Arbeitsgemeinschaft der Vermessungsverwaltungen
der Länder der Bundesrepublik Deutschland (AdV)

Bundeseinheitliche Transformation für ATKIS (BeTA2007)

Version: 1.5

Bearbeitet von der Projektgruppe Bundeseinheitlicher Transformationsansatz für ATKIS
im Arbeitskreis Geotopographie der AdV

Bearbeitungsstand: 21. Februar 2012

Inhalt:

1 Motivation

Mit der zunehmenden Verbreitung von Geoinformationssystemen wächst die Anforderung, bestehende geotopographische Datenbestände in ein einheitliches Koordinatenreferenzsystem zu überführen. Durch die Beschlüsse der AdV wurde das europaweit verwendete Bezugssystem ETRS89 mit der UTM-Abbildung in der Landesvermessung und im Liegenschaftskataster deutschlandweit festgelegt. Gemäß Beschluss 118/9 (September 2006) des Plenums der AdV wird für die Transformation geotopographischer Daten nach ETRS89/UTM unter besonderer Berücksichtigung des Erhalts der zwischen den Ländern bereits harmonisierten ATKIS®-Landesgrenzen bundeseinheitlich der international verwendete und als OpenSource verfügbare Ansatz *National Transformation Version 2* (NTv2) eingeführt. Diese „Bundeseinheitliche Transformation für ATKIS“, kurz BeTA, wird im Folgenden erläutert und verfolgt nachstehende Ziele:

- möglichst einfacher mathematischer Ansatz
- exakt ein Ansatz für das gesamte Bundesgebiet
- Berücksichtigung regionaler Genauigkeitsansprüche
- einfache und kostenoptimierte Integration in vorhandene Software-Infrastrukturen
- Erstellung einer bundesweiten Gitterdatei für den Datumswechsel mit Submetergenauigkeit
- kostenfreie Bereitstellung des Gitters und Dokumentation des Transformationsansatzes im Internet unter http://www.adv-online.de bzw. im Informationssystem CRS-EU http://www.crs-geo.eu/BeTA2007.

2 Transformationsansatz

Die Transformationslösung beruht im Wesentlichen auf der Kombination zweier mathematischer Ansätze:

- der Umrechnung ebener konformer (Gauß-Krüger: GK bzw. UTM) in geographische Koordinaten (Länge, Breite: LB) und zurück
- dem gitterbasierten Bezugssystemwechsel mittels NTv2.

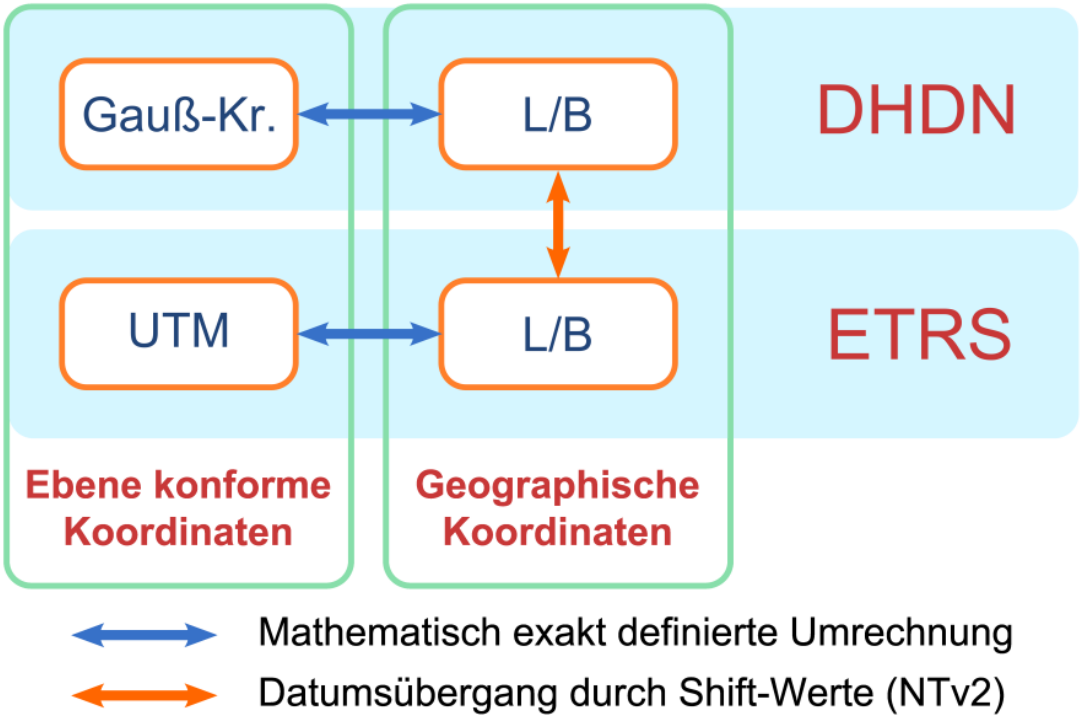


Abbildung 1

Auf dieser Grundlage und dem in Abbildung 1 dargestellten Zusammenhang, lassen sich Hin- und Rücktransformation in jeweils 3 Schritten durchführen:

- DHDN90/GK ⇒ DHDN90/LB ⇒ ETRS89/LB ⇒ ETRS89/UTM
- ETRS89/UTM ⇒ ETRS89/LB ⇒ DHDN90/LB ⇒ DHDN90/GK

Im ersten Schritt erfolgt die Umrechnung der ebenen konformen Koordinaten in geographische Koordinaten. Der Bezugssystemwechsel geschieht mit Hilfe des NTv2-Ansatzes über ein achsparalleles Gitter von Versatzwerten (*shift values*, daher im folgenden Shiftwerte genannt). Die hieraus resultierenden geographischen Koordinaten im Zielbezugssystem werden dann in ebene konforme Koordinaten umgerechnet.

2.1 Umrechnung ebener konformer in geographische Koordinaten

Ebene konforme Koordinaten aus Gauß-Krüger- und UTM-Abbildung basieren auf einem identischen mathematischen Ansatz, der transversalen Mercatorabbildung. Beide bilden die Oberfläche

eines Rotationsellipsoids auf einen Zylinder ab, dessen Achse in der Äquatorebene liegt. Dies führt dazu, dass der gleiche Formelansatz mit unterschiedlichen Parametersätzen für beide Umrechnungen verwendet werden kann. In der Literatur finden sich hierzu zahlreiche Lösungen. Unter 6.1.2.3 ist ein Ansatz aus verschiedenen Quellen beschrieben, der sich an die Implementierung der PROJ.4-Library orientiert.

2.2 NTv2 (National Transformation version 2)

NTv2 bezeichnet einen Ansatz zur Berechnung von Bezugssystemübergängen basierend auf gitterförmigen Shiftwerten auf Grundlage geographischer Koordinaten. Dazu werden in einem einmaligen Vorprozess für vorgegebene Gitterpunkte in regelmäßigen Abständen nach einem präzisen Ansatz die Shiftwerte bestimmt und in einer Gitterdatei bereitgestellt. Die realen Shiftwerte der Koordinaten werden in der Anwendung durch bilineare Interpolation innerhalb der Gittermasche ermittelt.

2.2.1 Historie

NTv2 ist ein nationaler kanadischer Standard und wurde entwickelt, um den Bezugssystemwechsel NAD27 ⇔ NAD83 zu realisieren. NTv2 stammt aus dem Jahre 1996 und ist der Nachfolger der jetzt NTv1 genannten Methode aus dem Jahre 1992. Der Ansatz ist als nationaler Standard dokumentiert [Goit05] und einige GIS-Hersteller haben ihn funktional in ihr Produkt integriert.

Australien, Neuseeland und Frankreich haben diesen Ansatz adaptiert und verwenden ihn dementsprechend für Transformationslösungen mit Datumsübergang.

