Tweaking the Pipeline Script

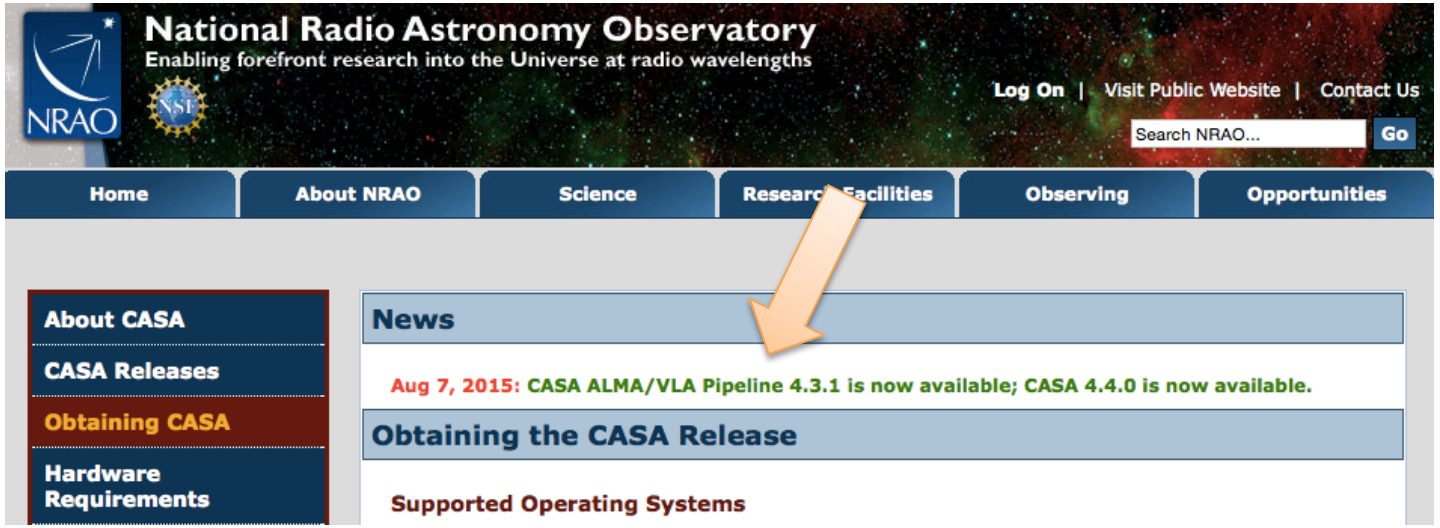
Original version by Liz Humphreys (ESO)

Rie E. Miura (NAOJ)

ALMA Pipeline (PL) Deliveries

- If the Pipeline has been used in a delivery, this will be marked in the **README** file of the delivery **package**

- If the Pipeline has been used, then a **special version of CASA 4.3.1 (->CASA 4.5)** must be installed which contains Pipeline tasks to re-create calibrated data



- Start using **casapy –pipeline**

Delivered Package

- $ ls …/2013.1.12345.S/science_goal.uid___A001_X100_X10a/group.uid___A001_X100_X10b/member.uid___A001_X100_X10c/
- |-calibration/ #← flagtemplate.txt, flux.csv, jeyperk.csv
  |- log/
  |- product/ #<- FITS files
  |-qa/ #← WebLog
  |-**README**
  |-script/
  |-casa_piperestorescript.py
  |-casa_pipescript.py
  |-PPR_uid___A001_X100_X10d.xml
  |-**scriptForImaging.py**
  |-scriptForPI.py

Re-creating PL calibrated data
Interferometry data

- If the goal is to re-create the calibrated data produced during the ALMA Quality Assurance process, then **scriptForPI.py** should be used.

- **scriptForPI.py** calls **casa_piperestorescript.py**. **scriptForPI.py** is the fastest way to obtain calibrated data, e.g. before re-imaging ALMA data

- the **casa_pipescript.py** should only be used to fully re-calibrate ALMA raw data when it is wanted to tweak an aspect of the calibration

Re-creating PL calibrated data
TP data

- Single Dish Pipeline products contain image FITS cubes with native resolution. If you want to those with preferable resolution, just use the CASA task imregrid.

- If the goal is to re-create the calibrated data produced during the ALMA Quality Assurance process, then **scriptForPI.py** should be used.

