CASA_Test

July 23, 2019

1 CASA Calibration Test

Tashalee Billings and the HERA Validation Team

1.1 Setup and Metadata

1.1.1 Abstract

This notebook tests how well CASA is able to recover the bandpass effects that were generated and inserted into the simulated data.

First we start by examining the simulated data before adding the bandpass components. We look at things like the antenna position, uv coverage, and the visibilitiy.

Next we generate the gains and add them to the data to make an Uncalibrated version of the simulated data and write them out into two uvfits file, one with only xx-pol and the other with all 4-pol. These two uvfits files will later be converted to CASA MS.

I've included the lines of code that are to be ran in CASA showing the exact commands I used to calibrate the Measurement Sets.

The third part of the notebook is dedicated to comparing the solutions (*amplitude and phase*) antenna by antenna for both.

1.1.2 Imports

In [394]:
```
from __future__ import (absolute_import, division,
                        print_function, unicode_literals)
#https://python-future.org/quickstart.html

#%matplotlib inline
%matplotlib notebook
import matplotlib.pyplot as plt, numpy as np
import matplotlib.colors as mcolors
import hera_cal, hera_qm, hera_sim
import linsolve
import scipy
import os
import platform

from matplotlib import gridspec
from pyuvdata import UVData
```

```
from datetime import datetime
from IPython.display import Markdown
from hera_cal import io, redcal, abscal, utils, apply_cal
from hera_cal.datacontainer import DataContainer
from copy import deepcopy
from hera_cal.utils import split_pol, split_bl
from importlib import import_module
```

```
In [162]: print("Last executed: %s"%str(datetime.now()))
```

```
Last executed: 2019-07-22 10:58:31.281290
```

- Major Step Description: Test whether or not CASA will recover Bandpass Solution.
- Minor Variation Description: Examine antenna by antenna solutions.
- Pipelines Tested: Common Astronomy Software Application
- Criteria:

    1. Recover input (amplitude and phase) banpass gains:
        1. XX Calibrated Data
        2. Polarization Calibrated Data

1.1.3 Summary

The results of this validation test, in reference to the outlined criteria, are

1. Recover input (amplitude and phase) banpass gains:
    1. XX Calibrated Data
    2. Polarization Calibrated Data

In this notebook I will calibrate to simulated data found on NRAO:
PATH: **/lustre/aoc/projects/hera/Validation/test-2.0.0/randsrc_airybeam_Nsrc100_fullband.uvh5**

1.1.4 Software

```
In [395]: print("Python Version:\n \t",platform.sys.version)
```

```
for module in ["pyuvdata", "hera_stats", "hera_sim", "hera_qm", "hera_pspec", "linsol
    try:
        _mdl = import_module(module)
    except ModuleNotFoundError:
        pass

    if hasattr(_mdl, 'version'):
        gh = getattr(_mdl.version, 'git_hash', None)
    print("Module {:<11}....\tVersion {:<7}.......\tGit {}".format(module, _mdl.__ver
```

```
Python Version:
          3.7.1 | packaged by conda-forge | (default, Feb 25 2019, 21:02:05)
[Clang 4.0.1 (tags/RELEASE_401/final)]
Module pyuvdata   ...        Version 1.4.0  ...        Git d1829efacb60da384f64a8f25a280441bfa
Module hera_stats ...        Version 1.4.0  ...        Git d1829efacb60da384f64a8f25a280441bfa
Module hera_sim   ...        Version 0.0.1  ...        Git b"b'eea7ebae86797c627237c9d676d3a05
Module hera_qm    ...        Version 1.0    ...        Git 400ee8f93321fb27078533083a2cc46ee56
Module hera_pspec ...        Version 1.0    ...        Git 400ee8f93321fb27078533083a2cc46ee56
Module linsolve   ...        Version 0.0.1  ...        Git
Module uvtools    ...        Version 0.1.0  ...        Git 92faf0f37a4e33c217b9331d27b3a3397ff
Module numpy      ...        Version 1.16.4 ...        Git None
Module healvis    ...        Version 1.16.4 ...        Git None
Module healpy     ...        Version 1.12.9 ...        Git None
```

1.1.5 Data

The following paths reflect the exact locations of all data used in this test:
(/lustre/aoc/projects/hera/Validation/test-2.0.0/randsrc_airybeam_Nsrc100_fullband.uvh5)

```
In [163]: path = '/Users/tashaleebillings/Desktop/data/'
          simfile = 'randsrc_airybeam_Nsrc100_fullband.uvh5'
          if not os.path.exists(path+simfile):
              print('File not found.')
```

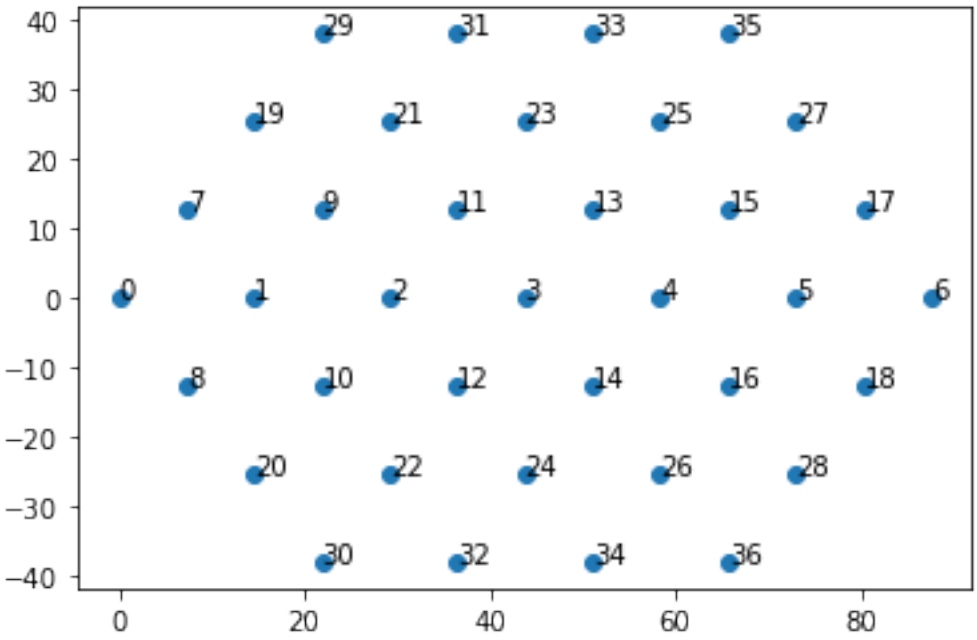
1.2 Examine Simulated Data

Use pyuvdata to take a quick look at simulated data.

```
In [165]: uvd_val = UVData()
          uvd_val.read(path+simfile)
          print("Polarization Array: ",uvd_val.polarization_array)
          print("Polarization Array: ",uvd_val.get_pols())
          print("Number of Antenna: ", len(np.unique(uvd_val.ant_1_array)))

          min_ = (np.unique(uvd_val.time_array)).min()*24.*60.*60. #[Sec]
          max_ = (np.unique(uvd_val.time_array)).max()*24.*60.*60. #[Sec]
          print("Time width of file",end=': ')
          print(max_-min_,"sec")

Polarization Array:  [-5 -6 -7 -8]
Polarization Array:  ['xx', 'yy', 'xy', 'yx']
Number of Antenna:  37
Time width of file: 42.949676513671875 sec
```

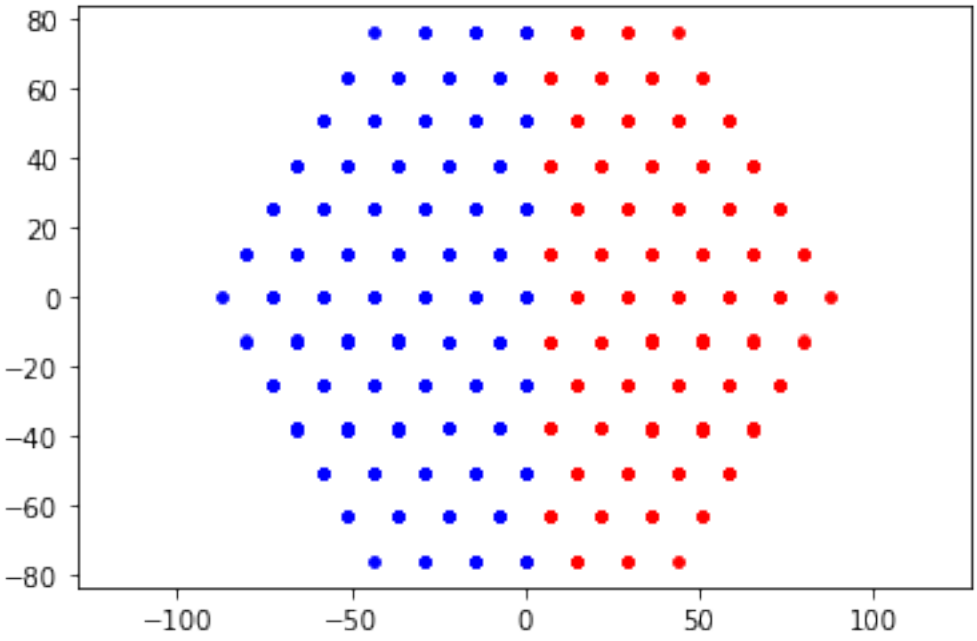
```
In [106]: # get antenna positions and the names associated
          antpos_xyz, antnum = uvd_val.get_ENU_antpos()
```

```
# Plot antennas with names
plt.plot(antpos_xyz[:,0],antpos_xyz[:,1],'o')
for iant,ant in enumerate(antnum):
    plt.text(antpos_xyz[iant,0],antpos_xyz[iant,1],str(ant))
```

