Lagranto - Tutorial

Michael Sprenger and Heini Wernli

August 5, 2011

1 Definition of the Problem

In this tutorial we consider a simple way how to find airstreams which transport air from the surface into the upper troposphere and lower stratosphere, i.e. the UTLS region. More specifically, we intend

1) to initialize starting positions over the North Atlantic and Europe (80 W to 20 E, 40 N to 80 N) for 00 UTC, 20 October 1989. The starting positions are horizontally equidistant with 80 km horizontal spacing and are set 100 hPa above ground level.

2) to calculate trajectories several hours forward in time.

3) to select trajectories which ascend to levels above 400 hPa within 48 hours and are found at starting time below 700 hPa.

4) to trace several meteorological fields along the trajectories: potential vorticity (PV), potential temperature (TH), relative (RH) and specific humidity (Q).

4 to select subsamples of trajectories: a) those reaching the stratosphere; b) those travelling at most 2000 km; ...

5) to show the densities of the trajectories on a geographical map.

2 Meteorological Data

The following dates and netCDF files are needed for a Lagranto calculation covering the time period from 07 UTC 14 February 2011 to 11 UTC 14 February 2011.

```
> datelist stdout -create 20110214_07 20110214_11 interval 1
 20110214_07
 20110214_08
 20110214_09
 20110214_10
 20110214_11
```

There are two different files involved, the P and the S files. Lagranto expects them to be in the running directory:

```
> ls -1
 P20110214_07 P20110214_08 P20110214_09 P20110214_10 P20110214_11
 S20110214_07 S20110214_08 S20110214_09 S20110214_10 S20110214_11
```

The meteorological fields on the primary P files are at least: zonal wind (U, in m/s); meridional wind (V, in m/s); vertical wind (OMEGA, in Pa/s); surface pressure (PS, in hPa). Additional fields might be available on the P files, e.g. temperature (T), specific humidity (Q),... Secondary fields can be saved in the S files, which must have the same grid structure as the P files. In the example the following fields are saved on the S files: potential temperature (TH, in K); Ertel potential vorticity (PV, in pvu); relative humidity (in %). Furthermore, the surface pressure (PS) is also saved on the S files; it must be exactly identical to the one in the P files.

If surface pressure is not saved on the S files, it must be copied from the corresponding P files. This can most readily be done using the NCO tools (http://nco.sourceforge.net/):

```
> ncks -A -v PS P20110214_07 S$20110214_07
```

If the P and S files are stored at another place, they might be linked with a simple Shell script (in csh), usimg the command *datelist*:

```
> foreach date ( ‘datelist stdout -create 20110214_07 20110214_11 -interval 1‘ )
>   ln -s {SOURCE DIR}/P${date} {DEST DIR}/P${date}
>   ln -s {SOURCE DIR}/S${date} {DEST DIR}/S${date}
> end
```

Note that the command *datelist* offers several options how to work with date list - creating, stepping through, comparing.

3 Starting Positions

In a first step the starting positions must be specified. To this aim a file *regionf* must be created with the definition of the region. Note that the UM uses a rotated coordinated system, whereas the starting region must be specified in true longitude and latitude in *regionf*. The position of the rotated north pole can easily be extracted with

```
> rot2geo -file 20110214_07
112 157.5
```

where 112.0 and 157.5 are longitude and latitude of the rotated north pole. The file *regionf* is in the same directory as the meteorological data (section 1):

```
> more regionf
"1 -69.0 -67.0 -68.0 -66.0"
```

The first number specifies a region ID (here 1) and the other values are: west boundary (69 W), east boundary (67 W), southern boundary (68 S) and northern boundary (66 S). It is easily checked whether the lat/lon points are within the rotated domain. To this aim, use for instance

```
> geo2rot -69.0 -68.0 -file P20110214_07
  -0.3746045      -0.5030226
```

where now the output give the rotated longitude and latitude. These coordinates can be compared to the ones (domxmin,domxmax,domymin,domymax) on the netCDF files:

```
> ncdump -h P20110214_09
...
/ global attributes:
                :domxmin = 359.3602f ;
                :domxmax = 362.9647f ;
                :domymin = -1.5665f ;
                :domymax = 2.038f ;
                :domzmin = 1000.f ;
                :domzmax = 50.f ;
...
```

The starting positions are then created with

```
> create_startf 20110214_07 startf.2 'region.eqd(1,10) ...
                             ... @ level(100) @ hPa,agl' -changet
```

The starting positions are written to *startf.2*, i.e. in format 2, and cover the region 1 specified in *regionf*. The horizontal start points are equidistantly distributed with 10 km spacing, and they are all at 100 hPa above ground. All points refer to the starting date 07 UTC, 14 February 2011. In total, 3750 starting positions are written to *startf.2*. The first few lines of the file look as follows:

```
> head -10 startf.2
Reference date 20110214_0700 / Time range       0 min

  time      lon     lat     p     level
---------------------------------------


   0.00   -68.996  -67.928   908   100.000
   0.00   -68.756  -67.928   908   100.000
   0.00   -68.516  -67.928   908   100.000
   0.00   -68.276  -67.928   908   100.000
   0.00   -68.037  -67.928   908   100.000
```

The different columns are: time, longitude, latitude, pressure (in hPa) and level (in hPa,agl). Note that the 'same' starting file could have been created without a region file. In this case, the command would have been:

```
> create_startf 19891020_00 startf.2 'box.eqd(-69,-67,-68,-66,10) ...
                             ... @ level(100) @ hPa,agl' -changet
```

However, note that the starting positions are not exactly the same as with the previous command: it is only guaranteed that the starting points are equidistantly distributed within the region.

