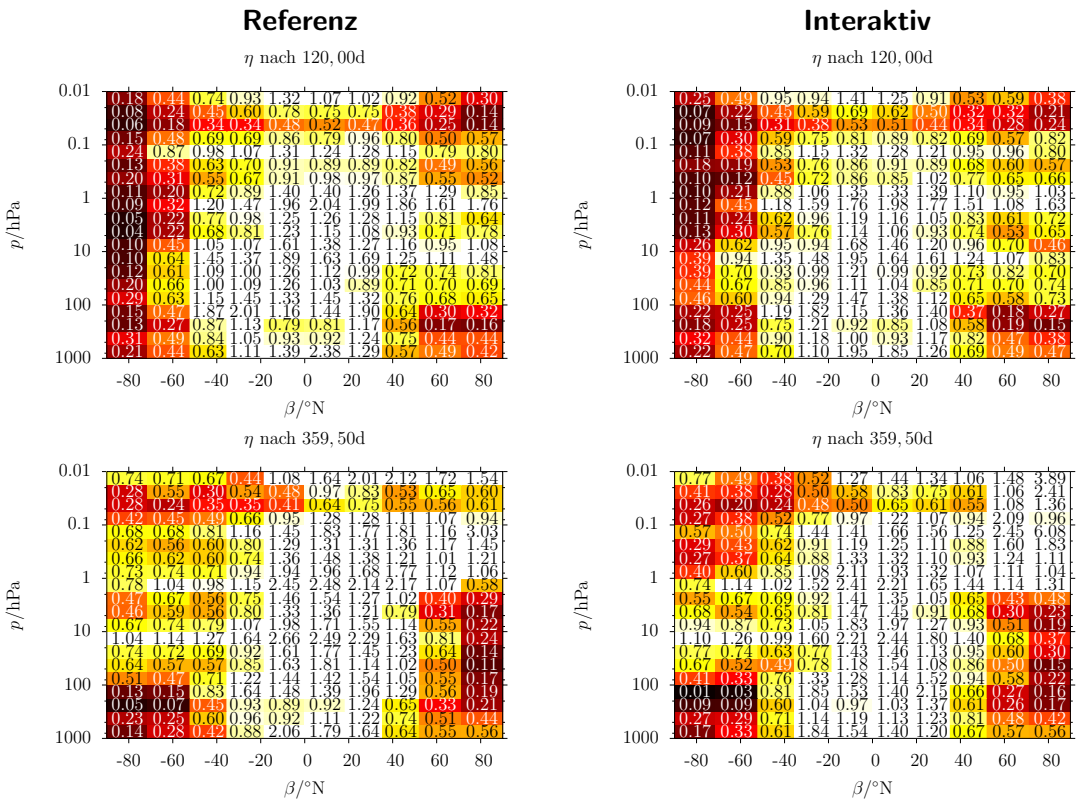


Abbildung 5.12: Vergleich Massenerhaltung Referenz und interaktiv
...von zwei frühen d1000-3 Läufen. Man sieht im Detail (konkreter Wert einer Zahl) einige Unterschiede, während das globale Bild sich sehr ähnelt.

Ich schließe dieses Kapitel mit einem Vergleich der Massenerhaltung im d1000-3 – Lauf mit Wind aus dem interaktiven ECHO-GiSP und dem Referenzmodelllauf[18] , zu sehen in Abb. 5.12. Gravierende Unterschiede sind hier allerdings nicht zu erkennen. Für eine detaillierte Analyse ziehe ich es auch vor, noch größere Ensembles zu betrachten: Um eine Milliarde Partikel (ohne explizite Speicherung der Trajektorien) sollten mit einigen Wochen / wenigen Monaten Rechenzeit auf dem Grotrian-Cluster möglich sein und werden in Angriff genommen.

[18] Diese d1000-3 sind etwas frühere Daten als zuvor gezeigt, da die Neuberechung aufgrund einer Detail-Fehlerbehebung in der Integration noch nicht für den Referenzlauf erfolgte. Sie sind also nicht direkt mit Abb. 5.11 aber durchaus untereinander vergleichbar.

6 Zusammenfassung

In der vorliegenden Diplomarbeit habe ich einen numerischen Ansatz für den dreidimensionalen lagrangeschen Transport in der Atmosphäre, getrieben durch vorgegebene Windfelder aus einem Klimamodell, von Grund auf erarbeitet. Die Rechnung eines Integrationsschrittes in lokalen Koordinaten umgeht das Problem der geographischen Koordinaten in Polnähe und ermöglicht so eine wirklich globale Berechnung von Partikelbahnen mit einem einheitlichen Verfahren. Die skizzierte und unter [PEP-Tracer] als kompletter Quelltext einsehbare moderne Softwarearchitektur wurde nicht nur entworfen, sondern ist aktuell in der Lage, große Daten auf der Ein- und Ausgabeseite zu verarbeiten. Der modulare Aufbau ermöglicht die konsequente Weiterentwicklung und Einbindung von fortgeschrittenen Analysemethoden und alternativen Ansätzen z.B. zur Interpolation und Integration.