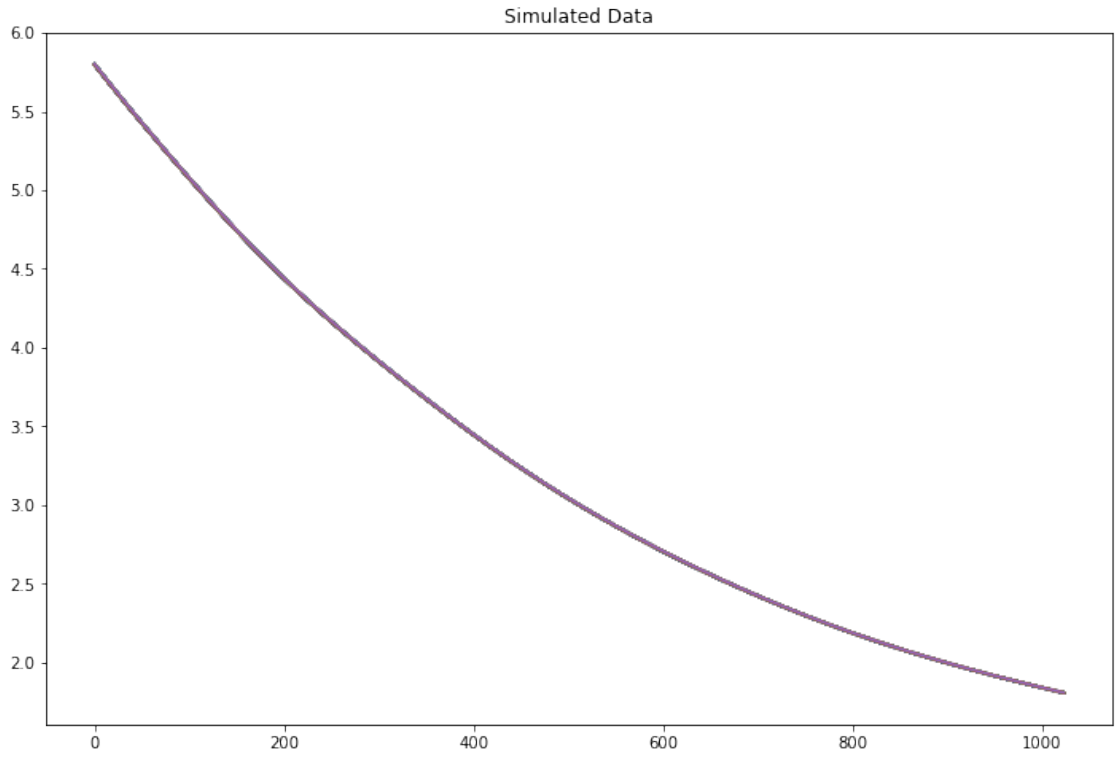


In [107]:
```
# Plot uv coverage
plt.plot(uvd_val.uvw_array[:,0],uvd_val.uvw_array[:,1],'r.')
plt.plot(-uvd_val.uvw_array[:,0],uvd_val.uvw_array[:,1],'b.')
plt.axis('equal')
```

Out[107]:
```
(-96.35165634155274, 96.35165634155274, -83.48934631347656, 83.42118530273437)
```



In [33]:
```
ants_val = np.unique(uvd_val.ant_1_array)

# Check the simulated visibilities of simulated uvh5 data
plt.figure(figsize=(12,8))
plt.title("Simulated Data")
for ant in ants_val:
    uvd_val.select(ant_str='{}x_{}x'.format(ant,ant))
    data_val = (uvd_val.data_array).squeeze()
    plt.plot(data_val.T.real)

    del uvd_val
    uvd_val = UVData()
    uvd_val.read(path+simfile)

#plt.savefig("visibility_{}.png".format(filename))
plt.show()
```



1.2.1 Generate Uncalibrated Data

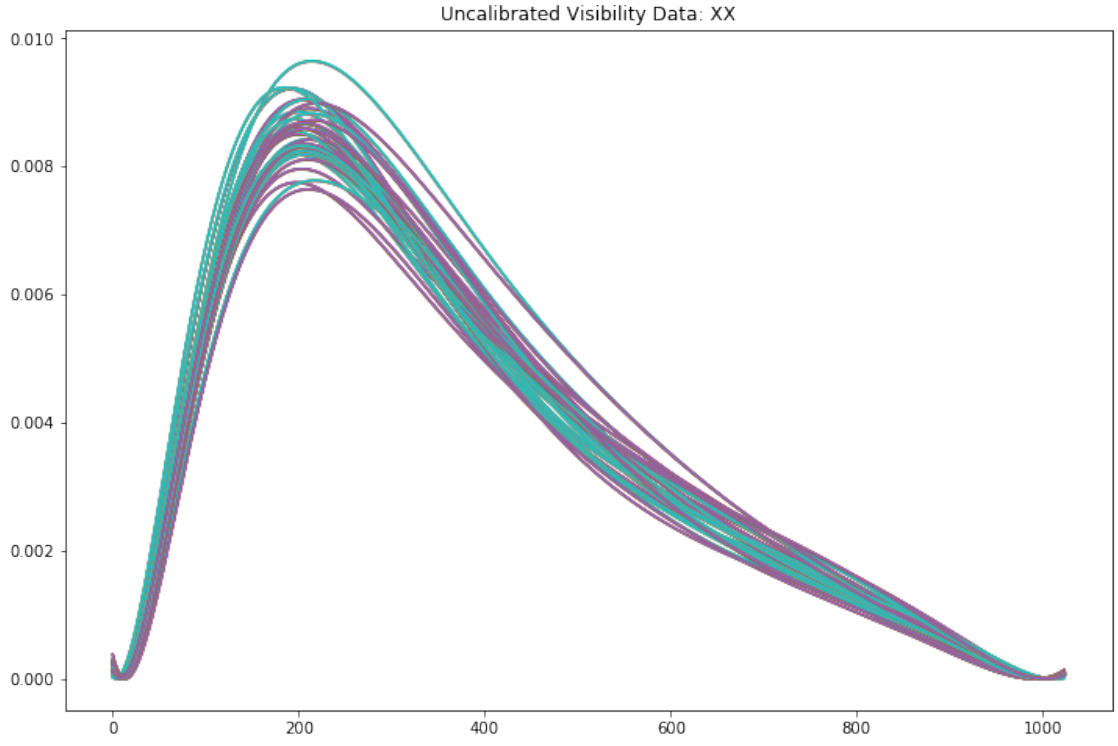
In [20]:
```
hd = io.HERAData(path+simfile)
data, flags, nsamples = hd.read()
```

In [21]:
```
%%time

np.random.seed(21)
ants = sorted({ant: 0 for bl in data.keys() for ant in split_bl(bl)}.keys())

# generate gains with a realistic bandpass and delays between -20 and 20 ns
true_gains = hera_sim.sigchain.gen_gains(hd.freqs/1e9, ants, dly_rng=(-20, 20))

# add random phase offsets to each antenna's gain
phase_offsets = {ant: 2 * np.pi * np.random.rand() for ant in true_gains.keys()}

# uncalibratate data and save
true_gains = {ant: g * np.ones((hd.Ntimes, hd.Nfreqs)) * np.exp(1.0j * phase_offsets[a
              for ant, g in true_gains.items()}
apply_cal.calibrate_in_place(data, true_gains, gain_convention='multiply')
hd.rdate ='' # some python 3 issue
hd.update(data=data)
hd.write_uvfits(path+'uncalibrated_' + simfile.replace("uvh5","uvfits"),
                force_phase=True, spoof_nonessential=True)
```

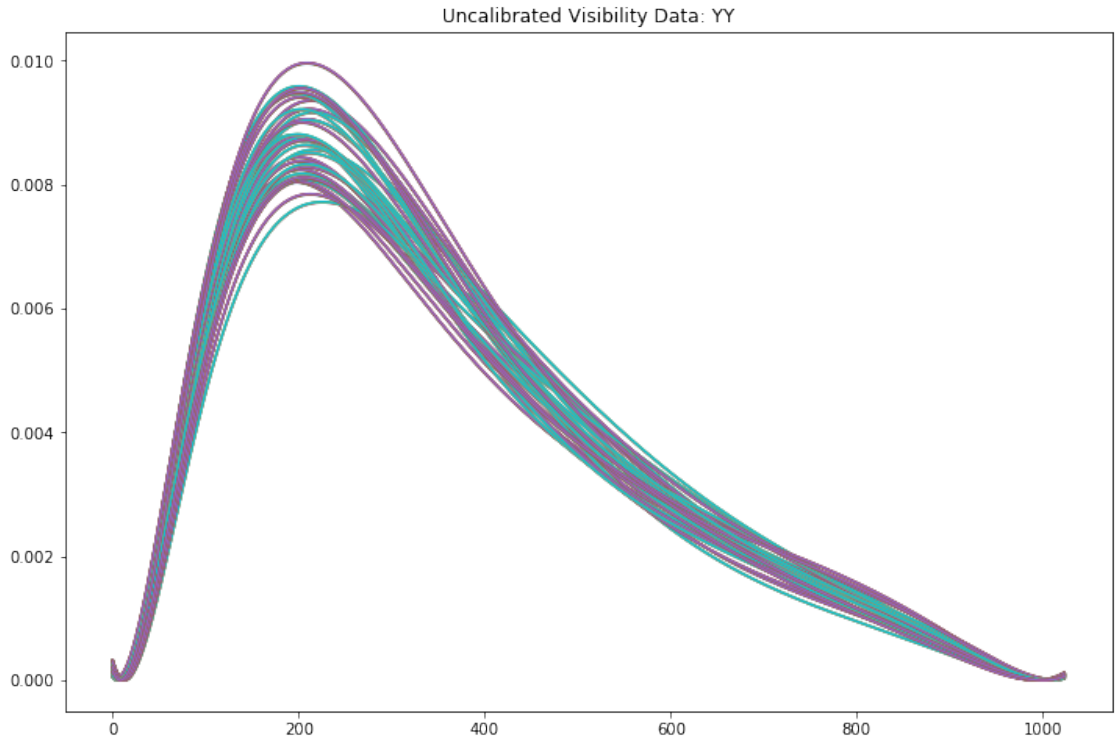
Plot if Uncalibrated Data

The uncalibrated data with all 4 correlation components, [XX,YY,XY,YX], is called: '/Users/tashaleebillings/Desktop/data/uncalibrated_randsrc_airybeam_Nsrc100_fullband.uvfits'

```
In [181]: uncal_hd = UVData()
          uncal_hd.read(path+'uncalibrated_' + simfile.replace("uvh5","uvfits"))

          # Uncalibrated Visibilties for "uncalibrated_randsrc_airybeam_Nsrc100_fullband.uvfit
          ants_val = np.unique(uncal_hd.ant_1_array)

          # Check the simulated visibilities of simulated uvh5 data
          plt.figure(figsize=(12,8))
          plt.title("Uncalibrated Visibility Data: XX")
          for ant in ants_val:
              uncal_hd.select(ant_str='{}x_{}x'.format(ant,ant))
              uncal_data_val = (uncal_hd.data_array).squeeze()
              plt.plot(uncal_data_val.T.real)

              del uncal_hd
              uncal_hd = UVData()
              uncal_hd.read(path+'uncalibrated_' + simfile.replace("uvh5","uvfits"))

          #plt.savefig("visibility_{}.png".format())
          plt.show()
```



