PHYS 788 Report

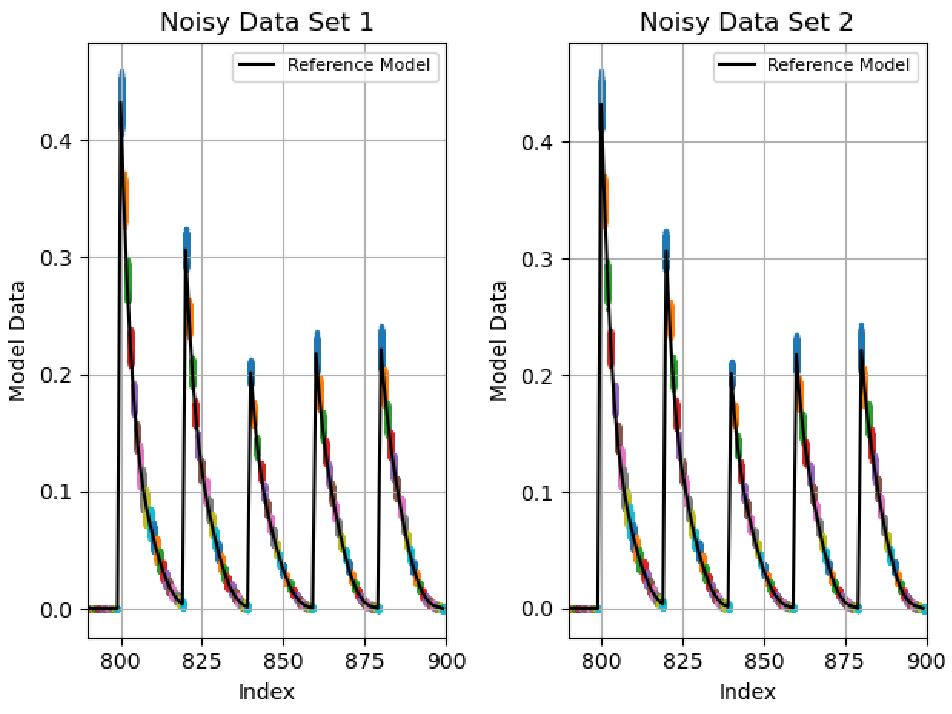
Martine Campbell

22/04/2024

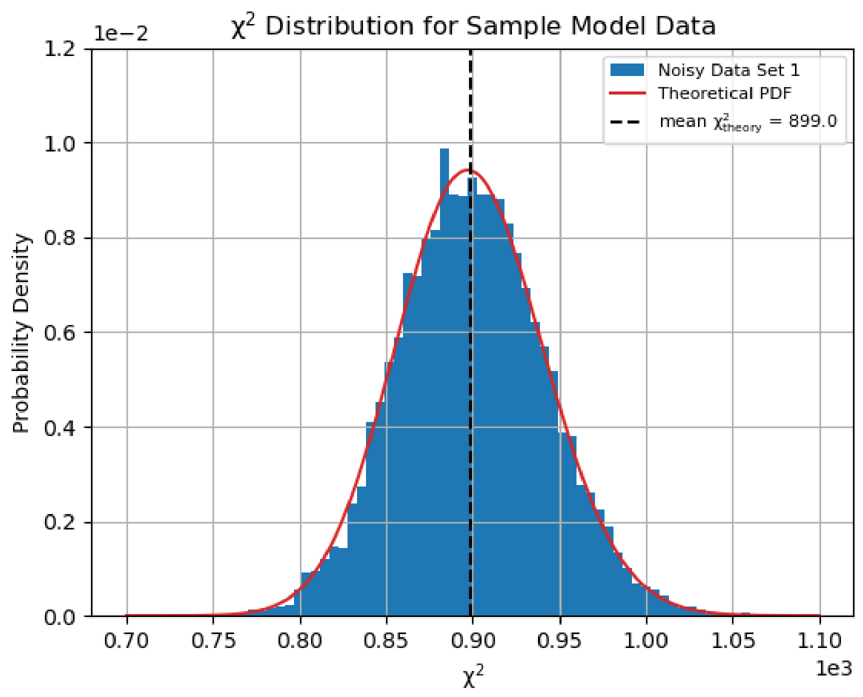
1 Assignments Overview

1.1 Assignment 1

This assignment focused on the numerical stability of the covariance matrix, chi-squared distributions, and Hartlap factor correction. To start with, I made two sets of noisy, Gaussian data vectors using a reference model and an analytical covariance matrix. Then, I made sure the noisy data sets scattered around the truth (reference model) by plotting the following:

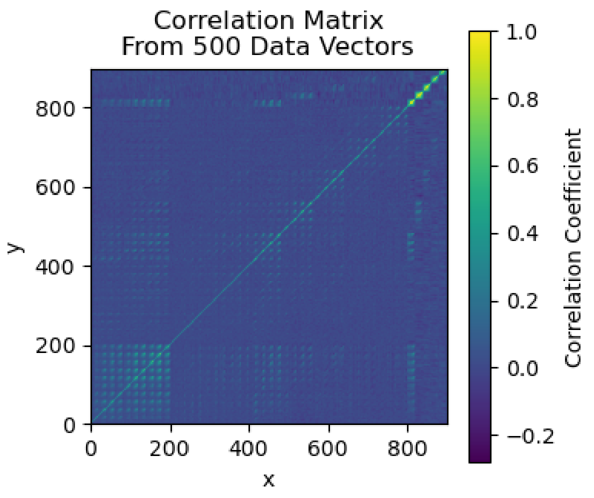


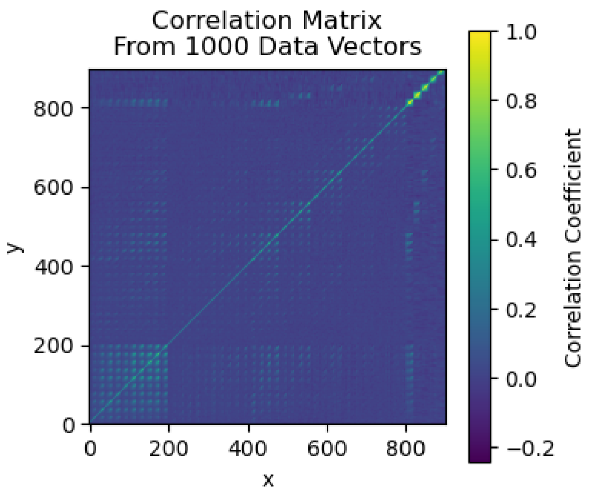
After, I tested whether Set 1 follows a chi-squared distribution by computing the chi-squared value along each row and making a histogram of all the chi-squared values. The theoretical chi-squared distribution was plotted on top of the histogram, resulting in the following:

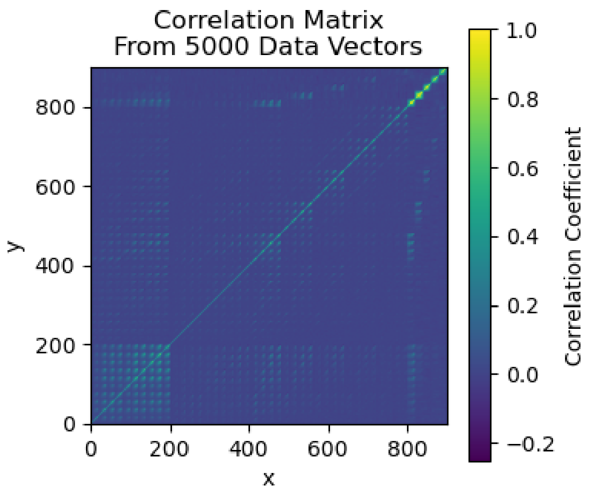


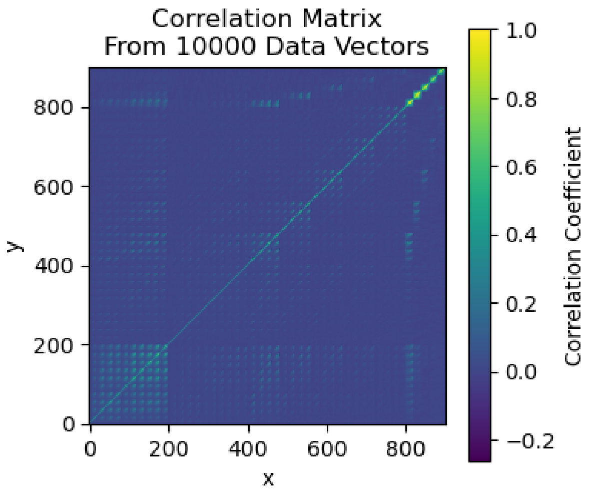
The experimental mean was 899.8911562144137 and the experimental variance was 1799.83793040223. On the other hand, the theoretical mean was 899 and the theoretical variance was 1798. Since the experimental and theoretical means and variances were very close, it was concluded that Set 1 follows a chi-squared distribution. Not to mention, the theoretical probability density function traced out the histogram. Additionally, since the two noisy data sets were created in the same way, it could be said that Set 2 also follows a chi-squared distribution.