4 Trajectory Calculation

In a next step, the trajectories are calculated 72 h forward in time, with starting date 00 UTC, 20 October 1989. The command is:

```
> caltra 19891020_00 19891023_00 startf.2 traj.4 -j
```

The taring positions are taken from*startf.2* and the output is written to *traj.4*. Furthermore, the jumping flag *-j* is set, i.e. if trajectories run into the ground they are lifted a little and allowed to move on.

Note that the output file *traj.4* is not in ASCII format. To look at the file, use the command *trainfo*, for instance:

```
> trainfo traj.4 vars
time lon lat p

> trainfo traj.4 dim
3750           13           4

> trainfo trai.4 startdate
19891020_0000
```

It is also possible to convert the trajectory file into ASCII format with the command *reformat*:

```
> reformat traj.4 traj.1
> more traj.1
Reference date 19891020_0000 / Time range    4320 min

  time      lon     lat     p
-----------------------------

   0.00   -79.61   40.45   862
   6.00   -80.57   43.23   791
  12.00   -82.23   45.89   782
  18.00   -84.94   47.07   744
```

The command *trafinfo* cna also be used to look at the trajectory tables, i.e. without a conversion to ASCII format. To this aim, use

```
> trainfo traj.4 list
```

Whereas the ASCII format is most convenient for visual inspection, it is the least compact format. In particular, if the output of *caltra* should be further processed, e.g. with *trace* or *select*, the binary formats should be used (see documentation for *reformat*).

5 Pre-Selection of Trajectories

Quite often, the position of the air parcels is sufficient to select trajectories. It is then most efficient to pre-select these trajectories and do the tracing of additional fields along the trajectories only on the pre-selected ones. In the example, airstreams should be identified which ascend from below 700 hPa at initial time to levels above 300 hPa. The ascent has to take place within 48 h. This selection can be achieved with the command:

```
> select traj.4 wcb.1 'GT:p:700:FIRST & LT:p(MIN):400:0 to 48'
```

The first criterion selects all trajectories for which the pressure (p) at the initial time (FIRST) is greater than (GT) 700 hPa. This criterison is logically AND-combined with the second criteron: consider all times between 0 and 48 h and take the minimum pressure, i.e. p(MIN), over this time interval; if the minium pressure is less than (LT) 400 hPa, the trajectory is selected. A sample trajectory looks like:

```
> trainfo wcb.1 list
Reference date 19891020_0000 / Time range    4320 min

  time      lon     lat     p
-----------------------------

   0.00   -72.98   40.45   918
   6.00   -76.45   43.14   879
  12.00   -78.53   46.69   808
  18.00   -80.08   48.70   770
  24.00   -84.49   48.71   563
  30.00   -87.89   43.32   377
  36.00   -80.69   37.24   396
  42.00   -73.05   39.00   477
  48.00   -67.62   47.21   488
  54.00   -63.53   54.61   455
  60.00   -53.79   58.53   447
  66.00   -38.79   59.08   452
  72.00   -27.72   55.51   493
```

In total, 99 trajectories are selected. Further Lagrangian selection criteria might be reasonable. For instance, it could be of interest whether the air parcels are far away from their initial position after 48 h:

```
> select traj.4 wcb.1 'GT:p:700:FIRST & LT:p(MIN):400:0 to 48 & GT:DIST0:5000:48'
```

The last criterion test whether the spherical distance from the initial position exceeds 5000 km at 48 h (only met by 4 trajectories). Note that the field *DIST0* is not available on the trajectory file *traj.4*, but is implicitely calculated.

Similarly, it can be tested whether a trajectory passes through a target region (e.g. 20E-30E,50N-60N). Such a region might be defined in the region file *regionf*:

```
> more regionf
# Starting positions
"1 -80  20 40 80"
# Target region
"2 20 30 50 60"
```

The call to *select* now looks as follows:

```
> select traj.4 wcb.1 'GT:p:700:FIRST & LT:p(MIN):400:0 to 48 & ...
                       ... TRUE:INREGION:2:42 to 54(ANY)'
```

The last criterion is interpreted in the following way: consider the times from 42 h to 48 h (42 to 54) and check whether a trajectory is at any of these times (ANY) in the traget region 2, as specified in *regionf*. A sample trajectory is given below:

```
> more wcb.1
Reference date 19891020_0000 / Time range    4320 min

  time      lon     lat     p
-----------------------------

   0.00   -44.56   40.45   914
   6.00   -41.22   38.95   885
  12.00   -37.50   37.59   849
  18.00   -33.54   36.76   823
  24.00   -29.45   36.45   770
  30.00   -24.56   37.44   622
  36.00   -18.89   41.17   429
  42.00   -10.48   49.48   349
  48.00     8.75   57.02   355
  54.00    28.74   56.97   354
  60.00    37.14   53.71   320
  66.00    38.99   49.75   334
  72.00    37.57   45.94   349
```

6 Tracing Meteorological Fields

Meteorological fields can be traced along the trajectories with the command *trace*. Most often, a list of fields to trace will be listed in a file *tracevars*:

```
> more tracevars
PS    1.  0 P
Q  1000.  0 P
TH    1.  0 S
RH    1.  1 *
```

The following fields are to be traced: surface pressure (PS), specific humidity (Q), potential temperature (TH) and relative humidity (RH). PS and Q are available on the P files and need not to be calculated; Q will be scaled by a factor 1000 to convert from Kg/kg to g/kg. TH is found on the S file and need not to be calculated. Finally, RH is found neither on the P nor on the S file and must becomputed - the flag 1 in the third column.

