A brief introduction[1] to retrieving ERA Interim via the web and webapi

Adrian Tompkins (ICTP), Email: tompkins@ictp.it

Last updated May 9, 2017

1 Analysis and Reanalysis Overview

For a full description of (re)analysis methods and the database in general see the full presentation by Tompkins (View by clicking here). Briefly, the (re)analysis products are the output of an âĂIJanalysis systemâĂİ, which attempts to combine many available data sources (satellite, radiosondes, aircraft, buoy data, stations etc) into a coherent and balanced picture of the atmospheric dynamic and thermodynamical state. Many systems are based on the use of a technique known as 4DVAR (**?**).

Over time, the analysis system of a forecast centre is subject to upgrades. these improvements may be to allow new satellite data sources to be Incorporated into the system or may be improvements to the data assimilation system physics or structure itself (for example, allowing ice supersaturation in the assimilation physics). This means that the operational analysis is not coherent in time. The analysis ten years ago was made using a very different system to the one in use operationally today.

In order to attempt to produce an analysis series that is as coherent as possible (bearing in mind different data sources changing over time), Reanalysis systems select a version of the assimilation system, and use it to rerun the assimilation for the past. These systems may them be maintained in parallel to the operational analysis in order to extend the series in near real-time. In this way the analysis system remains constant. ERA Interim was started around 2006 and uses a system that was operational at that time. This of course means that, while the system was far more advanced than those used in the 1980s and 1990s, on the other hand it does not contain

[1]Note that more complete WIKIs are available online at ECMWF. Wherever possible, I have tried to include a link to the relevant webpage. These links are dynamic and you can directly click on them if viewing this document in Adobe acrobat and many other similar PDF viewers

any of the improvements introduced since 2006, and can not incorporate new satellite information. Moreover, to make the task computationally achievable, the horizontal resolution of the reanalysis is much coarser than that employed in the present operational analysis.

Further details of ERAI reanalysis are contained in Dee et al. (2011).

Click here for further details of ECMWF Quality Issues

In the next sections we will outline the structure of the data and then discuss the two main methods for external users to retrieve data, namely the web interface and using webapi scripts.

1.1 Structure of data

The analysis procedure uses an assimilation "window" of 12 hours, but the assimilation fields are sampled twice during each window, to produce fields at 00, 06, 12 and 18Z each days. Fields are available on model levels and pressure level as well as at the surface.

Many variables that one expect to find are not available from the analysis, such as surface precipitation or radiative fluxes. This is because they are *fluxes*. If you consider for a moment, an instantaneous "snap-shot" of the rain-rate is of limited use, since it is highly variable in time. Estimating a daily rainfall total from a single instantaneous rate would lead to large errors.

Thus, instead of doing this, the rainfall and other fluxes are generated by running a short forecast starting at 00Z and 12Z each day.

2 Access via web interface

The web site address is found at http://apps.ecmwf.int/datasets/data/interim_full_daily/. In order to download data you will need to register before starting.

Opening the site you will see several groups of options, the first three referring to time-related variables.

- Date
- Time

- Step
- variable

date and time
The date and time are straightforward and refer to the analysis. Thus if you want analysis data for temperature at 12Z on the 25/03/2008, you simply enter those dates and times

Step
What does step refer to? This is the forecast step. Step 0 refers to the analysis, while any number exceeding zero refers to the number of hours the forecast has proceeded. Thus if you select temperature (T2m) and enter date=25/03/2008, time=12 and step=12, then you are selecting the temperature 12 hours into the forecast, and refers to the 00Z of 26/03/2008.

variables
A list of surface variables are given in the last space. You may select more than variable, but at first it may be wise to start with a single variable. You will notice that if you select some variables, some time and step options are blanked out and no longer selectable. For example, if you select precipitation, the option step=0 is not available. This is to remind you that fluxes are only available from the analysis.

You can select to retrieve in netcdf or grib format. Once you select your data format (e.g. netcdf) you will be asked to confirm your request. At this stage, you will have the option of changing the resolution of your data request and also the are for which you retrieve the data.

Note that the resolution of the model system creating ERA-Interim is 0.75 degree. Selecting a resolution higher than this does not add any information, since the higher resolution is obtained by simply interpolation techniques.

