profileDomain

August 5, 2016

```
In [1]: # %load ProfileDomain.py

        %matplotlib inline
        %config InlineBackend.figure_format = 'retina'
        %load_ext autoreload
        %autoreload 2

        from __future__ import division
        import numpy as np
        from scipy.optimize import minimize

        import matplotlib
        import matplotlib.pyplot as plt
        matplotlib.rcParams['savefig.dpi'] = 1.5 * matplotlib.rcParams['savefig.dpi

        import psrchive
        from libstempo.libstempo import *
        import libstempo as T

        import corner as corner

        import PTMCMCSampler
        from PTMCMCSampler import PTMCMCSampler as ptmcmc

        from Class import *
```

```
/home/stephen.taylor/anaconda/lib/python2.7/site-packages/IPython/kernel/__init__.p
  "You should import from ipykernel or jupyter_client instead.", ShimWarning)
```

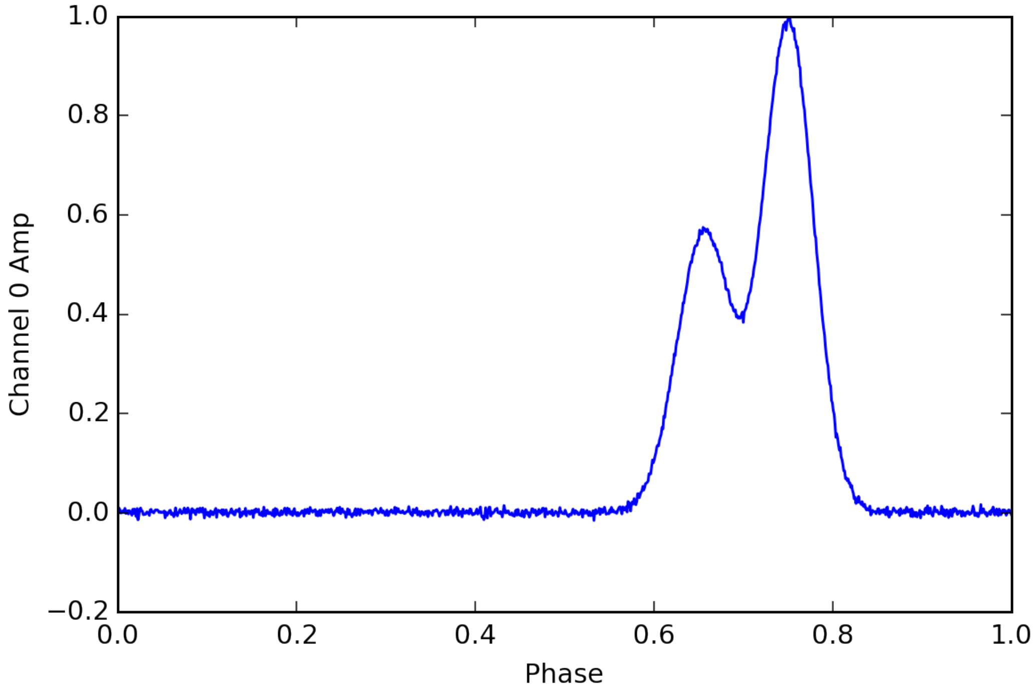
```
In [2]: lfunc = Likelihood()
        lfunc.loadPulsar("OneChan.par", "OneChan.tim", root='Sim1-OneChan')
```

```
fitting for:  RAJ 4.51091490212 3.69276954403e-10
fitting for:  DECJ 0.136026597599 7.13055014906e-10
fitting for:  F0 218.811840441 6.57924360336e-13
fitting for:  F1 -4.08406924053e-16 2.09108775722e-20
Loading Data:
Percent: [##########] 99.0%
```

