Draft version July 21, 2017
Typeset using LaTeX **twocolumn** style in AASTeX61

INFERENCE OF THE CONDITIONAL LUMINOSITY FUNCTION BY EXPLICIT MARGINALIZATION OVER ALL INDIVIDUAL GALAXY PARAMETERS

David Thomas,[1] Phil Marshall,[1,2] and Risa Wechsler[1,2]
(LSST Dark Energy Science Collaboration)

[1]*Kavli Institute for Particle Astrophysics & Cosmology, P. O. Box 2450, Stanford University, Stanford, CA 94305, USA*
[2]*SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA*

ABSTRACT

We explore a hierarchical model for galaxy luminosities and halo masses whose hyperparameters govern the conditional luminosity function, and whose many individual object parameters are explicitly marginalized over. We investigate four techniques to perform the high dimensional marginalization, and comment on their performance.

Keywords: Galaxies: halos, Galaxies: statistics, Methods: numerical

1. INTRODUCTION

The large scale structure of the universe is dominated by the collective gravitational influence of dark matter particles and the repulsive force of dark energy. There are ongoing efforts to infer the masses and locations of dark matter halos and improve measurements of dark energy parameters. Observational progress has been gradual, creating opportunity for simulations. Given a cosmology, large N-body simulations can predict characteristics like clustering and the halo mass function with high precision (eg. Springel, Frenk, & White 2006). Simulated universes are readily available and inspire our effort to connect theory to empirical observations.

The halo mass and galaxy luminosity relation is a conduit between dark matter halos and galaxy observations. The conditional probability allows us to convert the dark matter halos from simulations into distributions of galaxy luminosities in different redshift bins that we can compare with observations. The exact form of the relation is an active area of research; models that separate central and satellite galaxies have been proposed (eg. Yang et al. 2005).

Weak lensing is another technique that can characterize the halo composition of our universe. The gravitational influence of a massive dark matter halo causes the light from nearby galaxies to bend as it passes by, leading to a shearing effect. Over a swathe of sky, typically on the order of 1000s of square arcminutes, the collective shears allow the masses and locations of halos to be weakly inferred. These inferences grow stronger as the density of galaxy observations increases. The ongoing Dark Energy Survey and upcoming Large Synoptic Survey Telescope will provide dense galaxy catalogs which will increase the potential of this approach (Dark Energy Survey Collaboration et al. 2016; LSST Science Collaboration et al. 2009).

Traditionally the mass luminosity relation and weak lensing have been pursued in isolation, each ignoring information the other could provide. It is our goal to harness the full constraining power of observations by modelling both weak lensing and the mass luminosity relation in a self consistent, heirarchical framework. We employ the mass luminosity relation described in Reddick (2014) and the mass mapping infrastructure from Marshall (2006). Figure 1 outlines the dependencies of our ultimate model.

The model partitions galaxies into two groups. Background galaxies are sources that lie in a redshift bin immediately behind the three dimensional space of foreground galaxies. The background galaxies provide the observed shears which we use to infer the halo masses of foreground galaxies. We plan to use around 200,000 foreground galaxies and 1,600 background galaxies in our model and analysis. In the course of the heirarchical inference, each of the 200,000 galaxies contributes and has a corresponding mass and luminosity which must be marginalized out. Given the large size of our inference, we expect the primary challenges to be computational in nature - how can we achieve inference at this unprecedent scale? In order for us to rapidly iterate towards an answer to this question, we study a model with simpler dependencies. This condensed model, shown in Figure 2, offers the benefit of being easier to test while retaining the computational challenges that need to be adressed.

The data we use is derived from the Millennium Simulation (Springel et al. 2005). Our selection is driven

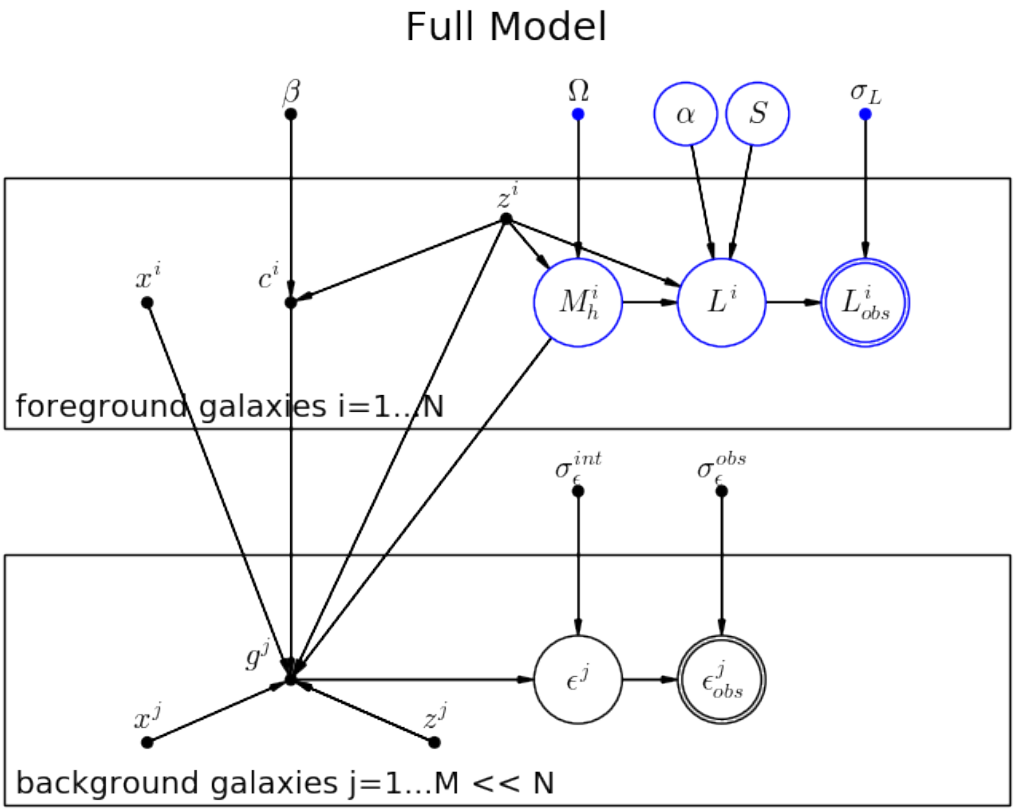

Figure 1. Above is a probabilistic graphical model for the full model we aspire to. There are two plates; one for foreground galaxies, one for background galaxies. The cosmology hyperparameters are colored green, the mass luminosity relation hyperparameters are colored blue, and the noise hyperparameters are colored red. Below is a table with all the parameters and corresponding descriptions.

Parameter	Description
β	Form of the halo concentration.
Ω	Cosmology used to generate the halos in the Millennium Simulation.
α	Four parameters that convert mass to mean luminosity.
S	Scatter of the mass luminosity relationship.
M	Halo mass.
L	Galaxy luminosity.
x	Angular position of galaxy.
z	Galaxy redshift.
c	Halo concentration.
g	Reduced shear.
ϵ	Galaxy shear.
σ	Various forms of noise.

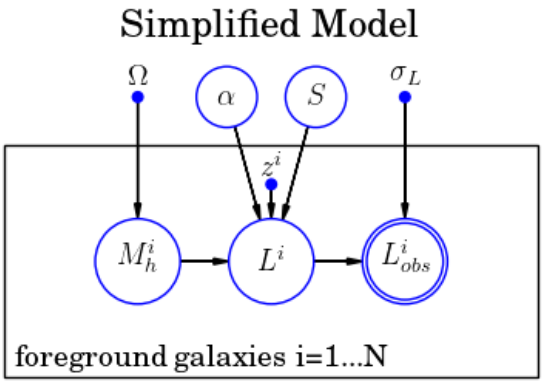

Figure 2. A probabilistic graphical model for the simplified model we will use for testing. The parameters are a subset of those in Figure 1.

by our desire to incorporate weak lensing into our ultimate model, which necessitates the large number of halos. Furthermore, Hilbert et al. (2009) have also performed raytracing on this dataset which we plan to make use of in our weak lensing inference. There are two transformations that are applied on top of the simulation output. First, a friend-of-friend (FoF) group finding algorithm identifies halos from the particle concentrations in the simulation output. Second, a 40×40 square arcminute window of sky out to redshift 3.5 is carved out from the output. The interior halos form the foreground galaxies in our analysis. The resulting dataset consists of 115,919 halos with accompanying positions (right-ascension, declination, and redshift).

The statistical inference we perform follows the canonical inference formula: posterior equals prior times likelihood. In the analysis that follows we use an overline, such as $\overline{z}$, to distinguish a vector over all foreground galaxies from the variable corresponding to a single halo, or z. Given the mass luminosity hyperparameters α, S; a vector of observed luminosities $\overline{L^{obs}}$; a vector of redshifts $\overline{z}$; and the observational noise in luminosity measurements σ_L^{obs} - the posterior we seek is

$$\underbrace{P(\alpha, S|\overline{L^{obs}}, \overline{z}, \sigma_L^{obs})}_{\text{Posterior}} = \underbrace{P(\alpha, S)}_{\text{Prior}} \underbrace{P(\overline{L^{obs}}|\alpha, S, \overline{z}, \sigma_L^{obs})}_{\text{Likelihood}}$$

Using the probabilistic graphical model we can further factor this to

$$P(\alpha, S|\overline{L^{obs}}, \overline{z}, \sigma_L^{obs}) = P(\alpha)P(S) \iint \overline{dM}\ \overline{dL}\ P(\overline{L^{obs}}|L, \sigma_L^{obs})P(\overline{L}|\overline{M}, \alpha, S, \overline{z})P(\overline{M}|\overline{z})$$

The likelihood integral involves integrating over 200,000 variables - the mass and luminosity for each galaxy in our dataset. The Methods section of this paper describes various approaches to make such a high dimensional integral computationally tractable.

