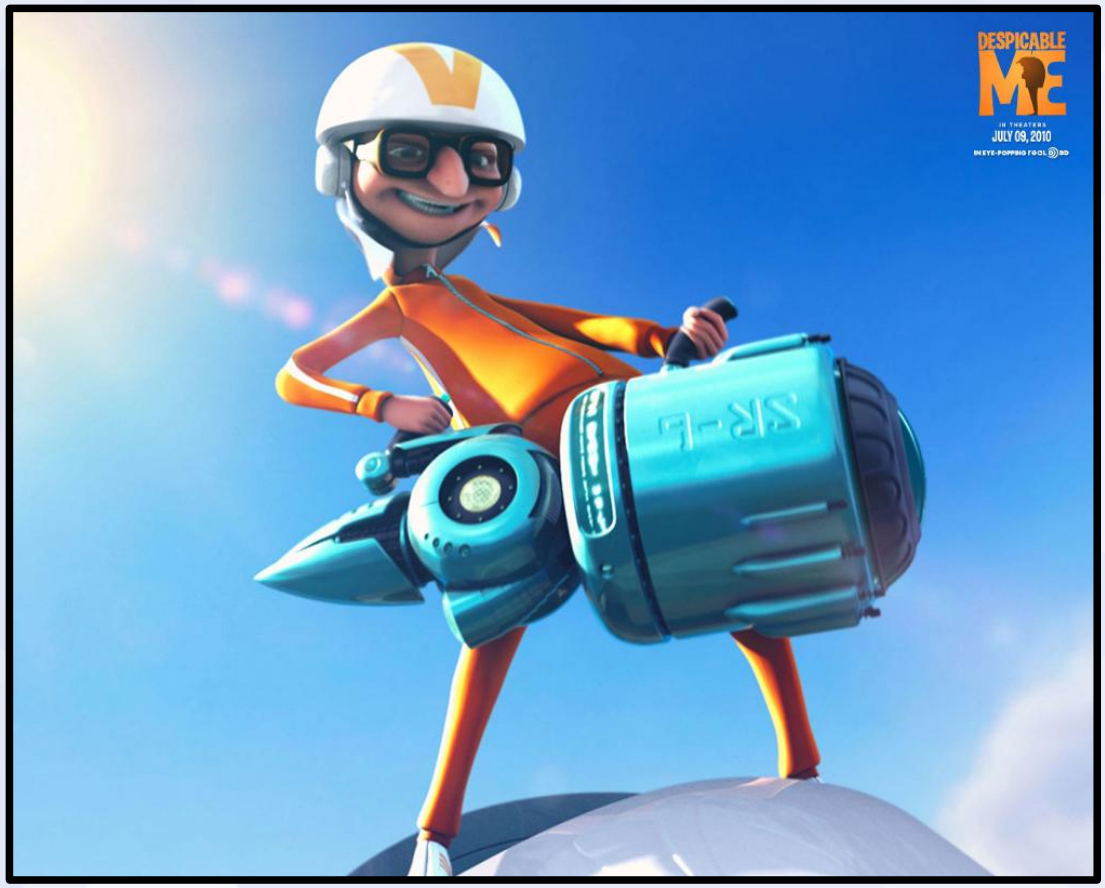
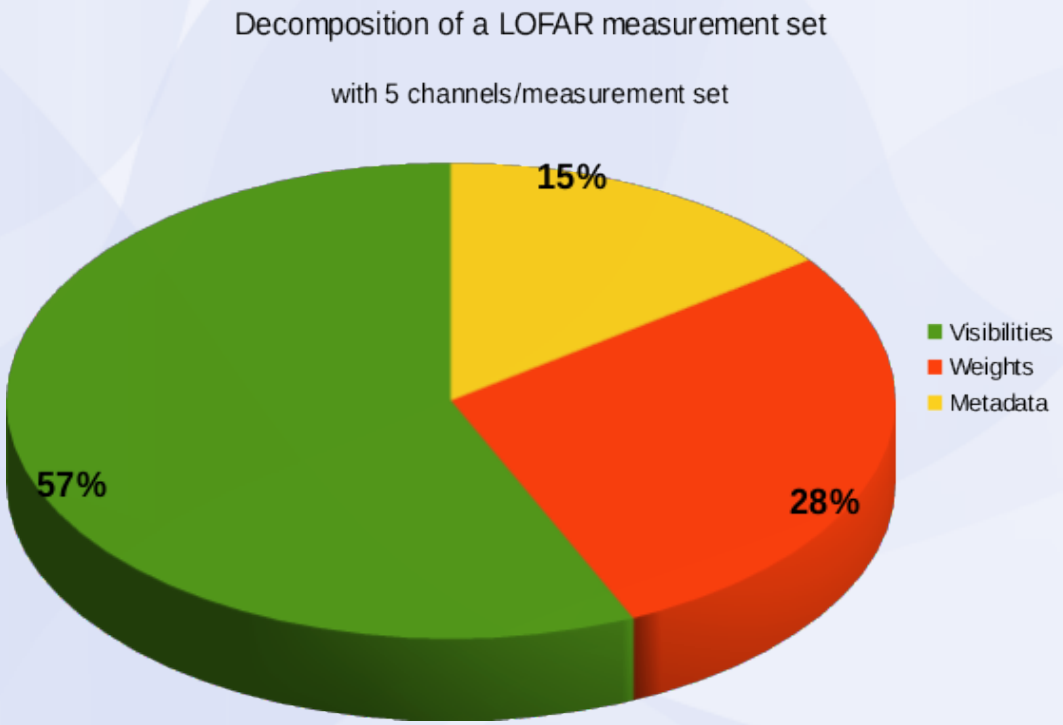
A technique for compressing LOFAR visibilities

