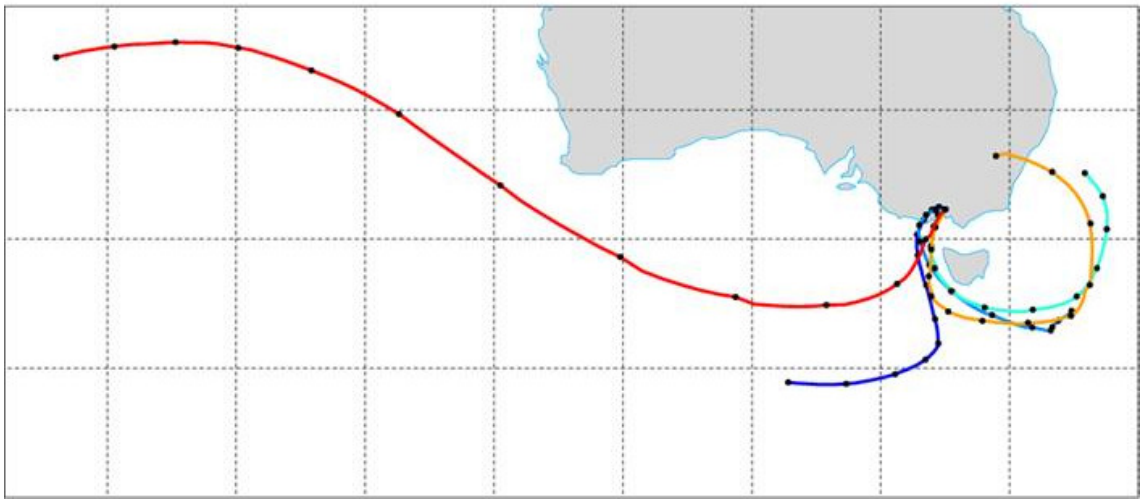
A three-dimensional spherical trajectory algorithm: `traj3d`

Ian Simmonds and Kevin Keay

School of Earth Sciences
The University of Melbourne 3010 Australia

Version 1.0

December 22 2009



Contents

1. Introduction

Parcel trajectories are often calculated to obtain an appreciation of the history of air masses (e.g. Fuelburg et al., 1996). The direction and length of trajectories are useful in diagnosing processes that may affect particular air masses under certain conditions. In this regard it is important that parcel trajectories are determined accurately.

The three-dimensional (3D) algorithm presented here (`traj3d`) builds on the two-dimensional (2D) model of Law (1993) and reported in application by Perrin and Simmonds (1995). From a specified parcel location in the atmosphere, $\mathbf{x}_n$ at time n, a finite integral is solved to advect the parcel and generate the trajectory path. Given the 3D wind $\mathbf{v}(\mathbf{x}_n)$ the governing prognostic equation for the trajectory path over a short time interval Δt is $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}\,\Delta t$. The wind at a given point is found by cubically interpolating from a spatial grid then linearly in time. For back trajectories the wind direction is reversed. The finite integral is solved using a fourth-order Runge-Kutta scheme to obtain an estimate of the wind. This method is considerably more accurate for trajectory calculations than a simple first-order approach. For mathematical details about the 3D algorithm see Noone and Simmonds (1999) and Barras and Simmonds (2009).

We present an implementation of the 3D algorithm in Fortran 77 that is suitable for any platform given an appropriate compiler. A precompiled Linux version is also available. The input to the program is a simple binary format referred to as CMP. At present the input data (usually 3D winds) are limited to constant pressure levels (e.g. 1000, 925, 850 hPa etc) as is the case with the commonly available reanalysis products e.g. NCEP reanalysis. The output of the program is a NetCDF file (and therefore NetCDF libraries need to be accessible for compilation) and this may be imported into many software packages e.g. Matlab, NCL, for subsequent analysis and plotting. We include some utilities to convert from NetCDF (a common format for 3D meteorological fields like winds) to CMP as well as plotting the trajectories based on NCAR Graphics/NCL.

2. Running the program

Firstly, the program `traj3d` needs to be compiled or a suitable binary version installed – see Section 5.1.
Secondly, the relevant CMP data files need to be available. This will probably require a conversion from NetCDF to CMP. In addition auxiliary 2D and 3D fields may be prepared.
Thirdly, a Fortran *namelist* file should be set up to control the behaviour of the program. Normally this is to compute the 3D trajectories based on the input CMP files containing the wind fields (U, V, W) with any (optional) auxiliary variables interpolated to the trajectory locations.
Then to run the program:

`traj3d` *`namelistfile`*

e.g. `traj3d input_backward_ncep.4.nml`

During execution some information and other diagnostic messages will appear on the screen. Some of these are related to issues like the need to reverse the reading of level data if the levels are not arranged in increasing order or excluding locations if they are outside of the grid. If the program completes successfully then an output NetCDF file will be created. This contains the input locations (start or end depending on the value of the parameter forward) and the trajectory variables: longitude, latitude, level and optionally date. If auxiliary variables were selected then these will also appear in the NetCDF file.
If the diagnostic options `debug` and `debug2` have been activated then there will be additional files (fort.*) and datetraj.txt.

3. Description of namelist parameters for trajectory program (traj3d)

The program is controlled by a Fortran namelist file which contains a list of parameters and their values. This is a text file of the form:

```
$traj3d
parameter1=value1
parameter2=value2
parameter3=value31,value32, parameter4='value4'
…
$end
```

A parameter could be a single number, an array of numbers or a character string
e.g.
```
x = 3.45
vec=1.2,-3.8,10.1
label= 'Some text'
```
A line may contain more than one parameter
e.g. `x = 3.45, vec=1.2,-3.8,10.1, label= 'Some text'`

A description of these parameters is given in Table 1. The parameters in this table are sufficient to compute trajectories from input CMP data files containing the wind fields (U,V,W) organised by levels. Note: it is assumed that U and V are in m/s and W is in Pa/s. For 2D trajectory computation the W field is not used (or may be omitted from the input file).

Table 1 Namelist parameters for `traj3d`

Parameter	Description	Value	Default	Comments
`traj2d`	Turns on 2D trajectory computation (default is 3D trajectory computation)	`.true.` or `.false.`	`.false.`	Use with a single level (`nlev=1`) or for computing 2D trajectories with multi-level data (`nlev` > 1) For a single level it may be desirable to set the parameters `longlev1` and `levunit1`
`debug`	Turns of diagnostic information. See files: fort.21: fort.22: If `ncdate=.true.`: fort.2: Flag, tsdate (flag: 0 forward 1 backward) datetraj.txt: List of dates for each output time step	`.true.` or `.false.`	`.false.`	Use for checking data input
`debug2`	Turns on additional diagnostic information. See file: fort.24:	`.true.` or `.false.`	`.false.`	Requires `debug=.true.` too
`forward`	Controls the direction (forward or backward) of trajectories	`.true.` or `.false.`	`.true.`	Set to `.false.` for computing backward trajectories
`delt`	Time step – interval between output trajectory points – in seconds	Integer	3600 (1 hour)	

nleng	Path length of output trajectory in hours		144 (6 days)	
nslice	Time slices per input CMP file	Integer	1	For this version use only one time slice per CMP file (nslice > 1 has not been tested)
ntime	Time interval between slices of input CMP data as a mutiple of delt	Integer	6 (6 x delt = 6 hours)	
nlev	Number of levels per input time slice. For the computation of 2D trajectories specify nlev=1 and traj2D=.true. Note: for 3D trajectories it is recommended to have nlev ≥ 4. The cases nlev=2,3 are padded to 4 levels but the results may not be optimum.	Integer	0	Please specify the actual number of levels e.g. nlev=7
levs	A list of level values in the same order as in the input CMP data. These are expected to be in Pa; see also lscale	Integer	0	Please specify the levels e.g. levs=100000,92500,85000,70000, The level values are in Pa; if they are in hPa also set lscale = 100
lscale	Scales the level values (levs) by lscale e.g. hPa to Pa, lscale = 100	Real	1 (levels in Pa)	For levels in hPa set lscale = 100 e.g. levs = 700,850,925,1000, lscale = 100

headcheck	Checks that the level value in the CMP header matches the expected level as defined by the parameters nlev, levs	.true. or .false.	.true.	Set to .false. with traj2d=.true. or if the input CMP data has non-standard headers
datadir	Data directory for input CMP files	Character	None	e.g. datadir='../experiments/ncep/winds'
filefile	Text file containing a list of input CMP filenames that reside in the directory specified by datadir The order of these filenames must be consistent with forward i.e. for backward trajectories (forward=.false.) the CMP files should be listed in *reverse* time order. The input CMP files contain the wind fields (U,V,W) organised by levels. Note: it is assumed that U and V are in m/s and W is in Pa/s. If traj2d=.false.	Character*80	None	e.g. filefile='filelist_winds' containing: 1996042000.cmp 1996041918.cmp 1996041912.cmp … 1996041412.cmp 1996041406.cmp 1996041400.cmp These are located in the directory specified by datadir='../experiments/ncep/winds'

	then the W field is not used (or may be omitted from the input file).			