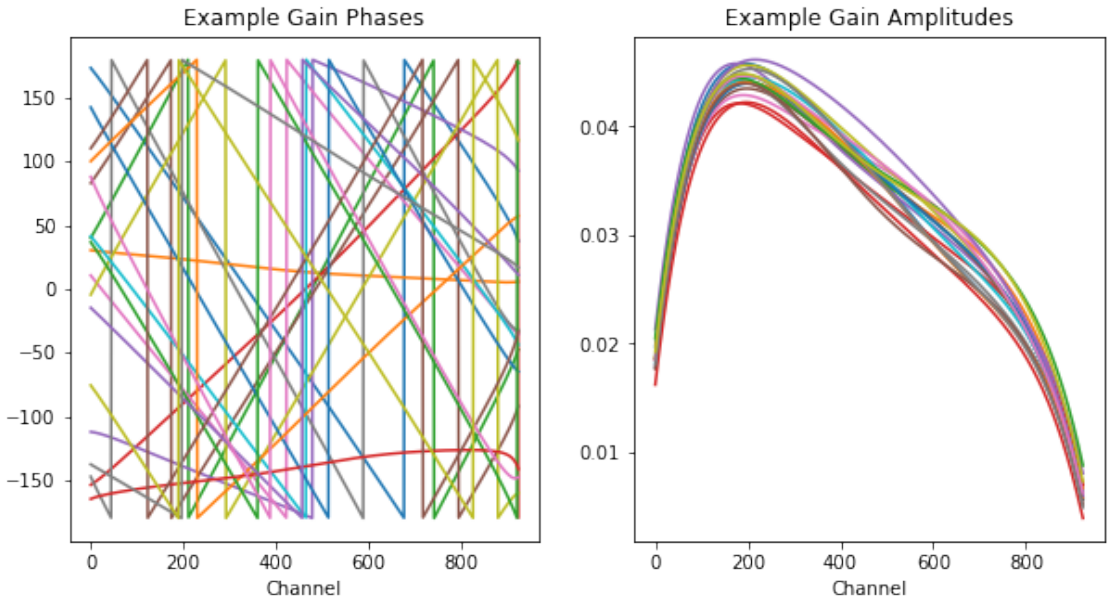
```
In [182]: uncal_hd = UVData()
          uncal_hd.read(path+'uncalibrated_' + simfile.replace("uvh5","uvfits"))

          # Uncalibrated Visibilties for "uncalibrated_randsrc_airybeam_Nsrc100_fullband.uvfit
          ants_val = np.unique(uncal_hd.ant_1_array)

          # Check the simulated visibilities of simulated uvh5 data
          plt.figure(figsize=(12,8))
          plt.title("Uncalibrated Visibility Data: YY")
          for ant in ants_val:
              uncal_hd.select(ant_str='{}y_{}y'.format(ant,ant))
              uncal_data_val = (uncal_hd.data_array).squeeze()
              plt.plot(uncal_data_val.T.real)

              del uncal_hd
              uncal_hd = UVData()
              uncal_hd.read(path+'uncalibrated_' + simfile.replace("uvh5","uvfits"))

          #plt.savefig("visibility_{}.png".format())
          plt.show()
```



Plot of Simulated Bandpass Gains

```
In [143]: fig, ax = plt.subplots(1,2, figsize=(10,5))
          for ant in ants[0::4]:
```

```
        #ax[0].plot(hd.freqs[50:-50]/1e6, np.angle(np.mean(true_gains[ant], axis=0))[50:
        ax[0].plot(np.angle(np.mean(true_gains[ant], axis=0),deg=True)[50:-50])
        ax[0].set_title('Example Gain Phases')
        ax[0].set_xlabel('Channel')
        #ax[1].plot(hd.freqs[50:-50]/1e6, np.abs(np.mean(true_gains[ant], axis=0))[50:-5
        ax[1].plot(np.abs(np.mean(true_gains[ant], axis=0))[50:-50])
        ax[1].set_title('Example Gain Amplitudes')
        ax[1].set_xlabel('Channel')
```

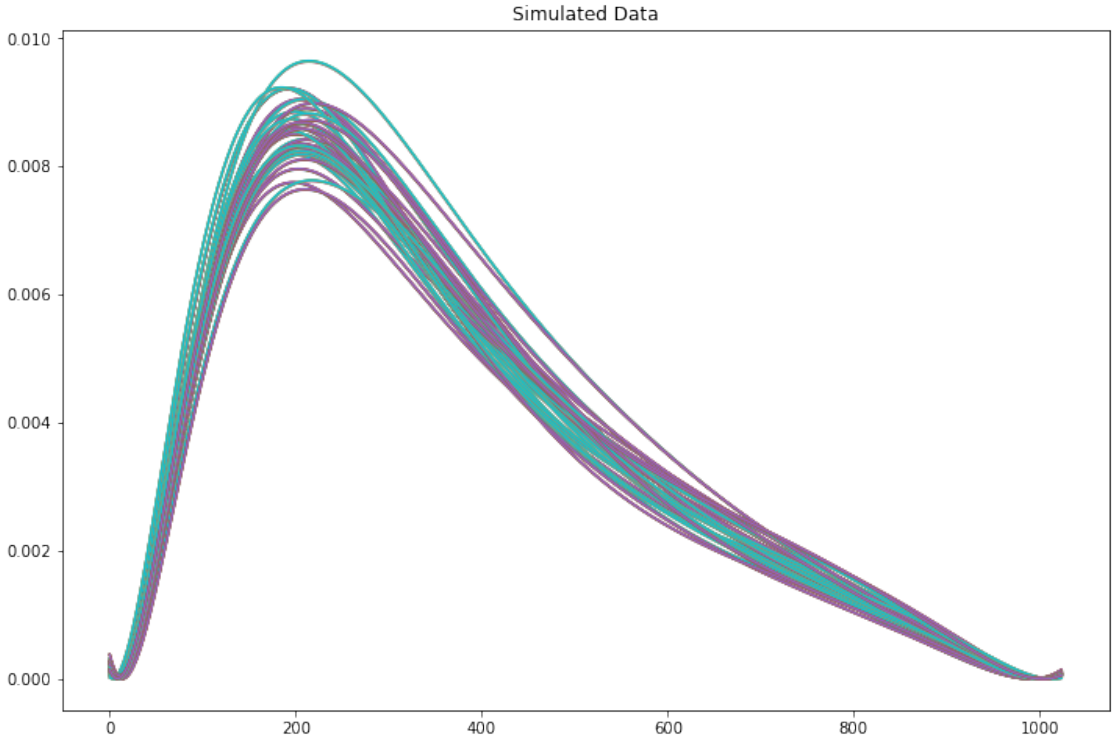



```
In [105]: # Select xx data
          uncal_hd = UVData()
          uncal_hd.read(path+'uncalibrated_' + simfile.replace("uvh5","uvfits"))

          uncal_hd.select(ant_str="(0x,1x,2x,3x,4x,5x,6x,7x,8x,9x,10x,11x,12x,13x,14x,15x,16x,
          uncal_hd.rdate ='' # some python 3 issue
          uncal_hd.write_uvfits(path+'uncalibrated_xx_' + simfile.replace("uvh5","uvfits"),
                                force_phase=True, spoof_nonessential=True)
```

The uncalibrated data with all 4 correlation components, [XX,YY,XY,YX], is called: '/Users/tashaleebillings/Desktop/data/uncalibrated_xx_randsrc_airybeam_Nsrc100_fullband.uvfits'

```
In [119]: # Check the simulated visibilities for "uncalibrated_xx_randsrc_airybeam_Nsrc100_ful
          plt.figure(figsize=(12,8))
          plt.title("Uncalibrated Simulated Data")
          for ant in ants_val:
              uncal_hd.select(ant_str='{}x_{}x'.format(ant,ant))
              uncal_hddata_val = (uncal_hd.data_array).squeeze()
```

```
        plt.plot(uncal_hddata_val.T.real)

        del uncal_hd
        uncal_hd = UVData()
        uncal_hd.read(path+'uncalibrated_xx_' + simfile.replace("uvh5","uvfits"))

    #plt.savefig("visibility_{}.png".format(filename))
    plt.show()
```



