

COM INTRO 2017: GRIB Decoding - Solutions to practicals

Solution to Practical 1: using grib_dump and grib_ls

1. To list the GRIB messages in t2m.grib1

```
% grib_ls t2m.grib1
t2m.grib1
edition      centre      typeOfLevel  level      dataDate    stepRange    dataType     shortName   packingType  gridType
1            ecmf        surface      0          20170217    0            cf           2t          grid_simple  reduced_gg
1            ecmf        surface      0          20170217    6            cf           2t          grid_simple  reduced_gg
1            ecmf        surface      0          20170217    12           cf           2t          grid_simple  reduced_gg
1            ecmf        surface      0          20170217    18           cf           2t          grid_simple  reduced_gg
1            ecmf        surface      0          20170217    24           cf           2t          grid_simple  reduced_gg
5 of 5 grib messages in t2m.grib1

5 of 5 total grib messages in 1 files
```

The file contains 2 metre temperature (**shortName=2t**) from the ECMWF EPS control forecast (**dataType=cf**) for 6-hourly steps from 0 to 24 hours for **dataDate=20170217**.

Similarly, for t2m.grib2:

```
% grib_ls t2m.grib2
t2m.grib2
edition      centre      dataDate     dataType     gridType     stepRange    typeOfLevel  level      shortName   packingType
2            ecmf        20170217    cf           reduced_gg   0            meanSea     0          2t          grid_simple
2            ecmf        20170217    cf           reduced_gg   6            meanSea     0          2t          grid_simple
2            ecmf        20170217    cf           reduced_gg   12           meanSea     0          2t          grid_simple
2            ecmf        20170217    cf           reduced_gg   18           meanSea     0          2t          grid_simple
2            ecmf        20170217    cf           reduced_gg   24           meanSea     0          2t          grid_simple
5 of 5 grib messages in t2m.grib1

5 of 5 total grib messages in 1 files
```

Note the same default keys that are printed for the two editions but the order of the columns output is different. Again, the file contains 2 metre temperature (**shortName=2t**) from the ECMWF EPS control forecast (**dataType=cf**) for 6-hourly steps from 0 to 24 hours for **dataDate=20170217**. The only difference seen with grib_ls is for the value of the **typeOfLevel** key. In fact, they are the same fields as in t2m.grib1 but encoded in GRIB edition 2 and obtained from the TIGGE archive.

To print the mars keys, use `grib_ls -m`:

```
% grib_ls -m t2m.grib1
t2m.grib1
domain      levtype    date       time       step      param      class      type       stream     expver
g           sfc        20170217  0000       0          167.128    od         cf         enfo       0001
g           sfc        20170217  0000       6          167.128    od         cf         enfo       0001
g           sfc        20170217  0000       12         167.128    od         cf         enfo       0001
g           sfc        20170217  0000       18         167.128    od         cf         enfo       0001
g           sfc        20170217  0000       24         167.128    od         cf         enfo       0001
5 of 5 grib messages in t2m.grib1

5 of 5 total grib messages in 1 files
```

For `t2m.grib2`, the same command gives:

```
% grib_ls -m t2m.grib2
t2m.grib2
origin      date       time       step      levtype    number     param      expver     class      model      type       stream
ecmf        20170217  0000       0          sfc        0          167        prod       ti         glob      cf         enfo
ecmf        20170217  0000       6          sfc        0          167        prod       ti         glob      cf         enfo
ecmf        20170217  0000       12         sfc        0          167        prod       ti         glob      cf         enfo
ecmf        20170217  0000       18         sfc        0          167        prod       ti         glob      cf         enfo
ecmf        20170217  0000       24         sfc        0          167        prod       ti         glob      cf         enfo
5 of 5 grib messages in t2m.grib2

5 of 5 total grib messages in 1 files
```

To print the `shortName` along with the list of mars keys use the `-P` option e.g.:

```
% grib_ls -m -P shortName t2m.grib1
t2m.grib1
shortName domain      levtype    date       time       step      param      class      type       stream     expver
2t         g           sfc        20170217  0000       0          167.128    od         cf         enfo       0001
2t         g           sfc        20170217  0000       6          167.128    od         cf         enfo       0001
2t         g           sfc        20170217  0000       12         167.128    od         cf         enfo       0001
2t         g           sfc        20170217  0000       18         167.128    od         cf         enfo       0001
2t         g           sfc        20170217  0000       24         167.128    od         cf         enfo       0001
5 of 5 grib messages in t2m.grib1

5 of 5 total grib messages in 1 files
```