With the *tracevars* file ready, the tracing is started with:

```
> trace wcb.1 wcb.1
```

Note that the input and output file are allowed to have the same name. The following table shows the first few lines of the new trajectory file:

```
> more wcb.1
Reference date 19891020_0000 / Time range    4320 min

  time      lon     lat     p        PS         Q        TH        RH
---------------------------------------------------------------------

   0.00   -44.56   40.45   914  1014.093     9.664   294.921    86.667
   6.00   -41.22   38.95   885  1012.965     8.380   296.549    77.992
  12.00   -37.50   37.59   849  1014.056     8.565   299.122    81.321
  18.00   -33.54   36.76   823  1012.074     8.720   300.798    85.407
  24.00   -29.45   36.45   770  1012.254     7.666   303.487    85.262
```

If it is later found that additional fields should be traced, this can be done with a new *tracevars* file or for a single field with (the second number being the scaling factor):

```
> trace wcb.1 wcb.1 -f PV 1.
```

Note that several fields are allowed for online computation, i.e. with the computation flag set in *tracevars*. This is convenient for interactive mode and for few trajectories. However, if tracing is needed for many trajectories, a pre-calculation and saving on the S files is much more efficient! A list of fields for online computation is found in the reference guide for *trace*.

It is also possible to trace the surrounding of a trajectory position, i.e. to get for instance not the temperature at the air parcel's position, but at 50 hPa above or below it. This is done with:

```
> trace wcb.1 wcb.1 -f T:-50HPA 1.
> trace wcb.1 wcb.1 -f T:+50HPA 1.
```

Finally, if it is decided that a field is no longer needed in the trajectory file, or if it has to be corrected, it is possible to remove columns from the trajectory file. This can be achieved with *extract* - for instance if only PS, TH and RH should be kept:

```
> extract wcb.1 wcb.1 -var PS TH RH
```

Note that *extract* can also be used to extract different times or starting positions (see the reference documentation).

7 Final Selection of Trajectories

In this section the selection of trajectories should be refined, i.e. it is not only based on positional information but also on further meteorological parameters. We look at several questions:

a) Is there a trajectory which reaches saturation ($RH > 99\%$)? The trajectories should be saved in a new trajectory file.

```
> select wcb.1 sat.1 'GT:RH:99:0 to 72(ANY)'
> more sat.1
Reference date 19891020_0000 / Time range    4320 min

  time      lon      lat     p        PS          Q        TH         RH         PV
--------------------------------------------------------------------------------
   0.00   -72.98   40.45   918  1018.161      9.503   292.020   100.722      0.920
   6.00   -76.45   43.14   879   986.319      8.723   294.933    92.837      1.101
  12.00   -78.53   46.69   808   972.550      7.621   297.875    97.737      0.794
  18.00   -80.08   48.70   770   973.957      6.147   297.914    97.912      1.078
  24.00   -84.49   48.71   563   962.279      2.327   307.548    87.923      1.034
  30.00   -87.89   43.32   377   977.415      0.319   314.210    65.759      0.108
  36.00   -80.69   37.24   396   939.606      0.303   312.705    52.845      0.323
  42.00   -73.05   39.00   477  1013.693      0.298   314.614    16.248      0.309
  48.00   -67.62   47.21   488   970.025      0.442   312.975    23.890      0.463
  54.00   -63.53   54.61   455   950.011      0.386   313.047    30.182      0.479
  60.00   -53.79   58.53   447  1007.039      0.392   311.951    36.578      0.487
  66.00   -38.79   59.08   452  1006.532      0.319   311.316    29.286      0.443
  72.00   -27.72   55.51   493  1009.871      0.279   311.428    15.950      0.513
```

b) Get a list of all trajectories which pass through a circle around 20 W/40 N and radius 500 km.

```
> select wcb.1 indlist 'TRUE:INCIRCLE:-20,40,500:ALL(ANY)' -index
> more indlist
          4
          5
          6
         11
         12
         13
         14
         19
         20
         21
         22
         47
```

Hence, the trajectories 4,5,... pass through the circle. The trajectories themselves can be extracted in a second step with

```
> extract wcb.1 pass.1 -index indlist
```

where now the selected trajectories are written to the trajectory file *pass.1*.

c) Select all trajectories which are in the stratosphere after 48 h. The dynamical tropopause is defined as the 2-PVU isosurface?

```
> select wcb.1 out 'GT:PV:2:48 to 72(ALL) & LT:P:500:48 to 72(ALL)'
```

The second criterion guarantees that the air parcel is at a height above 500 hPa; indeed, low-level high-PV regions might mimick a stratosphere, although they are of diabatic origin.

d) Select all trajectories which are within 2000 km distance of their starting position after 72 h.

```
> select wcb.1 sel.1 'LT:DIST0:2000:LAST'
```

Note that the fields *DIST0* needs not to be available on the trajectory file - it is calculated during the selection. *DIST0* refers to the spherical distance (in km) from the strting position. If the path length (in km) is needed, *DIST* can be used instead.

e) How many trajectories ascend more than 550 hPa between 12 h and 54 h? We are only interested in the number of selected trajectories.

```
> select wcb.1 count 'GT:P(DIFF):550:12,54' -count
> more count
3
```

f) We would like to select all trajectories which reach potential vorticity (PV) greather than 2 PVU at levels above 500 hPa. In a first attempt, this might be accomplished with the criterion

```
> select wcb.1 wcb.1 'GT:PV:2:ALL(ANY) & LT:p:500:ALL(ANY)'
```