1.2.2 Calibrate XX Simulated Data

To Be Ran In CASA

```
In [ ]: import shutil,os,glob
        import numpy as np

        path = '/Users/tashaleebillings/Desktop/data/'
        uvfits = [path+"uncalibrated_xx_randsrc_airybeam_Nsrc100_fullband.uvfits"]
        #uvfits = [path+"uncalibrated_randsrc_airybeam_Nsrc100_fullband.uvfits"]
        for uvfit in uvfits:
            msfile=uvfit.strip('uvfits') + 'MS'
            importuvfits(vis=msfile,fitsfile=uvfit)
```

```
        # I'm not flagging any data

        # Create Model Data Column
        def makeinitmodel(visib,image_model):
            ft(vis=visib , model =image_model , usescratch = True)

In [ ]: # Insert Model
        makeinitmodel(visib=msfile,image_model=path+"randsrc_airybeam_Nsrc100_fullband_xxModel
        #
        ## Begin Calibration ##
        refant = '0'
        gaintables = ''#[]
        msin=msfile
        image_path = '/Users/tashaleebillings/Desktop/Research/images/'

        # Calibration File Name
        bc = os.path.basename(msin)  + ".B.cal"

        # Bandpass
        bandpass(vis=msin, bandtype="B", combine='scan',caltable=bc,
                 gaintable=gaintables, solint='inf', refant=refant)
        plotcal(bc, xaxis='chan', yaxis='amp', figfile=image_path+"{}.amp.png".format(bc), shou
        plotcal(bc, xaxis='chan', yaxis='phase', figfile=image_path+"{}.phs.png".format(bc), sh

        # Apply Calibration Solution
        applycal(msin, gaintable=[bc])

        # CREATE .NPZ FILE IN CASA

        tb.open(bc)
        gains=tb.getcol('CPARAM')
        np.savez(bc+'.npz',gains=gains)

        # Make CASA mfs Image
        imsize_=512
        spw_='0:100~924'
        stokes_='XX'
        imagename = os.path.splitext(msin)

        tclean(vis=msin,imagename='calibrated_xx_randsrc_airybeam_Nsrc100_fullband',
               spw=spw_, niter=1, cycleniter=-1, weighting='briggs', stokes='IQUV',
               robust=0, imsize=[imsize_,imsize_], cell='500.0arcsec', specmode='mfs',
               deconvolver="clark", threshold='0.1mJy',interactive=True, pblimit=-1)
        #clean(vis=msin,imagename='calibrated_xx_randsrc_airybeam_Nsrc100_fullband',
        #      niter =0,weighting = 'briggs',robust =0,imsize =[imsize_ ,imsize_],
        #      cell=['500 arcsec'] ,mode='mfs',nterms =1,spw=spw_,stokes=stokes_,
        #      interactive=True,npercycle=5,threshold='0.1mJy/beam')
```

```
viewer("calibrated_xx_randsrc_airybeam_Nsrc100_fullband.image",
       outfile=image_path+"calibrated_xx_randsrc_airybeam_Nsrc100_fullband.image.png")

# Make CASA Spectrum Image
imagename = '1024_chan_calibrated_xx_randsrc_airybeam_Nsrc100_fullband.image'
fitsname = imagename+'.fits'

tclean(vis=msin,imagename[:-6], spw='0',niter=1, cycleniter=-1, weighting='briggs',
       robust=0, imsize=[512,512], cell='500.0arcsec', deconvolver="clark", stokes='IQ
       interactive=False, threshold='0.1mJy', pblimit=-1,
       specmode='cube', start=0, width=32, nchan=32)
#clean(vis=msin,imagename[:-6],niter=0,weighting = 'briggs',robust =0,imsize =[512 ,51
#      ,cell=['500 arcsec'],nterms =1,spw='0', stokes=stokes_
#      ,mode='channel',nchan=1024, start=0, width=1)

exportfits(imagename,fitsname)
```

```
In [ ]: tclean(vis=msin,imagename='calibrated_xx_randsrc_airybeam_Nsrc100_fullband',
              spw=spw_, niter=1, cycleniter=-1, weighting='briggs', stokes='IQUV',
              robust=0, imsize=[imsize_,imsize_], cell='500.0arcsec', specmode='mfs',
              deconvolver="clark", threshold='0.1mJy',interactive=True, pblimit=-1)

        tclean(vis=msin,imagename[:-6], spw='0',niter=1, cycleniter=-1, weighting='briggs',
              robust=0, imsize=[512,512], cell='500.0arcsec', deconvolver="clark", stokes='IQ
              interactive=False, threshold='0.1mJy', pblimit=-1,
              specmode='cube', start=0, width=32, nchan=32)
```

1.2.3 Compare Bandpass Solutions

Compare the Bandpass Solutions CASA derived to the ones I generated.

```
In [198]: ants_val
```

```
Out[198]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
                 34, 35, 36], dtype=int32)
```

```
In [201]: list(map(lambda a:str(a),ants_val))
```

```
Out[201]: ['0',
           '1',
           '2',
           '3',
           '4',
           '5',
           '6',
           '7',
           '8',
```

```
            '9',
            '10',
            '11',
            '12',
            '13',
            '14',
            '15',
            '16',
            '17',
            '18',
            '19',
            '20',
            '21',
            '22',
            '23',
            '24',
            '25',
            '26',
            '27',
            '28',
            '29',
            '30',
            '31',
            '32',
            '33',
            '34',
            '35',
            '36']
```

```
In [238]: list(np.load(npzlist[0]).keys())
```

```
Out[238]: ['gains']
```

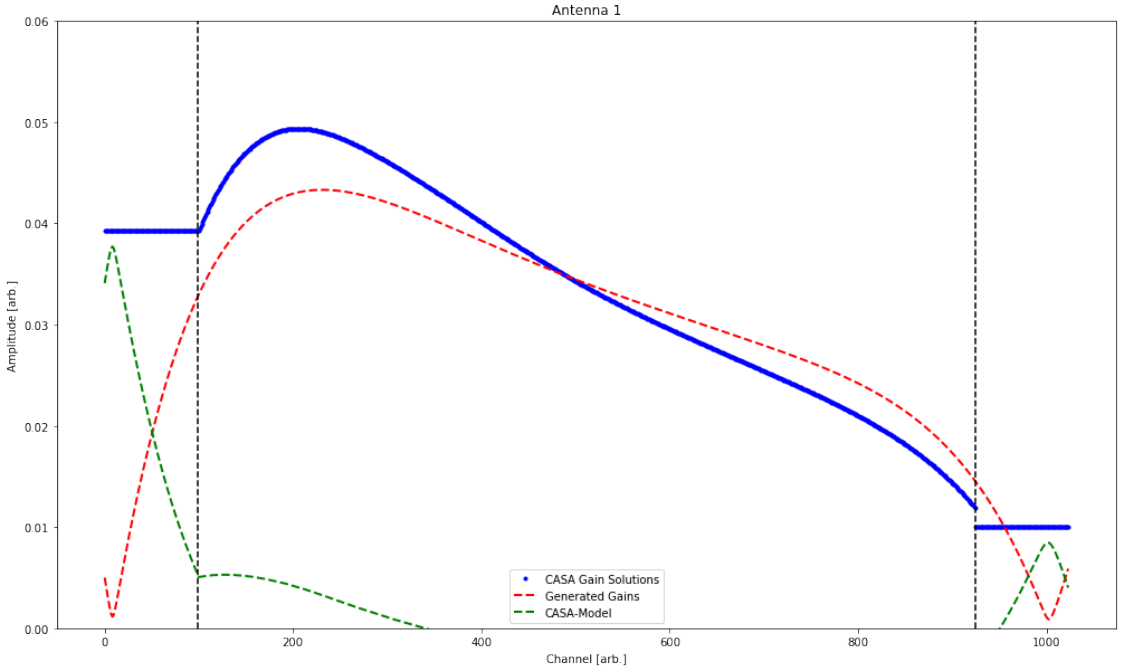
```
In [ ]: bc = "/Users/tashaleebillings/Desktop/data/new_uncalibrated_xx_randsrc_airybeam_Nsrc10
        npzlist = [bc+'.npz']
        d = np.load(npzlist[0])

        # Isolate xx and yy components
        xxants=ants[0::2] # ants comes from the section called "Generate Uncalibrated Data"
        yyants=ants[1::2]

        good_ants = list(map(lambda a:str(a),ants_val))
        freqs = np.linspace(100,200,num=1024)
```

Amplitude Solutions Simply change "depenvar"/"depenvarmodel" to xx or yy components.

In the CASA solutions (blue) The highband and lowband are flat because CASA flagged those solutions but for some reason they don't have the value of 1.

In [360]:
```
%matplotlib inline
plt.figure(figsize=(17,10))
plt.title("Antenna 1")
plt.plot(np.abs(d['gains'][0,:,1]),'b.',label='CASA Gain Solutions',lw=2)
plt.plot(np.abs(np.mean(true_gains[xxants[1]], axis=0)),'r--',label='Generated Gains
plt.plot(np.abs(d['gains'][0,:,1])-np.abs(np.mean(true_gains[xxants[0]], axis=0)),
         'g--',label='CASA-Model',lw=2)
plt.axvline(99, color='k', linestyle='--')
plt.axvline(925, color='k', linestyle='--')
plt.ylim(0.0,0.06)
plt.xlabel('Channel [arb.]')
plt.ylabel('Amplitude [arb.]')
plt.legend()
```

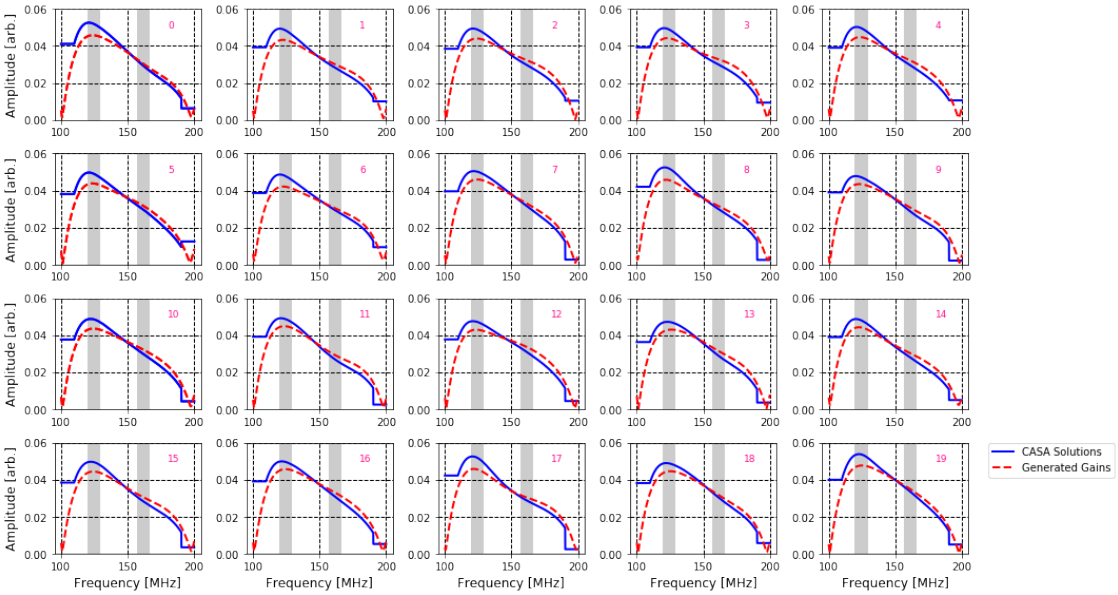
Out[360]: <matplotlib.legend.Legend at 0x137f0ba20>



In [328]:
```
nrow,ncol = 4,5
numplots = nrow*ncol
ndays = len(npzlist)
iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

#msk_spec = np.zeros((1024))

f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
#f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
```

