grib_compare

DESCRIPTION

Compare GRIB messages contained in two files. If some differences are found it fails returning an error code. Floating point values are compared exactly by default, different tolerance can be defined see -P -A -R. Default behaviour: absolute error=0, bit-by-bit compare, same order in files.

USAGE

grib_compare [options] grib_file1 grib_file2

OPTIONS

-r

Compare files in which the messages are not in the same order. This option is time expensive.

-b key,key,...

All the keys in this list are skipped in the comparison. Bit-by-bit compare on.

-e

Edition independent compare. It is used to compare grib edition 1 and 2.

-2

Enable two-way comparison.

-c key[:i|d|s|n],key[:i|d|s|n],...

Only the listed keys or namespaces (:n) are compared. The optional letter after the colon is used to force the type in the comparison: i->integer, d->float, s->string, n->namespace. See -a option. Incompatible with -H option.

-S start

First field to be processed.

-E end

Last field to be processed.

-a

-c option modifier. The keys listed with the option -c will be added to the list of keys compared without -c.

-H

Compare only message headers (everything except data and bitmap). Bit-by-bit compare on. Incompatible with -c option.

-R key1=relative_error1,key2=relative_error2,...

Compare floating point values using the relative error as tolerance. key1=relative_error will compare key1 using relative_error1. all=relative_error will compare all the floating point keys using relative_error. Default all=0.

-A absolute error

Compare floating point values using the absolute error as tolerance. Default is absolute error=0

-P

Compare data values using the packing error as tolerance.

-T factor

Compare data values using factor multiplied by the tolerance specified in options -P -R -A.

-w key[:{s|d|i}]{=|!=}value,key[:{s|d|i}]{=|!=}value,...

Where clause. Messages are processed only if they match all the key/value constraints. A valid constraint is of type key=value or key!=value. For each key a string (key:s), a double (key:d) or an integer (key:i) type can be specified. Default type is string. In the value you can also use the forward-slash character '/' to specify an OR condition (i.e. a logical disjunction) Note: only one -w clause is allowed.

-f

Forcefully compare, do not stop after first difference.

-V

Version.

-7

Does not fail when the message has wrong length

-v

Verbose.

grib_compare examples

1. The default behaviour for grib_compare without any option is to perform a bit by bit comparison of the two messages. If the messages are found to be bitwise different then grib_compare switches to a "key based" mode to find out which coded keys are different. To see how grib_compare works we first set the shortName=2d (2 metre dew point temperature) in the file regular_latlon_surface.grib1

   ```
   > grib_set -s shortName=2d regular_latlon_surface.grib1 2d.grib1
   ```

   Then we can compare the two fields with grib_compare.

   ```
   > grib_compare regular_latlon_surface.grib1 2d.grib1

   -- GRIB #1 -- shortName=2t paramId=167 stepRange=0 levelType=sfc level=0 packingType=grid_simple
   gridType=regular_ll --
   long [indicatorOfParameter]: [167] != [168]
   ```

   In the output we see that the only "coded" key with different values in the two messages is indicatorOfParameter which is the relevant key for the parameter information. The comparison can be forced to be successful listing the keys with different values in the -b option.

   ```
   > grib_compare -b indicatorOfParameter regular_latlon_surface.grib1 2d.grib1
   ```

2. Two grib messages can be very different because they have different edition, but they can contain the same identical information in the header and the same data. To see how grib_compare can help in comparing messages with different edition we do

   ```
   > grib_set edition=2 reduced_gaussian_model_level.grib1 reduced_gaussian_model_level.grib2
   ```