The integrand of the likelihood contains three probabilities. The first is the conditional observed luminosity probability. Since luminosities are usually reporeted in log-space, we assume that there are gaussian errors in the log-space luminosity measurements, or that the distribution is log-normal. We believe 5% errors seem reasonable and fix $\sigma_L^{obs} = 0.05$. Our conditional distribution is

$$P(\overline{L^{obs}}|\overline{L}, \sigma_L^{obs}) = \frac{1}{\overline{L^{obs}}\sigma_L^{obs}\sqrt{2\pi}} \exp\left(-\frac{(\ln \overline{L^{obs}} - \ln \overline{L})^2}{2\sigma_L^{obs\ 2}}\right)$$

The second factor in the likelihood integrand is the mass luminosity relation $P(\overline{L}|\overline{M}, \alpha, S, \overline{z})$. We use the form described in Reddick (2014). Reddick follows the convention from previous work where the conditional luminosities is divided into two parts, central and satellite galaxy contributions. The central contribution is described by a lognormal distribution and the satellite contribution is described by a Schechter function. In our analysis, we ignore the satellite contribution. This decision is made for convenience, and we can easily insert the satellite contribution in future analyses. We use $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ to represent the parameters for computing the mean luminosity and S for the scatter. This gives

$$P(\overline{L}|\overline{M}, \alpha, S, \overline{z}) = \frac{1}{\overline{L}S\sqrt{2\pi}} \exp\left(-\frac{(\ln \overline{L} - \ln \overline{\mu_L})^2}{2S^2}\right)$$
$$\mu_L = \exp(\alpha_1) \cdot \left(\frac{M}{\alpha_3}\right)^{\alpha_2} \cdot (1+z)^{\alpha_4}$$

The third factor in the likelihood integrand is the halo mass function $P(\overline{M}|\overline{z})$. For this we rely on the python software package *hmfcalc* (Murray, Power, & Robotham 2013). In the API we select the functional form from (Tinker et al. 2008) and the cosmology from the Millennium Simulation to govern the mass function. The API returns 400 logarithmically spaced mass samples and the derivative of the number density $\frac{dn}{dM}$. We convert this into a probability density with trapezoidal integration. The function also has a dependence on redshift, which we account for by employing 20 redshift bins and matching the halos in the different redshift bins with the corresponding prior. Figure 3 shows the priors for each redshift bin.

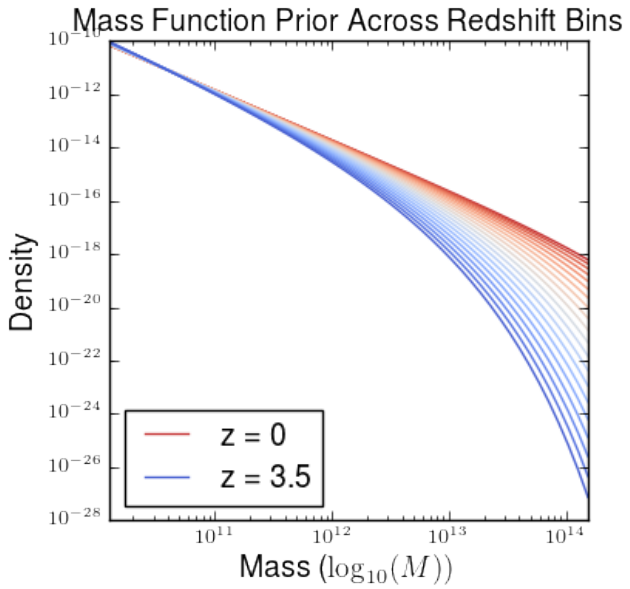

Figure 3. The mass function prior for the range of redshift bins from z = 0 to z = 3.5.

In Subsection 5.2 we discuss further modifications to the prior that allow it to work in concert with both sampling and logarithms.

2. METHODS

The critical challenge of the inference is computing the high dimensional likelihood. In this section we describe the test dataset used in the inference and four different methods for computing the high dimensional likelihood.

2.1. *Test Dataset*

The scaffold for the test dataset is the catalogue of dark matter halos from Hilbert et al. 2009. This scaffold provides the redshift and mass of each halo. We augment this initial dataset with a luminosity and observed luminosity for each halo. In order to do this we need to use the mass luminosity relation which requires the aforementioned set of hyperparameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4, S$. We draw a set of hyperparameters from the distributions suggested in Reddick (2014) and fix it for the entire dataset. Then we use this fixed set of hyperparameters and the mass luminosity relation to generate luminosities for each halo. Then the noise model is used to generate observed luminosities for each halo.

After testing various methods for computing the likelihood, we can use the dataset to do a complete test of the inference. The inference only consumes the redshift and observed luminosity fields of the dataset and produces a hyperparameter posterior. This posterior can then be compared against the fixed set of hyperparameters that generated the dataset. If the process produces posteriors consistent with the fixed values then we can be confident that it will also be able to provide accurate hyperparameter posteriors for real datasets.

The original likelihood integral can be factored into a product of integrals corresponding to each halo. Moving forward in this section, we use the convenient substitutions

$$P_1 = P(L^{obs}|L, \sigma_L^{obs})$$
$$P_2 = P(L|M, \alpha, S, z)$$
$$P_3 = P(M|z)$$

Then letting D be the dataset of halos we have

$$\mathcal{L}(\overline{L^{obs}}|\alpha, S, \sigma_L^{obs}, \overline{z}) = \prod_{(L^{obs},z)\in \mathrm{D}} \mathcal{L}(L^{obs}|\alpha, S, \sigma_L^{obs}, z)$$
$$= \prod_{(L^{obs},z)\in \mathrm{D}} \int dM dL \; P_1 P_2 P_3$$

Initially, we narrow our focus to a single one of these integrals. After we characterize the tradeoffs in the single integral case we expand our analysis to computations of the entire product of integrals.

2.2. *Numerical Integration*

Numerical integration is the canonical way to approximate a continuous integral on a computer. The domain of integration is partitioned into small rectangles. Then the integrand is evaluated at the center of each rectangle and multiplied by the surrounding rectangle's area to approximate. This approximates the true region under the curve over the rectangle. Summing up the contributions from all the rectangles in the domain provides an approximation of the entire integral.

The precision can be sensitive to the partitioning of the domain. Smaller rectangles around a point will provide more precise approximations. By increasing the number of rectangles in the partition, and decreasing their area, one can expect to achieve a more precise approximation of the integral. The partitioning in the mass dimension is somewhat limited by our prior, which is comprised of a normalized, linear approximation on the 410 mass values provided by *hmfcalc*. Breaking the mass up into more than 410 pieces would increase our reliance on the linear approximation of our prior when computing the integrand, which limits its benefit. In our implementation we fix the mass partitioning to be the same 410 masses provided by *hmfcalc*. The runtime

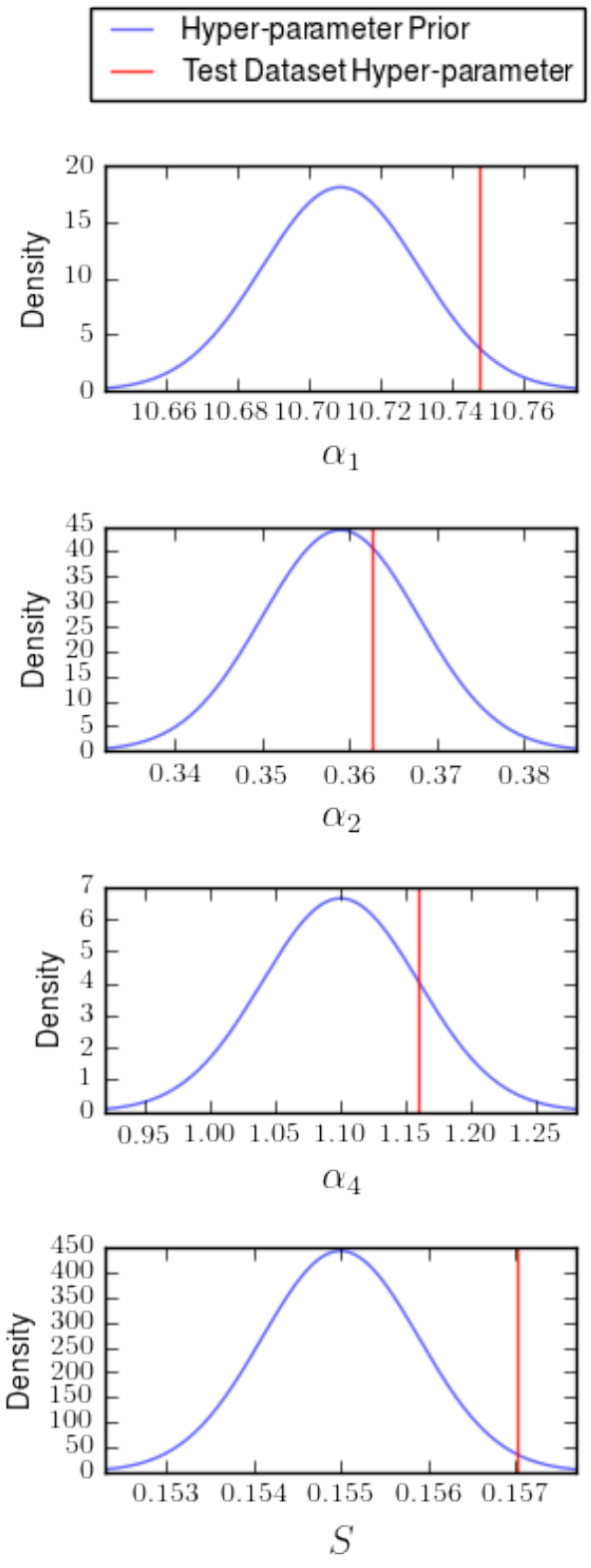

Figure 4. The hyperparameter prior is shown in blue and the drawn hyperparameter value for the test dataset is shown in red.