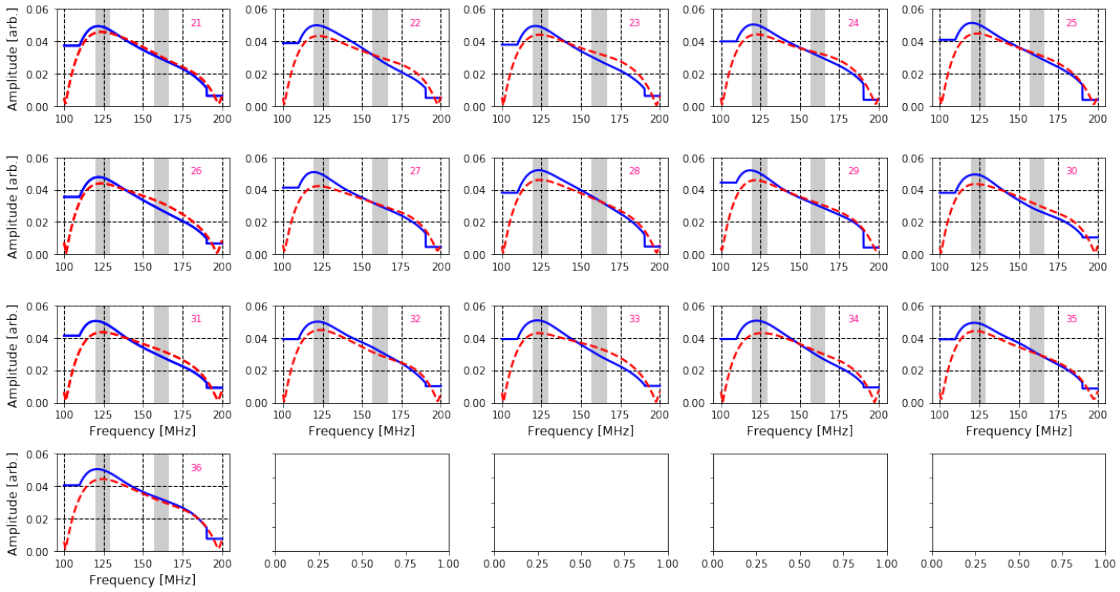
```
gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

for i,a in enumerate(good_ants[:20]):

    #for ind in range(numplots):
    ax=plt.subplot(gs[iarr[i],jarr[i]])
    ax.set_ylim(0,0.06)
    #ax.set_xlim(100,200)
    ax.annotate(str(a), xy=(180,0.05), size=9, color='deeppink')
    ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
    ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

    for daynum in range(ndays):
        data = np.load(npzlist[daynum])
        xgain = np.abs(data['gains'][0,:,int(a)])
        ygain = np.abs(data['gains'][1,:,int(a)])


#print(list(msk_xgain))
        depenvar =xgain
        depenvarmodel = xxants
        if i > 14:
            ax.set_xlabel('Frequency [MHz]',size=12)
            ax.grid(color='k', linestyle='--', linewidth=1)
        if i == 0:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
        if i == 5:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
        #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
        if i == 10:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--

        if i == 15:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--

        else:
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
            ax.grid(color='k', linestyle='--', linewidth=1)
```

```
        ax.legend(bbox_to_anchor=(2, 1), loc='upper right', borderaxespad=0.)
        plt.tight_layout()
        #f.savefig('2458115.24482.HH.uvOCR_BP.png')
        plt.show()
```



In [326]:
```
nrow,ncol = 4,5
numplots = nrow*ncol
ndays = len(npzlist)
iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
#f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

for i,a in enumerate(good_ants[21:]):

    #for ind in range(numplots):
    ax=plt.subplot(gs[iarr[i],jarr[i]])
    ax.set_ylim(0,0.06)
    #ax.set_xlim(100,200)
    ax.annotate(str(a), xy=(180,0.05), size=9, color='deeppink')
    ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
    ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

    for daynum in range(ndays):
        data = np.load(npzlist[daynum])
        xgain = np.abs(data['gains'][0,:,int(a)])
```

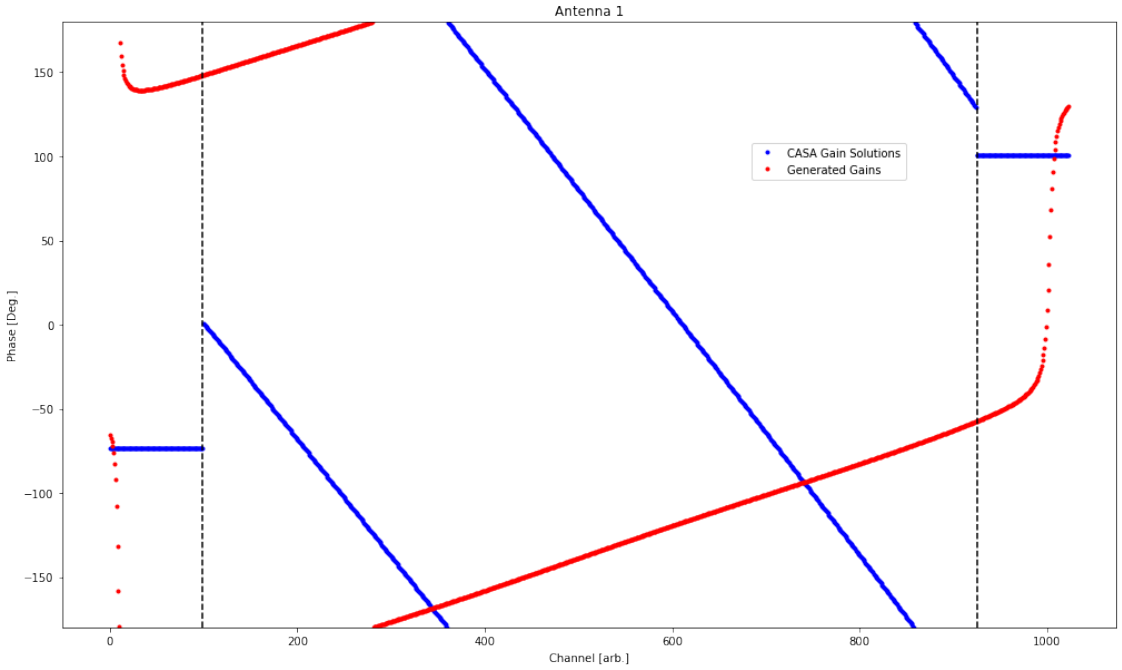
```
        ygain = np.abs(data['gains'][1,:,int(a)])

#print(list(msk_xgain))
        depenvar =xgain
        depenvarmodel = xxants
        if i > 9:
            ax.set_xlabel('Frequency [MHz]',size=12)
            ax.grid(color='k', linestyle='--', linewidth=1)
        if i == 0:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
        if i == 5:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
        #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
        if i == 10:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--

        if i == 15:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--

        else:
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
            ax.grid(color='k', linestyle='--', linewidth=1)

#ax.legend(bbox_to_anchor=(3.3, 0.8), loc='upper right', borderaxespad=0.)
plt.tight_layout()
#f.savefig('2458115.24482.HH.uvOCR_BP.png')
plt.show()
```



Phase Solutions Simply change "depenvar"/"dependvarmodel" to xx or yy components.

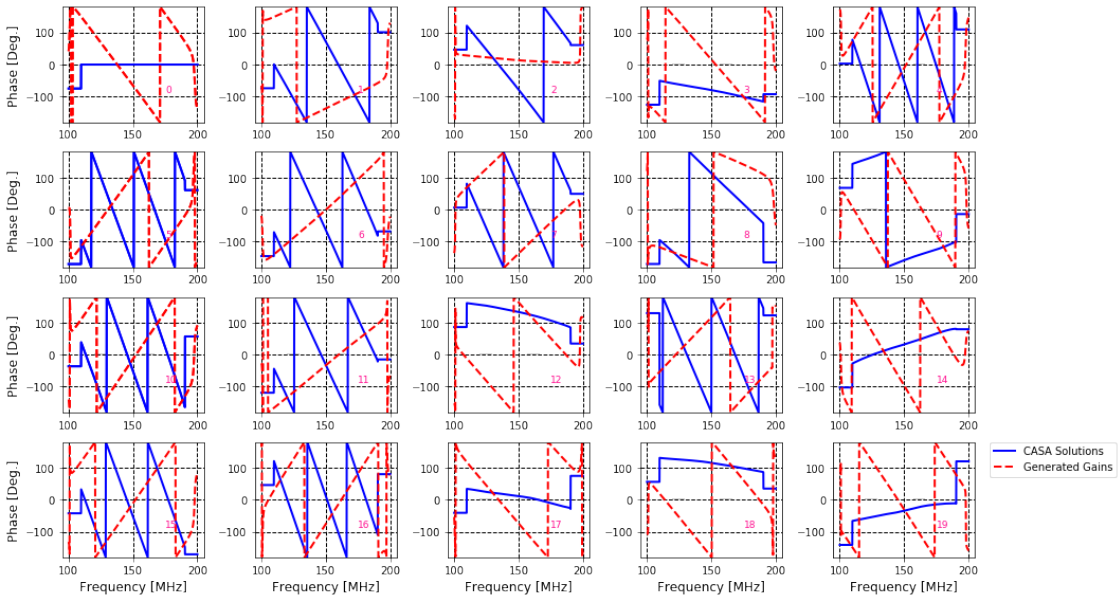
In the CASA solutions (blue) The highband and lowband are flat because CASA flagged those solutions but for some reason they don't have the value of 0.

```
In [371]: %matplotlib inline
          plt.figure(figsize=(17,10))
          plt.title("Antenna 1")
          plt.plot(np.angle(d['gains'][0,:,1],deg=True),'b.',label='CASA Gain Solutions',lw=2)
          plt.plot(np.angle(np.mean(true_gains[xxants[1]], axis=0),deg=True),'r.',label='Genera
          plt.axvline(99, color='k', linestyle='--')
          plt.axvline(925, color='k', linestyle='--')
          plt.ylim(-180.,180.)
          plt.xlabel('Channel [arb.]')
          plt.ylabel('Phase [Deg.]')
          plt.legend(bbox_to_anchor=(0.8, 0.8), loc='upper right', borderaxespad=0.)

Out[371]: <matplotlib.legend.Legend at 0x1361e6eb8>
```