But note that this is not exactly what we want - the first criterion might be fulfilled at a time 48 h, for instance, whereas the second criterion is fulfilled at another time, say 72 h. Hence they are not both fulfilled at the same time! A way around this problem is possible if a TRIGGER column is used to mark the two events. The original trajectory file looks as follows:

```
> more wcb.1
Reference date 19891020_0000 / Time range    4320 min

  time      lon     lat     p        PS          RH        PV
------------------------------------------------------------

   0.00   -19.56   46.94   905  1005.242     83.514     0.291
   6.00   -14.72   48.17   892   999.182     88.325     0.242
  12.00   -10.58   50.53   862   993.145     97.718     0.293
  18.00    -7.22   53.02   792   972.076     99.216     0.738
  24.00    -3.71   55.89   724   956.135     93.218     1.076
  30.00    -0.19   58.87   629   971.334     70.088     1.076
  36.00     1.46   61.62   452   966.406     66.056     0.558
  42.00     0.01   62.49   328   977.209     65.319     1.754
  48.00    -1.54   63.41   313   983.930     56.822     2.727
  54.00    -3.59   64.77   322   984.627     58.328     1.874
  60.00    -9.91   66.07   323   988.185     57.894     2.052
  66.00   -20.91   66.02   316   976.560     57.989     2.565
  72.00   -28.89   66.19   319  1007.175     54.477     2.693
```

Then we mark the two events with a TRIGGER:

```
> select wcb.1 wcb.1 'GT:PV:2:1(TRIGGER) & LT:p:500:2(TRIGGER)' -trigger
```

The first criterion (PV) gets the trigger 1 (in binary system 01), the second one (pressure) get the trigger 2 (in binary system 10). If both criteria are fulfilled, the trigger column becomes 3, which corresponds in the binary system to 11 - i.e. each flag corresponds to a bit in the trigger value. With the option '-trigger' the trigger column is written to the output trajectory file:

```
> more wcb.1
Reference date 19891020_0000 / Time range    4320 min

  time      lon     lat     p        PS         RH        PV     TRIGGER
-------------------------------------------------------------------------

   0.00   -19.56   46.94   905  1005.242    83.514     0.291       0.000
   6.00   -14.72   48.17   892   999.182    88.325     0.242       0.000
  12.00   -10.58   50.53   862   993.145    97.718     0.293       0.000
  18.00    -7.22   53.02   792   972.076    99.216     0.738       0.000
  24.00    -3.71   55.89   724   956.135    93.218     1.076       0.000
  30.00    -0.19   58.87   629   971.334    70.088     1.076       0.000
  36.00     1.46   61.62   452   966.406    66.056     0.558       2.000
  42.00     0.01   62.49   328   977.200    65.319     1.754       2.000
  48.00    -1.54   63.41   313   983.930    56.822     2.727       3.000
  54.00    -3.59   64.77   322   984.627    58.328     1.874       2.000
  60.00    -9.91   66.07   323   988.185    57.894     2.052       3.000
  66.00   -20.91   66.02   316   976.560    57.989     2.565       3.000
  72.00   -28.89   66.19   319  1007.175    54.477     2.693       3.000
```

Now the selection can be achieved by refering to the TRIGGER column:

```
> select wcb.1 wcb.1 'ALL:TRIGGER:1,2:ALL(ANY)'
```

This selection means that the trigger values 1 and 2 must be set - the operator ALL (first term) guaranteeing that all selected triggers are set. The time specification ALL(ANY) is as before, i.e. the check is performed for all times and he criterion must be fulfilled at any of these times.

g) Select all trajectories which pass at time 60 h over Switzerland! The coordinates of the Swiss boundary are listed in a file *borders.dat*:

```
> more borders.dat
8.55 47.45
7.942863 46.002075
7.949024 46.001195
7.956945 46.000022
7.984226 46.000022
7.989800 46.001489
8.000068 46.007356
8.011508 46.018503
...
```

The first line is a point (longitude, latitude) within Switzerland (Zurich), the other lines define the boundary of Switzerland (as 1373 points). With this polygon file, the selection command becomes

```
> select wcb.1 out.1 'TRUE:INPOLYGON:borders.dat:60'
```

Note that in a criterion only one polygon can be specified.

g) New criteria can easily be implemented into the Fortran code; to this aim the file *special.f* in directory *select* must be edited. The following example shows the implementation of an identification for warm conveyor belts (WCB). The calling sequence for the criterion is *SPECIAL:WCB:ascent,first,last*, the air stream must ascend at least *ascent* hPa between time *first* and time *last*. The corresponding Fortran looks as follows:

```
> more select/special.f

      SUBROUTINE special (flag,cmd,tra,ntim,ncol,
     >                    vars,times,param,nparam)

c     ***************************************************************************
c     *                                                                         *
c     * OUTPUT:  flag           -> 1 if trajectory is selected, 0 if not        *
c     *                                                                         *
c     * INPUT:   cmd            <- command string (e.g. WCB)                    *
c     *          tra(ntim,ncol) <- single trajectory: indices time,column       *
c     *          ntim           <- number of times                              *
c     *          ncol           <- number of columns (including time,lon,lat,p) *
c     *          vars(ncol)     <- names of columns                             *
c     *          times(ntim)    <- List of times
c     *          param(nparam)  <- parameter values                             *
c     *          nparam         <- number of parameters                         *
c     *                                                                         *
c     ***************************************************************************

      implicit none

c     ---------------------------------------------------------------------------
c     Declaration of subroutine parameters
c     ---------------------------------------------------------------------------

      integer       flag           ! Boolean flag whether trajectory is selected
      character*80  cmd            ! Command string
      integer       ntim,ncol      ! Dimension of single trajectory
      real          tra(ntim,ncol) ! Single trajectory
      character*80  vars(ncol)     ! Name of columns
      real          times(ntim)    ! List of times
      integer       nparam         ! # parameters
      real          param(nparam)  ! List of parameters

c     ---------------------------------------------------------------------------
c     Declaration of local variables
c     ---------------------------------------------------------------------------

      integer       i
      integer       ip,i0,i1

c     --------------------------------------------------------------------------  %)
c     SPECIAL:WCB:ascent,first,last                                               %)
c         : Detect Warm Conveyor Belts (WCB); the air stream must ascend at least %)
c         : <ascent=param(1)> hPa between the two times <first=param(2)> and      %)
c         : <last=param(3)>. Note, the lowest pressure is allowed to occur at any %)
c         : time between <first> and <last>.                                      %)
c     --------------------------------------------------------------------------- %)

      if ( cmd.eq.'WCB' ) then

c        Reset the flag for selection
         flag = 0

c        Pressure is in the 4th column
         ip = 4
```

