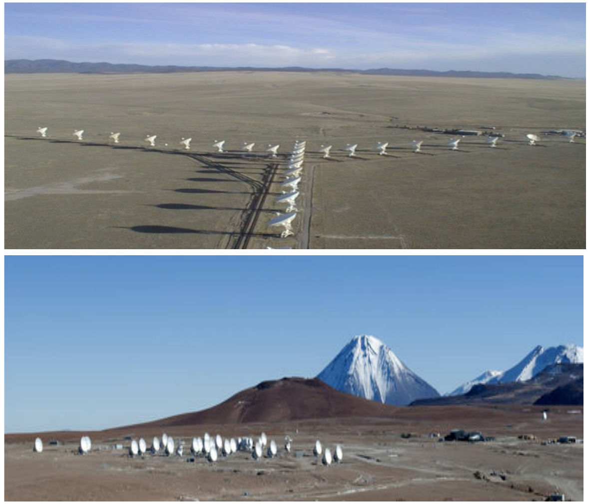
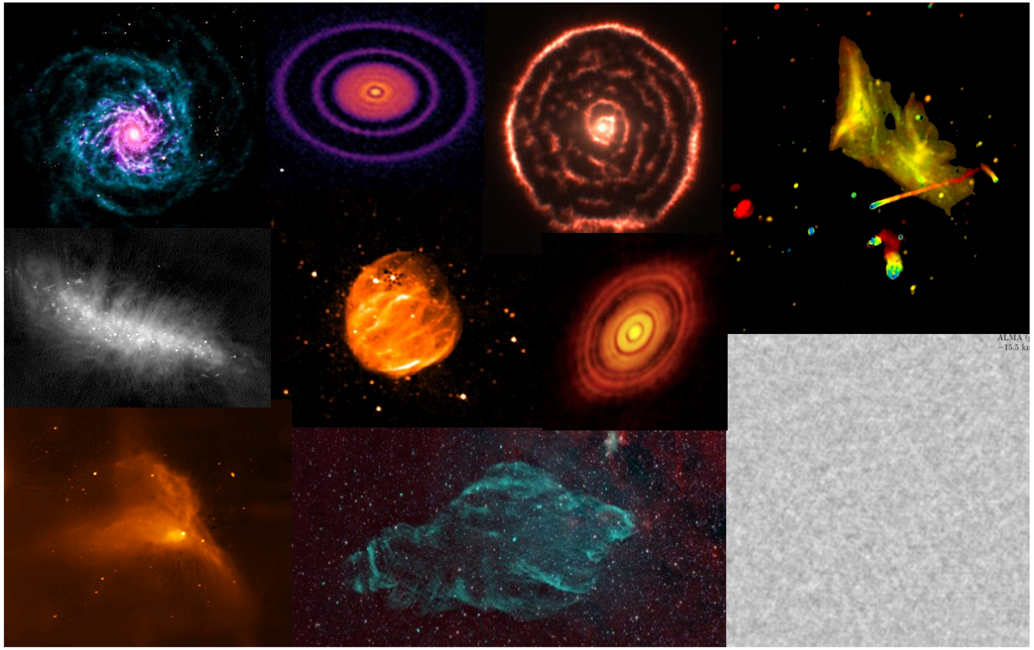
Quantifying Scientific Correctness in Radio Interferometric Imaging