In the next part of the assignment, I made numerical covariance matrices from different numbers of data vectors (500, 1000, 5000, and 10,000) from Set 2. I then computed and plotted the corresponding correlation matrices:



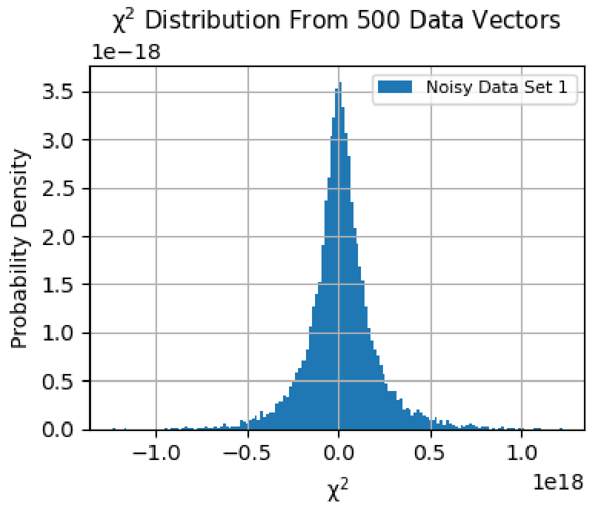


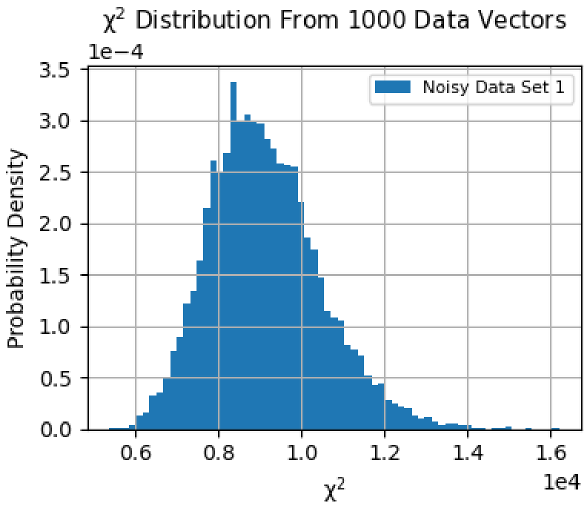


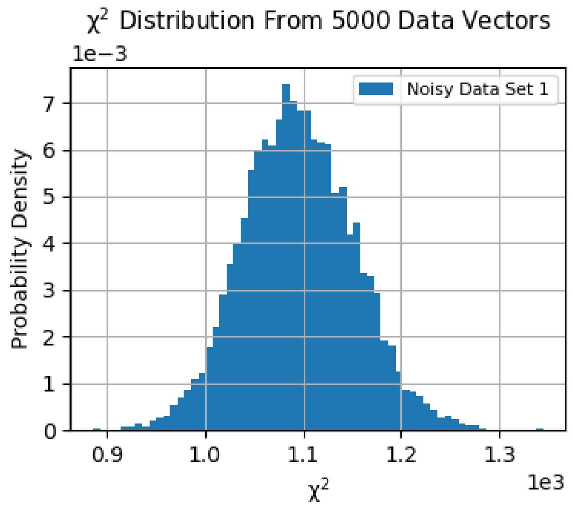


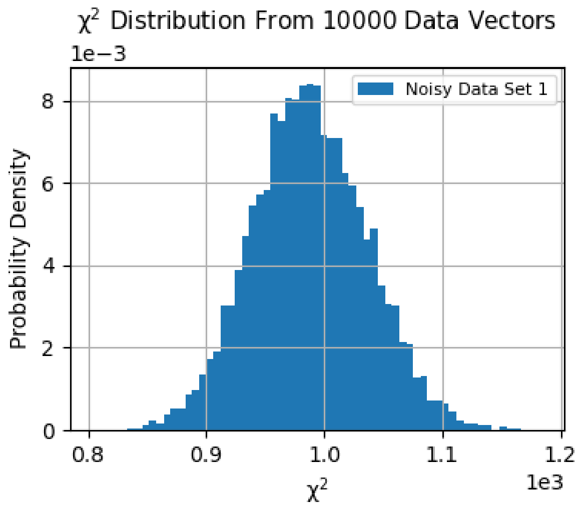
As can be seen from the plots above, the correlation matrices are symmetric, and their diagonal elements have a value of 1. Both of these results are expected. After making these plots, I wanted to verify whether the covariance matrices are positive semi-definite. Note: a positive semi-definite matrix has non-negative eigenvalues. Negative eigenvalues were found in the covariance matrix from 500 data vectors, making it the only covariance matrix that is not positive semi-definite.

For the next task, I made histograms of chi-squared values using the numerical covariance matrices:





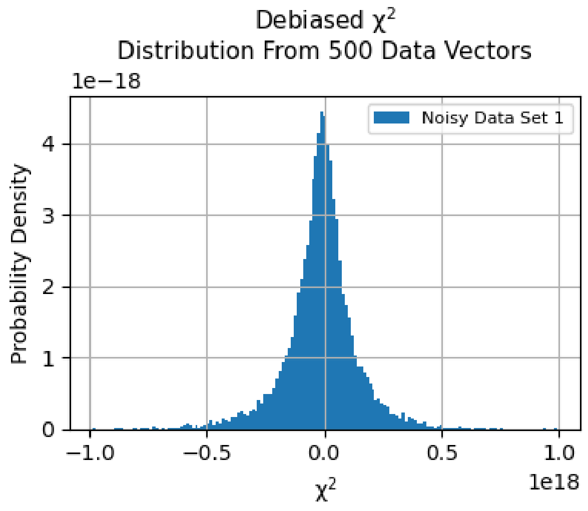


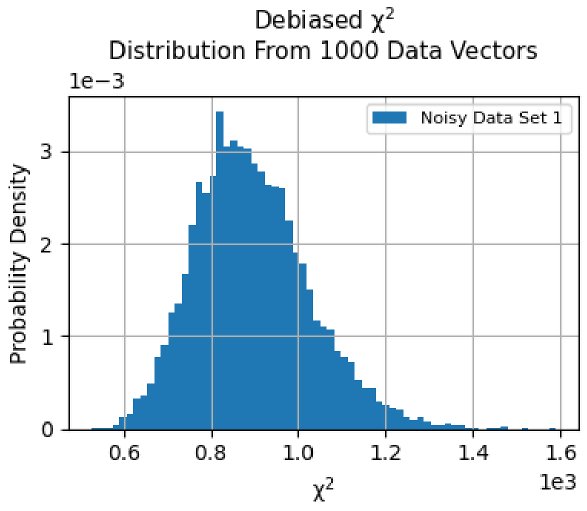


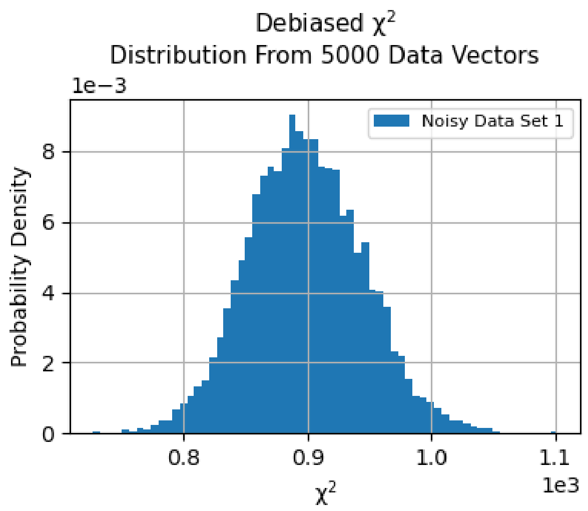
The means and variances of these distributions were quite off compared to the theoretical values previously mentioned. That being said, one can easily see that the histograms get closer to the theoretical chi-squared distribution as the number of data vectors increases. After, this task was repeated with debiased covariance matrices. To debias a covariance matrix, one can compute the *Hartlap* factor:

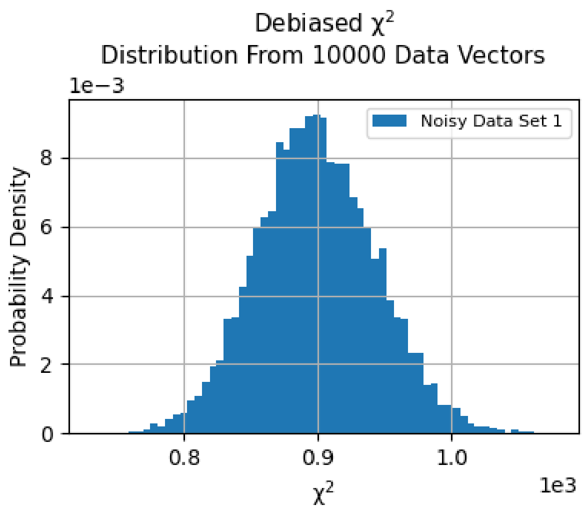
$$h = \frac{n - 1}{n - m - 2}$$

Where n is the number of realizations used to compute the covariance matrix and m is the rank of the covariance matrix. The debiased covariance matrix would be the numerical covariance matrix multiplied by the Hartlap factor. Below are the histograms from this task:





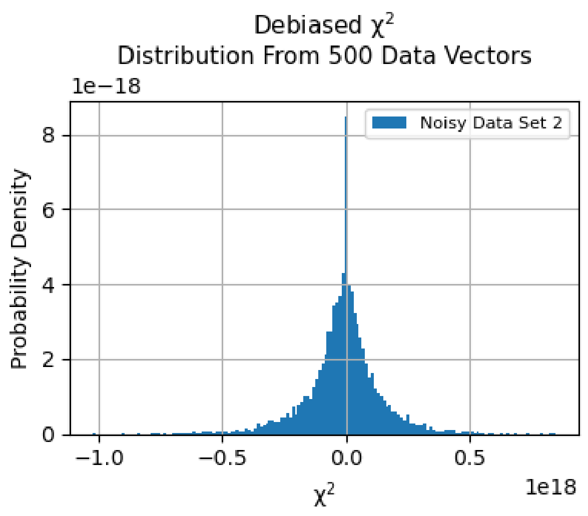


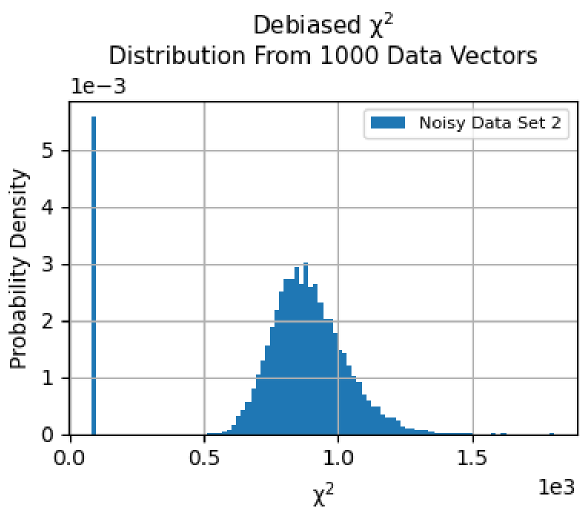


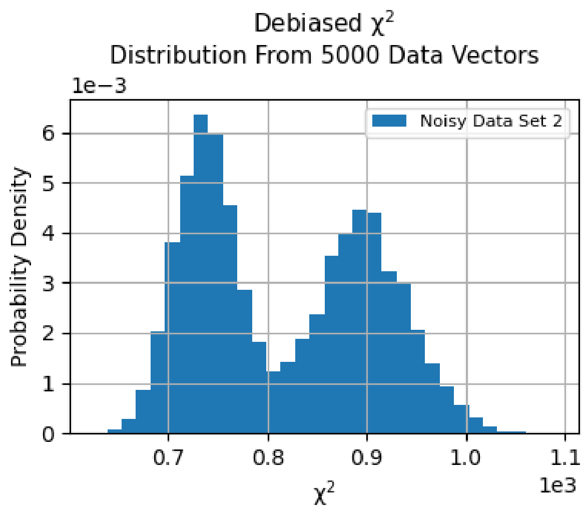
The distributions above have means that are very close to the theoretical mean, although the variances are still off (too large). So, correcting with the Hartlap factor can shift our chi-squared distributions to the true mean, but it cannot account for the noise in the covariance matrix. To reduce the variance/get it closer to the theoretical value, one needs to use more data vectors when creating the numerical covariance matrix.

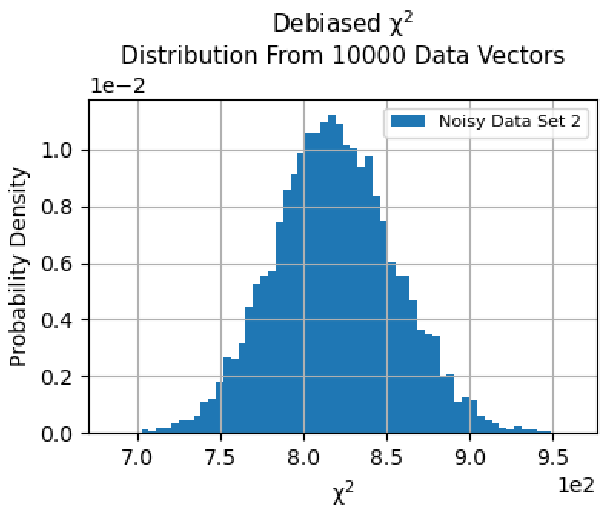
Note that the distributions from 500 data vectors differ significantly from other distributions. The reason is that this covariance matrix had negative eigenvalues, meaning it was not invertible. However, to compute a chi-squared value, one needs to take the inverse of a covariance matrix. To ensure a covariance matrix is invertible, the number of realizations must be greater than or equal to the rank of the matrix. In this case, the numerical covariance matrices should have been made from at least 900 data vectors.

Lastly, the previous task was repeated using Set 2 for computing chi-squared values, resulting in the following plots:









Looking at the plots above, it is clear that a chi-squared distribution cannot be recovered if the same data set is used to 1) make a numerical covariance matrix and 2) measure chi-squared values. Recall: the numerical covariance matrices were made from Set 2.