`locfile`	Text file containing a list of trajectory start (forward) or end (backward) locations. Comments begin with a `#`; blank lines are permitted.
	There is one location per line comprising: longitude in degrees (real), latitude in degrees (real), level (default is in Pa) (integer) and label (up to 80 characters).
	Note that if the level may be given in hPa if `hlscale = 100`
		Character*80	None	e.g. `locfile='melbourne_locs'`
				containing:
				`#`
				`# Initial trajectory points for low-tropospheric air over Melbourne`
				`#`
				`10.0 -45.0 100000 Pacific1`
				`10.0 -45.0 92500 Pacific1`
				`10.0 -45.0 85000 Pacific1`
				`10.0 -45.0 70000 Pacific1`
				
				`145.0 -37.7 100000 Melbourne`
				`145.0 -37.7 92500 Melbourne`
				`145.0 -37.7 85000 Melbourne`
				`145.0 -37.7 70000 Melbourne`

`hlscale`	Scales the level values in the location file by lscale e.g. hPa to Pa, hlscale = 100	Real	1 (levels in Pa)	For location levels in hPa seth lscale = 100 e.g. 10.0 -45.0 925 Pacific1
`outfile`	Output NetCDF file containing the trajectory data for each input location in locfile	Character*80		e.g. outfile = ‘study1.nc’
`ncdate`	Adds a date variable	`.true.` or	`.false.`	At this stage the date variable is only

	(ddatet) to the output NetCDF file (see outfile)	.false.		defined for delt ≥ 3600 (1 hour)
usehdate	The start or end date (tsdate) is read from the first CMP header of the first file from filefile using the Fortran format given by dfmt	.true. or .false.	.false.	If ncdate=.true. and usehdate=.false. then tsdate needs to be set
dfmt	Fortran format for reading tsdate from CMP header (see usehdate)	Character*80	'(40x,I4,2I2,1x,I4)'	This reads yyyy,mm,dd,hhhh == YYYYMMDDHHHH (all integers) Currently HHHH is effectively HH00 e.g. 199604201800 Requires usehdate=.true.
tsdate	Specified start or end date of trajectories in YYYYMMDDHHHH (or YYYYMMDDHH)	Character*12	None	e.g. tsdate = '199604201800' (or '1996042018') Requires ncdate=.true. and usehdate=.false.
gauss	Intended for a 2D surface wind field e.g. 10 m winds	.true. or .false.	.false.	Use with traj2d=.true.
levlong1	Output NetCDF 'long name' for 2D surface wind field	Character*80	None	Use with traj2d=.true. e.g. levlong1='10 m winds'
levunit1	Output NetCDF units for 2D surface wind field	Character*80	None	Use with traj2d=.true. e.g. levunit1= '10 m'

3.1 Auxiliary 2D variables

In addition to computing trajectories based on the wind fields (U,V,W) it is possible to find the value of an *auxiliary* field. Generally such variables are characteristic of the environment that the trajectory traverses.
This is accomplished by specifying the parameter `ntr2` > 0. These fields are either truly 2D e.g. MSLP, or are extracted levels from a 3D variable that may be regarded as 2D for input purposes. There should be a one-to-one correspondence in time of these files with the input CMP files i.e. the entries in the files `filefile` and `filefile2` should be paired. Table 2 describes the additional parameters required for processing the 2D auxiliary variables.

Table 2 Namelist parameters for auxiliary 2D variables

Parameter	Description	Value	Default	Comments
`ntr2`	Number of 2D auxiliary variables to be added to output NetCDF file (see outfile). These are usually 2D variables e.g. MSLP.	Integer	0	Set `ntr2 > 0` to enable output of auxiliary 2D variables
`datadir2`	Data directory for auxiliary CMP files.	Character	None	e.g. `datadir2='../experiments/ncep/aux_2D'`

`filefile2`	Text file containing a list of auxiliary 2D CMP filenames that reside in the directory specified by datadir2.
	There should be a one-to-one correspondence of these files with the input CMP files i.e. the entries in `filefile` and `filefile2` should be paired.
	Each CMP file contains a set of 2D data slices.
	The order of these filenames must be consistent with forward i.e. for backward trajectories (`forward=.false.`) the CMP files should be
		Character*80	None	e.g. `filefile2 = 'filelist_tr2d'` containing:
				`pmsl.1996042000.cmp`
				…
				`pmsl.1996041400.cmp`

	listed in *reverse* time order.			
`tr2name`	Output NetCDF name(s) for auxiliary 2D variable(s)	Character*80	None	e.g. `tr2name = 'MSLP'`
`tr2long`	Output NetCDF 'long name(s)' for auxiliary 2D variable(s)	Character*80	None	e.g. `tr2long = 'Mean sea level pressure'`
tr2unit	Output NetCDF units for auxiliary 2D variable(s)	Character*80	None	e.g. `tr2unit = 'hPa'`

3.2 Auxiliary 3D variables

The inclusion of 2D auxiliary variables may be extended to the 3D situation. In some contexts such a variable may be referred to as a 'tracer' if it is a quantity being carried along with the air parcel. More generally such variables are characteristic of the environment that the trajectory traverses.
In this case each CMP file contains a set of 3D data slices with the same level structure as the input CMP files i.e. each 3D auxiliary variable has the same levels as the winds fields. This is accomplished by specifying the parameter `ntr3` > 0. There should be a one-to-one correspondence in time of these files with the input CMP files i.e. the entries in the files `filefile` and `filefile3` should be paired. Table 3 describes the additional parameters required for processing the 3D auxiliary variables.

Table 3 Namelist parameters for auxiliary 3D variables

Parameter	Description	Value	Default	Comments
`ntr3`	Number of 3D auxiliary variables to be added to output NetCDF file (see outfile). These should have the same levels as the input CMP data.	Integer	0	Set `ntr3` > 0 to enable output of auxiliary 3D variables
`datadir3`	Data directory for auxiliary CMP files.	Character	None	e.g. `datadir3='../experiments/ncep/aux_3D'`

`filefile3`	Text file containing a list of auxiliary 3D CMP filenames that reside in the directory specified by datadir3.
	There should be a one-to-one correspondence of these files with the input CMP files i.e. the entries in `filefile` and `filefile3` should be paired.
	Each CMP file contains a set of 3D data slices with the same level structure as the input CMP files.
	The order of these filenames must be consistent with forward i.e. for backward
		Character*80	None	e.g. `filefile3 = 'filelist_tr3d'` containing:
				`1996042000.cmp`
				…
				`1996041400.cmp`
				Each file has the variables T and Q with the same level structure as the input CMP files specified by:
				`datadir='../experiments/ncep/winds'`

	trajectories (`forward=.false.`) the CMP files should be listed in *reverse* time order.			
`tr3name`	Output NetCDF name(s) for auxiliary 3D variable(s)	Character*80	None	e.g. `tr3name = 'T', 'Q'`
`tr3long`	Output NetCDF 'long name(s)' for auxiliary 3D variable(s)	Character*80	None	e.g. `tr3long = 'Temperature', 'Water vapour mixing ratio'`
`tr3unit`	Output NetCDF units for auxiliary 3D variable(s)	Character*80	None	e.g. `tr3unit = 'Kelvin', 'g/kg'`

4. Some examples

4.1 Multiple levels – 3D

The usual application of the program is for the computation of 3D trajectories. For optimum results it is recommended that a *minimum* of four levels be used. If possible use the order of 10 levels as is common with many reanalysis data sets. In this example we have wind fields (U,V,W) from NCEP reanalysis data on seven levels. There are more levels available but we wish to use water vapour mixing ratio as an auxiliary variable and this is only defined up to 200 hPa (see Section 4.3).
We want to compute 6-day backward trajectories for two locations with the trajectories arriving at all seven levels (see: melbourne_locs8).
The namelist file (input_backward_ncep.8.nml) is:

```
$traj3d
    headcheck= .true.
    ncdate= .true.
    usehdate = .true.
    forward  = .false.
    delt     = 3600
    nleng    = 144
    locfile  = 'melbourne_locs8'
    filefile = 'filelist_winds'
    outfile  = 'test_ncep.8.nc'
    datadir  = '../experiments/ncep/lev7'
    nslice   = 1
    ntime    = 6
    nlev=7,
    levs= 100000,92500,85000,70000,50000,30000,20000
$ END
```

The input CMP wind data files are located in ../experiments/ncep/lev7 and the filenames are listed in filelist_winds; there is one file per time step.
To compute the trajectories:

```
traj3d  input_backward_ncep.8.nml
```

The trajectories are written to the NetCDF file test_ncep.8.nc.
We may plot all of the trajectories by converting the NetCDF file to a MPL file (tnc2mpl) and then plotting with kmapline:

```
tnc2mpl  test_ncep.8.nc      (this creates test_ncep.8.mpl)
kmapline -n -0.010,60 -s "1,1,1, 15,60,3, 5,5,1, 0,0,0" -S
test_ncep.8.mpl
```

The NCAR Graphics meta file (gmeta) from kmapline may be converted to Postscript or PNG:

```
g2ps gmeta
gv g.ps
convert -trim -density 120 g.ps g,png
display g.png
```

Note: `convert` and `display` are (free) ImageMagick tools that are available on most Linux/UNIX platforms. Figure 1 shows the plot of the trajectories.

