

# Lagranto - Reference

Michael Sprenger and Heini Wernli

August 19, 2016

**NAME**

**caltra** - calculate air parcel trajectories

**SYNOPSIS**

**caltra** *startdate* *enddate* *startfile* *filename* [ *optional arguments* ]

**DESCRIPTION**

Calculate trajectories for the air parcels starting at the positions specified in *startfile*. The trajectories cover the time period from *startdate* to *enddate* and the trajectories are saved in the output file *filename*. Forward and backward trajectories can be calculated according to the order of the start and end date.

**PARAMETERS**

*startdate* start time of the air parcels in the format YYYYMMDD\_HH(MM) (e.g. 20100101\_00 or 20100101\_0030 for 1 January 2010, 00 UTC and 00:30 UTC). Note that the minutes (MM) are optional.

*enddate* end time of the air parcels (same format as the *startdate* ). If the end time is after the start time, forward trajectories are calculated; otherwise, i.e. for end date before the start date, backward trajectories result.

*startfile* file with the starting positions of the trajectories (possibly created with **create\_startf** ). Different formats for the "startfile" are supported (see **reformat** for details). If no format specifier (appendix .[1234]) is given, a simple (longitude,latitude,pressure) list is expected.

*filename* output trajectory file with trajectories. Different formats are supported (see **reformat** for details).

**OPTIONAL**

*-j* Jumping flag: if a trajectory crosses the lower boundary, it is raised a little and hence is allowed to move on. Otherwise, i.e. no "-j" flag set, the trajectory would stick at the same position. The default is that "-j" is **not** set.

*-i hours* time increments (in hours) for input P and S files. If not explicitly specified, this is determined from the P and S files in the current directory.

*-t min* time step (in minutes) for trajectory calculation. Per default, the time step is 1/12 the time increment of the input files. For instance, 6-h input P and S files result in a time step of 5 min. The time step must be consistent with the output interval (see next optional parameter "-o").

*-o hours* Output interval (in minutes) of the air parcel positions. Per default it is the same as the time increment between the input P and S files (see option "-i"). Note that the output interval must be a multiple of the time step for trajectory calculation (see optional argument "-t").

*-p* Periodicity flag. If set, a periodic domain is assumed in zonal direction. Per default, the flag is **not** set.

*-changet* flag whether the times of the P and S files should be changed or not before a calculation; the default is that the times are **not** changed.

*-noclean* flag whether parameter and criterion files should be kept; this is particularly helpful for debugging.

*-notimecheck* take the first time on the netCDF file - do no explicit test that the requested time is available on the file. This is particularly helpful if you have no write permission for the P files.

**EXAMPLES**

[1] **caltra 19891020\_00 19891020\_18 startf OUT.1**

Calculate forward trajectories from 20/10/1989 00 UTC to 20/10/1989 18 UTC. The starting positions are given on the file "startf" as a list of (longitude,latitude,pressure) values. The output

trajectories are written to the file "OUT.1", where the appendix 1 denotes ASCII format..B

**[2] caltra 19891020\_18 19891020\_00 startf OUT.1**

As in example 1, but backward trajectories from 20/10/1989 18 UTC to 20/10/1989 00 UTC.

**[3] caltra 19891020\_00 19891020\_18 startf OUT -j**

As in example [1], but with jumping flag set: if a trajectory crosses the lower boundary (topography), it is raised a little and then is allowed to move on.

**[4] caltra 19891020\_00 19891020\_18 startf OUT -j -o 15 -t 15**

As in example [3], but the output interval is set to 15 min with the optional argument "-o". Note that the output interval (15 min) must be a multiple of the time step, which is here set explicitly to 15 min with "-t".

**[5] caltra 19891020\_0130 19891020\_1730 startf1 OUT -j -o 15 -t 15 -changet**

Start from non-analysis time 01:30 UTC to non-analysis time 17:30 UTC. Furthermore, the times on the primary netCDF files are changed accordingly.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

changechst()

changechst()

**NAME**

**changechst - change constants file name on the netCDF file**

**SYNOPSIS**

**changechst** *filename new\_cst\_file*

**DESCRIPTION**

Change the name of the constants file for "filename" to the new name "new\_cst\_file"

**PARAMETERS**

**filename**            name of the netCDF file

**new\_cst\_file**        name of the new constants file name

**EXAMPLES**

[1] **changechst P20110102\_18 ml\_cst**

change the constants filename for P20110102\_18 to its new name "ml\_cst".

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

changet()

changet()

## NAME

**changet** - change time on a netCDF file with filename P{YYYYMMDD\_HH} or S{YYYYM-MDD\_HH}

## SYNOPSIS

**changet** *startdate file*

## DESCRIPTION

Change the time value of a netCDF file with filename P{YYYYMMDD\_HH} or S{YYYYMMDD\_HH}; the time is set in hours relative to the startdate (in format YYYYMMDD\_HH).

## PARAMETERS

**startdate** reference date in format YYYYMMDD\_HH.

**file** name of the netCDF file with filename P{YYYYMMDD\_HH} or S{YYYYM-MDD\_HH}.

## EXAMPLES

[1] **changet 20110102\_00 P20110102\_18**

sets the time of the netCDF file "P20110102\_18" to 18 h.

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)

**NAME**

**create\_startf** - create starting files for Lagranto

**SYNOPSIS**

**create\_startf** *date filename specifier* [ *optional arguments* ]

**DESCRIPTION**

Create starting files for a Lagranto calculation. The starting positions are based on the P and S files for *date* and are as specified in a *specifier*. The starting coordinates (longitude, latitude, pressure [in hPa]) are written to the file *filename*.

**PARAMETERS**

*date*                    date of input P and S file (e.g. 20100101\_00). If the date is between two P and S files, linear interpolation is used between the two times.

*filename*                output file with starting points (e.g. startf). Different formats are supported (see **reformat** for details)

*specifier*                detailed description of starting positions. The specifier has the following format: *<horizontal>* @ *<vertical>* @ *<unit>* @ *<selection>*. The components of the specifier are described in greater detail in the following sections.

**HORIZONTAL**

- **file[filename]**    read lat/lon from file "filename"; each line contains one lat/lon pair.
- **line[lon1,lon2,lat1,lat2,n]**  
n points from (lon1,lat1) to (lon2,lat2); the points are linearly interpolated in lat/lon space.
- **box.eqd[lon1,lon2,lat1,lat2,ds]**  
lat/lon box bounded with south-western point (lon1,lat1) and north-eastern point (lon2,lat2); the equidistant points within the box have a horizontal distance ds (in [km]).
- **box.grid[lon1,lon2,lat1,lat2,]**  
lat/lon box with south-western point (lon1,lat1) and north-eastern point (lon2,lat2) grid points; all grid points within this box are taken as starting points.
- **point[lon,lat]**    single lon/lat point.
- **shift[lon,lat,dlon,dlat]**  
lon/lat points and dlon/dlat shifted ones, i.e. in total five points: central one and four shifted ones: (lon,lat), (lon+dlon,lat), (lon-dlon,lat), (lon,lat+dlat), (lon,lat-dlat).
- **polygon.eqd[filename,ds]**  
equidistant within arbitrary polygon (ds in [km]). The file with the polygon points has the following format: 1st line a lon/lat point within the polygon; further lines lon/lat points of the vertices (max 500) of the polygon.
- **polygon.grid[filename]**  
grid points within arbitrary polygon. The file with the polygon points has the following format: 1st line a lon/lat point within the polygon; further lines lon/lat points of the vertices (max 500) of the polygon.
- **circle.eqd[lonc,latc,radius,ds]**  
circle with centre at (lonc,latc) and radius "radius" (in km); the equidistant points within the circle have a horizontal distance ds (in [km]).
- **circle.grid[lonc,latc,radius]**  
circle with centre at (lonc,latc) and radius "radius" (in km); all grid points within the circle are selected.
- **region.eqd[id,ds]**  
Read region specification from region file ("default regionf", to be changed with option "-regionf") and fill it equidistantly with starting points (ds in km). The region

identification is "id", see below in section REGION FILE.

- **region.grid[id]** Read region specification from region file ("default regionf", to be changed with option "-regionf") and fill it with starting points on the input grid. The region identification is "id", see below in section REGION FILE.

## VERTICAL

- **file[filename]** read levels from file "filename"; each line in the file contains one level.
- **level[lev]** a single level
- **list[lev1,lev2,lev3,...]**  
a list of levels; if many levels are needed they are better passed to "create\_startf" with the option "file[filename]".
- **profile[lev1,lev2,n]**  
n equidistant levels between lev1 and lev2.

## UNIT

- **hPa** pressure (in hPa).
- **hPa,agl** pressure (in hPa) above ground level.
- **K** potential temperature (in K).
- **PVU** potential vorticity (in PVU). Note that potential vorticity (PV) might not be unique as a vertical coordinate; if several levels have a given PV value, the highest one is chosen.

## SELECTION

- **criterion** Selection criteria based on meteorological fields applied to the starting position; The criteria follow the syntax of the program **select**.
- **nil** If no selection criteria should be invoked, the argument "nil" should be given.

## REGION FILE

Several starting regions can be defined for every case in a region file (default filename is "regionf"; to be changed with optional parameter "-regionf filename"). There are two possible formats for specifying a region (they require either a line with 5 or 9 entries):

### **regnum lonw lone lats latn**

a regular latitude-longitude square: regnum=integer region number; lonw=westernmost longitude of starting region; lone=easternmost longitude; lats=southernmost latitude; latnNorthernmost latitude.

### **regnum lon1 lat1 lon2 lat2 lon3 lat3 lon4 lat4**

an irregular latitude-longitude square: regnum=integer region number; lon{x},lat{x} = longitude and latitude of the x-th corner. Note that the 4 corners must be arranged counterclockwise. For a triangle the 4th corner can be specified identically to the 3rd.

**Note: (1) if a line starts with '#' it is regarded as comment and not further considered; (2) each line in the region file must start with ' '!**

**101 -40. -24. 52. 60. :**

region in the central Atlantic from 40 W to 24 W and 52 N to 60 N; the region identifier is 101.

**250 -30. 43. -24. 36. -18. 50.2 -35.2 50.2 :**

irregular square in the central Atlantic; the region identifier is 250.