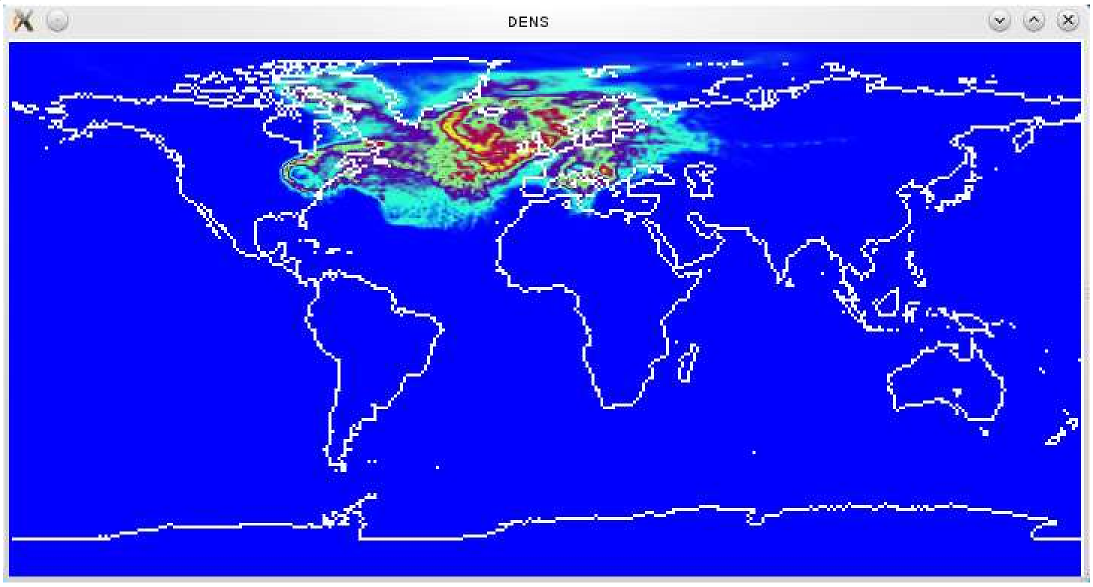
```
c         Get indices for times <first> and <last>
          i0 = 0
          i1 = 0
          do i=1,ntim
             if ( param(2).eq.times(i) ) i0 = i
             if ( param(3).eq.times(i) ) i1 = i
          enddo
          if ( (i0.eq.0).or.(i1.eq.0) ) then
             print*,' ERROR: invalid times in SPECIAL:WCB... Stop'
             stop
          endif

c         Check for ascent
          do i=i0+1,i1
             if ( ( tra(1,ip)-tra(i,ip) ) .gt. param(1) ) flag = 1
          enddo

       endif

c      ----------------------------------------------------------------------------

       end
```

8 Trajectory Densities

Single trajectories can be visualised e.g. with Matlab or wit NCL (see template scripts in the Lagranto folder). If many trajectories should be visualised instead, it is much more convenient to show trajectory densities. The easiiest way to get trajectory densities is:

```
> density wcb.1 densisty
> ncview density
```

This will project the trajectories in the trajectory file *wcb.1* onto a global longitude/latitude grid with 1 degree horizontal resolution. A filter radius of 100 km will be used