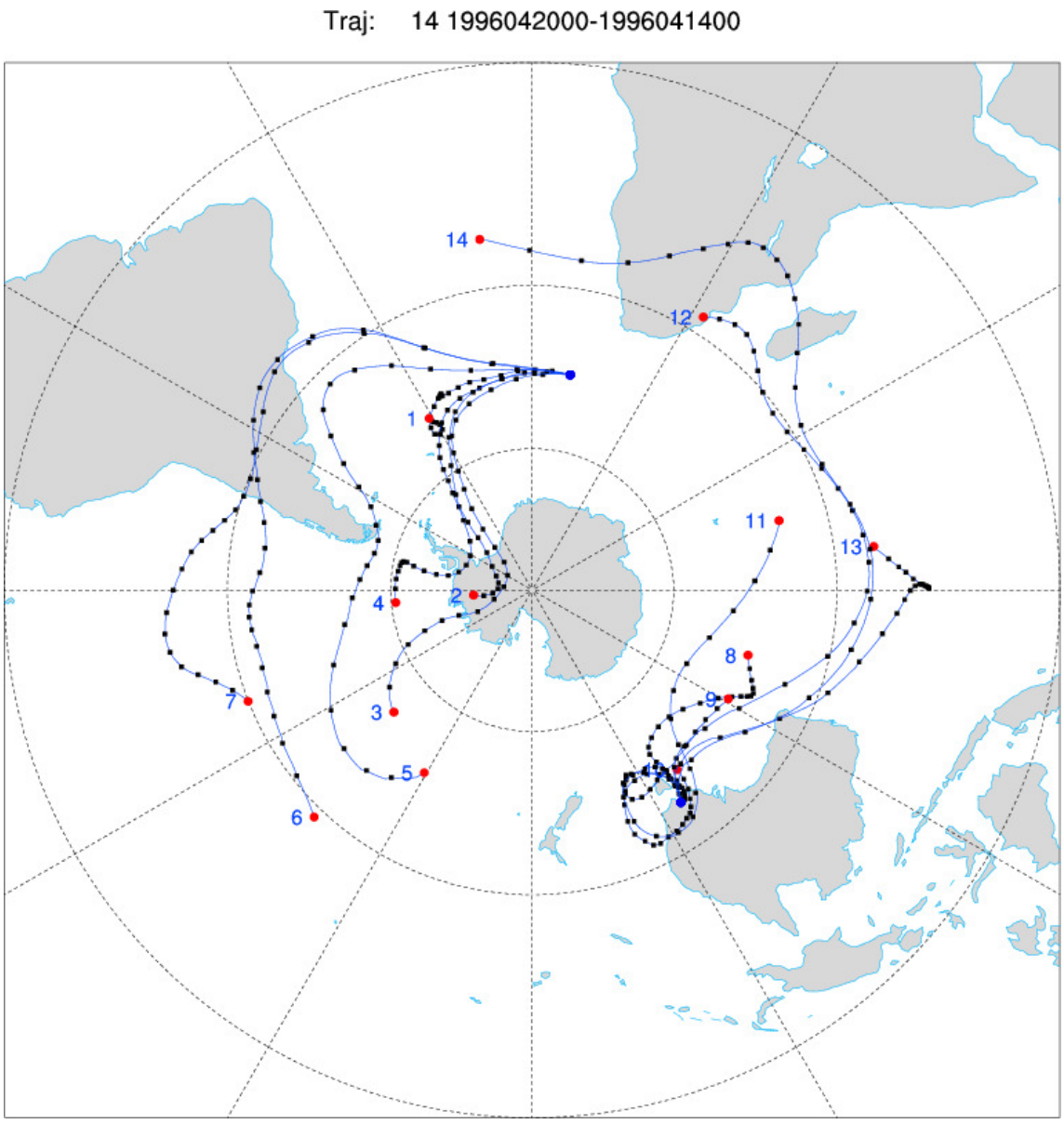


Figure 1: Trajectories from the example in section 4.1.

4.2 Single level – 2D

The computation of 2D trajectories may be accomplished using the parameter `traj2d=.true.` in conjunction with multi-level data. In this case if a number of locations at different levels are specified then the trajectories will remain at those levels. These input locations must be on levels that exist in the data e.g. if the data contains levels 1000, 925, 850, 700 hPa then a location like: 120.0 65.0 92500 Test1 is valid but: 120.0 65.0 97500 Test1 is not. Alternatively the input data may consist of a single level (`nlev=1`) e.g. 500 hPa, 10 m. In the case of surface winds the descriptive parameters `levlong1` and `levunit1` may be specified.

4.2.1 Multiple levels run in 2D mode

This is the same as the example in Section 4.1 except we set:

```
traj2d= .true.
```

The output NetCDF file will show the level remaining constant for each trajectory e.g. if the location is given as 925 hPa then the (2D) trajectory will be confined to this level. This is equivalent to setting `nlev=1` and specifying a single level e.g. 925 hPa and running the program on data containing that single level. Figure 2 shows these 2D trajectories for comparison with the example in Section 4.1.

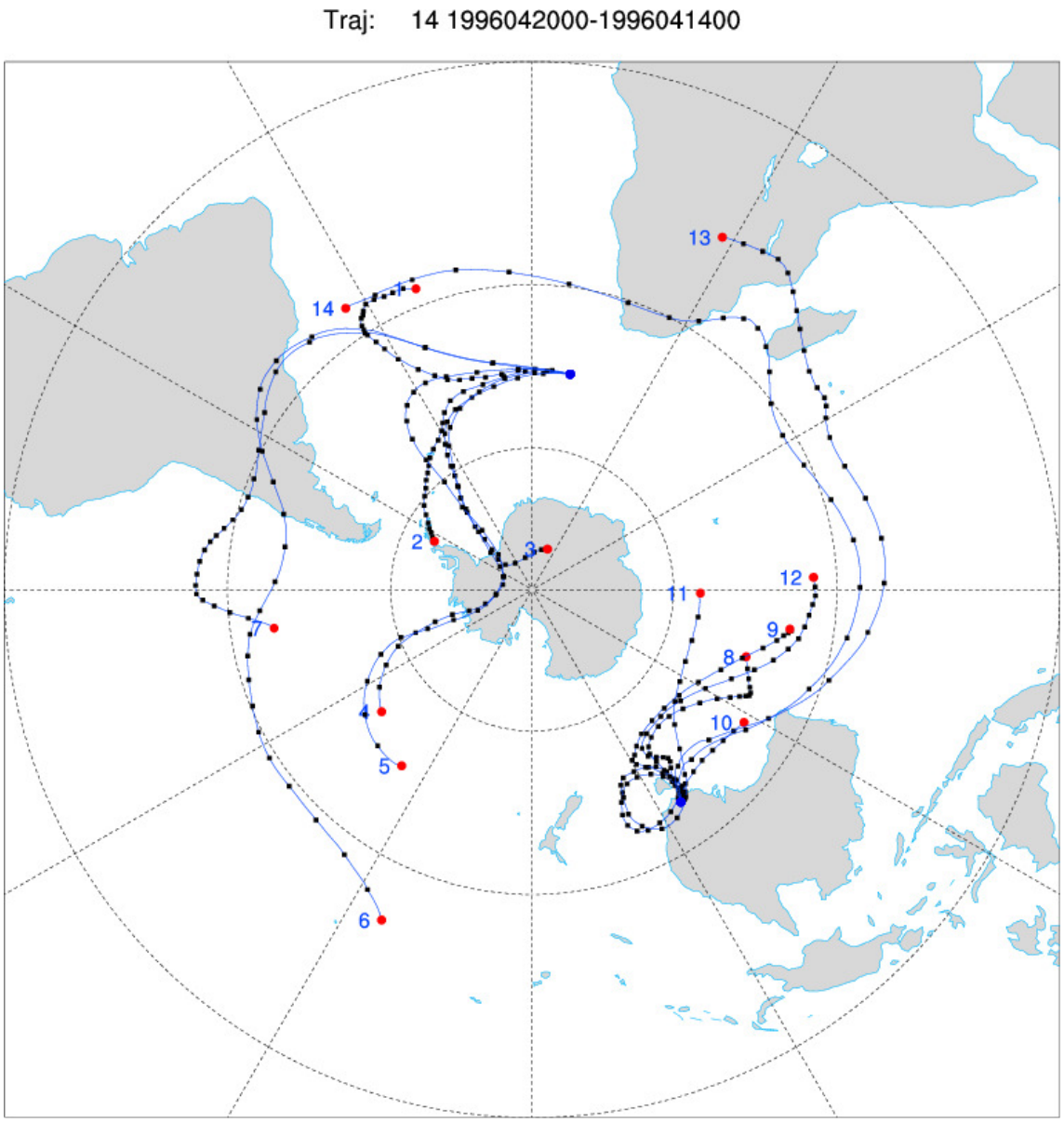


Figure 2: Trajectories from the example in Section 4.2.1.