## OPTIONAL

- t tracefile* tracing file with variables for selection criteria (see **trace** for format of the file). If no file is specified, the default "tracevars" is used. Further, if no selection criterion is invoked, no tracing file is necessary.
- changet* flag whether the times of the P and S files should be changed or not before a calculation; the default is that the times are not changed.

- noclean* flag whether parameter and criterion files should be kept; this is particularly helpful for debugging.
- regionf filename* change the region file from its default value "regionf" to a new file name: the syntax is "-regionf filename".
- notimecheck* take the first time on the netCDF file - do no explicit test that the requested time is available on the file. This is particularly helpful if you have no write permission for the P files.

## EXAMPLES

**[1] create\_startf 19891020\_00 startf 'point(-10,50) @ list(450,500,550) @ hPa'**

Starting points are (longitude, latitude, pressure in hPa): (-10,50,450); (-10,50,500); (-10,50,550). No selection criterion is applied; the positions are written to file "startf".

**[2] create\_startf 19891020\_00 startf 'line(-10,-5,40,50,10) @ level(450) @ hPa,agl'**

10 points are equidistantly specified between lon/lat point (-10,40) and (-5,50); all trajectories start at 450 hPa above ground level - the surface pressure is taken from the primary file P19891020\_00. The positions are saved in "startf".

**[3] create\_startf 19891020\_00 startf 'box.grid(-10,-5,40,50) @ list(300,320) @ K'**

All grid points in the box with the south-eastern lon/lat point (-10,40) and the north-eastern one (-5,50) are taken - the horizontal grid spacing is specified in the primary file P19891020\_00. In the vertical, two isentropic levels are chosen: 300 K and 320 K. The potential temperature for the calculation is taken from the secondary file S19891020\_00.

**[4] create\_startf 19891020\_00 startf 'shift(-10,40,1,1) @ profile(1000,200,100) @ hPa'**

A profile of 100 equidistant levels between 1000 hPa and 200 hPa; in the horizontal the central lon/lat point (-10,40) is taken and four horizontally displaced ones, the displacement being 1 degree in zonal and meridional direction.

**[5] create\_startf 19891020\_00 startf.1 'shift(-10,40,1,1) @ profile(1000,200,100) @ hPa'**

As in the previous example [4], but the starting positions are saved as a trajectory file instead of a (lon,lat,p)-list.

**[6] create\_startf 19891020\_00 startf.1 criterion**

As in the previous example [5], but the criterion is saved on a file with filename "criterion".

**[7] create\_startf 19891020\_00 startf 'polygon.grid(polygon) @ level(500) @ hPa'**

A polygon is specified in the file "polygon"; the different lines in the file are: -5. 45. / -10. 40. / 10. 40. / 10 50. / -10. 45. The first lon/lat point lies within the polygon, all other lon/lat points are the vertices of the polygon. All grid points within the polygon are taken as starting point, at level 500 hPa.

**[8] create\_startf 19891020\_00 startf 'polygon.eqd(polygon,50) @ level(500) @ hPa'**

As in the previous example [7], except that the starting points are distributed equidistantly within the polygon. The horizontal distance between the starting points is 50 km in zonal and meridional direction.

**[9] create\_startf 19891020\_00 startf 'shift(-10,40,1,1) @ profile(1000,200,100) @ hPa @ GT:TH:310'**

As in example [4], but a selection criterion is additionally applied: only starting positions with potential temperature (TH) greater than (GT) 310 K are kept. Potential temperature must be available on the secondary file S19891020\_00 and the file "tracevars" must have a line with "TH 1. 0 S". Further examples for selection criteria can be seen in **select**.

**[10] create\_startf 19891020\_00 START.1 'region.eqd(3,10) @ level(500) @ hPa'**

get equidistant starting points (10 km distance) in the region with identifier 3, as listed in the region file "regionf" (the default).

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)



create\_startf()

create\_startf()

**NAME****datelist - handling of datelists****SYNOPSIS****datelist** *filename mode [ parameters ]***DESCRIPTION**

A date list is a file of dates in the format {YYYYMMDD\_HH}, e.g. 19900101\_00 for 00 UTC, 1 January 1990. This command offers several ways how to create date lists and to work with them.

**PARAMETERS**

*filename* name of the date list file. If the name **'stdout'** or **'screen'** is given, the output will be directed to standardoutput: no file will be created.

*mode* one of several modes (see below).

**CREATING DATE LISTS****-create startdate enddate**

create a datelist from startdate (in format {YYYYMMDD\_HHMM}) to enddate; the time interval is per default 6 h (see option -interval). If the start and end date do not match with the analysis times, the date list will contain the enclosing analysis times: for instance, for **-create 20100201\_04 20100201\_19** the date list will contain the following dates: 20100201\_00, 20100201\_06, 20100201\_12, 20100201\_18, 20100202\_00.

**-indir dirname**

search for dates (in format {YYYYMMDD\_HH}) in the directory given with {dirname} - the dates are written in ascending order to the datelist file and repeating dates are removed.

**-interval value**

change the interval to {value} hours, instead of the default 6 hours.

**INFO ABOUT DATE LISTS****-ndates**

write the number of dates in the list

**-timerange**

write the time range {last date} - {first date} [in hours].

**-isin date**

check whether the date is in the list (1) or not (0).

**STEPPING THROUGH DATE LISTS****-first**

write the first date of the date list

**-last** write the last date of the date list**-next date**

find the date {date} in the list and write the **next** date to screen; if no next date is in the list, i.e. the end of the list is reached, 'nil' will be returned.

**-prev date**

find the date {date} in the list and write the **previous** date to screen; if no previous date is in the list, i.e. the beginning of the list is reached, 'nil' will be returned.

**COMPARING DATE LISTS****-overlap file1 file1**

determine the overlap of two date lists.

**-onlyin1 file1 file1**

determine the dates which only occur in date list 1, but not in datelist 2

**-onlyin2 file1 file1**

determine the dates which only occur in date list 2, but not in datelist 1

datelist()

datelist()

## **Examples**

**[1] datelist screen -indir ./**

look for dates in the current directory and write them to screen

**[2] datelist dates -create 19890101\_00 20110101\_00 -interval 2**

creates dates from 00 UTC, 1 January 1989 to 00 UTC, 1 January 2011 with an interval of 2 hours.  
The output will be written to the file 'dates'.

## **AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

**NAME**

**density** - gridding of trajectory files: either single trajectories or trajectory densities

**SYNOPSIS**

**density** *infile outfile* [ *optional arguments* ]

**DESCRIPTION**

Gridding of a trajectory file "infile"; the output is written to a netCDF file "outfile). The trajectories can be interpolated to equal-distance intervals or to a higher time resolution; furthermore, an interpolation is provided to give a continuous line in a longitude/latitude grid. Besides densities of trajectories, possibly at different times, gridding of traced meteorological fields is accepted.

**PARAMETERS**

**infile**                   input trajectory file (in one of the accepted formats, see **reformat** for details)  
**outfile**                   output netCDF file with a regular or rotated longitude/latitude grid. The output can be either in IVE or in CF netCDF format (see switches below).

**OPTIONAL PARAMETERS****- field name**

Field name (according to the column names of the trajectory file) which should be gridded. If no field is specified, it is assumed that the trajectory density (counts per grid point) should be gridded.

**- create**

determines the behaviour if the output file already exists: 1) if the file does NOT exist, it will be created; 2) if the file already exists, it will be overwritten, i.e. newly created. If the flag is not set and the file already exists, the output of the new gridding will be appended to the existing file. As a typical scenario: **density TRAJECTORY DENSITY**; **density TRAJECTORY DENSITY -field p**; **density TRAJECORY DENSITY -field PV**;... -> all fields (p, PV,...) will be written to the same output file "DENSITY".

**- index filename**

perform only a gridding of the trajectories in the index list "filename"; an index list is just a list of the indices of the single trajectories - it can easily be created with **select** and also is used in **extract**.

**- boolean filename**

perform only a gridding of the trajectories in the boolean list "filename"; a boolean list has for each trajectory an entry 1 (trajectory selected) or 0 (not selected) - it can easily be created with **select** and also is used in **extract**.

**- interp val unit**

interpolate the trajectories to new positions before the gridding; different modes are accepted: **1) -interp 1 h** interpolates the trajectories to a new time resolution (here 1 h); **2) -interp 20 km** interpolates the trajectories to a new space resolution (here 20 km). Note that nearly stationary trajectories will be reduced in their influence considerably if this mode is applied; **3) -interp 2 deg** interpolates to a new resolution, expressed in deg lon/lat. This mode is particularly helpful if a "line" should be drawn on a lon/lat grid.

**- radius val unit**

set the filtering radius for the smoothing (default is 100 km); this radius determines over which radius each trajectory point is smeared out during the gridding. It should be adapted to the grid resolution: if it is too small, and no grid point falls into the circle with this radius, an error message is written. A good choice might be **-radius <dlat|dlon [in km]>**. Further Examples: **-radius 2 deg** for smoothing over 2 degrees lat/lon; **-radius 20 km** for a 20 km smoothing.

**- latlon [ nlon nlat lonmin latmin dlon dlat | dynamic ]**

specification of the regular output lon/lat grid; the default output grid has parameters: nlon = 360, nlat = 180, lonmin = -180, lonmax = 180, dlon = 1, dlat = 1 (global). If the option **dynamic** is chosen, the grid is automatically adapted to the trajectory file: the longitude and latitude boundaries are determined and the resulting ranges divided in 400 grid pixel in each direction. In a second step the grid distance is set equal in zonal and meridional direction, and the other grid parameters correspondingly

adjusted.

**- rotated clon clat nlonlat dlonlat**

specification of the rotated output lon/lat grid; the centre of the rotated grid is given with the central longitude (clon) and latitude (clat) - this point corresponds to 0/0 in the new coordinate system. The horizontal resolution (dlonlat) of the new grid is equal in rotated longitude and latitude. The final domain extensions are:  $-(nlonlat-1)/2*dlonlat$ ,  $+(nlonlat-1)/2*dlonlat$  and corresponding for rotated latitude.

**-time value**

do only a gridding of the time specified; e.g. -time 18.00 would perform a gridding of the trajectory time 18 h. If the netCDF already exists, and if the "-create" flag is not set, the new time will be added to the already existing file - the grid parameters will automatically be taken from the netCDF file.