To order the output in descending step order use the `-B` option:

```
% grib_ls -B "step:i desc" t2m.grib1
edition  centre      typeOfLevel  level      dataDate    stepRange   dataType    shortName   packingType  gridType
1        ecmf             surface      0          20170217    24         cf          2t          grid_simple  reduced_gg
1        ecmf             surface      0          20170217    18         cf          2t          grid_simple  reduced_gg
1        ecmf             surface      0          20170217    12         cf          2t          grid_simple  reduced_gg
1        ecmf             surface      0          20170217    6          cf          2t          grid_simple  reduced_gg
1        ecmf             surface      0          20170217    0          cf          2t          grid_simple  reduced_gg
```

Note that you need to specify “step:i” so that step is treated as an integer rather than a character string.

2. To specify a set of keys to print, use the `-p` option:

```
% grib_ls -p centre,dataDate,stepRange,typeOfLevel,shortName t2m.grib1
t2m.grib1
centre      dataDate    stepRange   typeOfLevel  shortName
ecmf        20170217    0           surface      t2m
ecmf        20170217    6           surface      t2m
ecmf        20170217    12          surface      t2m
ecmf        20170217    18          surface      t2m
ecmf        20170217    24          surface      t2m
5 of 5 grib messages in t2m.grib1

5 of 5 total grib messages in 1 files
```

To print the keys for forecast step 6 only, use the `-p` option with the `-w` option:

```
% grib_ls -w stepRange=6 -p centre,dataDate,stepRange,typeOfLevel,shortName t2m.grib1
t2m.grib1
centre      date        stepRange   levelType    shortName
ecmf        20170217    6           surface      t2m
1 of 5 grib messages in t2m.grib1

1 of 5 total grib messages in 1 files
```

Using the “centre:i” and “centre:s” prints the centre both as an integer and a string:

```
% grib_ls -w stepRange=6 -p centre:1,centre:s,dataDate,stepRange,typeOfLevel,shortName t2m.grib1
t2m.grib1
centre      centre      dataDate    stepRange   typeOfLevel  shortName
98          ecmf        20170217    6           surface      t2m
1 of 5 grib messages in t2m.grib1

1 of 5 total grib messages in 1 files
```

Using the same commands with the file `t2m.grib2` gives similar output.

3. To use `grib_dump` to inspect the contents of the fourth message in `t2m.grib1` in Octet mode use:

```
% grib_dump -w count=4 -O t2m.grib1
```

The key aliases can be seen with:

```
% grib_dump -w count=4 -Oa t2m.grib1
```

All of the keys in this message are seen by using `grib_dump` without any options:

```
% grib_dump -w count=4 t2m.grib1
```

For `t2m.grib1`, you should find:

Section 1 gives information about the product (parameter, date, time, etc):

- the parameter is **indicatorOfParameter=67** [2 metre temperature (K)]
- the code table is **tables2Version=128**
- the time is 0 (**hour=0 and minute=0, dataTime=0000**)
- the date is 20170217 (**yearOfCentury=17, month=2, day=17, centuryOfReferenceTimeOfData=21, dataDate=20170217**)
- the forecast step is **P1=18 (stepRange=startStep=endStep=18)**.

Section 2 gives the description of the grid and shows:

```
6      dataRepresentationType = 4 [Gaussian Latitude/Longitude Grid (grib1/6.table) ]
7-8    Ni = MISSING
9-10   Nj = 1280
11-13  latitudeOfFirstGridPoint = 89892
14-16  longitudeOfFirstGridPoint = 0
17     resolutionAndComponentFlags = 0 [00000000]
18-20  latitudeOfLastGridPoint = -89892
21-23  longitudeOfLastGridPoint = 359860
24-25  iDirectionIncrement = MISSING
26-27  N = 640
```

The grid is a O640 octahedral reduced Gaussian grid and it is a global field (0 to 359.860 Longitude, -89.892 to 89.892 Latitude). The type of grid is most easily seen as the key **gridType=reduced_gg**. You should also find the key **isOctahedral** is set to 1 to indicate the grid is an octahedral reduced Gaussian grid.

In Section 1 you should find that a Local Definition is being used and this is Local Definition 30 (**localDefinitionNumber = 30**).

Similar information can be found in the file t2m.grib2 except there is no Local Definition.