4.2.2 10 m surface winds – 2D

In this example we wish to compute 4-day backward trajectories for 10 m surface winds which happen to be on a Gaussian (irregular) latitude) grid. The namelist file (input_backward_ncep.0.nml) is:

```
$traj3d
    traj2d = .true.,
    headcheck= .false.
    ncdate= .true.
    usehdate = .false.
    tsdate= '1997060700'
    gauss= .true.
    levlong1= '10 m winds'
    levunit1= '10 m'
    forward  = .false.
    delt     = 3600
    nleng    = 96
    locfile  = 'melbourne_locs.2d'
    filefile = 'filelist_2d'
    outfile  = 'test_2d.nc'
    datadir  = './'
    nslice   = 1
    ntime    = 6
    nlev=1,
    levs= 99999
$end
```

Note that we set `traj2D=.true.` and `gauss=.true.` (due to the Gaussian grid).
The CMP headers in this particular case are non-standard so we turn off the header check i.e. `headcheck=.false.` Futhermore we set the NetCDF level variable (`lev`) to have the correct attributes i.e. we set `levlong1` and `levunit1`.
In the location file (melbourne_locs.2d) we have entries like:

```
    145.0 -38.3  99999 Melbourne
```

i.e. we are using level 99999. We can use any level as long as it is consistent between the namelist and location files. We are also accessing CMP data files in the current directory.
After running the program:

```
    traj3d input_backward_ncep.0.nml
```

we obtain a NetCDF file which has the 2D trajectories at a level of 99999, which corresponds to 10 m. The trajectories are shown below in Figure 3.
Note: In this example:

```
    kmapline -n -0.010,60 -s "1,1,1, 15,60,3, 5,5,1, 0,0,0"
test_ncep.0.mpl < imap.sh25
```

where:

File: imap.sh25

```
ST
-90,0
n
-25,270
-25,90
```

```
-25,180
-25,0
n
y
c
y
30
y
```

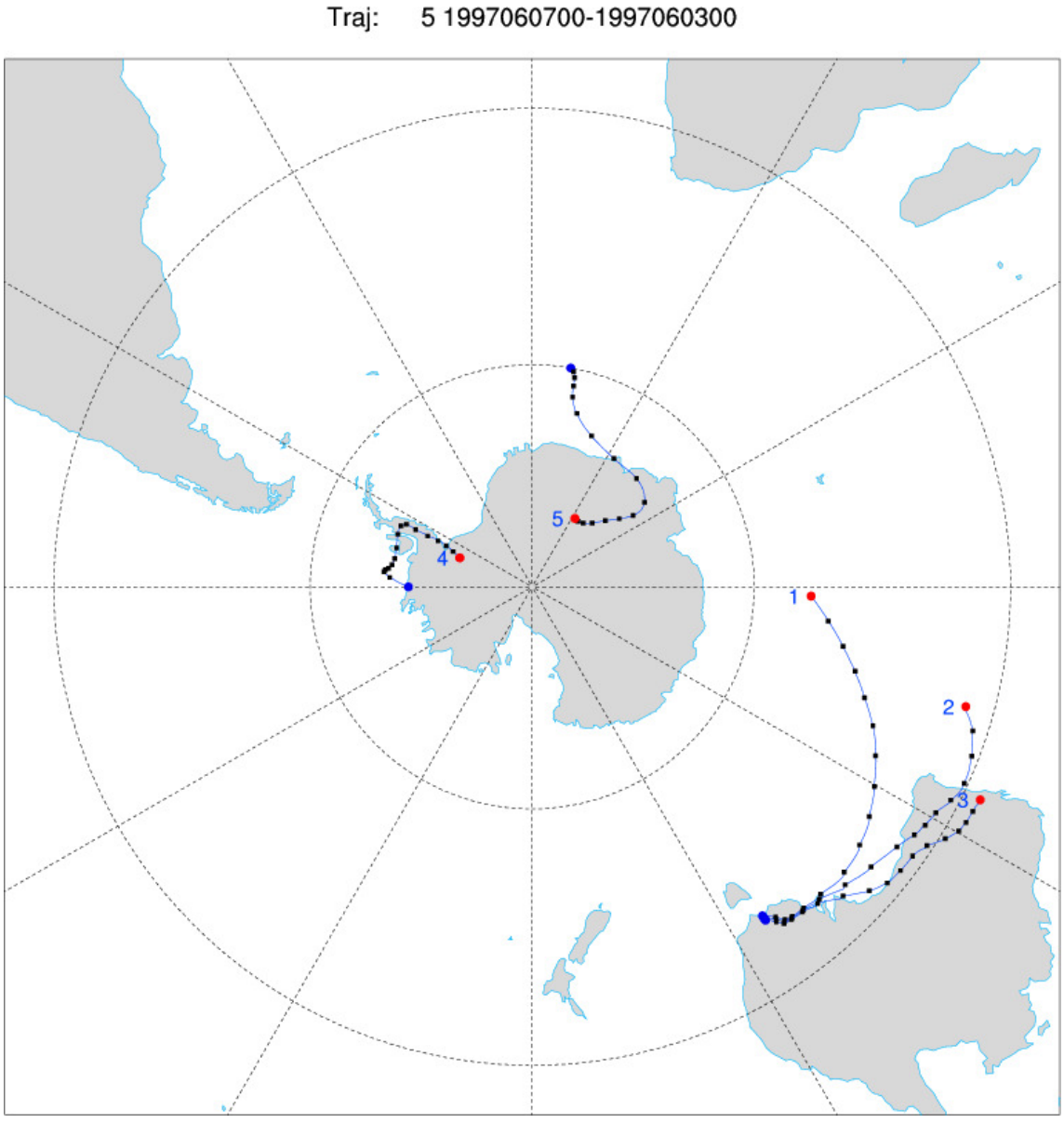


Figure 3: Trajectories from the example in Section 4.2.2.

4.3 Multiple levels with auxiliary variables

This is essentially a variation on the example in Section 4.1. In this case we are 'tacking on' some auxiliary variables that are interpolated to the trajectory locations at each time step. In some situations these variables may be referred to as

'tracers' but generally they are characteristic of the environment that the trajectory traverses. For illustration we will add one 2D field and two 3D fields.
First we make one change to the namelist file from the example in 4.1:

```
outfile = 'test_ncep.9.nc'
```

and save it as input_backward_ncep.9.nml.
The new namelist file (input_backward_ncep.9.nml) contains two extra sections. The first one is for 2D auxiliary fields:

```
datadir2  = '../experiments/ncep/lev7/aux2'
filefile2 = 'filelist_aux2'
ntr2 = 1
tr2name = 'MSL'
tr2long = 'Mean sea level pressure'
tr2unit = 'hPa'
```

There is one 2D field (ntr2=1): mean sea level pressure.
The second is for 3D auxiliary fields:

```
datadir3  = '../experiments/ncep/lev7/aux3'
filefile3 = 'filelist_aux3'
ntr3 = 2
tr3name = 'T', 'ABSV'
tr3long = 'Temperature', 'Absolute vorticity'
tr3unit = 'Kelvin', '/s'
```

There are two 3D fields: temperature and water vapour mixing ratio.
After running the program:

```
traj3d input_backward_ncep.9.nml
```

we see that the output NetCDF file (test_ncep.9.nc) will contain the same trajectory variables as before (test_ncep.8.nc) but with additional variables (T,ABSV) appended. The plot of the trajectories will be identical to that shown in Section 4.1 (Figure 1).

5. Preliminaries

This section provides information on compiling traj3d and aspects of the data formats, including conversion from NetCDF to CMP.

5.1 Compilation of traj3d

Follow these steps to install `traj3d`:

1. Firstly, you will need to contact Ian Simmonds (simmonds@unimelb.edu.au) for a username and password.
2. Download the software package: http://www.earthsci.unimelb.edu.au/~kevin/traj3d_1.0/ztraj3d.V1.0.zip (about 1.5 MB)
3. Optionally download the examples and data:

http://www.earthsci.unimelb.edu.au/~kevin/traj3d_1.0/ztraj3d.V1.0_examples+data.zip
(about 34 MB)

4. `unzip ztraj3d.V1.0.zip` into a convenient directory (e.g. traj3d.dir). See: README.package and the README.* files throughout the package. The folder bin contains precompiled Linux binaries that should work on many Linux operating systems. Otherwise you will need to compile the programs.
5. The compilation of `traj3d` and the utilities is achieved via the C-shell script `run-make` in folder src. On a typical Linux OS: `run-make`
This script runs a set of Makefiles (one per program) based on the Fortran 77 compiler `g77` (usually duplicated as `f77`). If you use another Fortran compiler you may need to change some of the compiler options (see the `FFLAGS` variable in a Makefile). Note that NetCDF libraries need to be available on your OS. For the utility `kmapline` you will need NCAR Graphics or NCL. Finally, the binaries in the folder bin may be copied to an appropriate location on your network i.e. somewhere in your path.
6. Optionally:
`unzip ztraj3d.V1.0_examples+data.zip` into the same directory specified in step 4 (e.g. traj3d.dir). Look at the README* files in the folder examples and its subdirectories for information about the examples.

