

# ecCodes GRIB Advanced Topics

## Part I

Shahram Najm

Development Section

Forecast Department

# Overview

- **Simple Packing**
- **Constant fields**
- **Bitmap**

# Simple packing: Loss of information

**IEEE 64 floating point**

Simple packing


**N-bits scaled integer**

**Usually  $N = 8, 10, 16, 24$**

# Simple packing: Keys


- **values**
  - **decimalPrecision**
  - **changeDecimalPrecision**
  - **packingError (read only)**
- **referenceValue (read only)**
  - **bitsPerValue**
  - **decimalScaleFactor**
  - **binaryScaleFactor (read only)**

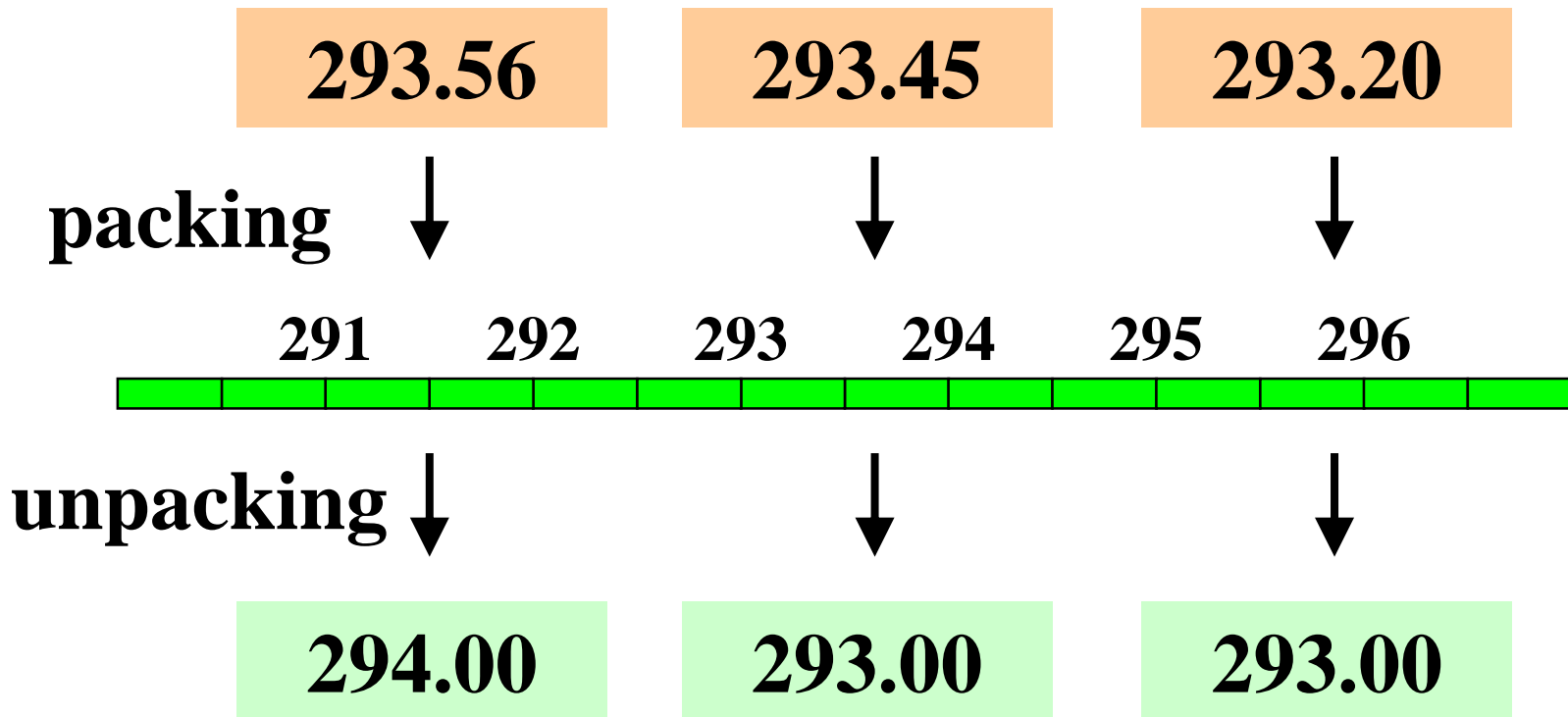
**Use these keys  
only if you know  
how packing  
works**