If you are only interested in data for a country or region, (and particularly if you have slow internet speeds), it is strongly recommended to use the region options to cut your fields down and reduce the file size significantly.

Once the file is ready, you will need to click on the link to “download” the file. Choose a relevant location to save the file (your firefox browser may save the data automatically in “Downloads” if you haven’t changed the default behaviour in your system prefereances.)

2.1 Exercises

1. Retrieve the two metre surface temperature (T2m) from ERAIfor the period 2014 to 2016 at 00,06,12 and 18Z in netcdf format.

2. Retrieve the rainfall for January 2017 in netcdf format.

3 Aside: What *are* netcdf and grib files?

It is useful to briefly introduce the concept of netcdf and grib file formats. Let’s consider a simple datafile that contains a list of numbers: 25, 16, 4, 28 and 5. If someone passes you this file, how do you know what these numbers represent? What are the units? They may give you a file with some ulterior information (called metadata) that explains this (e.g. in a csv or excel spreadsheet), but even if this is comprehensive, if you want to read this data, you still need to write a dedicated code to do so, which is time consuming and tedious.

So, why not design a standard data format, which is *self describing*, for which many standard plotting and analysis programmes can have built-in modules to read(and write) the data files? This is the purpose of NETCDF and GRIB files. There are many such formats around, but the two most common for meteorological and climate applications are GRIB and NETCDF. GRIB tends to be used for meteorological operational products, while NETCDF is more common for research models and climate model output.

GRIB and NETCDF formats are designed to be self-contained. That is, in addition to the data fields themselves, all relevant *metadata* in contained within the files to describe their content to a user without needing to provide auxiliary files. GRIB is arranged as a list of self-describing records, the data packing is very efficient, but the record header is fairly rigid in its structure reducing the flexibility of the format (it was designed with a specific usage in mind and was not necessarily intended as a general file format).

NETCDF is more flexible, with a header describing all fields and dimensions in the file; think of the header like a detailed contents page of a book. With the latest generation netcdf4, the packing methods have improved and the format can now produce files of similar size to grib. The flexibility of NETCDF means it is becoming the method of choice for writing output from a wide range of models, not only for climate applications.

4 Introduction to BASH

Most people are familiar with the windows system and are used to interacting with a computer system through the use of the mouse. Programmes are lauched by clicking on an icon, and the user then interacts with the programme by entering information in a GUI interface.

With Linux, one can still launch programmes in this way, but in addition it is common to interact with the computer by typing commands into a terminal window. These commands may also launch programmes, or they may navigate the user throughthe directory structure, create directories, move or delete files and so on. All these functions can be carried out with applications, just as with windows, but the terminal screen is a more powerful, flexible way, but perhaps slightly less intuitive.

For example, most windows users are familiar with the interactive method to open folders or "directories". Directories can be nested, i.e. directories nested within directories. On a linux system, directories are still used, but you can navigate through them using the command line in a terminal.

When you open a new terminal, you are ensential starting in a location, which is usually your home directory. This is just like opening "my folders" in Windows, which opens a folder at the "top level of your files". With the terminal window though it is not obvious *where you are.* So let us start by asking the terminal "where am I"? The appropriate question to ask is, please "print working directory", which is abbreviated to "pwd":

```
$ pwd
```

If you do this you should find that the computer responds with something like

```
  $ pwd
/home/username/
```

Normally with a graphical interface you can see the icons for files and directories in a window. With linux you need to instruct the computer to show them. The command list is abbreviated by "ls":

```
  $ ls
```

This will list your files and folders.

Most commands have options that modify their behaviour - For example you can generate a long listing

```
  $ ls -l
```

If you want to see the list of instructions available, use the manual page (essentially a help command)

```
  $ man ls
```

Now you may have saved your data in Downloads or another directory. You can *change directory* (cd):

```
  $ cd Downloads
```

Note that shell commands are *case sensitive*!

Where are you now? (pwd!)