is $O(mn)$ where m is the number of mass partitions and n is the number of luminosity partitions. In our experiments we found the scale of the luminosity partition, linear or logarithmic, to have limited impact on the result and have chosen to stick with logarithmic for slightly improved accuracy when n is small. Letting X and Y be the logarithmically spaced ranges of M and L respectively, gives

$$\mathcal{L}(L^{obs}|\alpha, S, \sigma_L^{obs}, z) = \int dM dL \; P_1 P_2 P_3 \approx \sum_{L\in X} \sum_{M\in Y} \Delta_L \Delta_M \; P_1 P_2 P_3$$

Most of the contribution to the value of the integral comes from a small region in the domain. Intuition suggests that if we were able to selectively sample from this region we would be able to compute the integral more efficiently. This line of thought leads us to simple monte carlo integration.

2.3. *Simple Monte Carlo*

Simple monte carlo integration is a method for computing an integral when the variables of integration are probabilities in the integrand. We can approximate the integral over the probability distribution with samples from the distribution. In our likelihood, we employ this approximation for both the conditional luminosity $P(L|M, \alpha, S, z)$ and the prior $P(M|z)$. With these modifications our integral becomes

$$\mathcal{L}(L^{obs}|\alpha, S, \sigma_L^{obs}, z) = \iint dL dM P_1 P_2 P_3 \approx \frac{1}{N_s} \sum_{L\sim \; P_2} \sum_{M\sim \; P_3} P_1$$

This joint sampling generates samples with larger products $P(L|M, \alpha, S, z)P(M|z)$. It could be that the remaining factors in the integrands, $P(L^{obs}|L, \sigma_L^{obs})$, have low weights and mitigate the benefits of this sampling. This would happen if a halo has an outlier observed luminosity and is not sampling nearby luminosities. Our setup is particularly prone to this because of the major disparity between the distributions for the mass function prior and the masses of the dataset in the high mass region. This suggests that this approach may not be able to accurately compute the likelihood for halos with large observed luminosities.

2.4. *Importance Sampling*

Consider the rare halo with exceptionally large observed luminosity. Simple monte carlo will generate many low mass, low luminosity samples and rarely sample the high weight region of the likelihood integral. The sampling would be more efficient if the scheme could adjust its sampling to accomodate the large observed luminosity. Importance sampling is a statistical technique that makes such strategic sampling possible without altering the result.

The basic idea behind importance sampling is multiplying the integrand by $1 = Q/Q$, where Q is a strategically designed distribution, which is also nonzero over the domain of integration. Then instead of sampling from the original distribution P, the technique samples from Q and weighs the samples by P/Q. For example, if the simple monte carlo integral is $\int dx\ P(x) = \int dP(x)$, then the importance sampling integral is $\int dx\ \frac{P(x)Q(x)}{Q(x)} = \int dQ(x)\ \frac{P(x)}{Q(x)}$. A powerful property of importance sampling is that is an unbiased estimator (Appendix 5.3 contains the proof). This guarantee gives us the freedom to design a biased distribution Q to generate efficient samples for our likelihood integral. To get the most efficient sampling, we strive to develop a biased distribution Q that samples the 'important' region where the weights P/Q are high.

Recall that both the conditional luminosity and conditional observed luminosity relations are lognormal. Given an observed luminosity, we can reverse the lognormal, keeping the same mean, to generate an approximate distribution of luminosities. For example, to get L^{obs} from L we would take samples from the integrand factor $P(L^{obs}|L, \sigma_L^{obs})$. To get an approximation of L from L^{obs} we can use

$$Q(L|L^{obs}, \sigma_L^{obs}) = \frac{1}{L\sigma_L^{obs}\sqrt{2\pi}} \exp\left(-\frac{(\ln L - \ln L^{obs})^2}{2\sigma_L^{obs\ 2}}\right)$$

Similarly, to get an approximation of M from L we reverse the conditional luminosity relation, and reverse the corresponding mean mass luminosity relation as well.

$$Q(M|L, \alpha, S, z) = \frac{1}{MS_Q\sqrt{2\pi}} \exp\left(-\frac{(\ln M - \ln \mu_M)^2}{2S_Q^2}\right)$$
$$\mu_M = \alpha_3 \left(\frac{L}{\exp(\alpha_1)(1+z)^{\alpha_4}}\right)^{1/\alpha_2}$$

The reader may notice how on the right hand side of the first equation above we have replaced S with S_Q. This is because we use $S_Q = \lambda \cdot S$, where λ is the constant which converts the scatter in conditional luminosity to the scatter in conditional mass. We derive λ through an error propagation technique described further in Appendix 5.5.

For convenient notation we let

$$Q_1 = Q(L|L^{obs}, \sigma_L^{obs})$$
$$Q_2 = Q(M|L, \alpha, S, z)$$

Combining the above distributions into a joint biased distribution for importance sampling yields the formula below for the single halo likelihood integral.

$$\mathcal{L}(L^{obs}|\alpha, S, \sigma_L^{obs}, z) = \iint dLdM \frac{P_1P_2P_3Q_1Q_2}{Q_1Q_2}$$
$$= \frac{1}{N_s} \sum_{L\sim Q_1} \sum_{M\sim Q_2} \frac{P_1P_2P_3}{Q_1Q_2}$$

To validate the effectiveness of the biased joint distributions (the Q's) we generate nine different test datasets. For each dataset, hyperparameters are drawn from the prior distribution (defined in Subsection 2.1), and used to generate a mass, luminosity, and observed luminosity for each halo in the catalogue. Then we generate samples from the joint distribution via simple monte carlo and importance sampling and compare how efficiently the two approaches cover the true joint distribution. Figure 5 shows the nine corresponding scatterplots of the first 1,000 samples in the mass and luminosity dimensions. As expected, simple monte carlo has poor coverage of the the high mass, high luminosity region. Importance sampling has much better coverage.

2.5. *Laplace Approximation*

The final scheme we explore adheres to paradigms from optimization. In this paradigm we view the integrand as a smooth surface with a hill that corresponds to the high weight, or important region of the integrand. To compute the integral we iteratively hike up the landscape to the highest point on the hill, the maximum of the integrand. From this vantage point we approximate the hill with an easy-to-integrate multivariate distribution centered on the maximum. The integral of the distribution serves as a rough approximation of the original integral.

To find the maximum of the integrand we use the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method implemented in *scipy.stats.minimize* (Jones et al. 2001–).