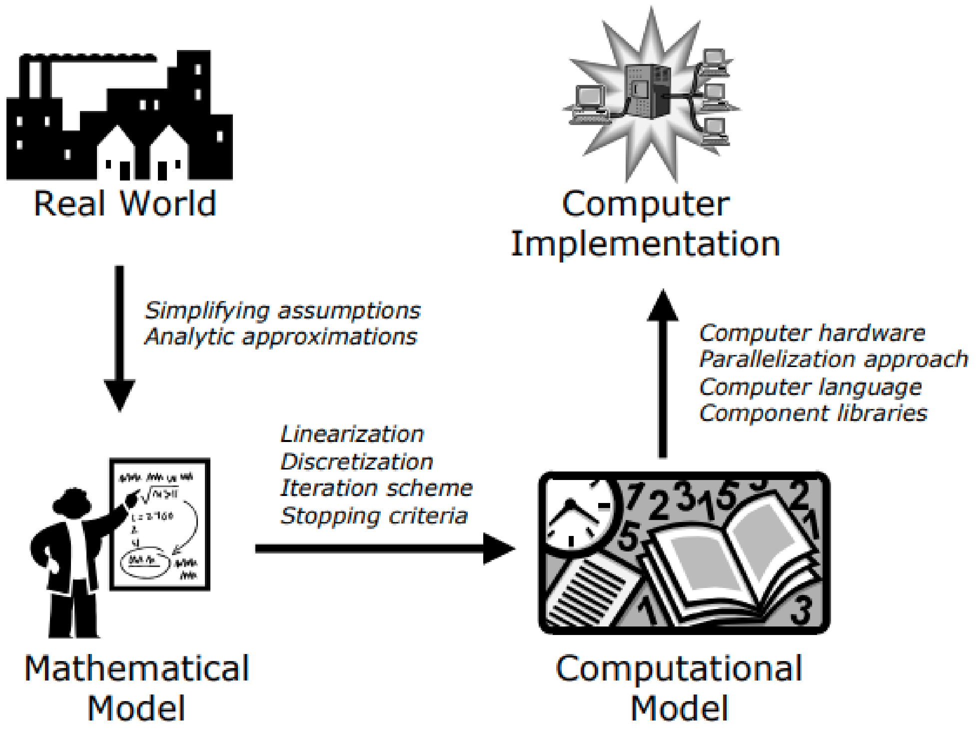




Urvashi Rau
National Radio Astronomy Observatory, Socorro, NM, USA

28 Oct 2021, ADASS XXXI

Scientific Software



Ref : Accuracy and Reliability in Scientific Computing (SIAM, 2005)

Observational Astronomy

- Observe unknown structures
- Use instruments whose characteristics must be modeled and corrected for in software

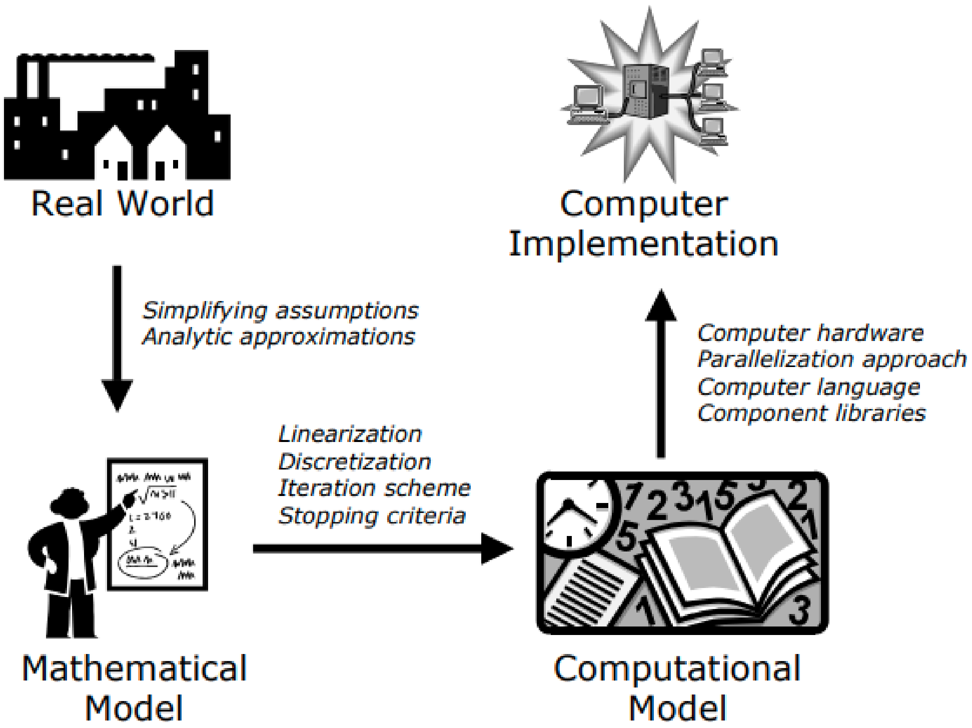
Practical Scientific Software

- What defines absolute correctness ?
- What defines the operational readiness of the software ?
- As code/software evolves, what to do when numbers change ?

Requirements, Specifications, and Tests

- Truth values
- Tolerances

Scientific Software



Ref : Accuracy and Reliability in Scientific Computing (SIAM, 2005)

Truth Values

Simulations

- Controlled (limited) environment
- Truth : Known exactly.