## OUTPUT FIELDS

**- COUNT**

number of trajectory points attributed to grid points; the integration of this field over the whole domain is equal to the total number of gridded trajectory points, i.e. equal to the product #trajectories x #times.

**- RESIDENCE**

residence time of trajectory points attributed to grid points; the integration of this field over the whole domain is equal to the total time of all trajectories, i.e. equal to the product #trajectories x #time x time interval.

**- AREA**

area (in square kilometers) attributed to each grid point, i.e.  $dlat \times dlon \times \coslatitude$ ). This field can be used to convert the units of COUNT and RESIDENCE to grid-independent values. For instance, **RESIDENCE/AREA** is the residence time per square kilometer.

**- FILED**

gridded field: e.g gridded pressure, temperature, potential vorticity.

## EXAMPLES

**[1] density TRAJECTORY DENSITY**

bring the TRAJECTORY file into a netCDF file DENSITY with gloabl coverage (longitude/latitude grid). The netCDF file is in CF format and the all trajectory points are gridded, based on a smoothing radius of 100 km.

**[2] density TRAJECTORY DENSITY -latlon dynamic**

as in example [1], but now the lon/lat grid is automatically adapted to the range of the trajectory file.

**[3] density TRAJECTORY DENSITY -interp 1 h**

as in exaxmple [1], but the trajectories are interpolated to a 1-h time interval. This gives smoother tra-  
jectories.

**[4] density TRAJECTORY DENSITY -interp 20 km**

as in exaxmple [1], but the trajectories are interpolated to a 20-km distance interval.

**[5] density TRAJECTORY DENSITY -interp 1 deg**

as in exaxmple [1], but the trajectories are interpolated to a 1 deg distance interval. If the output grid (as specified in option "-latlon" or "-rotated") has the same spacing (1 deg) as given in "-interp 1 deg", a continuous line is drawn.

**[6] density TRAJECTORY DENSITY -radius 100 km**

the trajectory points are spread out over a circle with radius 100 km; this is equivalent to a smoothing of the resulting density field. Note that in a equidistant cylindrical projection, the circles become dis-  
torted towards the pole. If this is not appropriate, the option **-radius 2 deg** can be given, which is independent of geographical latitude.

**[7] density VALID DENS -time 18.00**

only gridding of the trajectory points corresponding to time 18 h; if no time is specified or if **-time all**, then all trajectory times are included. Note that interpolation with "-interp" (see above) is not allowed, if only one time step is selected, i.e. "-interp" works only with "-time all".

**[8] density TRAJECTORY DENSITY -rotated 30 50 401 0.1 -interp 0.2 deg**

all trajectory times are gridded, but this time onto a rotated lon/lat grid; for instance, an interesting feature was found at (lon=30,lat=50). The new coordinate system, a rotated lon/lat grid, is centered at this point and spreads from 20 W to 20 W and from 20 S to 20 N, comprising 401 grid points in rotated longitude and latitude direction. The new grid resolution is 0.1 degrees, which was taken into account in the interpolation to 0.2 deg intervals between trajectory points.

**[9] density TRAJECTORY DENSITY -index filename**

all trajectories in the trajectory file TRAJECTORY are gridded which are listed in the index file "filename".

**[10] density TRAJECTORY DENSITY -field p -time 0.00 -latlon dynamic**

in addition of positional gridding (COUNT, RESIDENCE), do also a gridding of the pressure (p). This gives the pressure at the grid points, averaged over all trajectories passing over the grid point.

**[11] density TRAJECTORY DENSITY -time 0.00; density TRAJECTORY DENSITY -time 6.00**

the trajectory density for "TRAJECTORY" is first written to "OUTPUT" for time 0; then, with the second call, the densities are written for the time 6 h and included into the already existing netCDF file "DENSITY". Similarly, additional fields can be added to a already existing netCDF file, e.g. with the second call "density TRAJECTORY DENSITY -field p -time 0.00".

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

difference()

difference()

## NAME

**difference - calculate the difference of two trajectory files**

## SYNOPSIS

**difference** *infile1 infile2 outfile field [ -single|-max ]*

## DESCRIPTION

Calculate the difference of two trajectory files "infile1" and "infile2" and write it to a new trajectory file "outfile". The difference is calculated of the field "field", which must be available on both input files. If the field "LATLON" is specified, the spherical distance between the two trajectories is taken. Furthermore, with "-single" the difference is written at all times, with "-max" only the time of maximum difference is written.

## PARAMETERS

<i>infile1</i>	first trajectory file
<i>infile2</i>	second trajectory file
<i>outfile</i>	output trajectory file - note that this is not a standard trajectory file which can be further processed! The position of both trajectories are listed and also the field for both trajectories and their difference.
<i>field</i>	Name of field for which the difference should be calculated. It must be available in both trajectory files. IF "LATLON" is specified, the spherical distance between the trajectory positions is calculated.

## OPTIONAL PARAMETERS

<i>-single</i>	the difference is written to "outfile" for all trajectory times
<i>-max</i>	the difference is written to "outfile" only for the trajectory time with maximum difference.

## EXAMPLES

### [1] **difference tra1 tra2 out LATLON -single**

gives the spherical distance (LATLON) between trajectories in "tra1" and "tra2". The difference is written for all trajectory times.

### [2] **difference tra1 tra2 out TH -max**

gives the difference of potential temperature (TH) between trajectory file "ra1" and "tra2" and writes the maximum difference ("-max") to the output file "out".

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)

extract()

extract()

## extract

**extract - extract columns, times, single trajectories and starting positions**

### SYNOPSIS

**extract** *inpra outtra mode*

### DESCRIPTION

Extract columns, times, single trajectories or starting positions from an input trajectory file *inpra* and write output to a new trajectory file *outtra*. The different extraction modes are specified with *mode*. Note: the time, longitude, latitude and pressure need not be extracted because they are an integral part of every trajectory file - they are extracted by default.

### EXTRACTION MODE

**-var** extract columns of a trajectory file; the columns can be listed by name (e.g. -var TH PV RH) or a range of columns can be specified by the "to" operator (e.g. -var TH to PV). The two modes can also be combined: "-var TH to PV RH" extracts all columns between TH and PV, and additionally RH.

**-time**

extract trajectory times; the times can be given as a list of times (e.g. -time 6 12) or as a time range (e.g. -time 6 to 18).

**-tra** extract single trajectories; the index of the trajectories can be specified as a list (e.g. -tra 10 12 14) or as a range of trajectories (e.g. -tra 10 to 20).

**-startf**

extract list (longitude, latitude, pressure) of starting positions of the trajectory file (corresponding to time 0).

**-index**

extract single trajectories - the trajectory indices (from 1 to #trajectories) are given on a file (e.g. -index filename).

**-boolean**

extract single trajectories - the trajectory are specified on a boolean (0/1) file (e.g. -boolean filename).

### EXAMPLES

**[1] extract inpra outtra -time 6 to 36 72**

read input trajectory file "inpra", extract times 6 to 36 and additionally time 72, and write output to trajectory file "outtra".

**[2] extract inpra outtra -index infile**

reads input trajectories from "inpra" and write all trajectories to "outtra" which are listed in the file "infile". In "infile" the indices of selected trajectories are listed line-by-line.

### AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)



**NAME**

future plans for Lagranto

**MAJOR****[1] CF netCDF [until 1 April 2011]**

interface to CF netCDF format

**[2] GRIB**

interface to GRIB format

**[3] box tracing**

Instead of tracing point meteorological fields, trace the mean, maximum, minimum, median, variance,... within a box around the trajectory position

**[4] stochastic trajectories**

allow in caltra the addition of a random noise to the velocities; option "-random 0.01" would allow random variations of U, V and OMEGA in the range of 1 %. As a further step, a physical based "parameterisation" of turbulence could be included (based on Ri, DEF, TKE,...). This would allow to calculate an ensemble of trajectories, e.g. for one receptor point.

**[5] Parallelisation**

parallelisation of caltra, and possibly also of trace and select.

**[6] webLagranto**

web interface to Lagranto - in analogy to the HYSPLIT model

**[7] cosmoLagranto**

new version of COSMO Lagranto

**MINOR****[1] SPECIAL**

remember the settings for a SPECIAL command in the parameter list; this will make the evaluation of SPECIAL commands more efficient.

**[2] offset in K**

allow vertical offsets in K for tracing, e.g. 'PV:+5K' traces PV 5 K above the trajectory position

**[3] SPECIAL:STE**

special criterion for STE, STT and TST in the form SPECIAL:STE:380,2,0,24,48; tropopause 2 PVU/380 K, STE within time 0-24 h, residence before and after the exchange at least 48 h.

**[4] compact data format**

additional formats for trajectory files: 1) compressed format; 2) FLEXPART format.

**[5] Goody: residence**

calculate the residence time in a specified region for all trajectories in a file

**[6] Goody: distance**

calculate the minimum distance from a lon/lat/p point for all trajectories in a file.

**[6] Extension of Lagranto run**

extend forward and backward in time an existing Lagranto run

**[7] Definition of polygons**

Matlab tool for the definition of polygon files.

**[8] datelist**

Goody for handling of datelists: 1) datelist file -create start end; 2) datelist file -len; 3) datelist file -first|last; 4) datelist file -sort; 5) datelist file1 file2 -onlyin1|onlyin2|overlap|difference

**REPORT FOR NEW IDEAS**

Please report new ideas to <michael.sprenger@env.ethz.ch>.

**NAME****geo2rot - rotate from true to rotated longitude and latitude****SYNOPSIS****geo2rot** [ *infile outfile* ] [ *optional parameters* ]**geo2rot** [ *lon lat* ] [ *optional parameters* ]**geo2rot** [ *optional parameters* ]**DESCRIPTION**

Rotate the longitude and latitude coordinates from their true geographical values into their values in a rotated coordinate system. The rotation can be done either for a single <lon/lat> point or for a complete trajectory file <infile>. In the latter case, a new trajectory <outfile> file with rotated longitude and latitude is written.