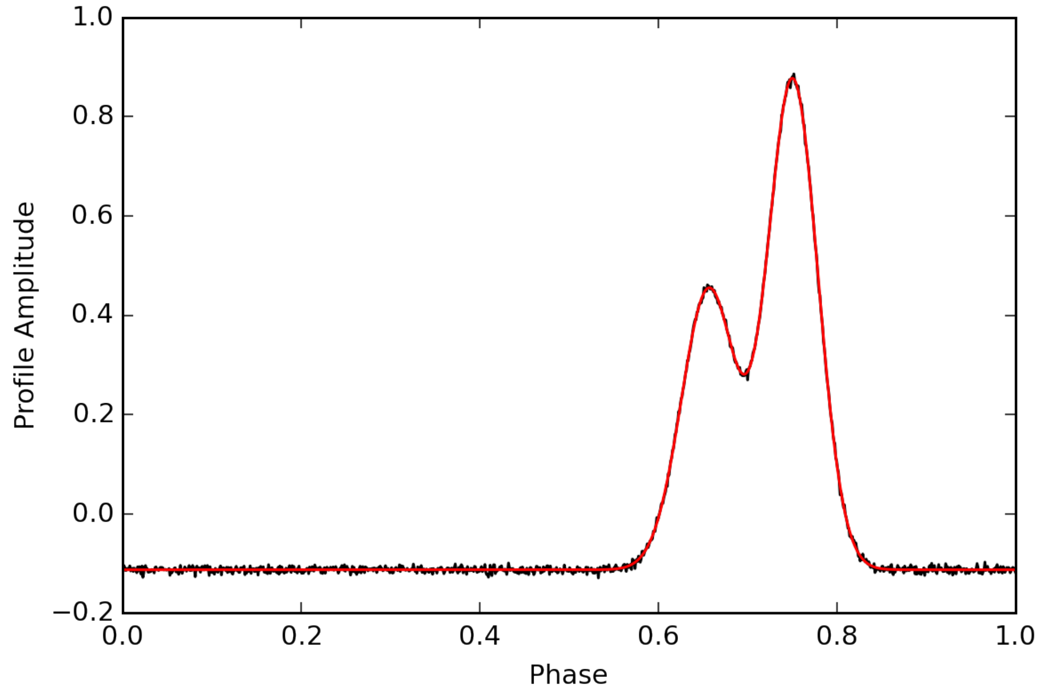
0.1 Get initial Fit to the Profile

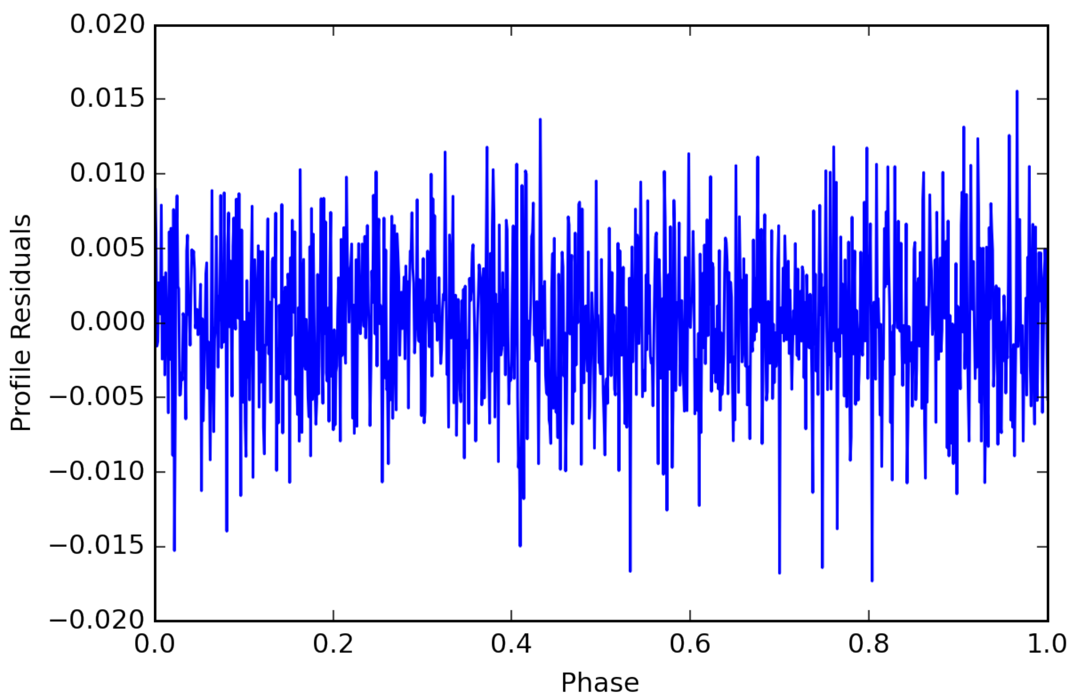
```
In [3]: lfunc.TScrunch(doplot = True, channels = 1)

        lfunc.getInitialParams(MaxCoeff = 20, cov_diag=[0.01, 0.1, 0.1],
                               resume=True, outDir = './InitFFTMNChains/Max20-',
                               sampler='multinest', incScattering = False,
                               mn_live = 1000,  fitNComps = 1, doplot = True)
```

```
Averaging All Data In Time:
Percent: [##########] 99.0%
```




```
Getting initial fit to profile using averaged data, fitting for:  ['Phase_0', 'Log1
ML: [ 0.1972955  -1.49933087  0.88310288]
```

