posterior

May 29, 2019

In [ ]:
```
import numpy as np
from scipy.interpolate import interp1d

class posterior:

    def lnlike(self, p, arg):

        cl_theo = p[0]*(self.cl_scal(p[1]) + p[2]*self.cl_tens)

        l = np.arange(2,self.lmax+1)
        like = -np.sum((2.*l+1.)/2.*(self.cl_data[l]/cl_theo[l]+np.log(cl_theo[l])))

        return like

    def lnprior(self, p, arg):
        if p[0]>0. or p[1]>arg[0] or p[1]<arg[1] or p[2]>0.:
            return 0.
        else:
            return -np.inf

    def lnpos(self, p, arg_p, arg_l):

        lnprior = self.lnprior(p, arg_p)

        if lnprior == -np.inf:
            return -np.inf
        else:
            return self.lnlike(p, arg_l)+lnprior

    def load_theory(self, file_scal, file_tens, lmax, ns_range):

        self.lmax = lmax

        ns = np.arange(ns_range[0], ns_range[1], ns_range[2])
        data_scal = np.load(file_scal)

        l = np.arange(1, lmax+1)
```

```
        fl = np.append(1., l*(l+1)/(2.*np.pi))

        self.interp = []
        for l in range(lmax+1):
            self.interp.append(interp1d(ns, data_scal[:,l]/fl[l]))
        del data_scal

        self.cl_tens = np.load(file_tens)[:lmax+1]/fl

    def cl_scal(self, ns):
        out = []
        for l in range(self.lmax+1):
            out.append(self.interp[l](ns))
        return np.array(out)

    def load_data(self, file_data, lmax):

        self.cl_data = np.load(file_data)

        l = np.arange(1, lmax+1)
        fl = np.append(1., l*(l+1)/(2.*np.pi))

        self.cl_data = self.cl_data[:lmax+1]/fl
```