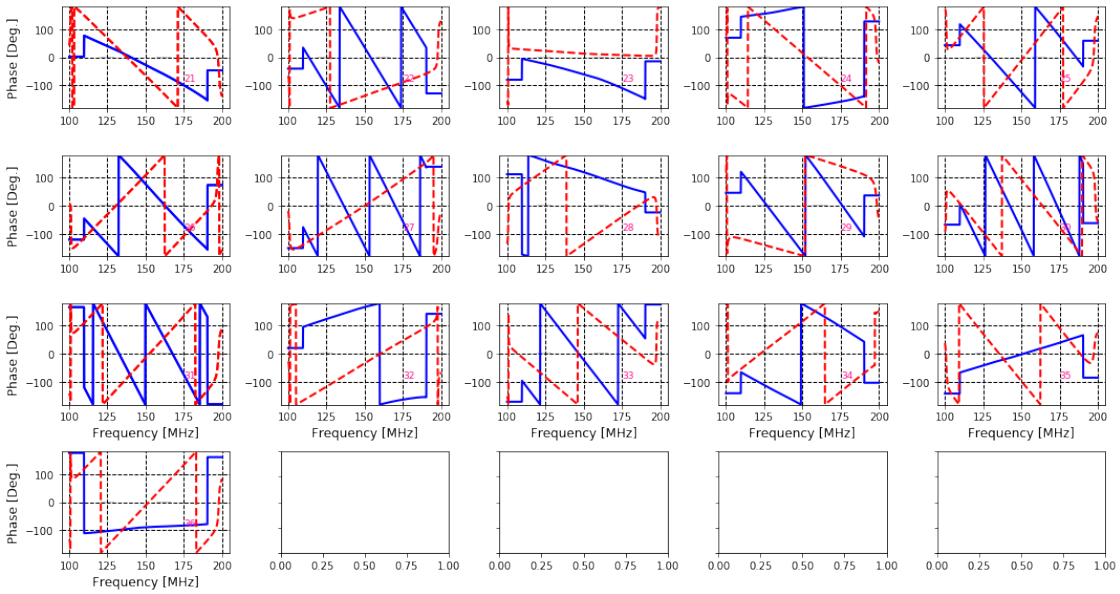
In [380]:
```
nrow,ncol = 4,5
numplots = nrow*ncol
ndays = len(npzlist)
iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

#msk_spec = np.zeros((1024))

f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
#f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

for i,a in enumerate(good_ants[:20]):

    #for ind in range(numplots):
    ax=plt.subplot(gs[iarr[i],jarr[i]])
    ax.set_ylim(-180.,180.)
    #ax.set_xlim(100,200)
    ax.annotate(str(a), xy=(175,-85), size=9, color='deeppink')
    ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
    ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

    for daynum in range(ndays):
        data = np.load(npzlist[daynum])
        xgain = np.angle(data['gains'][0,:,int(a)],deg=True)
        ygain = np.angle(data['gains'][1,:,int(a)],deg=True)
```

```
#print(list(msk_xgain))
        depenvar =xgain
        depenvarmodel = xxants
        if i > 14:
            ax.set_xlabel('Frequency [MHz]',size=12)
            ax.grid(color='k', linestyle='--', linewidth=1)
        if i == 0:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=
        if i == 5:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=
        #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
        if i == 10:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=

        if i == 15:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=

        else:
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=
            ax.grid(color='k', linestyle='--', linewidth=1)

ax.legend(bbox_to_anchor=(2, 1), loc='upper right', borderaxespad=0.)
plt.tight_layout()
#f.savefig('2458115.24482.HH.uvOCR_BP.png')
plt.show()
```




```
In [381]: nrow,ncol = 4,5
          numplots = nrow*ncol
          ndays = len(npzlist)
          iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
          jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

          #msk_spec = np.zeros((1024))

          f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
          #f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
          gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

          for i,a in enumerate(good_ants[21:]):

              #for ind in range(numplots):
              ax=plt.subplot(gs[iarr[i],jarr[i]])
              ax.set_ylim(-180.,180.)
              #ax.set_xlim(100,200)
              ax.annotate(str(a), xy=(175,-85), size=9, color='deeppink')
              ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
              ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

              for daynum in range(ndays):
                  data = np.load(npzlist[daynum])
                  xgain = np.angle(data['gains'][0,:,int(a)],deg=True)
                  ygain = np.angle(data['gains'][1,:,int(a)],deg=True)
```

```
#print(list(msk_xgain))
        depenvar =xgain
        depenvarmodel = xxants
        if i > 9:
            ax.set_xlabel('Frequency [MHz]',size=12)
            ax.grid(color='k', linestyle='--', linewidth=1)
        if i == 0:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg
        if i == 5:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg
        #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
        if i == 10:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg

        if i == 15:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg

        else:
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg
            ax.grid(color='k', linestyle='--', linewidth=1)

#ax.legend(bbox_to_anchor=(2, 1), loc='upper right', borderaxespad=0.)
plt.tight_layout()
#f.savefig('2458115.24482.HH.uvOCR_BP.png')
plt.show()
```



1.2.4 Polarization Calibration

To Be Ran In CASA

```
In [ ]: import shutil,os,glob
        import numpy as np

        path = '/Users/tashaleebillings/Desktop/data/'
        uvfits = [path+"uncalibrated_randsrc_airybeam_Nsrc100_fullband.uvfits"]
        for uvfit in uvfits:
            msfile=uvfit.strip('uvfits') + 'MS'
            importuvfits(vis=msfile,fitsfile=uvfit)

        # I'm not flagging any data

        # Create Model Data Column
        def makeinitmodel(visib,image_model):
            ft(vis=visib , model =image_model , usescratch = True)

In [ ]: # Insert Model
        makeinitmodel(visib=msfile,image_model=path+"randsrc_airybeam_Nsrc100_fullband_fullStok
        #
        ## Begin Calibration ##
        refant = '0'
        gaintables = ''#[]
        msin=msfile
        image_path = '/Users/tashaleebillings/Desktop/Research/images/'
```

```
# Calibration File Name
bc = os.path.basename(msin)  + ".B.cal"

# Bandpass
bandpass(vis=msin, bandtype="B", combine='scan',caltable=bc,
         gaintable=gaintables, solint='inf', refant=refant)
plotcal(bc, xaxis='chan', yaxis='amp', figfile=image_path+"{}.amp.png".format(bc), shou
plotcal(bc, xaxis='chan', yaxis='phase', figfile=image_path+"{}.phs.png".format(bc), sh

# Apply Calibration Solution
applycal(msin, gaintable=[bc])

# CREATE .NPZ FILE IN CASA
tb.open(bc)
gains=tb.getcol('CPARAM')
np.savez(bc+'.npz',gains=gains)

# Make CASA mfs Image
imsize_=512
spw_='0:100~924'
stokes_='IQUV'
imagename = os.path.splitext(msin)

tclean(vis=msin,imagename='calibrated_randsrc_airybeam_Nsrc100_fullband',
       spw=spw_, niter=1, cycleniter=-1, weighting='briggs', stokes='IQUV',
       robust=0, imsize=[imsize_,imsize_], cell='500.0arcsec', specmode='mfs',
       deconvolver="clark", threshold='0.1mJy',interactive=True, pblimit=-1)
#clean(vis=msin,imagename='calibrated_randsrc_airybeam_Nsrc100_fullband',
#      niter =0,weighting = 'briggs',robust =0,imsize =[imsize_ ,imsize_],
#      cell=['500 arcsec'] ,mode='mfs',nterms =1,spw=spw_,stokes=stokes_,
#      interactive=True,npercycle=5,threshold='0.1mJy/beam')

viewer("calibrated_randsrc_airybeam_Nsrc100_fullband.image",
       outfile=image_path+"calibrated_randsrc_airybeam_Nsrc100_fullband.image.png")

# Make CASA Spectrum Image
imagename = '1024_chan_calibrated_randsrc_airybeam_Nsrc100_fullband.image'
fitsname = imagename+'.fits'

tclean(vis=msin,imagename[:-6], spw='0',niter=1, cycleniter=-1, weighting='briggs',
       robust=0, imsize=[512,512], cell='500.0arcsec', deconvolver="clark", stokes='IQ
       interactive=False, threshold='0.1mJy', pblimit=-1,
       specmode='cube', start=0, width=32, nchan=32)
#clean(vis=msin,imagename[:-6],niter=0,weighting = 'briggs',robust =0,imsize =[512 ,51
#      ,cell=['500 arcsec'],nterms =1,spw='0', stokes=stokes_
#      ,mode='channel',nchan=1024, start=0, width=1)
```

```
exportfits(imagename,fitsname)
```

1.2.5 Compare Bandpass Solutions

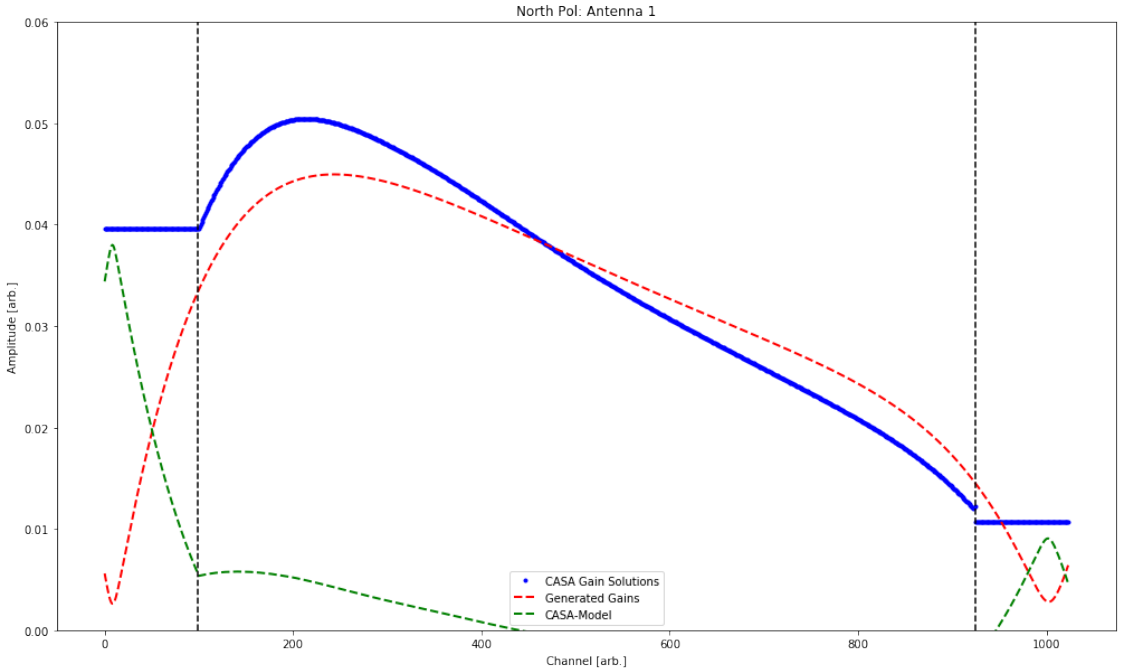
Compare the Bandpass Solutions CASA derived to the ones I generated. Since the solutions above are for the East-pol, these solutions below are North-pol.

```
In [382]: bc = "/Users/tashaleebillings/Desktop/data/ncalibrated_randsrc_airybeam_Nsrc100_full
          npzlist = [bc+'.npz']
          d = np.load(npzlist[0])

          # Isolate xx and yy components
          xxants=ants[0::2] # ants comes from the section called "Generate Uncalibrated Data"
          yyants=ants[1::2]

          good_ants = list(map(lambda a:str(a),ants_val))
          freqs = np.linspace(100,200,num=1024)
```

Amplitude Solutions Simply change "depenvar"/"dependvarmodel" to xx or yy components.