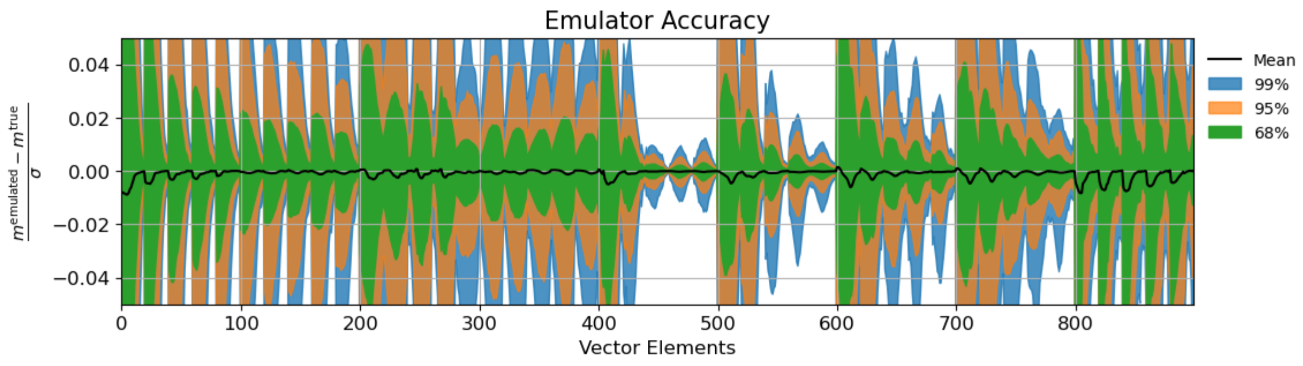
1.2 Assignment 2

This assignment focused on neural-network emulators and data compression. First, I trained a neural-network emulator using *CosmoPower*. I reserved 70% of my data samples for training, and the other 30% for testing. The hyperparameters I ended up using were as follows:

- Number of Nodes for Each Hidden Layer: [512, 512, 512, 512]
- Validation Split: 0.1
- Learning Rates: [1e-2, 1e-3, 1e-4, 1e-5, 1e-6]
- Batch Sizes: [500, 500, 500, 500, 500]

- Patience Values: [100, 100, 100, 100, 100]
- Max Epochs: [1000, 1000, 1000, 1000, 1000]
- Gradient Accumulation Steps: [1, 1, 1, 1, 1]

Additionally, the training features were normalized for better performance. Initially, these hyperparameters were varied until the emulator reached a desired level of accuracy while not taking too long to train. Generally speaking, the accuracy was improved by increasing the number of layers/nodes per layer, using more learning steps, decreasing the learning rate by an order of magnitude for each successive learning step, and using smaller batch sizes. To check the emulator's accuracy, the following plot was made after each training session:



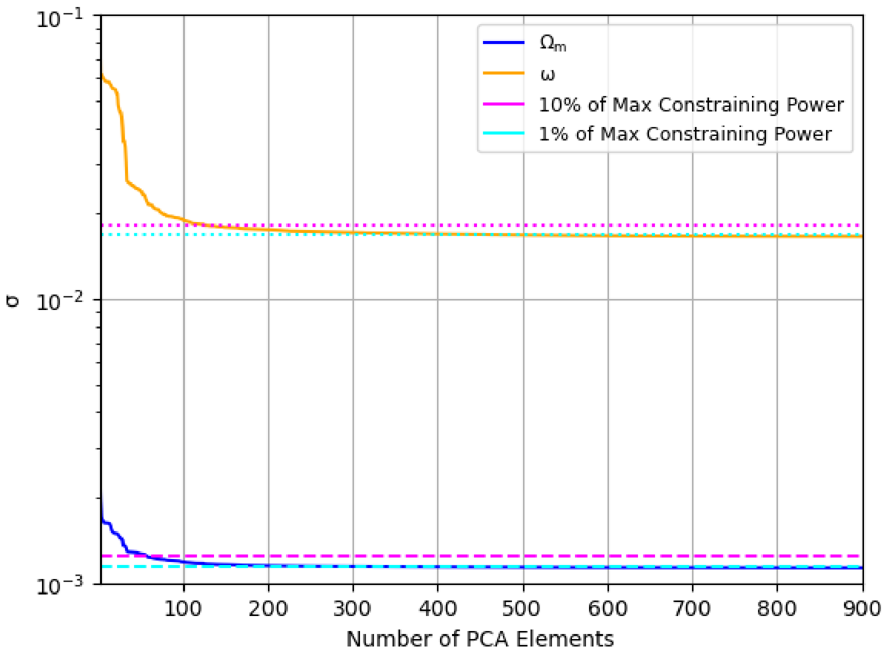
The values on the y-axis are essentially a numerically stable relative error between the emulated features and test features. Since the relative errors spanned a narrow range of small values, the emulator was deemed accurate enough for this assignment.

In the next part of the assignment, I computed the standard deviations of Ω_m and ω using a Fisher analysis:

$$F_{ij} = \left(\frac{\partial m(\Theta)}{\partial \Theta_i}\right)^T C^{-1} \left(\frac{\partial m(\Theta)}{\partial \Theta_i}\right) ,$$

$$\frac{\partial m\Theta}{\partial \Theta_i} \approx \frac{-m(\Theta_i + 2\Delta\Theta_i) + 8m(\Theta_i + \Delta\Theta_i) - 8m(\Theta_i - \Delta\Theta_i) + m(\Theta_i - 2\Delta\Theta_i)}{12\Delta\Theta_i}$$

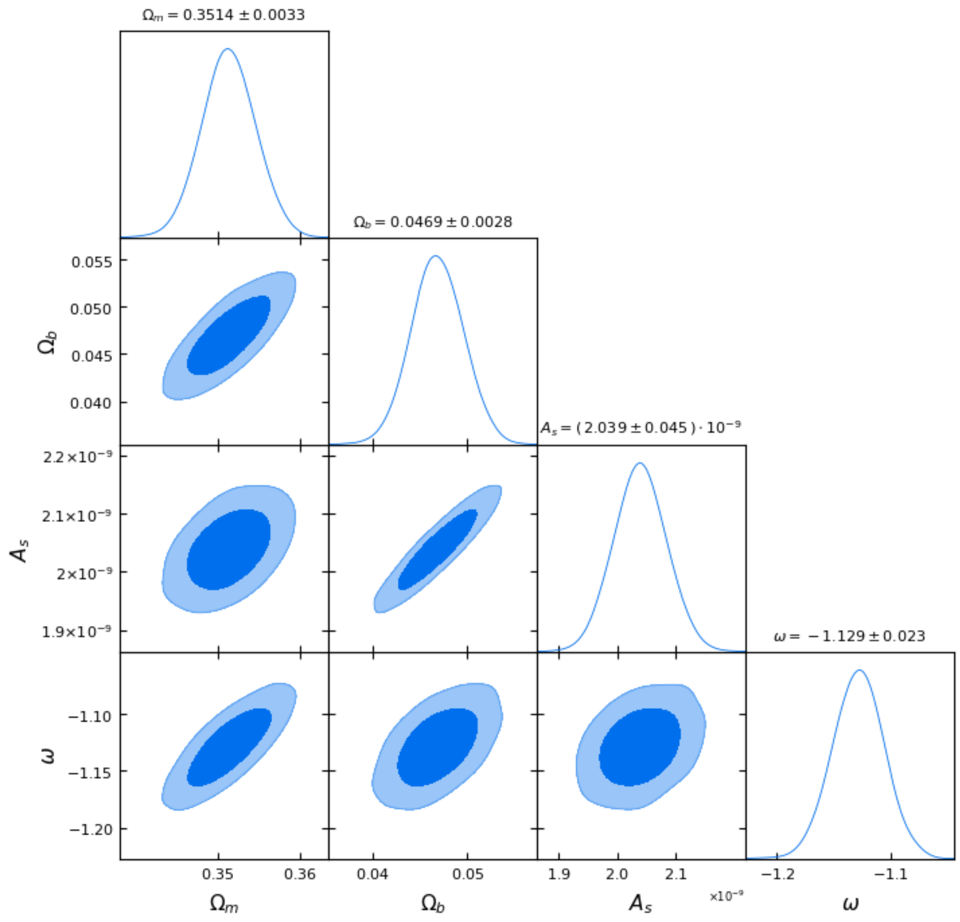
Where C is the analytical covariance matrix and m represents the emulated features. For both parameters, $\Delta\Theta_i$ was 0.01. The parameter covariance matrix was computed by taking the inverse of the Fisher information matrix (F_{ij}), giving standard deviations of 0.00113098 and 0.0165963 for Ω_m and ω, respectively. After, *Principal Component Analysis* (PCA) was used to compress our data for various numbers of PCA elements (1-900). Then, for each N_{pca}, the standard deviations of Ω_m and ω were computed using the same Fisher analysis. The standard deviations from all PCA elements and the original Fisher analysis were essentially the same. This result makes sense, since if all PCA elements are used, then the data is not compressed. Finally, I plotted the standard deviations of each parameter versus N_{pca}. I also plotted (on the same figure) 10% and 1% of the max constraining power for each parameter, which is the standard deviation from all PCA elements:



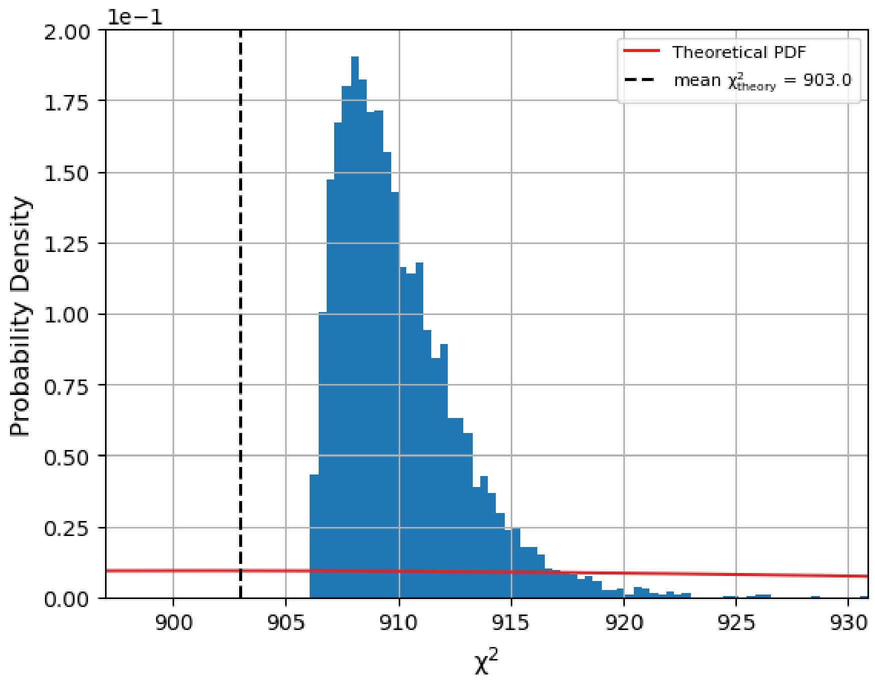
As can be seen from the plot above, you only need a fraction of the available PCA elements to get decent parameter constraints. By around 100 PCA elements, we have attained 10% of the max constraining power for both parameters. Thus, we are able to lose some information while retaining fairly accurate parameter constraints.

1.3 Assignment 3

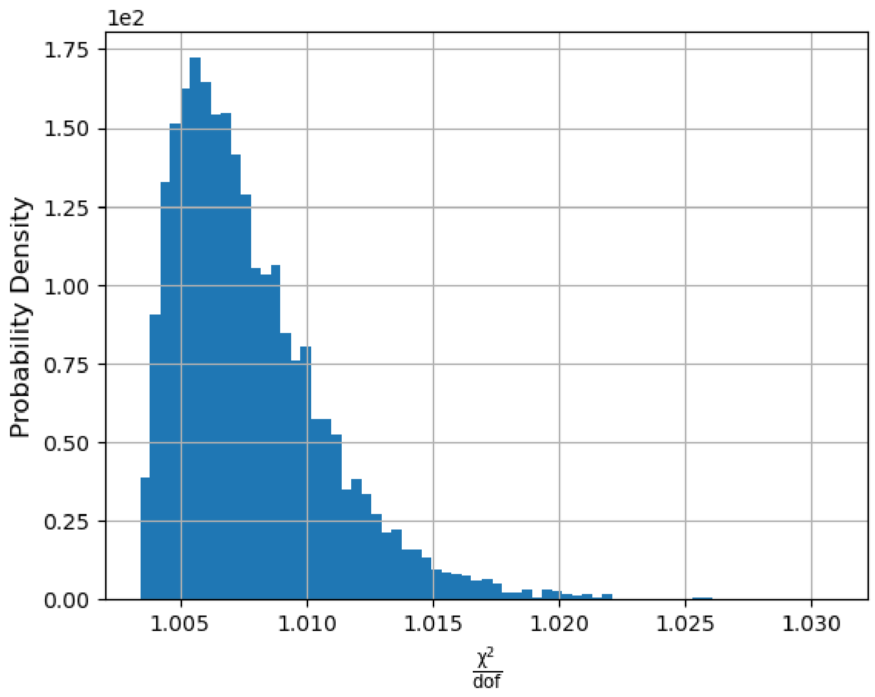
This assignment focused on running an MCMC (Markov Chain Monte Carlo) using the emulator from Assignment 2. First, I ran an MCMC where all four cosmological parameters (Ω_m, Ω_b, A_s, and ω) were varied. Also, I used the noisy reference model and analytical covariance matrix. Across all MCMC runs, I used the *Ensemble Sampler*, 5000 or 5500 steps, and 35 walkers. These numbers of steps and walkers allowed the MCMC chains to complete within 15 minutes, which is relatively quick. Additionally, for each run, the integrated autocorrelation times and mean acceptance fractions were printed to confirm the sampling went well. The autocorrelation time is the number of steps needed for the chain to forget where it started, and the mean acceptance fraction is the fraction of proposed steps that are accepted. For the sampling to go well, the chain's length should be greater than or equal to 50 times the autocorrelation time and the mean acceptance fraction should be around 0.5. Below are the posterior and chi-squared distributions from this run:



$\Omega_m = 0.3514 \pm 0.0033$
$\Omega_b = 0.0469 \pm 0.0028$
$A_s = (2.039 \pm 0.045) \cdot 10^{-9}$
$\omega = -1.129 \pm 0.023$
Ω_m
Ω_b
A_s
ω