2.2.2 Bezeichnungen

Ziel des Verfahrens ist die Transformation einer Koordinate vom Quell- in das Zielsystem und zurück. Für die formale Beschreibung des Ansatzes wird auf Abschnitt 6.1.1 verwiesen. Für die Ein- und Ausgabedaten gelten folgende Bezeichnungen:

```
qlat : Geogr. Breite im Quellsystem
qlon : Geogr. Länge  im Quellsystem
zlat : Geogr. Breite im Zielsystem
zlon : Geogr. Länge  im Zielsystem
```

2.2.3 Spezifikation

Für eine detaillierte Dokumentation sei auf die Originaldokumentation [JuFa95] und die australische Dokumentation [MiCo00] verwiesen. Es folgt eine kurze Beschreibung der Gitterdatei und des Transformationsverfahrens, um den Einstieg in die englischsprachige Literatur zu erleichtern.

2.2.3.1 Gitterdatei

Die Gitterdatei existiert in zwei Formen:

- als ASCII-Version (Dateiendung *.gsa*)
- als Binär-Version (Dateiendung *.gsb*)

Beide Versionen haben den gleichen Inhalt. Die binäre Version kann programmgestützt aus der ASCII-Version abgeleitet werden und umgekehrt. Jede Gitterdatei beschreibt den Datumswechsel von einem Quell-System in ein Ziel-System. Der Dateiaufbau wird im Folgenden beschrieben.

Die Gitterdatei (*gridfile*) beinhaltet ein oder mehrere, bezogen auf die geographischen Koordinaten des Quellsystems, rechteckige Gitter, sog. Subgitter (*sub grids*). Jedes Subgitter besteht aus einer Anordnung von Punkten mit regelmäßigen Abständen in geographischer Länge und Breite, die von Subgitter zu Subgitter variieren können. Zu jedem Gitterpunkt werden die Shiftwerte für Länge und Breite und deren Genauigkeitswerte abgelegt. Die Speicherung in der Datei erfolgt zeilenweise, beginnend in der südöstlichen und abschließend mit der nordwestlichen Ecke des Gebietes.

Achtung: Diese für den mitteleuropäischen Raum ungewöhnliche Ablage – hier verwendet man meist die Ablage der Daten von links unten nach rechts oben – resultiert aus der Lage des Ursprungslandes von NTv2, Kanada. Es liegt westlich vom Nullmeridian und erreicht durch diese Spiegelung die Ablage der Shiftwerte nach aufsteigenden Längen- und Breitenkoordinaten. Um dieses Prinzip auch im Bereich östlicher Länge beizubehalten, ist es erforderlich für **alle** geographische **Längen**angaben (Header-Werte und Shiftwerte) einen Vorzeichenwechsel durchzuführen!

Eine Gitterdatei hat den schematischen Aufbau:

Datei Header
Subgitter 1: Header
Subgitter 1: Shiftwerte
...
...
Subgitter N: Header
Subgitter N: Shiftwerte
Ende-Satz

Tabelle 1

Jeder Satz umfasst genau 16 Byte. Folgende Datentypen finden Verwendung:

Typ	Byte	Bemerkung
Integer	4	In der binär Version durch 4 NULL-Bytes auf 8 Byte aufgefüllt
Float	4	IEEE 754
Double	8	IEEE 754
String	8	8 Zeichen, kürzere Zeichenketten werden durch Blanks aufgefüllt

Tabelle 2

Die Originaldokumentation legt explizit keine Zeichenkodierung fest. Vorhandene Musterdateien verwenden ISO-8859 (7 Bit), so dass dies als Kodierung für BeTA festgeschrieben wird. Auch über die Byte-Reihenfolge und die Kodierung der Fließkommazahlen trifft die Originaldokumentation keine Aussage. Die in [MiCo00] mit *Little Endian* vorgeschriebene Kodierung ist auch für BeTA maßgebend. Zusätzlich wird der Standard IEEE 754 (siehe [IEEE85]) für die Kodierung der Datentypen Float und Double festgelegt.

Der Datei-Header (*Overview*) besteht aus 11 Einträgen:

#	Kennung	Wert	Beschreibung	ASCII-Format
1	NUM_OREC	Integer	Anzahl der Header Records (11)	%-8s%3d
2	NUM_SREC	Integer	Anzahl der Header Records beim Subgitter	%-8s%3d
3	NUM_FILE	Integer	Anzahl Subgitter	%-8s%3d
4	GS_TYPE	String	Einheit der Shiftwerte (SECONDS)	%-8s%-8s
5	VERSION	String	Versionsbezeichnung	%-8s%-8s
6	SYSTEM_F	String	Name des Quell-Ellipsoids	%-8s%-8s
7	SYSTEM_T	String	Name des Ziel-Ellipsoids	%-8s%-8s
8	MAJOR_F	Double	Großer Halbmesser des Quell-Ellipsoids	%-8s%12.3f
9	MINOR_F	Double	Kleiner Halbmesser des Quell-Ellipsoids	%-8s%12.3f
10	MAJOR_T	Double	Großer Halbmesser des Ziel-Ellipsoids	%-8s%12.3f
11	MINOR_T	Double	Kleiner Halbmesser des Ziel-Ellipsoids	%-8s%12.3f

Tabelle 3

Beispiel:

```
NUM_OREC 11
NUM_SREC 11
NUM_FILE  1
GS_TYPE SECONDS
VERSION NTv2.0
SYSTEM_FDHDN90
SYSTEM_TETRS89
MAJOR_F  6377397.155
MINOR_F  6356078.963
MAJOR_T  6378137.000
MINOR_T  6356752.314
```

Der Subgitter-Header setzt sich wie folgt zusammen:

#	Kennung	Wert	Beschreibung	ASCII-Format
1	SUB_NAME	String	Name des Subgitters	%-8s%-8s
2	PARENT	String	Name des Elterngitters (NONE)	%-8s%-8s
3	CREATED	String	Erstellungsdatum YY-MM-DD	%-8s%-8s

#	Kennung	Wert	Beschreibung	ASCII-Format
4	UPDATED	String	Letzte Änderung YY-MM-DD	%-8s%-8s
5	S_LAT	Double	Südliche Begrenzung (Einheit GS_TYPE)	%-8s%15.6f
6	N_LAT	Double	Nördliche Begrenzung	%-8s%15.6f
7	E_LONG	Double	Östliche Begrenzung	%-8s%15.6f
8	W_LONG	Double	Westliche Begrenzung	%-8s%15.6f
9	LAT_INC	Double	Gitterabstand Breite	%-8s%15.6f
10	LONG_INC	Double	Gitterabstand Länge	%-8s%15.6f
11	GS_COUNT	Integer	Anzahl Gitterwerte	%-8s%6d

Tabelle 4

Beispiel:

```
SUB_NAMEDHDN90
PARENT  NONE
CREATED 06-11-09
UPDATED 06-11-09
S_LAT     169200.000000
N_LAT     199080.000000
E_LONG    -56400.000000
W_LONG    -19800.000000
LAT_INC      360.000000
LONG_INC     600.000000
GS_COUNT  5208
```

Die Ablage der Shift- und Genauigkeitswerte erfolgt zeilenweise unmittelbar im Anschluss an den Subgitter-Header, beginnend mit der südöstlichen Ecke. Jeder Satz besteht aus 4 Float-Werten ohne Kennung:

Feld	Kennung	Wert	Beschreibung	ASCII-Format
1		Float	Shiftwert Breite	`%10.6f`
2		Float	Shiftwert Länge	`%10.6f`
3		Float	Genauigkeit Breite	`%10.6f`
4		Float	Genauigkeit Länge	`%10.6f`

Tabelle 5

Beispiel:

```
  -4.979990  5.288690  0.000000  0.000000
  -4.982930  5.194760  0.000000  0.000000
  -4.985970  5.100410  0.000000  0.000000
  -4.990572  5.005587  0.000000  0.000000
  -5.001711  4.914125  0.000000  0.000000
  -5.010515  4.825731  0.000000  0.000000
  -5.012240  4.735130  0.000000  0.000000
  -5.013570  4.644530  0.000000  0.000000
  -5.014720  4.553790  0.000000  0.000000
  -5.015870  4.462900  0.000000  0.000000
```

Den Abschluss der Datei markiert der Ende-Satz:

Kennung	Wert	Beschreibung	ASCII-Format
`END`	String	Ende-Satz	`%-8s%-8s`

Auch wenn die Originaldokumentation es nicht ausdrücklich vorschreibt, so gilt die Empfehlung, die restlichen Bytes des Satzes (neben den 8 Bytes für die Kennung) mit 0-Bytes (binäre Variante) bzw. Leerzeichen (ASCII Variante) aufzufüllen.