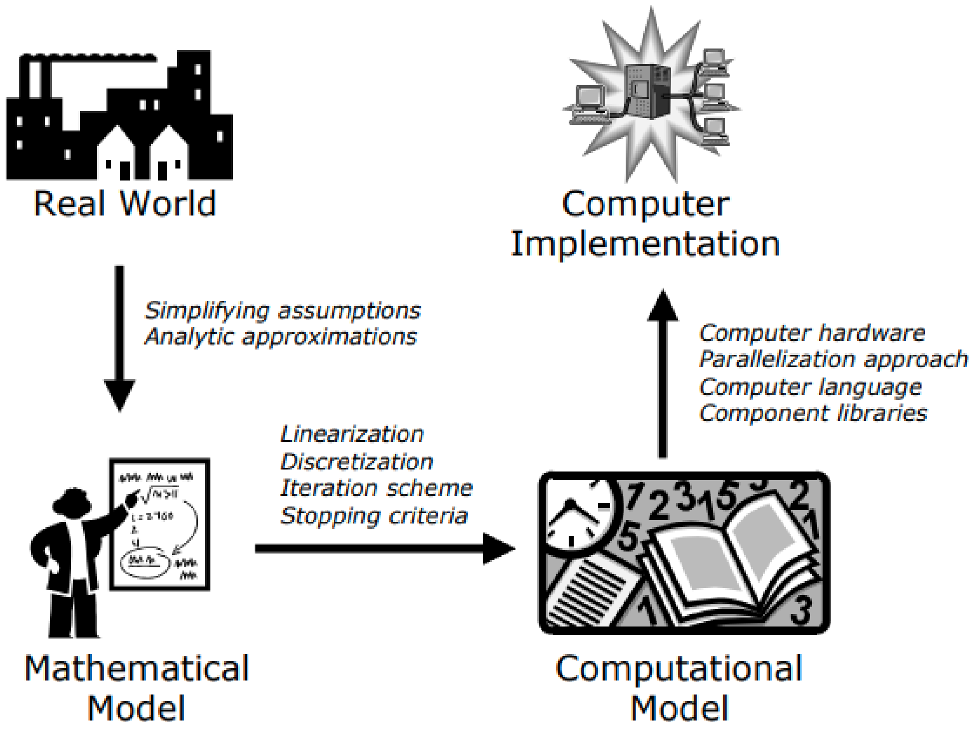
Test observations

- Realistic environment
- Truth : An independent measurement
- Truth : Value obtained when the test was written

Tolerances

- Accuracy needed for astrophysics ?
- Accuracy defined by instrument limits ?
- Accuracy of the algorithms/implementations ?
- Machine precision ?
- Include effects of error propagation ?

Scientific Software



As code and algorithms evolve….

…. numbers change.

- What changes are ok, and what are not ?
- Which tolerance to use ?
- ‘Best result’ truth values can change.
- What happens when a bug and a legitimate instrumental artifact produce a similar change in output ?
- Must fix bugs, but must also consider the Cost vs Benefit of change/error/bug analyses

Ref : Accuracy and Reliability in Scientific Computing (SIAM, 2005)

A Case Study

Radio Interferometry

– Sources of uncertainty and error

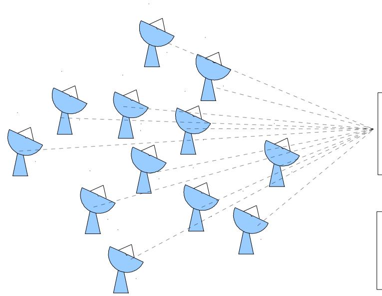
The CASA software package

– Navigating this situation

Radio Interferometry : Data acquisition and analysis

An indirect imaging technique

=> **Measurements** : An incomplete sampling of the 2D spatial Fourier Transform of the sky brightness.
=> **Noise** : Gaussian random
=> **Reconstruction** : Iterative numerical optimization to solve for instrument and sky model parameters



Correlation : Time Series → Correlation → Spectral Channels → Integrate

Data Archive : Each observation is stored as a database

Post Processing

Flagging → Calibration → Imaging

Flagging

Identify and mask corrupted data
(RFI, Instrument errors, etc)

- **Data Loss**
- **Accuracy of outlier detectors**

Calibration

Solve for and apply corrections to undo the effects of complex valued antenna gains