Solution to Practical 2: using grib_ls -l

1. To list the nearest points to ECMWF (Lat 51.42° N, Lon 0.95° W) use

```
% grib_ls -l 51.42,-0.95 t2m.grib1
t2m.grib1
...
5 of 5 grib messages in t2m.grib1

5 of 5 total grib messages in 1 files
Input Point: latitude=51.42 longitude=-0.95
Grid Point chosen #2 index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km)
Other grid Points
- 1 - index=156198 latitude=51.38 longitude=359.35 distance=21.64 (Km)
- 2 - index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km)
- 3 - index=155082 latitude=51.52 longitude=359.35 distance=23.65 (Km)
- 4 - index=155081 latitude=51.52 longitude=359.03 distance=11.09 (Km)
```

Note we specify the longitude of 0.95° W with -0.95 !

The nearest grid point is at latitude=51.38° longitude=359.03° (0.77° W) and is 4.79 km from ECMWF. Note that GRIB API converts all Longitude values so that they lie in the interval 0° to 360°. In GRIB 2 all longitude values must be positive !

2. To output only the forecast step and 2 metre temperature value at the nearest grid point use:

```
% grib_ls -l 51.42,-0.95,1 -p stepRange t2m.grib1
t2m.grib1
stepRange      value
0              279.889
6              278.54
12             282.242
18             278.133
24             276.823
5 of 5 messages in t2m.grib1

5 of 5 total messages in 1 files
Input Point: latitude=51.42 longitude=-0.95
```

```

Grid Point chosen #2 index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km)
Other grid Points
- 1 - index=156198 latitude=51.38 longitude=359.35 distance=21.64 (Km)
- 2 - index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km)
- 3 - index=155082 latitude=51.52 longitude=359.35 distance=23.65 (Km)
- 4 - index=155081 latitude=51.52 longitude=359.03 distance=11.09 (Km)

```

Use **MODE=1** to print the value of the nearest grid point only.

To output the values at the four grid points nearest to ECMWF, use **MODE=4** which is the default:

```

% grib_ls -F"%2f" -l 51.42,-0.95,1 -p step t2m.grib1
t2m.grib1
step      value
0         279.89
6         278.54
12        282.24
18        278.13
24        276.82
5 of 5 messages in t2m.grib1

5 of 5 total messages in 1 files
Input Point: latitude=51.42 longitude=-0.95
Grid Point chosen #2 index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km)
Other grid Points
- 1 - index=156198 latitude=51.38 longitude=359.35 distance=21.64 (Km)
- 2 - index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km)
- 3 - index=155082 latitude=51.52 longitude=359.35 distance=23.65 (Km)
- 4 - index=155081 latitude=51.52 longitude=359.03 distance=11.09 (Km)

```

Here we also format the output of the values using the **-F** option.

3. To specify the land-sea mask, use:

```

% grib_ls -F"%2f" -l 51.42,-0.95,1,lsn.grib1 -p step t2m.grib1
t2m.grib1
step      value
0         279.89

```

```
6          278.54
12         282.24
18         278.13
24         276.82
5 of 5 messages in t2m.grib1
```

5 of 5 total messages in 1 files

Input Point: latitude=51.42 longitude=-0.95

Grid Point chosen #2 index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km)

Mask values:

```
- 1 - index=156198 latitude=51.38 longitude=359.35 distance=21.64 (Km) value=1.00
- 2 - index=156197 latitude=51.38 longitude=359.03 distance=4.79 (Km) value=0.99
- 3 - index=155082 latitude=51.52 longitude=359.35 distance=23.65 (Km) value=0.99
- 4 - index=155081 latitude=51.52 longitude=359.03 distance=11.09 (Km) value=0.99
```

Here the value=1.00 (or 0.99) at the end of the line showing the four nearest grid points indicates which are land points (mask=1.0). Land points have mask ≥ 0.5 ; sea points have mask < 0.5 .

Once the index of the nearest grid point is known, its value can be obtained directly with the -l option:

```
% grib_ls -F"%2f" -i 156197 -p step t2m.grib1
t2m.grib1
step          value(156197)
0            279.89
6            278.54
12           282.24
18           278.13
24           276.82
5 of 5 messages in t2m.grib1

5 of 5 total messages in 1 files
```

Solution to Practical 3: using grib_get and grib_get_data

1. To list all of the pressure levels available for the parameter T in the file tz_an_pl.grib1 use grib_get with the -w option to specify a shortName=t:

```
% grib_get -w shortName=t -p level tz_an_pl.grib1
1000
850
700
500
400
300
```

Note that you can also use any of the other aliases for level "-p lev" or "-p levelist" will produce the same list.