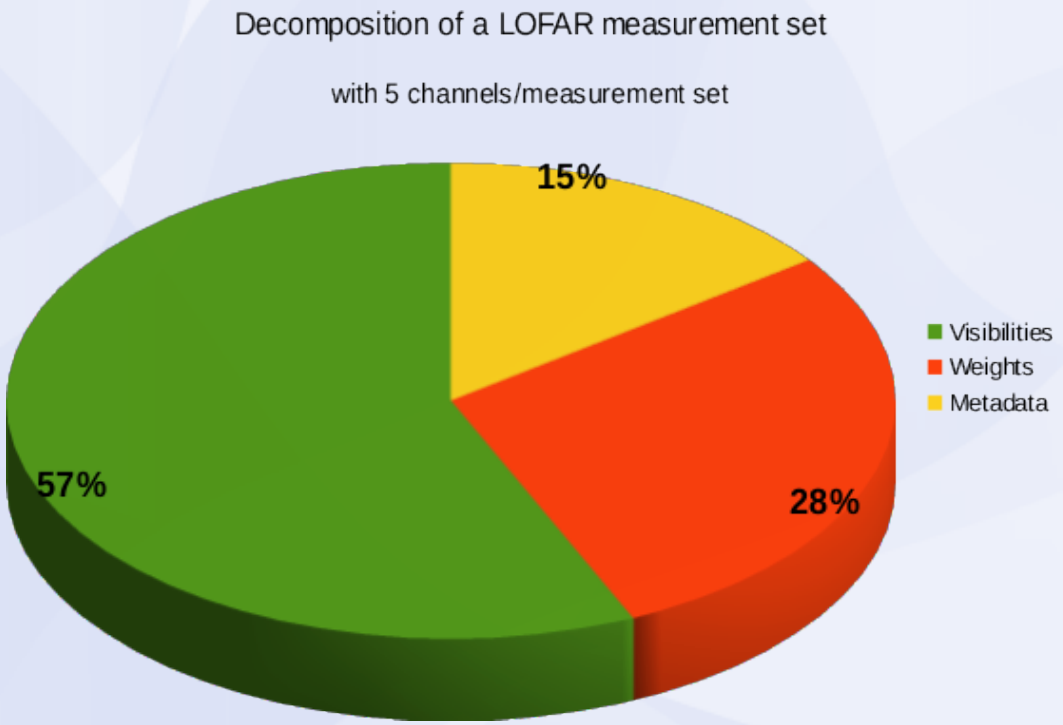


A shrink ray gun for LOFAR data



- Visibilities and weights make up >85% of the size of a measurement set
- Larger nr. channels / ms → Rel. smaller metadata
- Each visibility uses 3 x 32-bit floats

  (real, imaginary, weight)



- Compressing weights is easy:
  just store 1 of the 4 polarizations
- Further quantization possible to compress further

Compression

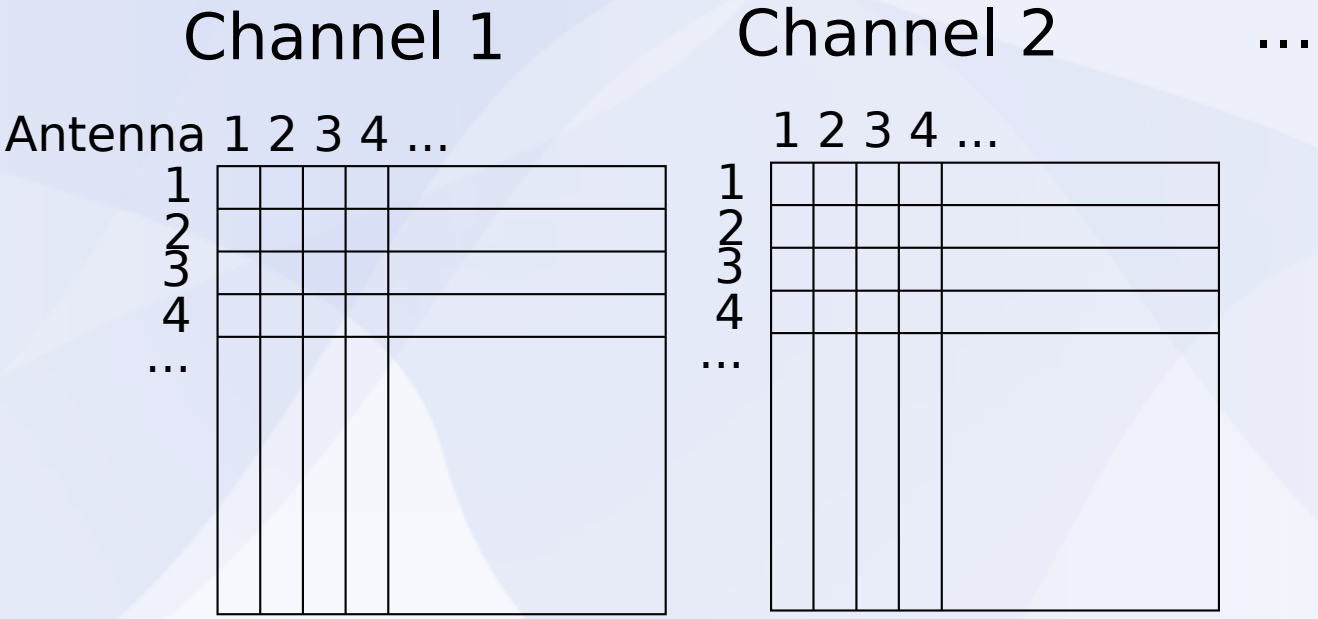
- Compression can be lossless or lossy
- Lossless compression is limited by the randomness of noise
  - At best a reduction from 100% to ~75% of the visibilities
- However, lossy compression needs to be tested carefully
  - E.g. What are the consequences for long time integrations? And for flux levels?