2.2.3.2 Transformationsverfahren

Die unter 2.2.3.1 beschriebene Gitterdatei ist aus Sicht des Transformationsverfahrens vom Quellsystem (`SYSTEM_F`) in das Zielsystem (`SYSTEM_T`) aufgestellt (Hintransformation). Für die Rücktransformation wird kein eigenes Gitter aufgebaut, sondern ein Verfahren angewendet, das die dafür notwendigen Shiftwerte aus dem Gitter für die Hintransformation iterativ herleitet. Die

Beschreibung des Transformationsverfahrens erfolgt daher getrennt für Hin- und Rücktransformation.

2.2.3.2.1 Transformation Quellsystem nach Zielsystem (Hintransformation)

Bei der Transformation eines Punktes `q` mit den Koordinaten `{qlon qlat}` aus dem Quellsystem in das Zielsystem ist darauf zu achten, den Wert `qlon` für Positionen östlicher Länge mit einem negativen Vorzeichen zu versehen!

Der erste Schritt ist die Suche des zuständigen Subgitters. BeTA verwendet nur ein einziges Subgitter ohne weitere Verdichtungsgitter, so dass diese Suche nicht erforderlich ist (für die formale Beschreibung der Suche siehe [JuFa95]). Ein Subgitter ist für die Transformation eines Punktes zuständig, wenn der Punkt folgende Bedingung erfüllt:

```
E_LONG <= qlon <= W_LONG
S_LAT  <= qlat <= N_LAT
```

Die für die Interpolation benötigte Gittermasche kann durch:

```
fcol = (qlon – E_LONG) / LONG_INC
frow = (qlat – S_LAT)  / LAT_INC
col  = floor (fcol)
row  = floor (frow)
ppr  = floor ((W_LONG – E_LONG) / LONG_INC + 0.5) + 1
ppc  = floor ((N_LAT  – S_LAT)  / LAT_INC + 0.5)  + 1
se   = row * ppr + col
```

ermittelt werden. Die Variable `se` verweist dann auf die Nummer des Satzes der rechten unteren Ecke (Südost) der Gittermasche, beginnend hinter dem Subgitter-Header[1]. Die Satznummern der restlichen 3 Punkte erhält man durch

```
sw = se + 1             /* unten links, suedwest */
ne = se + ppr           /* oben rechts, nordost  */
nw = ne + 1              /* oben links,  nordwest */
```

Für die eindeutige Zuordnung der Gitterpunkte zu einem Gitterfeld gelten folgende Regeln:

- Punkte auf der unteren Schranke (Süd und Ost) liegen **innerhalb**
- Punkte auf der oberen Schranke (Nord und West) liegen **außerhalb**

Eine Ausnahme hiervon bilden die Punkte auf den oberen Limits (Nord- bzw. Westrand) des Subgitters. Diese werden dem jeweiligen Randgitterfeld zugeordnet. Dies erlaubt die Transforma-

[1] Achtung: Die Algorithmen basieren auf der Annahme, dass die Satz-Zählung bei 0 (C-Indizierung) beginnt!

tion von Punkten, die auf dem Nord- bzw. Westrand des Subgitters liegen. Für diese Punkte ist folgende Anpassung der Satznummern notwendig:

```
if (col >= ppr - 1) {   /* Punkt auf dem westl. Rand */
   sw = se
   nw = sw
}
if {row >= ppc - 1} {   /* Punkt auf dem noerdl. Rand */
   ne = se
   nw = sw
}
```

Die Shiftwerte für die konkrete Position des Punktes q auf Basis der 4 ermittelten Wertepaare erhält man über jeweils eine bilineare Interpolation für die Längen- und Breiten-Werte:

```
sse : Shiftwert Südost Ecke       (Länge oder Breite)
ssw : Shiftwert Südwest Ecke         -     ″     -
sne : Shiftwert Nordost Ecke         -     ″     -
snw : Shiftwert Nordwest Ecke        -     ″     -

dx  = fcol – floor (fcol)
dy  = frow – floor (frow)

sv =   (1 – dx) * (1 – dy) * sse
     +      dx  * (1 – dy) * ssw
     + (1 – dx) *      dy  * sne
     +      dx  *      dy  * snw
```

Hiernach beinhaltet sv den zu addierenden Shiftwert (Länge oder Breite, je nach Belegung der Ausgangswerte).

2.2.3.2.2 Transformation Zielsystem nach Quellsystem (Rücktransformation)

Die Ablage der Shiftwerte in der Gitterdatei erfolgt im Quellsystem. Für die Rücktransformation können die Shiftwerte daher nicht, wie unter 2.2.3.2.1 beschrieben, *direkt* berechnet sondern müssen über ein iteratives Verfahren ermittelt werden:

```
(1)  Initialisiere Quellsystemkoordinaten und Shiftwerte
   qlat = zlat
   qlon = zlon
(2)  Wiederhole 4 mal:
      a. Berechne Shiftwerte nach 2.2.3.2.1
         {slat slon} <- berechneShift (qlat, qlon)
      b. Subtraktion der Shiftwerte
         qlat = zlat – slat
         qlon = zlon - slon
(3)  Ermittelte Quellsystemkoordinaten:
   {qlat qlon}
```

Angesichts dieses Ablaufs sei darauf hingewiesen, dass nicht jede Koordinate aus einer Hintransformation eine Rücktransformation mit dem gleichen (Sub-)Gitter zulässt. Koordinaten in der Nähe des Gitterandes können bei der Transformation in das Zielsystem aus der Gitterbegrenzung herausgefallen sein.

3 Realisierung

3.1 Integration der Gitterdatei in ein GIS mit NTv2 Unterstützung

Einige marktverfügbare GI-Systeme unterstützen bereits einen auf NTv2 beruhenden Bezugssystemwechsel und erlauben die Transformationen geotopographischer Daten nach einem BeTA konformen Ansatz. Die Integration der NTv2-Gitterdatei ist von System zu System unterschiedlich. In der Regel erfolgt sie durch Ablage der Gitterdatei in einem speziellen Verzeichnis und/oder der Registrierung im GIS eigenen Transformations-Modul. Einzelheiten hierüber und der anschließenden Anwendung der Transformation auf reale Datenbestände sind der produktspezifischen Dokumentation und den Informationsquellen des jeweiligen Produkts (Webseite des Herstellers, Foren, Wiki, etc.) zu entnehmen.

3.2 Verwendung der OpenSource-Bibliothek PROJ.4

Steht in einer GIS-Produktionsumgebung keine NTv2-Unterstützung zur Verfügung, muss nach einer Möglichkeit gesucht werden, das System um diese Fähigkeit zu erweitern. Eine umfassende Analyse ist aufgrund der Vielgestaltigkeit der Softwareumgebungen nicht praktikabel und würde, auch bei Beschränkung auf einige beispielhafte Lösungen, den Rahmen dieses Dokuments sprengen. Stellvertretend wird unter Zuhilfenahme der OpenSource-Bibliothek PROJ.4 gezeigt, wie die Lösung der Grundaufgabe „Transformation eines Punktes“ aussieht, die als Start für eine spezialisierte Integrationslösung dienen kann.

Die Library PROJ.4 (http://proj.maptools.org/) ist eine OpenSource C-Library unter MIT-Lizenz (siehe [MITLIC]) und bildet die Grundlage für die Transformationslösungen zahlreicher anderer Projekte[2]. Zusätzlich zu den Programmquellen sind auf der Homepage auch vorbereitete binäre Programme und Bibliotheken für einige verbreitete Plattformen verfügbar. Die folgenden Beispiele wurden mit einer UNIX-Installation durchgeführt, sollten aber unter Anpassung der Zeilenumbruch-Mimik auch auf anderen Plattformen nachvollziehbar sein.