**PARAMETERS**

*infile*           input trajectory file (with true lon/lat)  
*outfile*          output trajectory file (with rotated lon/lat)  
*lon lat*           true longitude and latitude

**OPTIONAL PARAMETERS**

*-lonlat pollon pollat*  
                   specify the coordinates (pollon, pollat) of the rotated north pole

*-file filename*   get the coordinates of the rotated north pole from the file <filename>. The file must be a netCDF file, either a constants or a P,S file.

**EXAMPLES**

- [1] **geo2rot -lonlat -170 45**  
       sets the rotated north pole to 170 W and 45 N. The new position is written to the file <geo2rot.pole>.
- [2] **geo2rot 45 50**  
       takes coordinate (lon/lat)=(45/50) and rotates it to the rotated coordinate system. The position of the rotated pole is read from the file <geo2rot.pole>.
- [3] **geo2rot 45 50 -file eraint\_cst**  
       rotates (lon/lat) = (45/50) to the new system - the new pole position is read from the file <eraint\_cst>.
- [4] **geo2rot trainp traout**  
       read the trajectory file <trainp> and rotate all coordinates. The trajectory file with rotated coordinates is written to <traout>.
- [5] **geo2rot**       write the pole position to screen.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (July 2011)

gettidiff()

gettidiff()

## NAME

**gettidiff** - get the difference (in hours) between two dates

## SYNOPSIS

**gettidiff** *date1 date2*

## DESCRIPTION

Get the time difference (in hours) between two dates ( in form YYYYMMDD\_HH(MM), i.e. the minutes are optional): *date1* - *date2*. The output is in format HH or HH.MM

## PARAMETERS

**date1**                    first date ( in form YYYYMMDD\_HH(MM) )

**date2**                    second date ( in form YYYYMMDD\_HH(MM) )

## EXAMPLES

[1] **gettidiff 20110102\_18 20110205\_00**

gives the time difference between 20110102\_18 and 20110205\_00; the result is -798 h.

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)

getvars()

getvars()

## **NAME**

**getvars - get a list of a fields on a netCDF file**

## **SYNOPSIS**

**getvars** *filename*

## **DESCRIPTION**

Get a list of all fields on a netCDF file - the names of the fields are listed line-by-line.

## **PARAMETERS**

**filename**            name of the netCDF file (e.g. P20110102\_00).

## **EXAMPLES**

[1] **getvars P20110102\_00**

gives a list of all fields on the netCDF file P20110102\_00; the results is: time, Q, LWC, IWC, T, U, V, OMEGA, PS, SLP

## **AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

**NAME**

**lagranto** - master script for a trajectory calculation, including definition of the starting positions, tracing of meteorological fields and selection of trajectories

**SYNOPSIS**

**lagranto** *caseid*[.label] *startdate* *enddate* *startf* *select* [ *optional flags* ]

**DESCRIPTION**

Calculate trajectories for the time period *startdate* to *enddate* for the starting positions given in *startf*, either as position file or as criteria for starting positions. Furthermore, some selection criteria can be applied to the trajectory file, as given either in a selection file or an explicit selection criterion (given in *select* ). Each trajectory calculation is given a case identifier *caseid* which determines where the input data and output files are located.

**PARAMETERS**

**caseid** identifier for a trajectory calculation; "caseid" determines where the input netCDF files are found and where the output trajectory file is written. The different options are: **1) local** (the input P and S files must be ready in the directory where Lagranto is called; the output directory is also written to the local directory); **2) casename** (the input files are found in "\${HOME}/cdf/casename/" and the output is written to "\${HOME}/tra/casename/"; **3) interim** (the input files are taken from the ERA-Interim archive, the output is written to the local directory); **4) analysis** (as in 3), but for the ECMWF operational analysis); **5) forecast** (as in 3), but for the ECMWF deterministic forecast).

**caseid.label**

the specification of "label" is optional; it allows to attribute to the output directory name this label and hence to distinguish between several Lagranto runs. Note that "label" has no influence where the input files are found and where the output directory is written to. It only is added to the output directory name!

**startdate** starting date for the trajectory calculation in format YYYYMMDD\_HH(MM). This date defines the reference date and time for the trajectory output, i.e. it corresponds to time 0.

**enddate** end date for the trajectory calculation in format YYYYMMDD\_HH(MM); if "enddate" is later than "startdate", a forward trajectory calculation is performed, otherwise a backward calculation. As an example: "20100101\_00 20100105\_00" is forward, and "20100105\_00 20100101\_00" is backward in time.

**startf** definition of the starting positions; they can be either available as a **1) (lon/lat/pressure)-list** in a file; as an **2) explicit criterion** (e.g. "point(50,40) @ list(100,200,300,400) @ hPa") - for details, see documentation of **startf**; or as **3) a single point** in the format "longitude latitude pressure".

**select** definition of selection criterion; it can be passed either in a file or as an explicit selection criterion (e.g. "GT:PV:2:LAST"). For further details, see documentation of command **select**.

**OPTIONAL PARAMETERS****-o filename**

name of the output trajectory file; default filename is "Isl\_{startdate}".

**-j**

jumping flag; if the trajectory runs into the ground, it is lifted a little and allowed to move on. This flag is directly passed to the command **caltra**. See documentation for "caltra" for further details.

**-v tracefile**

name of the tracing file which enlists all fields to be traced along the trajectories; the tracing file is directly passed to **create\_startf** and **trace** (see documentation of these two commands for further details).

**-r regionfile**

name of the region file which enlists all regions; the region file is directly passed to **create\_startf** and **select** (see documentation of these two commands for further details).

- changet** change the times on the netCDF files relative to the starting date; Lagranto expects the netCDF times to be relative to the starting date. The default value of "-changet" is false.
- noclean** do no cleaning after a trajectory run; the default is that the output directory will be cleaned. If cleaning is requested (the default), the following files will be kept in the output directory: **1) the output trajectory file; 2) the log file of the trajectory run; 3) the run script.** All other files are deleted.
- log** write log of Lagranto run on screen instead into a file. No log file will be created with this option.
- prep** create the run directory, prepare all files and build the run script - but do not run it. Hence, everything is ready for the Lagranto run, but it is not launched. It can be launched manually, possibly after some manual modifications to the run script, with the following steps: **1) change to run directory** (for instance with "lagranto -open caseid.label"); **2) start the run script** (with ./runscript.sh, where you have to pass the name of your runscript name).

## INPUT FILES

A successful Lagranto runs needs several input files; the following list shows all mandatory and optional input files

### P and S files [mandatory]

input netCDF files; at least the following meteorological fields must be available on the P files: U=zonal wind [m/s]; V=meridional wind [m/s]; OMEGA=vertical wind [Pa/s]; PS=surface pressure [hPa]. Secondary field can be made available on the S file. Both P and S files have the following format: [P|S]YYYYMMDD\_HH, e.g. P20100101\_00 for 1st January 2010, 00 UTC.

### tracevars [optional]

tracing file where all meteorological fields are listed which should be traced along the trajectories. The tracing file is needed by the program **create\_startf** and particularly **trace** (for further details about the format of the file, consider the documentation for these two commands). If no tracing of meteorological fields is needed, no tracing file must be specified. Furthermore, the name of the file can be changed from its default (tracevars) with the optional parameter "-v filename" (see above).

### regionf [optional]

region specification for definition of string positions (with **create\_startf**) or application of Lagrangian selection criteria (with **select**). If no region is used in either "select" or "create\_startf", no region file must be specified. The name of the region file can be changed from its default (regionf) with the option "-r filename".

### startf [optional]

definition of the starting positions, either as an explicit list of longitude, latitude, pressure; or as a criterion saved on the file. If the specification of the starting positions is done with an explicit specification in **create\_startf** (e.g. "point(50,40) @ list(100,200,300) @ hPa") no starting file is needed.

### select [optional]

definition of a selection criterion for command **select**. If the selection criterion is given explicitly in the Lagranto call (e.g. GT:PV:2), no selection file is needed.

## OUTPUT FILES AND STRUCTURE

For a Lagranto run a new directory will be created where all needed files are prepared. The name of the directory and the file within it (if cleaning is invoked) are:

**ntr\_\${startdate}\_{\$dir}\$\_{timerange}\_{\$startf}\_{\$select}**

for instance, the Lagranto call "lagranto local 20100101\_00 20100102\_00 startf selectf" will create the directory "ntr\_19891020\_00\_f24\_local\_startf\_selectf". Correspondingly, for a backward calculation the name would be "ntr\_19891021\_00\_b24\_local\_startf\_selectf", i.e. the {dir} is set to 'b'.

**ls\_\${startdate}**

default name of the output trajectory file placed in the ntr directory. The name can be changed with the option "-o filename". Note further that different output formats are supported, as described in the documentation for command **reformat**.

**runscript.sh**

name of the run script, i.e. the script within the ntr directory which calls all Fortran programs. It can be manually started with "./runscript.sh".

**runscript.logfile**

name of the log file; all status information is written to this file. If a Lagranto run fails, this is the place where to start looking for the reason!

**SPECIAL COMMANDS**

The main focus of **lagranto** is to combine the calls to "create\_startf", "caltra", "trace" and "select" into one convenient call. In addition to this, "lagranto" offers some handy special commands which allow more efficient working.

**-open caseid.label**

open a new **xterm** window and change to the run directory. If several directories with the same case ID are found, the user is interactively asked to choose one.

**-remove caseid.label**

remove a run directory. If several directories with the same case ID are found, the user is interactively asked to choose one.

**-show caseid.label**

show the contents of the trajectory file as a list. If several directories with the same case ID are found, the user is interactively asked to choose one.

**EXAMPLES****[1] `lagranto local 19891020_00 19891021_00 startf nil -changet`**

a forward trajectory calculation from 00 UTC, 20 October 2010 to 00 UTC, 21 October 2010. The starting positions are taken from the file "startf". No selection criterion is applied (nil), and the times on the input netCDF files are set relative to the starting date in advance.

**[2] `lagranto local 19891020_00 19891021_00 startf 'GT:PV:2:LAST' -changet`**

as in example [1], but now an explicit selection criterion is applied (GT:PV:2:LAST) - the potential vorticity at the end date (19891021\_00) must be larger than 2 PVU.