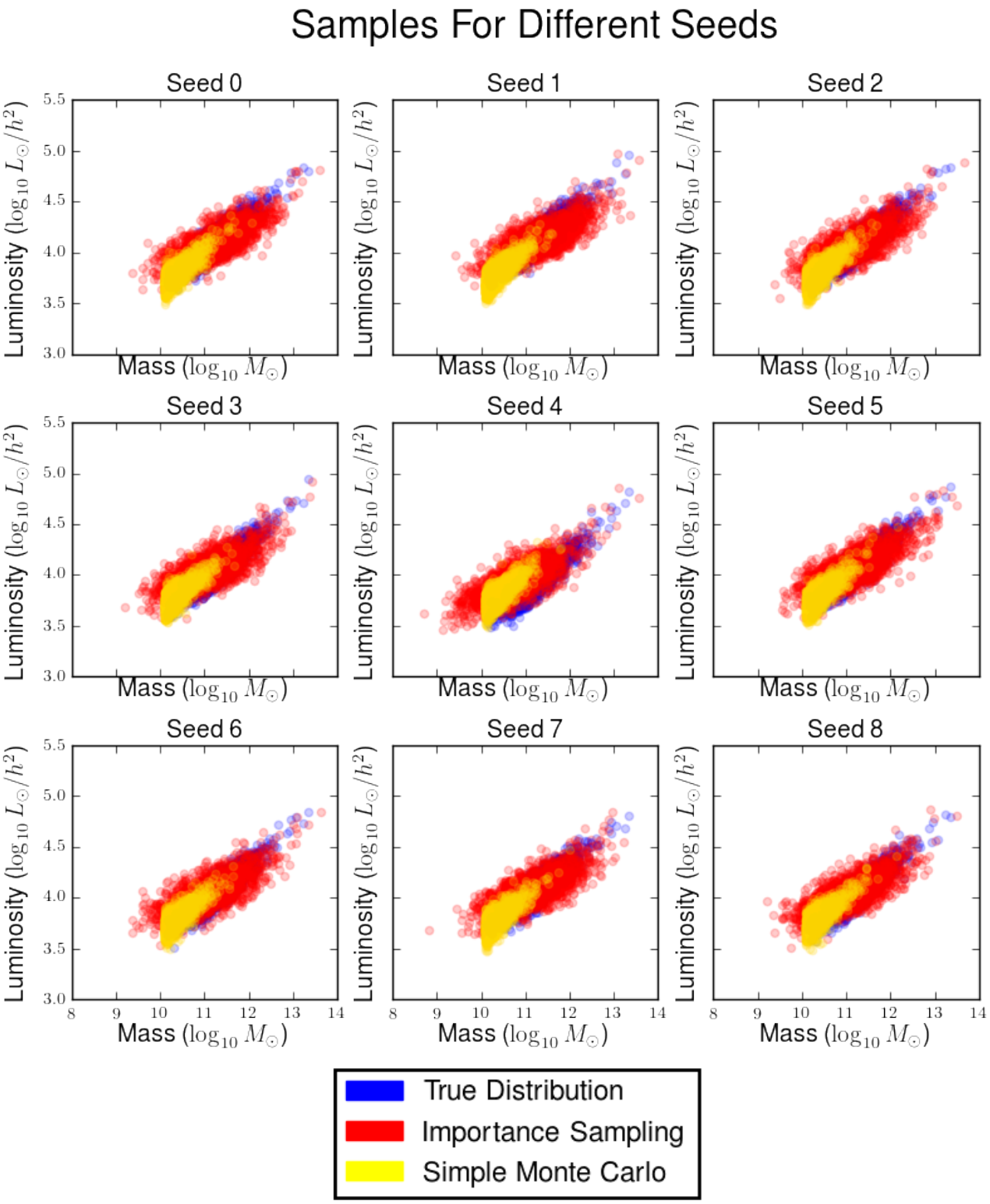

Figure 5. Each seed corresponds to a different draw of hyperparameter values. The true joint distribution from the test dataset is in blue, the importance sampling sampling distribution is in red, and the simple monte carlo sampling distribution is in yellow.

We pass the negative-log-integrand, $-\ln I(M, L)$, to the optimizer which returns the optimal point (M^{opt}, L^{opt}) and the inverse hessian matrix $H^{opt\ -1}$ at the optimum. We then approximate the integrand as

$$I^*(M, L) = I(M^{opt}, L^{opt}) \exp\left(\frac{v^T H^{opt} v}{2}\right)$$
$$v = (M, L) - (M^{opt}, L^{opt})$$

which has the closed form integral

$$\iint dM dL\ I^*(M, L) = \sqrt{\frac{4\pi^2}{\det(H^{opt})}}$$

In practice we must also apply a numerical trick to produce satisfactory inverse hessian matrices from the optimizer. Appendix 5.6 describes this technique in detail.

3. RESULTS

We start by examining the behavior of the four methods in the context of a single halo. Next we scale two of the methods to run over a field of view with multiple halos. Then we analyze the accuracy and show the results from the full inference.

3.1. *Single Halo Likelihood*

The key difference between the four likelihood integration methods described in Section 2 is how they marginalize over the masses and luminosities of the halos. To see these differences more clearly we narrow the scope of the likelihood function to a single halo evaluated on a fixed set of hyperparameters. Each single likelihood of a halo is dependent on the halo to which it corresponds. To get a sense of how the single halo likelihoods vary across the halos in our dataset, we create a test suite of four halos (Halo 0, Halo 1, Halo 2, Halo 3). The test halos are a combination of boring representative halos and outlier halos which we anticipate will further expose key differences. Halo 0 comes from a moderately probable part of the joint distribution; Halo 1 is an outlier with low mass and luminosity, Halo 2 is close to the peak of the distribution; Halo 3 is an outlier with high mass and luminosity. Figure 6 shows the joint mass and luminosity distribution of the entire dataset and the test halos we have chosen.

For each test halo we compute the likelihood by four methods: numerical integration, simple monte carlo, importance sampling, and laplace approximation. The result and a breakdown of the techniques is shown in Figure 7 and its companion Table 1.

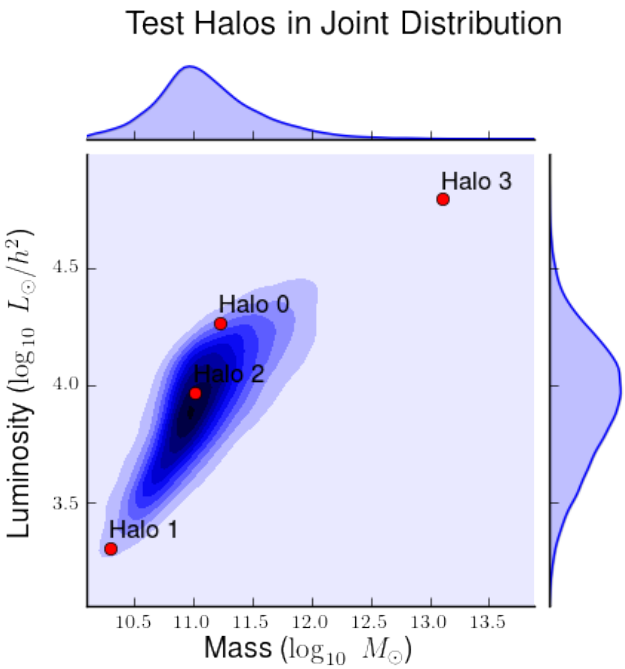

Figure 6. The mass and luminosity joint distribution from the test dataset is shown in violet and the four chosen test halos are shown in red.

The column of numerical integration plots in Figure 7 shows the technique attains solid coverage of the domain and suggests that this method be the most accurate. Since the mass prior drops precipitously outside the displayed mass range, it is safe to assume that even for the numerical integration for Halo 1 which appears cutoff in the figure, the technique still contains the contributing region. We use numerical integration as an answer key that we from which we judge the accuracy of other methods.

Simple monte carlo generates a narrow, diagonal slice of samples, which is consistent with the conditional luminosity relation. Halo 3 shows how terribly suited this approach is for outlier halo likelihoods. Even for Halos 1 and 2 where it produces some high weight samples, it misses out on capturing the true oval shape of the high weight region. While this approach has attractive computational aspects, its inefficient sampling, fully exposed by outliers, renders it completely unsatisfactory for our purposes.

Importance sampling is an attractive remedy to the problems that plague simple monte carlo. Almost all the samples are high weight and the oval nature of the high weight region is captured. Even for the outlier Halo 3, this technique performs very well. It is also encouraging that the likelihood values produces from this technique

