Slide 1 Title

Slide 2 Outline

Slide 3: For this project, I worked with data from the James Clerk Maxwell Telescope in the NESS group, or Nearby Evolved Stars survey, which has long term goals to investigate physical mechanisms such as AGB mass loss and dust formation.

As some of you may know, JCMT is a dish telescope designed for sub-mm observation, and for dishes, or antennae, the radiation pattern has larger values for a certain range of pointing direction than the rest of the sphere, and this area is called the main beam or main lobe, which you can see in this picture. The rest are called the side and back lobes, as you can see here. The main beam efficiency, or the fraction of power concentrated in the main beam, depends on elevation and weather, among other factors, and will therefore vary with time. It is required for calibration and therefore makes calibration somewhat difficult.

Slide 4: Here are two plots of main beam efficiency over time for two of JCMT’s heterodyne receivers, HARP on the left and RxA3 on the right. Each of these dots represents observations of planets, and the main beam efficiency calculated from them, as planets are convenient sources with known diameters that can be used for calibration. The main beam efficiency is essentially a multiplicative factor on the flux for all of JCMT data. The dots in red here represent daytime observations, which we removed from our dataset. The dotted line for HARP indicates a change to a more accurate pointing method, and the dotted lines for RxA3 are periods of misalignment of the receiver and the subsequent corrections. (click) This data on the left is the set I started with.

The full drawn line at the beginning of 2016 indicates the transition from RxA3 to RxA3m, when the receiver got a new mixer.

Slide 5: The initial project was to fit a function to characterize the time dependence of main beam efficiency and eventually integrate it into STARLINK, a suite of data reduction software, for Ness and the larger JCMT community. In other words we’d like to input a date and time, and get out a value for main beam efficiency to use as well as the uncertainty to propagate through. So in order to fit this model, we started by trying a method using Bayesian inference and MCMC.

Slide 6: Bayesian Inference is an approach that uses Bayes’ Theorem to incorporate new or already known information, such as physical limitations, into computing probabilities. Instead of saying “given a model, how likely is it to observe my data?”, we say “given my data (which I already have and have observed), how likely is this model?” which is the question that scientists usually actually want to answer.

Slide 7: This is Bayes’ Theorem in the case of a modeling problem. On the left side, (click) we have the posterior, and the way this is read is the probability of the model given our collected data is true, where our model is represented by some parameters theta, and our data by the variable y. On the right we have the likelihood (click), the probability of our collected data given the model is true, and the prior (click), which is the probability of the model being true before collecting the data. Finally, we have the evidence (click), the probability of collecting the data under all models. So we have the posterior equals the likelihood times the prior divided by the evidence.

Slide 8: The next part of the method is using MCMC, which stands for Markov Chain Monte Carlo. The Markov Chain part has to do with a particular sequential method of generating random samples, where new samples depend exclusively on the previous one. The Monte Carlo part has to do with drawing samples from a distribution in order to estimate properties about it. Essentially, MCMC is a set of algorithms for sampling from a probability distribution without having to know all of its mathematical properties, for example its exact height. I used Emcee, which is a specific Python implementation of an MCMC ensemble sampler by Foreman-Mackey and collaborators. This process works by having a set of “walkers,” as we’ll call them, step through the parameter space, at each step accepting or rejecting samples. We end up with a distribution of values for each parameter, from which we can obtain statistics. What we are essentially doing in exploring the parameter space, or set of possible parameters, in order to find the maximum likelihood.

Slide 9: So to reiterate quickly, we write up our likelihood, which describes the distribution of the data, we write up our priors, which include information about how we know the parameters must be limited, and we obtain something that is proportional to the posterior. Since the denominator is the probability of the data under all models, it is in reality difficult to compute. So instead we use MCMC to sample from the numerator, since it doesn’t require the height of the distribution anyway.

Slide 10: Let’s take a look at this example from the emcee documentation, which was also the starting point for my project, and illustrates a how the process should work. We start with data generated from a line mx+b with some underestimated uncertainty f. Here we can see the paths that the walkers take in exploring the parameter space, and you can see that they roughly spread around a central point for each parameter. They’re all exploring the space well, and keep in mind this is what we’d expect the walker paths to look like.

* Starting positions for walkers
* Range around starting positions for walkers
* Number of walkers
* Number of steps and step size, Point out burn in period

Slide 11: So after the MCMC process, we get a nice contour plot, which is a visual display of the sampling results. Here we have m, b, and the fractional uncertainty, and we can see that the parameter distributions look roughly Gaussian, as we’d expect from a simple case like this, and from those we can get an uncertainty estimate. We can also see how the parameters are related, so m and b are dependent, as this is more narrow, whereas looking at f vs m and b, the samples are distributed more widely, and show no dependence. So let’s move on to constructing a new likelihood.

Slide 12: So in our model, we wanted to account for uncertainty in the y direction, scatter, and bias, or systematic error. To quickly distinguish between these terms, the uncertainty is due to the measurement of the data, the scatter is due to sorts of extra complicated physics that we can’t put into the model explicitly, and the systematic error is a consistent mismeasurement, in our case the misalignment of the telescope. We started again with a simple linear model with a 10% uncertainty. This resulting fit is pretty similar to that of a simple least squares fit, as it hasn’t accounted for factors that we know to be present, such as the bias.

