```
##############################################################################
#                                                                            #
# Use Case Script for NGC 5921                                               #
#                                                                            #
# Converted by STM 2007-05-26                                                #
# Updated      STM 2007-06-15 (Alpha Patch 1)                                #
# Updated      STM 2007-09-05 (Alpha Patch 2+)                               #
# Updated      STM 2007-09-18 (Alpha Patch 2+)                               #
# Updated      STM 2007-09-18 (Pre-Beta) add immoments                       #
# Updated      STM 2007-10-04 (Beta) update                                  #
# Updated      STM 2007-10-10 (Beta) add export                              #
# Updated      STM 2007-11-08 (Beta Patch 0.5) add RRusk stuff               #
# Updated      STM 2008-03-25 (Beta Patch 1.0)                               #
# Updated      STM 2008-05-23 (Beta Patch 2.0) new tasking/clean/cal         #
# Updated      STM 2008-06-11 (Beta Patch 2.0)                               #
# Updated      STM 2008-06-13 (Beta Patch 2.0) demo version                  #
# Updated      STM 2008-06-14 (Beta Patch 2.0) post-school update            #
# Updated      STM 2008-07-06 (Beta Patch 2.0) regression version            #
#                                                                            #
# Features Tested:                                                           #
#    The script illustrates end-to-end processing with CASA                  #
#    as depicted in the following flow-chart.                                #
#                                                                            #
#    Filenames will have the <prefix> = 'ngc5921.usecase'                    #
#                                                                            #
#    Input Data           Process          Output Data                       #
#                                                                            #
#   NGC5921.fits --> importuvfits  -->  <prefix>.ms   +                      #
#   (1.4GHz,               |            <prefix>.ms.flagversions             #
#    63 sp chan,           v                                                 #
#    D-array)           listobs    -->  casapy.log                           #
#                          |                                                 #
#                          v                                                 #
#                     flagautocorr                                           #
#                          |                                                 #
#                          v                                                 #
#                        setjy                                               #
#                          |                                                 #
#                          v                                                 #
#                       bandpass   -->  <prefix>.bcal                        #
#                          |                                                 #
#                          v                                                 #
#                       gaincal    -->  <prefix>.gcal                        #
#                          |                                                 #
#                          v                                                 #
#                      fluxscale   -->  <prefix>.fluxscale                   #
#                          |                                                 #
#                          v                                                 #
#                      applycal    -->  <prefix>.ms                          #
#                          |                                                 #
#                          v                                                 #
#                        split     -->  <prefix>.cal.split.ms                #
#                          |                                                 #
#                          v                                                 #
#                        split     -->  <prefix>.src.split.ms                #
#                          |                                                 #
#                          v                                                 #
#                    exportuvfits  -->  <prefix>.split.uvfits                #
#                          |                                                 #
#                          v                                                 #
#                      uvcontsub   -->  <prefix>.ms.cont +                   #
#                          |            <prefix>.ms.contsub                  #
#                          v                                                 #
#                        clean     -->  <prefix>.clean.image +               #
#                          |            <prefix>.clean.model +               #
#                          |            <prefix>.clean.residual              #
#                          v                                                 #
#                      exportfits  -->  <prefix>.clean.fits                  #
#                          |                                                 #
#                          v                                                 #
#                        imhead    -->  casapy.log                           #
#                          |                                                 #
#                          v                                                 #
#                        imstat    -->  xstat (parameter)                    #
#                          |                                                 #
#                          v                                                 #
#                      immoments   -->  <prefix>.moments.integrated +        #
#                          |            <prefix>.moments.weighted_coord      #
#                          v                                                 #
##############################################################################

import time
import os

scriptmode = True
#
# Set up some useful variables
#
# Get to path to the CASA home and stip off the name
pathname=os.environ.get('CASAPATH').split()[0]

# This is where the NGC5921 UVFITS data will be
fitsdata=pathname+'/data/demo/NGC5921.fits'
#
# Or use data in current directory
#fitsdata='NGC5921.fits'

# The prefix to use for all output files
prefix='ngc5921.demo'

# Clean up old files
os.system('rm -rf '+prefix+'*')

#
#=====================================================================
#
# Import the data from FITS to MS
#
print '--Import--'

# Safest to start from task defaults
default('importuvfits')

# Set up the MS filename and save as new global variable
msfile = prefix + '.ms'

# Use task importuvfits
fitsfile = fitsdata
vis = msfile

saveinputs('importuvfits',prefix+'.importuvfits.saved')

importuvfits()

#
# Note that there will be a ngc5921.usecase.ms.flagversions
```