   Then we compare the two fields with grib_compare.

```
> grib_compare reduced_gaussian_model_level.grib1 reduced_gaussian_model_level.grib2

-- GRIB #1 -- shortName=t paramId=130 stepRange=0 levelType=ml level=1 packingType=grid_simple
gridType=reduced_gg --
long [totalLength]: [10908] != [10996]
long [editionNumber]: [1] != [2]
long [section1Length]: [52] != [21]
[table2Version] not found in 2nd field
[gridDefinition] not found in 2nd field
[indicatorOfParameter] not found in 2nd field
[indicatorOfTypeOfLevel] not found in 2nd field
[yearOfCentury] not found in 2nd field
[unitOfTimeRange] not found in 2nd field
[P1] not found in 2nd field
[P2] not found in 2nd field
[numberIncludedInAverage] not found in 2nd field
[numberMissingFromAveragesOrAccumulations] not found in 2nd field
[centuryOfReferenceTimeOfData] not found in 2nd field
[reservedNeedNotBePresent] not found in 2nd field
[perturbationNumber] not found in 2nd field
[numberOfForecastsInEnsemble] not found in 2nd field
[padding_local1_1] not found in 2nd field
long [section2Length]: [896] != [17]
[pvlLocation] not found in 2nd field
[dataRepresentationType] not found in 2nd field
long [latitudeOfFirstGridPoint]: [87864] != [87863799]
long [latitudeOfLastGridPoint]: [-87864] != [-87863799]
long [longitudeOfLastGridPoint]: [357188] != [357187500]
[padding_grid4_1] not found in 2nd field
long [section4Length]: [9948] != [770]
[dataFlag] not found in 2nd field
```

It is clear that the two messages are coded in a very different way. If we now add the -e option, the tool will compare only the higher level information common between the two messages.

```
> grib_compare -e reduced_gaussian_model_level.grib1 reduced_gaussian_model_level.grib2

-- GRIB #1 -- shortName=t paramId=130 stepRange=0 levelType=ml level=1 packingType=grid_simple
gridType=reduced_gg --
string [param]: [130.128] != [130]
```

The comparison is successful because the two messages contain the same information coded in two different ways. We can display the list of keys used by grib_compare adding the option -v (verbose).

```
> grib_compare -ve reduced_gaussian_model_level.grib1 reduced_gaussian_model_level.grib2
reduced_gaussian_model_level.grib2
  comparing centre as string
  comparing paramId as long
  comparing units as string
  comparing name as string
  comparing shortName as string
  comparing typeOfLevel as string
  comparing level as long
  comparing pv as double
  (184 values) tolerance=0         using compare_double_absolute
  comparing bitmapPresent as long
  comparing latitudeOfFirstGridPointInDegrees as double
  (1 values) tolerance=0.0005         using compare_double_absolute
  comparing longitudeOfFirstGridPointInDegrees as double
  (1 values) tolerance=0.0005         using compare_double_absolute
  comparing latitudeOfLastGridPointInDegrees as double
  (1 values) tolerance=0.0005         using compare_double_absolute
  comparing longitudeOfLastGridPointInDegrees as double
  (1 values) tolerance=0.0005         using compare_double_absolute
  comparing iDirectionIncrementInDegrees is set to missing in both fields
  comparing N as long
  comparing iScansNegatively as long
  comparing jScansPositively as long
  comparing jPointsAreConsecutive as long
  comparing pl as long
  comparing gridType as string
  comparing packedValues as double
  (6114 values) tolerance=0         using compare_double_absolute
  comparing domain as string
  comparing levtype as string
  comparing levelist as long
  comparing date as long
  comparing time as long
  comparing step as long
  comparing param as string

-- GRIB #1 -- shortName=t paramId=130 stepRange=0 levelType=ml level=1 packingType=grid_simple
gridType=reduced_gg --
string [param]: [130.128] != [130]
  comparing class as string
  comparing type as string
  comparing stream as string
  comparing expver as string
```

For each key the type used in the comparison is reported and for the floating point keys also the tolerance used is printed.

3. Some options are provided to compare only a set of keys in the messages. The option -H is used to compare only the headers coded in the message, it doesn't compare the data values. The option "-c key1:[i|d|s|n],key2:[i|d|s|n],... " can be used to compare a set of keys or namespaces. The letter after the colon is optional and it is used to force the type used in the comparison which is otherwise assumed to be the native type of the key. The possible types are:

    - :i -> integer
    - :d -> floating point (C type double)
    - :s -> string
    - :n -> namespace.