- **scriptForPI.py** calls **casa_pipescript.py** which fully re-calibrates ALMA raw data when it is wanted to tweak an aspect of the calibration.

- Note that Single Dish Pipeline does not have restore task. Re-calibration will take hours to complete pipeline processing.

Interferometry Pipeline

casa_pipescript.py
(get from the **script** directory**)**

To reproduce QA2 Reduction: **flux.csv** and ***flagtemplate.txt** must be in the same directory
(get from the **calibration** directory)

```
_rethrow_casa_exceptions = True
h_init()
try:
    hifa_importdata(vis=['uid___A002_X839000_X2a15',
'uid___A002_X839000_X2dc6'], session=['session_1', 'session_2'])
    hifa_flagdata(pipelinemode="automatic")
    hifa_fluxcalflag(pipelinemode="automatic")
    hifa_rawflagchans(pipelinemode="automatic")
    hif_refant(pipelinemode="automatic")
    hifa_tsyscal(pipelinemode="automatic")
    hifa_tsysflag(pipelinemode="automatic")
    hifa_wvrgcalflag(pipelinemode="automatic")
    hif_lowgainflag(pipelinemode="automatic")
    hif_setjy(pipelinemode="automatic")
    hif_bandpass(pipelinemode="automatic")
    hifa_spwphaseup(pipelinemode="automatic")
    hifa_gfluxscale(pipelinemode="automatic")
    hifa_timegaincal(pipelinemode="automatic")
    hif_applycal(pipelinemode="automatic")
    hif_makeimlist(intent='PHASE,BANDPASS,CHECK')
    hif_makeimages(pipelinemode="automatic")
finally:
    h_save()
```

Single Dish Pipeline Cycle3+ data

casa_pipescript.py
(get from the **script** directory**)**

To reproduce QA2 Reduction: **jyperk.csv** and ***flagtemplate.txt** must be in the same directory
(get from the **calibration** directory)

```
_rethrow_casa_exceptions = True
h_init()
try:
    hsd_importdata(vis=['uid___A002_X839000_X2a15'])
    hsd_flagdata(pipelinemode="automatic")
    hifa_tsyscal(pipelinemode="automatic")
    hifa_tsysflag(pipelinemode="automatic")
    hsd_mstoscantable(pipelinemode="automatic")
    hsd_inspectdata(pipelinemode="automatic")
    hsd_calsky(pipelinemode="automatic")
    hsd_applycal(pipelinemode="automatic")
    hsd_baseline(pipelinemode="automatic")
    hsd_blflag(pipelinemode="automatic")
    hsd_baseline(pipelinemode="automatic")
    hsd_blflag(pipelinemode="automatic")
    hsd_imaging(pipelinemode="automatic")
    hsd_exportdata(pipelinemode="automatic")
finally:
    h_save()
```

Single Dish Pipeline Cycle1,2 data

casa_pipescript.py
(get from the **script** directory**)**

To reproduce QA2 Reduction: ***jyperk.csv** and ***flagtemplate.txt** must be in the same directory
(get from the **calibration** directory)

```
_rethrow_casa_exceptions = True
h_init()
try:
    hsd_importdata(vis=['uid___A002_X839000_X2a15'])
    hsd_flagdata(pipelinemode="automatic")
    hifa_tsyscal(pipelinemode="automatic")
    hifa_tsysflag(pipelinemode="automatic")
    hsd_mstoscantable(pipelinemode="automatic")
    hsd_inspectdata(pipelinemode="automatic")
    hsd_calsky(pipelinemode="automatic")
    hsd_applycal(pipelinemode="automatic")
    hsd_simplescale(pipelinemode="automatic")
    hsd_baseline(pipelinemode="automatic")
    hsd_blflag(pipelinemode="automatic")
    hsd_baseline(pipelinemode="automatic")
    hsd_blflag(pipelinemode="automatic")
    hsd_imaging(pipelinemode="automatic")
    hsd_exportdata(pipelinemode="automatic")
finally:
    h_save()
```

Since Single Dish Pipeline is optimized for Cycle3 data, some Cycle1 & 2 data needs a few tweaks after importdata.

Using the casa_pipescript.py (1)

- To re-create the calibration performed by the ALMA Observatory: move the **ALMA raw data**, the **casa_pipescript.py**, the **flux.csv** **(for Interferometry data),** the **jyperk.csv** **(for TP data)** and ***flagtemplate.txt** into the same directory

- **flux.csv** contains **calibrator fluxes**. The flux calibrator values may have been edited during the ALMA QA2 process

- **jyperk.csv** contains **Jy per Kelvin conversion factors**. The factors are obtained from the AmpCal data belonging to the same project.