Es wurde der Transport von Ensembles von mehreren Millionen Partikeln über einen Zeitraum von bis zu 80 Jahren berechnet. Die Zeit zum Berechnen der Trajektorien auf dem Grotrian – Cluster lag dabei jeweils unter einer Woche. Die Effizienz kann sicher noch spürbar gesteigert werden, da Geschwindigkeitsoptimierung bisher nur in Form des Puffers von `echog_pressure_cached` stattfand.

Die Wichtung von Partikeln unter Nutzung einer aus den Modelldaten abgeleiteten lokalen Luftdichte ermöglicht theoretisch quantitative Untersuchungen von Transport und Vermischung auch über weite Distanzen in der vertikalen Richtung (vom Erdboden zur Stratosphäre und umgekehrt). Die ausgeglichene Partition einer Kugelschale mit Zellen gleicher Grundfläche bei Erhaltung der Breitenauflösung bietet dafür eine Grundlage, die statistischen und numerischen Probleme der Verkleinerung der Zellen eines einfachen Gradgitters zu den Polen hin zu vermeiden. Praktische Grenzen der Berechnung von Massenverhältnissen müssen noch geklärt, die Methode mit Blick auf das Maß des Massenerhaltungsgrades verfeinert werden.

Die bisherigen Resultate wurden bereits zu mehreren Gelegenheiten präsentiert. Erste Gehversuche mit zweidimensionalem Transport mit einfachster Integration waren in Form einer Animation auf dem Potsdamer Wissenschaftsmarkt anlässlich des Einsteinjahres 2005 ([WissMarkt]) sowie als Poster auf der IEEE NDES 2005 ([NDES2005]) und der EGU Generalkonferenz 2006 ([EGU2006A] und [EGU2006N]) zu sehen. Die dreidimensionalen Betrachtungen traten auf der diesjährigen EGU-Generalkonferenz in Form eines Posters auf ([Orgis2007]). Dazu hat das entwickelte Programmpaket auch weitere Arbeit and den ECHO-GiSP – Daten gefördert, präsentiert in einem Vortrag auf der diesjährigen EGU Generalkonferenz ([Chandra2007]).

In Zukunft werden noch größere Ensembles zur detaillierteren Evaluierung der Methode berechnet werden. Die Auswertung wird ausgebaut. Die produzierten Trajektorien bedürfen weiterer statistischer sowie Methoden der Nichtlinearen Dynamik, um letztendlich weitere atmosphärenphysikalische Fragestellungen zu bearbeiten.

Mehr ein Nebenprodukt – aber dennoch ein für mich wichtiges – ist, dass habe ich die Fragestellung der linearen Interpolation in einer Gitterzelle von beliebiger Anzahl an Dimensionen einmal konsequenter dargestellt habe als ich das bisher in der Literatur finden konnte[1].

[1] Ich meine Gl. 3.9 *und* Gl. 3.13

Literaturverzeichnis

[Brand2007] Sascha Brand, K. Dethloff, D. Handorf: *Influence of stratospheric ozone chemistry on the atmospheric variability in a coupled atmosphere-ocean-sea ice model*, J. Geophys. Res. (Atm.) submitted 2007 (Poster-Präsentation auf EGU General Assembly: `http://www.cosis.net/abstracts/EGU2007/10114/EGU2007-J-10114.pdf`)

[Bronstein2000] I.N. Bronstein, K.A.Semendjajew, G. Musiol und H. Mühlig: *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Thun und Frankfurt am Main, 5. Aufl., 2000/2001

[Chandra2007] Komalapriya Chandrasekaran, Th. Orgis, U. Schwarz, J. Kurths, S. Brand, K. Dethloff: *Recurrence plots for investigation of nonlinear low frequency variability in atmosphere*, `http://www.cosis.net/abstracts/EGU2007/07719/EGU2007-J-07719.pdf` (Vortrag auf EGU General Assembly 2007)

[EGU2006A] Thomas Orgis, S. Brand, U. Schwarz, J. Kurths, K. Dethloff: *Tracer advection in ECHO-GiSP GCM*, `http://www.cosis.net/abstracts/EGU06/03189/EGU06-J-03189.pdf` (Poster zu EGU General Assembly 2006, als Bilddatei: `http://www.agnld.uni-potsdam.de/%7Eorgis/pep/2006/egu-atmo-poster.png`)

