AIRS

Der Weg der AIRS-Daten hinein in die Randwerte-Datei

Version 6

Informationen

- Die AIRS-Daten liegen im HDF-Format auf dem JUROPA (GPFS):
  - `ssh juropagpfs.fz-juelich.de`
  - im Verzeichnis:
    /gpfs/arch2/slmet/slmet000/airs/level2 bzw.
    /gpfs/arch2/slmet/slmet000/airs/level2_v6
  - ein Unterverzeichnis je Jahr mit jeweils einem tar-File pro Tag, jedes tar-File besteht aus 240 HDF-Files
- Anschauen der HDF-Files mit `hdfview` möglich
- Infos zu AIRS unter
  `http://disc.sci.gsfc.nasa.gov/services/AIRS/additional/tools.shtml`
- Wiki-Seite: `https://clams.icg.kfa-juelich.de/CLaMS/AIRS`
- Der Sourcecode und die Dokumentation kann mit CVS ausgecheckt werden:
  `cvs co -P airs`
- Unter `airs` vorhandene Unterverzeichnisse:
  - *nasa_doc*: AIRS Version 5 und Version 6 Dokumentation
  - *idlreader_v5* bzw. *idlreader_v6*:
    - `read_airs_swath.pro`: Lesen eines Level 1/2 granule data file im HDF-EOS swath format
    - `airs2ncdf.pro`: Konvertiere Airs-Datensatz nach Netcdf (jeweils ein DS für CO und CH4)
    - `plotairs.pro`: Plotte CO/CH4 aus Netcdf-Ds
  - *source_v5* und *source_v6*: Routinen, um AIRS-Datensätze nach Netcdf zu konvertieren und Boundfile zu erstellen
  - *doc*: Dokumentation zur Erstellung der Boundfiles (mit Hilfe der Routinen unter `source_v5` bzw. `source_v6`)

Schritt 1: Entpacken der AIRS-Daten und Konvertierung von HDF in NCDF

Mit dem Shell-Script `run_airs2ncdf.ksh` werden die archivierten AIRS-Daten vom Juropa-Server kopiert, die HDF-Dateien entpackt und mit der IDL-Routine `airs2ncdf.pro` verarbeitet. Die IDL-Routine liest alle HDF-Dateien und konvertiert sie in das NCDF-Format; es werden ζ und die Mittagszeit (noontime) sowie die Mittagspositionen (noonlon, noonlat), nachdem sie mit Trajektorienrechnung bestimmt wurden, hinzugefügt.

Shell-Skript `run_airs2ncdf.ksh`:

Um größere Datenmengen vom Juropa-Server zu holen, diese automatisch zu entpacken und anschließend die IDL-Routine `airs2ncdf.pro` zu öffnen, kann das Shell-Skript `run_airs2ncdf.ksh` genutzt werden. Das Programm verarbeitet zunächst einen kompletten Tag, ehe es mit dem nächsten fortschreitet. Die TAR- und HDF-Dateien werden, wenn sie nicht mehr benötigt werden, sofort wieder gelöscht, um nicht benötigten Speicherplatz freizugeben.

Vor dem Aufrufen des Shell-Skripts noch Folgendes beachten:

- Im Shell-Script folgende Variablen anpassen:
  - *start* und *end*: Start- und Endtag (von 1 bis 366) eines Jahres von dem Datensatz, der konvertiert werden soll. Normalerweise *start=1* und *end=367* für ein ganzes Jahr (der letzte Tag wird nicht mehr durchlaufen!)
  - Für mehrere Jahre die Schleife *year* entsprechend anpassen
  - *hdf_dir*: Das Verzeichnis, in das die TAR und HDF-Dateien vorübergehend abgelegt werden sollen
  - Userid auf JUROPA
  - Im Shell-Skript muss `airs2ncdf` für jede Spezies aufgerufen werden
- In IDL SAV-File (*airs2ncdf.sav*) erstellen:

```
.r airs2ncdf
resolve_all
save,/routines,filename='airs2ncdf.sav'
```

- ACHTUNG: Es tritt ein Fehler in IDL beim Aufruf von *write_ncdf* auf. Daher wurde eine lokale Version von *write_ncdf* abgelegt, in der die Ausgabe von Attributen auskommentiert wurde, um den Fehler zu umgehen.