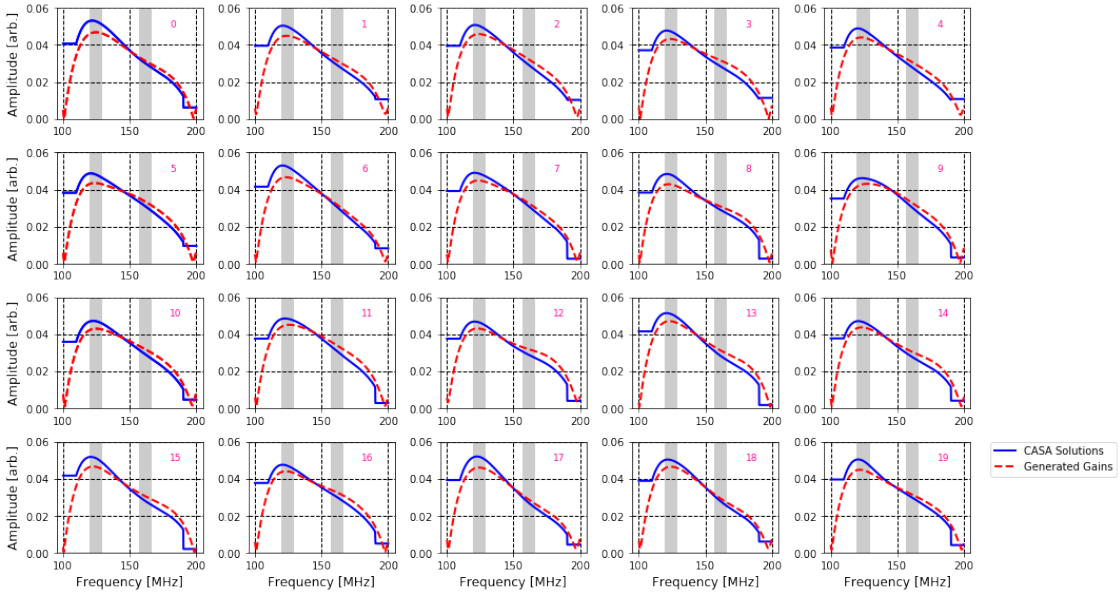
In the CASA solutions (blue) The highband and lowband are flat because CASA flagged those solutions but for some reason they don't have the value of 1.

```
In [386]: %matplotlib inline
          plt.figure(figsize=(17,10))
          plt.title("North Pol: Antenna 1")
          plt.plot(np.abs(d['gains'][1,:,1]),'b.',label='CASA Gain Solutions',lw=2)
          plt.plot(np.abs(np.mean(true_gains[yyants[1]], axis=0)),'r--',label='Generated Gains
          plt.plot(np.abs(d['gains'][1,:,1])-np.abs(np.mean(true_gains[xxants[0]], axis=0)),
                   'g--',label='CASA-Model',lw=2)
          plt.axvline(99, color='k', linestyle='--')
          plt.axvline(925, color='k', linestyle='--')
          plt.ylim(0.0,0.06)
          plt.xlabel('Channel [arb.]')
          plt.ylabel('Amplitude [arb.]')
          plt.legend()

Out[386]: <matplotlib.legend.Legend at 0x1399f1ba8>
```



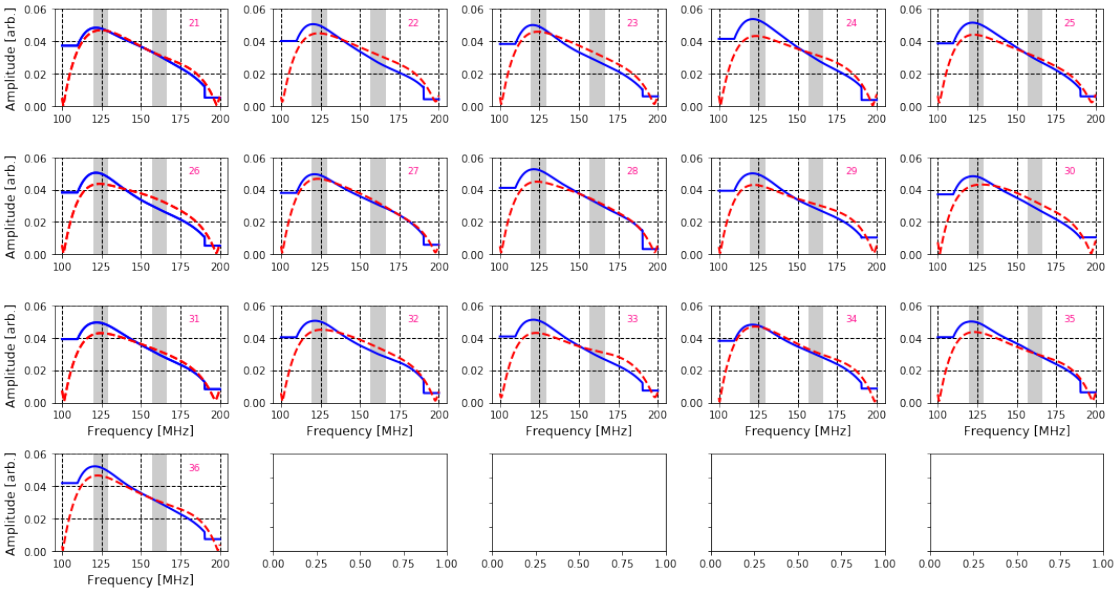
In [389]:
```
nrow,ncol = 4,5
numplots = nrow*ncol
ndays = len(npzlist)
iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

#msk_spec = np.zeros((1024))

f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
#f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

for i,a in enumerate(good_ants[:20]):

    #for ind in range(numplots):
    ax=plt.subplot(gs[iarr[i],jarr[i]])
    ax.set_ylim(0,0.06)
    #ax.set_xlim(100,200)
    ax.annotate(str(a), xy=(180,0.05), size=9, color='deeppink')
    ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
    ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

    for daynum in range(ndays):
        data = np.load(npzlist[daynum])
        xgain = np.abs(data['gains'][0,:,int(a)])
```

```
        ygain = np.abs(data['gains'][1,:,int(a)])


#print(list(msk_xgain))
        depenvar =ygain
        depenvarmodel = yyants
        if i > 14:
            ax.set_xlabel('Frequency [MHz]',size=12)
            ax.grid(color='k', linestyle='--', linewidth=1)
        if i == 0:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
        if i == 5:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
        #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
        if i == 10:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--

        if i == 15:
            ax.set_ylabel('Amplitude [arb.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--

        else:
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.abs(np.mean(true_gains[depenvarmodel[i]], axis=0)),'r--
            ax.grid(color='k', linestyle='--', linewidth=1)

ax.legend(bbox_to_anchor=(2, 1), loc='upper right', borderaxespad=0.)
plt.tight_layout()
#f.savefig('2458115.24482.HH.uvOCR_BP.png')
plt.show()
```




```
In [390]: nrow,ncol = 4,5
          numplots = nrow*ncol
          ndays = len(npzlist)
          iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
          jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

          f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
          #f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
          gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

          for i,a in enumerate(good_ants[21:]):

              #for ind in range(numplots):
              ax=plt.subplot(gs[iarr[i],jarr[i]])
              ax.set_ylim(0,0.06)
              #ax.set_xlim(100,200)
              ax.annotate(str(a), xy=(180,0.05), size=9, color='deeppink')
              ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
              ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

              for daynum in range(ndays):
                  data = np.load(npzlist[daynum])
                  xgain = np.abs(data['gains'][0,:,int(a)])
                  ygain = np.abs(data['gains'][1,:,int(a)])

          #print(list(msk_xgain))
                  depenvar =ygain
                  depenvarmodel = yyants
```

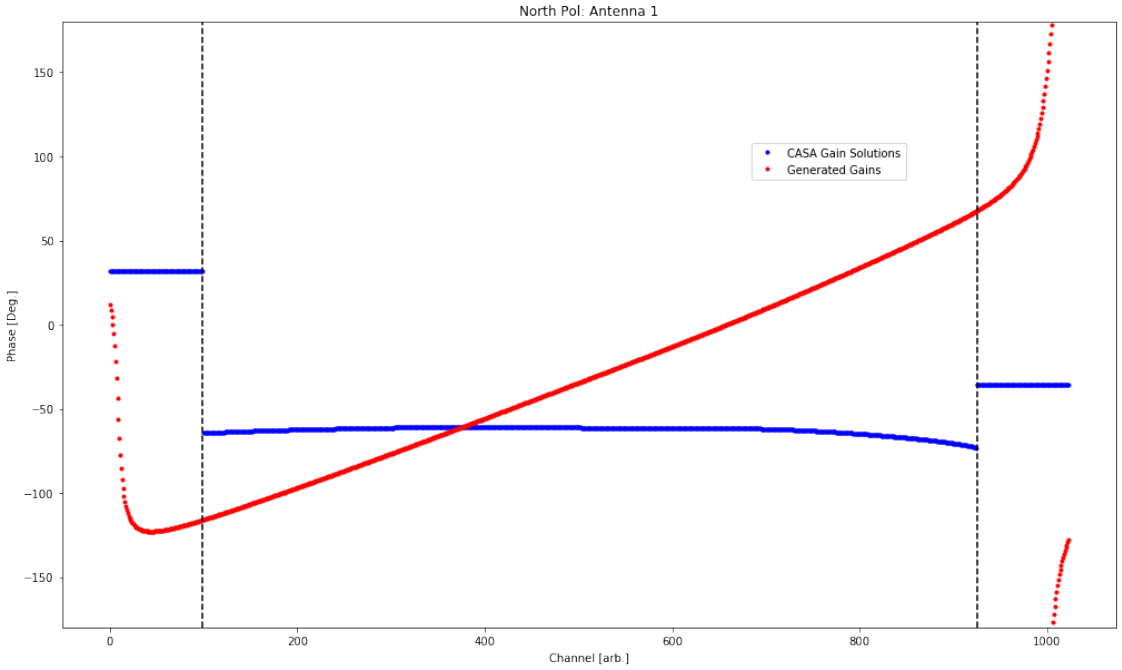
```
            if i > 9:
                ax.set_xlabel('Frequency [MHz]',size=12)
                ax.grid(color='k', linestyle='--', linewidth=1)
            if i == 0:
                ax.set_ylabel('Amplitude [arb.]',size=12)
                ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
                ax.plot(freqs,np.abs(np.mean(true_gains[dependentvarmodel[i]], axis=0)),'r--
            if i == 5:
                ax.set_ylabel('Amplitude [arb.]',size=12)
                ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
                ax.plot(freqs,np.abs(np.mean(true_gains[dependentvarmodel[i]], axis=0)),'r--
            #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
            if i == 10:
                ax.set_ylabel('Amplitude [arb.]',size=12)
                ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
                ax.plot(freqs,np.abs(np.mean(true_gains[dependentvarmodel[i]], axis=0)),'r--

            if i == 15:
                ax.set_ylabel('Amplitude [arb.]',size=12)
                ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
                ax.plot(freqs,np.abs(np.mean(true_gains[dependentvarmodel[i]], axis=0)),'r--

            else:
                ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
                ax.plot(freqs,np.abs(np.mean(true_gains[dependentvarmodel[i]], axis=0)),'r--
                ax.grid(color='k', linestyle='--', linewidth=1)

    #ax.legend(bbox_to_anchor=(3.3, 0.8), loc='upper right', borderaxespad=0.)
    plt.tight_layout()
    #f.savefig('2458115.24482.HH.uvOCR_BP.png')
    plt.show()
```



Phase Solutions Simply change "depenvar"/"dependvarmodel" to xx or yy components.