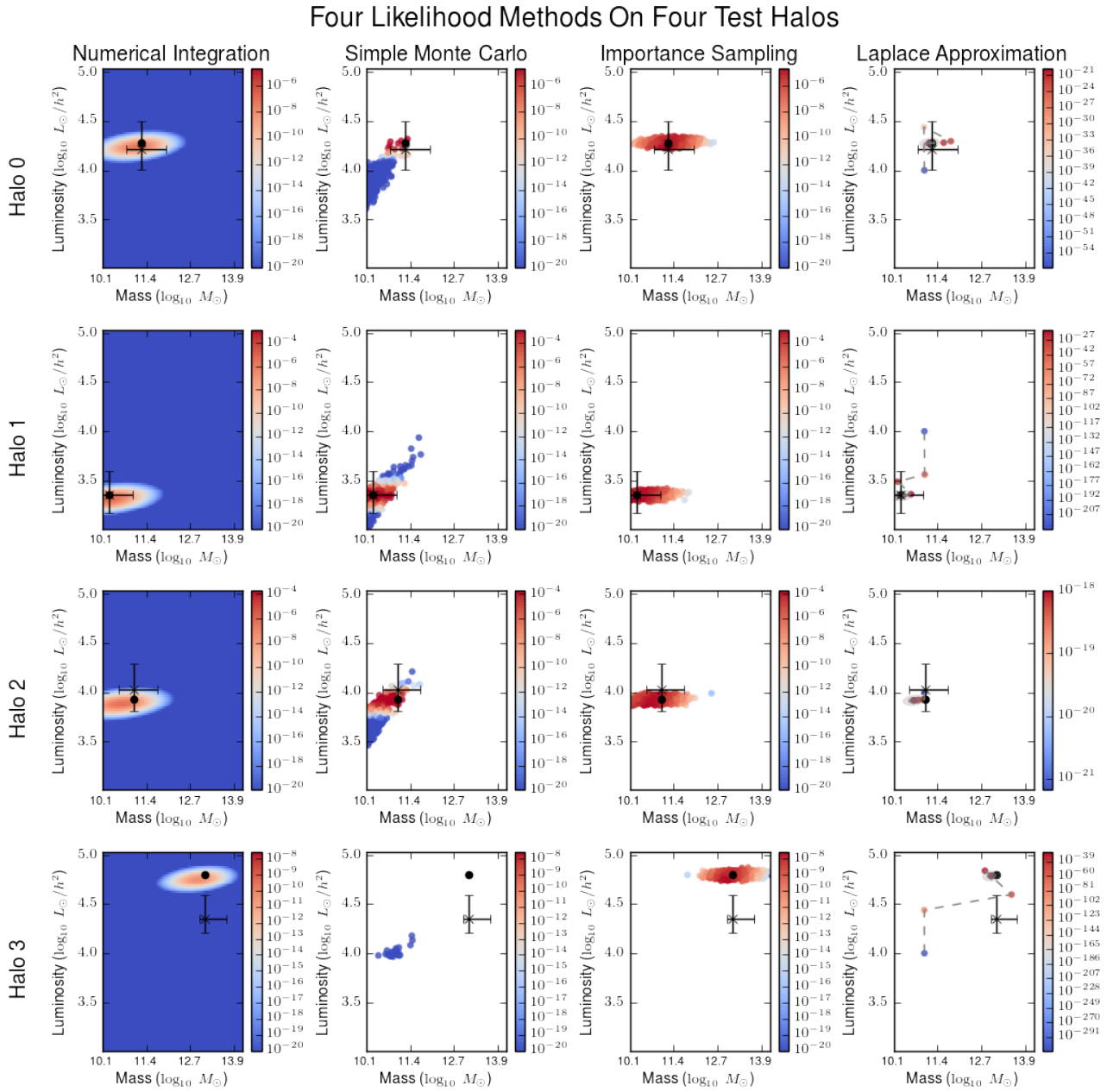

Figure 7. An array of plots characterizing how the different methods marginalize the mass and luminosity joint distribution for the different test halos. Each row corresponds to one of the different test halos. Each columns corresponds to one of the different methods. All plots have the same axes and the plots in the first three columns have the same color weight scale. Each of the first three plots is a scatter plot of the samples taken by the corresponding method colored according to their contribution to the likelihood. Red samples contribute heavily; blue samples contribute lightly. The black dot is the true mass and luminosity of the test halo and the black bars are the 2σ error bars for the joint distribution. The fourth column contains a scatter plot of the points generated in the iterative optimization method, with sequential points connected by a gray dashed line. The resulting multivariate normal distribution's 1σ and 2σ are in light gray. Table 1 shows the likelihood values corresponding to these plots.

Test Halo	Halo 0
Numerical Integration	6.40358058377 $\times 10^{-06}$
Simple Monte Carlo	8.57179497702 $\times 10^{-07}$
Importance Sampling	6.39959705909e $\times 10^{-06}$
Laplace Approximation	3.68701955823e $\times 10^{-06}$
Test Halo	Halo 1
Numerical Integration	0.000487808661776
Simple Monte Carlo	0.000487547191603
Importance Sampling	0.000487206370881
Laplace Approximation	0.000316487709918
Test Halo	Halo 2
Numerical Integration	6.27613065943e-05
Simple Monte Carlo	2.82394491619e-05
Importance Sampling	6.29604927168e-05
Laplace Approximation	3.80374065466e-05
Test Halo	Halo 3
Numerical Integration	5.57536196939e-09
Simple Monte Carlo	3.49945536295e-53
Importance Sampling	5.57880154767e-09
Laplace Approximation	2.89024212204e-09

Table 1. Companion table to Figure 7. Contains the single halo likelihood value for each of the test halos (rows) under the different methods (columns).

are in great agreement with the values from numerical integration.

Both simple monte carlo and importance sampling are sampling techniques; we can use effective sample size to compare their sampling efficiencies (further described in Appendix 5.4). The effective sample size is roughly the number of samples which are contributing to the estimate in a nontrivial manner. Figure 2 shows the effective sample sizes for the four points for both sampling methods. The exceptionally low values for n_{eff}^{SMC} show that simple monte carlo is unreliable, especially on the outlier Halo 3. Importance sampling however, is both efficient, $n_{eff}^{IS} \approx n/2$, and consistent, n_{eff}^{IS} is similar for different points, even on the outliers. The extreme difference between n_{eff}^{SMC} and n_{eff}^{IS} reinforces that importance sampling is a dramatic improvement over simple monte carlo and suggests this improvement is necessary to sample the likelihood effectively.

The laplace approximation is a different approach entirely. We see that the optimization finds the optimal point quickly, getting to within 0.1 distance within a handful of iterations. But the hessian and the multivariate normal distribution it produces are a bit small. This visual intuition from Figure 7 is confirmed by the consistent laplace approximation underestimation seen in Table 1. The laplace approximation solutions are smaller than the solutions from numerical integration and importance sampling by a factor hovering around two. The estimate may be sensitive to the form of the approximating distribution. The sensitivity may be explored in future work.

Test Halo	n_{eff}^{SMC}	n_{eff}^{IS}
Halo 0	3	624
Halo 1	191	562
Halo 2	45	619
Halo 3	1	518

Table 2. For each of the four test halos we compare the effective sample size of simple monte carlo n_{eff}^{SMC} to the effective sample size of importance sampling n_{eff}^{IS}, both with 1000 samples.

The goal of pursuing alternative likelihood computation methods is to achieve an accurate answer with lower computational costs. The cost is related to sampling efficiency, but there are other factors that contribute. Figure 8 shows the likelihood convergence and runtime for Halo 0 versus the number of samples used in the method for our Python implementation of these methods (Appendix 5.1). Since the laplace approximation is not a sampling technique, and does not have number of samples as a parameter, it is approximately constant in both plots.

The convergence plot in Figure 8 confirms our suspicions from Figure 7. Simple monte carlo is widely inaccurate and differs by numerical integration by over an order of magnitude. Even with many samples this method exhibits instability. The laplace approximation is an improvement, but underpredicts by about a factor of two. If this underprediction is consistent across posterior samples then the technique may be salvageable. Importance sampling and numerical integration are more promising. Both converge to stable values rapidly and are in remarkable agreement. These patterns remain when we switch the likelihood to different halos.

The runtime plot in Figure 8 shows that numerical integration, importance sampling and simple monte carlo have constant runtimes initially and transition to linear runtime. Our software profiling confirms that this is due to the fixed memory allocation cost dominating for small sample sizes and cpu processing dominating for larger sample sizes. The cpu time dominates sooner in numerical integration because of the large grid of com-

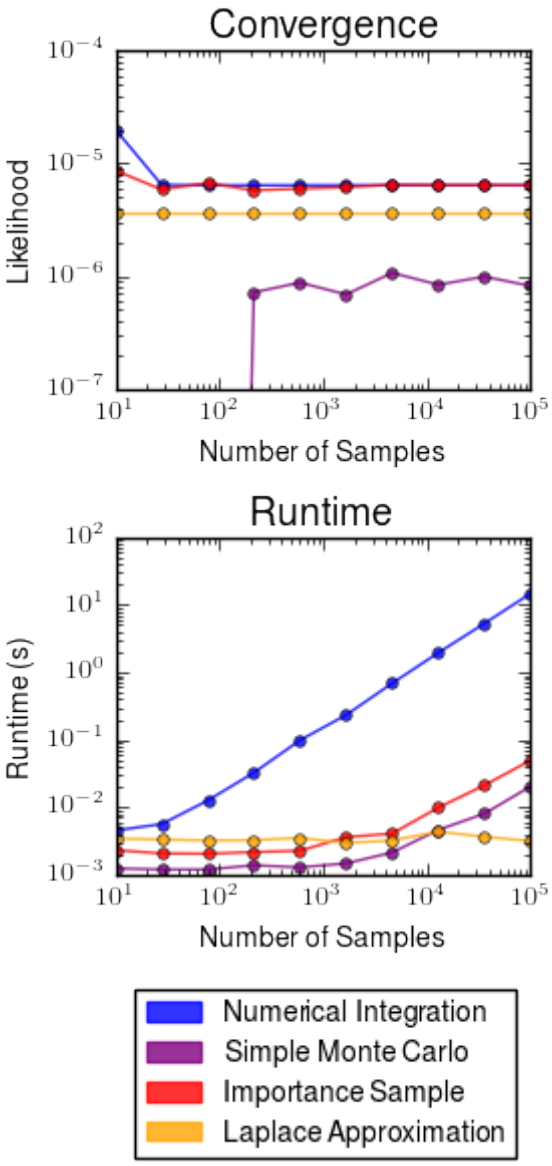

Figure 8. Top panel: the likelihood value for Halo 0 for varying numbers of samples. Bottom panel: the Python implementation runtime of the likelihood computation for varying numbers of samples.

putations it makes. It is also worth recalling that the number of mass values used in numerical integration is constant while the number of luminosity samples is a parameter. This is why the runtime grows linearly with the number of samples as opposed to quadratically. Numerical integration takes $\sim 10^3$ times longer than simple monte carlo. Fortunately, importance sampling only takes ~ 2 times longer and the laplace approximation has constant runtime in theory.

Given the strict grid sampling of numerical integration, we expect it to be the most accurate likelihood method. While we cannot compare this approximation method to a closed form solution for the problem, we can test its self consistency. As the number of samples increases, the sampling resolution improves, which should improve the accuracy in turn. Table 3 shows the numerical integration values for the four test halos at varying numbers of samples. The likelihood value gains two digits of precision for each order of magnitude increase in the number of samples from 10^2 to 10^5. Numerical integration is self consistent and achieves significant precision with a small number of samples. We use it to measure the accuracy of the other methods.