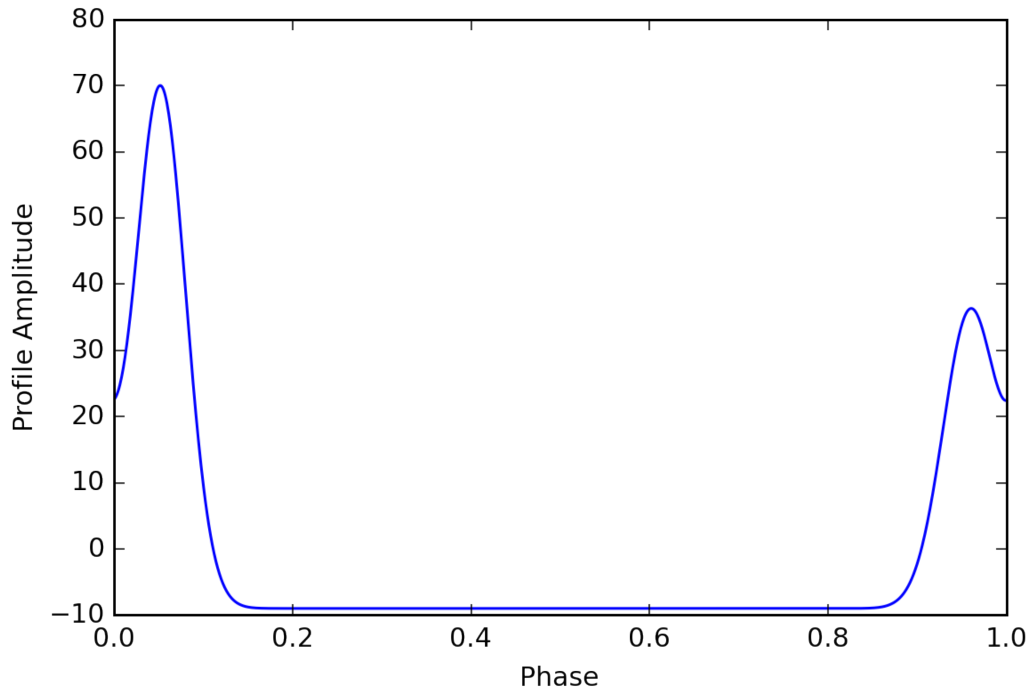

1.0
0.8
0.6
0.4
0.2
0.0
−0.2
Profile Amplitude
0.0
0.2
0.4
0.6
0.8
1.0
Phase


0.020
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.020
Profile Residuals
0.0
0.2
0.4
0.6
0.8
1.0
Phase