5.2 Output trajectory file (NetCDF)

The trajectories are output to a NetCDF file. This has the advantage of being compatible with many common software packages e.g. Matlab, NCL.

5.3 Plotting of trajectories – NCAR Graphics/NCL

There are potentially many methods of plotting the trajectories in various software packages e.g. NCL, Matlab. We will outline two approaches, both are connected with NCAR Graphics/NCL.

5.3.1 tnc2mpl/kmapline

Our in-house method is to convert the NetCDF file to a MPL file using `tnc2mpl` (Section 6.3) and then plot this latter file with `kmapline` (Section 6.5), a program based on NCAR Graphics which is now part of NCL. See the main examples in Section 4.1 with additional material in Section 6.5.

5.3.2 NCL

Another approach is to use NCL which is a toolkit based on NCAR Graphics. There are probably suitable scripts on the NCL web site that could be modified for this purpose. See Section 6,7 for details about downloading NCL.

5.4 Conversion of NetCDF to CMP format

A key step in computing trajectories is to convert your data to CMP format (Section 5.6). This is a simple binary format that was originally designed to work with NCAR Graphics. It is also used with the University of Melbourne cyclone

tracking software and occasionally used by CSIRO in Australia where it is referred as CIF.
Note: if your data is in GRIB format it may be convenient to use a free tool like Xconv to convert from GRIB to NetCDF first, then convert to CMP – see Section 5.5.
The conversion from NetCDF to CMP is performed by `read_nc2cmp` – see Section 6.1. Essentially you need to specify the dimensions and variables names as well as the desired level indices as arguments to the program. This information is obtained via a NetCDF file header dump using the NetCDF utility `ncdump`
e.g. `ncdump -h test.nc`
If the dump shows that the level variable is called (say) p then:

```
ncdump -v p
```

will print out the level values e.g. p = 1000, 925, 850, 700, 600, 500
The following script (C-shell) using NCEP Reanalysis 2.5 x 2.5 degree 6 hourly data is an example of the conversion process.

```
#!/bin/csh -f
read_nc2cmp -i 199604.winds.ncep.nc -d "longitude,latitude,t" -l p
-L "1,2,3,4"  -u "U,V,VVEL" -v "U,V,W" -U "'m/s','m/s','Pa/s'" -G
-S -t "1996041400,1996042000" \
 -p ../udunits.dat
exit
```

We are extracting U,V,W (NetCDF names U,V,VVEL) for levels 1000,925,850,700 (indices 1,2,3,4 of the NetCDF level variable p) and the time range 1996041400 to 1996042000. Each time step (this is 6 hourly data) is output as a separate CMP file i..e. 1996041400.cmp, 1996041406.cmp, …, 1996042000.cmp. The –p option refers to a local copy of the UDUNITS data file udunits.dat if it is not accessible via a normal network path.
The file 199604.winds.ncep.nc was in fact a multi-level GRIB file converted to NetCDF with grb2nc (based on `convsh`, the shell version of Xconv):

```
grb2nc -i 199604.grb -o 199604.winds.ncep.nc
```

5.5 Conversion of GRIB to CMP conversion

If you have GRIB data you may be able to use the freely available utility Xconv (`xconv`)to convert your GRIB file to a NetCDF file – see Section 6.1. Then you can convert this NetCDF file to CMP format with `read_nc2cmp` – see Section 5.4.

5.6 CMP format description

For much of our in-house work at the University of Melbourne it is convenient to take a complex binary file e.g. NCEP Reanalysis NetCDF, and convert it o a simple binary format which we call *conmap* (CMP) format (also known as CIF at CSIRO in Australia). This is capable of representing a variable e.g. pressure, on a longitude-latitude grid. A fragment of code to read a CMP file) given below. The lines in **bold** represent a block of information for a given map (i.e. grid or field). A set of maps consists of a concatenation of such blocks.

```
implicit none
 real xmiss
```

```
parameter (xmiss= 99999.9)    ! conmap (CMP) missing value code
integer num
parameter(num= 200)           ! Max. size of lat.-lon. grid
real xlats(num),xlons(num)    ! Arrays for lats and lons
real dat(num,num)             ! Array for input data
character*80 head1            ! Header (description)
character*80 infile           ! CMP file name
integer i,j,nlats,nlons
open(1,file=infile,form='unformatted')
read(1)nlats
read(1)(xlats(i),i=1,nlats)
read(1)nlons
read(1)(xlons(i),i=1,nlons)
read(1)head1
read(1)((dat(i,j),i=1,nlons),j=1,nlats)
close(1)
write(*,'('' No. lats, no. lons: '',2I6)')nlats,nlons
write(*,'(1X,A)')head1
```

We use a `parameter` statement to set the maximum size of our longitude and latitude arrays to 200 points (`num`). Thus our two-dimensional data array (matrix) named `dat` is a grid of maximum size 200 x 200 points
The integers `nlons` and `nlats` hold the actual *number* of longitudes and latitudes e.g. nlons = 144, nlats = 73
The array `xlons` holds the actual longitude *values* e.g. 0.0, 2.5, … and the array `xlats` holds the actual latitude values from *south to north* e.g. –90.0, -87.5, …
The array `xlats` is read using an implied DO loop:

```
read(1)(xlats(i),i=1,nlats)
```
is equivalent to
```
read(1)xlats(1),xlats(2),…,xlats(nlats)
```

The array xlons is read in a similar way. There is an 80 character text header (`head1`) containing some information about the data.
Finally we read in the two-dimensional array `dat` using a pair of 'nested' implied DO loops:

```
read(1)((dat(i,j),i=1,nlons),j=1,nlats)
```

where i is longitude index and j is latitude index.
This works from the 'outside' to the 'inside' loop. We start with `j = 1` and read the inner loop `(dat(i,1),i=1,nlons)`. Then `j = 2` and we read `(dat(i,2),i=1,nlons)` and so on.
For each latitude index j we read the values of `dat` at each longitude index i i.e. we read the data on *latitude circles* from south to north. If we specified each array element individually we would need:

```
read(1)dat(1,1),dat(2,1),…,dat(nlons,1),
dat(1,2),dat(2,2),…
dat(nlons,2),…,dat(1,nlats),dat(2,nlats),…dat(nlons,nlats)
```

To write out a CMP file we just *replace* read by write in the above code.

The CMP format has a code to represent *missing* or *undefined* data (99999.9) as set by `xmiss`. This is useful to process files with missing data since we want to exclude these from our summations etc.
If we expect missing values then we should check for them.
This piece of code computes the average of *non-missing* ('live') values at each gridpoint:

```
do j=1,nlat
  do i=1,nlon
    if (dat(i,j).eq.xmiss)then
      nmiss= nmiss + 1  ! Count of missing values (for
                        ! information only)
    else
      nsum(i,j)= nsum(i,j) +1  ! Count of 'live' values at this
                               ! gridpoint
      sum(i,j)= sum(i,j) + dat(i,j) ! Sum of 'live' values
    endif
  enddo
enddo

do j=1,nlat
  do i=1,nlon
    ave(i,j)= sum(i,j)/nsum(i,j) ! Average at the gridpoint (i,j)
  enddo
enddo
```

5.7 MPL format

The MPL (binary) format comprises an 80 character header, the number of time steps in a trajectory and then the latitude and longitude data. This code fragment summarises the MPL format:

```
implicit none
integer mtraj,mt
parameter (mtraj=100,mt=500)
real tlats(mtraj,mt)
character*80 head
integer np,it,i
it= 4  ! e.g. trajectory number 4
read(4)head
read(4)np
read(4)(tlats(it,i),i=1,np)
read(4)(tlons(it,i),i=1,np)
```

A set of trajectories are represented by concatenating blocks of information corresponding to the lines highlighted in **bold**.

6. Utilities

This is a resource of useful utilities that may assist to converting the input data e.g. NetCDF, to CMP or for generating the filename containing a list of CMP files used by the `filefile` parameter `in traj3d`.

6.1 read_nc2cmp: Converts NetCDF to CMP

The program `read_nc2cmp` will read 'usual' NetCDF files that have a variable var (time, level, lat, lon)

in this order as indicated by the output of `ncdump -h`.
Note that the dimensions and variables may have different names
e.g. longitude instead of lon.