**Note: setting “decimalPrecision” does not repack data but setting “changeDecimalPrecision” does!**

# Simple packing = discretization

packingError=0.5 →   
decimalPrecision=0



# Simple packing

$$\text{Original value} = \text{Unpacked value} + \text{packingError}$$

**Packing error depends on the packing parameters:**

**bitsPerValue, decimalScaleFactor, binaryScaleFactor, referenceValue**

# Decimal precision

**Decimal precision = decimal digits to be preserved**

**decimalPrecision = 0** → **packingError = 0.5**

**decimalPrecision = 1** → **packingError = 0.05**

**decimalPrecision = 2** → **packingError = 0.005**

# Simple packing: Example

- **Imagine a hypothetical 12-hour 500 hPa geopotential height forecast with values ranging from 5340 to 5460 gpm**
- **For a decimal precision of 1 we scale all values by 10 so now they will range from 53400 to 54600**
- **The “decimalScaleFactor”  $D$  is chosen such that when the original data is multiplied by  $10^D$ , the integer part of the result will have enough precision to contain all the information**
- **The “referenceValue” is the minimum (i.e. 53400) . Subtract this from all values to leave non-negative *residuals* ranging from 0 to 1200**
- **The calculated bit-length for this range is 11 bits**
- **All values are now packed into words 11 bits long**



# Constant fields

- **In a constant field all the values are the same**
- Repeating the same value N times is very inefficient
- The constant value is the only value stored and the data section is empty
- Constant fields are very small and they are very precisely encoded
- A constant field can be easily created with:  

```
grib_set -d 1 in.grib out.grib
```
- In a constant field the packing parameters are not defined (**bitsPerValue=0**)

# Constant fields problem

## WARNING

At this point the packing parameters are not known.

We load a constant field

```
codes_grib_new_from_file(infile, igrib)
```

We set some non-constant values

```
codes_set(igrib, 'values', values)
```

We write the field

```
codes_write(igrib, outfile)
```

What packingError can we expect?

In the constant field the packing parameters are not set.  
ecCodes doesn't know what precision we require.  
A safe choice is made **bitsPerValue=24**.

## Constant fields

It is better practice to set **decimalPrecision** or **bitsPerValue** before packing the values

```
codes_grib_new_from_file(infile,igrib)
codes_set(igrib,'decimalPrecision',4)
codes_set(igrib,'values',values)
codes_write(igrib,outfile)
```

```
codes_grib_new_from_file(infile,igrib)
codes_set(igrib,'bitsPerValue',16)
codes_set(igrib,'values',values)
codes_write(igrib,outfile)
```

# Constants and precision: Practicals

```
cd $SCRATCH  
tar -xf ~trx/ecCodes/eccodes_grib_packing.tar  
cd grib_packing/constant
```

- 1. You have a GRIB file constant.grib**
- 2. Set values = {23.26, 42.51, 61.22, 45.95} and print packingError and bitsPerValue**
- 3. Set decimalPrecision=1 and set the same values. Print again packingError and bitsPerValue**
- 4. Compare file sizes and packingErrors**

(Hint: you can use grib\_filter)

# Bitmap

- The bitmap is an array of binary values. Its purpose is to indicate the **presence** or **absence** of data at each of the grid points. A value of '0' means data is missing and a '1' means data is present
- In order to conserve space, the bitmap is used to efficiently indicate those data points that actually appear in the Data Section

0	0	0	0				
0	1	1	0		2.45	4.67	
0	0	1	0			9.11	

Bitmap section

Data section

# Bitmap

- The bitmap size is the number of points in the grid (numberOfPoints)
  - 0 -> value is missing
  - 1 -> value is present
- When encoding, you can use the key **missingValue** to tell the library where data is missing
- By default this is 9999 but it can be changed by the user e.g. a value out of the range of normal data
- You must also set the key **bitmapPresent** to 1
- When the library encounters a value equal to the missing value in the data array, it will set the bitmap entry to 0 for that grid point
- When decoding, you can directly query the bitmap to discover missing data values

# Bitmap: Practicals

```
cd $SCRATCH  
cd grib_packing/bitmap
```

- 1. You have a GRIB start.grib with 4 messages. Set**
  - 1.bitsPerValue=8, bitmapPresent=0 in the first message**
  - 2.bitsPerValue=16, bitmapPresent=0 in the second message**
  - 3.bitsPerValue=24, bitmapPresent=0 in the third message**
  - 4.bitsPerValue=8, bitmapPresent=1 in the fourth message**
- 2. Set values = {0.2, 0.4, 0.6, 0.7, 9999}**
- 3. Print the values**  
(Hint: you can use grib\_filter)