IDL-Routine `airs2ncdf.pro`:

In den IDL-Routinen folgende Variablen anpassen:

- In `airs2ncdf.pro`:
  - *dir_in*: Verzeichnis, in dem die HDF-Dateien vorübergehend liegen
  - *dir_out*: Verzeichnis, in das die NCDF-Dateien konvertiert werden sollen
  - *quality*: Qualitätsfilter für die Daten (0: nur die besten Messungen, 1: auch schlechtere Messwerte werden konvertiert)

  - *dir_wind*: Verzeichnis der Wind-Daten für `add_zeta`

- In `add_zeta.pro`: Aufruf von `pos_add` anpassen
- In `add_noon_pos_airs.pro`:
  - Variable *prefix_wind* anpassen
  - evtl. Übergabeparameter für `calc_traj` ändern
- In `calc_traj.pro`: Aufruf von `traj` anpassen

HINWEIS: Diese IDL-Routine ist auch ohne Skript lauffähig, dafür müssen die Variablen *startdate*, *enddate* und *type* entsprechend verändert werden.

Diese Routine ruft folgende Programme/Unterroutinen auf, die nicht der icglib oder der CLaMS-suite entstammen:

- `get_dates.pro`
- `noontime.pro`
- `create_coordinates.pro`
- `add_zeta.pro` (ruft `clams/clams-tools/pos_add auf`)
- `add_noon_pos_airs.pro`
  - `calc_traj.pro` (ruft `clams/traj/traj` auf)
- `check_noonpos.pro`
- `lib_tmp.pro`

Die Bennenung der Dateien folgt dem Schema:

AIRS_*Spezies[CO/CH4]*_*Qualitätsfilter*_*(Mittagsposition)yyyymmddhh*.nc

Beispiel: AIRS_CO_flag0_2012010112.nc

IDL-Routine `check_airs.pro`:

Mit der IDL-Routine `check_airs` kann überprüft werden, wie viele und welche NCDF-Dateien nicht konvertiert worden, nicht vorhanden sind und an welchen Tagen keine 240 HDF-Dateien zur Verfügung stehen. Dazu müssen nur das Start- und Enddatum, sowie das Verzeichnis und die Dateinamen angepasst werden.

Schritt 2: Je 3 Tage zusammenfassen

Mit der IDL-Routine `mult_airs_synpos.pro` werden jeweils mehrere Tage zusammengefasst und die Mittagspositionen mittels Trajektorienrechnung auf einen synoptischen Zeitpunkt gebracht. Für größere Zeitspannen kann auch das Skript `run_airs_synpos.ksh` verwendet werden.

Shell-Skript `run_airs_synpos.ksh`:

Um IDL-Abstürze in `mult_airs_synpos.pro` zu vermeiden, kann das Shell-Skript `run_airs_synpos.ksh` genutzt werden. Dieses bearbeitet alle Datensätze in dem angegebenen Zeitraum und ruft das IDL-Programm für jeden dritten Tag erneut auf. Sollte das IDL-Programm abstürzen, wird mit dem nächsten Datum fortgefahren. Das IDL-Programm `mult_airs_synpos.pro` ist entsprechend angepasst und liest das aktuelle Datum von einer im Skript gesetzten Umgebungsvariablen.