To compile:

```
g77 -o read_nc2cmp{,.f} -lnetcdf -ludunits
```

i.e. you need the NetCDF and UDUNITS libraries.
You also need the include files netcdf.inc and udunits.inc

If you type the name of the program you get a brief usage message:

```
read_nc2cmp
```

read_nc2cmp: Version 2.4 (Apr 14 2009)

```
 Usage: read_nc2cmp [--help][-D idbg][-i ncfile][-o cmpfile][-d
"lon,lat,time"]
  [-u uservars][-U vunits][-v vtypes][-l levelvars][-L ilev]
  [-g gridtype][-r rtype][-s uscal][-p udunits]
  [-m no_maps][-M "map1,map2"][-t "date1,date2"][-T "ttype"][-G][-
S]

 Options:
 D: 0= None 1= Basic 2= Verbose 3= Print dimension arrays to file
fort.10
 G: Duplicate data at lon to 360 (default: no)
 S: Output one file per time (default: all in one)
 T: ttype: CDO, fcst
    fcst: Adds 6 hr to date-time; use with NCEP/NCEP2 10m winds
 Note: Max. sizes (characters) of text variables
 gridtype: 10 rtype: 5 vtypes: 8 units: 8

 --help: Gives some examples
```

The -d option is set for NCEP and NCEP2 by default, assuming that the longitude, latitude and time variables are named lon, lat and time.
You may need to give the location of the UDUNITS data file with the
-p option:

`-p '/usr/local/etc/udunits.dat'` or change the source code at line **172**:

```
udpath= '/usr/local/etc/udunits.dat'
```

This is only necessary if the file is not in the usual place.

-D is the debug option; 3 prints the dimensions to a file called fort.10.
Note: For ERA-40 geopotential, the program will divide by g= 9.807 m s**-2 to give geopotential height (m).

Example 1: NCEP mean sea level pressure (2D variable)

Consider the (edited) header dump of the NetCDF file slp.2004.nc i.e. `ncdump -h slp.2004.nc`:

```
netcdf slp.2004 {
dimensions:
```

```
   lon = 144 ;
   lat = 73 ;
   time = UNLIMITED ; // (1464 currently)
variables:
   float lat(lat) ;
         lat:units = "degrees_north" ;
         lat:actual_range = 90.f, -90.f ;
         lat:long_name = "Latitude" ;
   float lon(lon) ;
         lon:units = "degrees_east" ;
         lon:long_name = "Longitude" ;
         lon:actual_range = 0.f, 357.5f ;
   double time(time) ;
         time:units = "hours since 1-1-1 00:00:0.0" ;
         time:long_name = "Time" ;
         time:actual_range = 17557968., 17566746. ;
         time:delta_t = "0000-00-00 06:00:00" ;
   short slp(time, lat, lon) ;
         slp:long_name = "4xDaily Sea Level Pressure" ;
         slp:valid_range = 87000.f, 115000.f ;
         slp:actual_range = 92700.f, 111370.f ;
         slp:units = "Pascals" ;
         slp:add_offset = 119765.f ;
         slp:scale_factor = 1.f ;
         slp:missing_value = 32766s ;
         slp:precision = 0s ;
         slp:least_significant_digit = -1s ;
         slp:GRIB_id = 2s ;
         slp:GRIB_name = "PRMSL" ;
         slp:var_desc = "Sea Level Pressure\n",
}
```

To decode maps 5-8 of this mean sea level pressure file use the following command:

```
read_nc2cmp -i slp.2004.nc -o out.cmp -u slp -r NCEP -v PMSL -s
0.01 -M "5,8"
```

The pressure variable is named slp (-u option).
We need to scale the pressure in Pa to hPa i.e. apply a scaler of 0.01 (-s option).
The -r and -v options are for setting the CMP header for the cyclone tracking scheme.
The -M option gives the map range to be decoded i.e. maps 5-8.

The screen output is:

```
NOTE: User scaler:   0.00999999978
 Output map range:  5 -  8
 NetCDF file opened successfully (ncid= 3)

 Inquiring about variables ...
 Reading longitudes ...
 Reading latitudes ...
 Reading times ...
 Reading attributes ...
 No. of maps to be extracted:  4
 Reading user variable ...
```

```
     5:PMSL                              NCEP      20040102 0000    MB
2.5x2.5DEG
     6:PMSL                              NCEP      20040102 0600    MB
2.5x2.5DEG
     7:PMSL                              NCEP      20040102 1200    MB
2.5x2.5DEG
     8:PMSL                              NCEP      20040102 1800    MB
2.5x2.5DEG
 NetCDF file closed successfully (ncid= 3)
 Output conmap file: out.cmp
 Finished!
```

The file out.cmp contains the four decoded maps.

Example 2: Extraction of a single level from multi-level ERA-40 geopotential height

The (edited) output from ncdump for hgt.200208.nc in folder ncdata is:

```
netcdf hgt.200208 {
dimensions:
   longitude = 144 ;
   latitude = 73 ;
   levelist = 23 ;
   time = UNLIMITED ; // (62 currently)
variables:
   short z(time, levelist, latitude, longitude) ;
         z:scale_factor = 7.53787087081502 ;
         z:add_offset = 241940.397676004 ;
         z:_FillValue = -32767s ;
         z:missing_value = -32767s ;
         z:units = "m**2 s**-2" ;
         z:long_name = "Geopotential" ;
}
```

There are 23 levels. If you use `ncdump -v levelist` then you can see the levels:

```
 levelist = 1, 2, 3, 5, 7, 10, 20, 30, 50, 70, 100, 150, 200, 250,
300, 400, 500, 600, 700, 775, 850, 925, 1000 ;
```

If you want the 500 hPa level then you require `levelist(17)`.

Hence:

```
read_nc2cmp -i ncdata/hgt.200208.nc -o j.cmp -d
"longitude,latitude,time" -u z -D 3 -r ERA40 -l levelist -L 17
```

will decode the 500 hPa geopotential data (-r ERA40 will cause it to be divided by g).

Example 3: Extraction of a set of levels form a multi-level NCEP wind file

```
read_nc2cmp -i 199604.winds.ncep.nc -d "longitude,latitude,t" -l p
-L "1,2,3,4"  -u "U,V,VVEL" -v "U,V,W" -U "'m/s','m/s','Pa/s'" -G
-S -t "1996041400,1996042000" \
```

```
-p ../udunits.dat
```

The slash (\) is a line continuation character for the C-shell. The set of levels does not have to be consecutive e.g. one may require `-L "1,3,4,7,9"`.
See Section 5.2 for more details.

6.2 datetraj: Generates a sequence of time steps

This utility is useful for generating a sequence of time steps such as those of a computed trajectory or the input data files. Currently the program can generate time series for a time step of one hour or greater.

To compile: `g77 -o datetraj datetraj.f`

For usage type: `datetraj`

```
Usage: datetraj date0 tinc ndays
 date0: yyyymmddhh e.g 1997060100
Note: Use a -ve value of tinc for backward trajectories
Examples: datetraj 1997060100  6 4
          datetraj 2000020112 -6 5
The list of dates is written to: datetraj.2.txt
The backward flag (1= backward) and start/end dates
are written to: datetraj.3.txt
```

This example generates the input time steps of a backward 4-day trajectory from 6-hour data:

e.g. `datetraj 1996042000 -6 4`

prints on the screen:

```
1996042000  1996041918  1996041912  1996041906  1996041900
1996041818  1996041812  1996041806  1996041800  1996041718
1996041712  1996041706  1996041700  1996041618  1996041612
1996041606  1996041600
```

In addition the following files are created.
File: datatraj.2.txt

```
1 1996042000
2 1996041918
3 1996041912
4 1996041906
5 1996041900
…
13 1996041700
14 1996041618
15 1996041612
16 1996041606
17 1996041600
```

File: datetraj.3.txt

```
1 1996042000  1996041600
```

The screen output may be of use in scripts. For instance you can create a file containing the list of required data files (one per time step) using this C-shell (`csh`) script:

```
set ff = ` datetraj 1996042000 -6 4`  # Send screen output to $ff
@ nf = $#ff    # Number of items in $ff
rm filelist_test  # Ensure that filelist_test does not exist
@ i = 1
while ($i <= $nf)
  set f = $ff[$i]
  echo "$f.cmp" >> filelist_test  # Add each filename (based on date
$f) to list
  @ i ++   # Increment counter i by one
end
```

The file filelist_test contains:

```
1996042000.cmp
1996041918.cmp
…

1996041600.cmp
```

6.3 tnc2mpl: Converts output NetCDF trajectory file to MPL format

This reads the output NetCDF from `traj3d` and outputs selected trajectories in mapline (MPL) format for plotting by `kmapline`.

To compile: `make tnc2mpl`

For usage type: `tnc2mpl`

```
Usage: tnc2mpl [-{n,l,t} min[,max]] trajfile(s).nc
  Extract trajectory data from NetCDF file.
  -D Debug printout
  -n mini[,max]  select trajectory range
  -l mini[,max]  select length range
  -t mini[,max]  select time (unlimited) range
  -p             prints initial point for trajs
  -m Exclude missing values with code  -999
  -o outfile (default: Input file with nc->mpl)
If no range is given, all are converted.
Output is appropriate for kmapline.
```

In this example we print out the initial point of each trajectory:

```
tnc2mpl -p test_ncep.1.nc
```

Here we exclude missing values (these are inserted into the NetCDF file if the trajectory attempts to go outside a regional grid):

```
tnc2mpl -m test_ncep.1.nc
```

Note: The output MPL file will be called test_ncep.1.mpl.