**[3] `lagranto local 19891020_00 19891021_00 startf 'GT:PV:2:LAST' -changet -prep`**

as in the previous two examples, but now only the files and runscript are prepared: no Lagranto run is launched! To do so, you might change to the run directory with **lagranto -open local** and then start it manually with **./runscript.sh**.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

**NAME**

**Lagranto** - calculate trajectories for ECMWF analyses and forecasts

**SYNOPSIS**

**lagrantohelp** [**topic**|**tutorial**|**reference**|**future**] - show this man page or the one corresponding to a specific topic. All *underlined* names in this document have their own man page. If "tutorial" is chosen, a detailed tutorial for Lagranto is opened. For "reference" a PDF document with all man pages is opened. Finally, with option "future" plans for additional features will be listed.

**DESCRIPTION**

Lagranto is a software tool (UNIX shell-scripts and Fortran programs) to analyze Lagrangian aspects of atmospheric phenomena. It requires a time-series of 3-dimensional wind fields (and if necessary further variables) on netCDF files.

**CORE PROGRAMS**

<i>create_startf</i>	create starting files
<i>caltra</i>	calculate trajectory positions
<i>trace</i>	trace meteorological fields along trajectories
<i>select</i>	select trajectories based on several criteria
<i>density</i>	create netCDF files with trajectory densities

**TRAJECTORY TOOLS**

<i>extract</i>	extract single trajectories, times or columns from a trajectory file
<i>list2lsl</i>	transform a (longitude,latitude,pressure) list into a trajectory file
<i>lsl2list</i>	transform a trajectory file into a (longitude,latitude,pressure) list
<i>mergetra</i>	merge two trajectory files
<i>reformat</i>	change the format of a trajectory file
<i>timeres</i>	change the time resolution of a trajectory file with interpolation
<i>trainfo</i>	write some information about a trajectory file
<i>difference</i>	calculate the difference between two trajectory files

**NETCDF TOOLS**

<i>changecest</i>	change the constants filename on a netCDF file
<i>changet</i>	change the time on a netCDF file
<i>getmima</i>	get the minimum and maximum value of a field on a netCDF file
<i>getvars</i>	get a list of fields on a netCDF file

**GENERAL TOOLS**

<i>newtime</i>	get a new date from an initial date and an offset in hours
<i>gettidiff</i>	get the time difference between two dates

**VISUALISATION**

Lagranto comes with two packages for visualisation of trajectories: one based on Matlab ([www.mathworks.com](http://www.mathworks.com)) and one based on NCL ([www.ncl.ucar.edu](http://www.ncl.ucar.edu)). Examples are provided in folder "figure" of the Lagranto installation.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)



**NAME**

**list2lsl** - convert a list of longitude, latitude and pressure into a trajectory file.

**SYNOPSIS**

**list2lsl** *infile* *outfile* [ *optional arguments* ]

**DESCRIPTION**

Convert an input file "infile" with a list of longitude, latitude and pressure into a trajectory file.

**PARAMETERS**

*infile*                   input file (a list of longitude, latitude and pressure)

*outfile*                  output trajectory file

**OPTIONAL PARAMETERS****-ref refdate**

set the reference date of the trajectory file to "refdate", e.g. "-ref 20110102\_00".

**-time value**

set the time of the trajectory file to "value" (in format HH.MM): e.g. "-time 6" sets the time to 6 h.

**EXAMPLES**

**[1] list2lsl listfile trafile -refdate 20110101\_00 -time 6**

converts the list "listfile" into a trajectory file "trafile", where the reference date is 20110101\_00 and the time is 6 h.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

lsl2list()

lsl2list()

**NAME**

**lsl2list - convert a trajectory file into a list of longitude, latitude and pressure**

**SYNOPSIS**

**lsl2list *infile* *outfile***

**DESCRIPTION**

Convert an input trajectory file "infile" into a list of longitude, latitude and pressure; all times are included in this list.

**PARAMETERS**

*infile*                   input trajectory file

*outfile*                 output file (a list of longitude, latitude and pressure)

**EXAMPLES**

**lsl2list *trafile* *listfile***

converts all entries of the trajectory file "trafile" into a listfile with longitude, latitude and pressure.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

**NAME**

**mergetra** - combine two trajectory files

**SYNOPSIS**

**mergetra** *inpf1* *inpf2* *outfile*

**DESCRIPTION**

Combines two input trajectory files "inpf1" and "inpf2" and writes a new combined trajectory file "outfile". Three different modes for combination are supported (see below).

**PARAMETERS**

*inpf1*            first input trajectory file  
*inpf2*            second input trajectory file  
*outfile*          output trajectory file

**MODES FOR COMBINATION**

Three different combination modes are supported. They are tested for sequentially and the first mode which is accepted, based on the criteria listed below, is performed. The three modes are:

**- column**

The input files have the same trajectory times, longitudes, latitudes and pressures, and both files contain the same number of trajectories. The new trajectory file contains all columns from both input files, duplicate columns only included once, e.g. if "inpf1" has columns "time,lon,lat,p,TH,PV" and "inpf2" columns "time,lon,lat,p,PV,Q", the output file will have "time,lon,lat,p,TH,PV,Q".

**- append**

The input files have the same trajectory times and the same columns, but differ in the number of trajectories. Then the second file is appended to the first one and written as a new trajectory file.

**- times**

The input files differ only in the time values, e.g. the first file might contain times -96 h to 0 h and the second the times from 0 h to 96 hours. Then the new combined file will contain the time -96 h to 96 h. Note that duplicate times are eliminated. Furthermore, the output times are sorted in increasing order.

**EXAMPLES**

[1] **mergetra file1 file2 newfile**

combines the two trajectory files "file1" and "file2" and writes a new trajectory file; the mode for combination is automatically detected.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

newtime()

newtime()

## NAME

**newtime** - get a new date string from an initial date and an offset in hours

## SYNOPSIS

**newtime** *date offset*

## DESCRIPTION

Given an initial date string in format "YYYYMMDD\_HH" and an offset in hours, create a new date string - again in the form "YYYYMMDD\_HH".

## PARAMETERS

*date*                    initial date in form "YYYYMMDD\_HH", e.g. 20110102\_18 for 18 UTC, 2nd January 2011.

*offset*                 Offset in hours; the offset can be positive (into the future) or negative (into the past).

## EXAMPLES

[1] **newtime 20110101\_00 45**  
gives 20110102\_21, the datestring 45 h after 20110101\_00.

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)

reformat()

reformat()

## NAME

**reformat - convert trajectory files between different formats**

## SYNOPSIS

**reformat** *infile* *outfile*

## DESCRIPTION

Convert a Lagranto trajectory file *infile* from one format to another format; the new file is written to *outfile*. The formats are specified with an appendix to the filename, e.g. "trajectory.1" specifies format 1. If no appendix is given, format 1 is chosen.

## PARAMETERS

*infile*                   input trajectory file  
*outfile*                  output trajectory file (can be the same as infile).

## FORMATS

Formats must be specified with an appendix to the filename. If no appendix is given, format 1 is chosen.

- .1**   ASCII file; the trajectories are sorted according to their starting positions. Different trajectories are separated by a blank line. This format is the only one supported by the Matlab and NCL visualisation scripts (see **lagrantohelp/visualisation**).
- .2**   ASCII file; the trajectories are sorted according to their times. Different times are separated with a blank line.
- .3**   unformatted Fortran; this is the most efficient format, but least portable one.
- .4**   netCDF; portable and compact data format.

## EXAMPLES

**[1] reformat trainp.1 trainp.4**

Convert the input trajectory file in format 1 (ASCII) to format 4 (netCDF).

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)

**NAME****rot2geo** - rotate from rotated to true longitude and latitude**SYNOPSIS****rot2geo** [ *infile outfile* ] [ *optional parameters* ]**rot2geo** [ *r lon rlat* ] [ *optional parameters* ]**rot2geo** [ *optional parameters* ]**DESCRIPTION**

Rotate the longitude and latitude coordinates from their values in a rotated coordinate system (r lon/r lat) to their true geographical values (lon/lat). The rotation can be done either for a single <lon/lat> point or for a complete trajectory file <infile>. In the latter case, a new trajectory <outfile> file with true longitude and latitude is written.

**PARAMETERS***infile* input trajectory file (with rotated lon/lat)*outfile* output trajectory file (with true lon/lat)*r lon rlat* rotated longitude and latitude**OPTIONAL PARAMETERS***-lonlat pollon pollat*

specify the coordinates (pollon, pollat) of the rotated north pole

*-file filename*

get the coordinates of the rotated north pole from the file &lt;filename&gt;. The file must be a netCDF file, either a constants or a P,S file.

**EXAMPLES****[1] rot2geo -lonlat -170 45**

sets the rotated north pole to 170 W and 45 N. The new position is written to the file &lt;rot2geo.pole&gt;.

**[2] rot2geo 45 50**

takes roated coordinate (r lon/r lat)=(45/50) and rotates it to the geographical coordinate system. The position of the rotated pole is read from the file &lt;rot2geo.pole&gt;.

**[3] rot2geo 45 50 -file eraint\_cst**

rotates (r lon/r lat) = (45/50) to the new system - the new pole position is read from the file &lt;eraint\_cst&gt;.

**[4] rot2geo trainp traout**

read the trajectory file &lt;trainp&gt; and rotate all coordinates. The trajectory file with geographical coordinates is written to &lt;traout&gt;.

**[5] geo2rot**

write the pole position to screen.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (July 2011)

select()

select()

## NAME

**select** - select trajectories

## SYNOPSIS

**select** *inpra outtra criterion*

## DESCRIPTION

Select trajectories from the input trajectories in "inpra" based upon meteorological conditions specified in "criterion" (to be described below). The selected trajectories are then written to a new trajectory file "outtra".

## PARAMETERS

*inpra*                   input trajectory file  
*outtra*                   output trajectory file  
*criterion*               specification of the selection criterion; the specification is either an explicit criterion or a file containing the specification. Each selection criterion has the following form:  
**COMMAND : FIELD : PARAMETERS : [ TIME ]**.  
Several selection criteria can be combined with logical operator & (AND) and | (OR), the AND having higher priority than the OR.