Samples	Halo 0
100	$\boxed{6.40}23729274 \times 10^{-06}$
1000	$\boxed{6.4035}688384 \times 10^{-06}$
10000	$\boxed{6.403580}5837 \times 10^{-06}$
100000	$\boxed{6.40358070}10 \times 10^{-06}$
Samples	**Halo 1**
100	$\boxed{4.87}71666550 \times 10^{-04}$
1000	$\boxed{4.8780}776704 \times 10^{-04}$
10000	$\boxed{4.878086}6177 \times 10^{-04}$
100000	$\boxed{4.87808670}70 \times 10^{-04}$
Samples	**Halo 2**
100	$\boxed{6.27}49470451 \times 10^{-05}$
1000	$\boxed{6.2761}191479 \times 10^{-05}$
10000	$\boxed{6.276130}6594 \times 10^{-05}$
100000	$\boxed{6.27613077}43 \times 10^{-05}$
Samples	**Halo 3**
100	$\boxed{5.57}43105124 \times 10^{-09}$
1000	$\boxed{5.5753}517431 \times 10^{-09}$
10000	$\boxed{5.575361}9693 \times 10^{-09}$
100000	$\boxed{5.57536207}14 \times 10^{-09}$

Table 3. Numerical integration values for different numbers of samples (rows) and test halos (columns). The boxed digits are rougly the precision.

To measure the accuracy of importance sampling we first get the numerical sampling value and then run importance sampling many times with different random seeds to generate a distribution of results. We repeat this for varying numbers of samples. Figure 9 is the result of this procedure. The relative error is relatively small and decreases with increasing numbers of samples. The 2σ band is tight, which means importance sampling is fairly stable. The relative error distributions for the different halos are similar and suggest that

we may be able to model it as an independent identically distributed random variable.

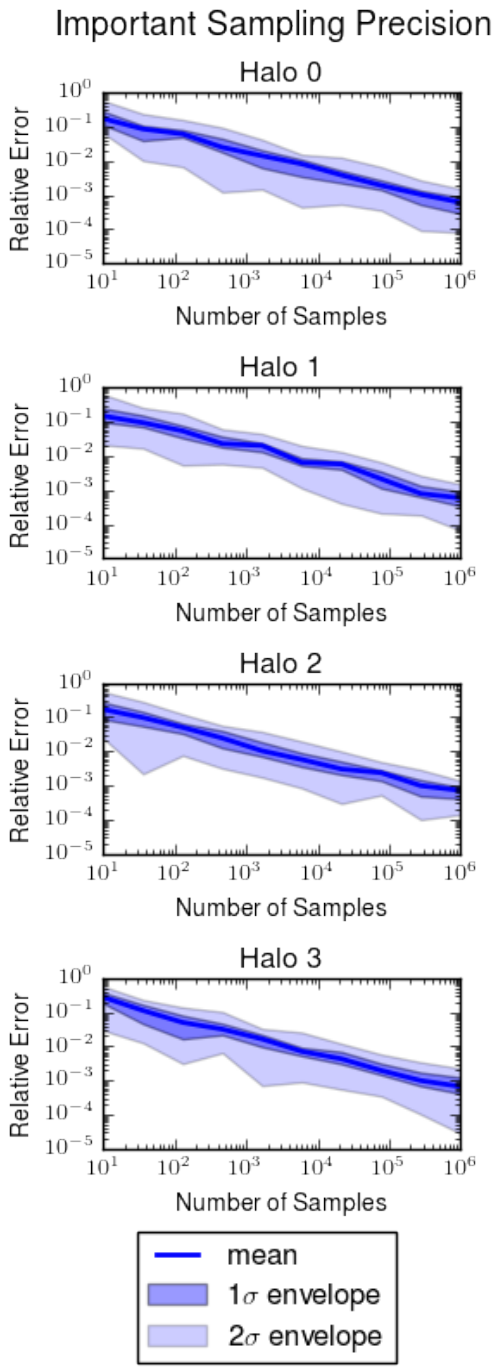

Figure 9. The accuracy of importance sampling against the number of mass and luminosity samples it consumes. The distribution for the 1σ and 2σ envelopes is generated from the random samples in different importance sampling runs. We use the numerical integration result with 100 samples as the true value for determining accuracy.

So far we have been examining the likelihood for a single halo at a fixed set of hyperparameters. We have found that importance sampling closely matches numerical integration and is reasonably stable. Next we examine how these methods perform on different likelihood samples for Halo 0. For each of the four varying hyperparameters, we sample the likelihood along the parameter while holding the others fixed (see Figure 10). This provides a rough sense of the shape of the four dimensional likelihood surface. The likelihoods from importance sampling are almost indistinguishable from likelihoods from numerical integration. The laplace approximation likelihood surfaces have the same general shape but are too jagged and erratic to be employed for inference.

This section has exposed key inadequacies of using simple monte carlo or laplace approximation to compute the likelihood. We move forward with only numerical integration and importance sampling as viable methods for the full likelihood.

3.2. *Multiple Halo Likelihood*

In this section we examine critical question: can numerical integration or importance sampling scale to our target of almost 116,000 halos? First we measure runtimes of high performance implementations, then we model errors, and ultimately we highlight our best computation of the posterior.

If the computation scales linearly with the number of halos, then the runtime gets multiplied by a factor of almost 116,000 - making performance critical as we transition from the single halo context to the multiple halo context. We have pursued three different implementations of our methods to attain the best performance.

The first implementation is a Python implementation with a few enhancements. It heavily relies on the NumPy package and follows the suggested practices in Van Der Walt et al. 2011: vectorizing calculations, avoiding copying data in memory, and minimizing operation counts. All objects that can be precomputed or reused are. It also employs a custom implementation of a lognormal evaluation because we have found *scipy.lognorm.pdf* to be an order of magnitude slower, even when called on a scipy.lognorm instance many times (its not just the constructor of the object) (Jones, Oliphant, Peterson, et al. 2001–). This code has been heavily profiled and we believe offers the best performance we can get from single process Python.

The second implementation is a Python implementation that uses the Numba package to take advantage of its just-in-time (jit) compilation (num 2015). In order to fully take advantage of the jit it is best to have as

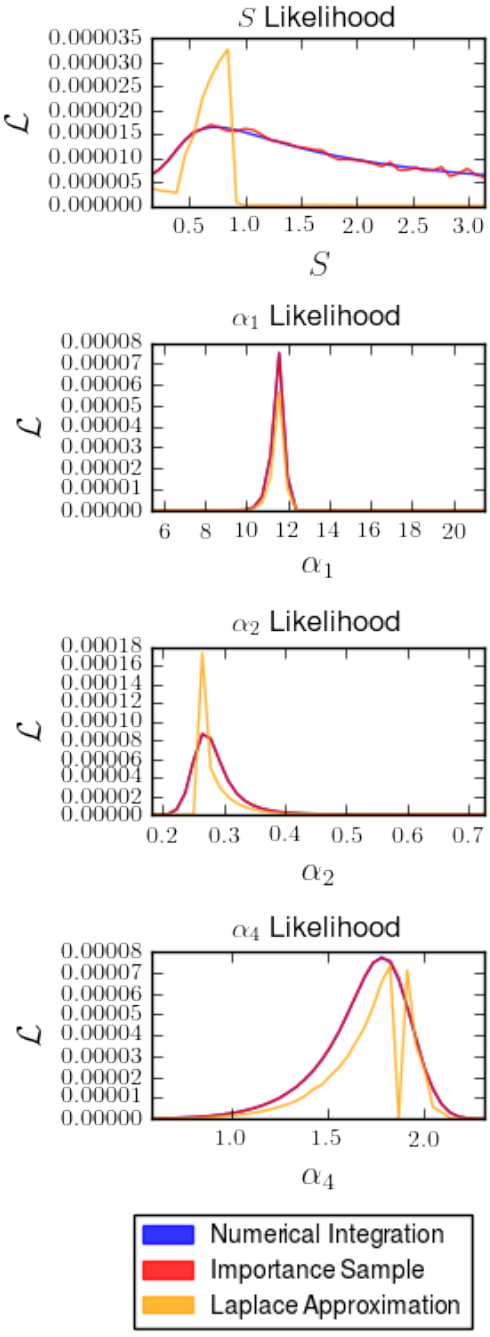

Figure 10. The single integral likelihood surfaces over lines in hyperparameters $S, \alpha_1, \alpha_2, \alpha_4$.

much of the code as possible as standard Python source code as opposed to invoking external packages. NumPy methods in particular, cannot be tracked and optimized by this method because many of them call native code. We have attempted to exploit the faster *nopython* compilation mode by rewriting much of the original implementation in standard python loops and math operations but there are still a few inconvertible NumPy objects that pollute the jit and prevent us from realizing a speedup. In our experiments, the jit overhead outweighs any improvements to performance.

The third implementation is in C++. We are able to re-use memory more effectively and realize a solid performance improvement. Figure 11 shows how the runtimes of these implementations scale with respect to the number of samples and the number of halos. As expected, the C++ implementations are the fastest.

