Appendix B

Nested Sampling

The *nested sampling* algorithm is another example of a stochastic sampling algorithm that can be used to generate a sequence of independent, random samples from a target distribution.

One of the draw backs we identified with the Metropolis-Hastings MCMC algorithm was that it didn't give us the Bayesian evidence. Nested sampling does calculate the evidence. In fact, nested sampling is primarily an integration algorithm and produces random samples from the target distribution as something of a by-product.

The nested sampling algorithm is significantly more complicated than the Metropolis-Hastings algorithm and will not be described here; for more details see chapter 9 of D. S. Sivia, "Data Analysis: A Bayesian Tutorial" or the the paper by Skilling, J. *Nested Sampling*, AIP Conference Proceedings. 735: 395–405 (2004) DOI:10.1063/1.1835238.

As was the case with MCMC, there are ready to use implementations of the algorithm available. One such implementation is CPNest[1]

The following code snippet demonstrates how to use CPNest to sample from the ring distribution defined in appendix A.

```
import cpnest
from cpnest.model import Model

class ModelClass(Model):
```

[1] https://github.com/johnveitch/cpnest

```
    r0 = 5 # the radius of the ring
    names = ['x', 'y'] # we give a name to each of our parameters
    bounds = [[-8,8], [-8,8]] # we have to provide ranges for each parameter

    def log_likelihood(self, params):
        r = np.sqrt( params['x']**2 + params['y']**2 )
        return -0.5*(r-self.r0)**2
```

The above code defines a model class which is required by the CPNest package. This class contains a method called "log_likelihood". We can optionally also provide a "log_prior" method, but by default CPNest will use a flat prior. The code below creates an instance of this class (called "my_model") and samples from the distribution. The algorithm requires as input an integer number of live points (typically, a large number will give a more accurate estimate of the evidence integral).

```
my_model = ModelClass() # create an instance of the model class

nest = cpnest.CPNest(my_model,
                     nlive=1024,
                     output='output_folder') # setup the sampler

nest.run() # run the sampler

samples = nest.get_posterior_samples() # load the posterior samples
samples = np.loadtxt('output_folder/posterior.dat')[:, 0:2]

import corner # produce a corner plot
corner.corner(samples, labels=[r'$x$', r'$y$'])
```

The algorithm also produces the following estimate of the evidence integral[2]

$$Z = \int \mathrm{d}x \; P(x) \approx \exp(-1.18)\,. \tag{B.1}$$

[2] Warning: what CPNest calls the "evidence" in the output files is really the "log-evidence".

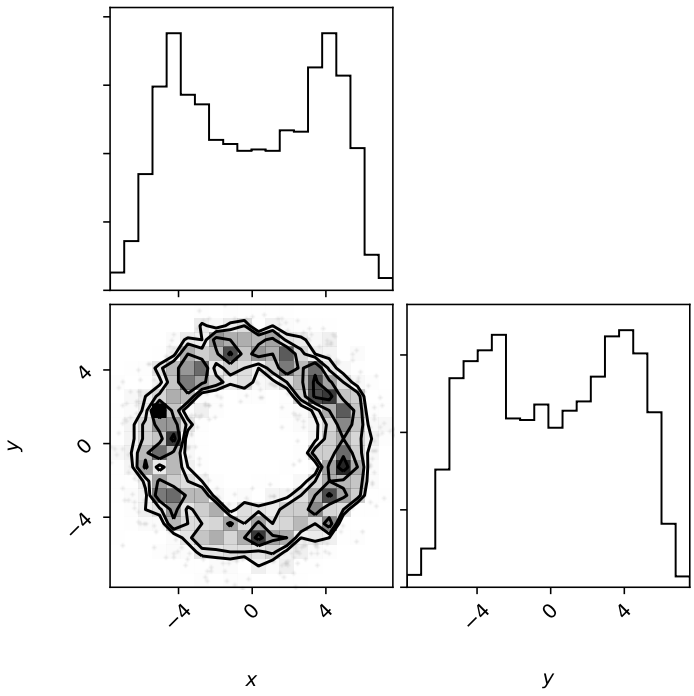

Figure B.1: The 2-dimensional histogram of the chain produced by the CPNest algorithm.