## OPTIONAL PARAMETERS

*-noclean*  
keep temporary files for debugging.  
*-boolean*  
Write a boolean list (0/1) instead of a trajectory file; the file has #trajectories entries, each line corresponding to an input trajectory /1=trajectory selected, 0=not selected).  
*-index*  
Write an index list instead of a trajectory file: the index of all selected trajectories is written to the output file - the index ranges from 1 to #trajectories.  
*-count*  
Write only the number of selected trajectories to the output file.  
*-startf*  
Write only the starting positions of selected trajectories to the output file.  
*-regionf filename*  
change the region file from its default value "regionf" to a new file name: the syntax is "-regionf filename".

## COMMAND

- **GT**  
greater than: e.g. **GT:PV:2** selects trajectories with first potential vorticity (PV) larger than 2 PVU.  
- **LT** less than: e.g. **LT:RH:70** selects trajectories with first relative humidity (RH) below 70 %.  
- **IN** within: e.g. **IN:lon:30,40** selects trajectories with first longitude between 30 and 40 deg.  
- **OUT**  
outside: e.g. **OUT:lat:-30,30** selects trajectories with first latitude outside -30 and 30 deg - neglecting an equatorial/subtropical band.  
- **EQ**  
equal: e.g. **EQ:p:460** selects trajectories with first pressure equal to 460 hPa.  
- **TRUE**  
check whether value is different from zero (logical TRUE): e.g. **TRUE:CYCL::ALL(ANY)** checks whether the trajectory passes through a cyclone, which is marked as 0/1 field, at any time. Note that the command 'TRUE' has no arguments of its own!

select()

select()

**- FALSE**

check whether value is equal to zero (logical FALSE): e.g. **FALSE:CYCL::ALL(ALL)** checks whether the trajectory never passes through a cyclone, which is marked as 0/1 field. Note that the command 'TRUE' has no arguments of its own!

**- ALL, ANY, NONE**

these are special commands which only apply for the TRIGGER field. Further explanations are given below in section TRIGGER.

**FIELD**

**- VALUE**

take value of the field: e.g. **GT:PV(VALUE):2** selects the trajectories with first potential vorticity (PV) value greater than 2 PVU. This selection criterion is equivalent to **GT:PV:2**, i.e. the VALUE argument is taken as default.

**- MEAN**

take the mean over the selected times: e.g. **GT:RH(MEAN):70:ALL** selects all trajectories for which the mean relative humidity (RH) over all times (ALL) is greater than 70 %.

**- VAR**

take the variance over the selected times: e.g. **GT:lat(VAR):10:ALL** selects all trajectories for which the variance of latitude (lat) over all times (ALL) is greater than 10.

**- MIN**

take the minimum of the selected times: e.g. **LT:p(MIN):300:ALL** select all trajectories which have a minimum pressure (p) less than 300 hPa over all times (ALL).

**- MAX**

take the maximum of the selected times: e.g. **LT:p(MAX):300:ALL** select all trajectories which have a maximum pressure (p) less than 300 hPa over all times (ALL).

**- SUM**

take the sum over the selected times: e.g. **GT:LHR(SUM):2:ALL** selects all trajectories for which the sum over all latent heating rates (LHR) over all times (ALL) is greater than 2 K.

**- CHANGE**

take change between two times: e.g. **GT:p(CHANGE):600:FIRST,LAST** selects all trajectories which have a pressure difference  $|p(\text{FIRST})-p(\text{LAST})|$  greater than 600 hPa between the first and last time or vice versa. Note that the change can be positive or negative, i.e. it is not clear whether it is ascent or descent.

**- DIFF**

take difference between two times: e.g. **GT:p(DIFF):600:FIRST,LAST** selects all trajectories which have a pressure difference  $p(\text{FIRST})-p(\text{LAST})$  greater than 600 hPa between the first and last time - corresponding to an ascending air stream. Correspondingly **GT:p(DIFF):600:LAST,FIRST** finds a descending air stream.

**TIME MODE**

The command are applied to a set of trajectory times; if no time is specified, the the command is only applied to the first time. Most generally, the time mode consists of two parts: time list( time mode), where the first specifies a list of times and the second how to apply the criterion to the selected times.

**- FIRST**

first time: e.g. **IN:lat:-20,20:FIRST** selects all trajectories with first latitude between 20 S to 20 N, i.e. which start in an equatorial band.

**- LAST**

last time: e.g. **IN:lat:-20,20:LAST** selects all trajectories with last latitude between 20 S to 20 N, i.e. which end in an equatorial band.



select()

select()

#### - T1,T2,T3

an explicit list of times: e.g. **IN:lat:-20,20:6,12** selects all trajectories which are in the equatorial band at times 6 h and 12 h. The criterion must apply at both times (see below ALL, ANY, NONE).

#### - T1 to T2

a time range: e.g. **IN:lat:-20,20:6 to 18** selects all trajectories which are in the equatorial band from 6 h to 18 h. The criterion must apply at all times between 6 h and 18 h (see below ALL, ANY, NONE).

#### - ALL

all times: e.g. **IN:lat:-20,20:ALL** selects all trajectories which stay at all times in the equatorial band. This time mode is the same as **ALL(ALL)**, i.e. all times are selected and the criterion must apply to all times. With **IN:lat:-20,20:12-24(ALL)** the criterion must apply for all times between 12 h and 24 h.

#### - ANY

any times: e.g. **IN:lat:-20,20:ALL(ANY)** selects all trajectories which stay at any times in the equatorial band. Note that with the first "ALL" the times are selected, i.e. all times in this case, and with the second "ANY" it is specified that the criterion must only apply to at least one of the selected times.

#### - NONE

at no time: e.g. **IN:lat:-20,20:ALL(NONE)** selects all trajectories which never stay in the equatorial band. **OUT:lat:-20,20:FIRST(NONE)** selects the trajectories which are not outside the equatorial time at the first time: they must be inside.

#### - TRIGGER

the trajectory is automatically selected, but the trigger column is updated. A selection of trajectories might then be accomplished based on this trigger column: e.g. **GT:p:700:1(TRIGGER)** would set the trigger 1 for all trajectory times where the pressure (p) is greater than 700 hPa. Similarly, **GT:p:800:1(TRIGGER) & GT:lat:50:2(TRIGGER)** would set the trigger 1 for all times where the pressure (p) is larger than 800 hPa and set the trigger 2 for all times where the latitude (lat) is larger than 50 degrees north. Note, that both events might apply: then the resulting trigger is 3. The triggers are internally saved as the bits of an integer variable, i.e. trigger 1 corresponds to value  $1=2^0$ , trigger 2 to  $2=2^1$ , trigger 3 to  $4=2^2$ ...

### LOGICAL OPERATORS

- & logical and: e.g. **GT:lat:34:FIRST & GT:lon:50:FIRST** selects the trajectories to the north of 34 N and to the east of 50 E at first time. Several selection criteria can be combined with '&'.

- | logical or: e.g. **GT:lat:34:FIRST | GT:lon:50:FIRST** selects the trajectories to the north of 34 N or to the east of 50 E at first time. Several selection criteria can be combined with '|'. Note that logical OR has a lower priority than logical AND, i.e. in an expression like **T1 | T2 & T3** first the expression **T2 & T3** is evaluated and only then logically OR-combined with **T1**.

### IMPLICIT FIELDS

Implicit variables can be used in the selection criteria, although they do not explicitly appear as a column in the trajectory file. They are calculated on-the-fly during the selection.

#### - DIST

length of the trajectory (in km), integrated along great circle sections between the trajectory vertices: e.g. **GT:DIST:1000:LAST** tests whether the total path length of the trajectory (DIST) exceeds 1000 km.

#### - DIST0

distance of the trajectory (in km) from its starting position: e.g. **GT:DIST0:1000:18,24(ANY)** tests whether the air parcel is more than 1000 km away from its starting position, either at time 18 h or at time 24 h.

select()

select()

#### - INPOLYGON

specification of a polygon: e.g. **TRUE:INPOLYGON:filename:ALL(ANY)** selects all trajectories which pass through the polygon specified in "filename". The polygon is specified as described in **create\_startf** (see comments there). With **FALSE:INPOLYGON:filename:ALL(ALL)** all trajectories are selected which never pass through the polygon. Note that for every call of **select** only one polygon can be used in the criteria!

#### - INBOX

specification of a longitude/latitude rectangle <lonw,lonl,lats,latn>: e.g. **TRUE:INBOX:20,40,30,60:ALL(ANY)** selects all trajectories which pass through the longitude/latitude rectangle with lower-left corner at 20 E / 30 N and the upper-right corner at 40 E / 60 N. Correspondingly, with **TRUE:INBOX:20,40,30,60:ALL(NONE)** the trajectories are selected which never pass through the rectangle.

#### - INCIRCLE

specification of circle around a specified point: e.g. **TRUE:INCIRCLE:40,50,500:LAST** select all trajectories which have their final position (LAST) in the circle centred at 40 E / 50 N and with a radius of 500 km.

#### - INREGION

specification of target regions in a region file (default "regionf") - please consider the documentation of "create\_startf" for details concerning the format of the region file. As an example, if a region 1 is defined on the region file, the criterion **TRUE:INREGION:1:18** selects all trajectories which are within region 1 at time 18 h.

### SPECIAL CRITERIA

Special criteria are and can be implemented into **select**. The call to the special criteria is of the following form: **SPECIAL:command:parameters**, where "command" is a command string (e.g. WCB) and "parameters" is a list of parameter values.

#### - WCB

identification of Warm Conveyor Belts (WCB): e.g. **SPECIAL:WCB:300,0,24** selects trajectories which ascend more than 300 hPa between time 0 and 24 h. Note, the ascent is determined as  $\min\{p(0...24)\}-p(0)$ , i.e. the first pressure  $p(0)$  is fixed whereas the lowest pressure can occur at any time between 0...24 h.

### TRIGGER FIELD

A trigger (or flag) field can be defined in **select**. This special column of the trajectory file allows to mark specified events for each trajectory and its times. As an example, you would like to select all trajectories which are below 700 hPa at a certain time and have relative humidity above 80 % **at the same times**: Then you could set a first trigger for the pressure criterion and a second one for the relative humidity, and then you would check for the simultaneous occurrence of the two triggers. More specifically,

#### **GT:p:700:1(TRIGGER) & GT:RH:80:2(TRIGGER)**

this will define the triggers for the two events. Note that both events might be fulfilled, in which case both triggers are set. The value of the trigger is: 1 if only the first criterion is fulfilled (binary 01); 2 if only the second is fulfilled (binary 10) and 3 if both are fulfilled (binary 11).