```
# there containing the initial flags as backup for the main ms
# flags.
#
#==========================================================================
#
# List a summary of the MS
#
print '--Listobs--'

# Don't default this one and make use of the previous setting of
# vis.  Remember, the variables are GLOBAL!

# You may wish to see more detailed information, like the scans.
# In this case use the verbose = True option
verbose = True

listobs()

# You should get in your logger window and in the casapy.log file
# something like:
#
# MeasurementSet Name:/home/sandrock2/smyers/Testing2/Sep07/ngc5921.usecase
.ms
# MS Version 2
#
# Observer: TEST     Project:
# Observation: VLA
#
# Data records: 22653       Total integration time = 5280 seconds
#    Observed from   09:19:00   to   10:47:00
#
#    ObservationID = 0         ArrayID = 0
#   Date        Timerange                Scan  FldId FieldName      SpwIds
#   13-Apr-1995/09:19:00.0 - 09:24:30.0     1      0 1331+30500002_0  [0]
#               09:27:30.0 - 09:29:30.0     2      1 1445+09900002_0  [0]
#               09:33:00.0 - 09:48:00.0     3      2 N5921_2        [0]
#               09:50:30.0 - 09:51:00.0     4      1 1445+09900002_0  [0]
#               10:22:00.0 - 10:23:00.0     5      1 1445+09900002_0  [0]
#               10:26:00.0 - 10:43:00.0     6      2 N5921_2        [0]
#               10:45:30.0 - 10:47:00.0     7      1 1445+09900002_0  [0]
#
# Fields: 3
#   ID   Code Name          Right Ascension  Declination   Epoch
#   0    C    1331+30500002_013:31:08.29      +30.30.32.96  J2000
#   1    A    1445+09900002_014:45:16.47      +09.58.36.07  J2000
#   2         N5921_2       15:22:00.00      +05.04.00.00  J2000
#
# Spectral Windows:(1 unique spectral windows and 1 unique polarization set
ups)
#   SpwID  #Chans Frame Ch1(MHz)    Resoln(kHz) TotBW(kHz)  Ref(MHz)    Cor
rs
#   0          63 LSRK  1412.68608  24.4140625  1550.19688  1413.44902  RR
 LL
#
# Feeds: 28: printing first row only
#   Antenna   Spectral Window     # Receptors    Polarizations
#   1         -1                  2              [         R, L]
#
# Antennas: 27:
#   ID   Name  Station   Diam.    Long.          Lat.
#   0    1     VLA:N7    25.0 m   -107.37.07.2  +33.54.12.9
#   1    2     VLA:W1    25.0 m   -107.37.05.9  +33.54.00.5
#   2    3     VLA:W2    25.0 m   -107.37.07.4  +33.54.00.9
#   3    4     VLA:E1    25.0 m   -107.37.05.7  +33.53.59.2
#   4    5     VLA:E3    25.0 m   -107.37.02.8  +33.54.00.5
#   5    6     VLA:E9    25.0 m   -107.36.45.1  +33.53.53.6
#   6    7     VLA:E6    25.0 m   -107.36.55.6  +33.53.57.7
#   7    8     VLA:W8    25.0 m   -107.37.21.6  +33.53.53.0
#   8    9     VLA:N5    25.0 m   -107.37.06.7  +33.54.08.0
#   9    10    VLA:W3    25.0 m   -107.37.08.9  +33.54.00.1
#   10   11    VLA:N4    25.0 m   -107.37.06.5  +33.54.06.1
#   11   12    VLA:W5    25.0 m   -107.37.13.0  +33.53.57.8
#   12   13    VLA:N3    25.0 m   -107.37.06.3  +33.54.04.8
#   13   14    VLA:N1    25.0 m   -107.37.06.0  +33.54.01.8
#   14   15    VLA:N2    25.0 m   -107.37.06.2  +33.54.03.5
#   15   16    VLA:E7    25.0 m   -107.36.52.4  +33.53.56.5
#   16   17    VLA:E8    25.0 m   -107.36.48.9  +33.53.55.1
#   17   18    VLA:W4    25.0 m   -107.37.10.8  +33.53.59.1
#   18   19    VLA:E5    25.0 m   -107.36.58.4  +33.53.58.8
#   19   20    VLA:W9    25.0 m   -107.37.25.1  +33.53.51.0
#   20   21    VLA:W6    25.0 m   -107.37.15.6  +33.53.56.4
#   21   22    VLA:E4    25.0 m   -107.37.00.8  +33.53.59.7
#   23   24    VLA:E2    25.0 m   -107.37.04.4  +33.54.01.1
#   24   25    VLA:N6    25.0 m   -107.37.06.9  +33.54.10.3
#   25   26    VLA:N9    25.0 m   -107.37.07.8  +33.54.19.0
#   26   27    VLA:N8    25.0 m   -107.37.07.5  +33.54.15.8
#   27   28    VLA:W7    25.0 m   -107.37.18.4  +33.53.54.8
#
# Tables:
#    MAIN                   22653 rows
#    ANTENNA                   28 rows
#    DATA_DESCRIPTION           1 row
#    DOPPLER             <absent>
#    FEED                      28 rows
#    FIELD                      3 rows
#    FLAG_CMD             <empty>
#    FREQ_OFFSET         <absent>
#    HISTORY                  273 rows
#    OBSERVATION                1 row
#    POINTING                 168 rows
#    POLARIZATION               1 row
#    PROCESSOR            <empty>
#    SOURCE                     3 rows
#    SPECTRAL_WINDOW            1 row
#    STATE                <empty>
#    SYSCAL              <absent>
#    WEATHER             <absent>
#
#
#==========================================================================
#
# Get rid of the autocorrelations from the MS
#
print '--Flagautocorr--'

# Don't default this one either, there is only one parameter (vis)

flagautocorr()

#
#==========================================================================
#
# Set the fluxes of the primary calibrator(s)
#
print '--Setjy--'
default('setjy')
```

```
vis = msfile

#
# 1331+305 = 3C286 is our primary calibrator
# Use the wildcard on the end of the source name
# since the field names in the MS have inherited the
# AIPS qualifiers
field = '1331+305*'

# This is 1.4GHz D-config and 1331+305 is sufficiently unresolved
# that we dont need a model image.  For higher frequencies
# (particularly in A and B config) you would want to use one.
modimage = ''

# Setjy knows about this source so we dont need anything more

saveinputs('setjy',prefix+'.setjy.saved')

# Pause script if you are running in scriptmode
if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

setjy()

#
# You should see something like this in the logger and casapy.log file:
#
# 1331+30500002_0  spwid=  0  [I=14.76, Q=0, U=0, V=0] Jy, (Perley-Taylor 9
9)
#
# So its using 14.76Jy as the flux of 1331+305 in the single Spectral Windo
w
# in this MS.
#
#=====================================================================
#
# Bandpass calibration
#
print '--Bandpass--'
default('bandpass')

# We can first do the bandpass on the single 5min scan on 1331+305
# At 1.4GHz phase stablility should be sufficient to do this without
# a first (rough) gain calibration.  This will give us the relative
# antenna gain as a function of frequency.

vis = msfile

# set the name for the output bandpass caltable
btable = prefix + '.bcal'
caltable = btable

# No gain tables yet
gaintable = ''
gainfield = ''
interp = ''

# Use flux calibrator 1331+305 = 3C286 (FIELD_ID 0) as bandpass calibrator
field = '0'
# all channels
spw = ''
```

```
# No other selection
selectdata = False

# In this band we do not need a-priori corrections for
# antenna gain-elevation curve or atmospheric opacity
# (at 8GHz and above you would want these)
gaincurve = False
opacity = 0.0

# Choose bandpass solution type
# Pick standard time-binned B (rather than BPOLY)
bandtype = 'B'

# set solution interval arbitrarily long (get single bpass)
solint = 'inf'
combine = 'scan'

# reference antenna Name 15 (15=VLA:N2) (Id 14)
refant = '15'

saveinputs('bandpass',prefix+'.bandpass.saved')

# Pause script if you are running in scriptmode
if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

bandpass()

#
#=====================================================================
#
# Use plotcal to examine the bandpass solutions
#
print '--Plotcal (bandpass)--'
default('plotcal')

caltable = btable
field = '0'

# Set up 2x1 panels - upper panel amp vs. channel
subplot = 211
yaxis = 'amp'
# No output file yet (wait to plot next panel)

saveinputs('plotcal',prefix+'.plotcal.b.amp.saved')

if scriptmode:
    showgui = True
else:
    showgui = False

plotcal()
#
# Set up 2x1 panels - lower panel phase vs. channel
subplot = 212
yaxis = 'phase'

saveinputs('plotcal',prefix+'.plotcal.b.phase.saved')

#
# Note the rolloff in the start and end channels.  Looks like
# channels 6-56 (out of 0-62) are the best
```

```
# Pause script if you are running in scriptmode
if scriptmode:
    # If you want to do this interactively and iterate over antenna, set
    # iteration = 'antenna'
    showgui = True
    plotcal()
    user_check=raw_input('Return to continue script\n')
else:
    # No GUI for this script
    showgui = False
    # Now send final plot to file in PNG format (via .png suffix)
    figfile = caltable + '.plotcal.png'
    plotcal()

#=====================================================================
#
# Gain calibration
#
print '--Gaincal--'
default('gaincal')

# Armed with the bandpass, we now solve for the
# time-dependent antenna gains

vis = msfile

# set the name for the output gain caltable
gtable = prefix + '.gcal'
caltable = gtable

# Use our previously determined bandpass
# Note this will automatically be applied to all sources
# not just the one used to determine the bandpass
gaintable = btable
gainfield = ''

# Use nearest (there is only one bandpass entry)
interp = 'nearest'

# Gain calibrators are 1331+305 and 1445+099 (FIELD_ID 0 and 1)
field = '0,1'

# We have only a single spectral window (SPW 0)
# Choose 51 channels 6-56 out of the 63
# to avoid end effects.
# Channel selection is done inside spw
spw = '0:6~56'

# No other selection
selectdata = False

# In this band we do not need a-priori corrections for
# antenna gain-elevation curve or atmospheric opacity
# (at 8GHz and above you would want these)
gaincurve = False
opacity = 0.0

# scan-based G solutions for both amplitude and phase
gaintype = 'G'
solint = 'inf'
combine = ''
calmode = 'ap'
```

```
# minimum SNR allowed
minsnr = 1.0

# reference antenna 15 (15=VLA:N2)
refant = '15'

saveinputs('gaincal',prefix+'.gaincal.saved')

if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

gaincal()

#
#=====================================================================
#
# Bootstrap flux scale
#
print '--Fluxscale--'
default('fluxscale')

vis = msfile

# set the name for the output rescaled caltable
ftable = prefix + '.fluxscale'
fluxtable = ftable

# point to our first gain cal table
caltable = gtable

# we will be using 1331+305 (the source we did setjy on) as
# our flux standard reference - note its extended name as in
# the FIELD table summary above (it has a VLA seq number appended)
reference = '1331*'

# we want to transfer the flux to our other gain cal source 1445+099
transfer = '1445*'

saveinputs('fluxscale',prefix+'.fluxscale.saved')

# Pause script if you are running in scriptmode
if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

fluxscale()

# In the logger you should see something like:
# Flux density for 1445+09900002_0 in SpW=0 is:
#     2.48576 +/- 0.00123122 (SNR = 2018.94, nAnt= 27)

# If you run plotcal() on the tablein = 'ngc5921.usecase.fluxscale'
# you will see now it has brought the amplitudes in line between
# the first scan on 1331+305 and the others on 1445+099

#
#=====================================================================
#
# Now use plotcal to examine the gain solutions
#
print '--Plotcal (fluxscaled gains)--'
```

```
default('plotcal')

caltable = ftable
field = '0,1'

# Set up 2x1 panels - upper panel amp vs. time
subplot = 211
yaxis = 'amp'
# No output file yet (wait to plot next panel)

saveinputs('plotcal',prefix+'.plotcal.gscaled.amp.saved')

if scriptmode:
    showgui = True
else:
    showgui = False

plotcal()
#
# Set up 2x1 panels - lower panel phase vs. time
subplot = 212
yaxis = 'phase'

saveinputs('plotcal',prefix+'.plotcal.gscaled.phase.saved')

#
# The amp and phase coherence looks good

# Pause script if you are running in scriptmode
if scriptmode:
    # If you want to do this interactively and iterate over antenna, set
    #iteration = 'antenna'
    showgui = True
    plotcal()
    user_check=raw_input('Return to continue script\n')
else:
    # No GUI for this script
    showgui = False
    # Now send final plot to file in PNG format (via .png suffix)
    figfile = caltable + '.plotcal.png'
    plotcal()

#=====================================================================
#
# Apply our calibration solutions to the data
# (This will put calibrated data into the CORRECTED_DATA column)
#
print '--ApplyCal--'
default('applycal')

vis = msfile

# We want to correct the calibrators using themselves
# and transfer from 1445+099 to itself and the target N5921

# Start with the fluxscale/gain and bandpass tables
gaintable = [ftable,btable]

# pick the 1445+099 out of the gain table for transfer
# use all of the bandpass table
gainfield = ['1','*']

# interpolation using linear for gain, nearest for bandpass
interp = ['linear','nearest']

# only one spw, do not need mapping
spwmap = []

# all channels
spw = ''
selectdata = False

# as before
gaincurve = False
opacity = 0.0

# select the fields for 1445+099 and N5921
field = '1,2'

applycal()

# Now for completeness apply 1331+305 to itself

field = '0'
gainfield = ['0','*']

# The CORRECTED_DATA column now contains the calibrated visibilities

saveinputs('applycal',prefix+'.applycal.saved')

# Pause script if you are running in scriptmode
if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

applycal()

#
#=====================================================================
#
# Now use plotxy to plot the calibrated target data (before contsub)
#
print '--Plotxy (NGC5921)--'
default('plotxy')

vis = msfile

field = '2'
# Edge channels are bad
spw = '0:4~59'

# Time average across scans
timebin = '86000.'
crossscans = True

# Set up 2x1 panels - upper panel amp vs. channel
subplot = 211
xaxis = 'channel'
yaxis = 'amp'
datacolumn = 'corrected'
# No output file yet (wait to plot next panel)

saveinputs('plotxy',prefix+'.plotxy.final.amp.saved')

print "Amp averaged across time and baseline (upper)"
```

```
figfile = ''
if scriptmode:
    interactive = True
else:
    interactive = False

plotxy()
#
# Set up 2x1 panels - lower panel phase vs. time
subplot = 212
yaxis = 'phase'
datacolumn = 'corrected'
# Time average across scans and baselines
timebin = '86000.'
crossscans = True
crossbls = True

saveinputs('plotxy',prefix+'.plotxy.final.phase.saved')

print "Phase averaged across time and baseline (lower)"

print "Final calibrated data"

# Pause script if you are running in scriptmode
if scriptmode:
    interactive = True
    figfile = ''
    plotxy()
    user_check=raw_input('Return to continue script\n')
else:
    interactive = False
    # Now send final plot to file in PNG format (via .png suffix)
    figfile = vis + '.plotxy.png'
    plotxy()


#=====================================================================
#
# Split the sources out, pick off the CORRECTED_DATA column
#
#
# Split NGC5921 data (before continuum subtraction)
#
print '--Split NGC5921 Data--'
default('split')

vis = msfile
splitms = prefix + '.src.split.ms'
outputvis = splitms
field = 'N5921*'
spw = ''
datacolumn = 'corrected'

saveinputs('split',prefix+'.split.n5921.saved')

split()

print "Created "+splitms

# If you want, split out the calibrater 1445+099 field, all chans
#print '--Split 1445+099 Data--'
#
#calsplitms = prefix + '.cal.split.ms'
```

```
#outputvis = calsplitms
#field = '1445*'
#
#saveinputs('split',prefix+'.split.1445.saved')
#
#split()

#=====================================================================
#
# Here is how to export the NGC5921 data as UVFITS
# Start with the split file.
# Since this is a split dataset, the calibrated data is
# in the DATA column already.
# Write as a multisource UVFITS (with SU table)
# even though it will have only one field in it
# Run asynchronously so as not to interfere with other tasks
# (BETA: also avoids crash on next importuvfits)
#
#print '--Export UVFITS--'
#default('exportuvfits')
#
#srcuvfits = prefix + '.split.uvfits'
#
#vis = splitms
#fitsfile = srcuvfits
#datacolumn = 'data'
#multisource = True
#async = True
#
#saveinputs('exportuvfits',prefix+'.exportuvfits.saved')
#
#myhandle = exportuvfits()
#
#print "The return value for this exportuvfits async task for tm is
# "+str(myhandle)

#=====================================================================
#
# UV-plane continuum subtraction on the target
# use the split ms
# (this will update the CORRECTED_DATA column)
#
print '--UV Continuum Subtract--'
default('uvcontsub')

vis = splitms

field = 'N5921*'
# Use channels 4-6 and 50-59 for continuum
fitspw='0:4~6;50~59'

# Output all of spw 0
spw = '0'

# Averaging time (none)
solint = 0.0

# Fit only a mean level
fitorder = 0

# Do the uv-plane subtraction
fitmode = 'subtract'
```

```
# Let it split out the data automatically for us
splitdata = True

saveinputs('uvcontsub',prefix+'.uvcontsub.saved')

# Pause script if you are running in scriptmode
if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

uvcontsub()

# You will see it made two new MS:
# <vis>.cont
# <vis>.contsub

srcsplitms = vis + '.contsub'

# Note that ngc5921.usecase.ms.contsub contains the uv-subtracted
# visibilities (in its DATA column), and ngc5921.usecase.ms.cont
# the pseudo-continuum visibilities (as fit).

# The original ngc5921.usecase.ms now contains the uv-continuum
# subtracted vis in its CORRECTED_DATA column and the continuum
# in its MODEL_DATA column as per the fitmode='subtract'

# Done with calibration
#======================================================================
#
# Here is how to make a dirty image cube
#
#print '--Clean (dirty image)--'
#default('clean')

# Pick up our split source continuum-subtracted data
#vis = srcsplitms
#imname = prefix + '.dirty'
#imagename = imname
#
#mode = 'channel'
#nchan = 46
#start = 5
#width = 1
#
#field = '0'
#spw = ''
#imsize = [256,256]
#cell = [15.,15.]
#weighting = 'briggs'
#robust = 0.5

# No cleaning
#niter = 0

#saveinputs('clean',prefix+'.invert.saved')

# Pause script if you are running in scriptmode
#if scriptmode:
#    inp()
#    user_check=raw_input('Return to continue script\n')
#
#clean()
```

```
#dirtyimage = imname+'.image'

# Get the dirty image cube statistics
#dirtystats = imstat(dirtyimage)

#======================================================================
#
# Now clean an image cube of N5921
#
print '--Clean (clean)--'
default('clean')

# Pick up our split source continuum-subtracted data
vis = srcsplitms

# Make an image root file name
imname = prefix + '.clean'
imagename = imname

# Set up the output image cube
mode = 'channel'
nchan = 46
start = 5
width = 1

# This is a single-source MS with one spw
field = '0'
spw = ''

# Standard gain factor 0.1
gain = 0.1

# Set the output image size and cell size (arcsec)
imsize = [256,256]

# Do a simple Clark clean
psfmode = 'clark'
# No Cotton-Schwab iterations
csclean = False

# If desired, you can do a Cotton-Schwab clean
# but will have only marginal improvement for this data
#csclean = True
# Twice as big for Cotton-Schwab (cleans inner quarter)
#imsize = [512,512]

# Pixel size 15 arcsec for this data (1/3 of 45" beam)
# VLA D-config L-band
cell = [15.,15.]

# Fix maximum number of iterations
niter = 6000

# Also set flux residual threshold (in mJy)
threshold=8.0

# Set up the weighting
# Use Briggs weighting (a moderate value, on the uniform side)
weighting = 'briggs'
robust = 0.5

# Set a cleanbox +/-20 pixels around the center 128,128
mask = [108,108,148,148]
```

```
# But if you had a cleanbox saved in a file, e.g. "regionfile.txt"
# you could use it:
#mask='regionfile.txt'
#
# If you don't want any clean boxes or masks, then
#mask = ''

# If you want interactive clean set to True
#interactive=True
interactive=False

saveinputs('clean',prefix+'.clean.saved')

# Pause script if you are running in scriptmode
if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

clean()

# Should find stuff in the logger like:
#
# Fitted beam used in restoration: 51.5643 by 45.6021 (arcsec)
#     at pa 14.5411 (deg)
#
# It will have made the images:
# -----------------------------
# ngc5921.usecase.clean.image
# ngc5921.usecase.clean.model
# ngc5921.usecase.clean.residual
# ngc5921.usecase.clean.boxclean.mask

clnimage = imname+'.image'

#=====================================================================
#
# Done with imaging
# Now view the image cube of N5921
#
if scriptmode:
    print '--View image--'
    print "Use Spectral Profile Tool to get line profile in box in center"
    viewer(clnimage,'image')
    user_check=raw_input('Return to continue script\n')

#=====================================================================
#
# Here is how to export the Final CLEAN Image as FITS
# Run asynchronously so as not to interfere with other tasks
# (BETA: also avoids crash on next importfits)
#
#print '--Final Export CLEAN FITS--'
#default('exportfits')
#
#clnfits = prefix + '.clean.fits'
#
#imagename = clnimage
#fitsimage = clnfits
#async = True
#
#saveinputs('exportfits',prefix+'.exportfits.saved')
#
#myhandle2 = exportfits()