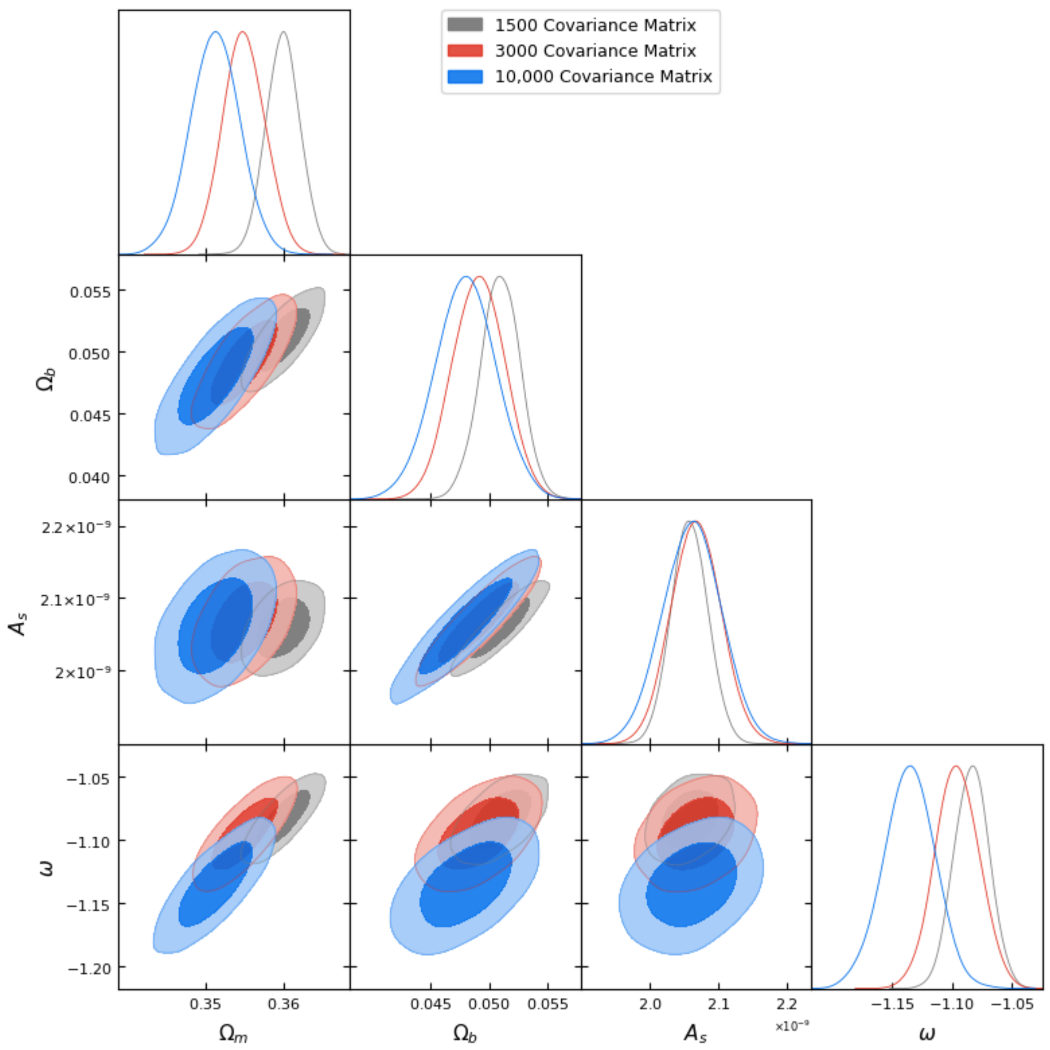


Theoretical PDF
mean χ^2_{theory} = 903.0
Probability Density
χ^2
1e−1

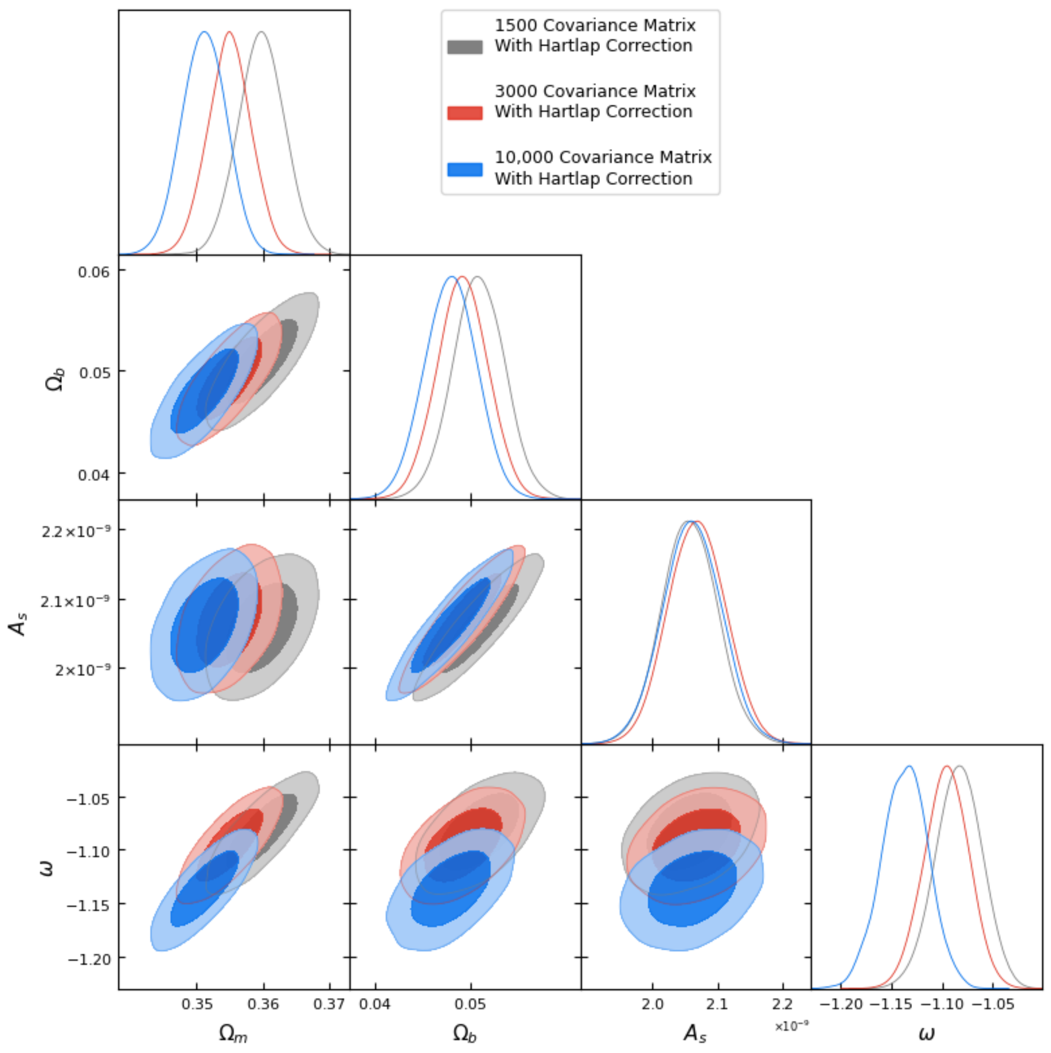


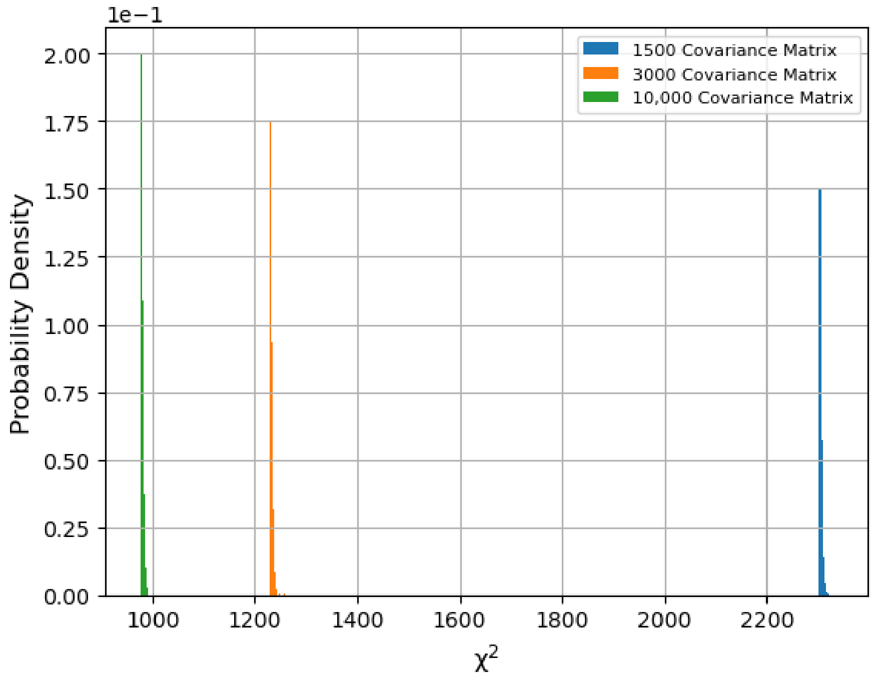
The posteriors are Gaussian, which is good. However, the chi-squared distribution differs a lot from the theoretical chi-squared distribution. That being said, the reduced chi-squared distribution peaks somewhere between 1.005 and 1.01. If the model being fit to data is good, then we expect the reduced chi-squared to be very close to 1. Here, most reduced chi-squared values are quite close to 1, and no values significantly exceed 1.