#### **ALL:TRIGGER:1,2:ALL(ANY)**

this will select all trajectories for which both triggers 1 and 2 are set - they can be set at any time of all the trajectory times; if, for instance, you would like to test whether the two triggers are set at the two times 6 and 12, the command would be **ALL:TRIGGER:1,2:6,12(ALL)**.

#### **ALL:TRIGGER:1,2:ALL(ANY)**

will only check whether one of the two triggers 1 and 2 is set

#### **NONE:TRIGGER:1,2:ALL(ANY)**

will check whether none of the two triggers 1 and 2 is set

select()

select()

## **IMPLEMENTING COMPLEX CRITERIA**

New special criteria can easily be implemented into the code - to this aim the following steps must be taken:

**[1] <special.f>**

in directory "\${LAGRANTO}/select/" must be modified. Take the example "WCB" to see how the Fortran code must be changed.

**[2] <install.sh select>**

must be invoked to recompile the program "select". For successful compilation, the executable "select" will be listed.

Type <select -special> to get a list of all special commands and a detailed description.

## **AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)

**NAME**

**create\_startf** - create starting files for Lagranto

**SYNOPSIS**

**create\_startf** *date filename specifier* [ *optional arguments* ]

**DESCRIPTION**

Create starting files for a Lagranto calculation. The starting positions are based on the P and S files for *date* and are as specified in a *specifier*. The starting coordinates (longitude, latitude, pressure [in hPa]) are written to the file *filename*.

**PARAMETERS**

*date*                    date of input P and S file (e.g. 20100101\_00). If the date is between two P and S files, linear interpolation is used between the two times.

*filename*                output file with starting points (e.g. startf). Different formats are supported (see **reformat** for details)

*specifier*                detailed description of starting positions. The specifier has the following format: *<horizontal>* @ *<vertical>* @ *<unit>* @ *<selection>*. The components of the specifier are described in greater detail in the following sections.

**HORIZONTAL**

- **file[filename]**    read lat/lon from file "filename"; each line contains one lat/lon pair.
- **line[lon1,lon2,lat1,lat2,n]**  
n points from (lon1,lat1) to (lon2,lat2); the points are linearly interpolated in lat/lon space.
- **box.eqd[lon1,lon2,lat1,lat2,ds]**  
lat/lon box bounded with south-western point (lon1,lat1) and north-eastern point (lon2,lat2); the equidistant points within the box have a horizontal distance ds (in [km]).
- **box.grid[lon1,lon2,lat1,lat2,]**  
lat/lon box with south-western point (lon1,lat1) and north-eastern point (lon2,lat2) grid points; all grid points within this box are taken as starting points.
- **point[lon,lat]**    single lon/lat point.
- **shift[lon,lat,dlon,dlat]**  
lon/lat points and dlon/dlat shifted ones, i.e. in total five points: central one and four shifted ones: (lon,lat), (lon+dlon,lat), (lon-dlon,lat), (lon,lat+dlat), (lon,lat-dlat).
- **polygon.eqd[filename,ds]**  
equidistant within arbitrary polygon (ds in [km]). The file with the polygon points has the following format: 1st line a lon/lat point within the polygon; further lines lon/lat points of the vertices (max 500) of the polygon.
- **polygon.grid[filename]**  
grid points within arbitrary polygon. The file with the polygon points has the following format: 1st line a lon/lat point within the polygon; further lines lon/lat points of the vertices (max 500) of the polygon.
- **circle.eqd[lonc,latc,radius,ds]**  
circle with centre at (lonc,latc) and radius "radius" (in km); the equidistant points within the circle have a horizontal distance ds (in [km]).
- **circle.grid[lonc,latc,radius]**  
circle with centre at (lonc,latc) and radius "radius" (in km); all grid points within the circle are selected.
- **region.eqd[id,ds]**  
Read region specification from region file ("default regionf", to be changed with option "-regionf") and fill it equidistantly with starting points (ds in km). The region

identification is "id", see below in section REGION FILE.

- **region.grid[id]** Read region specification from region file ("default regionf", to be changed with option "-regionf") and fill it with starting points on the input grid. The region identification is "id", see below in section REGION FILE.

## VERTICAL

- **file[filename]** read levels from file "filename"; each line in the file contains one level.
- **level[lev]** a single level
- **list[lev1,lev2,lev3,...]**  
a list of levels; if many levels are needed they are better passed to "create\_startf" with the option "file[filename]".
- **profile[lev1,lev2,n]**  
n equidistant levels between lev1 and lev2.

## UNIT

- **hPa** pressure (in hPa).
- **hPa,agl** pressure (in hPa) above ground level.
- **K** potential temperature (in K).
- **PVU** potential vorticity (in PVU). Note that potential vorticity (PV) might not be unique as a vertical coordinate; if several levels have a given PV value, the highest one is chosen.

## SELECTION

- **criterion** Selection criteria based on meteorological fields applied to the starting position; The criteria follow the syntax of the program **select**.
- **nil** If no selection criteria should be invoked, the argument "nil" should be given.

## REGION FILE

Several starting regions can be defined for every case in a region file (default filename is "regionf"; to be changed with optional parameter "-regionf filename"). There are two possible formats for specifying a region (they require either a line with 5 or 9 entries):

### **regnum lonw lone lats latn**

a regular latitude-longitude square: regnum=integer region number; lonw=westernmost longitude of starting region; lone=easternmost longitude; lats=southernmost latitude; latnNorthernmost latitude.

### **regnum lon1 lat1 lon2 lat2 lon3 lat3 lon4 lat4**

an irregular latitude-longitude square: regnum=integer region number; lon{x},lat{x} = longitude and latitude of the x-th corner. Note that the 4 corners must be arranged counterclockwise. For a triangle the 4th corner can be specified identically to the 3rd.

**Note: (1) if a line starts with '#' it is regarded as comment and not further considered; (2) each line in the region file must start with ' '!**

**101 -40. -24. 52. 60. :**

region in the central Atlantic from 40 W to 24 W and 52 N to 60 N; the region identifier is 101.

**250 -30. 43. -24. 36. -18. 50.2 -35.2 50.2 :**

irregular square in the central Atlantic; the region identifier is 250.

## OPTIONAL

- t tracefile* tracing file with variables for selection criteria (see **trace** for format of the file). If no file is specified, the default "tracevars" is used. Further, if no selection criterion is invoked, no tracing file is necessary.
- changet* flag whether the times of the P and S files should be changed or not before a calculation; the default is that the times are not changed.

- noclean* flag whether parameter and criterion files should be kept; this is particularly helpful for debugging.
- regionf filename* change the region file from its default value "regionf" to a new file name: the syntax is "-regionf filename".
- notimecheck* take the first time on the netCDF file - do no explicit test that the requested time is available on the file. This is particularly helpful if you have no write permission for the P files.

## EXAMPLES

**[1] create\_startf 19891020\_00 startf 'point(-10,50) @ list(450,500,550) @ hPa'**

Starting points are (longitude, latitude, pressure in hPa): (-10,50,450); (-10,50,500); (-10,50,550). No selection criterion is applied; the positions are written to file "startf".

**[2] create\_startf 19891020\_00 startf 'line(-10,-5,40,50,10) @ level(450) @ hPa,agl'**

10 points are equidistantly specified between lon/lat point (-10,40) and (-5,50); all trajectories start at 450 hPa above ground level - the surface pressure is taken from the primary file P19891020\_00. The positions are saved in "startf".

**[3] create\_startf 19891020\_00 startf 'box.grid(-10,-5,40,50) @ list(300,320) @ K'**

All grid points in the box with the south-eastern lon/lat point (-10,40) and the north-eastern one (-5,50) are taken - the horizontal grid spacing is specified in the primary file P19891020\_00. In the vertical, two isentropic levels are chosen: 300 K and 320 K. The potential temperature for the calculation is taken from the secondary file S19891020\_00.

**[4] create\_startf 19891020\_00 startf 'shift(-10,40,1,1) @ profile(1000,200,100) @ hPa'**

A profile of 100 equidistant levels between 1000 hPa and 200 hPa; in the horizontal the central lon/lat point (-10,40) is taken and four horizontally displaced ones, the displacement being 1 degree in zonal and meridional direction.

**[5] create\_startf 19891020\_00 startf.1 'shift(-10,40,1,1) @ profile(1000,200,100) @ hPa'**

As in the previous example [4], but the starting positions are saved as a trajectory file instead of a (lon,lat,p)-list.

**[6] create\_startf 19891020\_00 startf.1 criterion**

As in the previous example [5], but the criterion is saved on a file with filename "criterion".

**[7] create\_startf 19891020\_00 startf 'polygon.grid(polygon) @ level(500) @ hPa'**

A polygon is specified in the file "polygon"; the different lines in the file are: -5. 45. / -10. 40. / 10. 40. / 10 50. / -10. 45. The first lon/lat point lies within the polygon, all other lon/lat points are the vertices of the polygon. All grid points within the polygon are taken as starting point, at level 500 hPa.

**[8] create\_startf 19891020\_00 startf 'polygon.eqd(polygon,50) @ level(500) @ hPa'**

As in the previous example [7], except that the starting points are distributed equidistantly within the polygon. The horizontal distance between the starting points is 50 km in zonal and meridional direction.

**[9] create\_startf 19891020\_00 startf 'shift(-10,40,1,1) @ profile(1000,200,100) @ hPa @ GT:TH:310'**

As in example [4], but a selection criterion is additionally applied: only starting positions with potential temperature (TH) greater than (GT) 310 K are kept. Potential temperature must be available on the secondary file S19891020\_00 and the file "tracevars" must have a line with "TH 1. 0 S". Further examples for selection criteria can be seen in **select**.

**[10] create\_startf 19891020\_00 START.1 'region.eqd(3,10) @ level(500) @ hPa'**

get equidistant starting points (10 km distance) in the region with identifier 3, as listed in the region file "regionf" (the default).

