

Improvements in ASTRON's Interferometric Data Processing Software: a Community Update

Mick Veldhuis

On behalf of the pipeline software development team @ ASTRON



What Are We Developing & Maintaining?

ASTRON

Preprocessing

LINC

Rapthor

PILOT

STrIK

Python

CLI

DP3

*Streaming framework +
calibration*

C++

CLI

WSClean

Imaging

CLI

Python

Spinifex

Ionospheric tooling

CLI

Python

LSMTool

Sky-model editing

CLI

Python

PyBDSF

Source finder

CLI

Python

LoSoTo

Edit calibration solutions

CLI

Python

TraP

*Image-plane transient
search tool*

C++

CLI

Python

AOFlagger

Flagging

C++

Python

EveryBeam

Beam model / A-terms

C++

Python

IDG

*(De)gridding +
calibration*

C++

Python

Radler

Deconvolution

C++

Python

Casacore

C++

Dysco

Compression

<enhancements>

Improved DDECal Solver Constraints

- DDECal is DP3's direction dependent calibration step, which contains a wide range of different algorithms and constraints.
- The last year we introduced a bunch of improvements to the DDECal solvers and constraints, in particular we added the following constraints:
 - Variable smoothness constraints for different stations.
 - Variable solutions (time) intervals for different stations.
 - Specialised solvers for:
 - Differential faraday rotation.
 - Polarisation leakage terms.
 - Delays (clock terms) that simultaneously fit **TEC and delay** or **TEC, phase, and delay**.

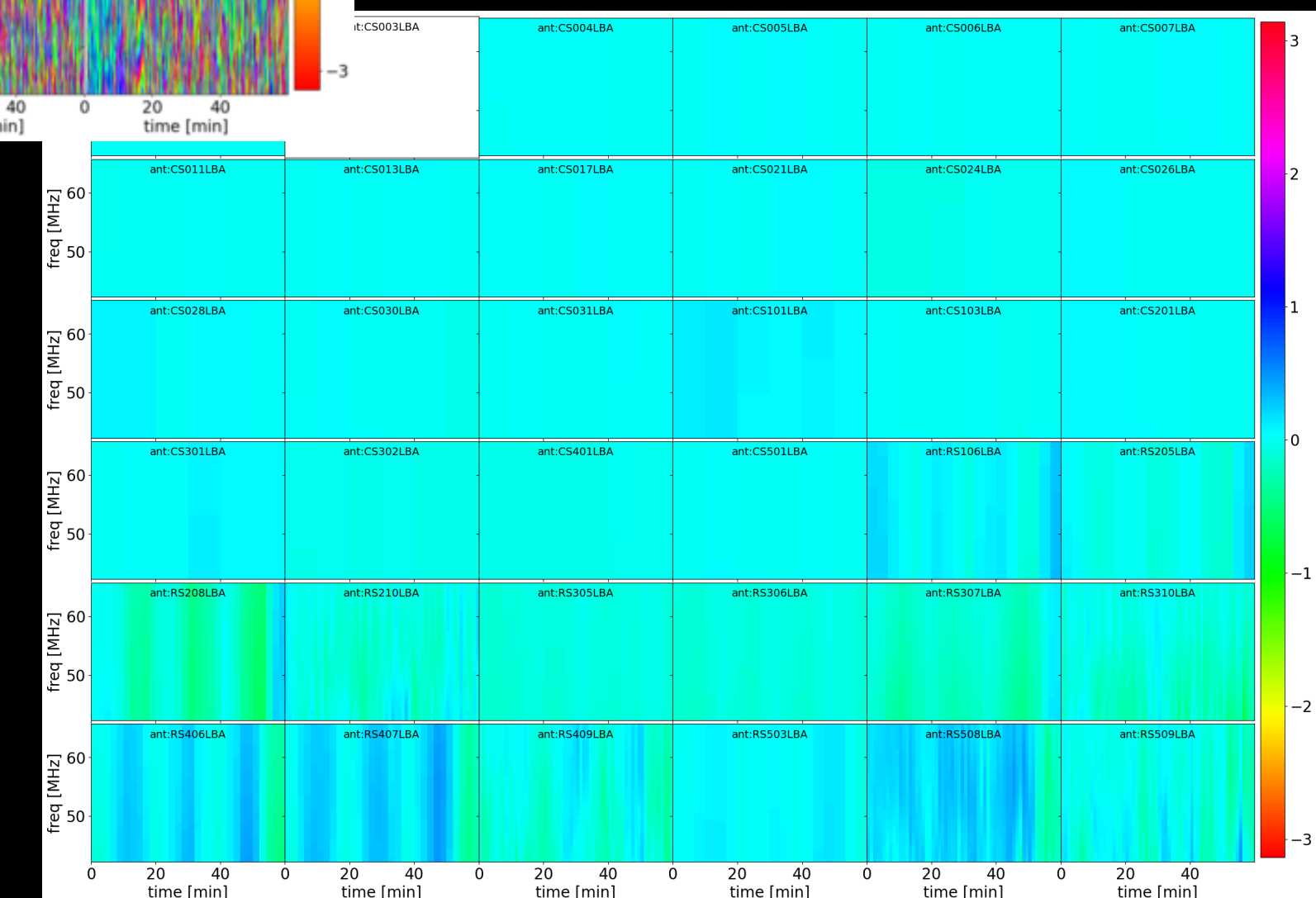


Scalar phase solve

CS have slower solution intervals and stronger smoothness leading to lower noise in the solutions

Variable time + smoothness constraints (works with all modes)

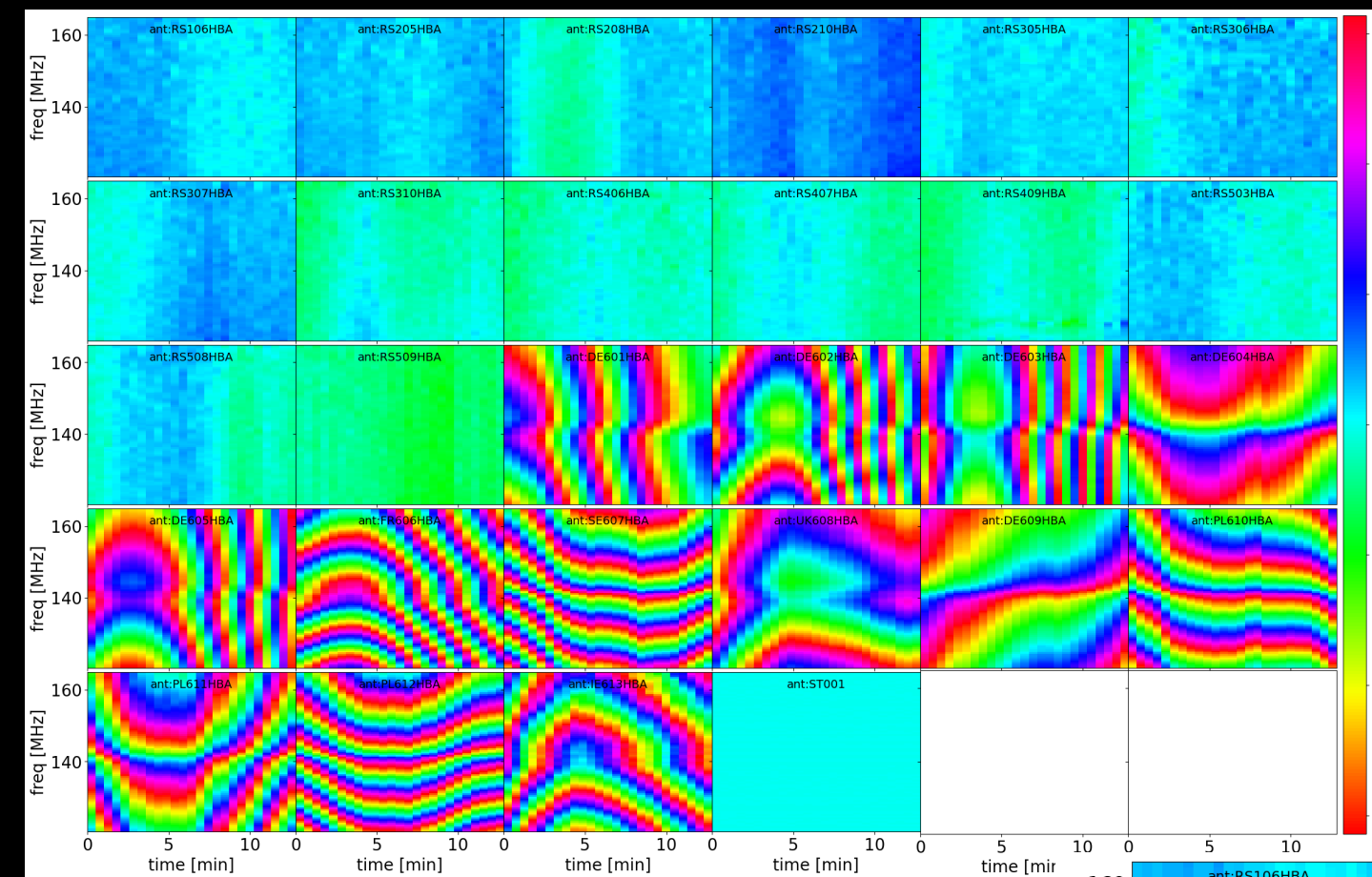
Images provided by Henrik Edler



Faraday rotation solve

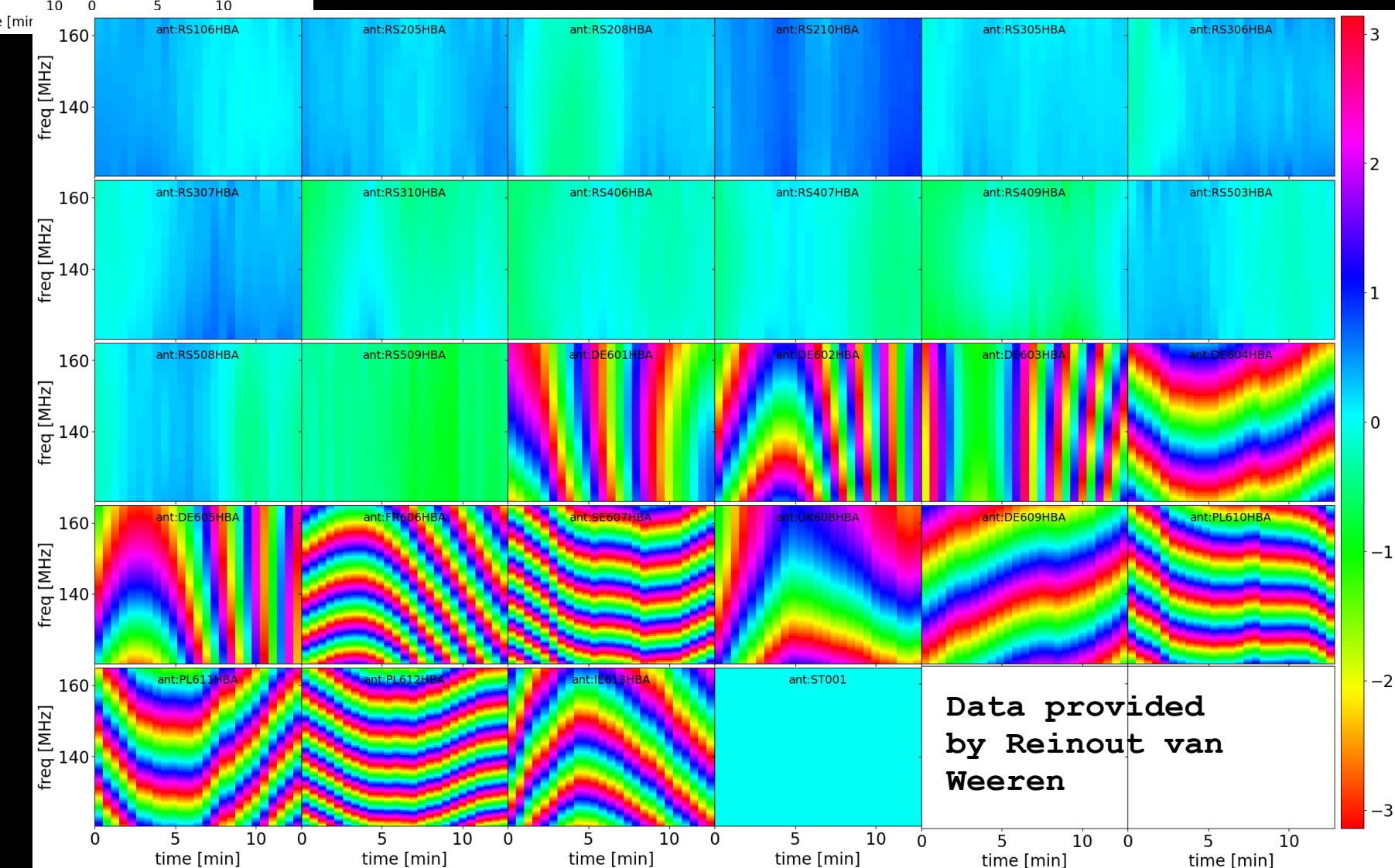
Improved DDECAL Solver Constraints

- DDECAL is DP3's direction dependent calibration step, which contains a wide range of different algorithms and constraints.
- The last year we introduced a bunch of improvements to the DDECAL solvers and constraints, in particular we added the following constraints:
 - Variable smoothness constraints for different stations.
 - Variable solutions (time) intervals for different stations.
 - Specialised solvers for:
 - Differential faraday rotation.
 - Polarisation leakage terms.
 - Delays (clock terms) that simultaneously fit **TEC and delay** or **TEC, phase, and delay**.



Initial solutions

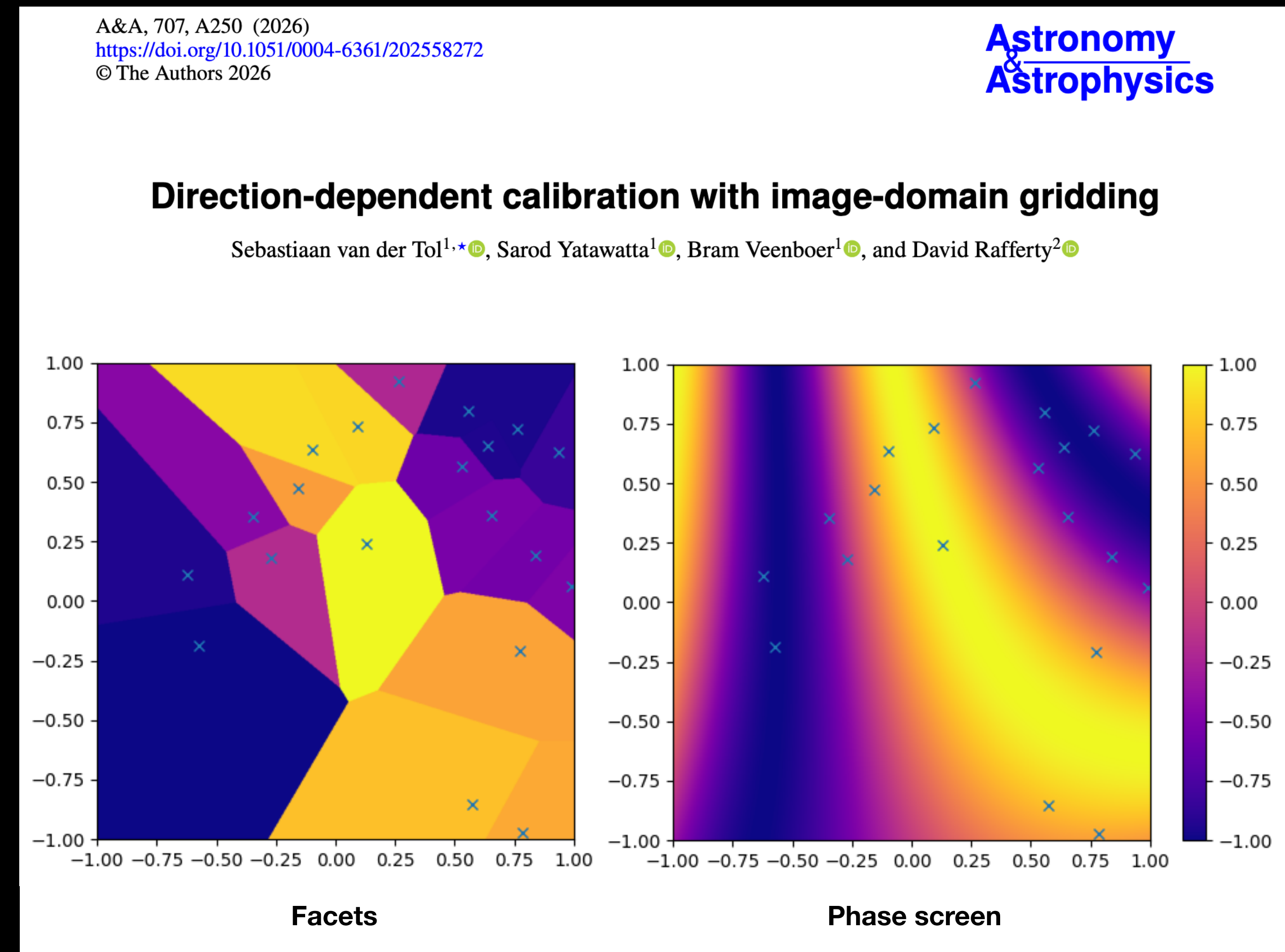
delay+tec+phase



After delay+tec+phase solve

IDG: Screen-based Calibration

- IDG-CAL offers an alternative to facet-based calibration:
 - Facet-based calibration: direction dependent effects are approximated as piece-wise constant functions.
 - Screen-based calibration (IDG-CAL): uses image-domain gridding to find a smoothly varying (continuous) phase-screen to model DD effects.
- IDG scales well to large images and maps efficiently to GPUs, thus being interesting for imaging large FoVs with the international stations.



IDG: Comparing Facets with Screens

7 x 7 deg² — 1.25" — facet-based

NGC 6338 @ 144 MHz w/ 12 MHz bandwidth

10 x 10 deg² — 1.2" — screen-based

Reduced bandwidth results are promising, but not yet demonstrated with full bandwidth (48 MHz)

How to Use IDG for Calibration & Imaging?

- Interested in experimenting with screen-based calibration and imaging with IDG(-CAL)?
 - Install the software: <https://git.astron.nl/RD/idg>.
 - The installation provides an IDG-CAL step DP3 step (**idgcal**).
 - For imaging you'd use WSClean with the **idg** gridded and provide the calibration solutions in an A-term configuration file.

```
1 # DP3 parset
2 steps=[idgcal]
3
4 idgcal.type=python
5 idgcal.python.module=idg.idgcaldpstep_phase_only_bfgs
6 idgcal.python.class=IDGCalDPStepPhaseOnlyBfgs
7 idgcal.maxiter=4
8 idgcal.solintphase=1
9 idgcal.polynomialdegphase=2
10 idgcal.subgridsize=32
11 idgcal.tapersupport=7
12 idgcal.wtermsupport=8
13 idgcal.atermsupport=8
14
15 # WSClean command
16 wsclean \
17 ...
18 -idg-mode gpu \
19 -aterm-config aterm-cycle-1.conf \
20 -aterm-kernel-size 20 \
21 -name cycle-1 \
22 observation.ms
23
24 # Contents of aterm-cycle-1.conf
25 atarms = [beam, idgcal solutions]
26
27 idgcal solutions.type = h5parm
28
29 idgcal solutions.files = [cycle-1.h5parm]
30 idgcal solutions.update_interval = 8
31
32 beam.differential = true
33 beam.update_interval = 601
34 beam.usechannelfreq = true
```

Time and Frequency Smearing Corrections in WSClean

- Because of time and frequency smearing: the image rotates over time and scales with frequency.
- This leads to blurring of sources further from the phase centre.
- And lower visibility amplitude for longer baselines.
- Can be absorbed only partially during calibration, since this effect is baseline and not station based.
- Available in WSClean since version 3.7:
`-apply-time-frequency-smearing.`

Without corrections

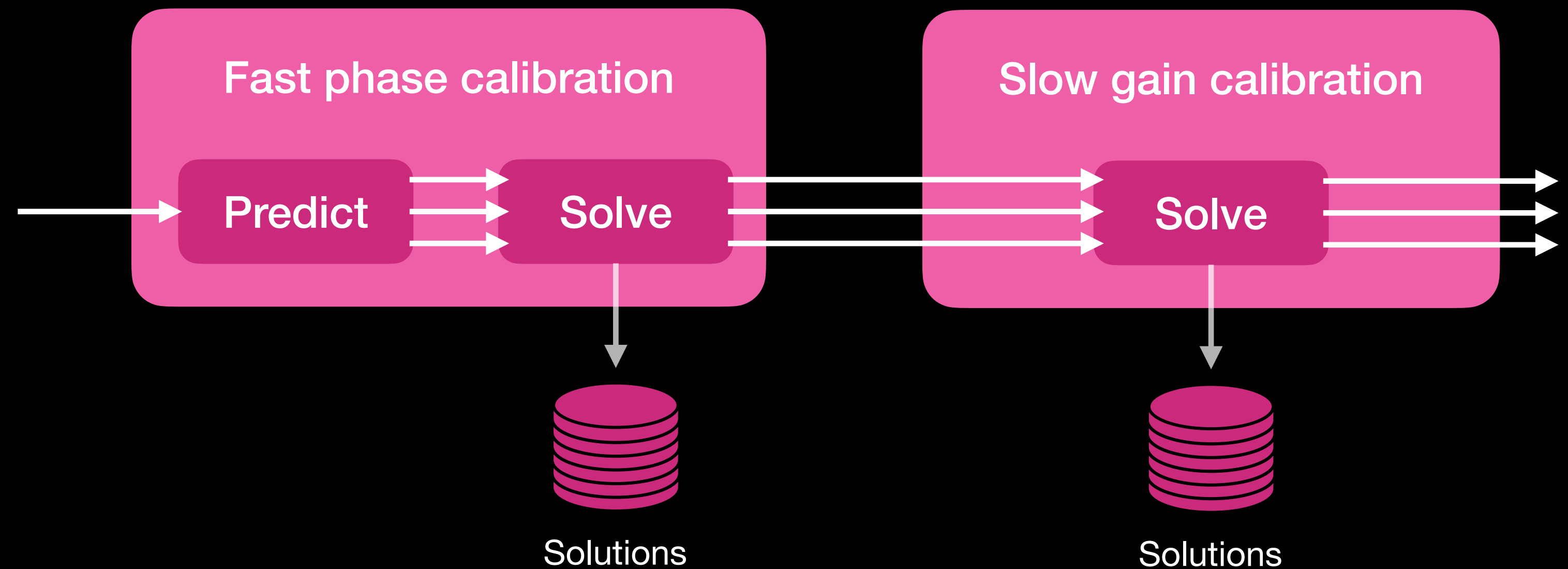
With corrections



<efficiency>

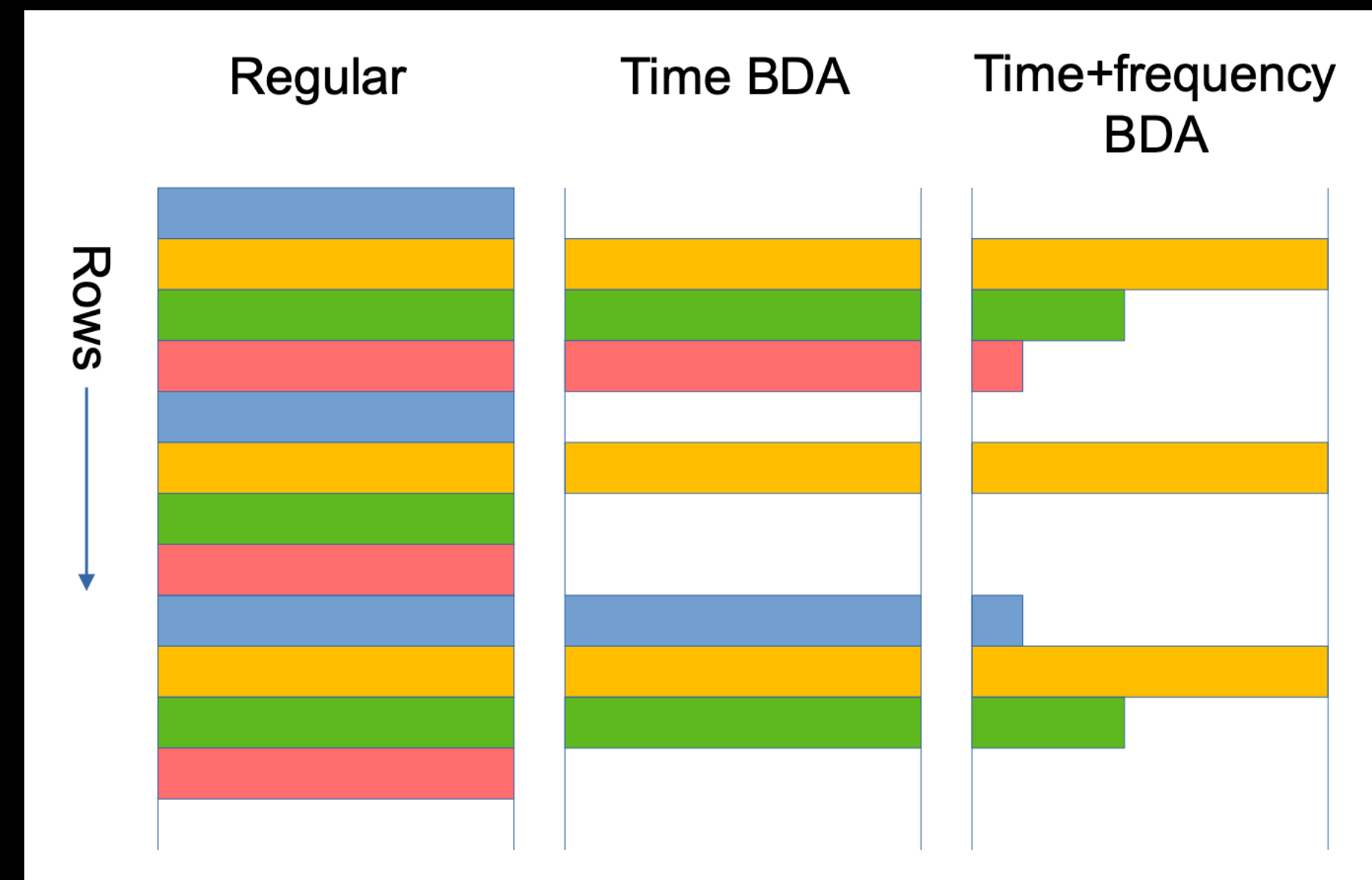
Multi-calibration in DP3

- “Multi-calibration” refers to performing multiple consecutive solves.
- ... while reusing the predicted model data — previously not possible.
- For example a fast phase solve to correct for quickly varying ionosphere for longer baselines and a slow gain solve for remaining instrumental effects.



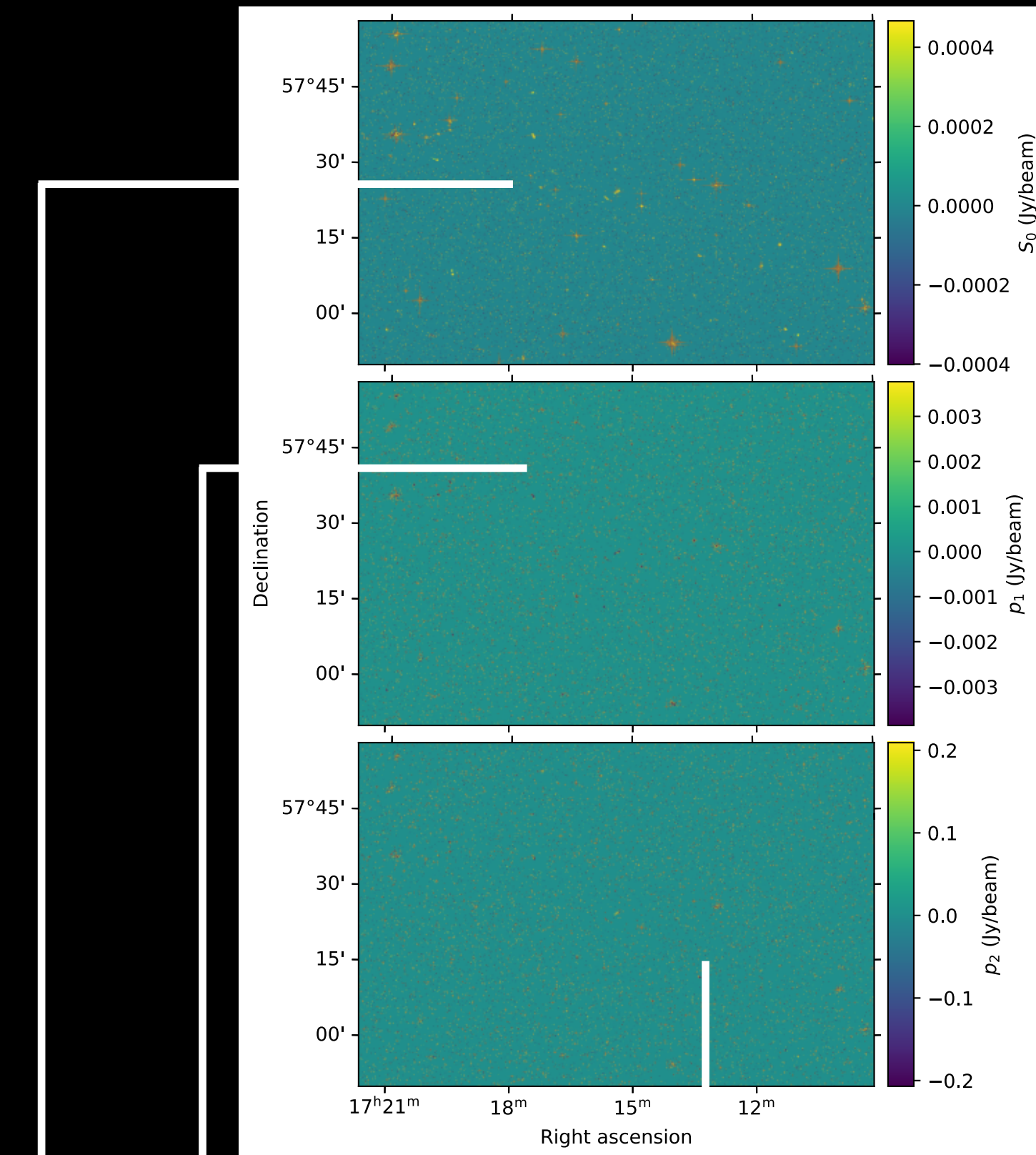
Spectral BDA in WSClean

- The idea behind baseline dependent averaging (BDA) is that it allows more averaging in time and freq. for shorter baselines
- This yields smaller chunks of data, thus — in theory — faster processing, e.g. faster gridding or calibration!
- Already supported in DP3 in a number of steps (averager, ddecal, ...) and now WSClean as well, **since version 3.7**.
- For now time-only BDA is significantly faster than no BDA, but time+freq. BDA is about as fast as without BDA.



Sky-model Rendering for Image-based Prediction

- For a large amount of sources predicting visibilities (i.e. solving the Measurement Equation) is expensive, because it scales with the number of sources and baselines.
- While image-based predict roughly scales with the number of pixels in your model image.
- To bridge this gap, we introduced an option to WSClean* `-draw-model <skymodel>` to render text sky models to images with sub-pixel accuracy

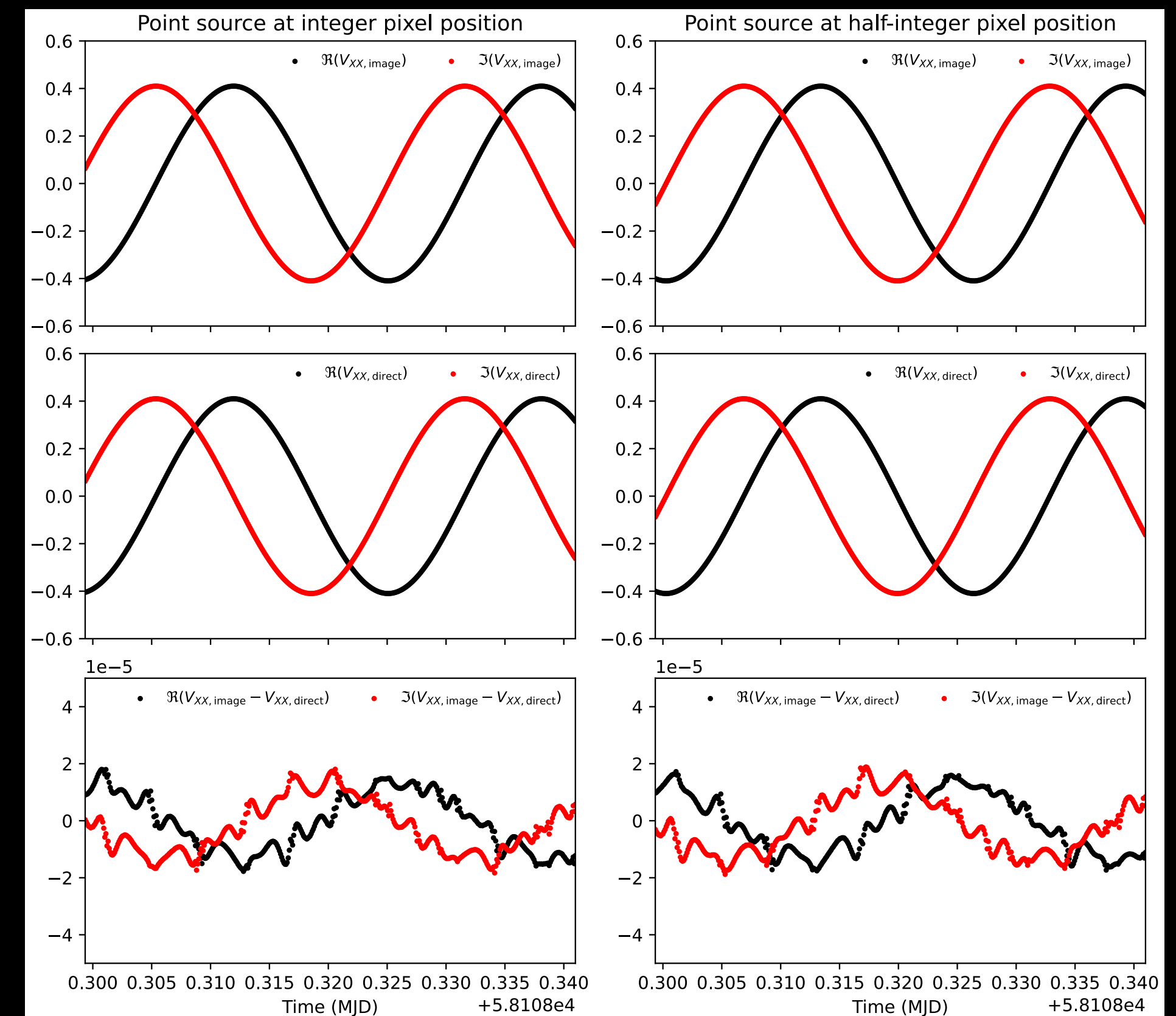


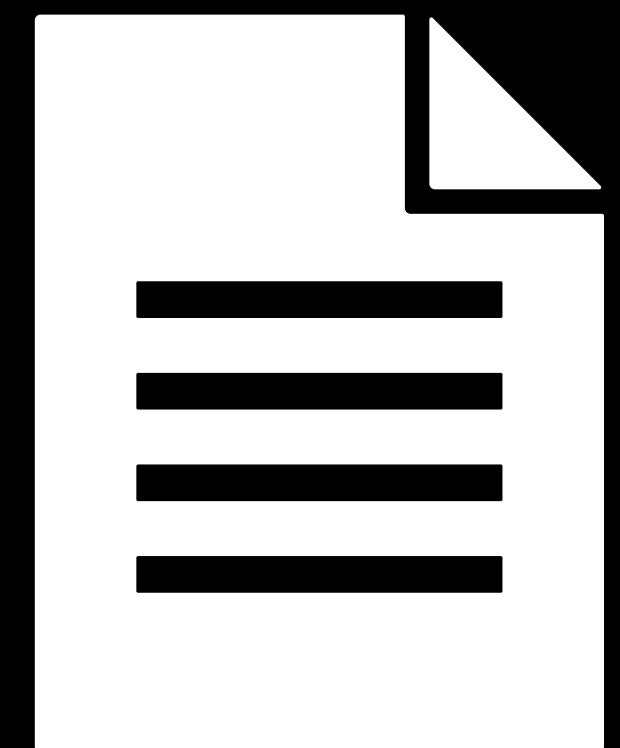
$$S(\nu) = S_0 + p_1 \left(\frac{\nu}{\nu_0} - 1 \right) + p_2 \left(\frac{\nu}{\nu_0} - 1 \right)^2$$

*Further details are available in the WSClean manual:
https://wsclean.readthedocs.io/en/latest/sub_pixel_rendering.html

Sky-model Resampling

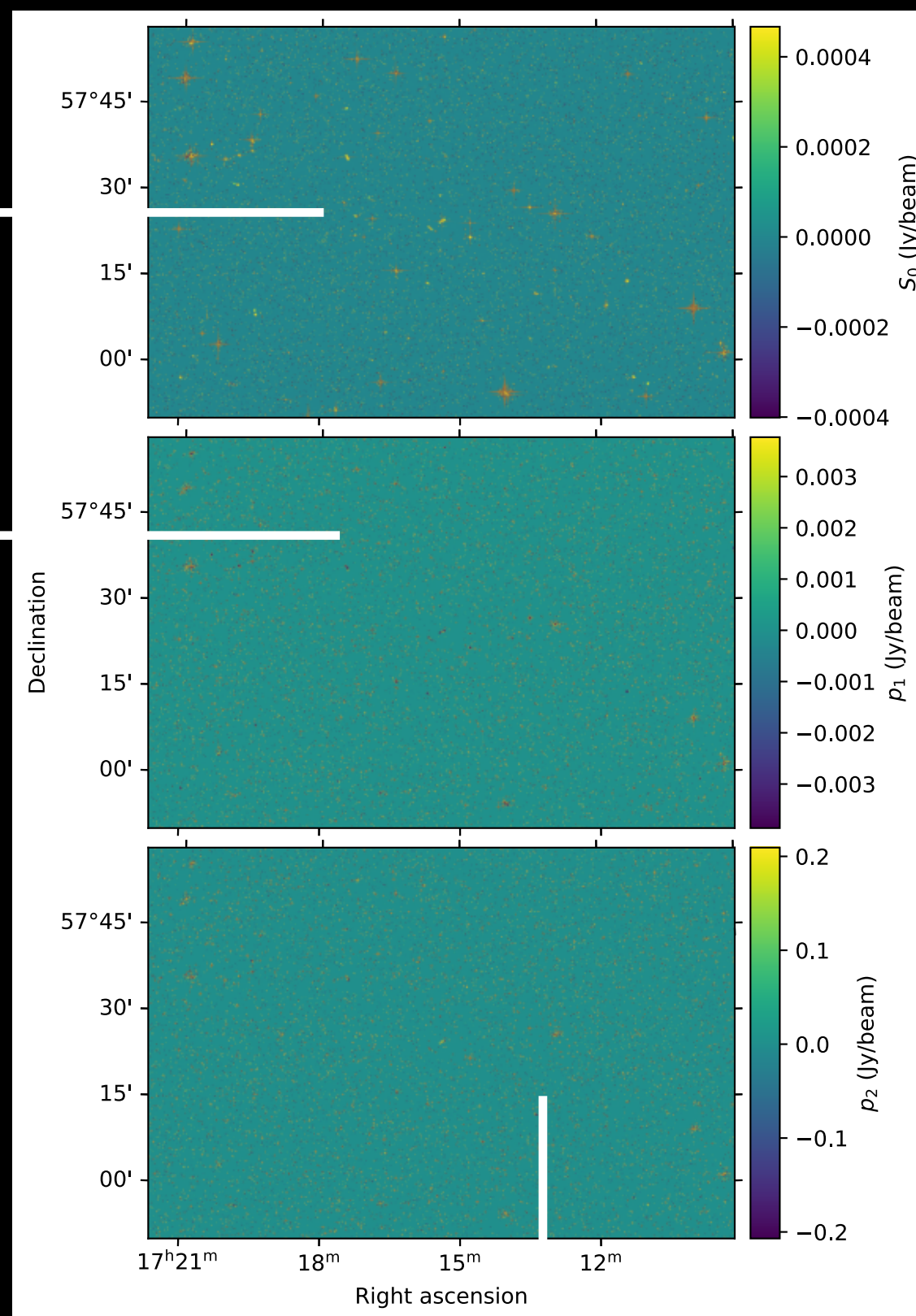
- What resampling methods do we use?
 - **Point sources** are convolved with a windowed sinc function.
 - **Small Gaussian sources** are rendered in the uv-domain
 - **Large Gaussian sources** are smooth enough to render directly onto the image.
- The accuracy of this approach is of similar order as the default gridding accuracy (10^{-4}) in WSClean.





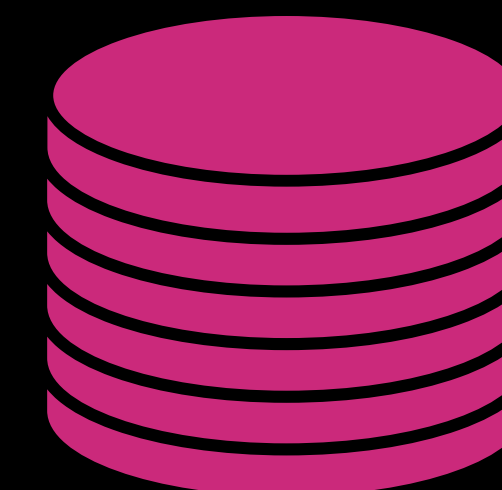
Sky model

WSClean



DP3

WGridderPredict



Model Visibilities

$$S(\nu) = S_0 + p_1 \left(\frac{\nu}{\nu_0} - 1 \right) + p_2 \left(\frac{\nu}{\nu_0} - 1 \right)^2$$

<scalability>

Improving Multi-node Processing with WSClean

- Significant effort from SKA developers (in particular Malcolm MacLeod) led to a “**shared reads & writes**” mode, improving the multi-node (MPI) mode of WSClean.
 - Distribution over nodes is done both over facets and output channels, which is often not efficient, because of repetitive I/O.
 - In this mode, larger chunks of gridded visibilities are shared between groups of facets, instead of individual facets.
 - We have observed a 10x speed-up when using 12 nodes for imaging.
 - It can even make single-node processing faster! In this case we’ve seen improvements of ~10%.

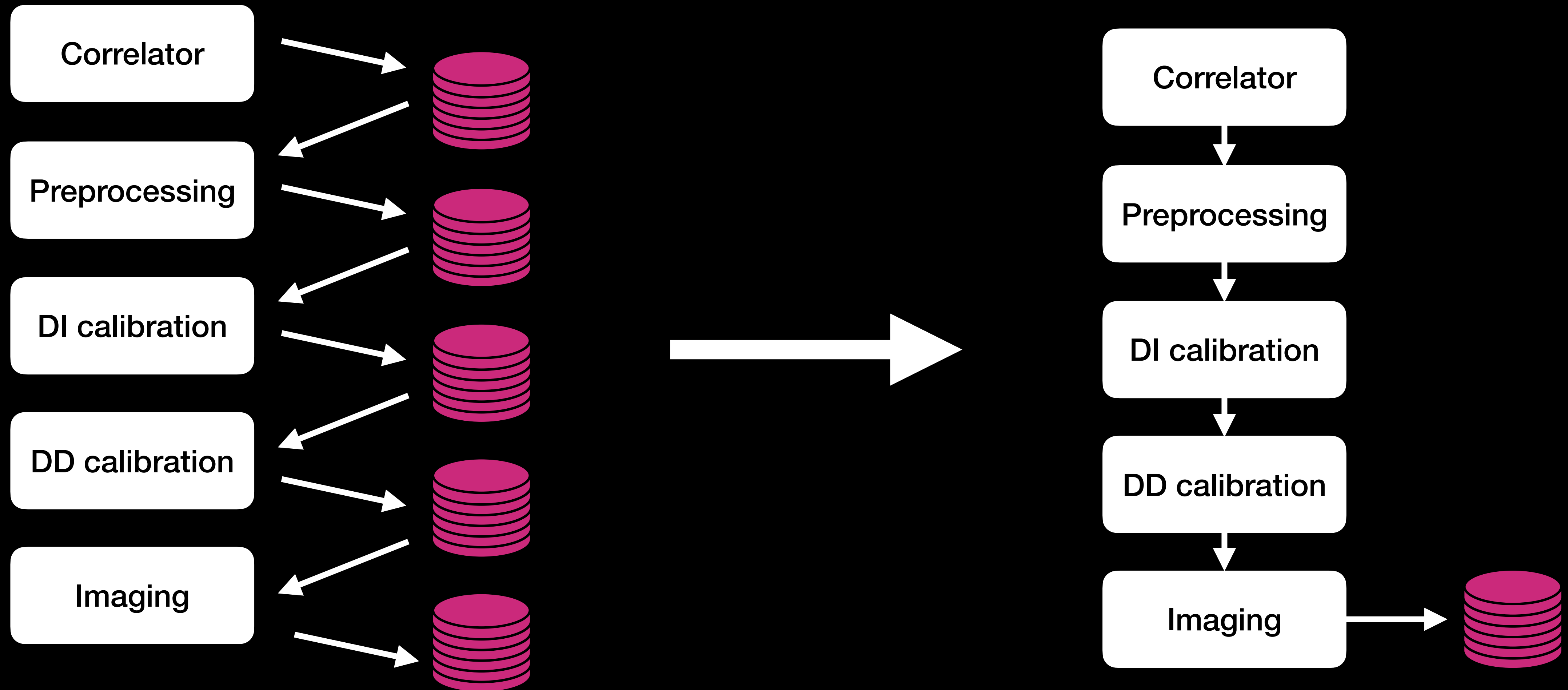
And Moving Processing to GPUs

In collaboration with the High Performance Computing team @ ASTRON

- ASTRON's HPC team is actively researching the possibility of porting our (mostly) CPU based software stack to GPUs.
- They started modularising the predict routines in DP3, optimising the CPU version, and are working on delivering a GPU version.
 - They created a library dubbed *FastPredict* that leverages a GPU's Special Function Units for sine/cosine operations.
- Over the past year, we've already started working on a new interface for EveryBeam that would map well to GPUs, and might be an attractive option to be picked up by the HPC team as well in the future.

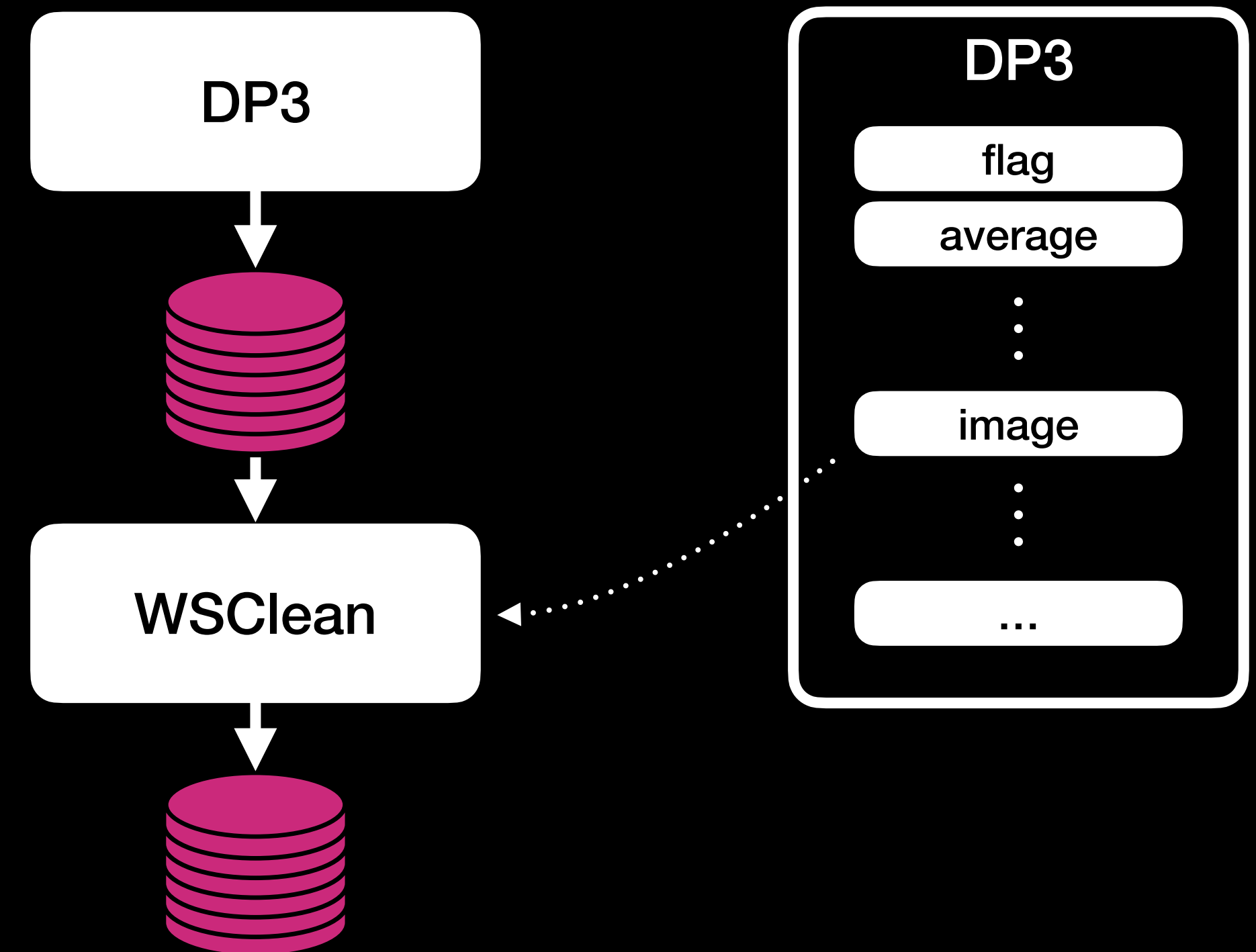
<storage>

Towards Streaming Interferometric Processing



Towards Streaming Processing: the **First** Steps

- To realise this we added an “in-memory mode” to WSClean:
 - Enabled using the `in-memory` command-line option.
 - Keeps the entire MS in memory, thus usage is limited to datasets that fit into the memory of your cluster.
 - WSClean provides a low-level (C++) interface to its imaging routines along with structures to define an in-memory MS.
 - Though beam corrections are not yet supported, because EveryBeam requires reading a MS.

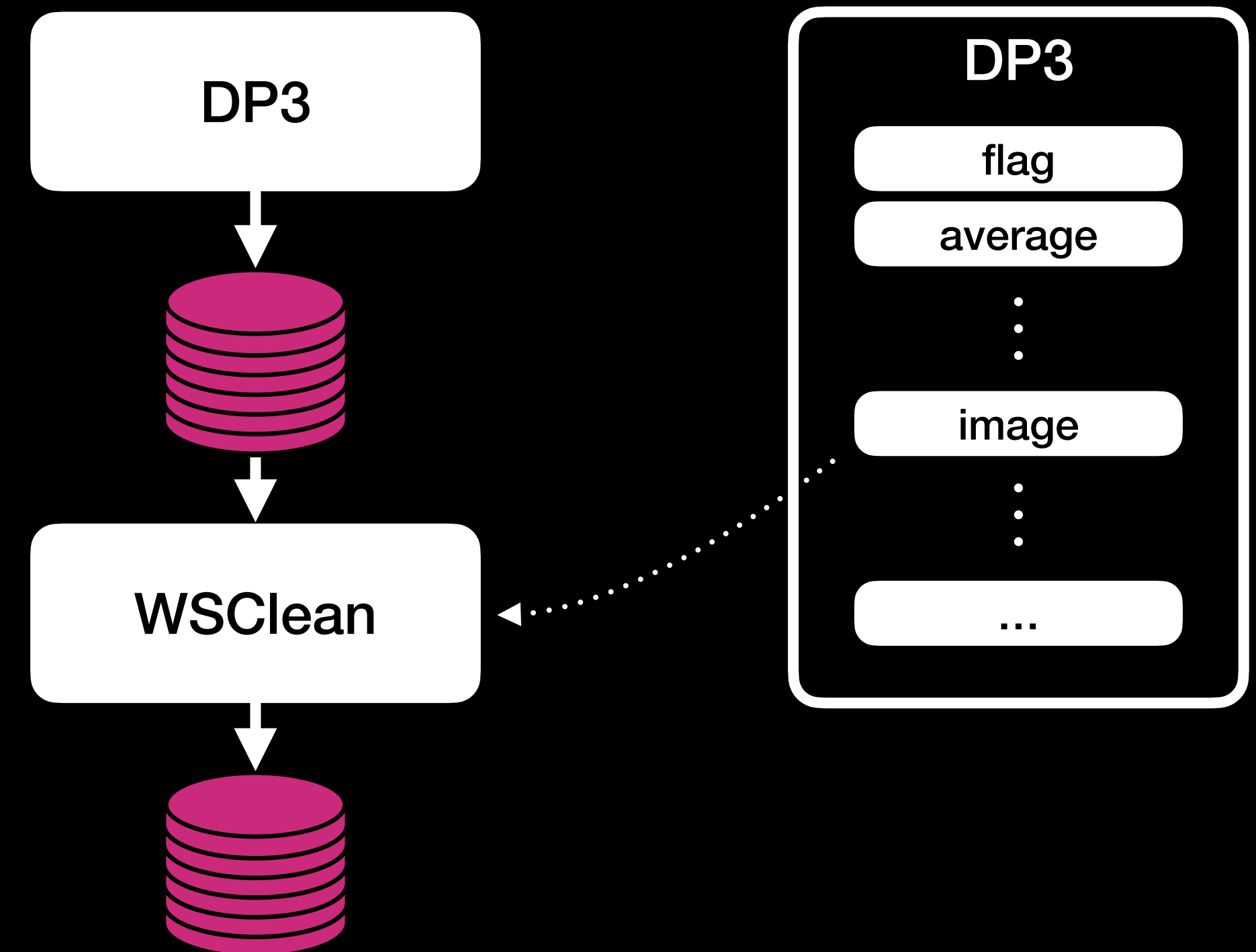


Towards Streaming Processing: the **First** Steps

- WSClean can now be called from within DP3 using the new `image*` step.
- This DP3 step supports most imaging operations of WSClean and can produce images at a provided cadence.

Example...

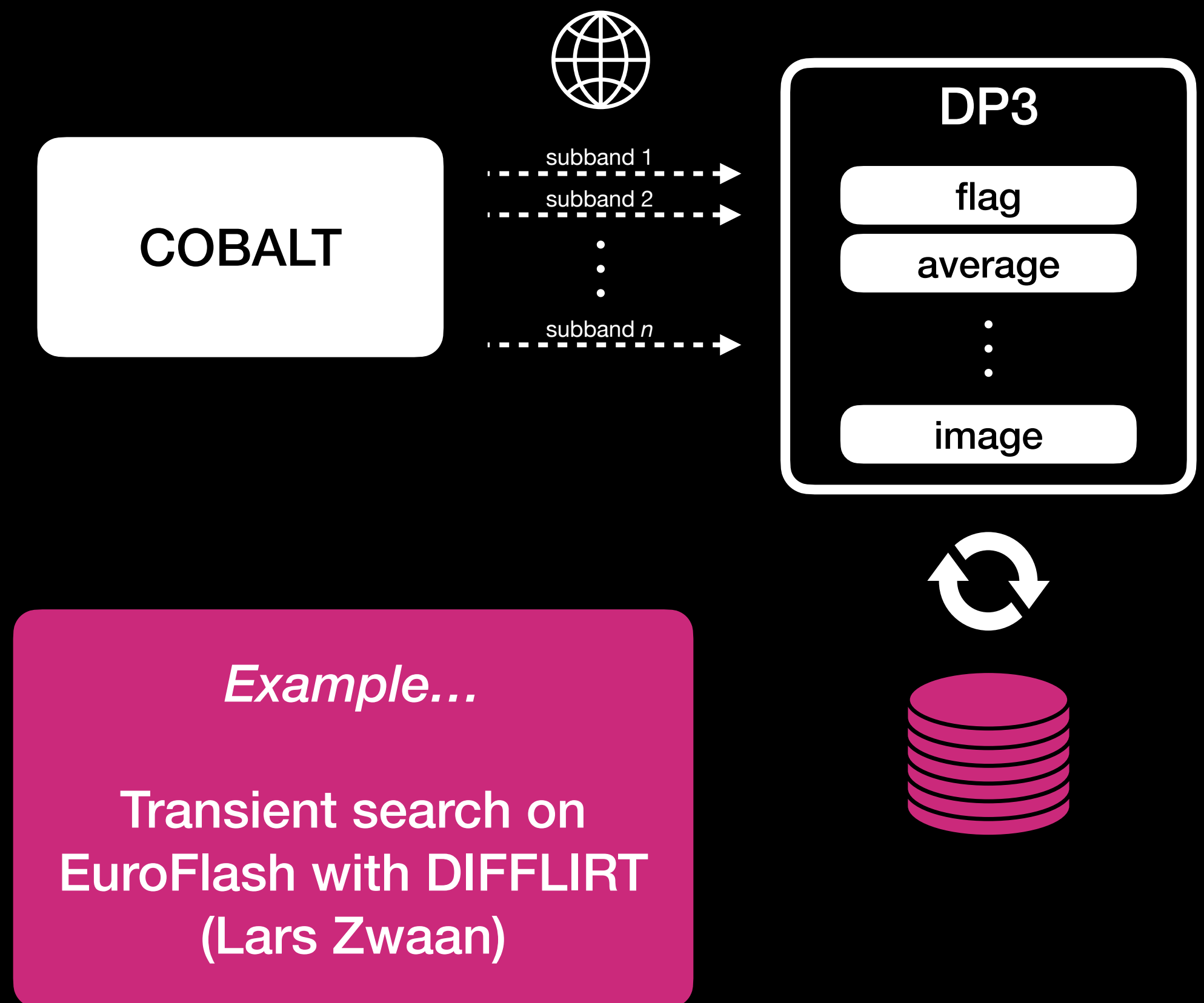
Could be used to speed-up PILOT's self-calibration, e.g. the “calibrate-apply-image” operations



* Further details are available in the DP3 manual:
<https://dp3.readthedocs.io/en/latest/steps/ImageStep.html>

Towards Streaming Processing: the **Next** Steps

- One of the goals of this in-memory processing / imaging with DP3 + WSClean is realising a connection with the correlator (COBALT).
- Already tested, but not *yet* production ready.
- *Near future*: enable streaming imaging from the correlator on the EuroFlash cluster to aid transient searches.
- *Slightly further future*: *potentially* skip writing to disk before direction-independent calibration.
- If processing is sufficiently fast, this allows real-time image output for the first time!



Compression

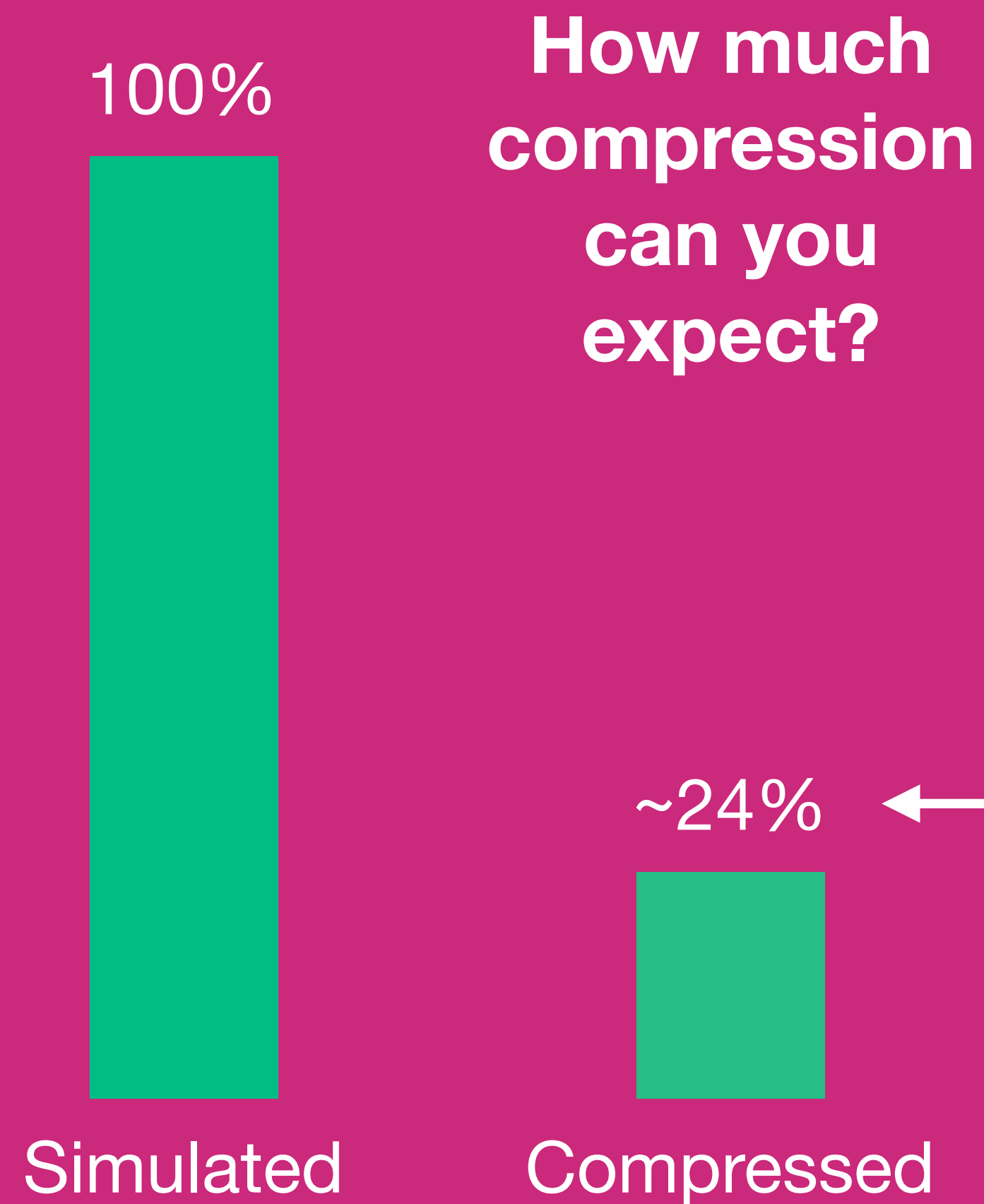
META DATA

- MSs by default store “a lot” of duplicate information:
 - Antenna name pairs repeat, thus only need to be stored once.
 - Often flags are set identically for each polarisation, thus storing 1 instead of 4 copies is sufficient.
 - Instead of storing a single UVW coordinate for each baseline, store them once per antenna instead.
- **On a LINC-processed 20-channel LOFAR MS, this saves 14% (9% uvw, 2.5% flags, 2.5% ant.)**
- *As of 3rd of June 2026 these are turned on by default in DP3.*

MODEL DATA

- We can now compress model data with Sisco (Simulated Signal Compression).
- Sisco provides lossless compression and is integrated as a storage manager in Casacore.
- How: Prediction → subtraction → grouping → DEFLATE .
- Note that lossy compression algorithms (e.g. Dysco) add noise to noiseness model data.

Model Data Compression



MODEL DATA

- We can now compress model data with Sisco (Simulated Signal Compression).
- Sisco provides lossless compression and is integrated as a storage manager in Casacore.
- How: Prediction → subtraction → grouping → DEFLATE.
- Note that lossy compression algorithms (e.g. Dysco) add noise to noisiness model data.

<summary>

Summary

- We've been working on a wide range of new features in DP3, WSClean, EveryBeam, IDG, and many more! *This was but a teaser.*
- ... ranging from general speed-ups to increasing scalability and reducing data sizes.
- ... a lot of these features were requested by the community ❤️
- Future plans include supporting BDA all throughout our software, continuing to modularise our software and port it to GPUs wherever sensible, etc.