Compression

- I am investigating ***lossy*** compression of visibilities
- Compression factor of ~4 seems possible
- I compress visibilities in 2 steps:
    1) Normalize the visibilities
    2) Use non-linear quantization and bitpacking

Visibility *normalization*

- Group visibilities of the same timestep and polarization
- Result: a cube of #ant x #ant x #channel visibilities:



Visibility *normalization*

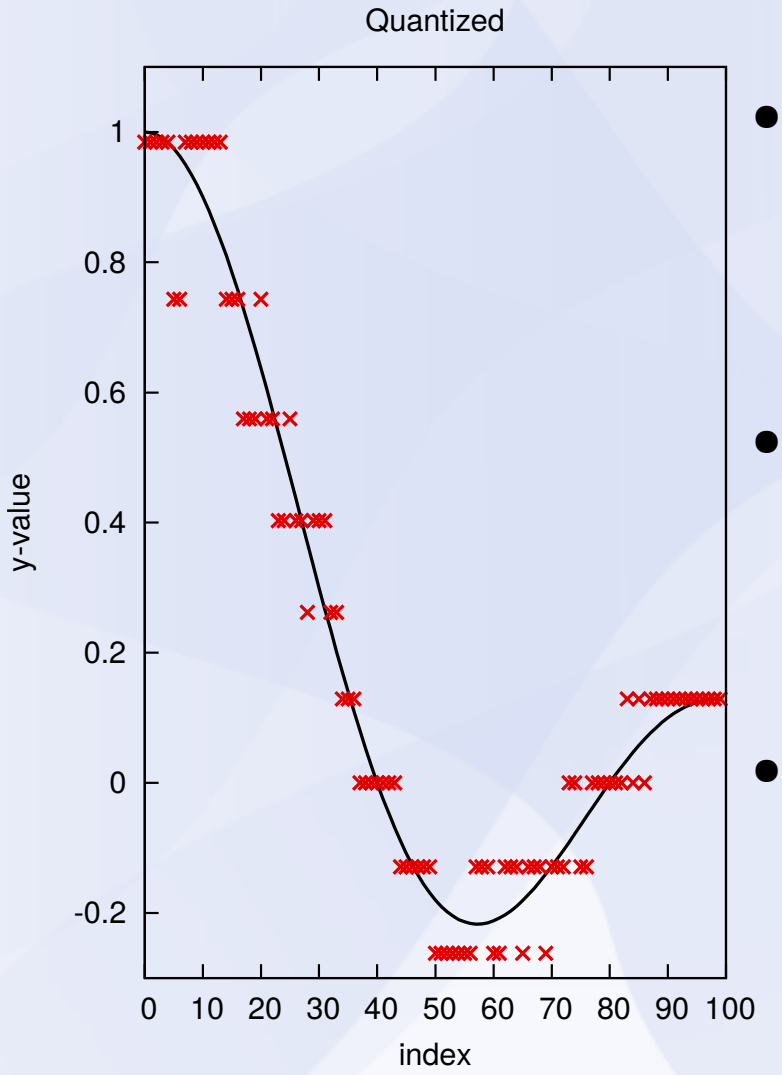
- Find per vis group a factor per antenna and per channel that normalizes the variance
- Antenna factors absorb different antenna noise levels.
- Channel factors absorb bandpass.
- Additionally, make sure highest value in time block can be quantized.

  This is an optimization problem, but it is easy to generate a proximate optimum.
- The (#ant + #channels) factors are stored as floats, along with the quantized values

Visibility *quantization*

- Quantization is “rounding” a value to a nearby quantity that can be represented with fewer bits.
- Normalized visibilities are ~pure Gaussian distributed noise values.
- Optimize the quantization: make smaller errors near 0, because we have many more “small” values

Visibility *quantization*



- Larger values → larger quantization errors
- Avoid bias by “dithering”: by chance select the 2nd closest quantization value
- Comparable with adding uniformly distributed noise