3.2.1 Das Kommandozeilenprogramm cs2cs

Für die Transformation einzelner Koordinaten bzw. Koordinatenlisten steht nach der Installation das Kommandozeilenprogramm `cs2cs` zur Verfügung, welches entsprechend parametrisiert bereits für BeTA konforme Transformationen zu nutzen ist. Als Beispiel sollen die Punkte:

1) `2490000.00R 5652000.00H` und
2) `2504000.00R 5628000.00H`

nach ETRS89/UTM transformiert werden. Die NTv2-Gitterdatei steht unter dem Namen `BETA2007.gsb` im Arbeitsverzeichnis zur Verfügung:

```
>>> cs2cs \
      +proj=tmerc +lat_0=0 +lon_0=6 +k=1.000000 \
      +x_0=2500000 +y_0=0 +ellps=bessel +units=m \
      +nadgrids=./BETA2007.gsb \
   +to \
      +proj=utm +ellps=GRS80 +zone=32 +nadgrids=@null<CR>
2490000.00   5652000.00<CR>
```

[2] PROJ.4 dient als Transformationsgrundlage für OpenSource-Projekte wie z.B. FWTools [http://fwtools.maptools.org/] und UMN Mapserver [http://mapserver.gis.umn.edu/]. Durch Installation dieser Pakete stehen auch die PROJ.4 Komponenten zur Verfügung.

```
279488.01       5654871.71 0.00   ;# Programmoutput, als dritter Wert
2504000.00   5628000.00<CR>       ;# wird die Höhe ausgegeben!
292503.36       5630318.18 0.00
...
```

Ein wichtiger Hinweis im Umgang mit `cs2cs` und anderen Programmen im PROJ.4-Umfeld:

> Die Parametrisierung erfolgt nicht immer intuitiv und ist mit Vorsicht einzusetzen. Insbesondere erfolgt keine Meldung, wenn Parameter falsch oder ganz ignoriert werden. Fehlt z.B. die mit dem Parameter `+nadgrids=...` spezifizierte Datei, erfolgt *keine* Fehlermeldung. Um etwas mehr Informationen über die internen Vorgänge zu erhalten, kann zu Testzwecken die Umgebungsvariable `PROJ_DEBUG` vor dem Programmstart auf einen beliebigen Wert gesetzt werden.

Da `cs2cs` als sog. Filter ausgelegt ist, lassen sich auch größere Koordinatenmengen, die in einer Datei abgelegt sind, durch einen Aufruf verarbeiten:

```
>>> cs2cs \
      +proj=tmerc +lat_0=0 +lon_0=6 +k=1.000000 \
      +x_0=2500000 +y_0=0 +ellps=bessel +units=m \
      +nadgrids=./BETA2007.gsb \
   +to \
      +proj=utm +ellps=GRS80 +zone=32 +nadgrids=@null\
   <Dhdn90Gk2.dat >Etrs89Utm32.dat<CR>
```

Transformiert alle Punkte aus `Dhdn90Gk2.dat` und schreibt das Ergebnis nach `Etrs89Utm32.dat`.

Über zusätzliche Optionen, die im Gegensatz zu den mit einem `"+"` beginnenden Transformationsparametern, mit einem `"-"` beginnen, lässt sich der Programmoutput beeinflussen. Die Option `-I` erzielt bei unveränderten Transformationsparametern die Berechnung der Rücktransformation. Mittels der Option `-f` kann das Ausgabeformat der berechneten Koordinaten geändert werden. Die Angabe `-f "%.6f"` bewirkt die Ausgabe der Koordinaten mit 6 Nachkommastellen.

3.2.2 Nutzung der Programmbibliothek

Neben den Kommandozeilenprogrammen bietet PROJ.4 eine Bibliothek mit C-Funktionen, die zur Nutzung in eigenen Programmumgebungen bereit stehen. Zur Durchführung einer zu BeTA konformen Transformation benötigt man lediglich die Funktionen `pj_init()`und `pj_transform()` aus dieser Bibliothek. Die Parameterübergabe an `pj_init()`erfolgt als Array aus Zeichenketten und ist bis auf die fehlenden +-Zeichen mit den `cs2cs` Parametern iden-

tisch. Ein kurzes Programm, welches exemplarisch die Koordinate 1) aus Abschnitt 3.2.1 ebenfalls nach ETRS89/UTM überführt, findet sich in Abschnitt 6.1.3.

Über den direkten Zugriff auf die C-Bibliothek ist auch die Einbettung der Transformationsfunktionalität in Skript-Sprachen wie Perl, Python, Tcl/Tk, etc. möglich. Für Programmentwickler, die diese Funktionalität benötigen, ist eine Recherche empfehlenswert, die feststellt, ob das gewünschte Wrapper-Interface, welches die Funktionalität in der Skript-Sprache zur Verfügung stellt, bereits in einem anderen Projekt enthalten ist.

3.3 Eigenentwicklung gemäß Spezifikation

Kommt keine der bisherigen Alternativen für die Einbindung von BeTA in die eigene Produktionsumgebung in Frage, weil z.B. der Einsatz von Fremdsoftware unerwünscht oder keine entsprechende Softwareumgebung vorliegt, ist eine Neuimplementierung in Betracht zu ziehen. Aufgrund des einfachen Ansatzes sollte dies in einem akzeptablen Zeitrahmen zu bewältigen sein. Als Leitfaden kann vorliegende Dokumentation dienen. Zusätzlich wird die englischsprachige Originaldokumentationen [JuFa95] bzw. [MiCo00] empfohlen.

4 Daten

Die Realisierung des Transformationsverfahrens erfolgt durch die Bereitstellung der relevanten Dateien zum Download im Informationssystem CRS-EU in deutscher Sprache direkt unter http://www.crs-geo.eu/BeTA2007:

- `BETA2007dokumentationV13.pdf`
  Vorliegende Datei mit der Dokumentation des Verfahrens.
- `BETA2007.gsb,BETA2007.gsa`
  Binäre und ASCII Gitterdatei im NTv2 Format.
- `BETA2007testdaten.csv`
  Datei mit Testdaten zur Verifizierung von Transformationslösungen.

Es folgt eine Beschreibung der beiden letztgenannten Datensätze.

4.1 Gitterdatei

Die Gitterdatei im NTv2-Format (`BETA2007.gsb`) definiert den Bezugssystemwechsel DHDN90 (`SYSTEM_F`) nach ETRS89 (`SYSTEM_T`), besteht aus einem einzigen Subgitter und hat eine Ausdehnung

- von 5˚30’ bis 15˚40’ östliche Länge und
- von 47˚00’ bis 55˚18’ nördlicher Breite

in einem Raster von 10’ * 6’ (Länge * Breite). Dies entspricht einem Bereich der die fiktiven TK25-Blätter von 07(-01)[3] (links oben) bis 8959 (rechts unten) umfasst. Laut 2.2.3 können nur Koordinaten, die innerhalb dieser Begrenzung liegen, transformiert werden.

Die Shiftwerte wurden aus den Transformationsmodellen der einzelnen Bundesländer berechnet. Um eine bundesweit homogene Gitterdatei zu erhalten und damit Unstetigkeiten an den Landesgrenzen zu vermeiden, wurden die abweichenden Shiftwerte der Gitterpunkte in der Nähe der Landesgrenzen gewichtet nach dem Flächenverhältnis gemittelt, so dass für jeden Gitterpunkt nur ein Shiftwert besteht. Innerhalb des Gebietes der Bundesrepublik resultiert hieraus eine Genauigkeit im Submeterbereich. Außerhalb wurde der Datensatz durch Extrapolation ergänzt. Auf die Belegung der Genauigkeitswerte in der NTv2-Gitterdatei wurde verzichtet, d.h. diese Positionen sind mit dem Wert `0.0` belegt.

4.2 Testdaten

Zur Verifikation einer Transformationslösung enthält die Datei `BETA2007testdaten.csv` Testdatensätze zu 31 Punkten. Diese Punkte wurden in verschiedenen Systemen (Meridianstreifen/Zonen) mit der Software aus 3.2 und der Gitterdatei `BETA2007.gsb` bestimmt. Eine Übersicht über die Verteilung der Testpunkte gibt die Grafik unter 6.3.2.

Die Datei hat die Form einer 8-spaltigen Tabelle und ist im Format `.csv` (*comma separated values*) abgelegt. Jeder Datensatz belegt eine Zeile, Dezimaltrenner bei Gleitkommazahlen ist der Punkt. Alle Datensätze haben den gleichen Inhalt, mit Ausnahme des ersten Datensatzes, der für jede Spalte einen kurzen Erläuterungstext enthält. Jeder Datensatz protokolliert die Transformation eines Punktes von einem Quell-System in ein Ziel-System. Zu jedem Punkt können mehrere Datensätze vorhanden sein, die durch eine laufende Nummer durchnummeriert sind.

[3] Die linke Begrenzung liegt östlich des Ursprungs der TK25-Blattbezeichnungen, so dass sich hier ein negativer Index ergibt.

Die Bezeichnung der Koordinatenreferenzsysteme entspricht [GID60K7]. Referenz für jeden Punkt ist seine Koordinate im System `DE_DHDN_Lat-Lon`, d.h. die geographischen Koordinaten bezogen auf DHDN90. Daher enthält der erste Datensatz mit laufender Nummer 1 immer die Abbildung `DE_DHDN_Lat-Lon` nach `ETRS89_Lat-Lon`, um die Referenz-Koordinate auszuweisen und einen Datensatz zum Testen des reinen NTv2-Shift-Ansatzes zur Verfügung zu stellen. Für jeden weiteren Datensatz wird zunächst die geographische Referenzkoordinate in das Quell-System umgerechnet und anschließend die Transformation in das Ziel-System vorgenommen. Alle Berechnungen wurden mit den Kommandozeilen-Programmen aus der PROJ.4-Library (`cs2cs` bzw. `proj`) in der Version 4.4.7 und dem `BETA2007.gsb` Gitter durchgeführt. Die Ablage der Gauß-Krüger- und UTM-Werte erfolgte in Meter mit einer Genauigkeit von 6 Nachkommastellen, die der geographischen Koordinaten in Dezimal-Grad mit 12 Nachkommastellen. Dies stellt keine absolute Genauigkeit dar, sondern dient lediglich der Verifizierung spezifischer Lösungen im Hinblick auf die Rechengenauigkeit! Aus historischen Gründen sind die Koordinaten im System DE_DHDN_3GKx, also die Gauß-Krüger-Werte, **mit** führender Meridiankennziffer abgelegt. Die unter 6.1.2.3 aufgeführten Formeln weisen diese nicht mit aus!

Die Spalten der Datensätze haben folgenden Inhalt:

Spalte	Inhalt
1	Punkt-Nummer Alle Datensätze mit identischer Punktnummer beziehen sich auf den gleichen Punkt
2	Laufende Nummer zum Punkt, wobei unter Nummer 1 immer die Abbildung `DE_DHDN_Lat-Lon` nach `ETRS89_Lat-Lon` abgelegt ist, um die Referenz-Koordinate auszuweisen und einen Datensatz zum Testen des reinen NTv2-Shift-Ansatzes bereitzustellen
3	Quell-Bezugssystem nach [GID06K7]
4	Geographische Länge bzw. Gauß-Krüger Rechtswert im Quellsystem
5	Geographische Breite bzw. Gauß-Krüger Hochwert im Quellsystem
6	Ziel-Bezugssystem nach [GID60K7]
7	Geographische Länge bzw. UTM East-Wert im Zielsystem
8	Geographische Breite bzw. UTM North-Wert im Zielsystem

Die Ausgabe des Rechtswertes der Gauß-Krüger-Koordinaten erfolgte mit vorangestellter Meridianstreifenkennung. Bei den UTM-Koordinaten wurde auf die Ausgabe der Zonenkennung verzichtet.

Beispielsätze für die ersten 2 Punkte in tabellarischer Form:

Pkt-Nr.	Lfd-Nr.	Quell-Bezugssystem	Quell-X	Quell-Y	Ziel-Bezugssystem	Ziel-X	Ziel-Y
1	1	DE_DHDN_Lat-Lon	7.483333333333	53.500000000000	ETRS89_Lat-Lon	7.482506019176	53.498461143331
1	2	DE_DHDN_3GK2	2598417.333192	5930677.980308	ETRS89_UTM32	399340.601863	5928794.177992
1	3	DE_DHDN_3GK3	3399371.190396	5930724.531323	ETRS89_UTM32	399340.601862	5928794.177992
2	1	DE_DHDN_Lat-Lon	10.466666666667	52.500000000000	ETRS89_Lat-Lon	10.465380298337	52.498573633365
2	2	DE_DHDN_3GK3	3599586.686397	5819391.659845	ETRS89_UTM32	599474.934168	5817502.626999
2	3	DE_DHDN_3GK4	4395886.918912	5819485.694352	ETRS89_UTM32	599474.934169	5817502.626999

5 Literatur

[Goit05] Government of Ontario IT Standards (GO-ITS). *NTv2 (National Transformation Version 2)*. 2005-06-14 (gesichtet 2006-09-18) http://www.geod.nrcan.gc.ca/pdf/ntv2_guide_e.pdf

[GID60K7] AdV: *Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok)*, Hauptdokument, Kapitel 7, Koordinatenreferenzsysteme und Maßeinheiten, Version 6.0 vom 11.04.2008

[Gros64] Großmann, W.: *Geodätische Rechnungen und Abbildungen in der Landesvermessung*, Verlag Konrad Wittwer, Stuttgart 1964

[Hris43] Hristow, W.K.: *Die Gauss-Krüger'schen Koordinaten auf dem Ellipsoid*, Verlag B. G. Teubner, Leipzig 1943

[IEEE85] IEEE *Standard for Binary Floating-Point Arithmetic for microprocessor systems* (ANSI/IEEE Std 754-1985); (gesichtet 2007-01-04) http://754r.ucbtest.org/standards/754.pdf

[Imnw99] Innenministerium Nordrhein-Westfalen: *Das Trigonometrische Festpunktfeld in Nordrhein-Westfalen* (TP-Erl.), RdErl. d. Innenministeriums v. 22. 07. 1999 – III C 3 – 4213

[Jord58] Jordan; Eggert; Kneissl: *Handbuch der Vermessungskunde, Band IV*, Erste Hälfte, Metzlersche Verlagsbuchhandlung, Stuttgart 1958

[Jord59] Jordan; Eggert; Kneissl: *Handbuch der Vermessungskunde, Band IV*, Zweite Hälfte, Metzlersche Verlagsbuchhandlung, Stuttgart 1959

[JuFa95] Junkins, D.R.; Farley, S.A.: *NTv2 National Transformation Version 2 – Developer's Guide*; 1995 (enthalten in [Goit05])

[MiCo00] Mitchell, D.J.; Collier, P.A.: *GDAit (GDA94 InTerpolation) – Software Documentation*; Version 2.0 2000-04 (gesichtet 2006-09-18) http://www.geom.unimelb.edu.au/gda94/SoftDoc.pdf

[MITLIC] (gesichtet 2007-01-04) http://www.opensource.org/licenses/mit-license.php

[Proj4] Programmbibliothek PROJ.4, http://proj.maptools.org

[Scho81] Schoedlbauer, A.: *Rechenformeln und Rechenbeispiele zur Landesvermessung*, Teil 1; Herbert Wichmann Verlag, Karlsruhe 1981

[Scho82] Schoedlbauer, A.: *Rechenformeln und Rechenbeispiele zur Landesvermessung*, Teil 2; Herbert Wichmann Verlag, Karlsruhe 1982

6 Anhang

6.1 Formeln und Programme

6.1.1 Pseudocode

Alle Formeln sind in einem der Programmiersprachen C bzw. FORTRAN angelehnten Pseudocode abgefasst. Zur Erläuterung folgt eine kurze Übersicht über die mathematischen Operatoren, auch wenn die Codefragmente größtenteils selbstdokumentierend sind.

```
      a * b : a multipliziert mit b
      a / b : a dividiert durch b
      a **b : a hoch b
   f**b (a) : f(a) hoch b
   sqrt (a) : Quadratwurzel aus a
    sin (a) : Sinus von a                    # a im Bogenmaß
    cos (a) : Cosinus von a                  #    "
    tan (a) : Tangens von a                  #    "
  floor (a) : Größte Integerzahl <= a
```

6.1.2 Umrechnung ebener konformer in geographische Koordinaten und umgekehrt

Wie bereits in 2.1 beschrieben, basiert die Umrechnung zwischen ebenen konformen Koordinaten einer Gauß-Krüger- bzw. UTM-Abbildung und ellipsoidischen geographischen Koordinaten auf einem identischen mathematischen Ansatz. Im Folgenden ist ein Ansatz, der aus verschiedenen Quellen abgeleitet wurde, wiedergegeben.

6.1.2.1 Bezeichnungen

```
R            =   Rechts- bzw. Ost-Wert der konformen Koordinaten (m)
                 ohne Meridianstreifen- bzw. Zonenkennziffer
H            =   Hoch- bzw. Nord-Wert der konformen Koordinaten (m)
B            =   geographische Breite (Grad)
L            =   geographische Länge (Grad)
a            =   große Halbachse des Ellipsoides (m)
b            =   kleine Halbachse des Ellipsoides (m)
             =   a * (1 - f)
e'**2        =   Quadrat der 2. numerischen Exzentrizität
f            =   Abplattung
Eo bis ED    =   Konstanten zur Berechnung der Meridianbogenlänge
rho          =   57.295779513082321 (Grad)

Parameter der Abbildungssysteme:
Dm           =   Abstand der Mittelmeridiane (Grad)
Kz           =   Meridianstreifen- bzw. Zonenkennziffer
kr           =   konstanter Zuschlag zum Rechtswert (m)
kh           =   konstanter Zuschlag zum Hochwert (m)
kl           =   konstanter Zuschlag zur geographischen Länge (Grad)
Mh           =   Maßstabsfaktor des Abbildungssystems
```

6.1.2.2 Parameter

6.1.2.2.1 Ellipsoid

Siehe [Imnw99].

```
Bessel
a            =   6377397.155
f            =   1 / 299.15281285

GRS80-Erdellipsoid
a            =   6378137.0 m
f            =   1 / 298.257222101
```

6.1.2.2.2 Abbildungssysteme

```
Gauß-Krüger-System
Dm            =   3
kl            =   0
kr            =   500000
kh            =   0
Mh            =   1

UTM-System Nord
(für Deutschland: 32, 33, ...)
Dm            =   6
kl            =   -183
kr            =   500000
kh            =   0
Mh            =   0.9996
```

6.1.2.3 Formeln

6.1.2.3.1 Abgeleitete Ellipsoidparameter

```
Krümmungshalbmesser des Ellipsoides am Pol
c             =   a**2 / b

Quadrat der 1. numerischen Exzentrizität des Ellipsoides
e**2          =   1 - b / c

Quadrat der 2. numerischen Exzentrizität des Ellipsoides
e'**2         = - 1 + c / b
```

6.1.2.3.2 Vorberechnete Konstanten

```
Konstanten zur Berechnung der Meridianbogenlängen
und der geographischen Breiten von Lotfußpunkten
(vergl. [Jord58])

Eo          =   1
              - e'**2  *     3 / 4
              + e'**4  *    45 / 64
              - e'**6  *   175 / 256
              + e'**8  * 11025 / 16384
              - e'**10 * 43659 / 65536

E2          = - e'**2  *     3 / 8
              + e'**4  *    15 / 32
              - e'**6  *   525 / 1024
              + e'**8  *  2205 / 4096
              - e'**10 * 72765 / 131072

E4          =   e'**4  *    15 / 256
              - e'**6  *   105 / 1024
              + e'**8  *  2205 / 16384
              - e'**10 * 10395 / 65536

E6          = - e'**6  *    35 / 3072
              + e'**8  *   315 / 12288
              - e'**10 * 31185 / 786432

E8          =   e'**8  *   315 / 131072
              - e'**10 *  3465 / 524288

ED          = - e'**10 *   693 / 1310720
```

6.1.2.3.3 Koordinatenumrechnungen

```
Gaußsche konforme in geographische Koordinaten
Meridianbogen nach [Jord58], Seite 79, siehe auch [Scho81], Seite 14
Iteration nach [Proj4]
Meridian- und Querkrümmungshalbmesser nach [Gros64], Seite 10
Geographische Länge nach [Jord59], Seiten 1109 und 1106 oder
[Hris43], Seite 47, siehe auch [Scho82], Seite 80
Geographische Breite sowie Auflösung der Konstanten und Faktoren
nach [Proj4]

Input:  R        Rechts- bzw. Ostwert (m)
                 ohne Meridianstreifen- bzw. Zonenkennziffer
        H        Hoch- bzw. Nordwert (m)
        Kz       Meridianstreifen- bzw. Zonenkennziffer
Output: B        geografische Breite (Grad)
        L        geografische Länge (Grad)

y            =  (R - kr) / Mh

Bfr          =  (H / Mh) / a

Wiederhole {
   x0           =   c * (Eo * Bfr +
                         E2 * sin(2*Bfr) +
                         E4 * sin(4*Bfr) +
                         E6 * sin(6*Bfr) +
                         E8 * sin(8*Bfr) +
                         ED * sin(10*Bfr))
   Dx0          =  (H / Mh) - x0
   n2           =   e'**2 * cos**2(Bfr)
   M            =   c / sqrt(1 + n2)**3
   Bfr          =   Bfr + Dx0 / M
} solange (abs(Dx0) > 10**-7)
```

```
n2          =   e'**2 * cos**2(Bfr)
M           =   c / sqrt(1 + n2)**3
N           =   c / sqrt(1 + n2)
F1          =   y / N
F2          =   y**2 / N**2
t2          =   tan**2(Bfr)

DBr         =   N * tan(Bfr) *
                 F2 * (1 -
                  F2 * (5 + t2*(3 - 9*n2) + n2*(1 - 4*n2) -
                   F2 * (61 + t2*(90 - 252*n2 + 45*t2) + 46*n2 -
                    F2 * (1385 + t2*(3633 + t2*(4095 + 1574*t2))
                         )/56
                        )/30
                       )/12
                      )/2 /M

DLr         =   F1 * (1 -
                 F2 * (1 + 2*t2 + n2 -
                  F2 * (5 + t2*(28 + 24*t2 + 8*n2) + 6*n2 -
                   F2 * (61 + t2*(662 + t2*(1320 + 720 * t2))
                        )/42
                       )/20
                      )/6
                     )/cos(Bfr)

B           =  (Bfr - DBr) * rho
L           =   Kz * Dm + kl  +  DLr * rho
```

```
Geographische in Gaußsche konforme Koordinaten
Meridianbogen nach [Jord58], Seite 79, siehe auch [Scho81], Seite 14
Meridian- und Querkrümmungshalbmesser nach [Gros64], Seite 10
Gaußsche Koordinaten nach [Jord59], Seite 1105,
siehe auch [Hris43], Seiten 47 und 48 oder [Scho82], Seite 21
Auflösung der Konstanten und Faktoren nach [Proj4]

Input:  B        geografische Breite (Grad)
        L        geografische Länge (Grad)
        Kz       gewünschte Meridianstreifen- bzw. Zonenkennziffer der
                 Ausgabekoordinaten
Output: R        Rechts- bzw. Ostwert (m)
        H        Hoch- bzw. Nordwert (m)

Br          =   B / rho

xo          =   c * (Eo * Br +
                     E2 * sin(2*Br) +
                     E4 * sin(4*Br) +
                     E6 * sin(6*Br) +
                     E8 * sin(8*Br) +
                     ED * sin(10*Br))

n2          =   e'**2 * cos**2(Br)
t2          =   tan**2(Br)
N           =   c / sqrt(1 + n2)
DLr         =  (L - Kz * Dm - kl) / rho
F1          =   cos(Br) * DLr
F2          =   cos**2(Br) * DLr**2
```

```
Dx            =   N * sin(Br) * DLr *
                   F1 * (1 +
                    F2 * (5 - t2 + n2*(9 + 4*n2) +
                     F2 * (61 + t2*(t2 - 58) + n2*(270 - 330*t2) +
                      F2 * (1385 + t2*(t2*(543 - t2) - 3111)
                           )/56
                          )/30
                         )/12
                        )/2

Dy            =   N *
                   F1 * (1 +
                    F2 * (1 - t2 + n2 +
                     F2 * (5 + t2*(t2 - 18) + n2*(14 - 58*t2) +
                      F2 * (61 +  t2*(t2*(179 - t2) - 479)
                           )/42
                          )/20
                         )/6)

H             =  (xo + Dx) * Mh
R             =   Dy * Mh + kr
```

6.1.3 Beispiel C-Programm zur Transformation mit PROJ.4

In folgendem Beispiel zur Transformation mit PROJ.4 wird der Einfachheit halber nur ein Punkt transformiert. Die verwendete Routine `pj_transform()` kann aber ein Koordinaten-Array entgegen nehmen und, entsprechend parametrisiert, alle Punkte in einem Aufruf transformieren. Analog zum Aufruf von `cs2cs` in 3.2.1 erfolgt die Definition der Parameter als Array aus Zeichenketten.

```
/*--------------------------------------------------------------*
 * Testprogramm PROJ.4 Library
 *--------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include "projects.h"
```

```
static projPJ   dhdn;
static projPJ   etrs;

int dhdnEtrsInit (void);
int dhdnEtrsTrf (double r, double h, double *pe, double *pn);

static char *pdhdn [] = {
   "proj=tmerc",
   "lat_0=0",
   "lon_0=6",
   "k=1.000000",
   "x_0=2500000",
   "y_0=0",
   "ellps=bessel",
   "nadgrids=./BETA2007.gsb",    /* Pfad zur NTv2 Gitterdatei */
   "units=m",
   "no_defs"
};

static char *petrs [] = {
   "proj=utm",
   "ellps=GRS80",
   "nadgrids=@null",
   "zone=32"
};

int dhdnEtrsInit (void) {
   if (! (dhdn = pj_init (sizeof (pdhdn) / sizeof (char *), pdhdn))) {
      fprintf (stderr,
         "Fehler bei der Initialisierung der Projektion DHDN:\n");
      fprintf (stderr, "   %s\n", pj_strerrno(pj_errno));
      exit (1);
   }
   if (! (etrs = pj_init (sizeof (petrs) / sizeof (char *), petrs))) {
      fprintf (stderr,
         "Fehler bei der Initialisierung der Projektion ETRS:\n");
      fprintf (stderr, "   %s\n", pj_strerrno(pj_errno));
      exit (1);
```

```
    }
    return 0;
}

int dhdnEtrsTrf (double r, double h, double *pe, double *pn) {
    double z = 0.0;    /* Dummy */
    *pe = r;
    *pn = h;

    if (pj_transform (dhdn, etrs, 1, 0, pe, pn, &z) != 0) {
        *pe = HUGE_VAL;
        *pn = HUGE_VAL;
    }
    return 0;
}

int main (int argc, char **argv)
{
    int rc = 0;

    double r = 2490000.00;
    double h = 5652000.00;
    double e, n;

    dhdnEtrsInit ();
    dhdnEtrsTrf (r, h, &e, &n);
    if (e == HUGE_VAL || n == HUGE_VAL) {
       fprintf (stderr, "Fehler bei der Transformation:\n");
       fprintf (stderr, "   %s\n", pj_strerrno(pj_errno));
       rc = 1;
    } else {
       printf ("%15.4f %15.4f ==> %15.4f %15.4f\n", r, h, e, n);
    }
    return rc;
}
```

Ist die PROJ.4-Library korrekt installiert, erfolgt das Übersetzen und Binden des Programms mittels:

```
cc -g -o dhdnEtrs dhdnEtrs.c -lproj –lm
```

Ein Start des Programms ergibt folgende Ausgabe (PROJ.4-Lib Version 4.4.7):

```
>>> dhdnEtrs
   2490000.0000    5652000.0000 ==>        279488.0076    5654871.7129
```

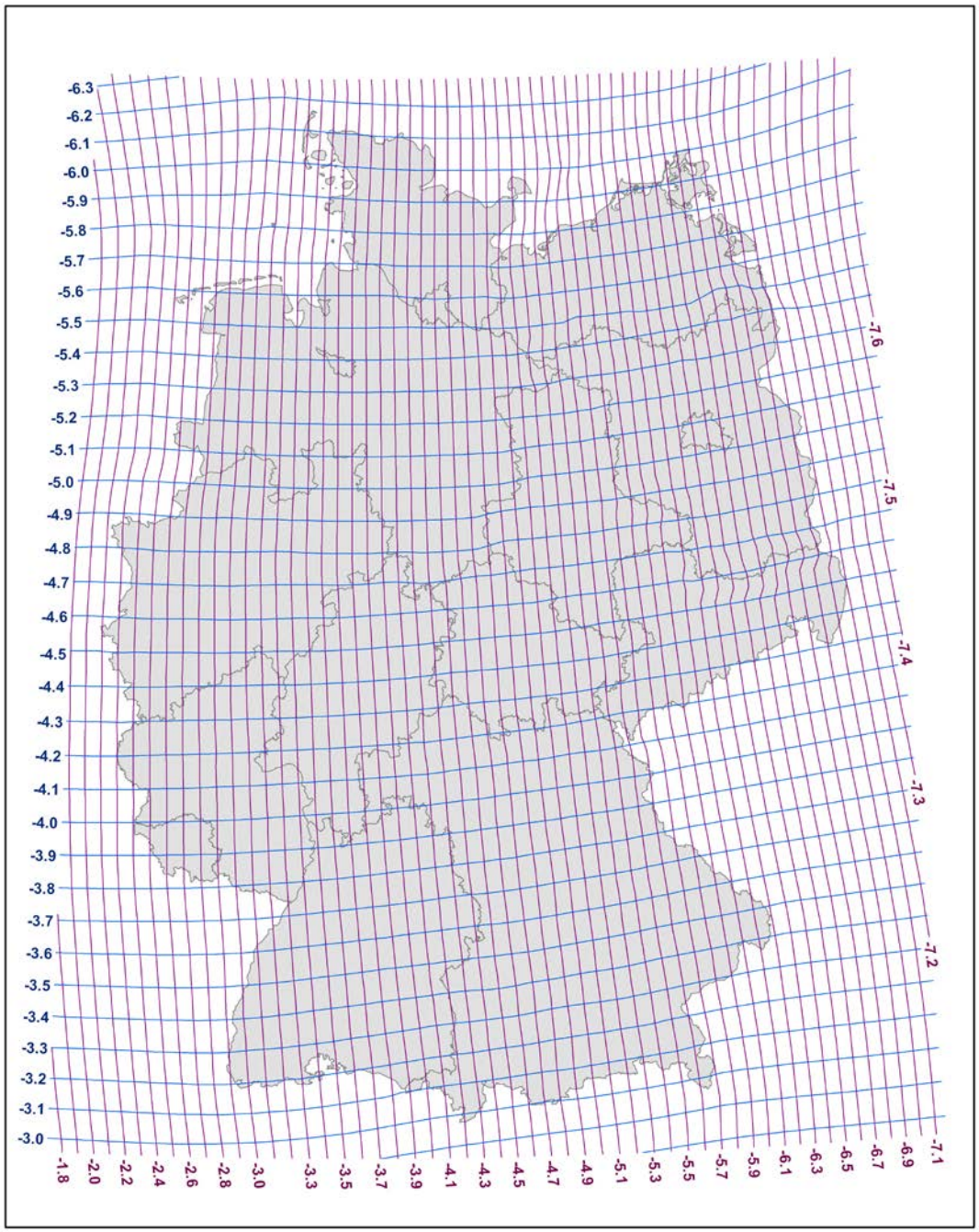
6.2 Abkürzungsverzeichnis

AdV	Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland
ASCII	American Standard Code for Information Interchange
ATKIS	Amtliches Topographisch-Kartographisches Informationssystem des amtlichen Vermessungswesens
BeTA	Bundeseinheitliche Transformation für ATKIS
ETRS89	European Terrestrial Reference System 1989
GI	Geoinformation
GIS	Geoinformationssystem
GRS80	Geodetic Reference System 1980, Referenzellipsoid des ETRS89
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
MIT-Lizenz	OpenSource Lizenz des Massachusetts Institute of Technology, siehe http://www.opensource.org/licenses/mit-license.php
NTv2	National Transformation Version 2

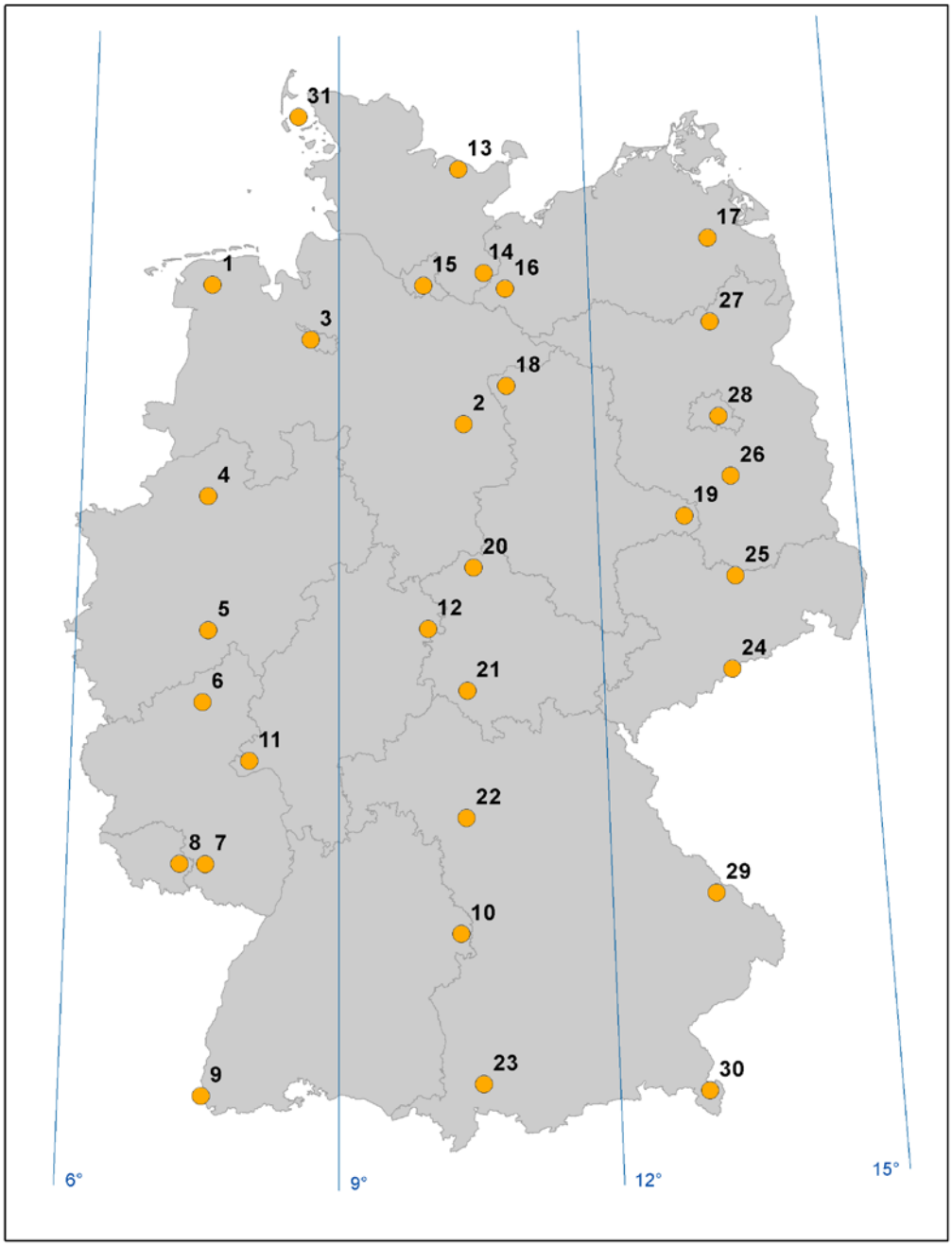
6.3 Grafiken

6.3.1 Isolinienmodell der Gitterdatei

Darstellung der Shiftwerte in Sekunden (Blau = ΔB, Rot = ΔL)



6.3.2 Lage der Punkte des Testdatensatzes



6.4 Änderungen

6.4.1 Version 1.1

- In Abschnitt 6.1.3 war die Ausgabe des Programms `dhdnEtrs` falsch wiedergegeben. Die Ausgabekoordinate wurde korrigiert.
- Ab der Version 4.4.6 führt die PROJ.4-Library eine Datumstransformation nur noch durch, wenn beim Zielsystem ein Datumsparameter angegeben ist. Da auch ältere Versionen mit diesem Parameter korrekt arbeiten, wurden die Aufrufe von `cs2cs` im Abschnitt 3.2.1 sowie die Parameterdeklaration im C-Programm `dhdnEtrs` in Abschnitt 6.1.3 um diesen Parameter ergänzt.

6.4.2 Version 1.2

- In Abschnitt 2.2.3.2.1 wurde der Sonderfall, dass ein Punkt auf den westlichen bzw. nördlichen Rand fällt, jetzt durch zwei zusätzliche `if`-Abfragen berücksichtigt.
- Bezug auf die GeoInfoDok 6.0

6.4.3 Version 1.3

- Auf Hinweis eines Anwenders wurden die Formeln in Abschnitt 6 unter anderem in folgenden Punkten überarbeitet:
    - Um Irritationen bei der Verwendung des Operators `**` bei Funktionen zu vermeiden, wurde die Bedeutung von `f**b (a)` explizit erläutert.
    - Eine wesentliche formale Änderung wurde bei den Routinen zur Koordinatenumrechnung (6.1.2.3.3) vorgenommen. Die Meridianstreifen- bzw. Zonenkennziffer `Kz` wird nun explizit über die Schnittstelle entgegengenommen. Hierdurch folgt die Dokumentation der, sich international als auch im AAA-Umfeld durchzusetzenden Konvention, die Kennziffer nicht dem Rechts- bzw. Ostwert voranzustellen. Diese Änderung hat nur Einfluss auf die Parametrisierung nicht auf den mathematischen Ansatz!
    - Im Zuge dessen wurde der Parametersatz "UTM-System Nord mit reduzierter Zonenkennziffer" entfernt.

- Die Erläuterung der trigonometrischen Funktionen wies `Grad` statt `Bogenmaß` als Einheit für das zu übergebende Argument aus.
- Da der Testdatensatz aus historischen Gründen die Gauß-Krüger-Rechtswerte noch mit vorangestellter Meridianstreifenkennziffer aufweist, wurde dies dokumentiert um Irritationen bzgl. der geänderten Umrechnungsroutinen vorzubeugen.

6.4.4 Version 1.4

- Auf Hinweis eines Anwenders wurden die Formeln in Abschnitt 6 in folgenden Punkten korrigiert:
  - Der Ausdruck zur Berechnung des Wertes `DLr` in Abschnitt 6.1.2.3.3 Unterpunkt "`Geographische in Gaußsche konforme Koordinaten`" muss korrekt lauten: `DLr = (L - Kz * Dm - kl) / rho`
  - Die Berechnung des Wertes `M` in Abschnitt 6.1.2.3.3 Unterpunkt "`Gaußsche konforme in geographische Koordinaten`" nach der Iteration macht nur Sinn, wenn zuvor `n2` erneut berechnet wird, obwohl sich die Werte nicht mehr signifikant ändern. Nun passen `Bfr`, `n2` und `M` streng zueinander.

6.4.5 Version 1.5

- Der Parameter `+datum=WGS84`, der zur Angabe des Datums bei der Transformation mit PROJ.4 dient, wurde in `+nadgrids=@null` geändert. Der ursprüngliche Parameter beinhaltete eine zusätzliche Ellipsoiddefinition, die unnötig und verwirrend ist. Die Spezifikation des Null-Gitters beim Zielsystem bewirkt die unveränderte Übernahme der Koordinaten des NTv2-Shifts in das Zielsystem.