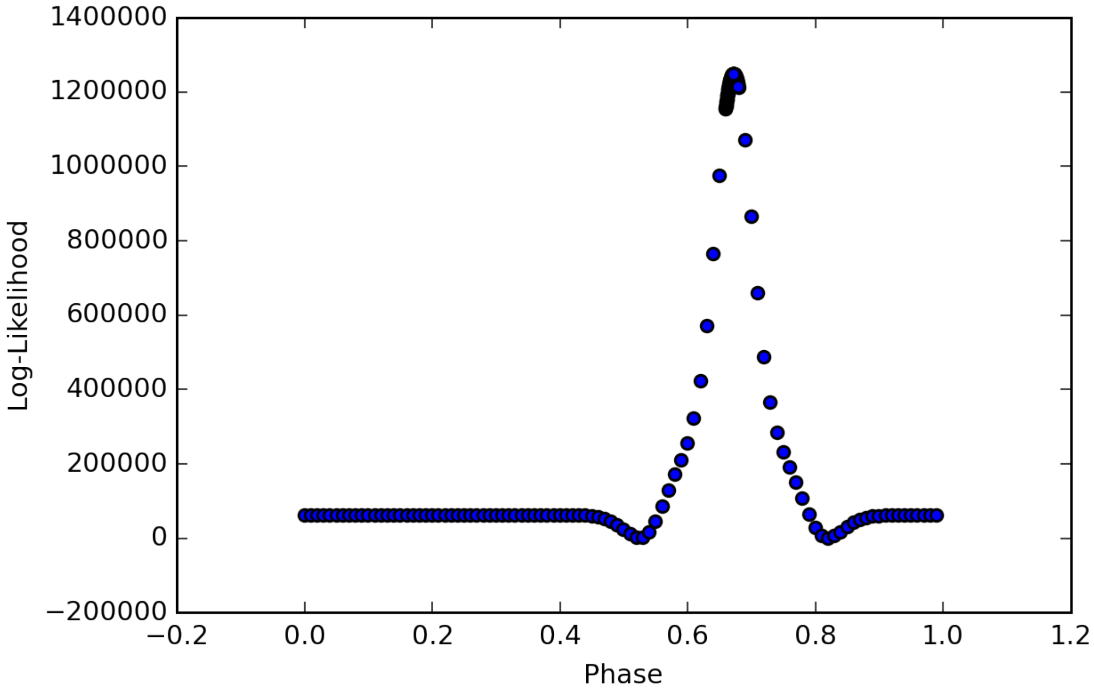
0.2 Make interpolation Matrix

```
In [4]: lfunc.PreComputeFFTShapelets(interpTime = 1, MeanBeta = lfunc.MeanBeta, dop
        lfunc.getInitialPhase(doplot = True)
        lfunc.ScatterInfo = lfunc.GetScatteringParams(mode = 'parfile')
```

```
('Calculating Shapelet Interpolation Matrix : ', 1, array([ 0.03167154]))
Percent: [##########] 99.9775935469% upper index is: 6 4201.8267943
upper index is: 11 3869.27937779
upper index is: 16 3647.81448924
upper index is: 21 789.528255895
upper index is: 26 32.8404566763
upper index is: 31 0.368376531381
upper index is: 36 0.00125879537642
upper index is: 41 1.39306356638e-06
upper index is: 46 5.17520155308e-10
upper index is: 51 6.60490586472e-14

Finished Computing Interpolated Profiles
```




```
Getting initial fit to phase using full data
Percent: [##########] 99.75%
```




```
Using Mean Phase:  [ 0.67239608]
```

0.3 Define parameter list and sampling ranges

```
In [5]: parameters = []
        parameters.append('Phase')
        for i in range(lfunc.TotCoeff-1):
                for j in range(lfunc.EvoNPoly+1):
                        parameters.append('S'+str(i+1)+'E'+str(j))
        for i in range(lfunc.numTime):
                parameters.append(lfunc.psr.pars()[i])
        for i in range(lfunc.NScatterEpochs):
                parameters.append("Scatter_"+str(i))


        print parameters
        n_params = len(parameters)
        print n_params
        lfunc.n_params = n_params

        pmin = np.array(np.ones(n_params))*-100
```

```
        pmax = np.array(np.ones(n_params))*100

        for i in range(lfunc.NScatterEpochs):
                pmin[-lfunc.NScatterEpochs+i] = -6
                pmax[-lfunc.NScatterEpochs+i] = 1

        lfunc.pmin = pmin
        lfunc.pmax = pmax
```

```
['Phase', 'S1E0', 'S2E0', 'S3E0', 'S4E0', 'S5E0', 'S6E0', 'RAJ', 'DECJ', 'F0', 'F1'
11
```