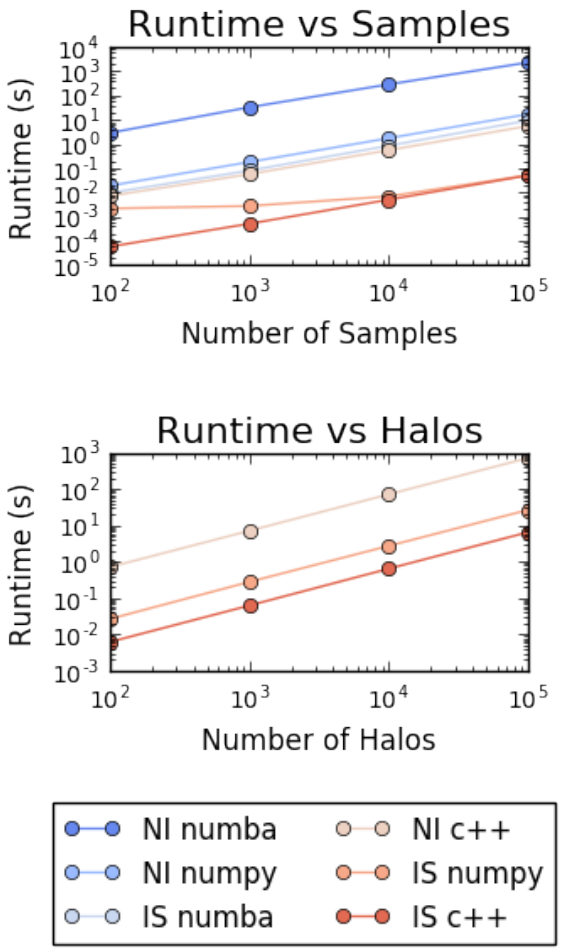

Figure 11. Runtimes of the three implementations with respect to the number of samples for one halo (top panel) and the number of halos used with 100 samples (bottom panel).

We can use Figure 11 to extrapolate the runtimes for computing the likelihood on full dataset with almost 116,000 halos. Importance sampling with 100 samples would take approximately 10 seconds; numerical integration with 100 samples would take approximately 5 minutes.

Presumably there is some error in each of the individual halo likelihoods in the importance sampling approach. Since the total likelihood is a product of

many individual halo likelihoods, error may accumulate rapidly. In order to model and constrain the error in this process we collect computationally expensive error statistics on importance sampling's performance on 2% of the total number of halos, then extrapolate these statistics to predict the error on the total number of halos.

We use a simple model to make error predictions. Let the set of halos be H. Then we assume that the log-likelihood corresponding to each halo has an intrinsic true value v_i and a random variable ϵ_i representing the error. The log-likehood over the entire dataset is then

$$\ell = \sum_{i \in H} (v_i + \epsilon_i)$$

Now let H' be the subset of 2,000 randomly drawn halos which we will use in our estimate. For each of these halos we extract three parameters: the 'true' numerical integration likelihood n_i, the standard deviation of the importance sampling likelihoods σ_i, and the mean difference between the numerical integration and the importance sampling likelihoods b_i. Given the precision and stability of numerical integration, we use it as the 'true' value of the likelihood, which means $v_i = n_i$. Under this assumption b_i represents the bias in an individual halo likelihood. We also model the individual likelihood errors as normal distributions. Let D(x) be the distribution corresponding to the histogram of samples from H'. For example, $D(n_i)$ is the distribution corresponding to $\{n_i\}_{i \in H'}$. Then we can estimate the log-likelihood over the entire dataset as

$$\begin{aligned}
\hat{\ell} &= \sum_{i \in H} (n'_i + \epsilon'_i) \\
\epsilon'_i &\sim \mathcal{N}(b'_i, \sigma_i^{2\prime}) \\
n'_i &\sim D(n_i) \\
b'_i &\sim D(b_i) \\
\sigma'_i &\sim D(\sigma_i)
\end{aligned}$$

Figure 12 shows the predicted and empirical full likelihood distributions. The dispersions of the predicted and empirical likelihood distributions are in great agreement. The standard deviations of the three predicted distributions are 15, 5.1, and 1.7 and the standard deviations of the empirical distributions are 13, 4.8, and 2.3.

There is a discrepancy between the means of the distributions when plotted on this scale. However, the relative discrepancy is about 0.02%. In the empirical distributions we see the likelihood slightly increase with the number of samples whereas in the predicted distributions we see it decrease. This is because the true single likelihood noise is not a normal distribution like we assume. Overall, we believe the simple error model does an effective job of representing the full likelihood behavior of importance sampling.

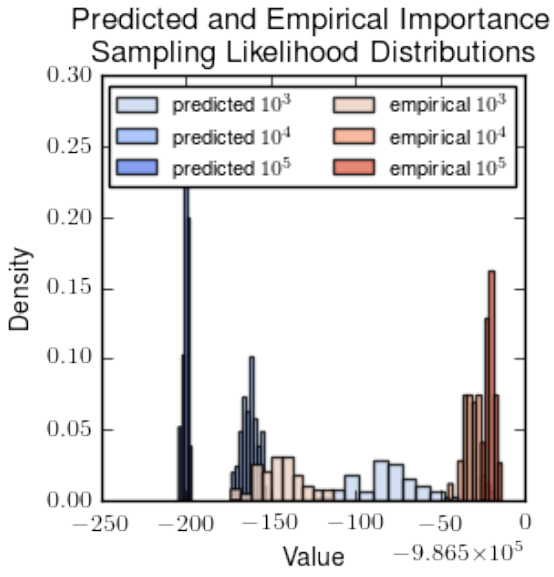

Figure 12. The predicted and empirical distributions for the full likelihood with $10^3, 10^4, 10^5$ samples used for each individual halo.

Todo: Discuss bias, show full likelihood results when computation finishes.

4. CONCLUSIONS

In this paper we explore four methods for solving a large heirarchical inference: numerical integration, simple monte carlo, importance sampling, and laplace approximation. The inference likelihood is a product of likelihoods for each halo in the field of view. We analyze how our methods perform on an individual halo before scaling numerical integration and importance sampling to the full likelihood. Our conclusions are:

- Simple monte carlo is not able to sample outliers effectively.
- Laplace approximation consistently underpredicts the likelihood and produces unstable posterior weights.
- Importance sampling improves on simple monte carlo by sampling outliers effectively. However, as the number of halos increases, the method accumulates error. Even with 10^5 samples per each

halo likelihood the posterior is dominated by a single sample.

- Numerical integration is more computationally expensive than the other methods, but should produce an accurate posterior when 10^4 samples per halo likelihood are used. We estimate 250,000 core-hours of compute should be enough to produce a reasonable posterior.

ACKNOWLEDGMENTS

We thank Phil Marshall for excellent mentorship and Risa Wechsler for proposing this project.

Author contributions are listed below.

David Thomas: initiated project, led development work.

Phil Marshall: advised on statistics.

Risa Wechsler: advised on galaxy formation and cosmology.

REFERENCES

2015, Numba: A LLVM-based Python JIT Compiler, LLVM '15 (New York, NY, USA: ACM), 7:1–7:6

Dark Energy Survey Collaboration, Abbott, T., Abdalla, F. B., et al. 2016, MNRAS, 460, 1270

Hilbert, S., Hartlap, J., White, S. D. M., & Schneider, P. 2009, A&A, 499, 31

Jones, E., Oliphant, T., Peterson, P., et al. 2001–, SciPy: Open source scientific tools for Python, [Online; accessed 2017/03/25]

Kong, A. 1992, A Note on Importance Sampling using Standardized Weights

LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, ArXiv e-prints, arXiv:0912.0201

Marshall, P. 2006, MNRAS, 372, 1289

Murray, S. G., Power, C., & Robotham, A. S. G. 2013, Astronomy and Computing, 3, 23

Nowozin, S. 2015, Effective Sample Size in Importance Sampling, http://www.nowozin.net/sebastian/blog/effective-sample-size-in-importance-sampling.html, online; accessed 4 July 2017

Owen, A. B. 2013, Monte Carlo theory, methods and examples

Reddick, R. 2014, PhD thesis

Springel, V., Frenk, C. S., & White, S. D. M. 2006, Nature, 440, 1137

Springel, V., White, S. D. M., Jenkins, A., et al. 2005, Nature, 435, 629

Tinker, J., Kravtsov, A. V., Klypin, A., et al. 2008, ApJ, 688, 709

Van Der Walt, S., Colbert, C., & Varoquaux, G. 2011, The NumPy Array: A Structure for Efficient Numerical Computation, doi:10.1109/2011.37

Yang, X., Mo, H. J., van den Bosch, F. C., et al. 2005, MNRAS, 362, 711

5. APPENDIX

5.1. *Code*

Todo. List where code can be found etc.

5.2. *Epsilon Mass Function Background*

Throughout the inference we use logarithms to improve numerical stability. The *log* function is not defined at zero which creates issues when sample halos are outside the bounds of the mass prior. A simple solution is adding a small background probability so that the function is always nonzero. There are two further modifications that must be made to make this work. First, the prior density must be recomputed so that it integrates to one. Forgetting to do this leads to nontrivial accumulated errors in the importance sampling. The second issue is when we sample from the prior we would not like many samples outside the original priors' support. By making the background probability, which we call ϵ, very small we can ensure that almost all samples are from the mass range of the original prior. We set $\epsilon = 10^{-30}$ over the mass range $[10^0, 100 * \text{MaxMass}]$ and renormalize the probability density accordingly. In expectation, every $\sim 10^{30-18} = 10^{12}$ samples we draw lie outside the original support. This small number will have negligible impact. Figure 13 shows the updated mass function prior.