Other useful commands:

```
$ rm file - to remove files (careful!)
$ mv file - to move files
$ cp file - to copy files
```

Remember, if you want to know how a command works, use the *man* command:

```
$ man command - man page, basically the help facility
```

5 Quick-look with ncview and ncdump

Two utilities that are very useful for examining a file are ncdump and ncview. If they are not installed then you can retrieve them using the apt-get utility:

```
$ sudo apt-get install ncdump
$ sudo apt-get install libnetcdf-dev
```

5.1 ncdump

This utility is very useful to get an idea of the contents of a file.

dumps the file header and contents:

```
$ ncdump filename
```

Only dump the header

```
$ ncdump -h filename
```

Dump the information one page at a time by *piping* it to the utility *less*:

```
$ ncdump -h filename | less
```

If a file has many fields, it is possible to display the values from one one field, or indeed the dimension variables using the -v option:

```
$ ncdump -v time filename
```

5.2 ncview

NCVIEW is a "quicklook" utility, which is useful for exploring netcdf datasets in rudimentary ways. It is not designed to produce quality plots for scientific articles or reports! A good strategy is to employ ncview for investigating output interactively, to avoid spending extensive time writing code to perform basic plots for analysis.

Quick-look the file by using ncview:

```
$ ncview filename
```

6 CDO and NCO

Climate data operators (CDO) is a command line utility that allows users to efficiently manipulate GRIB and NETCDF data. NCO is also very powerful and has many functions to manipulate meta data

It is best to ensure you have the netcdf libraries installed first

```
$ sudo apt-get install libnetcdf-dev
```

Then to install CDO on Ubuntu simply need to type:

```
$ sudo apt-get install cdo
```

The CDO manual is extremely useful and the intention is not to repeat this here, we will merely show a few commands that are useful to create timeseries data that can be potential used to drive an applications model.

6.1 Exercise 1: INSTANTANEOUS FIELDS

Here we will go through a number of ways to manipulate files, carrying out the functions you may need to carry out a project. We will focus on the things you are likely to need to do in order to get temperature for your area. You may need daily or monthly data, you probably want an average over the area of focus, or perhaps the gridcell nearest the point of interest.

As an exercise dataset, we will download two-metre atmosphere temperature (T2m) for 2014-2016, four times a day, (00,06,12,18) from the reanalysis.

If the server is slow, you can download the example temperature file directly from here

`http://clima-dods.ictp.it/Users/tompkins/workshops/smr3115/classes/`http://clima-dods.ictp.it/Users/tompkins/workshops/smr3115/classes/

We will now perform the following tasks

1. Examine the file with ncview and ncdump, does the header information look correct?

2. Make a timeseries of day mean temperatures

3. Determine if there is a trend in the temperatures

4. Make a timeseries of monthly mean temperatures
5. Make an annual cycle of the monthly mean temperatures
6. Make a timeseries for the mean over a specific box
7. Make a timeseries for a point

Interrogate the file with ncdump and ncview. cdo has some commands to also show the information of a file:

```
cdo info t2m.nc
```

that is a lot of information! Why not pipe it to a useful utility called `less`:

```
cdo info t2m.nc | less
```

If you are a user of `vi`, many of the commands (e.g. search) will work in `less`

You can get summary information with `sinfo` and `sinfon`:

```
$ cdo sinfon t2m.nc
```

What are the dates for the file? You can also examine the dates using cdo:

```
$ cdo showdate filename
```

Okay let's start the exercises:

Timeseries of day mean temperatures

We want to average the four steps each day to the day mean

```
$ cdo daymean t2m.nc t2m\_dm.nc
```

Timeseries of monthly mean temperatures

We want to average over months:

```
$ cdo monmean t2m.nc t2m_mm.nc
```

Note: you can directly download monthly means from the ERAI server. It should be pointed out that the command `cdo monmean in out` would take a reasonable programmer some time to do in R, python or FORTRAN, even if one is familiar with the packages with built in date manipulation.

Annual cycle at daily or monthly steps:

```
$ cdo ydaymean t2m.nc t2m_ydm.nc
$ cdo ymonmean t2m.nc t2m_ymm.nc
```

Was there a trend on the data.nc ?

Trend and detrend data:

```
$ cdo trend t2m_dm.nc offset.nc trend.nc
$ cdo detrend t2m_dm.nc t2m_dm_detrend.nc
```

What do you notice? If you open the trend.nc file, you can see a significant cooling in the Eastern Pacific, this is the recovery from the El Niño event of 15/16. As the even didn't start til 2015, the trend is presumably stronger if we only examine 2015/16. So let's cut out 2014...