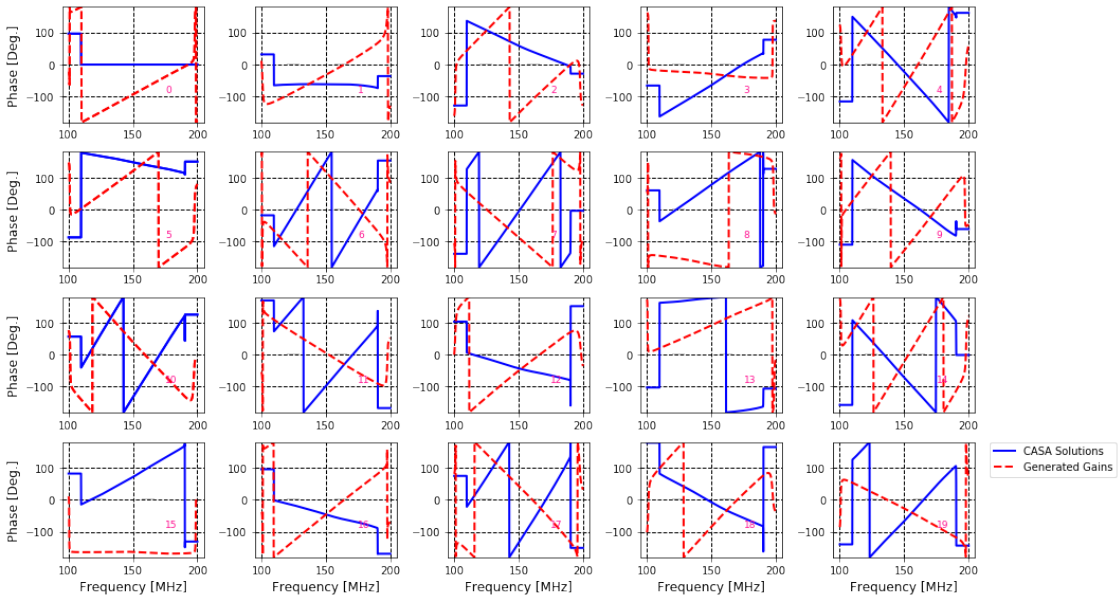
In the CASA solutions (blue) The highband and lowband are flat because CASA flagged those solutions but for some reason they don't have the value of 0.

```
In [388]: %matplotlib inline
          plt.figure(figsize=(17,10))
          plt.title("North Pol: Antenna 1")
          plt.plot(np.angle(d['gains'][1,:,1],deg=True),'b.',label='CASA Gain Solutions',lw=2)
          plt.plot(np.angle(np.mean(true_gains[yyants[1]], axis=0),deg=True),'r.',label='Genera
          plt.axvline(99, color='k', linestyle='--')
          plt.axvline(925, color='k', linestyle='--')
          plt.ylim(-180.,180.)
          plt.xlabel('Channel [arb.]')
          plt.ylabel('Phase [Deg.]')
          plt.legend(bbox_to_anchor=(0.8, 0.8), loc='upper right', borderaxespad=0.)

Out[388]: <matplotlib.legend.Legend at 0x13941cf28>
```



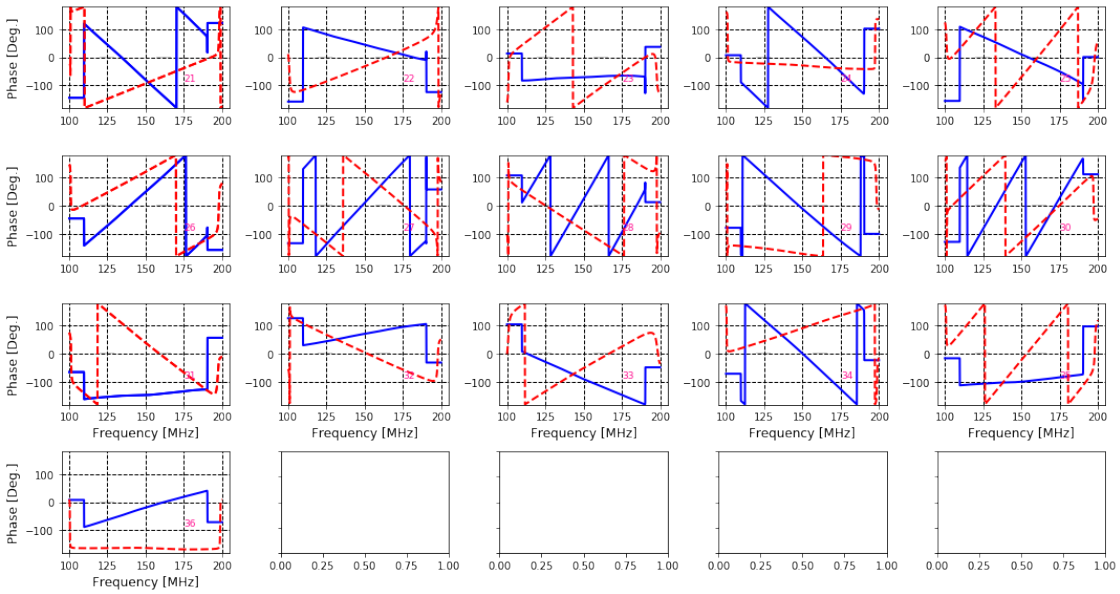
In [391]:
```
nrow,ncol = 4,5
numplots = nrow*ncol
ndays = len(npzlist)
iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

#msk_spec = np.zeros((1024))

f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
#f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

for i,a in enumerate(good_ants[:20]):

    #for ind in range(numplots):
    ax=plt.subplot(gs[iarr[i],jarr[i]])
    ax.set_ylim(-180.,180.)
    #ax.set_xlim(100,200)
    ax.annotate(str(a), xy=(175,-85), size=9, color='deeppink')
    ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
    ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

    for daynum in range(ndays):
        data = np.load(npzlist[daynum])
        xgain = np.angle(data['gains'][0,:,int(a)],deg=True)
        ygain = np.angle(data['gains'][1,:,int(a)],deg=True)
```

```
#print(list(msk_xgain))
        depenvar =ygain
        depenvarmodel = yyants
        if i > 14:
            ax.set_xlabel('Frequency [MHz]',size=12)
            ax.grid(color='k', linestyle='--', linewidth=1)
        if i == 0:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=
        if i == 5:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=
        #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
        if i == 10:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=

        if i == 15:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=

        else:
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg=
            ax.grid(color='k', linestyle='--', linewidth=1)

ax.legend(bbox_to_anchor=(2, 1), loc='upper right', borderaxespad=0.)
plt.tight_layout()
#f.savefig('2458115.24482.HH.uvOCR_BP.png')
plt.show()
```




```
In [392]: nrow,ncol = 4,5
          numplots = nrow*ncol
          ndays = len(npzlist)
          iarr = [0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3]
          jarr = [0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,0,1,2,3,4]

          #msk_spec = np.zeros((1024))

          f,axarr = plt.subplots(nrow,ncol,sharex=True,sharey=True,figsize=(15,8))
          #f.suptitle("Bandpass Solutions", fontsize=14)#Title centered above all subplots
          gs = gridspec.GridSpec(nrow,ncol)#, wspace=0.01, hspace=0.1)

          for i,a in enumerate(good_ants[21:]):

              #for ind in range(numplots):
              ax=plt.subplot(gs[iarr[i],jarr[i]])
              ax.set_ylim(-180.,180.)
              #ax.set_xlim(100,200)
              ax.annotate(str(a), xy=(175,-85), size=9, color='deeppink')
              ax.fill_between(freqs[201:300],-0.2,3.2, facecolor='k',alpha=0.2)
              ax.fill_between(freqs[581:680],-0.2,3.2,facecolor='k',alpha=0.2)

              for daynum in range(ndays):
                  data = np.load(npzlist[daynum])
                  xgain = np.angle(data['gains'][0,:,int(a)],deg=True)
                  ygain = np.angle(data['gains'][1,:,int(a)],deg=True)
```

```
#print(list(msk_xgain))
        depenvar =ygain
        depenvarmodel = yyants
        if i > 9:
            ax.set_xlabel('Frequency [MHz]',size=12)
            ax.grid(color='k', linestyle='--', linewidth=1)
        if i == 0:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Gain Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg
        if i == 5:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg
        #ax[0,0].set_title('Name')#gives plot at location (1,1) a title
        if i == 10:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg

        if i == 15:
            ax.set_ylabel('Phase [Deg.]',size=12)
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg

        else:
            ax.plot(freqs,depenvar,'b-',label='CASA Solutions',lw=2)
            ax.plot(freqs,np.angle(np.mean(true_gains[depenvarmodel[i]], axis=0),deg
            ax.grid(color='k', linestyle='--', linewidth=1)

#ax.legend(bbox_to_anchor=(2, 1), loc='upper right', borderaxespad=0.)
plt.tight_layout()
#f.savefig('2458115.24482.HH.uvOCR_BP.png')
plt.show()
```



In [ ]: