Commander and Computational Aspects

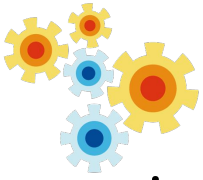
Mathew Galloway

BeyondPlanck online release conference, November 18-20, 2020

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776282

Commander3

- The BeyondPlanck results were generated using the Commander3 codebase
- Based on the legacy Commander code that was used for component separation for Planck
- Written in object oriented Fortran90 (Approved by Grey-Beards everywhere!)
- Implements the iterative Gibbs-sampling of specific data models, for LFI this was:



Commander

$$d_{j,t} = g_{j,t} \mathsf{P}_{tp,j} \left[\mathsf{B}^{\mathrm{symm}}_{pp',j} \sum_c \mathsf{M}_{cj}(\beta_{p'}, \Delta^j_{\mathrm{bp}}) a^c_{p'} + \mathsf{B}^{\mathrm{asymm}}_{j,t} \left(\boldsymbol{s}^{\mathrm{orb}}_j + \boldsymbol{s}^{\mathrm{fsl}}_t \right) \right] + n^{\mathrm{corr}}_{j,t} + n^{\mathrm{w}}_{j,t}.$$

- ~45000 lines of code, ~6000 for TOD processing, ~14000 in component separation, ~25000 in auxiliary modules

Code Availability + Advertising

- The Code is available under the GPL3 license here:
  https://github.com/Cosmoglobe/Commander
- Contains all the analysis code as well as data file generation scripts, examples of adding new data sets, etc.
- A data download tool is also available to make it easier to get started, the code installs with cmake
- Documentation and support can be found on our webpage:
  https://beyondplanck.science/
- The code is still under active development, and will be used in the future for datasets such as HFI, WMAP, LiteBIRD, Spider, DIRBE, FIRAS...
- Hands on code and data tutorial will he held next week Monday + Tuesday (Nov 23 + 24). Please join!

Algorithm Overview

Commander3 pseudo-code:

$$
\begin{aligned}
\boldsymbol{g} &\leftarrow P(\boldsymbol{g} \mid \boldsymbol{d}, \xi_n, \Delta_{\mathrm{bp}}, \boldsymbol{a}, \beta, C_\ell) \\
\boldsymbol{n}_{\mathrm{corr}} &\leftarrow P(\boldsymbol{n}_{\mathrm{corr}} \mid \boldsymbol{d}, \boldsymbol{g}, \xi_n, \Delta_{\mathrm{bp}}, \boldsymbol{a}, \beta, C_\ell) \\
\xi_n &\leftarrow P(\xi_n \mid \boldsymbol{d}, \boldsymbol{g}, \boldsymbol{n}_{\mathrm{corr}}, \Delta_{\mathrm{bp}}, \boldsymbol{a}, \beta, C_\ell) \\
\Delta_{\mathrm{bp}} &\leftarrow P(\Delta_{\mathrm{bp}} \mid \boldsymbol{d}, \boldsymbol{g}, \boldsymbol{n}_{\mathrm{corr}}, \xi_n, \boldsymbol{a}, \beta, C_\ell) \\
\beta &\leftarrow P(\beta \mid \boldsymbol{d}, \boldsymbol{g}, \boldsymbol{n}_{\mathrm{corr}}, \xi_n, \Delta_{\mathrm{bp}}, C_\ell) \\
\boldsymbol{a} &\leftarrow P(\boldsymbol{a} \mid \boldsymbol{d}, \boldsymbol{g}, \boldsymbol{n}_{\mathrm{corr}}, \xi_n, \Delta_{\mathrm{bp}}, \beta, C_\ell) \\
C_\ell &\leftarrow P(C_\ell \mid \boldsymbol{d}, \boldsymbol{g}, \boldsymbol{n}_{\mathrm{corr}}, \xi_n, \Delta_{\mathrm{bp}}, \boldsymbol{a}, \beta),
\end{aligned}
$$

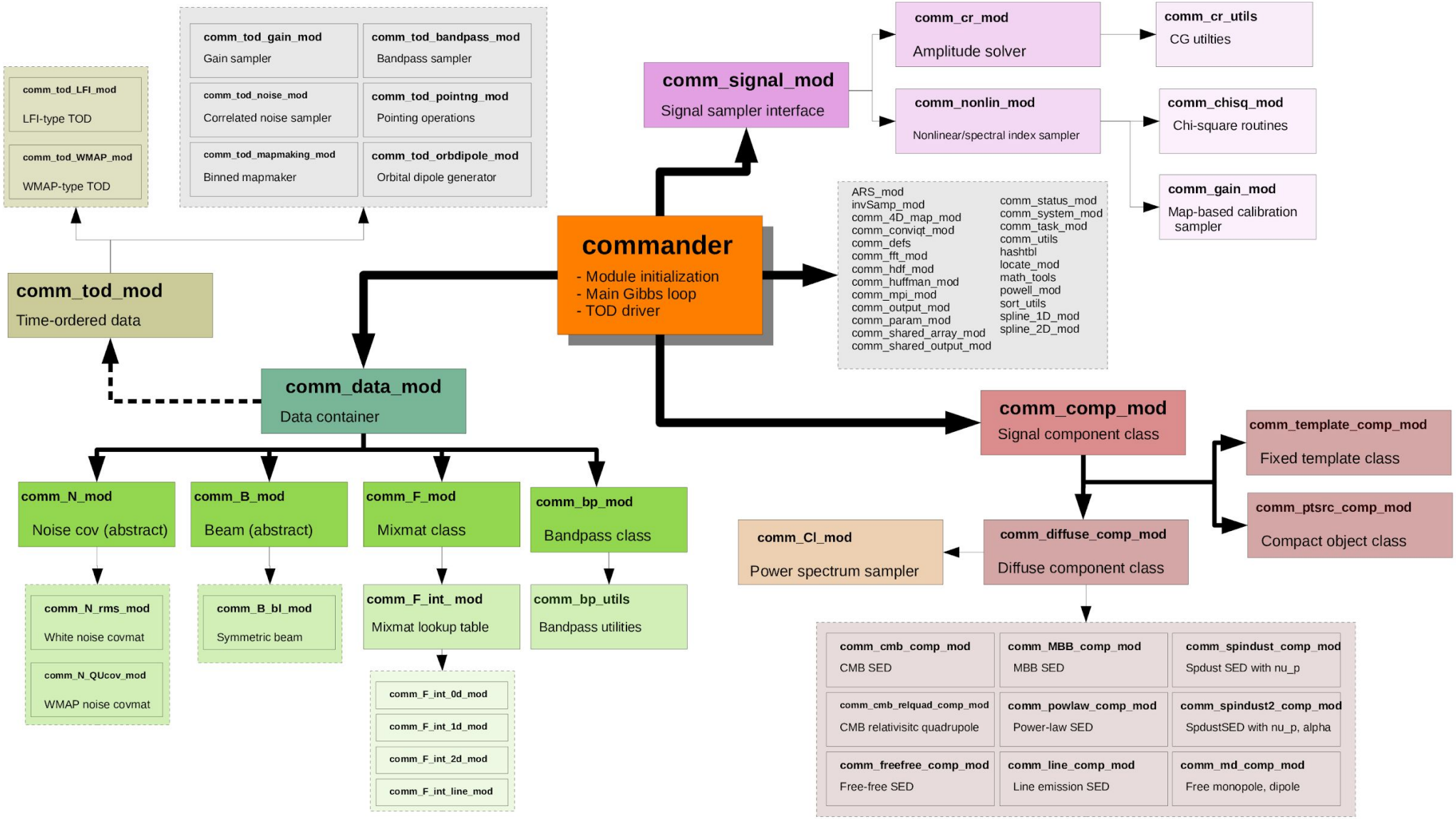
i) Read parameter file
ii) Initialize data sets; store in global array called `data`
iii) Initialize model components; store in global linked list called `compList`
iv) Initialize stochastic parameters

for $i = 1, N_{\mathrm{Gibbs}}$ do

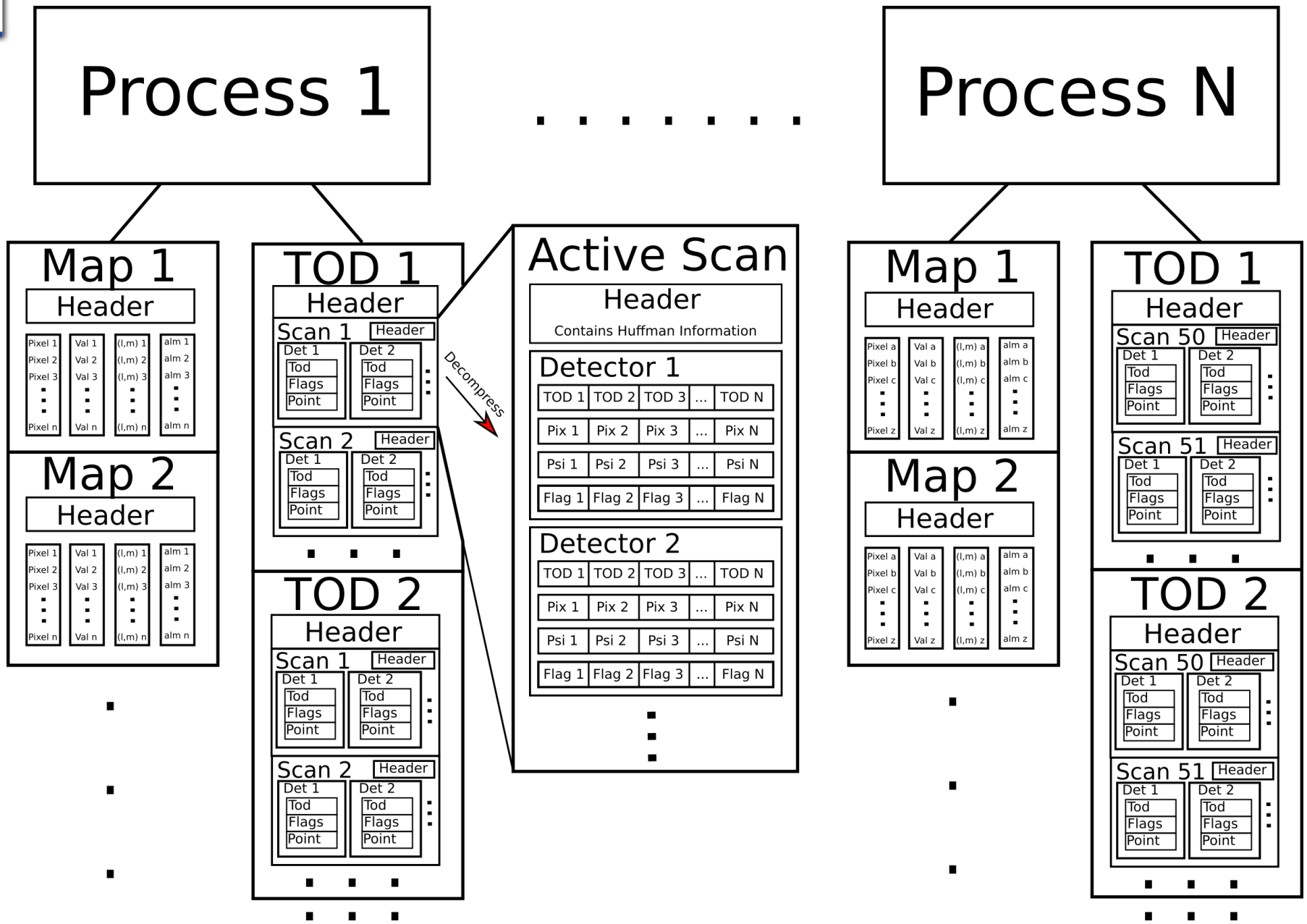
1) Sample TOD parameters; make frequency maps
   a) Sample gain
   b) Sample correlated noise
   c) Clean and calibrate TOD
   d) Sample bandpass corrections
   e) Bin TOD into maps
2) Sample astrophysical amplitude parameters
3) Sample angular power spectra
4) Output current parameter state to disk
5) Sample astrophysical spectral parameters
6) Sample global instrument parameters for non-TOD data sets, e.g., calibration, bandpass corrections

- Gibbs Sampling: Iteratively sample each parameter while holding the others fixed
- Easily suggests Object Oriented programming solutions, each sampling step can be one or more classes

Block Diagram



Memory Management



Data Formats

- All data is stored in RAM during pipeline execution, so the data discussed here is just inputs and outputs
- TODs are stored in HDF5 files, using a custom compression scheme (coming up)
- HDF5 has faster IO than fits, and supports more data customization (almost like a directory structure)
- Skymaps are stored in standard .fits format
- All of the samples are stored in a chains.h5 file, which contains the complete history of all the parameter values throughout the entire sampling run
- We have written a tool for Commander3 Post Processing (c3pp) that reads and makes plots from the output chains files, available here:

  https://github.com/Cosmoglobe/c3pp

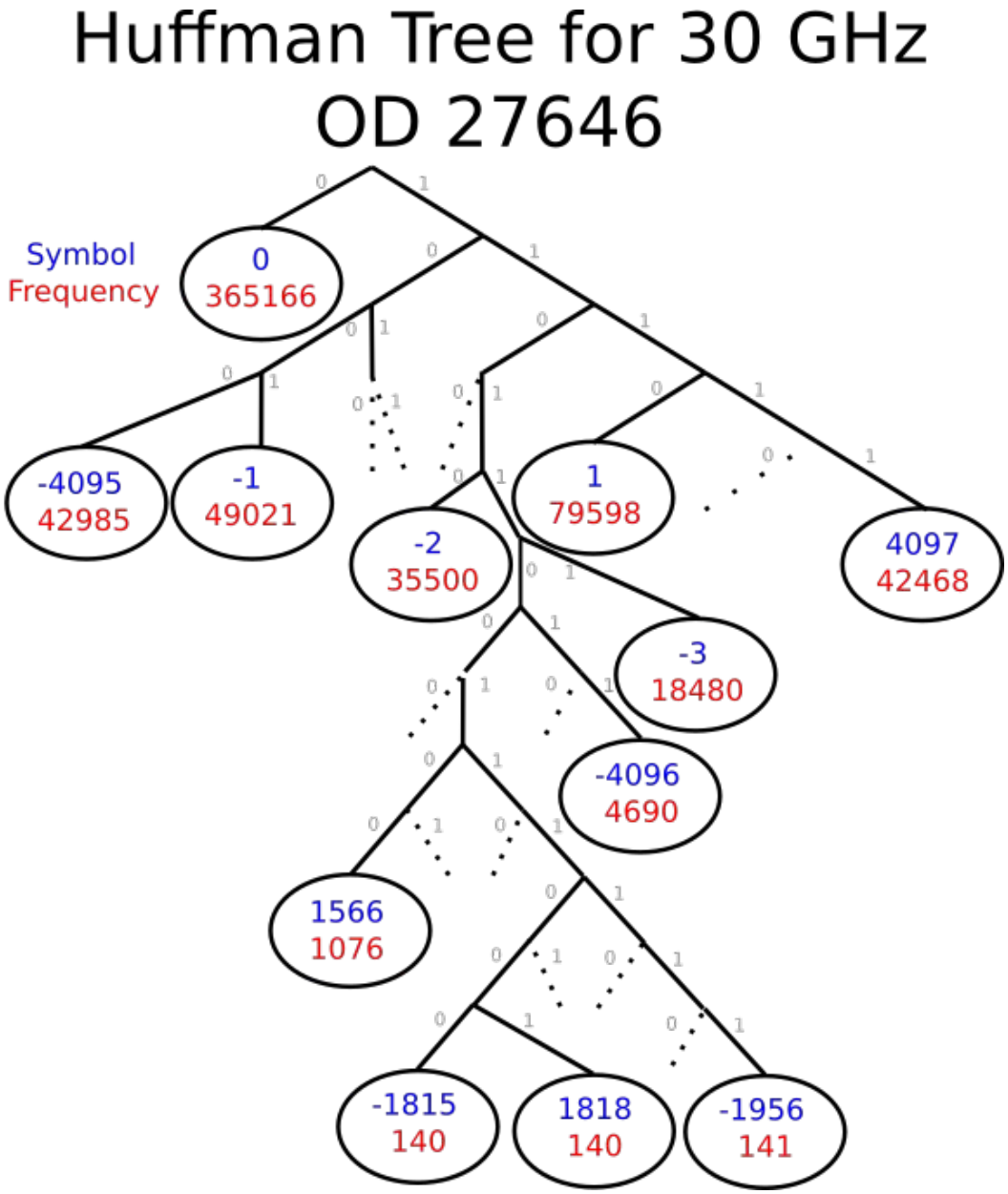
In-Memory Data Compression

Item	Explanation	30 GHz		44 GHz		70 GHz		Total Ratio
		Raw (MB)	Compressed (MB)	Raw (MB)	Compressed (MB)	Raw (MB)	Compressed (MB)	
TOD	Dtype reduction	90.3	45.2	193.9	97.5	656.2	328.1	0.5
Pixel number	Healpix, Differencing + Huffman Coding	180.5	9.8	387.8	17.5	1312.4	69.7	0.05
Psi	Discretization, Differencing + Huffman Coding	90.3	5.2	193.9	9.6	656.2	24.8	0.04
Flag	Differencing + Huffman Coding	45.1	2.7	96.9	5.9	328.1	10.3	0.03
Time	Runtime Extrapolation	135.4	0.0003	290.8	0.0003	984.3	0.0003	6.3e-7
Scalars	Included from RIMO	0	0.004	0	0.007	0	0.013	∞
Huffman Indexes		0	1.37	0	2.1	0	2.6	∞

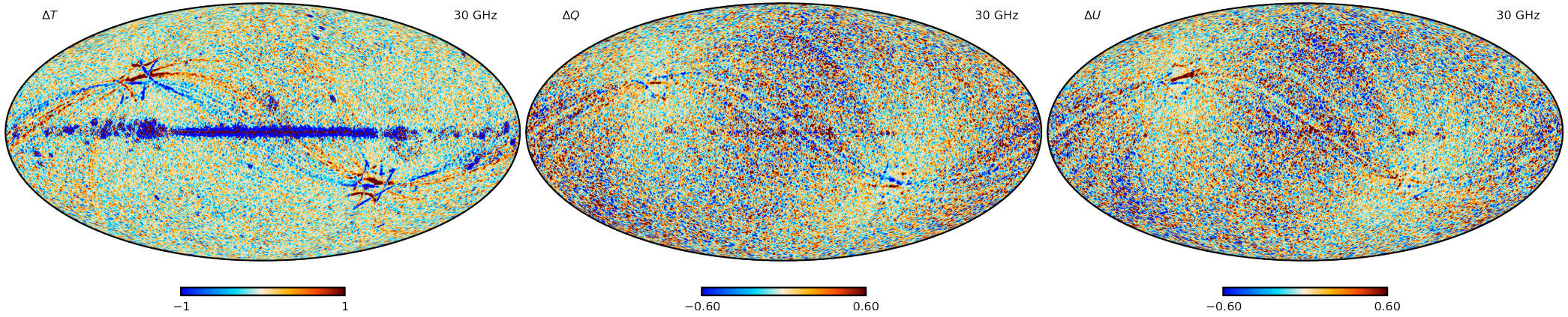
- In general, we discretize because fixed precision numbers like ints are much easier to compress than floats
- Pairwise differencing of most datasets like flags and pointing really reduces complexity. Most differences are ± 1, 0
- Once we have these differenced timestreams, they are very easy to further compress with Huffman Coding

Huffman Coding

- Technique for losslessly compressing datasets into binary representation
- High-frequency numbers are given short representations
- Can be visualized as a tree structure
- This example shows a tree with a total array size of 861276, 42% of these are 0 after pairwise differencing
- Full tree contains 670 unique symbols and their compressed representations



Compression Effects



- Pre-compressing the pointing and digitizing the polarization angle does lose some information
- We use nside=512 and npsi=4096 for LFI 30 GHz
- Performed a second run with nside=1024 and npsi=8192 at 30 GHz
- Difference maps show that the effects of the compression are noise of a negligible amplitude

Benchmarking

Item	30 GHz	44 GHz	70 GHz	Sum
Data volume				
Uncompressed data volume	761 GB	1 633 GB	5 522 GB	7 915 GB
Compressed data volume/RAM requirements	86 GB	178 GB	597 GB	**861 GB**
Processing time (cost per run)				
TOD initialization/IO time	176 sec	288 sec	753 sec	1217 sec
Other initialization				663 sec
Total initialization				**1880 sec**
Gibbs sampling steps (cost per sample)				
Data decompression	36 sec	105 sec	252 sec	393 sec
TOD projection (P operation)	33 sec	49 sec	248 sec	330 sec
Sidelobe evaluation ($\boldsymbol{s}_{\rm sl}$)	58 sec	85 sec	337 sec	480 sec
Orbital dipole ($\boldsymbol{s}_{\rm orb}$)	45 sec	61 sec	343 sec	449 sec
Gain sampling (g)	13 sec	10 sec	71 sec	94 sec
Correlated noise sampling ($\boldsymbol{n}_{\rm corr}$)	355 sec	390 sec	2393 sec	3138 sec
TOD binning (P^t operation)	22 sec	34 sec	442 sec	498 sec
Loss due to poor load-balancing	62 sec	305 sec	135 sec	502 sec
Sum of other TOD steps	32 sec	135 sec	139 sec	306 sec
TOD processing cost per sample	656 sec	1074 sec	4666 sec	**6396 sec**
Amplitude sampling, $P(\boldsymbol{a} \mid \boldsymbol{d}, \omega \setminus \boldsymbol{a})$				527 sec
Spectral index sampling, $P(\beta \mid \boldsymbol{d}, \omega \setminus \beta)$				1080 sec
Other steps				149 sec
Total cost per sample				**8168 sec**

Applications to Future Projects

- Modularity lets us add new datasets “easily”
- Data structures can be mostly reused for new projects, even WMAP wasn’t a huge re-write (see Watts et al. + talk tomorrow)
- Main limitation is data volume, as RAM increases we can add larger projects
- Parallelizing over many nodes is slow, Gibbs sampling requires MANY synchronization steps
- Compressing the sky signal would help reduce data volume significantly (currently ~75%), but would be easier to do to ADC compressed data
- This was avoided for LFI because it wasn’t necessary and it might have more of an effect on precision - needs more testing
- Still many existing datasets we can integrate on the current generation of hardware

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776282

- "*BeyondPlanck*"
  - COMPET-4 program
  - PI: Hans Kristian Eriksen
  - Grant no.: 776282
  - Period: Mar 2018 to Nov 2020

Collaborating projects:

- "*bits2cosmology*"
  - ERC Consolidator Grant
  - PI: Hans Kristian Eriksen
  - Grant no: 772 253
  - Period: April 2018 to March 2023
- "*Cosmoglobe*"
  - ERC Consolidator Grant
  - PI: Ingunn Wehus
  - Grant no: 819 478
  - Period: June 2019 to May 2024

Questions?

Beyond PLANCK

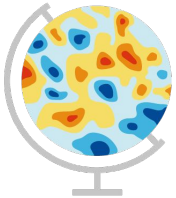
Commander

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

planetek hellas

JPL

HAVERFORD COLLEGE



Cosmoglobe