Select a subset of dates:

```
$ cdo seldate,20150101,20161231 t2m_dm.nc t2m_dm_1516.nc
$ cdo trend t2m_dm_1516.nc offset.nc trend_1516.nc
```

Piping: You will notice that we have used two separate commands to do this. One disadvantage is that we have an extra file (`t2m_dm_1516.nc`) to clean up, plus if we could do it in one step, there is less disk i/o.

Never fear, piping is here!

Piping:

```
$ cdo trend -seldate,20150101,20161231 t2m_dm.nc offset.nc trend_1516.nc
```

Open up the two trends in ncview, you can easily set common ranges to compare.

To turn the trend into an annual value (units were K per timestep, which is a day) we need to multiply the fiel by 365:

Scaling:

```
$ cdo mulc,365 trend_1516.nc antrend_1516.nc
```

What about selecting a region and taking the average?

Select a sub-region:

```
$ cdo sellonlatbox,lon1,lon2,lat1,lat2 t2m_dm.nc region.nc
```

Spatially average field over that region:

```
$ cdo fldmean region.nc region_ts.nc
```

Let's try that in one step:

```
  $ cdo fldmean -sellonlatbox,lon1,lon2,lat1,lat2


  t2m_dm.nc region_ts.nc
```

Or perhaps you prefer to extract a single point near your preferred lat/lon:

```
cdo remapnn,lon=X,lat=Y t2m_dm.nc outfile.nc
```

Time average the field:

```
$ cdo timmean region.nc region_mean.nc
```

6.2 Exercise 2: ACUMULATED FIELDS

In this exercise we will look at accumulated fields. This includes FLUXES (such as surface latent and sensible heat fluxes, rainfall) but also includes variables that need to be checked every timestep, such as max/min temperature (over postprocessing period). Max/Min temperature are often used in statistical models so we will focus on those variables here, especially as precipation is less reliable in the analysis is preferably obtained from other sources.

Recall that there are two forecasts per day starting at 00 and 12Z, and each forecast is post-processed every 3 hours. Fluxes are accumulated from the start of the forecast, so if you want the daily rainfall, you could simply select time=00,12 and step=12, you would then need to sum the steps to get the daily total.

However, Max/min temperature are a little more complicated as they are reset each postprocessing time. Thus you need to select all postprocessing times: time=00,12, step=3,6,9,12. The file size will be large, so we will just download for January 2017 for Europe.

We have two fields in the file, so let's separate them out using selvar:

```
$ cdo selvar,mx2t  tmaxmin.nc tmax.nc
$ cdo selvar,mn2t  tmaxmin.nc tmin.nc
```

Now, you might think we simply need to find the maximum over the day? (which command?) but first use the commands you know to look at the time/date on the fields

```
$ cdo showtime tmax.nc
$ cdo showdate tmax.nc
```

What's going on? The time stamp is associated with the end of the postprocessing time. e.g. A forecast starting at 00Z and step=3, has the time 3 (not 0). Why is this important? Well if do any daily processing, then you will not working on statistics for a day, but instead 21 hours of one day, and 3 hours of the next [2].

To fix this we need to shift the time by three hours before working out the daily max. Let's use the pipe again:

```
cdo daymax -shifttime,-3hours t2m_max.nc t2m_maxd.nc
```

This would be the same for rainfall:

```
cdo daysum -shifttime,-12hours rain.nc rain_daily.nc
```

7 Simple plots in R

We will now open these two files in R and make a simple plot. You may need to analyze these data in R.

open R:

```
$ R
```

[2] I strongly suspect this issue slips by people using ERAI in many articles

(mirror sites are here: http://cran.r-project.org/mirrors.html)

Install the ncdf4 package (you only need to do this once, unless in future you upgrade your R version)

```
> install.packages("ncdf4",   repos="http://www.vps.fmvz.usp.br/CRAN/")
```

load the package (you need to do this each session)

```
> require(âĂIJncdf4âĂİ)
```

Open the file:

```
> file1<-nc_open(âĂIJregion_ts.ncâĂİ)
```

Examine the metadata:

```
> file1
```

Get the variable:

```
> t2m<-ncvar_get(file1,âĂİt2mâĂİ)
```

Plot a timeseries:

```
> plot(t2m,type=âĂŹlâĂŹ)
```

Now we will try to plot a 2D field. Get the 2d time averaged field:

```
> file2 <-nc_open(âĂIJregion_mean.ncâĂİ)
> t2m2d<-ncvar_get(file2,âĂİt2mâĂİ)
```

This time we also need the x and y axes. Find their names:

```
> file2
```

We see the names are âĂIJlongitudeâĂİ and âĂIJlatitudeâĂİ âĂŞ so now we can read those in:

```
> lon<-ncvar_get(file2,âĂİlongitudeâĂİ)
> lat<-ncvar_get(file2,âĂİlatitudeâĂİ)
```

Now unfortunately, latitude runs north to south, but the filled.contour routine wants x and y to increase... so we need to reverse the rows:

```
> t2m2dr<-apply(t2m2d,2,rev)
```

Contour plot also reverses latitude:

```
> filled.contour(lon,rev(lat),t2m2dr)
```

8 Access ECMWF data via webapi

There is an alternative method to retrieve variables, namely using webapi python-based scripts. This a new method that allows you to flexibly retrieve data directly from scripts on your desktop. This has many advantages, in that it is flexible, you can automate retrievals, and you can embed retrievals inside larger codes. The webapi is easy to set up, and the retrieval commands will be very familiar to those used to using MARS commands at ECMWF.

In order to use the webapi facility there are three steps to complete:

1. Register to use the ECMWF system
2. Download your SSH key
3. Install the ecmwfapi python package

These steps are outlined on this WIKI.

The step you will need to do is download the ecmwfapi python module, which is outlined here ECMWFAPI.

This worksheet will not give extensive details of webapi as there is a comprehensive WIKI available online:

Here is an example script:

```
from ecmwfapi import ECMWFDataServer

server = ECMWFDataServer()

server.retrieve({
    'stream'    : "oper",
    'levtype'   : "sfc",
    'param'     : "165.128/41.128",
    'dataset'   : "interim",
    'step'      : "0",
    'grid'      : "0.75/0.75",
    'time'      : "00",
    'date'      : "2013-09-01/to/2013-09-30",
    'type'      : "an",
    'class'     : "ei",
    'target'    : "erai.grb"
})
```

Some of the classes are fairly obvious if you are now familiar with the web interface, for example, time, step and date. The others will be known if you already use MARS. However, there are some new classes that require explanation.

- stream: this should always be set to “oper” unless you want to access research experiments.
- levtype: sfc=surface variable, but using this mechanism you can also retrieve on pressure levels (levtype: “pl”) or model levels (levtype: “ml”) if the data is available.
- This is the parameter name you want to retrieve, you can use the gridcode.128 or the short name given in see grib table 128. For example the two metre temperature is designated by T2m.
- grid: the resolution - the products are interpolated to this resolution.
- type: the type of product:
  - an: Analysis
  - fc: forecast
- class: the product class, we will be only using “ei” for era-interim. Many of the operational products are blocked.

8.1 Exercises

1. retrieve the two metre surface temperature (T2m) from Era-Interim for the period 1st January 2000 to 1st January 2002 at 00Z.
2. Retrieve the two metre surface temperature from ERA Interim for yesterday. What happened? This emphasizes that the reanalysis is only near real-time.
3. Retrieve the precipitation for the same period using the forecast starting at 00Z and using the forecast range step=24.

9 Other databases of interest

Many public datasets are available at :
http://apps.ecmwf.int/datasets/

- S2S - sub-seasonal forecast

\href{https://software.ecmwf.int/wiki/display/S2S/Home}{https://software.ecm

- TIGGE - short-range forecasts.
- ERA-20C : Long term reanalysis for the whole of the 20th century made using only conventional data sources for long-term consistency. Must be considered a poorer product for more recent periods due to the lack of satellite sources.

References

Dee, D. P., and Coauthors, 2011: The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Q. J. R. Meteorol. Soc.*, **137 (656)**, 553–597.