2. To print the stepRange in hours or minutes or seconds use, for example, "-s stepUnits=s" to specify that the step should be printed in seconds:

```
% grib_get -s stepRange=s -p stepRange surface.grib1
10800
```

3. In surface2.grib1, the step is encoded as 119s. This is not expressible as either an integer number of hours or an integer number of minutes so the following fails with an error:

```
% grib_get -p stepRange surface.grib1
ECCODES ERROR   :  unable to represent the step in h
                  Hint: try changing the step units
ECCODES ERROR   :  Decoding invalid
```

You must set stepUnits to seconds in order to decode the stepRange:

```
% grib_get -s stepRange=s -p stepRange surface2.grib1
119
```

4. To print the data values for surface.grib1 in decimal format with 5 decimal places use grib_get_data:

```
% grib_get_data -F "%.4f" surface.grib1
Latitude, Longitude, Value
2.500 -20.000 301.8488
2.500 -17.500 302.1863
2.500 -15.000 301.8105
...
```

Exponential format with 10 decimal places is the default !

```
% grib_get_data surface.grib1
Latitude, Longitude, Value
2.500 -20.000 3.0184875488e+02
2.500 -17.500 3.0218627930e+02
2.500 -15.000 3.0181054688e+02
...
```

To print the centre, date and step, use the -p option to specify these keys:

```
% grib_get_data -p centre,dataDate,stepRange surface.grib1
Latitude, Longitude, Value, centre, dataDate, stepRange
2.500 -20.000 3.0184875488e+02 ecmf 20170221 3
2.500 -17.500 3.0218627930e+02 ecmf 20170221 3
2.500 -15.000 3.0181054688e+02 ecmf 20170221 3
...
```

Note that the additional keys are always printed after the latitude, longitude and values.

Missing values can be identified by using the -m option, for example:

```
% grib_get_data -m MISSING surface.grib1
Latitude, Longitude, Value
...
2.500 7.500 3.0176782227e+02
2.500 10.000 MISSING
0.000 -20.000 3.0134069824e+02
...
```

3. Using `grib_get_data` to print the data values in T500 **only** for the field in `tz_an_pl.grib1` use:

```
% grib_get_data -w shortName=t,level=500, tz_an_pl.grib1
Value
2.5816772461e+02
0.0000000000e+00
-3.3888750076e+00
0.0000000000e+00
-1.0073157310e+01
...
```

In this case, only the data values are printed; there are no values of latitude and longitude. This is because the field is in *spectral* representation. The values shown are not the values of temperature at 500 hPa but, instead, those of the data values themselves (which are the spectral components of the field). In this case, it is not possible for `grib_get_data` to determine latitude-longitude values.

Solution to Practical 4: modifying GRIB messages

1. To copy only those messages for parameter T from the GRIB file tz_an_pl.grib1 use grib_copy with the -w option to specify a shortName=t:

```
% grib_copy -w shortName=t tz_an_pl.grib1 t_an_pl.grib1
```

Similarly to copy only those messages for parameter Z, use:

```
% grib_copy -w shortName=z tz_an_pl.grib1 z_an_pl.grib1
```

Or, more simply and because the file contains only parameters T and Z, one can use:

```
% grib_copy tz_an_pl.grib1 "[shortName]_an_pl.grib1"
```

Using grib_ls of the two files confirms that the contents are correct. For example:

```
% grib_ls -p centre,paramId,typeOfLevel,level,date,dataType,time,shortName t_an_pl.grib1
t_an_pl.grib1
centre      paramId      typeOfLevel  level      date        dataType     time        shortName
ecmf        130          isobaricInhPa 1000       20170219    an           1200        t
ecmf        130          isobaricInhPa 850        20170219    an           1200        t
ecmf        130          isobaricInhPa 700        20170219    an           1200        t
ecmf        130          isobaricInhPa 500        20170219    an           1200        t
ecmf        130          isobaricInhPa 400        20170219    an           1200        t
ecmf        130          isobaricInhPa 300        20170219    an           1200        t
6 of 6 messages in t_an_pl.grib1

6 of 6 total messages in 1 files
```