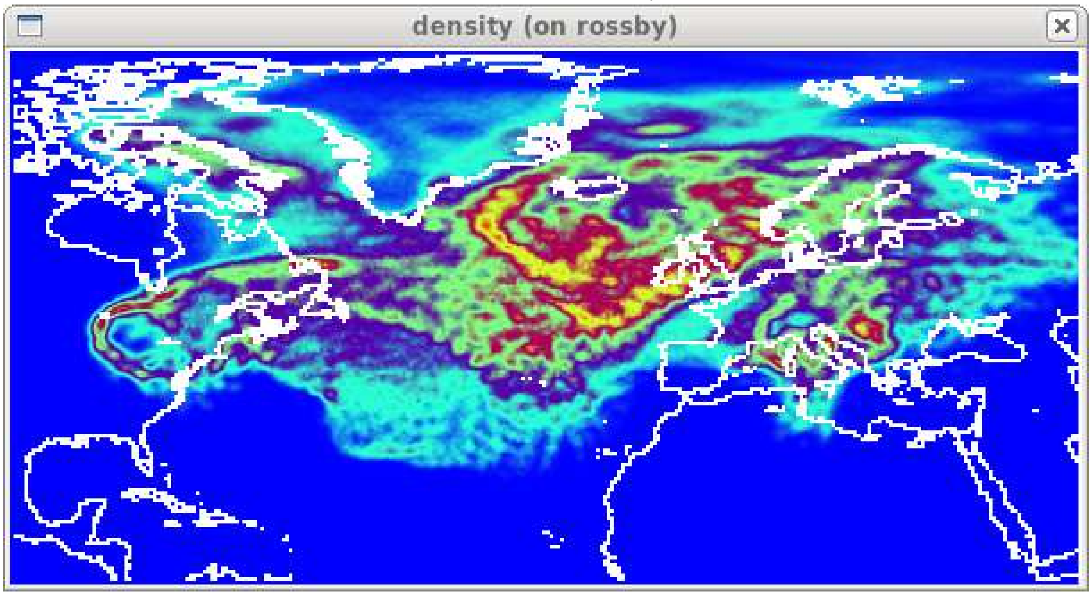
The CF-netCDF file contains several fields: a) the number of trajectory points associated to each grid point (COUNT); b) the residence time of the trajectories (in hours) associated to each grid point - the residence time being the time a trajectory stays at a certain grid cell (RESIDENCE); c) the area (im

km^2) associated with each grid cell. The area allows to change the unit of the gridded trajectory from counts per grid point to counts per km^2.

Often the trajectories do not spread over the whole globe; then the subdomain can be specified with

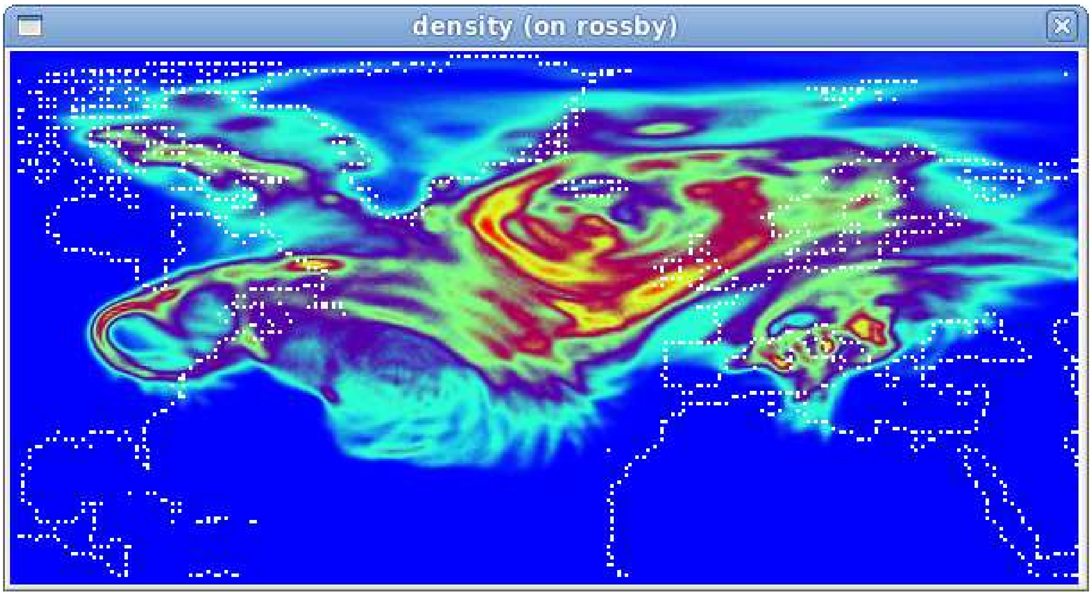
```
> density traj.1 density -latlon 300 150 -100 10 0.5 0.5 -create
```

where the new grid has 300x150 grid points in zonal and meridional direction with south-eastern corner at 100 W/10 S and resolution of 0.5 degree in zonal and 0.5 degree in meridional direction. The flag *create* forces the netCDF file to be created anew, even if it already exists.



It is also possible to re-parameterise the trajectories before they are gridded. For instance, the following command interpolates the positions to a 1-h time interval and then performs the grissing. This results in a smoother density plot:

```
> density traj.1 density -latlon 300 150 -100 10 0.5 0.5 -create -interp 1 h
```



In addition to a gridding of the complete trajectories, the single trajectory times can be gridded.

```
> density traj.1 density -create -time  0.00 -create
> density traj.1 density -create -time  6.00
> density traj.1 density -create -time 12.00
> density traj.1 density -create -time 18.00
> density traj.1 density -create -time 24.00
```

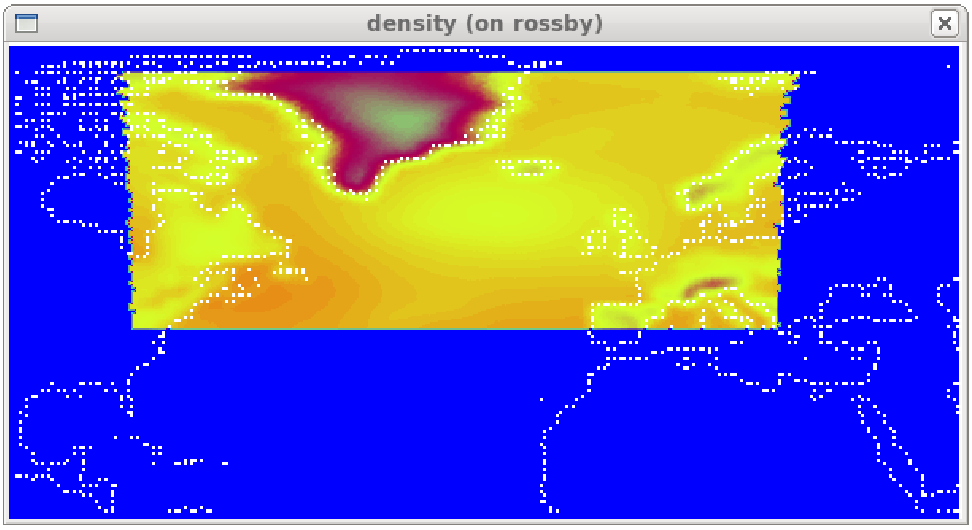
In the previous figures, the density of the trajectories was determined - i.e. the number of trajectories associated with a grid point or the residence associated with a grid cell was determined. In addition to this most basic information, it is also possible to perform a gridding of any trajectory field. For instance, the trajectory file contains