Visibility *quantization*

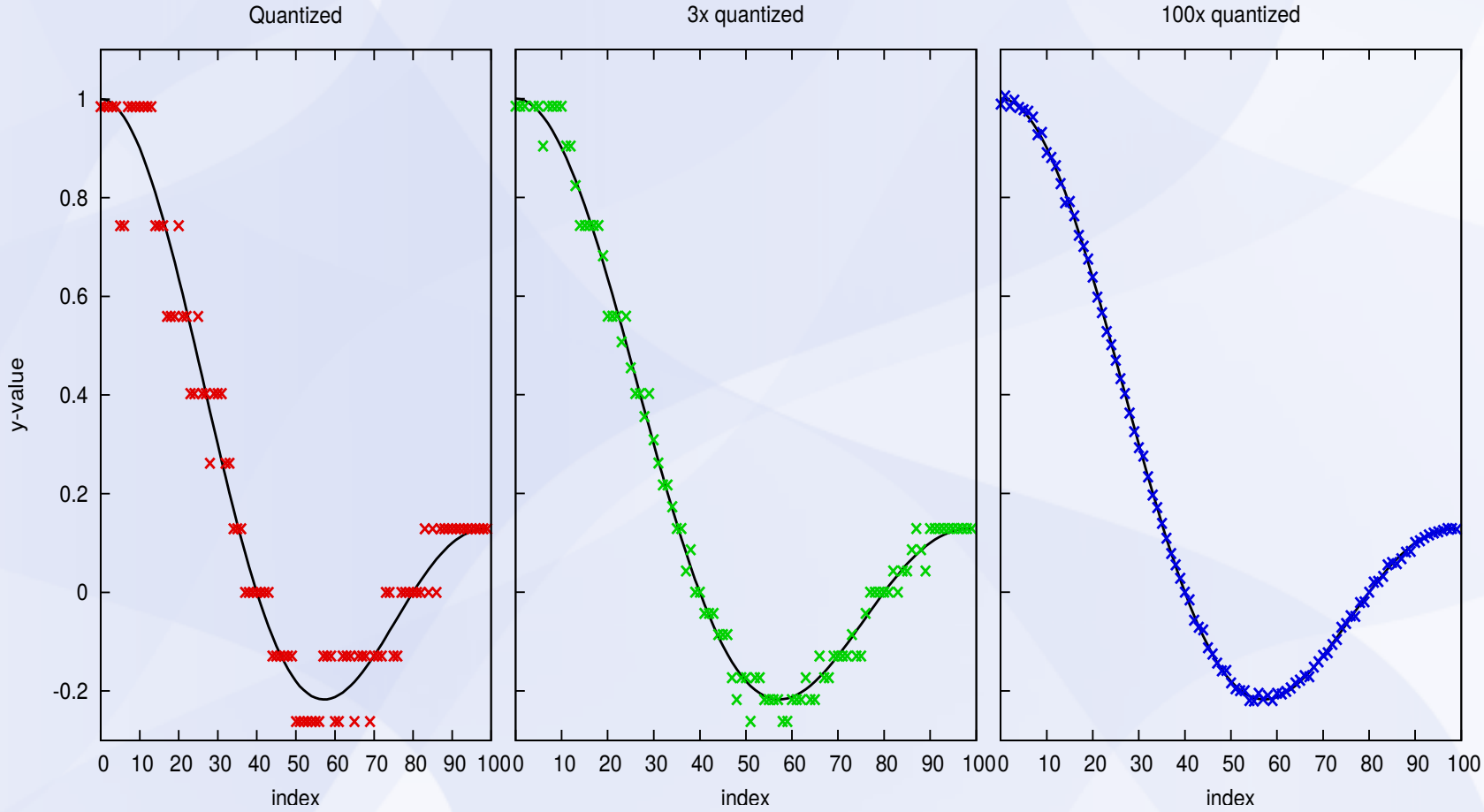
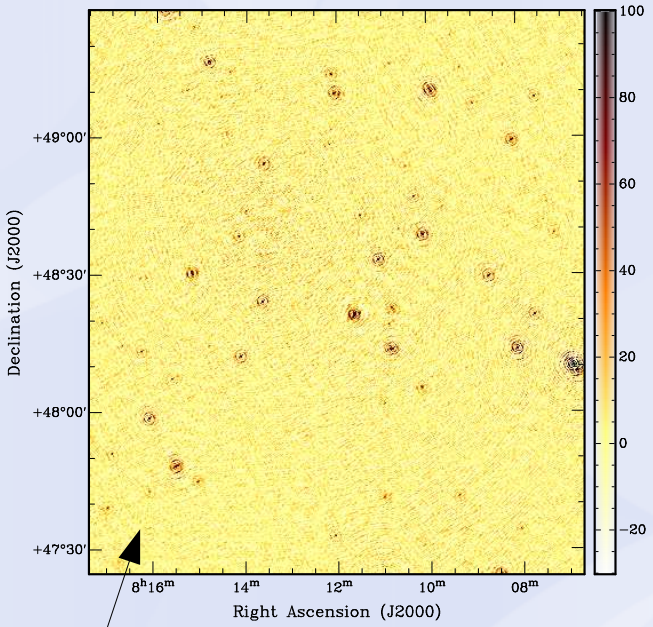
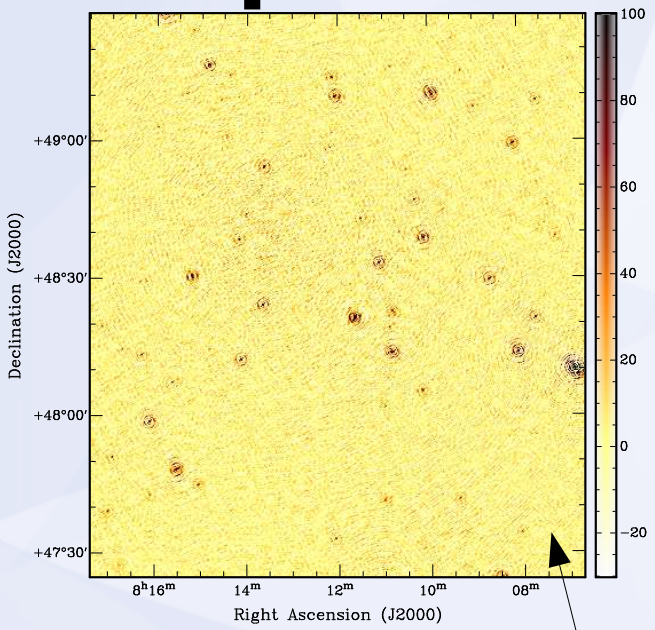