    When the type "n" is used all the set of keys belonging to the specified namespace are compared assuming their own native type. To illustrate how these options work we change the values coded in a message using grib_filter with the following rules file (see grib_filter).

```
set bitsPerValue=10;
set values={1,2.5,3,4,5,6,70};
write "first.grib1";
set values={1,2.5,5,4,5,6,70};
write "second.grib1";
```

We first compare the two files using the -H option (only headers are compared).

```
> grib_compare -H first.grib1 second.grib1
```

The comparison is successful because the data are not compared. To compare only the data we have to compare the "data namespace".

```
> grib_compare -c data:n first.grib1 second.grib1

-- GRIB #1 -- shortName=t paramId=130 stepRange=0 levelType=ml level=1 packingType=grid_simple
gridType=reduced_gg --
double [packedValues]: 1 out of 7 different
 max absolute diff. = 2.0000000000000000e+00, relative diff. = 0.4
        max diff. element 2: 3.0000000000000000000e+00 5.0000000000000000000e+00
        tolerance=0.0000000000000000e+00 packingError: [0.0625005] [0.0625005]
        values max= [70]  [70]         min= [1] [1]
```

The comparison is showing that one of seven values is different in a comparison with the (default) absolute tolerance=0. We can change the tolerance with the -A option:

```
> grib_compare -A 2 -c data:n first.grib1 second.grib1
```

and we see that the comparison is successful if the absolute tolerance is set to 2. We can also set the relative tolerance for each key with the option -R:

```
> grib_compare -R packedValues=0.4 -c data:n first.grib1 second.grib1
```

and we get again a successful comparison because the relative tolerance is bigger than the relative absolute difference of two corresponding values. Another possible choice for the tolerance is to be equal to the packingError, which is the error due to the packing algorithm. If we change the decimalPrecision of a packed field we introduce a packing error sometimes bigger than the original packing error.

```
> grib_set -s changeDecimalPrecision=0 first.grib1 third.grib1
```

and we compare the two fields using the -P option (tolerance=packingError).

```
> grib_compare -P -c data:n first.grib1 third.grib1
```

the comparison is successful because their difference is within the biggest of the two packing error. With the option -P the comparison is failing only if the original data coded are different, not if the packing precision is changed. If we try again to compare the fields without the -P option:

```
> grib_compare -c data:n first.grib1 third.grib1

-- GRIB #1 -- shortName=t paramId=130 stepRange=0 levelType=ml level=1 packingType=grid_simple
gridType=reduced_gg --
double [packedValues]: 1 out of 7 different
 max absolute diff. = 5.0000000000000000e-01, relative diff. = 0.166667
        max diff. element 1: 2.5000000000000000000e+00 3.0000000000000000000e+00
        tolerance=0.0000000000000000e+00 packingError: [0.0625005] [0.5]
        values max= [70]  [70]         min= [1] [1]
```

we see that some values are different and that the maximum absolute differenc is close to the biggest packing error (max diff=0.48 packingError=0.5). The packing error was chosen to be 0.5 by setting decimalPrecision to 0 which means that we don't need to preserve any decimal figure.

4. When we already know that the fields are not numerically identical, but have similar statistical characteristics we can compare their statistics namespaces:

```
> grib_compare -c statistics:n first.grib1 third.grib1

-- GRIB #1 -- shortName=t paramId=130 stepRange=0 levelType=ml level=1 packingType=grid_simple
gridType=reduced_gg --
double [avg]: [1.3071428571428571748e+01] != [1.3142857142857142349 6e+01]
        absolute diff. = 0.0714286, relative diff. = 0.00543478
        tolerance=0
double [sd]: [2.3290753179609058776 1e+01] != [2.3258967987353496909 0e+01]
        absolute diff. = 0.0317852, relative diff. = 0.00136471
        tolerance=0
double [skew]: [2.0229502795016589544 7e+00] != [2.0238567340070559019 7e+00]
        absolute diff. = 0.000906455, relative diff. = 0.000447885
        tolerance=0
double [kurt]: [2.1269752759397224650 7e+00] != [2.1290665824261889582 7e+00]
        absolute diff. = 0.00209131, relative diff. = 0.000982264
        tolerance=0
```