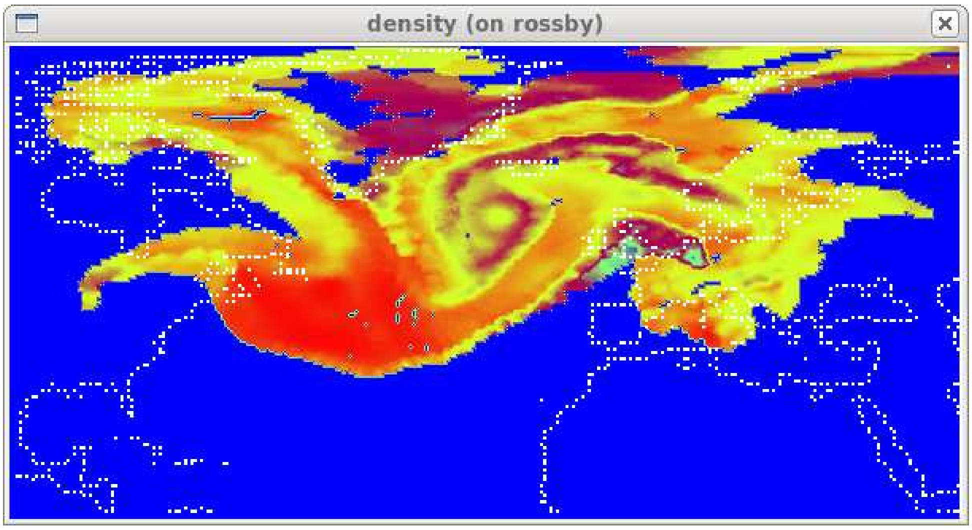
```
Reference date 19891020_0000 / Time range    4320 min

  time      lon      lat     p
-----------------------------


   0.00   -79.61   40.45   862
   6.00   -80.57   43.23   791
  12.00   -82.23   45.89   782
  18.00   -84.94   47.07   744
```

and we would like to know at what height the trajectories typically (in the mean) are at a specific grid point. Then we would grid the pressure "p" instead of the position:

```
> density traj.1 density -create -latlon 300 150 -100 10 0.5 0.5 -field p -time 0.00
> density traj.1 density  -field p -time 24.00
> density traj.1 density  -field p -time 48.00
```

The following figures show the gridded pressure at time 0.00 and 24.00 h; note that the trajectories were initialised 100 hPa above ground.





9 Interface Script

9.1 Start from local directory

In addition to the programs *create_startf*, *caltra*, *trace*, *select*, Lagranto offers a "master" script which combines the call to the individual programs into one single call. For instance,

```
> lagranto local 19891020_00 19891024_18 startf nil -changet
```

will start a Lagranto run starting from 00,UTC 20 Octiober 2010 to 18,UTC 24 Octiober 2010, based upon the starting positions in the file *startf*. No selection of trajectories is applied, as specified with the flag *nil*, and the times on the netCDF P and S files are set relative to the starting date prior to the Lagranto run. Finally, *local* means that all input files are expected in the directory where Lagranto was called.

The output for the above Lagranto call is saved in a newly created directory, which is located in the calling directory:

```
> ls -l ntr_19891020_00_f114_local_startf_nil/
-rw-r--r-- 1 michaesp wheel 5328945 2011-03-21 14:03 lsl_19891020_00
-rw-r--r-- 1 michaesp wheel   68195 2011-03-21 14:03 runscript.logfile
-rwxr--r-- 1 michaesp wheel    1025 2011-03-21 14:02 runscript.sh*
```

The three different files are:

a) **lsl_19891020_00**: the output trajectory file -the file name starts with *lsl* and contains the starting date of the Lagranto run:

```
> more lsl_19891020_00
Reference date 19891020_0000 / Time range    6840 min

  time      lon     lat     p        PS         Q        TH         RH
---------------------------------------------------------------------

   0.00   -79.61   40.45   862   961.659     6.434   290.838    97.722
   6.00   -80.57   43.23   791   980.984     5.334   293.824    98.773
   ...
```

Note that additional fields have been traced along the trajectories, as specified in the tracing file *tracevars*:

```
> more tracevars
PS    1.  0 P
Q  1000.  0 P
TH    1.  0 S
RH    1.  1 *
```

b) **runscript.logfile**: a log file with all status and error information of the Lagranto run. If the flag *-log* is set in a Lagranto call, the log will be written to screen.