You can take this further and split the file tz_an_pl.grib1 into separate files for each parameter/pressure level combination with:

```
% grib_copy tz_an_pl.grib1 "[shortName]_[level].grib[edition]"

% ls ?_*.grib1
t_1000.grib1  t_400.grib1  t_700.grib1  z_1000.grib1  z_400.grib1  z_700.grib1
t_300.grib1  t_500.grib1  t_850.grib1  z_300.grib1  z_500.grib1  z_850.grib1
```

Each file contains one message only, e.g.:

```
% grib_ls -p centre,paramId,typeOfLevel,level,date,dataType,time,shortName t_1000.grib1
t_1000.grib1
centre      paramId      typeOfLevel  level      date        dataType     time        shortName
ecmf        130          isobaricInhPa 1000       20170219    an           1200        t
1 of 1 messages in t_1000.grib1
```

1 of 1 total messages in 1 files

2. Use `grib_ls` to inspect the contents of the file `tp.grib1`, printing also the `paramId` with the `-P` option:

```
% grib_ls -P paramId tp.grib1
tp.grib1
paramId      edition      centre      typeOfLevel  level      dataDate      stepRange      dataType      shortName      packingType  gridType
142         1            ecmf        surface      0          20170221     24            fc            lsp           grid_simple  regular_ll
1 of 1 messages in tp.grib1

1 of 1 total messages in 1 files
```

Note that the `shortName` is set to **lsp** and `paramId` to **142**, i.e., large-scale precipitation.

The field was created using MARS compute to sum the convective and large-scale (stratiform) precipitation. When using MARS compute, the output field maintains the information from the GRIB header of the first input field so, in this case, we see that the field has `paramId` 142 even though the message now contains the total precipitation.

To change the parameter to total precipitation, use `grib_set` as follows:

```
% grib_set -s shortName=tp tp.grib1 tp_new.grib1
```

Check with `grib_ls`:

```
% grib_ls -P paramId tp_new.grib1
tp_new.grib1
paramId      edition      centre      typeOfLevel  level      dataDate      stepRange      dataType      shortName      packingType  gridType
228         1            ecmf        surface      0          20170221     24            fc            tp           grid_simple  regular_ll
1 of 1 messages in tp_new.grib1

1 of 1 total messages in 1 files
```

Note that changing the `shortName` has also changed the value of `paramId`. The same could be achieved by setting `paramId=228`:

```
% grib_set -s paramId=228 tp.grib1 tp_new.grib1
```

3. To convert the GRIB messages in `file.grib1` to NetCDF with data type `NC_SHORT` use:

```
% grib_to_netcdf -o out1.nc file1.grib1
grib_to_netcdf: Version 2.2.0
```

```
grib_to_netcdf: Processing input file 'file1.grib1'.
grib_to_netcdf: Found 4 GRIB fields in 1 file.
grib_to_netcdf: Ignoring key(s): method, type, stream, refdate, hdate
grib_to_netcdf: Creating netCDF file 'out1.nc'
grib_to_netcdf: NetCDF library version: 4.3.2 of May 10 2016 11:12:41 $
grib_to_netcdf: Creating large (64 bit) file format.
grib_to_netcdf: Defining variable 't2m'.
grib_to_netcdf: Done.
```

Using **ncdump** to print the values for the variable t2m shows:

```
% ncdump -v t2m out1.nc
...
data:

t2m =
  20497, -809,
  -16312, -32766,
  32767, 7042,
  -1312, -18041,
  17168, -12207,
  -4148, -28392,
  19382, -10827,
  -2506, -25406 ;
}
```

The data values appear as integers because of the **NC_SHORT** data format. These values need to be unpacked using the **scale_factor** and **add_offset** NetCDF attributes.

Repeating but setting the netCDF data format to **NC_FLOAT** gives:

```
% grib_to_netcdf -D NC_FLOAT -o out2.nc file1.grib1
grib_to_netcdf: Version 2.2.0
grib_to_netcdf: Processing input file 'file1.grib1'.
grib_to_netcdf: Found 4 GRIB fields in 1 file.
grib_to_netcdf: Ignoring key(s): method, type, stream, refdate, hdate
grib_to_netcdf: Creating netCDF file 'out2.nc'
grib_to_netcdf: NetCDF library version: 4.3.2 of May 10 2016 11:12:41 $
grib_to_netcdf: Creating large (64 bit) file format.
grib_to_netcdf: Defining variable 't2m'.
grib_to_netcdf: Done.
```

```

% ncdump -v out2.nc
...
data:

t2m =
  281.1666, 277.9943,
  275.6859, 273.236,
  282.9937, 279.1632,
  277.9194, 275.4285,
  280.671, 276.2971,
  277.4971, 273.8872,
  281.0007, 276.5026,
  277.7416, 274.3318 ;
}

```

In this case, the data values do not need further processing.