Fig. : A quantization example using the Gaussian-optimized least-squares quantization scheme with dithering to quantize a sinc function. 4-bit quantization and a single scaling factor were used. Left plot: result of encoding and decoding. Because the quantization is optimized for Gaussian distributed values, the quantization steps are smaller near zero. Central and right plot: average of 3 and 100 times encoding and decoding respectively.

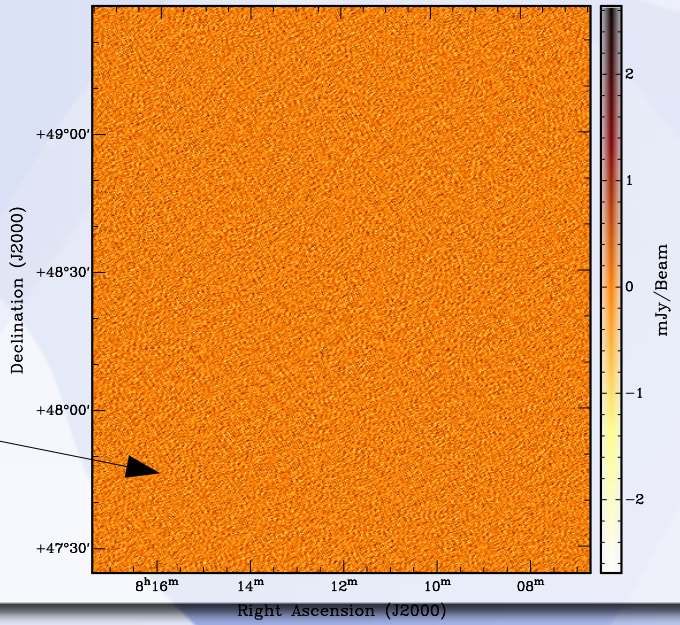
Result: 8-bit compression



Uncompressed



Compressed
to 8 bits
(no visual difference)



Difference
(Unstructured noise)