- ***flagtemplate.txt** files contain any **additional flags** added to the PL run manually during QA2

Using the casa_pipescript.py (2)

- If **flux.csv** and ***flagtemplate.txt** are not present then PL will still run, but it will create new versions of these files without any editing that was performed during QA2
- [**TP data**] If **jyperk.csv** is not present then PL will still run, but it does not convert the unit of the final images from Kelvin to Jy/beam.
- Finally, if the rawdata has the suffix ***asdm.sdm** then this suffix should be **removed** to use the script without editing
- To run with default values:
  - **casapy –pipeline**
  - **execfile('casa_pipescript.py') (for Interferometry data/TP data)**

A note on the pipelinemode

- You will notice in the script that **pipelinemode='automatic'** is given explicitly in each task
- This is to prevent the brackets from being empty, they mustn't be empty
- But the pipelinemode is important. To see all the parameters that can be changed in **pipelinemode='automatic'**, try typing e.g. **inp hifa_importdata**
- Then type **pipelinemode='interactive'** before doing the **inp** again
- *Many more PL parameters can be altered by changing the pipelinemode*

Pipeline Mode


```
CASA <12>: inp hifa_importdata
---------> inp(hifa_importdata)
#  hifa_importdata :: Imports data into the interferometry pipeline
vis                 =       ['']        #  List of input visibility data
session             =       ['']        #  List of visibility data sessions
pipelinemode        = 'automatic'       #  The pipeline operating mode
async               =     False         #  If true the taskname must be started using hifa_importdata(...)
```

```
CASA <13>: pipelinemode='interactive'

CASA <14>: inp hifa_importdata
---------> inp(hifa_importdata)
#  hifa_importdata :: Imports data into the interferometry pipeline
vis                 =       ['']        #  List of input visibility data
session             =       ['']        #  List of visibility data sessions
pipelinemode        = 'interactive'     #  The pipeline operating mode
     asis           = 'Antenna Station Receiver CalAtmosphere' #  ASDM to convert as is
     process_caldevice =      False     #  Import the caldevice table from the ASDM
     overwrite      =      False        #  Overwrite existing files on import
     bdfflags       =      False        #  Apply BDF flags on import
     dryrun         =      False        #  Run the task (False) or display task command (True)
     acceptresults  =       True        #  Add the results into the pipeline context

async               =     False         #  If true the taskname must be started using hifa_importdata(...)

CASA <15>:
```

Tweaking the script: (1) plotlevel

- PL has 3 plotlevels:
  - **default** : generates all plots apart from for the hif_applycal task.
  - **summary** : omits detail plots (e.g. hifa_timegaincal).
  - **all** : generates all plots.

- Plotting is a major overhead in the current PL

- To change from the 'default', edit:
  - **h_init()** →
  - **h_init(pipelinemode='interactive',plotlevel='summary')**

Tweaking the script: (2) data

- To change the ALMA raw datasets that need to be calibrated, edit the **vis** and **session** list in **hifa_importdata** e.g.

- **hifa_importdata(vis=['uid___A002_X72e960_X53d', 'uid___A002_X72e960_X53f'], session=['session_1', 'session_1']) →**

- **hifa_importdata(vis=['uid___A002_X72e960_X53d'], session=['session_1'])**

Tweaking the script: (3) editing the fluxscale

- If there was a QSO flux calibrator, the flux used for it is given in the **flux.csv** file

```
ms,field,spw,I,Q,U,V,comment
uid___A002_X9fddd8_X2172.ms,0,17,2.9741650916,0.0,0.0,0.0,"intent=ATMOSPHER
uid___A002_X9fddd8_X2172.ms,0,19,2.9741650916,0.0,0.0,0.0,"intent=ATMOSPHER
uid___A002_X9fddd8_X2172.ms,0,21,2.9741650916,0.0,0.0,0.0,"intent=ATMOSPHER
uid___A002_X9fddd8_X2172.ms,0,23,2.9741650916,0.0,0.0,0.0,"intent=ATMOSPHER
uid___A002_X9fddd8_X2172.ms,0,25,2.9741650916,0.0,0.0,0.0,"intent=ATMOSPHER
```