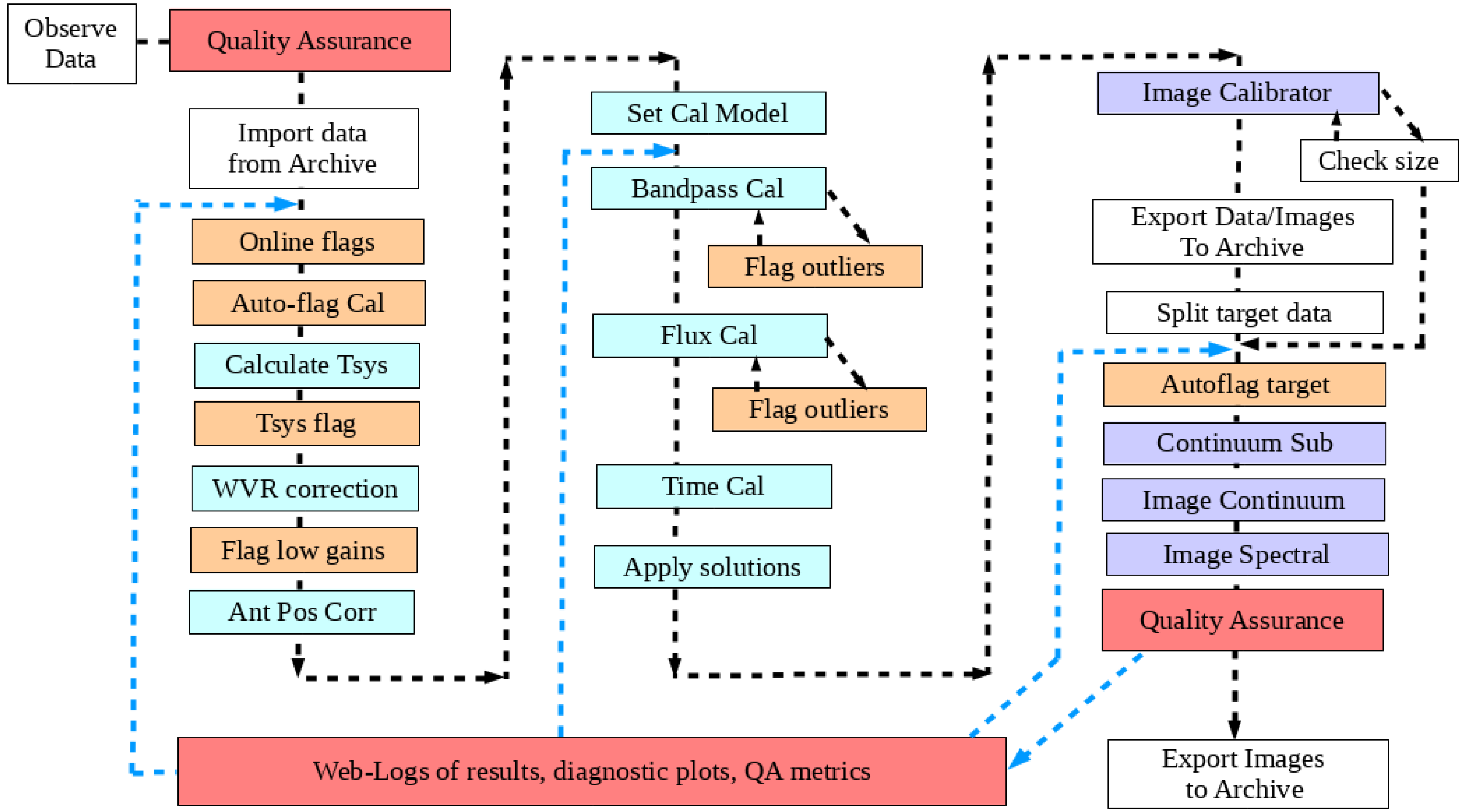
- **Solver (and model) accuracy**
- **Available signal-to-noise**

Imaging

Reconstruct images by iterative model fitting while correcting for other instrumental effects

- **Reconstruction uncertainty**
- **Approximations (instrument + sky)**
- **Available signal-to-noise**

Automated Pipelines



Sequences of steps

- Feedback loops
- Conditionals
- Heuristics developed on benchmark datasets
- Sequences vary per telescope and observing mode