#
#print "The return value for this exportfits async task for tm is
#                   "+str(myhandle2)

#=====================================================================
#
# Print the image header
#
print '--Imhead--'
default('imhead')

imagename = clnimage

mode = 'summary'

imhead()

# A summary of the cube will be seen in the logger

#=====================================================================
#
# Get the cube statistics
#
print '--Imstat (cube)--'
default('imstat')

imagename = clnimage

# Do whole image
box = ''
# or you could stick to the cleanbox
#box = '108,108,148,148'

cubestats = imstat()

# Statistics will printed to the terminal, and the output
# parameter will contain a dictionary of the statistics

#=====================================================================
#
# Get some image moments
#
print '--ImMoments--'
default('immoments')

imagename = clnimage

# Do first and second moments
moments = [0,1]

# Need to mask out noisy pixels, currently done
# using hard global limits
excludepix = [-100,0.009]

# Include all planes
planes = ''

# Output root name
momfile = prefix + '.moments'
outfile = momfile

saveinputs('immoments',prefix+'.immoments.saved')
```

```
# Pause script if you are running in scriptmode
if scriptmode:
    inp()
    user_check=raw_input('Return to continue script\n')

immoments()

momzeroimage = momfile + '.integrated'
momoneimage = momfile + '.weighted_coord'

# It will have made the images:
# --------------------------------------
# ngc5921.usecase.moments.integrated
# ngc5921.usecase.moments.weighted_coord

#
#=====================================================================
#
# Get some statistics of the moment images
#
print '--Imstat (moments)--'
default('imstat')

imagename = momzeroimage
momzerostats = imstat()

imagename = momoneimage
momonestats = imstat()

#=====================================================================
#
# Now view the moments
#
if scriptmode:
    print '--View image (Moments)--'
    viewer(momzeroimage)
    print "You can add mom-1 image "+momoneimage
    print "as a contour plot"
    user_check=raw_input('Return to continue script\n')

#=====================================================================
#
# Set up an output logfile
import datetime
datestring=datetime.datetime.isoformat(datetime.datetime.today())

outfile = 'out.'+prefix+'.'+datestring+'.log'
logfile=open(outfile,'w')
print >>logfile,'Results for '+prefix+' :'
print >>logfile,""

#=====================================================================
#
# Can do some image statistics if you wish
# Treat this like a regression script
# WARNING: currently requires toolkit
#
print ' NGC5921 results '
print ' =============== '

print >>logfile,' NGC5921 results '
print >>logfile,' =============== '
```

```
#
# Use the ms tool to get max of the MSs
# Eventually should be available from a task
#
# Pull the max cal amp value out of the MS (if you split this)
#ms.open(calsplitms)
#thistest_cal = max(ms.range(["amplitude"]).get('amplitude'))
#ms.close()
#oldtest_cal = 34.0338668823
#diff_cal = abs((oldtest_cal-thistest_cal)/oldtest_cal)
#
#print ' Calibrator data ampl max = ',thistest_cal
#print '   Previous: cal data max = ',oldtest_cal
#print '   Difference (fractional) = ',diff_cal
#print ''
#
#print >>logfile,' Calibrator data ampl max = ',thistest_cal
#print >>logfile,'   Previous: cal data max = ',oldtest_cal
#print >>logfile,'   Difference (fractional) = ',diff_cal
#print >>logfile,''

# Pull the max src amp value out of the MS
ms.open(srcsplitms)
thistest_src = max(ms.range(["amplitude"]).get('amplitude'))
ms.close()
oldtest_src =  46.2060050964 # now in all chans
diff_src = abs((oldtest_src-thistest_src)/oldtest_src)

print ' Target Src data ampl max = ',thistest_src
print '  Previous: src data max = ',oldtest_src
print '  Difference (fractional) = ',diff_src
print ''

print >>logfile,' Target Src data ampl max = ',thistest_src
print >>logfile,'  Previous: src data max = ',oldtest_src