- **The easiest way to change the flux used by PL is to edit this file**

- Although the fluxes of other calibrators may be listed in this file, there is no point in editing them as they are derived during PL processing

Tweaking the script: (4) adding additional flagging

- This can be added to the ***flagtemplate.txt** files
- Parameters are the same as the CASA task ***flagdata*** e.g.
  - mode=manual spw='25:0~3;122~127' timerange='2013/01/31/08:09:55.248~2013/01/31/08:10:01.296' reason='stage8_2'
- No white space should be in the parameter settings, only between parameters .e.g.
  - mode='manual' antenna='DV07,DV09' **OK**
  - mode='manual' antenna='DV07, DV09' **NOT OK**
- To flag auto-correlation data (for TP data) .e.g.
  - mode='manual' antenna='PM02**&&&**' (auto-correlation)
  - mode='manual' antenna='PM02**&&***' (cross-&auto-correlation)

Tweaking the script: (5) select a reference antenna

- By default Pipeline creates a ranked list of reference antennas

- This can be over-ridden to select a reference antenna

- **hif_refant(pipelinemode='automatic')** → e.g. **hif_refant(pipelinemode='automatic', hm_refant='manual', refant='DV15')**

Tweaking the script: (6) removing/ limiting calibrator imaging

- By default Pipeline creates images of the bandpass and phase calibrators and any check source (1 per spectral window)

- To e.g. image the phase calibrator only change
- **hif_makecleanlist(intent='PHASE,BANDPASS,CHECK')** →
- **hif_makecleanlist(intent='PHASE')**

- **To remove any calibrator imaging, then delete the lines:**
- hif_makecleanlist(intent='PHASE,BANDPASS,CHECK')
- hif_cleanlist(pipelinemode="automatic")

Tweaking the script: (7) changing reference amplitude calibrator

- By default Pipeline set fluxes for calibrator with AMPLITUDE intent by hif_setjy. The flux densities to be set in flux.csv are used or he reference flux densities, if QSO.

- To e.g. use BANDPASS as flux calibrator.
- **hif_setjy(pipelinemode="automatic”)** →
- **hif_setjy(pipelinemode="interactive",field="J1337-1257",intent="*BANDPASS*")**

- **hifa_gfluxscale(pipelinemode="automatic")** →
- **hifa_gfluxscale(pipelinemode="interactive",reference="J1337-1257",transfer="J1517-243, J1625-2527", refintent="*BANDPASS*",transintent="*AMPLITUDE*,*PHASE*")**

Save context

- Pipeline reduces data automatically by
  – Selecting the best processing strategies → **Heuristics**
  – Organizing the reduction environment / book-keeping → **Context**

- For diagnostics, check the WebLog after you run each task. Pipeline creates WebLog each task soon after completing the task.

- Handling context:
  – Pipeline always run with a context
    h_init()

  – Do not forget to save context before closing CASA.
    h_save('A.context')

  – To start again but from with a specific task:
    h_resume('A.context')