Accuracy of the Heuristics

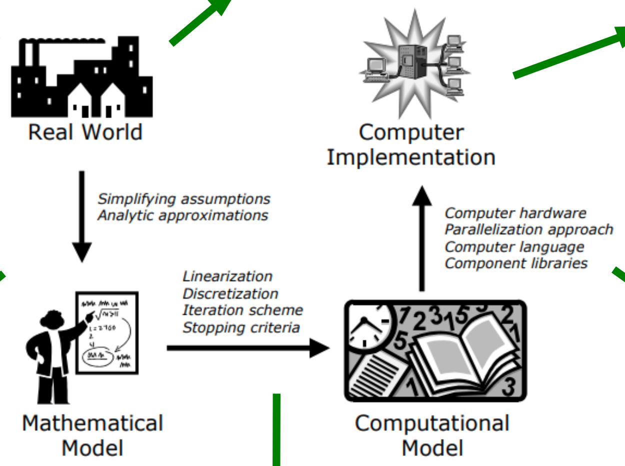
Error Propagation

Factors affecting accuracy

- Incomplete sampling of the data domain
- Instrument/system noise
- Fraction of data loss

- Error propagation

- Bugs



- Modeling the sky brightness with delta-functions (or other basis fns)
- Modeling an antenna power pattern with a simple Airy disk
- Assumptions about signal behaviour : perfect Gaussian random noise

- Different compilers
- Different versions of 3rd-party software dependencies
- Operating systems
- Serial, mpi, openmp, cuda
- HPC frameworks

- Choices of FFT padding : Aliasing errors
- Pixel or bin sizes : Quantization errors
- Robustness of the algorithm and range/granularity of the controls

A Case Study

Radio Interferometry

– Sources of uncertainty and error

The CASA software package

– Navigating this situation

CASA : Common Astronomy Software Applications

CASA : A general-purpose suite of radio interferometry analysis tools operable within a Python environment

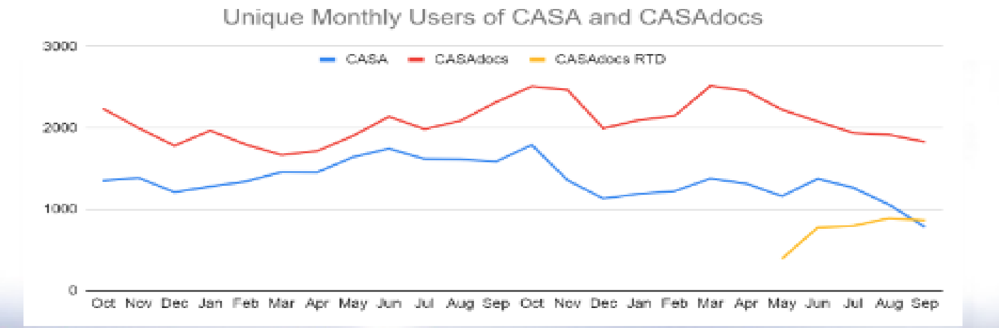
Team : ~20 software engineers, algorithm scientists and astronomers.
(Build/Release, Infrastructure, Science Dev, Verification, Documentation)

Stakeholders :
VLA users, ALMA users, Users of other telescopes (GMRT, MeerKAT, etc...)
VLA-Sky-Survey pipeline, ALMA pipeline(s), VLA/SRDP pipeline
ngVLA simulations/studies

Partners : Algorithm R&D group, Pipeline Dev team,
VLBI dev team, Single-dish dev team, CARTA-team

(Production pipelines are built using CASA methods + Heuristics)

User Base :

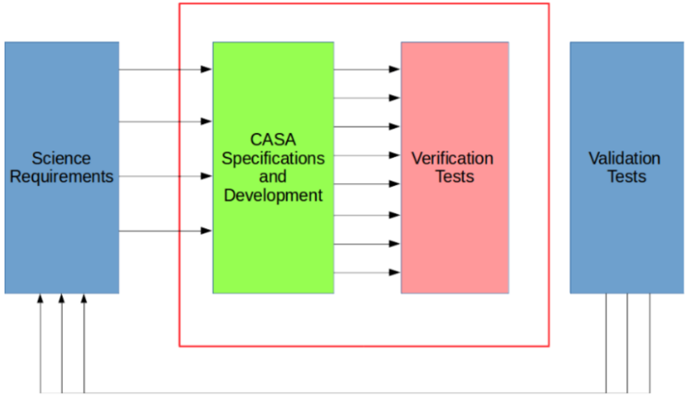


casadocs.readthedocs.io

CASA
Common Astronomy Software Applications

1 – 3 releases per year

Development Process



Operational Complexity

Usage modes

- Manual data reduction

- Interactivity (visualization, logs, GUIs…)
- Flexible tuning/exploration (lots of parameters/options)
- New options/features continuously added