0.4 Define starting point for sampling

```
In [6]: x0 = np.array(np.zeros(n_params))

        pcount = 0
        x0[pcount] = lfunc.MeanPhase
        pcount += 1

        for i in range(lfunc.TotCoeff-1):
                for j in range(lfunc.EvoNPoly+1):
                        x0[pcount] = lfunc.MLShapeCoeff[1+i][j]
                        pcount += 1


        for i in range(lfunc.numTime):
                x0[pcount+i] = 0
        pcount += lfunc.numTime
        for i in range(lfunc.NScatterEpochs):
                x0[pcount+i] = lfunc.MeanScatter
        pcount += lfunc.NScatterEpochs
```

```
In [7]: lfunc.calculateFFTHessian(x0)
        covM=np.linalg.inv(lfunc.hess)
        lfunc.PhasePrior = np.sqrt(covM[0,0])*lfunc.ReferencePeriod
        lfunc.MeanPhase = x0[0]*lfunc.ReferencePeriod
```

```
In [8]: lfunc.doplot=False
        burnin=1000
        sampler = ptmcmc.PTSampler(ndim=n_params,logl=lfunc.FFTMarginLogLike,logp=l
                                    cov=covM, outDir='./Chains/',resume=False)
        sampler.sample(p0=x0,Niter=20000,isave=10,burn=burnin,thin=1,neff=1000)
```

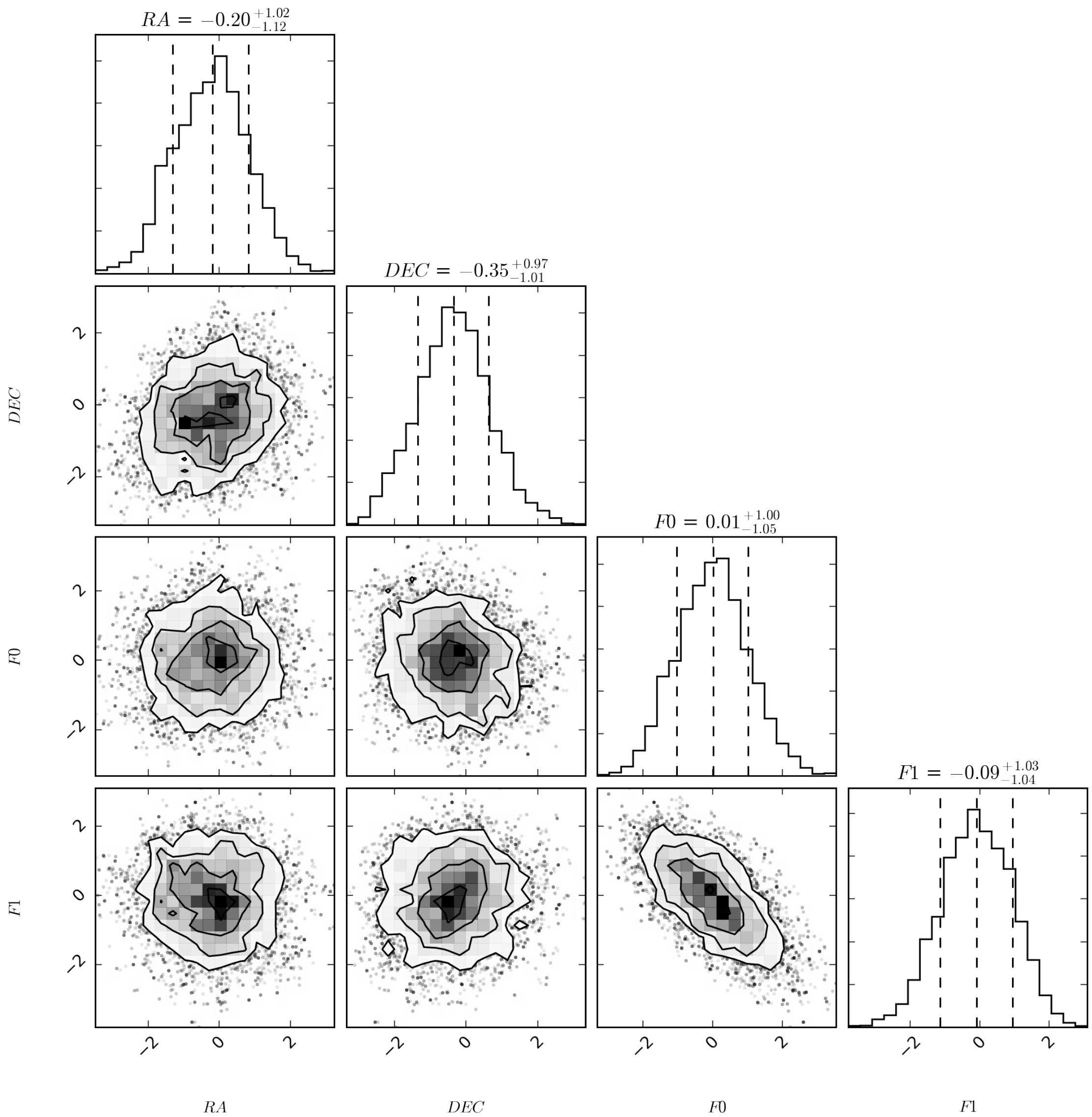
```
Finished 5.00 percent in 12.003634 s Acceptance rate = 0.351Adding DE jump with wei
Finished 99.95 percent in 235.164134 s Acceptance rate = 0.455178
Run Complete
```