c) **runscript.sh**: the calling script for the programs *create_startf*, *caltra*, *trace* and *select*. The basic idea of *lagranto* is to create the output directory, to prepare all netCDF and other files in this output directory and to create a Shell script with name *runscript.sh*. If all these preparations were successfull, Lagranto will change into the output directory and launch *runscript.sh*. The *runscript.sh* for the previous Lagranto call looks as follows:

```
#!/bin/csh
#
#----- Calling command
#
# lagranto local 19891020_00 19891024_18 startf nil -changet -log
#
#----- Output file
#
# lsl_19891020_00
#
#------ Abort if no startf is available
#
if ( ! -f startf ) then
  echo " ERROR: no start file available .... Stop"
  exit 1
endif
#
#------ Remove existing trajectory files
#
if ( -f lsl_19891020_00.4 ) then
  \rm -f lsl_19891020_00.4
endif
if ( -f lsl_19891020_00 ) then
  \rm -f lsl_19891020_00
endif
#
#------ Run <caltra>
#
/home/sprenger/lagranto//bin/caltra.sh 19891020_00 19891024_18 startf lsl_19891020_00.4
#
#------ Abort if caltra was not successful
#
if ( ! -f lsl_19891020_00.4 ) then
  echo " ERROR: caltra failed .... Stop"
  exit 1
endif
#
#------ Run <trace>
#
/home/sprenger/lagranto//bin/trace.sh lsl_19891020_00.4 lsl_19891020_00 -v tracevars
```

```
#
#------ Abort if trace was not successful
#
if ( ! -f lsl_19891020_00 ) then
  echo " ERROR: trace failed .... Stop"
  exit 1
endif
```

Note that you are free to change to the output directory and manually launch *runscript.sh*, possibly after having modified it to your needs. This way of working is uspported by the optional flag *-prep* which will only prepare all files and then changes to the output directory:

```
> lagranto local 19891020_00 19891024_18 startf nil -changet -prep
```

At the end of this call you will be asked to change to the output directory, which -after having agreed- will open a new *xterm* window. Note that you can always easily change to a output directory by calling

```
> lagranto -open local
```

If several trajectory runs are available in the local directory, you are asked to select one. Often, you would like to see the outcome of a run without changing to the output directory. This is most easily accomplished with the following call:

```
> lagranto -show local
```

9.2 Start from case directory

In this calling sequence, for instance

```
> lagranto tutorial 19891020_00 19891024_18 startf nil -changet
```

the input files are not expected in the local directory, but are specified by means of a case identifier. For instance, a case has the identifier *tutorial*. Then Lagranto will expect the input netCDF P and S files to be located in

```
> ls -l ${HOME}/cdf/tutorial
/home/sprenger/cdf/tutorial/P19891020_00
/home/sprenger/cdf/tutorial/P19891020_06
/home/sprenger/cdf/tutorial/P19891020_12
/home/sprenger/cdf/tutorial/P19891020_18
/home/sprenger/cdf/tutorial/P19891021_00
/home/sprenger/cdf/tutorial/P19891021_06
/home/sprenger/cdf/tutorial/P19891021_12
/hom
```

and all the other input files (starting positions, tracing file, region file, polygon specification) are expected in

```
> ls -l ${HOME}/tra/tutorial
startf
tracevars
```

The output of the trajectory calculation will be written to the following output directory, where now the case identifier *tutorial* is part of the directory name:

```
> cd /home/michaesp/tra/tutorial/ntr_19891020_00_f114_tutorial_startf_nil
> ls -1
-rw-r--r-- 1 michaesp wheel 5328945 2011-03-21 14:03 lsl_19891020_00
-rw-r--r-- 1 michaesp wheel   68195 2011-03-21 14:03 runscript.logfile
-rwxr--r-- 1 michaesp wheel    1025 2011-03-21 14:02 runscript.sh*
```

All other aspects are identical to the ones described in the previous section.

10 Installation

In this section you will find some hints how to install Lagranto on a Linux platform. Everthing is handled with the installation script *install.sh* which comes with the Lagranto distribution:

```
> install.csh
install.sh [lib|core|goodies|links|all]
```

The installation should proceed in several distinct steps:

a) Find the place of the netCDF (http://www.unidata.ucar.edu/software/netcdf/) installation on your system - note that the netCDF comes as a pre-compiled package for many Linux distributions and most often can be installed with the Linux software management. Define an environmental variable *NETCDF* which directs to your installation, e.g.

   ```
   > setenv NETCDF /usr/local/netcdf/
   ```

b) Set the environmental variable *LAGRANTO* to the place where you have stored the Lagranto source code and include Lagranto in your search path. In *csh* this might look as follows:

   ```
   > setenv LAGRANTO   /home/michaesp/lagranto/
   > set isLAGRANTO=`echo $PATH | grep $LAGRANTO | wc -l`
   > if ( $isLAGRANTO == 0 ) then
   >   setenv PATH $LAGRANTO/bin:${PATH}
   > endif
   ```

   You might include these statements also in your *.cshrc* file. If successful, you will then be able to open the lagranto help, e.g. with

   ```
   > lagrantohelp
   ```

c) Install the different components of Lagranto and create links - proceed step by step to ensure that each one was successfully completed:

   ```
   > install lib
   > install core
   > install goodies
   > install links
   ```

Lagranto should now be ready to run! As a next step you might want to consider the tutorial, which can be invoked with the command:

```
> lagrantohelp tutorial
```

If you are familiar with the most basic aspects of Lagranto, please refer to the reference guide which enlists all options of Lagranto:

```
> lagrantohelp refernce
```

The contents of the reference guide can also be called from within the Linux shell, e.g. the documentation of *caltra* can be seen in man page format with:

```
> lagrantohelp caltra
```