- Im Skript:
  - Start- und Endzeit eintragen
  - Ein Aufruf von `mult_airs_synpos` pro Spezies
- Im PRO-File Variablen anpassen
- In IDL: SAV-File erstellen (*mult_airs_synpos.sav*)
- ACHTUNG: Es tritt ein Fehler beim Aufruf von *write_ncdf* auf. Daher wurde eine lokale Version von *write_ncdf* abgelegt, in der die Ausgabe von Attributen auskommentiert wurde, um den Fehler zu umgehen.
- Aufruf des Shell-Skripts

IDL-Routine `mult_airs_synpos.pro`:

In der IDL-Routine folgende Variablen anpassen:

- In `mult_airs_synpos.pro`:
  - *dir_in*: Verzeichnis mit den konvertierten NCDF-Dateien inkl. Mittagspositionen
  - *dir_syn*: Ausgabeverzeichnis
  - *prefix*: NCDF-Dateinamen ohne Datum
  - *species*
  - *days*: Anzahl der Tage um den synoptischen Zeitpunkt (wenn 3 Tage zusammengefasst werden sollen, muss *days=1* sein)
  - Wenn die Routine ohne Skript ausgeführt wird, müssen auch Start- und Enddatum (*datefirst* und *datelast*) angepasst werden.
- In `add_syn_pos_airs.pro`:
  - *input_dir*: Verzeichnis mit den Wind-Daten
  - *prefix_wind*

- In `calc_traj.pro`: Aufruf von `traj` anpassen

Diese Routine ruft folgende Programme/Routinen auf, die nicht der icglib oder der CLaMS-suite entstammen:

- `add_syn_pos_airs.pro`
  - `calc_traj.pro`

Die Benennung der Dateien folgt diesem Schema:

AIRS_*Spezies[CO/CH4]*_*Qualitätsfilter*_synpos_*(synoptische Postion)yyyymmddhh*.nc

Beispiel: AIRS_CO_flag0_synpos_2012010212.nc

IDL-Routine `calc_synpos_date.pro`:

Der nächste synoptische Zeitpunkt kann mit Hilfe dieser IDL-Routine ermittelt werden.

Schritt 3: Gridden der Daten

Die dem vorangegangenen Schritt entstammenden Daten werden mit der IDL-Routine `mult_synpos_gridding_airs.pro` auf ein 2×6 Breiten-Längen-Gitter gebracht und gewichtet gemittelt.

`mult_synpos_gridding_airs.pro`:

Diese Routine ruft folgende Programme/Routinen auf, die nicht der icglib oder der CLaMS-suite entstammen:

- `get_cbar.pro`
- `fill_missing_values.pro`

In der IDL-Routine folgende Variablen anpassen:

- In `mult_synpos_gridding_airs.pro`:
  - *jssyn*: erster Synoptischer Zeitpunkt
  - *jsmin*
  - *jsmax*
  - *jsend*
  - *pvfile*
  - *filenames*
  - *spec*
  - *out_file*

Die Benennung der Dateien folgt diesem Schema:

grid_*Spezies*_AIRS_(ζ)_*yymmddhh*.nc

Beispiel: grid_CO_AIRS_zeta_12010212.nc

Die Routine muss für jede Spezies angepasst und aufgerufen werden.

Schritt 4: Erzeugen von ζ-Flächen

Die IDL-Routine `grid2zeta.pro` interpoliert die CO-Werte auf ζ-Flächen.

`grid2zeta.pro`:

Für die Erstellung einer Randbedingungsdatei auf Druckniveaus entfällt dieser Schritt !

Diese Routine ruft folgende Programme/Routinen auf, die nicht der icglib oder der CLaMS-suite entstammen:

- `fill_missing_values.pro`

Folgende Variablen anpassen:

- In `grid2zeta.pro`:
  - *start_js*
  - *end_js*
  - *spec*
  - *file_press*: Input
  - *file_zeta*: Output

Die Benennung der Dateien folgt diesem Schema:

grid_*Spezies*_AIRS_ζ_(interpoliert)_`yymmddhh`.nc

Beispiel: grid_CO_AIRS_zeta_interpol_12010212.nc

Die Routine muss für jede Spezies angepasst und aufgerufen werden.

Schritt 5: Randbedingungsdatei

In diesem Schritt wird aus den zuvor erzeugten Dateien der gewünschte Höhenlevel extrahiert und in eine Datei geschrieben, die von `bmix` aus der CLaMS-suite verwendet werden kann.