print >>logfile,'  Difference (fractional) = ',diff_src
print >>logfile,''

#
# Now use the stats produced by imstat above
#
# DIRTY IMAGE MAX & RMS (IF YOU MADE A DIRTY IMAGE)
#
#thistest_dirtymax=dirtystats['max'][0]
#oldtest_dirtymax = 0.0515365377069
#diff_dirtymax = abs((oldtest_dirtymax-thistest_dirtymax)/oldtest_dirtymax)
#
#print ' Dirty image max = ',thistest_dirtymax
#print '   Previous: max = ',oldtest_dirtymax
#print '   Difference (fractional) = ',diff_dirtymax
#print ''
#
#print >>logfile,' Dirty Image max = ',thistest_dirtymax
#print >>logfile,'   Previous: max = ',oldtest_dirtymax
#print >>logfile,'   Difference (fractional) = ',diff_dirtymax
#print >>logfile,''
#
#thistest_dirtyrms=dirtystats['rms'][0]
#oldtest_dirtyrms = 0.00243866862729
#diff_dirtyrms = abs((oldtest_dirtyrms-thistest_dirtyrms)/oldtest_dirtyrms)
#
#print ' Dirty image rms = ',thistest_dirtyrms
```

```
#print '   Previous: rms = ',oldtest_dirtyrms
#print '   Difference (fractional) = ',diff_dirtyrms
#print ''
#
#print >>logfile,' Dirty Image rms = ',thistest_dirtyrms
#print >>logfile,'   Previous: rms = ',oldtest_dirtyrms
#print >>logfile,'   Difference (fractional) = ',diff_dirtyrms
#print >>logfile,''

# Now the clean image
#
thistest_immax=cubestats['max'][0]
oldtest_immax = 0.052414759993553162
diff_immax = abs((oldtest_immax-thistest_immax)/oldtest_immax)

print ' Clean image max = ',thistest_immax
print '  Previous: max = ',oldtest_immax
print '  Difference (fractional) = ',diff_immax
print ''

print >>logfile,' Clean Image max = ',thistest_immax
print >>logfile,'  Previous: max = ',oldtest_immax
print >>logfile,'  Difference (fractional) = ',diff_immax
print >>logfile,''

thistest_imrms=cubestats['rms'][0]
oldtest_imrms = 0.0020218724384903908
diff_imrms = abs((oldtest_imrms-thistest_imrms)/oldtest_imrms)

print ' Clean image rms = ',thistest_imrms
print '  Previous: rms = ',oldtest_imrms
print '  Difference (fractional) = ',diff_imrms
print ''

print >>logfile,' Clean image rms = ',thistest_imrms
print >>logfile,'  Previous: rms = ',oldtest_imrms
print >>logfile,'  Difference (fractional) = ',diff_imrms
print >>logfile,''

# Now the moment images
#
thistest_momzeromax=momzerostats['max'][0]
oldtest_momzeromax = 1.40223777294
diff_momzeromax = abs((oldtest_momzeromax-thistest_momzeromax)/oldtest_momz
eromax)

print ' Moment 0 image max = ',thistest_momzeromax
print '  Previous: m0 max = ',oldtest_momzeromax
print '  Difference (fractional) = ',diff_momzeromax
print ''

print >>logfile,' Moment 0 image max = ',thistest_momzeromax
print >>logfile,'  Previous: m0 max = ',oldtest_momzeromax
print >>logfile,'  Difference (fractional) = ',diff_momzeromax
print >>logfile,''

thistest_momoneavg=momonestats['mean'][0]
oldtest_momoneavg = 1479.77119646
diff_momoneavg = abs((oldtest_momoneavg-thistest_momoneavg)/oldtest_momonea
vg)

print ' Moment 1 image mean = ',thistest_momoneavg
print '  Previous: m1 mean = ',oldtest_momoneavg
print '  Difference (fractional) = ',diff_momoneavg
print ''
print '--- Done ---'

print >>logfile,' Moment 1 image mean = ',thistest_momoneavg
print >>logfile,'  Previous: m1 mean = ',oldtest_momoneavg
print >>logfile,'  Difference (fractional) = ',diff_momoneavg
print >>logfile,''
print >>logfile,'--- Done ---'

# Should see output like:
#
# Clean image max should be  0.0524147599936
# Found : Image Max =  0.0523551553488
# Difference (fractional) =  0.00113717290288
#
# Clean image rms should be  0.00202187243849
# Found : Image rms =  0.00202226242982
# Difference (fractional) =  0.00019288621809
#
# Moment 0 image max should be  1.40223777294
# Found : Moment 0 Max =  1.40230333805
# Difference (fractional) =  4.67574844349e-05
#
# Moment 1 image mean should be  1479.77119646
# Found : Moment 1 Mean =  1479.66974528
# Difference (fractional) =  6.85586935973e-05
#
#=====================================================================
# Done
#
logfile.close()
print "Results are in "+outfile
```