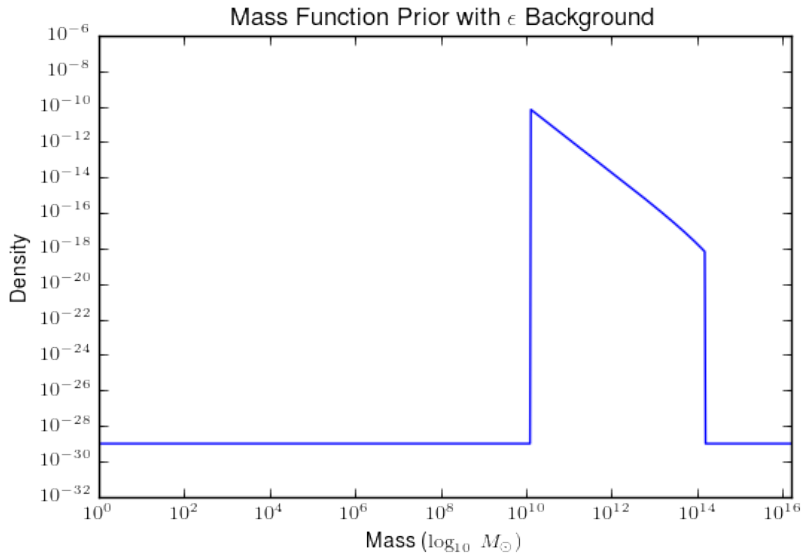

Figure 13. The mass function prior for z = 0 over the full range.

5.3. *Importance Sampling Is An Unbiased Estimator*

We prove that in expectation, the mean attained from importance sampling is the true mean of the distribution. Consider $\mu = \int_D dx\ f(x)p(x)$ where D is the domain of integration, f is the integrand, and p is the probability density function on D. Then the chosen bias density function q, which is nonzero over D, can be used so that

$$\mu = E_p[f(x)] = \int_D dx\ f(x)p(x) = \int_D dx\ \frac{f(x)p(x)}{q(x)}q(x) = E_q\left[\frac{f(x)p(x)}{q(x)}\right]$$

This shows that the expected mean from importance sampling is the true mean of the distribution, as desired. For readers who would like to learn more about this technique, we recommend perusing Chapter 9 in Owen 2013.

5.4. *Effective Sample Size Comparison*

The effective sample size n_{eff} of an estimator is roughly the number of samples which make nontrivial contribution to the estimate. For example if $n_{eff} << n$, then the primary contribution comes from only a few samples. On the other hand, if $n_{eff} \approx n$, then all samples provide a nontrivial contribution to the estimator. An ideal sampling scheme would have this property.

Consider a random vector X with distribution $p(x)$ and a function $h(X)$. Let the quantity of interest be

$$\mu = E_p[h(X)] = \int h(x)p(x)dx$$

Then simple monte carlo gives the unbiased estimate

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} h(X_i);\ X_i \sim p(x)$$

For simple monte carlo samples, the effective sample size is

$$n_{eff}^{SMC} = \frac{\mathrm{Var}[X_1]}{\mathrm{Var}_p[\hat{\mu}]}$$

The effective sample size for importance sampling is derived in Kong 1992, and also clearly explained in Nowozin 2015 and Owen 2013 (Chapter 9.3). In importance sampling, samples are drawn from a strategically designed proposal distribution $q(x)$. Then reweighting the samples with the ratio $p(X_i)/q(X_i)$ gives the standard importance sampling estimate

$$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \frac{p(X_i)}{q(X_i)} h(X_i);\ X_i \sim p(x)$$

When p is known, the so called self-normalized importance sampling estimate can be used. Denoting weights by $w(X_i) = \frac{p(X_i)}{q(X_i)}$ it is defined as

$$\bar{\mu} = \frac{\sum_{i=1}^{n} w(X_i)h(X_i)}{\sum_{i=1}^{n} w(X_i)};\ X_i \sim q(x)$$

Let $W_i = w(X_i)$ and $W = W_1$. Then after defining the estimators above, Kong 1992 derives

$$\frac{\mathrm{Var}_q[\bar{\mu}]}{\mathrm{Var}_p[\hat{\mu}]} \approx 1 + \mathrm{Var}_q[W]$$

where $\mathrm{Var}_q[W] \approx \frac{1}{n-1}\sum_{i=1}^{n}(w_i - \frac{1}{n})^2$, the sample variance of W. The variance of the self-normalized importance sampling estimate is approximately equal to the variance of the simple monte carlo estimate times $1 + \mathrm{Var}_q[W]$. Therefore, when taking n samples to compute $\bar{\mu}$, the effective sample size is

$$n_{eff}^{IS} = \frac{n}{1 + \mathrm{Var}_q[W]}$$

Now that we have the n_{eff} formulas for both sampling techniques, we can use effective sample size to compare their sampling efficiencies.

5.5. *Converting Conditional Luminosity Scatter To Conditional Mass Scatter With Error Propagation*

In Section 2.4 we describe the biased distribution $Q(M|L, \alpha, S, z)$. Here we derive a conversion factor λ such that $S_Q = \lambda \cdot S$ in the biased distribution. We use gaussian error propagation in log-space to approximate the log-variance in mass (S_Q) from the log-variance in luminosity (S).

$$S_Q^2 = \left(\frac{\partial \ln \mu_M}{\partial \ln L}\right)^2 \cdot S^2$$
$$S_Q = \left|\frac{\partial \ln \mu_M}{\partial \ln L}\right| \cdot S$$

Plugging in the mean mass formula above and differentiating with respect to L yields

$$\frac{\partial \ln \mu_M}{\partial \ln L} = 1/\alpha_2$$

It is in our best interest to choose a conservative biased distribution so that if the center of the bias distribution is a little off the 'important' region we can still get a few samples in the high weight region. In practice this means choosing a larger scatter. Therefore, we multiply the log-variance in luminosity by an extra factor of 2.

$$S_Q = \frac{2S}{\alpha_2}$$

We make a distribution of S_Q, shown in Figure 14, by sampling from the prior for S and plugging it into the formula above. The result is similar to the value we arrived at from visual inspection of joint distributions. In our code we hardcode the λ factor to be the mean of this distribution 5.6578015811698101.

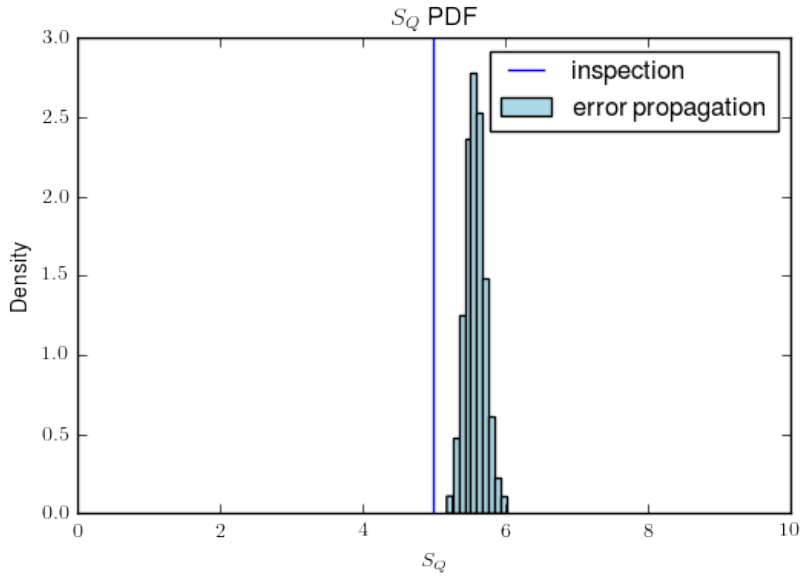

Figure 14. The distribution of conversion factors from error propagation and the conversion factor from visual inspection of the data.

5.6. *Re-scaling The Laplace Approximation*

When implementing the laplace approximation as described in Subsection 2.5, we run into numerical issues. The large difference in scale between masses and luminosities, differing by as much as 10^{10} at times, wipes out the precision in the off-diagonal terms of the Hessian matrix. This manifests as having zeroes for the off-diagonal elements of the Hessian. To avoid these issues we run the optimizer over the log mass and log luminosity. In this section we describe the mathematics that makes this possible.
Let f represent the log integrand,

$$f(M, L) = \ln P(L^{obs}|L, \sigma_L^{obs}) + \ln P(L|M, \alpha, S, z) + \ln P(M|z)$$

Then we use the BFGS algorithm to attain

$$\ln M_{opt}, \ln L^{opt}, H_{ln}^{opt\ -1} = \text{argmin}_{\ln M, \ln L} - [f(M, L)]$$

where we optimize over logarithmic mass and luminosity to get an accurate hessian. Then by the chain rule we have

$$-\frac{\partial^2 f}{\partial M \partial L} = -\frac{\partial^2 f}{\partial \ln M \partial \ln L}\frac{\partial \ln M}{\partial M}\frac{\partial \ln L}{\partial L} = -\frac{1}{ML}\frac{\partial^2 f}{\partial \ln M \partial \ln L}$$

After extending this to the other combinations of mass and luminosity we have

$$H^{opt} = (H_{ln}^{opt\ -1})^{-1} \odot \begin{pmatrix} M^{opt-2} & M^{opt-1}L^{opt-1} \\ M^{opt-1}L^{opt-1} & L^{opt-2} \end{pmatrix}$$

where $\odot$ is elementwise multiplication. Then we can approximate the likelihood as

$$\mathcal{L}(L^{obs}|\alpha, S, \sigma_L^{obs}, z) = \exp(f(M^{opt}, L^{opt}))\sqrt{\frac{(2\pi)^2}{\det(H^{opt})}}$$

Now the optimizer provides a Hessian with nonzero off-diagonal entries which produces a much better approximation.