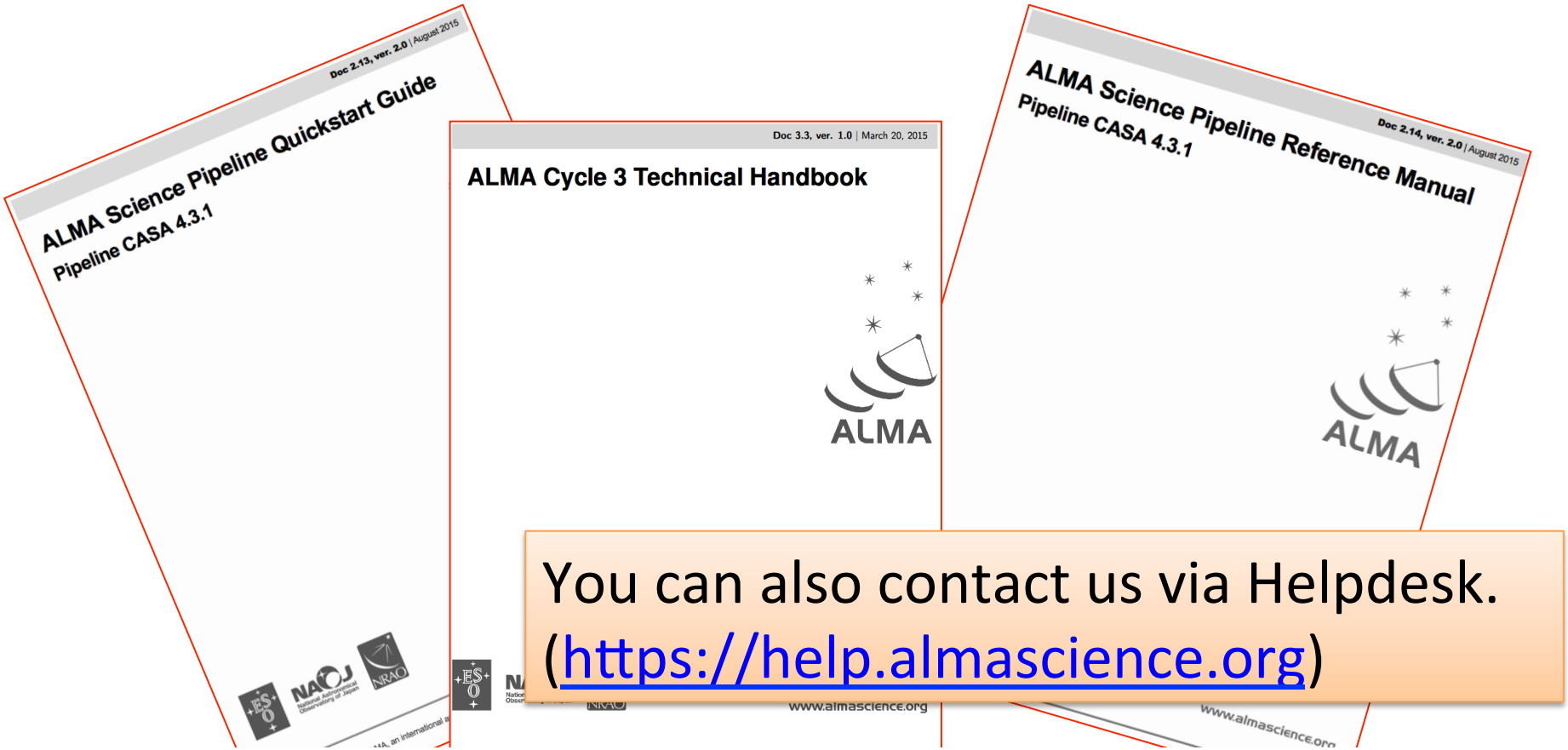
ALMA Pipeline

- Pipeline **CASA 4.5** commissioned for at least:
  - 12m array calibration (<= Band 7)
  - 7m array calibration (<= Band 7)
  - Single-dish end-to-end processing

- ALMA Data Deliveries
  - (1) Keep the same calibration as Observatory
    - Use **scriptForPI.py** to apply calibration tables (Faster and recommended! for Interferometry data. But slow for TP data).
  - (2) Alter the calibration
    - Use **casapipescript.py** (Slower)

ALMA Pipeline Information

- Available from the ALMA Science Portal
  - http://www.almascience.org



You can also contact us via Helpdesk. (https://help.almascience.org)

Appendix

Overview

- ALMA have CASA Pipelines
- Currently calibration only
  – Diagnostic calibrator images
  – ALMA science target imaging being commissioned
- The Pipelines use dedicated Pipeline tasks in CASA
- Execute using python scripts provided by the telescopes. Or self-build
- ALMA Pipelines common output: Pipeline WebLog

CASA Pipeline Versions

- CASA 4.2.2 and CASA 4.3.1 have special versions including the Pipeline
  – Obtain from http://casa.nrao.edu/casa_obtaining.shtml
- CASA 4.4 has no pipeline version
- Starting with CASA 4.5 (this month), one version
- Earliest CASA versions to be used with ALMA Cycle 3 data (manual or Pipeline) CASA 4.5

Pipeline Task Types

Prefix	Task Type	Purpose
h_	Common	Interferometry and single-dish ALMA & VLA
hif_	Interferometry	ALMA & VLA
hifa_	Interferometry	ALMA only
hifv_	Interferometry	VLA only
hsd_	Single-dish	ALMA single-dish

Pipeline Tasks vs CASA Tasks

Pipeline	CASA
hifa_importdata	importasdm
hifa_flagdata	flagdata
hifa_wvrgcalflag	wvrgcal
hifa_bandpass	bandpass
hifa_gfluxscale	fluxscale
higa_timpegaincal	gaincal
hif_applycal	applycal

Pipeline tasks use CASA tasks where possible.