Rms of 400 microJy

Test set: LOFAR 3c196
4 s / 36 kHz vis resolution
Calibrated after compression

Result: 2 bit (!!) compression

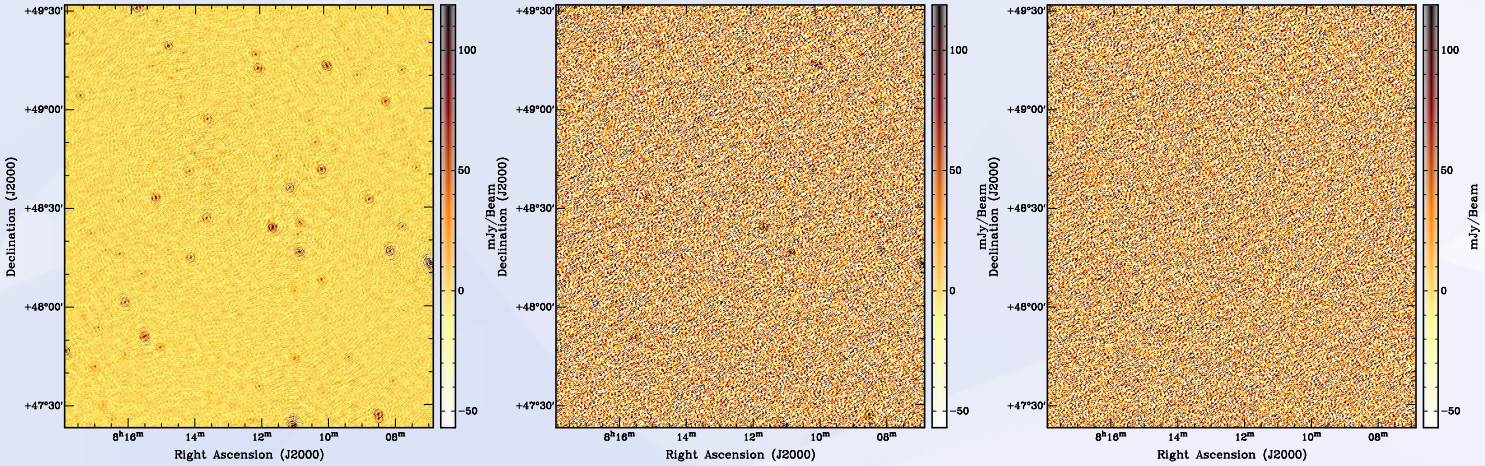
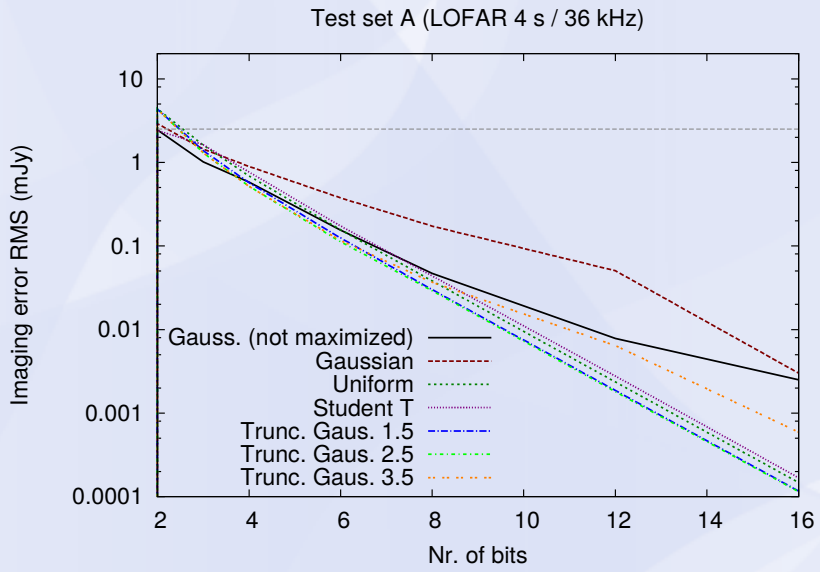
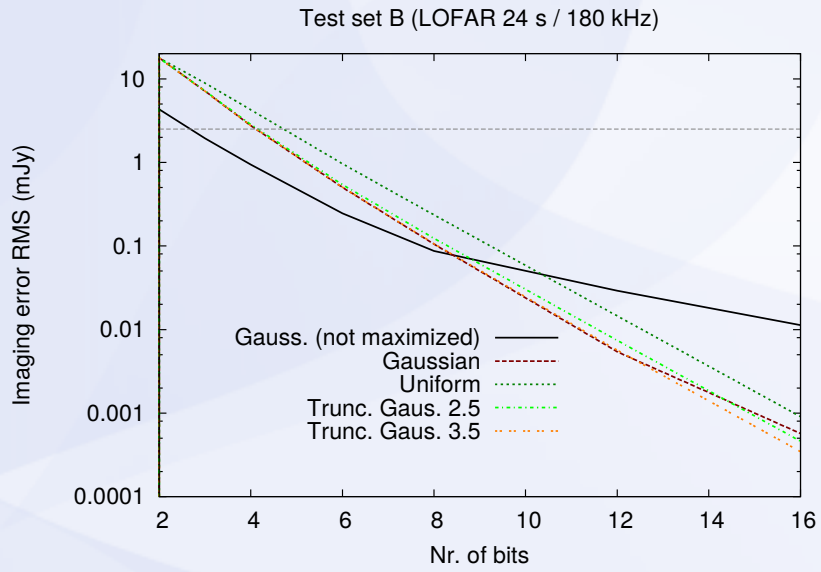


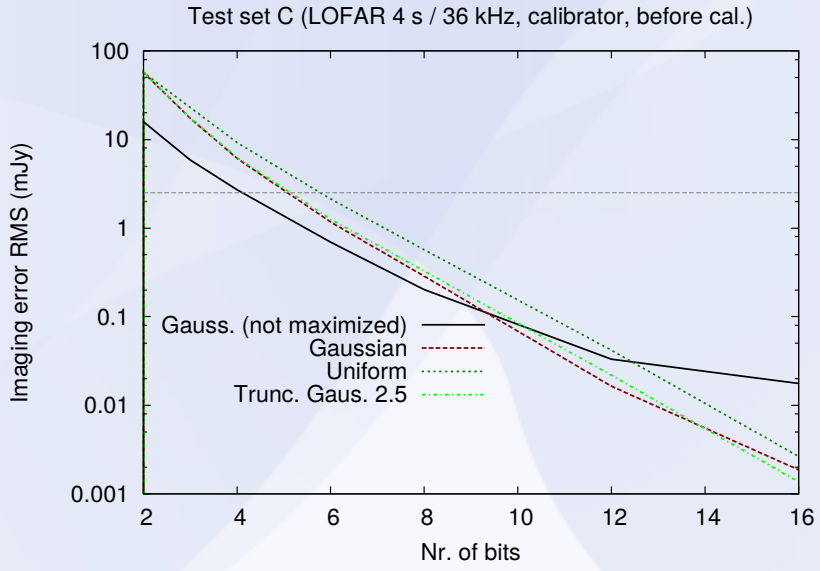
Fig. 5: Demonstration of added 2-bit compression noise using LOFAR test set C. Left image: Results of calibration, 3c196 subtraction and imaging without compression. Centre image: Same, but before processing the visibilities were compressed using the 2-bit quantization scheme (16× compression) with the maximized truncated Gaussian distribution, truncated at 2.5σ. Right image: Difference between left and centre images. While the added compression noise dominates the noise in the image, the compression has not affected the sources and the added noise is unstructured.