- Production pipelines of multiple telescopes/projects

- Stability & Reproduceability
- Algorithmic evolution + support for new modes
- Low tolerance for un-asked-for changes

- Algorithm R&D : Design modularity + stability

Operating platforms :

- Desktops/Laptops/Clusters/Cloud
- Parallelization : MPI, OpenMP, GPU

Code Base :

- C++, Fortran, Python
- Experimenting with Python / Dask / Xarray / Docker, etc…

Requirements :

– Usually written as feature requests, algorithms, or problems to solve

– Metrics are often not defined up front

- Based on ‘best possible outcome’ after experimenting with a solution.
- Independent analyses are sometimes available, but not always.

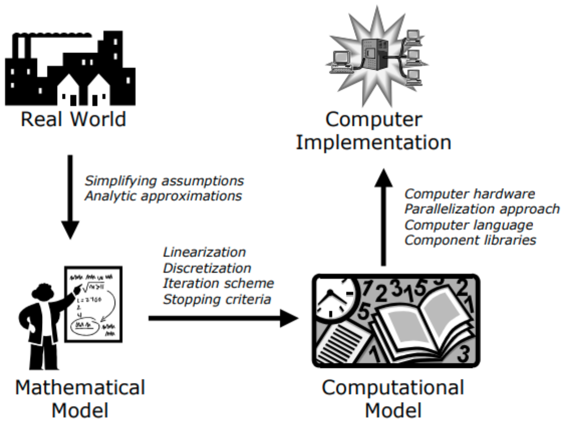
=> People are extremely wary of change.

=> Loss of objectivity.

=> Inefficient development process

Need ways to build trust….

Tests are growing



Functional Verification Tests

- Tests written against feature specifications. Emphasis is code coverage.
- Small and simple simulations/datasets. Test numerics and algorithm features

Algorithm Characterization

- Detailed simulations and analyses, with science metrics.

Stakeholder Verification Tests

- Pipeline benchmark datasets for major usage modes. Use analyses steps and metrics relevant to stakeholders.
- They also track diffs/changes (arising from CASA) at numerical precision level

Pipeline Validation Tests (run by pipelines, not CASA) : End-to-end tests for science readiness on ~100+ datasets

Performance tests : Monitor runtime and memory usage

Manual tests : Generic datasets. Use experience and technical expertise to assess ‘correctness’.

However, there is still a large variety in metrics, tolerances, and acceptance rules.

Metrics : Towards consistency

Science-driven accuracy limits for major usage modes :

Requirement : **X** → Good enough for most operations
Goal : **Y** → Best case. This is the algorithm-development target

Demonstrated and documented accuracy of software : **Z**

Use simulations or carefully-designed test observations. Z is defined w.r.to a known truth value.
(A required operational accuracy constraint : Z = X/1000 or Y/10 to account for error propagation)

Acceptance Rules

- If **X > Y > Z > Є** => All is well.
- If **Z > X** => Unacceptable, and needs algorithm R&D or re-evaluation of requirements.
- If **X > Z > Y** => Acceptable, but improvements are desired.
  - Changes above Y should be tracked/understood and communicated on a case-by-case basis.
  - Algorithm development should continue where relevant, to get Z < Y
- When numbers change, use tighter(Y, Z) as the tolerance for acceptance.

=> Allow the code to evolve within these limits.

Future

Current Operations (ALMA / VLA / VLASS / etc..) : Define metrics retrospectively and try to evolve....

New/Upgraded Telescopes (ngVLA + ALMA)

- New Software "ngCASA"
- Define use cases, metrics, acceptance rules at the start (but also plan for evolution....)

CASA Next Generation Infrastructure Project : https://cngi-prototype.readthedocs.io/en/stable

- Under evaluation
- Open-source 3rd party infrastructure for operational flexibility and some numerics (e.g. astropy)
  → Reduced in-house control of numerics

Thank you !