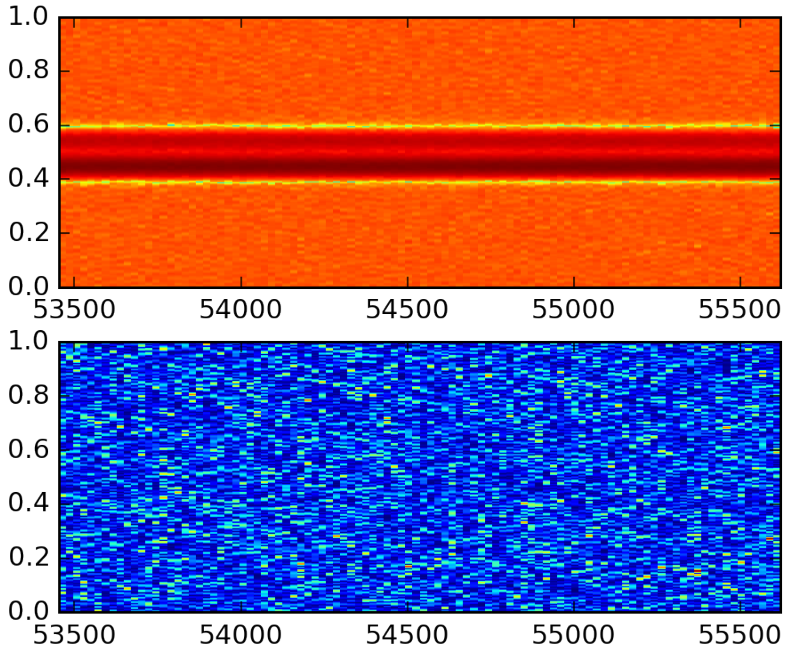
0.5 Load MCMC chain

```
In [9]: chains = np.loadtxt('./Chains/chain_1.txt').T
```

0.6 Make a plot

```
In [10]: chains = chains[:,burnin:]
         if(lfunc.numTime > 0):
                 Tchains = chains[1+lfunc.TotCoeff-1:1+lfunc.TotCoeff-1 + lfunc.num
                 figure = corner.corner(Tchains.T, labels=[r"$RA$", r"$DEC$", r"$F(
                                        quantiles=[0.16, 0.5, 0.84],
                                        show_titles=True, title_kwargs={"fontsize":

         ML = chains.T[np.argmax(chains[-3])][:n_params]
         lfunc.WaterFallPlot(ML)
```




1.0
0.8
0.6
0.4
0.2
0.0
53500
54000
54500
55000
55500
1.0
0.8
0.6
0.4
0.2
0.0
53500
54000
54500
55000
55500

```
In [ ]:
```