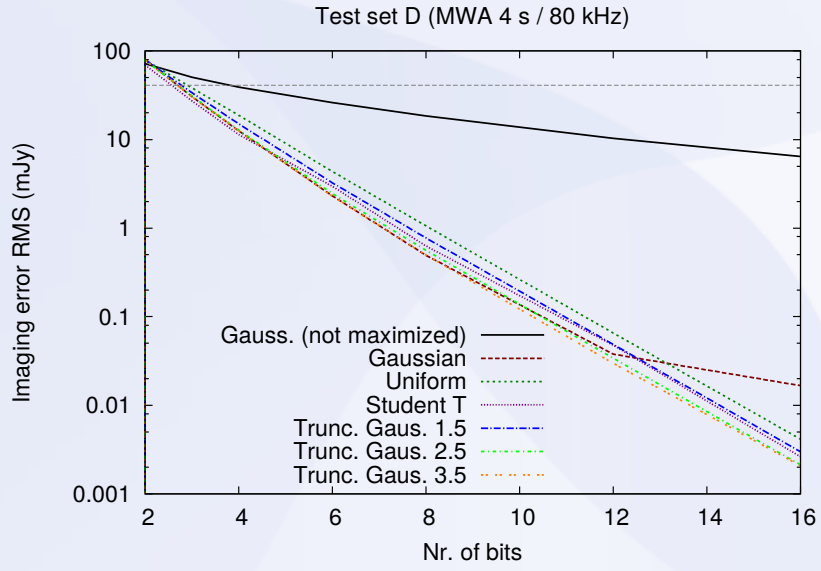
- 2-bit compression: maybe not a good idea
- ...but possible for very high time/freq res
- Added noise is still random unstructured noise, sources have the right flux.