Die IDL-Routine `make_mbound_interpol_airs.pro` kann sowohl für die Erstellung einer Randbedingungsdatei auf Drucklevel, als auch auf ζ-Level verwendet werden. Für Ersteres ist Schritt 4 nicht notwendig.

`make_mbound_interpol_airs.pro`:

Zum Ausführen des Programmes wird die Datei `create_mbound_struct.pro` benötigt.

- Allgemein müssen folgende Variablen angepasst werden:
  - *begindate* und *enddate*
  - *file_ out*
  - *specs*
- Für die Randbedingungsdatei auf Drucklevel:
  - *vert = 'press'* setzen
  - *lev*: Drucklevel
  - *all_files*
- Für die Randbedingungsdatei auf : ζ-Level
  - *vert = 'zeta'* setzen
  - *lev*: ζ-Level
  - *all_files*

Die Routine muss für jede Spezies angepasst und aufgerufen werden.

Laufzeiten

Auf icg1096:

Schritt 1	ca. 4 min pro Datensatz (1 Tag) und Spezies (+ ca.10min für das Holen der Daten) -> Für CO und CH4 pro Jahr: ca. 110 Std. (5 Tage)
Schritt 2	ca. 16 min pro Datensatz (3 Tage) und Spezies -> Für CO und CH4 pro Jahr: ca. 65 Std. (3 Tage)
Schritt 3	ca. 18 min pro Datensatz (3 Tage) und Spezies -> Für CO und CH4 pro Jahr: ca. 73 Std. (3 Tage)

Speicherplatzbedarf

HDF-Files	ca. 900 MB je Tag (240 Datensätze) -> pro Jahr: ca. 330 GB => Nach dem Konvertieren löschen (Datensätze liegen noch auf JUROPA)
NetCDF-Files mit NOONPOS (nach Schritt 1)	ca. 40 MB je Spezies und Datensatz -> Für CO und CH4 pro Jahr: ca. 30 GB
NetCDF-Files mit SYNPOS (nach Schritt 2)	ca. 180 MB je Spezies und Datensatz (für 3 Tage) -> Für CO und CH4 pro Jahr: ca. 44 GB
GRID-Files (nach Schritt 3)	ca. 0.5 MB je Spezies und Datensatz (für 3 Tage) -> Für CO und CH4 pro Jahr: ca. 120 MB
GRID-Files auf Zeta-Niveaus (nach Schritt 4)	ca. 0.5 MB je Spezies und Datensatz (für 3 Tage) -> Für CO und CH4 pro Jahr: ca. 120 MB

Vorhandene Daten

AIRS-Daten für 2012 und 2013:

1. Originaldaten in HDF-Format:
   juropagpfs:/gpfs/arch2/slmet/slmet000/airs/level2
   juropagpfs:/gpfs/arch2/slmet/slmet000/airs/level2_v6

2. Nach Netcdf konvertiert und NOONPOS hinzugefügt:
   icg1096:/private/icg112/airsdata/v6/noonpos/yyyy

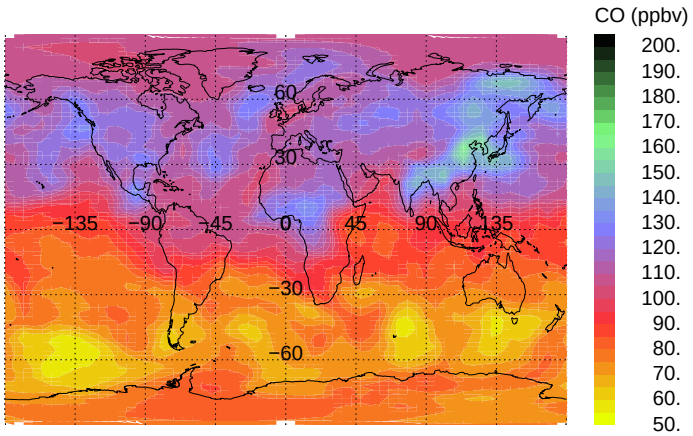
3. Jeweils 3 Tage zusammengefasst:
   icg1096:/private/icg112/airsdata/v6/synpos/yyyy