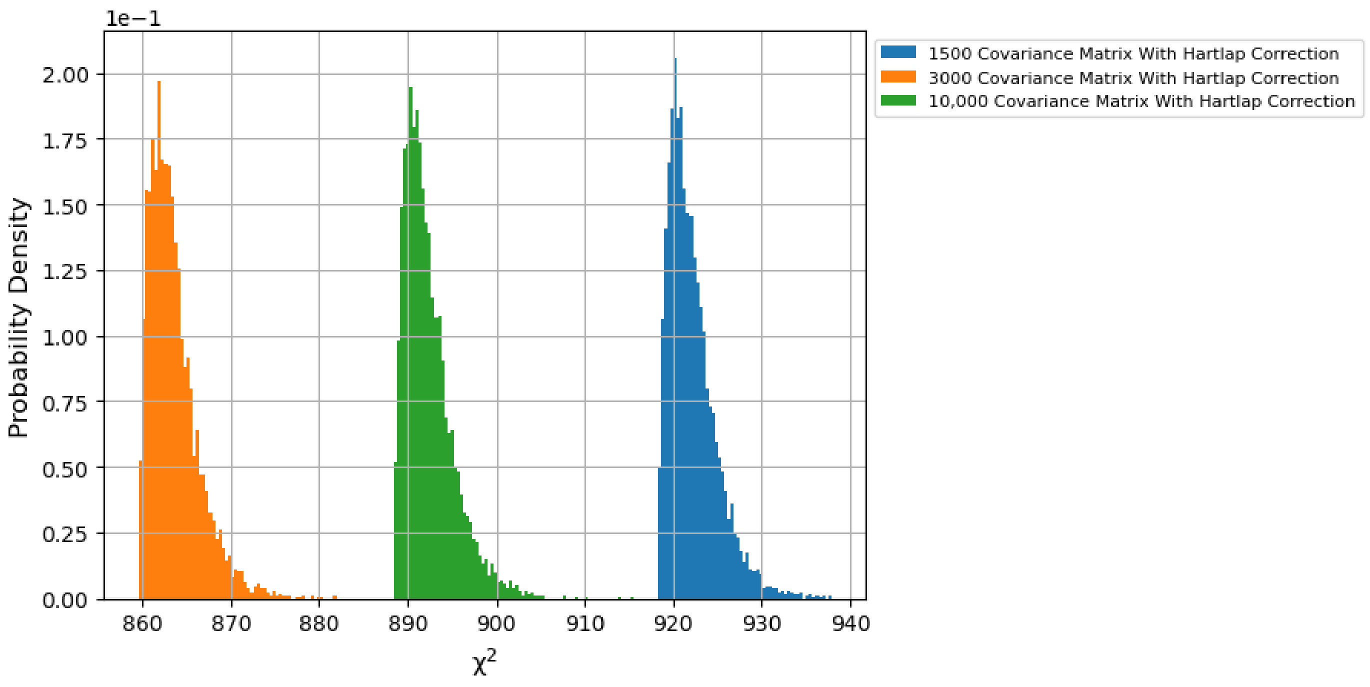
In the next part of the assignment, I ran multiple MCMCs using different numerical covariance matrices and the noisy reference model. Also, half of the runs used Hartlap factor correction, while the other half did not. The posterior and chi-squared distributions were as follows:

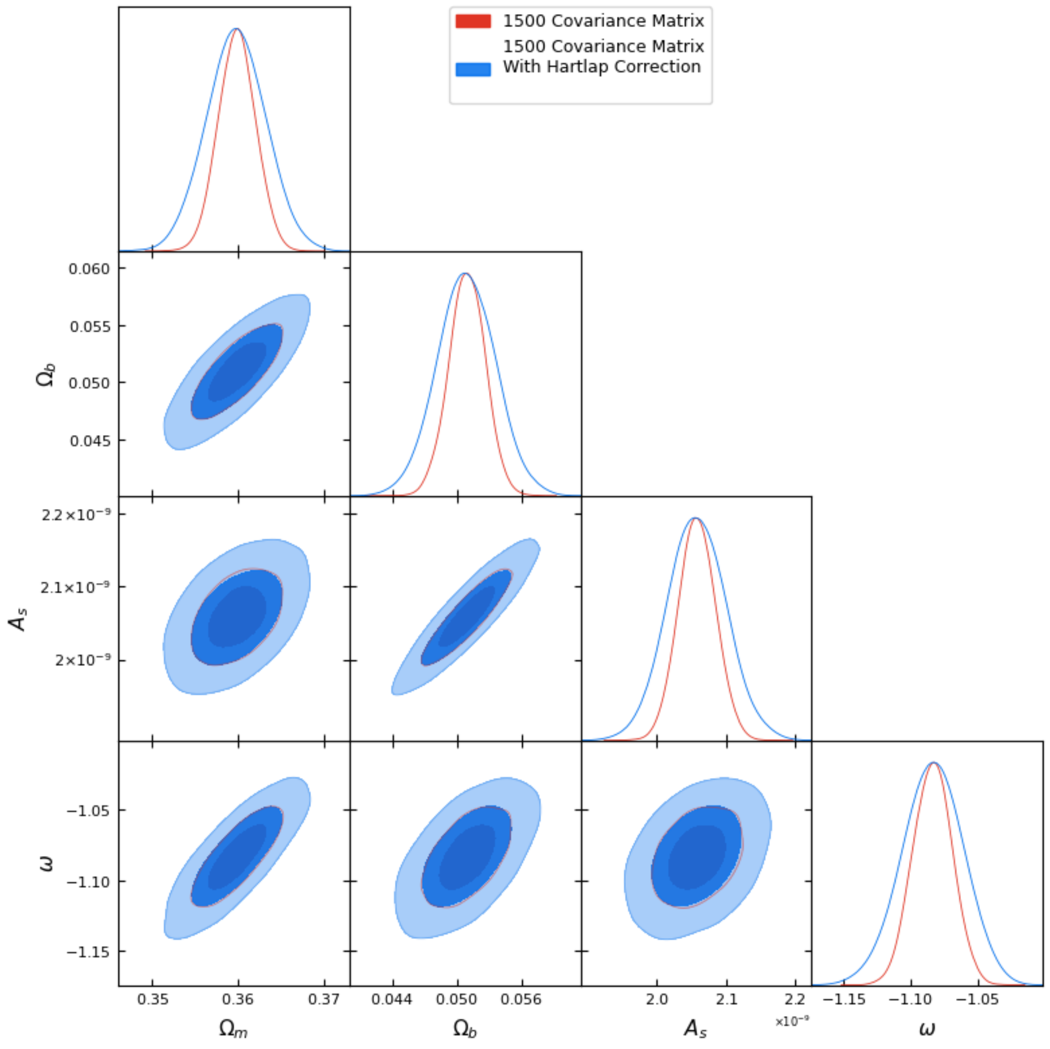


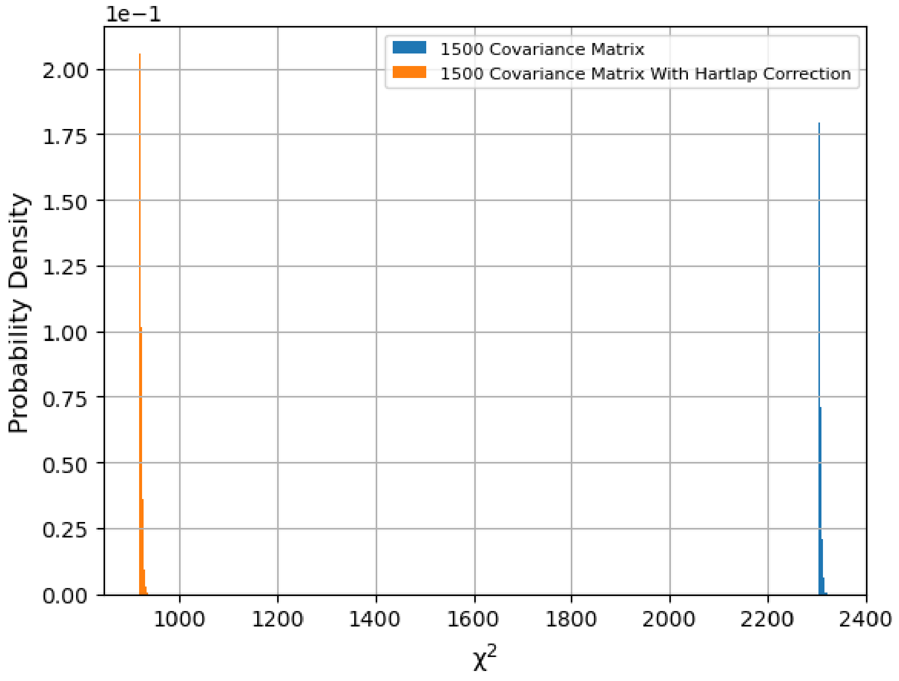
1500 Covariance Matrix
3000 Covariance Matrix
10,000 Covariance Matrix
Ω_b
A_s
ω
Ω_m
Ω_b
A_s
ω
0.055
0.050
0.045
0.040
2.2×10^{-9}
2.1×10^{-9}
2×10^{-9}
−1.05
−1.10
−1.15
−1.20
0.35
0.36
0.045
0.050
0.055
2.0
2.1
2.2
$\times10^{-9}$
−1.15
−1.10
−1.05