Here time steps 140-145 of trajectories 5-10 will be output to the MPL file mytest.mpl:

```
tnc2mpl -l 140,145 -n 5,10 -o mytest.mpl test_ncep.1.nc
```

6.4 mapmanip: Extracts trajectories from a MPL file

This reads in a MPL file and outputs a selected trajectory to a new MPL file or creates a text listing of the trajectories.

To compile: `g77 -o mapmanip mapmanip.f`

```
Usage: mapmanip [-dpF] [-o outmaplinefile] maplinefile
 Options:
   d: Debug - see: fort.21
   o: Output mapline file [default: manip.mpl]
   n trajno: Output specified trajectory
   p: Print mapline file in text format to: manip.lis
   F: Use input filename as header
 Example: mapmanip -p 1999020100.map   (text output is to
manip.lis)
```

The main use of the utility is to produce a text listing of the trajectories in the MPL file e.g. `mapmanip -p test_ncep.1.mpl`
See manip.lis for the text listing of these trajectories.
Another use is to output a single trajectory to a new MPL file
e.g. `mapmanip -n 2 test_ncep.1.mpl` (output trajectory 2 to manip.mpl)
`mapmanip -n 2 -o traj2.mpl test_ncep.1.mpl` (output trajectory 2 to traj2.mpl)

6.5 kmapline: Reads a MPL file and plots with NCAR Graphics/NCL

This utility reads a MPL file output from `tnc2mpl` and plots the trajectories using NCAR Graphics/NCL. The options are intended for use with our cyclone tracking software but are largely applicable to the trajectory situation. This utility is not extensively documented and serves as a useful tool when no other is readily available. A more flexible approach would be to investigate NCL directly.

To compile: `make kmapline`

For usage type: `kmapline`

```
Usage: kmapline [a][A fcol][C lcol][H "uhdr"]
 [I][c ccol,cthk][f fmapfile][g greyshade][L ltyp,lthk]
 [n fontsize,fontcol][p][s "sym_params"][Y hdry][GSN] mplfile [<
imapfile]

 Options:
  a: Fill in land areas
 G: Global CE projection
 N: Northern ST projection
 S: Southern ST projection
 A: fcol [3]: Land fill colour
```

```
 C: lcol [1]: 1= Black 2= Light blue 3= Light grey
    4-67 : Dark blue -> Bright red
 c: ccol [3]: Coastline colour
    cthk [1]: Coastline thickness
 f: File with entries:
    mapfile,lcol,ltyp,lthk
 g: greyshade land fill 0-1.0 [0.85]
 H: uhdr: User header
 I: Plot interpolated track (default: no interpolation)
 L: ltyp [1]: 1= Solid 2= Dashed 3= Dotted 4= Dashed-dotted
    lthk [1]: 1= standard thickness
 n: fontsize [0.011] - label with track number
    fontcol [15] - label colour
    Note: Set fontsize < 0 to label at opposite end of track
 p: Indicate track points with dots
    This is a special version of -s
    i.e. -p == -s "1,1,1,1,1,1,5,5,5,1,1,1"
 s: Indicate track points with symbols
    sym_params:
"sym_first,sym_last,sym_other,col_first,col_last,col_other,
    siz_first,siz_last,siz_other,sol_first,sol_last,sol_other"
    i.e. 3 symbols, colours, sizes, solid indicators
    sym: 1-5 : 1 (circle) 2 (diamond) 3 (triangle) 4 (square) 5
(star)
    col: 1-67
    siz: >= 1
    sol: 0 (solid) 1 (hollow)
 Y: hdry: Vertical location of header 0-1.0 [0.986]
```

Examples:

```
  kmapline -N j.map
  kmapline -a -A 32 -C 55 -c 2,5 j.map < imap
  kmapline -a -C 55 -c 2,5 j.map < imap
  kmapline -a -g 0.95 -C 55 -c 2,5 -Y 0.75 -p j.map < imap
  kmapline -a -g 0.95 -C 1 -L 1,0 -c 2,5 -Y 0.75
  -s "1,1,4,10,60,30,8,8,5,0,0,0" j.map < imap
```

Here we have turned off the track lines (-L)

```
  kmapline -a -g 0.95 -C 1 -L 1,0 -c 2,5 -Y 0.75
  -s "1,1,4,10,60,30,8,0,0,0,0,0" -H "Test of genesis"
  -n 0.007,1 j.map < imap
```

Here we are plotting the first points only (cyclogenesis)

Available projections are:

- MO Mollweide type
- ST Stereographic
- OR Orthographic
- LC Lambert conformal
- LE Lambert equal area
- GN Gnomonic
- AE Azimuthal equidistant
- SV Satellite view
- CE Cylindrical equidistant
- ME Mercator

If you type: kmapline *mplfile* e.g. `kmapline test_ncep.1.mpl`
you will be prompted for the response to a set of (hopefully) intuitive questions.

In this example the contents of a MPL file are plotted over the southern hemisphere (see Figure 4).

```
kmapline -n 0.010,60 -s "1,1,1, 15,60,3, 5,5,1, 0,0,0" -S test_ncep.1.mpl
```

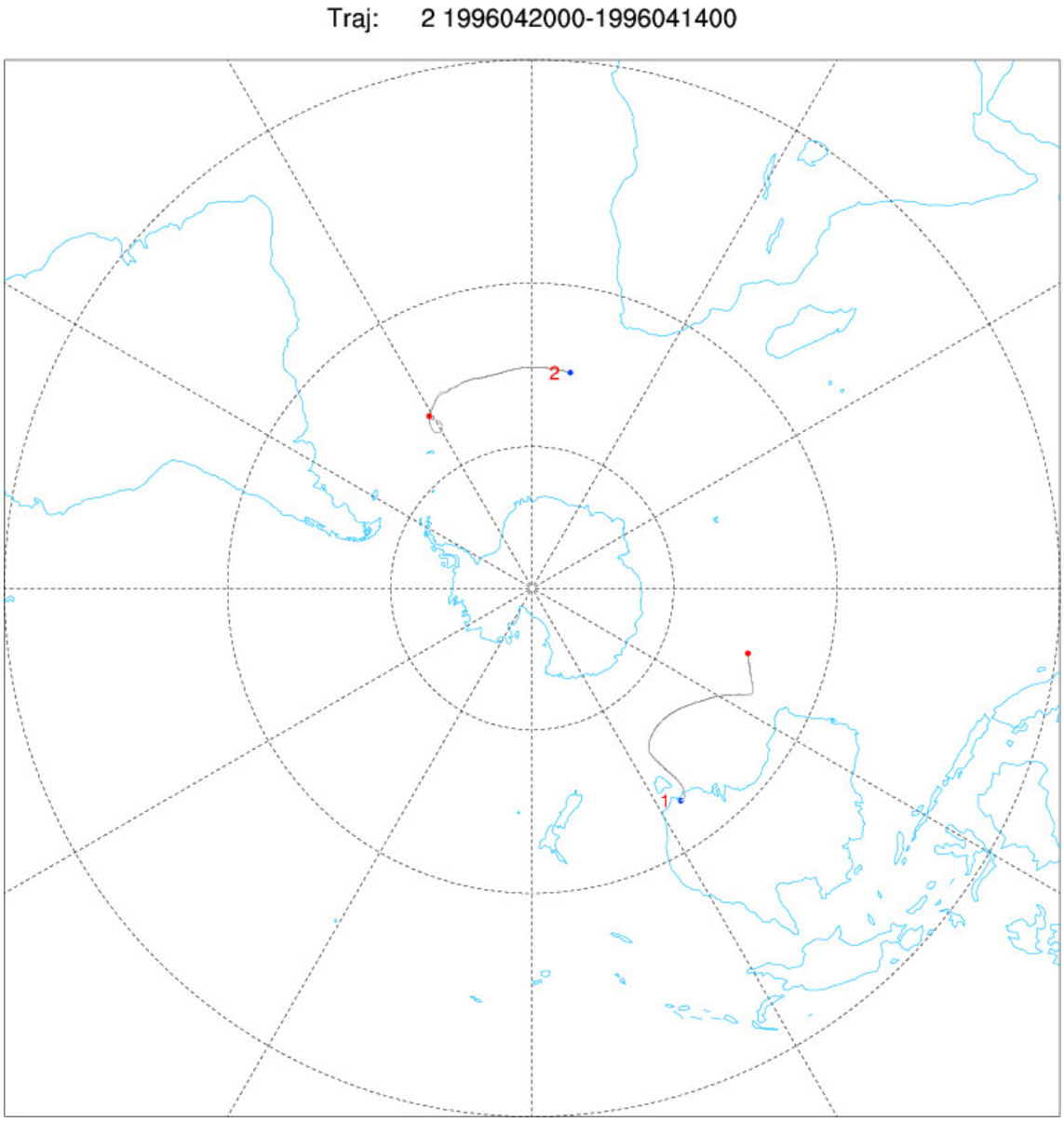


Figure 4: `kmapline` example – south polar projection (-S).