4. Auf 2x6 Breiten-Längen-Gitter:
   icg1096:/private/icg112/airsdata/v6/grid/yyyy

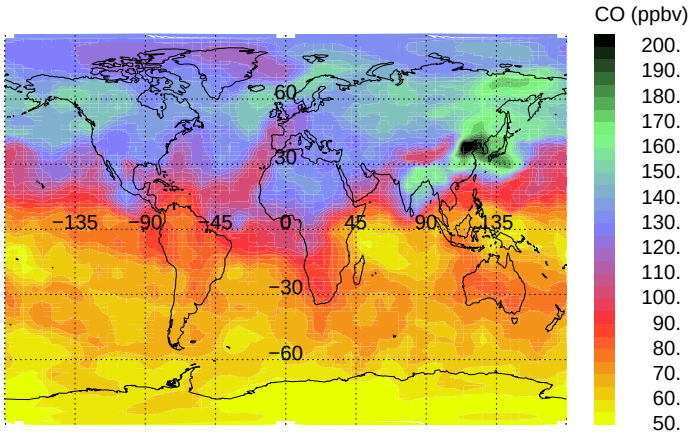
5. Randwertedateien:
   /usr/nfs/data/met_data/clim/config

1 Vergleich mit Mopitt für CO

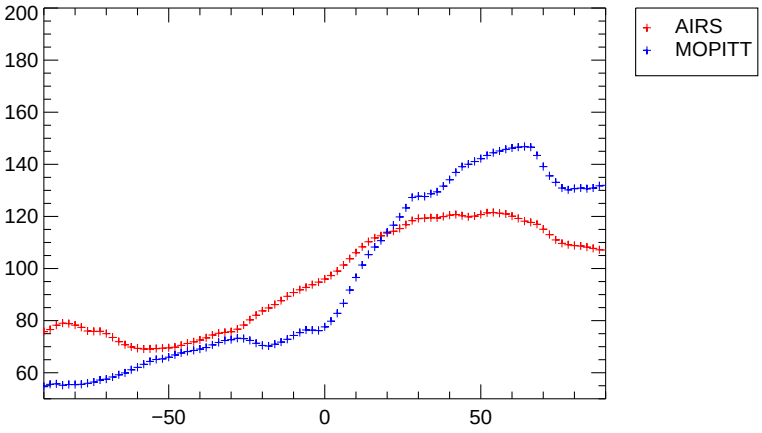
AIRS @ 200K / 10.05.2012



MOPITT @ 200K / 10.05.2012

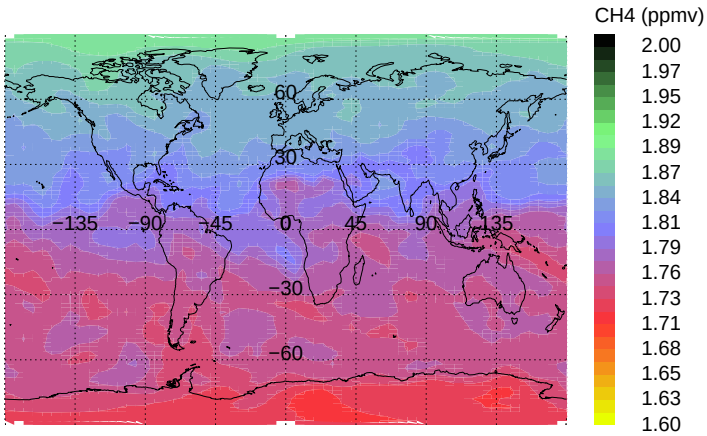


Zonales Mittel @ 200K / 10.05.2012



2 Vergleich mit CMDL für CH4

AIRS @ 200K / 16.05.2012



Zonales Mittel @ 200K / 16.05.2012