[EGU2006N] Thomas Orgis, S. Brand, U. Schwarz, J. Kurths, K. Dethloff: *Saddle Nodes in Wind Fields*, `http://www.cosis.net/abstracts/EGU06/03193/EGU06-J-03193.pdf` (Poster zu EGU General Assembly 2006, als Bilddatei: `http://www.agnld.uni-potsdam.de/%7Eorgis/pep/2006/egu-nlp-poster.png`)

[NCAR-Masse] Kevin E. Trenberth and Lesley Smith (National Center for Atmospheric Research): *The Mass of the Atmosphere: a Constraint on Global Analyses*, http://www.cgd.ucar.edu/cas/abstracts/files/kevin2003_6.html

[NDES2005] Thomas Orgis, S. Brand, K. Dethloff, J. Kurths: *Atmospheric Tracer Advection*, `http://www.agnld.uni-potsdam.de/%7Eorgis/pep/2005/ndes-poster.png` (Poster zur Konferenz Nonlinear Dynamics in Electronic Systems 2005 (Bilddatei))

[NetCDF] *Network Common Data Form*, `http://www.unidata.ucar.edu/software/netcdf/` (Definition und Software zum NetCDF Datenformat)

[Orgis2007] Thomas Orgis, S. Brand, U. Schwarz, J. Kurths, K. Dethloff: *3D Tracer advection in ECHO-GiSP GCM*, `http://www.cosis.net/abstracts/EGU2007/02313/EGU2007-J-02313.pdf` (Poster zu EGU General Assembly 2007, als Bilddatei: `http://www.agnld.uni-potsdam.de/~orgis/pep/2007/egu.png`)

[PEP] *Pole - Equator - Pole (PEP) — Variability of atmospheric trace constituents along a North-South Transect*, `http://www.awi-potsdam.de/www-pot/atmo/pep/` (Interdisziplinäres Forschungsprojekt mit Teilnahme von HFG-Forschungszentren (AWI, FZK) und Universitäten (Bremen, Potsdam, Karlsruhe))

[PEP-Tracer] Thomas Orgis: *PEP-Tracer Programmpaket zur Analyse von ECHO-GiSP Daten (vorläufiger Arbeitstitel)*, `http://www.agnld.uni-potsdam.de/~orgis/diplom/`, eMail: orgisagnld.uni-potsdam.de (Unter der URL finded man die SVN-Revision 1543 im Unterverzeichnis `pep-tracer` oder im Tarball `pep-tracer-r1543_20070521.tar.bz2`; für aktuellere bitte anfragen. In Zukunft evtl. auch über TOCSY: `http://tocsy.agnld.uni-potsdam.de/`)

[Rovatti1998] Riccardo Rovatti, Michele Borgatti, Roberto Guerrieri: *A Geometric Approach to Maximum-Speed n-Dimensional Continuous Linear Interpolation in Rectangular Grids*, IEEE Transactions on Computers 8/47Aug.1998, S. 894-899

[Shepard1968] Donald Shepard: *A two-dimensional interpolation function for irregularly-spaced data*, Proceedings of the 1968 ACM National Conference, 1968, S. 517-524

[Stohl1998] A. Stohl, P. Seibert: *Accuracy of trajectories as determined from the convervation of meteorological tracers*, Q.J.R. Meteorol. Soc. 1241998, S. 1465-1484

[WikiNorm] Wikipedia: *Normatmosphäre (22:26 Uhr, 15. Mai 2007)*, `http://de.wikipedia.org/w/index.php?title=Normatmosph%C3%A4re&oldid=31879482` (Ursprüngliche Datenquelle ist NCAR.)

[WissMarkt] Thomas Orgis: *Film der 2D Bewegung von 10000 Partikeln über 3 Monate*, `http://www.agnld.uni-potsdam.de/%7Eorgis/pep/2005/wissmarkt.mpeg` (erstellt für den Wissenschaftsmarkt anlässlich des Einstein-Jahres in Potsdam)

7 Dank & Erklärung

Danksagung

Ich danke Jürgen Kurths und Udo Schwarz für die Betreuung dieser Diplomarbeit.
Ich danke Sascha Brand und Klaus Dethloff vom Alfred-Wegener-Institut für die Daten, die meinen Rechnungen erst einen Sinn geben.
Ich danke Timo Felbinger für seine fortgeschrittenen LaTeX– Kenntnisse (und dass er sie mit mir teilt).
Ich danke der Welt für freie Software, die den freien Gedanken eines Wissenschaftlers Wege bereitet.
Ich bedanke mich für zahlreiche angeregte Gespräche und gewonnene Einsichten.

Ein spezieller Dank geht an alle, die mit mir Geduld hatten.

Erklärung

Hiermit erkläre ich, diese Diplomarbeit eigenständig unter Nutzung der angegebenen Quellen nach bestem Wissen und Gewissen angefertigt zu haben. Es wurden keine Inhalte von anderen als den genannten Quellen eingebunden.

Thomas Orgis