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)

create\_startf()

create\_startf()

timeres()

timeres()

## NAME

**timeres - change the time resolution of a trajectory**

## SYNOPSIS

**timeres** *infile outfile* *[-h/min] value* *[-cubic/-linear]*

## DESCRIPTION

Change the time resolution of an input trajectory file "infile" through interpolation and write a new trajectory file "outfile". The new time resolution "value" is given either in hours "-h" or in minutes "-min". The interpolation is performed on the trajectory file either in linear mode ("-linear") or in cubic spline mode ("-cubic"). The default is "-cubic".

## PARAMETERS

<i>infile</i>	input trajectory file
<i>outfile</i>	output trajectory file (can be the same as infile).
<i>-h value</i>	new time resolution in hours (e.g. "-h 1").
<i>-min value</i>	new time resolution in minutes (e.g. "-min 15").
<i>-linear</i>	linear interpolation between two trajectory times; note that this mode conserves the sign between two trajectory times - possibly of important for specific humidity, relative humidity,...
<i>-linear</i>	subic spline interpolation between two trajectory times; note that this mode conserves can change the sign between two trajectory times!

## EXAMPLES

**[1] timeres trfile trfile -min 15 -linear**

changes the time resolution to 15 minutes, using linear interpolation, and overwrites the old trajectory file.

## AUTHOR

Written by Michael Sprenger and Heini Wernli (January 2011)



**NAME****trace - trace meteorological fields along trajectories****SYNOPSIS****trace** *infile outfile* [ *optional arguments* ]**DESCRIPTION**

Trace meteorological fields along the trajectories given in the input file *infile* and write a new trajectory file *outfile* *tracing file* (default: *tracevars*). Partly they can be computed "online" (see ONLINE CALCULATIONS below), normally they are available on the primary and secondary P and S files.

**PARAMETERS**

*infile*                   input trajectory file; the appendix determines the format (see **reformat** for details).  
*outfile*                   output trajectory file; the appendix determines the format (see **reformat** for details).

**TRACING FILE**

Normally the meteorological fields for tracing are listed in a file with name **tracevars** . The name of the tracing file can be changed with the optional argument "-v" (see below). The format of the tracing file is as follows:

Format

*field[:shift] scale computation prefix*

Shifts (optional)

- **field:+100km[lat]** - get field at trajectory position + 100 km shifted to north. A shift to south is obtained with **field:-100km[lat]**.
- **field:+100km[lon]** - get field at trajectory position + 100 km shifted to east. A shift to west is obtained with **field:-100km[lon]**.
- **field:+2[dlat]** - get field at trajectory position + 2 grid spacings dlat shifted to north. A shift to south is obtained with **field:-2[dlat]**.
- **field:+2[dlon]** - get field at trajectory position + 2 grid spacings dlon shifted to east. A shift to west is obtained with **field:-2[dlon]**.
- **field:+50hPa** - get field at trajectory position + 50 hPa shifted in vertical. A shift to lower pressures is obtained with **field:-50hPa**.
- **field:+1dp** - get field at trajectory position + 1 grid spacing DP shifted in vertical. A shift to lower pressures is obtained with **field:-1dp**. Note that DP is not fixed but varies with height.
- **field:+6h** - get field at trajectory position, but 6 h in the future. Shifts to the past are possible with **field:-6h**. In addition to hours (h), the time shift can be specified in minutes (min).

Examples

- **TH 1. 0 S** : trace potential temperature (TH), scale it with 1 (no scaling); it is available on the S file (no computation is needed: 0).
- **Q 1000. 0 P** : trace specific humidity (Q), scale it with 1000 to have g/kg; it is available on the P file (no computation is needed: 0).
- **RH 1. 1 P** : trace relative humidity (RH), no scaling is needed (1.); relative humidity is not available on either P or S file and must be computed (1).
- **TH:100hPa 1. 0 S** : As in the first example, but now the potential temperature is taken 100 hPa below the air parcel position.

**OPTIONAL ARGUMENTS**

-*i hours*                   time increments (in hours) for input P and S files. If not explicitly specified, this is determined from the P and S files in the current directory.  
 -*v varfile*                   Change the name of the tracing file from its default value "tracevars" to "varfile".  
 -*f field scale*               Trace field (with scaling scale) along the trajectories; the computation flag and the prefix for the data file is automatically set. This options allows the quick tracing of a field, without specifying a tracing file.

- changet* flag whether the times of the P and S files should be changed or not before a calculation; the default is that the times are **not** changed.
- noclean* flag whether parameter and criterion files should be kept; this is particularly helpful for debugging.
- notimecheck* take the first time on the netCDF file - do no explicit test that the requested time is available on the file. This is particularly helpful if you have no write permission for the P files.

## ONLINE CALCULATIONS

If the computation flag in the tracing file is set to 1, a meteorological field is calculated based upon the already traced fields and/or based on the fields on the primary and secondary P and S files. The following fields are implemented for online calculations:

- **TH**  
potential temperature (in K).
- **RHO**  
density (in  $\text{kg/m}^3$ ).
- **RH**  
relative humidity (in %).
- **THE**  
equivalent-potential temperature (in K).
- **LHR**  
latent heating rate (K per input time step, typically K/6h).
- **D[U,V,T,TH]DX**  
horizontal derivative  $d[U,V,T,TH]/dx$  in west-east direction along pressure surfaces - zonal distance in m. U=zonal wind component (m/s), V=meridional wind component (m/s), T=temperature (deg C or K), TH=potential temperature (K).
- **D[U,V,T,TH]DY**  
horizontal derivative  $d[U,V,T,TH]/dy$  in south-north direction along pressure surfaces - meridional distance in m.
- **D[U,V,T,TH]DP**  
vertical derivative  $d[U,V,T,TH]/dp$  - pressure p in Pa.
- **NSQ**  
squared Brunt-Vaisala frequency (in  $\text{m}^{-2}$ ).
- **RELVORT**  
relative vorticity (in  $\text{s}^{-1}$ ) -  $\text{RELVORT} = \text{DV DX} - \text{DUDY}$ .
- **ABSVORT**  
absolute vorticity (in  $\text{s}^{-1}$ ) -  $\text{ABSVORT} = \text{DV DX} - \text{DUDY} + F$ , F being the Coriolis parameter.
- **DIV**  
horizontal divergence of the velocity field (in  $\text{s}^{-1}$ ) -  $\text{DIV} = \text{DUDX} + \text{DV DY}$ .
- **DEF**  
horizontal deformation of the velocity field (in  $\text{s}^{-1}$ ) -  $\text{DEF} = \text{SQRT}((\text{DV DX} + \text{DUDY})^2 + (\text{DUDX} - \text{DV DY})^2)$ .
- **PV** Ertel potential vorticity (in PVU) -  $\text{PV} = g * (\text{ABSVORT} * \text{DTH DP} + \text{DUDP} * \text{DTH DY} - \text{DV DP} * \text{DTH DX})$ .
- **RI** Richardson number -  $\text{RI} = \text{NSQ} / (\text{DUDP}^2 + \text{DV DP}^2)$ .
- **TI** turbulence indicator according to Ellrod & Knapp -  $\text{TI} = \text{DEF} * \text{SQRT}(\text{DUDP}^2 + \text{DV DP}^2) * (\text{RHO} * G)$ .

**- DIR**

wind direction relative to zonal flow: (U,V)=(1,1) -> 45 deg; (U,V)=(1,-1) -> -45 deg; (U,V)=(-1,-1) -> -135 deg; (U,V)=(-1,1) -> 135 deg. A westerly flow has 0 deg, a southerly flow 90 deg, and a northerly one -90 deg.

**- DIST0**

spherical distance (in km) from starting position.

**- DIST**

length of the trajectory (in km): integrated along great circle sections between the trajectory vertices.

**- HEAD**

heading of the trajectory: (DX,DY)=(1,1) -> 45 deg; (DX,DY)=(1,-1) -> -45 deg; (DX,DY)=(-1,-1) -> -135 deg; (DX,DY)=(-1,1) -> 135 deg. A path increment to east has heading of 0 deg; to the north 90 deg; to the south -90 deg; and to the west -180 deg.

**EXAMPLES****[1] /home/sprenger/lagranto/bin/trace TRAJECTORY.1 TRAJECTORY.1 -changet**

Read the trajectory file TRAJECTORY.1, trace all fields in the file "tracevars" along the trajectories and overwrite the existing trajectory file. In preparation, all times on the P and S files are changed prior to the tracing.

**[2] trace INPTRA.1 OUTTRA.1 -f PV 1.**

Trace PV (with scaling factor 1.) along the trajectories in trajectory file "INPTRA.1" and write a new trajectory file "OUTTRA.1".

**[3] trace INPTRA.1 OUTTRA.1 -f PV:-100HPA 1.**

As in example [2], but the PV is taken at a position 100 hPa higher (lower pressure) than the air parcel's position.

**[4] trace INPTRA.1 OUTTRA.1 -f DIST0 1.**

Get the spherical distance (in km) of the air parcel from its starting position.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011).

**NAME**

**trainfo - write meta-information for a trajectory file**

**SYNOPSIS**

**trainfo** *trafile* [ *option* ].

**DESCRIPTION**

Write meta-information for a trajectory file *trafile* to screen. If no option is given, all meta-information is written, otherwise the specific piece of information is passed with *option*.

**PARAMETERS**

*trafile*  
name of the input trajectory file

**OPTIONAL ARGUMENTS**

- **dim**  
dimensions of the trajectory file: #tra, #ntimes, #ncolumns.
- **ntra**  
number of trajectories.
- **ntim**  
number of times.
- **ncol**  
number of columns (including time, longitude, latitude, pressure).
- **vars**  
list of field names (columns) on the trajectory file.
- **refdate**  
reference date in format (YYYYMMDD\_HHMM).
- **times**  
list of times (relative to the reference date). Times are given in format: HH.MM.
- **startdate**  
starting date for the trajectory calculation (in format YYYYMMD\_HHMM).
- **enddate**  
end date for the trajectory calculation (in format YYYYMMD\_HHMM).
- **timerange**  
time range (in minutes) of the trajectories.
- **list** list all trajectories.

**EXAMPLES****[1] trainfo file dim**

Given a trajectory file with name "file", write the three dimensions of the file to screen. The dimensions are: number of trajectories, number of times and number of columns.

**AUTHOR**

Written by Michael Sprenger and Heini Wernli (January 2011)