In this example the contents of a MPL file are plotted over the region south of Australia. The start points are in red and the end points in blue; trajectory numbers are in red. The region is defined in the file imap.reg1 along with the responses to some (keyboard) questions.

File: imap.reg1

```
CE                    ← Cylindrical equidistant
```

```
0,180
n
-80,80          ← Box 80 – 30 S, 80 – 130 E
-30,180
-80,80          ← Repeat box for CE
-30,180
n
y
c
y
10              ← Grid spacing 10 degree
y
```

To create the plot (see Figure 5):

```
kmapline -n 0.010,60 -s "1,1,1, 15,60,3, 5,5,1, 0,0,0" test_ncep.1.mpl <
imap.reg1
```

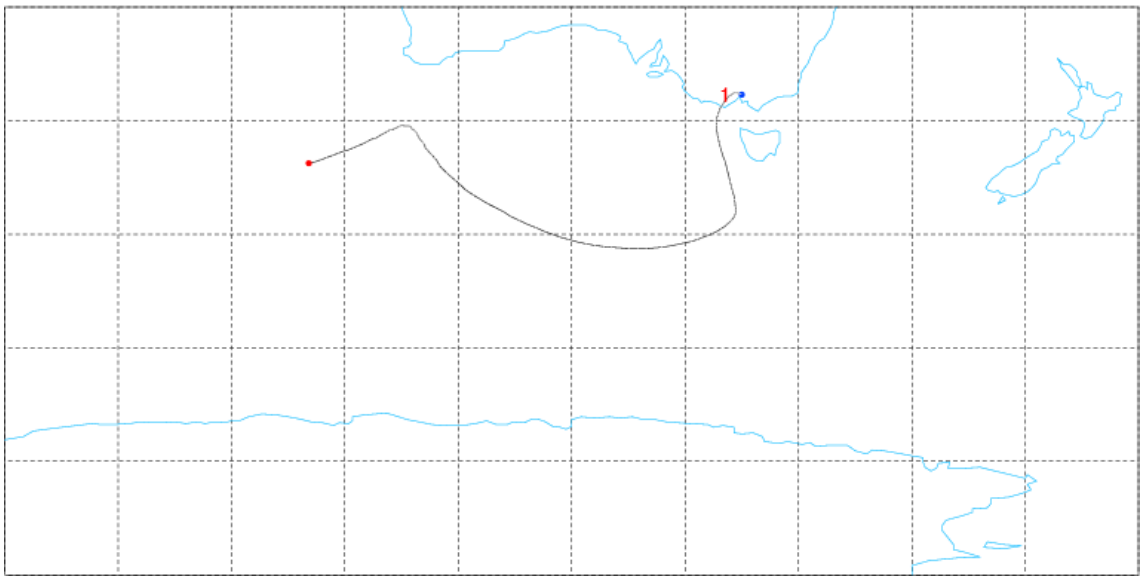


Figure 5: kmapline example – CE projection.

6.6 Xconv

If you have GRIB data you may be able to use the freely available utility Xconv to convert your GRIB file to a NetCDF file. Then you can convert this NetCDF file to CMP format with `read_nc2cmp`. The software may be used through a GUI (`xconv`) or via a shell version (`convsh`) which is useful in scripts or on the command-line. Binary versions of `xconv/convsh` for many platforms are available for download from:

http://badc.nerc.ac.uk/help/software/xconv/

For Xconv documentation see:

http://www.met.rdg.ac.uk/~jeff/xconv/

In particular, the following script (saved as `grb2nc`) will allow command-line conversion of GRIB to NetCDF
e.g. `grb2bc -i test.grb -o test.nc`

```
#!/home/kevin/bin/pc/convsh
#  Convsh script conv2nc.tcl
#
#  Convert input files into single Netcdf file.
#  All input files must contain the same fields and have
#  identical dimensions except for the time dimension.
#  For example to convert UM output files into a Netcdf file
#  use the following command:
#
#      conv2nc.tcl -i xaavaa.pc* -o xaava.nc
#  Write out Netcdf file
set outformat netcdf
#  Automatically work out input file type
set filetype 0
#  Convert all fields in input files to Netcdf
set fieldlist -1
#  Get command line arguments:
#      -i input files (can be more than one file)
#      -o output file (single file only)
set i false
set o false
foreach arg $argv {
   switch -glob -- $arg {
      -i      {set i true ; set o false}
      -o      {set i false ; set o true}
      -*      {puts "unknown option $arg" ; set i false; set o
false}
      default {
         if {$i} {
            set infile [lappend infile $arg]
         } elseif {$o} {
            set outfile [lappend outfile $arg]
         } else {
            puts "unknown option $arg"
         }
      }
   }
}
if {! [info exists infile]} {
   puts "input file name must be given"
```

```
   exit
}
if {[info exists outfile]} {
   if {[llength $outfile] > 1} {
      set outfile [lindex $outfile 0]
      puts "Only one output file can be specified, using $outfile"
   }
} else {
   puts "output file name must be given"
   exit
}
```

6.7 NCAR Graphics/NCL

The optional graphics functionality requires NCAR Graphics which is now part of the NCAR Control Language (NCL). See:

http://www.ncarg.ucar.edu/download.html

or for NCL directly:

http://www.ncl.ucar.edu/Download/

You will need to request a free ESG account if you don't have one.
Download and install the appropriate binaries e.g. Red Hat Linux, and follow the instructions. You can use NCL directly to plot your trajectories or use kmapline which uses the NCAR Graphics included in NCL. You will need the to set the environment variable `NCARG_ROOT` as well as placing th location of the NCL binaries in your `PATH` (usually this set up goes in C-shell initialisation file ~/.cshrc):
e.g.

```
# NCAR Graphics and NCL
setenv NCARG_ROOT /work/user/ncl/linux
setenv PATH   $NCARG_ROOT/bin:$PATH
```

Alternatively you may download a cut-down binary version of NCAR Graphics for Linux from:

http://www.earthsci.unimelb.edu.au/~kevin/cyc_L1.1/zncarg_linux_4.3.1_kk.zip

This is based on NCAR Graphics 4.3.1. This is provided for *convenience only*.
Then download the binary for `kmapline` included in:

http://www.earthsci.unimelb.edu.au/~kevin/cyc_L1.1/zutils_ncarg.linux.zip

In this case you will need to set the environment variable NCARG_ROOT to the location of the ncarg folder:
e.g.

```
setenv NCARG_ROOT /work/user/ncarg/
```

The directories ncarg/bin and ncarg/ubin need to be added to your PATH:

```
setenv PATH $NCARG_ROOT/bin:$NCARG_ROOT/ubin/:$PATH
```

This is mentioned in the NCAR Graphics installation notes.

7. Acknowledgements

- `traj3d`: Based on `lagadtr 2.0` written by David Noone (http://atoc.colorado.edu/~dcn/index.php) during his PhD at the University of Melbourne. Some CMP functionality was introduced by Vaughan Barras (http://www.bom.gov.au/bmrc/mdev/staf/vjb/) during his PhD at the University of Melbourne. This is turn was derived from a 2D trajectory program (`trajectory`) written by Rachel Law during her PhD at the University of Melbourne.
- `tnc2mpl`: Based on `tnc2mpl` written by David Noone.
- `read_nc2cmp`: Written by Kevin Keay (http://www.earthsci.unimelb.edu.au/~kevin). The program is based on an example NOAA program:

  http://www.cdc.noaa.gov/PublicData/readgeneral.f

  plus some suggested guidance from:

  ftp://ftp.unidata.ucar.edu/pub/netcdf/contrib/gennet.f

  by Barry Schwartz.

- `kmapline`: Based on `mapline3610` by David Noone and `mapline` by Rachel Law.
- NCL: See: http://www.ncl.ucar.edu/
- Xconv: Written by Jeff Cole (http://ncas-cms.nerc.ac.uk/people/people_research.php?user_ID=33).

8. References

Barras, V., and I. Simmonds, 2009: Observation and modeling of stable water isotopes as diagnostics of rainfall dynamics over southeastern Australia *Journal of Geophysical Research*, 114, D23308, doi:10.1029/2009JD012132, 2009.

Fuelburg, H.E., R.O. Loring Jr., M.V. Watson, M.C. Sinha, K.E. Pickering, A.M. Thomson, G.W. Sachse, D.R. Blade and M.R. Schoeberl, 1996: TRACE: A trajectory intercomparison 2. Isentropic and kinematic methods. *Journal of Geophysical Research,* 101, 23927-23939.

Noone, D., and I. Simmonds, 1999: A three-dimensional spherical trajectory algorithm. Research Activities in Atmospheric and Oceanic Modelling, Report No. 28, WMO/TD-No. 942. H. Ritchie, Ed., World Meteorological Organization, 3.26-3.27.

Law, R.M., 1993: Modelling the global transport of atmospheric constituents. PhD thesis, School of Earth Sciences, The University of Melbourne.

Perrin, G. and I. Simmonds, 1995: The origins and characteristics of cold air outbreaks over Melbourne. *Australian Meteorological Magazine*, **44**, 41-59.