Results



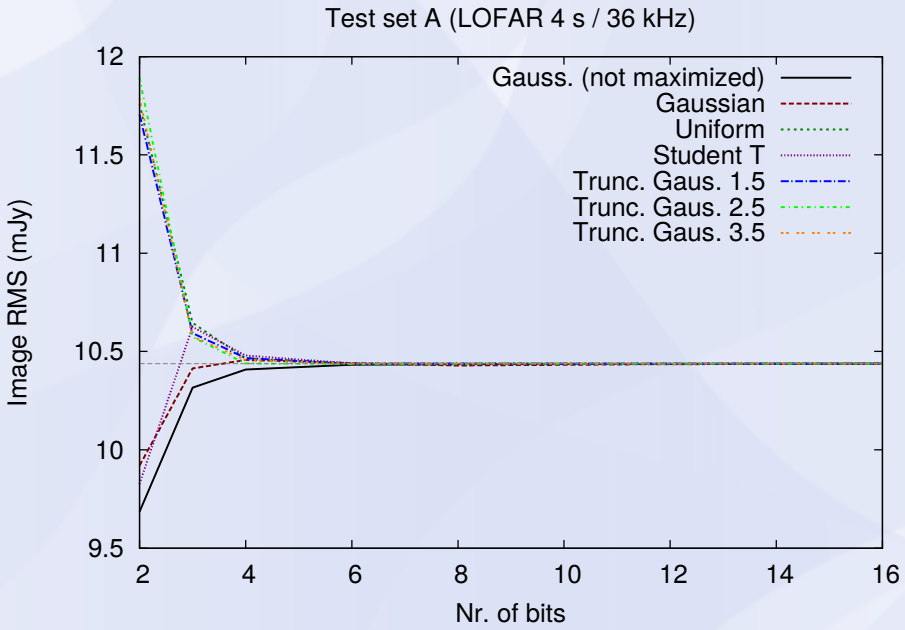


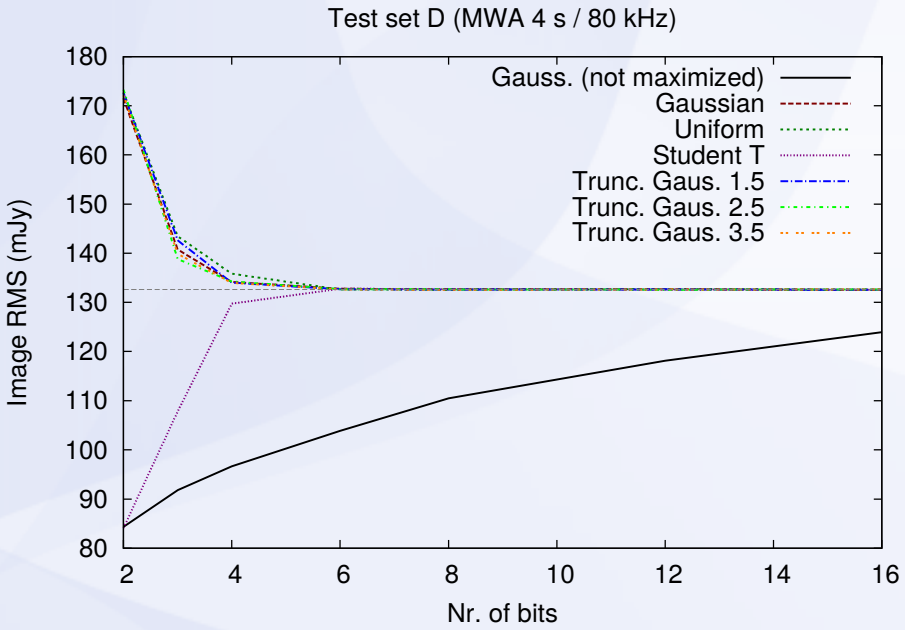




Gray dashed line: Stokes V noise level

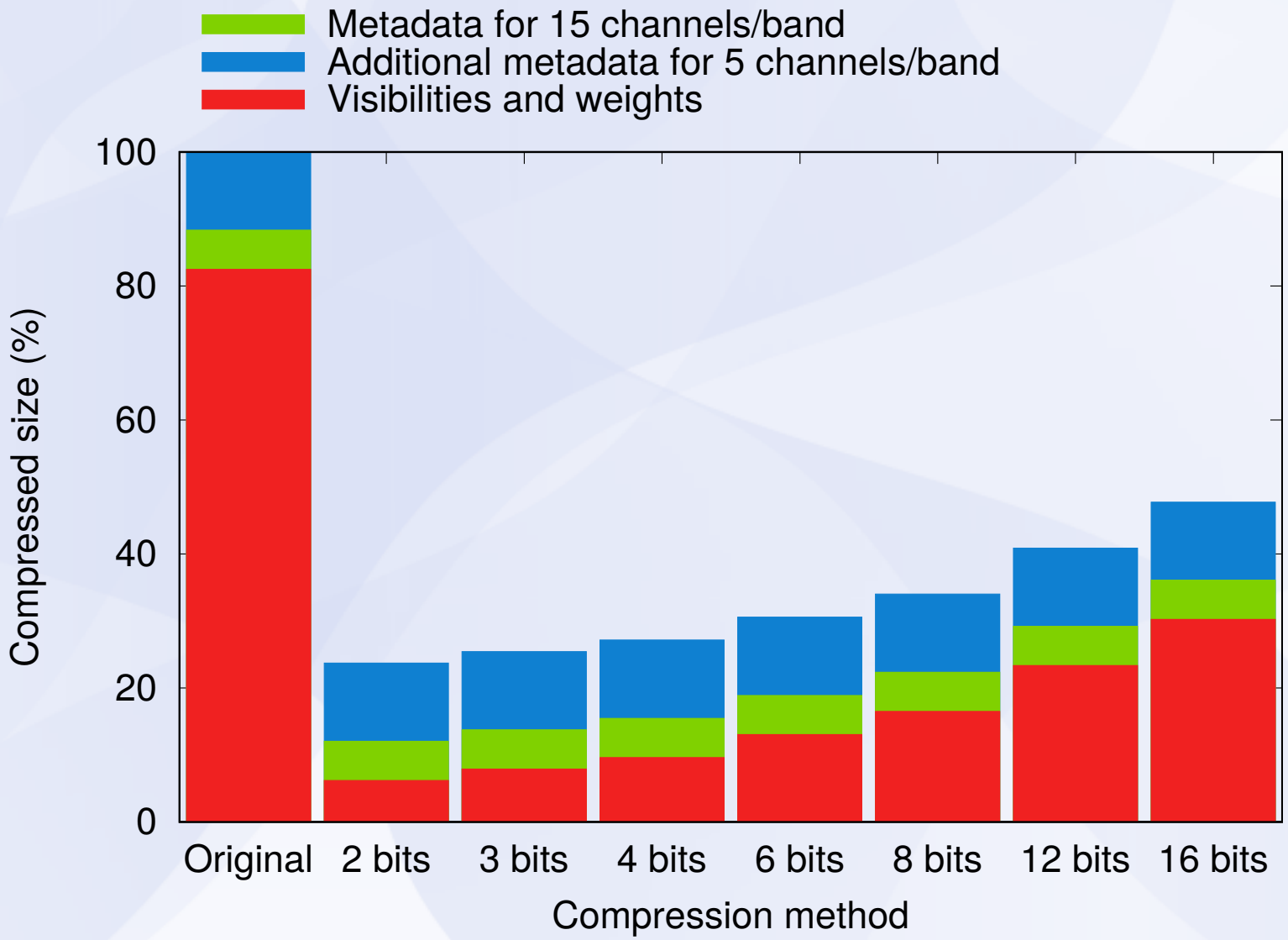
Results





- Approx. no added noise ≥ 6 bits
- data=Gaussian not always best assumption
- Assuming a truncated distribution is better (note that this does not imply the data is truncated)

Compression factor



Implementation

- Casacore has a transparent system allowing “storage managers”
- I’ve implemented a storage manager doing compression on the fly
- Once a storage manager of an MS is changed, it is smaller, but still compatible with all tools (casa, ndppp, wsclean, ...)

Implementation

- The storage manager is called

  The dynamical statistical compression storage manager

Implementation

- The storage manager is called

  The **dy**namical **s**tatistical **co**mpression storage manager

  so in short

  The Dysco storage manager (dyscostman)

Results: computational performance

- Decompression is fast
  - Single table lookup
  - IO is the bottleneck
  - reading+decompression is faster than reading the full data
- Compression is slower
  - Binary dictionary search, multi-threaded
  - On spinning disks, faster than full write
  - On fast SSD, can be slightly slower

Applications

- Transparent compression with a factor 4 possible for LOFAR observations
- Best to apply on **noisy** data
  - LOFAR data with 36 kHz, 4 s seems always noisy enough for 4x compression
- Best to apply *after flagging* to remove outliers that add extra noise

  Raw data → NDPPP → Compressed set → calibrate
- Fine for uncorrected, corrected and model data, as long as resolution is high. Uncorrected makes most sense.
- Auto-correlations are currently not preserved

Any questions?