and we see that maximum, minimum, average, standard deviation, skewness and kurtosis are compared. While the values are different by 0.48 the statistics comparison shows that the difference in the statistical values is never bigger than 0.052

```
> grib_compare -A 0.052 -c statistics:n first.grib1 third.grib1

-- GRIB #1 -- shortName=t paramId=130 stepRange=0 levelType=ml level=1 packingType=grid_simple
gridType=reduced_gg --
double [avg]: [1.3071428571428571748e+01] != [1.3142857142857142349 6e+01]
        absolute diff. = 0.0714286, relative diff. = 0.00543478
        tolerance=0.052
```

The statistics namespace is available also for spherical harmonics data and provides information about the field in the geographic space computing them in the spectral space for performance reasons.

1. When a file contains several fields and some keys are different, it is useful to have a summary report of the keys found different in the messages. This can be obtained with the option -f. We change few keys in a file:

```
> grib_set -w typeOfLevel=surface -s step=48 tigge_pf_ecmwf.grib2 out.grib2
```

and comparing with the -f option:

```
> grib_compare -f tigge_pf_ecmwf.grib2 out.grib2

-- GRIB #9 -- shortName=skt paramId=235 stepRange=96 levelType=sfc level=0 packingType=grid_simple
gridType=regular_ll --
long [forecastTime]: [96] != [48]

-- GRIB #10 -- shortName=sd paramId=228141 stepRange=96 levelType=sfc level=0 packingType=grid_simple
gridType=regular_ll --
long [forecastTime]: [96] != [48]

-- GRIB #11 -- shortName=sf paramId=228144 stepRange=0-96 levelType=sfc level=0 packingType=grid_simple
gridType=regular_ll --
long [dayOfEndOfOverallTimeInterval]: [26] != [24]
long [lengthOfTimeRange]: [96] != [48]

...  output deleted

## ERRORS SUMMARY #######
##
## Summary of different key values
## forecastTime ( 3 different )
## dayOfEndOfOverallTimeInterval ( 11 different )
## lengthOfTimeRange ( 11 different )
##
## 14 different messages out of 38
```

we get a list of all the different messages in the files and a summary report of the different keys.

2. We can change the order of the messages in a file using grib_copy with the -B option:

```
> grib_copy -B typeOfLevel tigge_pf_ecmwf.grib2 out.grib2
```

If we now compare the two files:

```
> grib_compare -f tigge_pf_ecmwf.grib2 out.grib2

-- GRIB #1 -- shortName=10u paramId=165 stepRange=96 levelType=sfc level=10 packingType=grid_simple
gridType=regular_ll --
long [discipline]: [0] != [2]
long [totalLength]: [1555] != [990]
long [parameterCategory]: [2] != [0]
long [parameterNumber]: [2] != [22]
long [scaledValueOfFirstFixedSurface]: [10] != [0]
long [typeOfSecondFixedSurface]: [255] != [106]
scaleFactorOfSecondFixedSurface is set to missing in 1st field is not missing in 2nd field
scaledValueOfSecondFixedSurface is set to missing in 1st field is not missing in 2nd field
long [numberOfValues]: [684] != [239]
double [referenceValue]: [-1.5722932815551757812e+01] != [4.1584381103515625000e+01]
        absolute diff. = 57.3073, relative diff. = 1.3781
        tolerance=3.8147e-06
long [binaryScaleFactor]: [-10] != [-15]
long [bitsPerValue]: [16] != [24]
long [section6Length]: [6] != [92]
long [bitMapIndicator]: [255] != [0]
long [section7Length]: [1373] != [722]
Different size for "codedValues"  [684]  [239]
...    very long output
```

the comparison is failing because of the different order of the messages. We can use the -r option to compare the files assuming that the messages are not in the same order:

```
> grib_compare -r tigge_pf_ecmwf.grib2 out.grib2
```

and we have a successful comparison because for each message in the first file an identical message is found in the second file. This option should be used carefully as it is very time expensive.