Slide 13: The next step was to construct a more complicated likelihood, one that accounts for uncertainty and scatter. This follows the convention and equations derived in Hogg, Bovy, and Lang 2010. We model the distribution beginning with a set of points lying on a narrow linear relation, with a Gaussian offset drawn from a 2D Gaussian with a covariance tensor. In our case, we only have y uncertainty, so the matrix algebra simplifies out nicely. We can then introduce a intrinsic Gaussian variance V, or the scatter, which is orthogonal to the line. The likelihood then becomes the convolution of the 2D uncertainty Gaussian with V, resulting in this likelihood. These terms essentially just show the formalism of constructing orthogonal distance.

Slide 14: We then moved on to implementing the bias using a similar scatter-like term. This didn’t work as we had hoped, as the scatter is a symmetric term, while the bias is asymmetric and can only lower values. Hogg and collaborators provide as well a section in their paper on non-Gaussian uncertainties, which allows one to model noise using a mixture of k Gaussians and is described by the probability function here. We can see here an “a” coefficient that controls the relative strength of the different Gaussians, and which we’ll discuss a bit later.

Slide 15: To implement this, we started with the case of 2 Gaussians for simplicity, temporarily ignoring scatter. You can see in this plot where the blue represents uncertainty and the orange the bias, that the distribution is very bimodal. This caused the walkers to branch, prioritizing strictly one Gaussian or the other. You can see that the a term for the first Gaussian, a1, takes the value of either 1 or 0. We then added another Gaussian to smooth the distribution, but for some reason this forced the walkers to constrain themselves to a narrow range near zero.

Slide 16: You can see this in the very strange walker paths, which were not fully exploring the parameter space and had some strange flare-ups at the end. The resulting means for the distributions provide what *looks* like a reasonable fit according to this plot, but the uncertainties for the bias in particular were nonsensically large, too large to plot here, on the order of 10 to the 7.

Slide 17: In the hopes of a more general method, I coded up a matrix version of the non Gaussian uncertainties equation derived by Sundar, and also implemented a Lagrange multiplier in the hopes that it would fix the weighting problem, which in a sense it did by forcing them to be non-zero, but also caused several parameters to be essentially unconstrained, you can see that the walkers here freeze in place all across the parameter space. In addition, while we had sets of resulting reasonable parameters, they seemed to be largely random and irreproducible.

Slide 18: In the end, we learned that it’s very complicated to use this method for non-Gaussian likelihoods or asymmetric uncertainties, as we had in our case with the bias. The method should have in principle worked, but the results were largely inconclusive and inconsistent. I did test the method on similar fake generated datasets, but similar problems occured. A few possible reasons for these results are that the model fails to account for everything that is going on, or perhaps this is just a poor method for this dataset, we’re not quite sure.

Slide 19: Instead, we moved on to a different method of fitting the data -- forward modeling. (click) The process is to first create a model that generates datasets based on input parameters. (click) Then to create a grid that represents your parameter space. (click) Then for each set of parameters we generate 1000 sample datasets, in order to obtain a distribution of results that will allow us to estimate our uncertainties -- this is a Monte Carlo method. We then (click) compare these datasets to the real data in order to (click) determine the best fit, complete with uncertainties.

Slide 20: Once again, we started with a simple generative model of a line mx+b. We added a y error (click) and scatter (click) drawn from normal distributions, and a bias term drawn from a half normal (click). And finally we have (click) the red representing the full distribution. We eventually removed the scatter term, as the y error was large enough to account for it. We also changed the bias to an exponential distribution to fully get the lower values, and added a multiplicative factor to the bias term as well.

Slide 21: In order to get a measure of goodness of fit, we used the standard KS test, which stands for Kolmogorov-Smirnov test. This test compares two Cumulative Distribution functions. The CDF of a variable big X evaluated some value is the probability that X will take that value or smaller. So for example in this plot, the red line looks like the CDF of a normal distribution, so if we look at the probability at 0, 50% of the points in the distribution will be smaller than zero, and we can see at the end this converges to 1, as eventually we’ll reach a value that encompasses all the values of the distribution. The KS test essentially looks at the maximum distance between two CDF’s as the test statistic, and what we did was minimize this stat. This plot shows the CDF’s of all the datasets that are tested against the data for a very coarse grid, with each different color being a different dataset.

Slide 22: Because of the computation time required to run an extremely fine grid, I instead ran coarse grids 5 times, each time refining the range of the grid based on the best fit from the previous one. Using the Monte Carlo method, I was also able to obtain uncertainties based on the distribution of parameters for the top 1% best fits. Here is a plot of the CDF of data generated using the best fit parameters in comparison with the CDF of the actual data, and as you can see it’s a fairly good fit. This is a plot of the actual data with the sampled best fit data overlaid as well.

Slide 23: So to summarize, I talked about two methods of fitting a model for main beam efficiency, and wanting to account for uncertainty, scatter, and bias. First the MCMC method, which gets very difficult when you’re not working strictly with Gaussians, and which wasn’t very effective or consistent for the HARP dataset, although it’s not very clear why. I talked also about the ultimately more productive method of forward modeling, which is somewhat less statistically robust, but actually produced reasonable best fit results and uncertainties.

Slide 24: While I was able to start looking at using a finer grid for forwarding modeling, I ran out of time to fully troubleshoot, and this would be useful for providing more robust uncertainties. I also didn’t get to looking at the RxA3 data, which as I mentioned previously is segmented by periods of misalignment and requires fitting a function to each individual section. This will pose its own set of difficulties. Finally, we could also look into maximizing entropy instead of the likelihood for the MCMC method.

Slide 25: Acknowledgements