Pipeline tasks can also contain heuristics and may be multiple CASA tasks bundled together.

Pipeline CASA Commands Log

```
# hifa_bandpass(pipelinemode="automatic")
#
# The spectral response of each antenna is calibrated. A short-solint phase
# gain is calculated to remove decorrelation of the bandpass calibrator before
# the bandpass is calculated.
#

gaincal(field='0', minblperant=4, antenna='0~36', solint='4.502099s',
      caltable='uid___A002_Xa43a0e_X115e.ms.hifa_bandpass.s11_3.spw9_11_13_15.solint4_502s.gpcal.tbl',
      interp=['linear,linear', 'nearest'], minsnr=3.0,
      gaintable=['uid___A002_Xa43a0e_X115e.ms.hifa_tsyscal.s6_1.tsyscal.tbl',
      'uid___A002_Xa43a0e_X115e.ms.hifa_wvrgcalflag.s8_4.sm2_016s.wvrcal.tbl'],
      spw='9,11,13,15', vis='uid___A002_Xa43a0e_X115e.ms', calmode='p',
      gaintype='G', spwmap=[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 11, 11, 13, 13,
      15, 15, 9, 11, 13, 15], []], intent='*BANDPASS*', solnorm=False,
    refant='DA49,DA59,DV08,DA57,PM02,DV18,DV19,PM03,DV04,DA41,PM01,DA61,DA63,DA53,DV01,etc',
      gainfield=['nearest', ''])


bandpass(field='0', bandtype='B', antenna='0~36', solint='inf,164.524257MHz',
       caltable='uid___A002_Xa43a0e_X115e.ms.hifa_bandpass.s11_3.spw9_11_13_15.channel.solintinf.bcal.tbl',
       interp=['linear,linear', 'nearest', 'linear,linear'], minsnr=3.0,
       gaintable=['uid___A002_Xa43a0e_X115e.ms.hifa_tsyscal.s6_1.tsyscal.tbl',
       'uid___A002_Xa43a0e_X115e.ms.hifa_wvrgcalflag.s8_4.sm2_016s.wvrcal.tbl',
       'uid___A002_Xa43a0e_X115e.ms.hifa_bandpass.s11_3.spw9_11_13_15.solint4_502s.gpcal.tbl'],
       spw='9', vis='uid___A002_Xa43a0e_X115e.ms', combine='scan', spwmap=[[0,
       1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 11, 11, 13, 13, 15, 15, 9, 11, 13, 15],
       [], []], intent='*BANDPASS*', solnorm=True,
       refant='DA49,DA59,DV08,DA57,PM02,DV18,DV19,PM03,DV04,DA41,PM01,DA61,DA63,DA53,DV01,etc',
       gainfield=['nearest', '', 'nearest'], minblperant=4)
```


Pipeline tasks use CASA tasks where possible.

Pipeline Design

- Pipeline reduces data automatically by
  – Selecting the best processing strategies → **Heuristics**
  – Organizing the reduction environment / book-keeping → **Context**

- • ALMA pipeline implements the two aspects using the “separation of concerns” design principle
- – Mix and match of steps is possible

Pipeline Implementation

- **Heuristics** and **Context** are implemented as Python Classes
- The variables for heuristics are “parameters”, the ones for context are “inputs”
- Pipeline runs handle the context automatically
- User interaction mainly via heuristics parameters
- However the context can be edited to insert own calibration tables

Appendix: Split SPW of unwanted Delaycal after import data (Cycle1 and 2 data)

- Cycle 1 and 2 data usually have delay cal (check source) SPWs. In case that science SPWs are not identical, it is needed to split the MS to exclude delay cal SPWs after importing data to properly process in Single Dish Pipeline.

```
CASA<1>:h_init()
CASA<2>:hsd_importdata(vis=['uid***'])
CASA<3>:h_save()
CASA<4>:os.system('mv uid***.ms uid***.ms.org')
CASA<5>:split2(vis='uid***.ms.org', outputvis='uid***.ms.split',
                    spw='0~8,25~40', datacolumn='all',keepflags=T)
CASA<6>:os.system('cp –rf uid***.ms.split uid***.ms')
CASA<7>: h_init()
CASA<8>:hsd_importdata(vis=['uid***.ms'])
….
```