4. Converting file2.grib to NetCDF gives the following:

```

% grib_to_netcdf -o out3.nc file2.grib1
grib_to_netcdf: Version 2.2.0
grib_to_netcdf: Processing input file 'file2.grib1'.
grib_to_netcdf: Found 1 GRIB field in 1 file.
grib_to_netcdf: Ignoring key(s): method, type, stream, refdate, hdate
grib_to_netcdf: Creating netCDF file 'out3.nc'
grib_to_netcdf: NetCDF library version: 4.3.2 of May 10 2016 11:12:41 $
grib_to_netcdf: Creating large (64 bit) file format.
ECCODES ERROR : First GRIB is not on a regular lat/lon grid or on a regular Gaussian grid. Exiting.

```

The conversion fails because the GRIB data is represented on a reduced Gaussian grid (**gridType=reduced_gg**). Conversion to NetCDF is possible only for GRIB data with **gridType=regular_ll** or **gridType=regular_gg**.

Solution to Practical 5: GRIB decoding with Fortran 90

1. The output from the `eccodes_demo` program should show the various GRIB section contents as obtained when inspecting the file using the `grib_dump` tool without specifying any options.

2. Using `grib_ls` to examine the data file:

```
% grib_ls grib_file.grib
```

lists the GRIB messages.

Using `grib_dump -O` will dump the contents of the message:

```
% grib_dump -O grib_file.grib
```

The file contains a mixture of ECMWF operational analysis surface, pressure and model level fields in both GRIB edition 1 and 2. This is typical of the mixture of data that might be received in dissemination. The idea now is to change the program so it can be used to decode only a few keys and values from each message **using the same function calls**.

3. Replace the call to `codes_dump(gribid)` with calls to `codes_get` for each of the keys. Remember it is also necessary to declare the variables that will hold the key values. The following is one possible solution:

```
CHARACTER (LEN=9)           :: shortName, date
CHARACTER (LEN=15)          :: levType
INTEGER                     :: time, level, edition, param
INTEGER                     :: numberOfValues
REAL                        :: maximum, minimum, average
REAL, DIMENSION (:), ALLOCATABLE :: values
...
!CALL codes_dump(gribid)

! Get the values for edition, shortName, dataDate, paramId, and level
CALL codes_get(gribid, 'edition', edition)
CALL codes_get(gribid, 'shortName', shortName)
```

```

CALL codes_get(gribid,'paramId',param)
CALL codes_get(gribid,'dataDate',date)
CALL codes_get(gribid,'typeOfLevel',levType)
CALL codes_get(gribid,'level',level)
WRITE(6,*) 'Edition=',edition,' shortName=',shortName, &
           ' paramId=',param,' Date=',date, &
           ' typeOfLevel=', levType, ' level=',level

CALL codes_get(gribid,'numberOfValues',numberOfValues)
ALLOCATE(values(numberOfValues),iret)
IF (iret /= 0) THEN
  STOP 'eccodes_demo: allocate failed'
END IF
CALL codes_get(gribid,'values',values)

WRITE(6,*) "First 20 values:"
WRITE(6,*) "-----"
WRITE(6,'(f15.4)')(values(i),i=1,20)

CALL codes_get(gribid,'max',maximum)
CALL codes_get(gribid,'min',minimum)
CALL codes_get(gribid,'average',average)

WRITE(6,*)
WRITE(6,'("max = ",f15.4," min = ",f15.4," average = ",f15.4)') &
      maximum, minimum, average
...
DEALLOCATE(values)

```

Remake the executable and re-run. With these changes, for each message the output is now of the form:

```

--- GRIB No.      1 ---
Edition =          1 Parameter 2t      typeOfLevel = surface      level =          0      date =
20170221
First 20 values:
-----
      296.8286

```

297.6882
297.6785
298.2374
298.5768
298.0291
297.7993
298.4066
299.2717
298.5341
299.0988
299.6472
299.7238
299.5298
299.8495
299.9126
300.2761
300.2351
299.7808
299.5432

max = 303.2025 min = 295.5269 average = 298.9595