1500 Covariance Matrix With Hartlap Correction
3000 Covariance Matrix With Hartlap Correction
10,000 Covariance Matrix With Hartlap Correction
Ω_b
A_s
ω
Ω_m
Ω_b
A_s
ω


1e−1
1500 Covariance Matrix
3000 Covariance Matrix
10,000 Covariance Matrix
Probability Density
χ^2

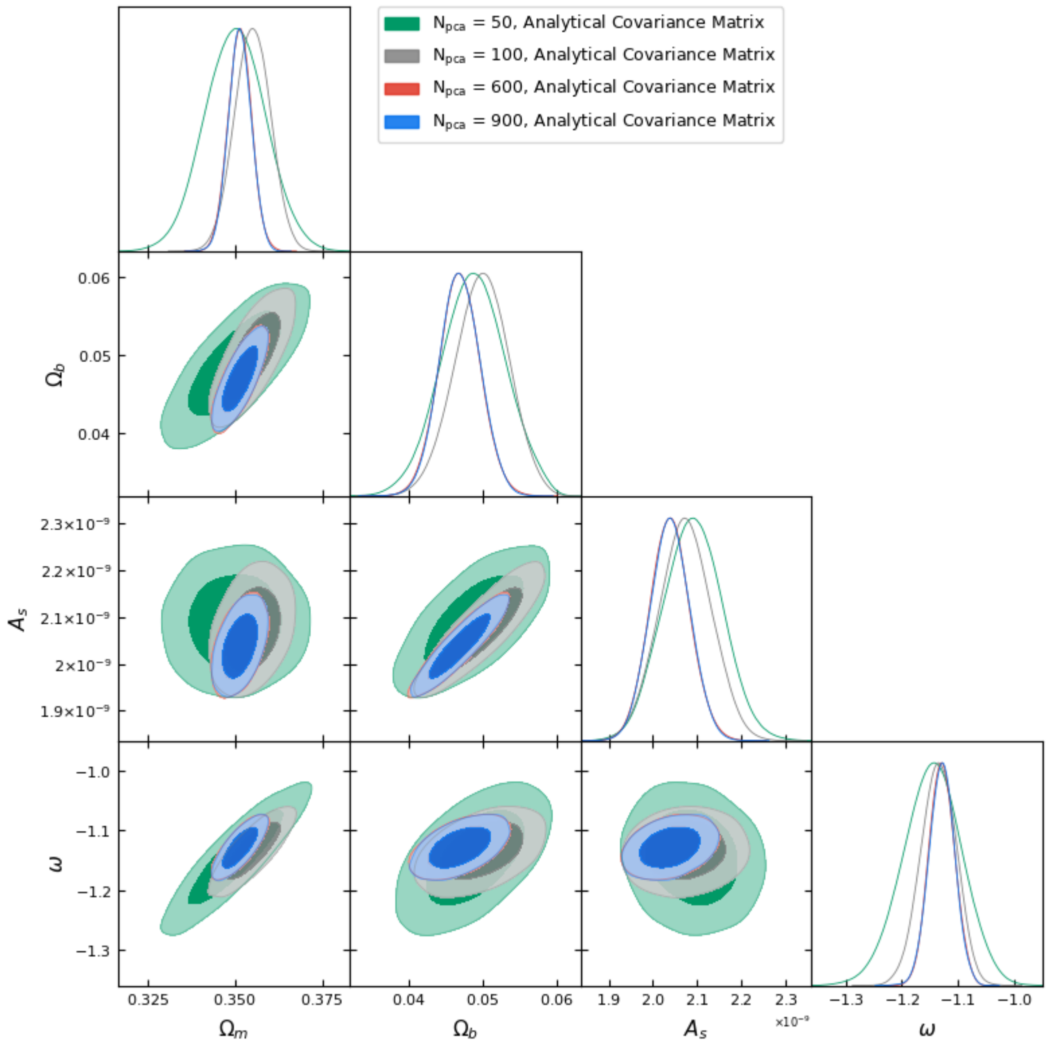

1e−1
Probability Density
χ²
1500 Covariance Matrix With Hartlap Correction
3000 Covariance Matrix With Hartlap Correction
10,000 Covariance Matrix With Hartlap Correction

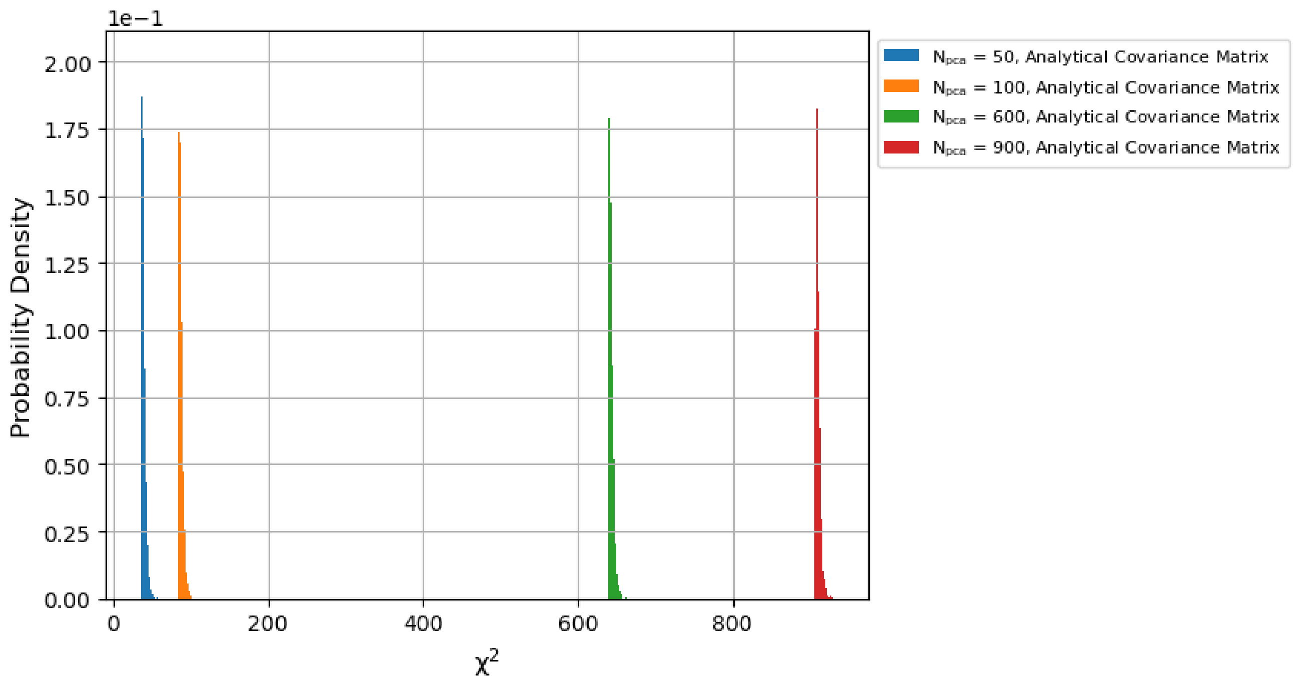

1500 Covariance Matrix
1500 Covariance Matrix With Hartlap Correction
Ω_m
Ω_b
A_s
ω


1e−1
1500 Covariance Matrix
1500 Covariance Matrix With Hartlap Correction
Probability Density
χ^2

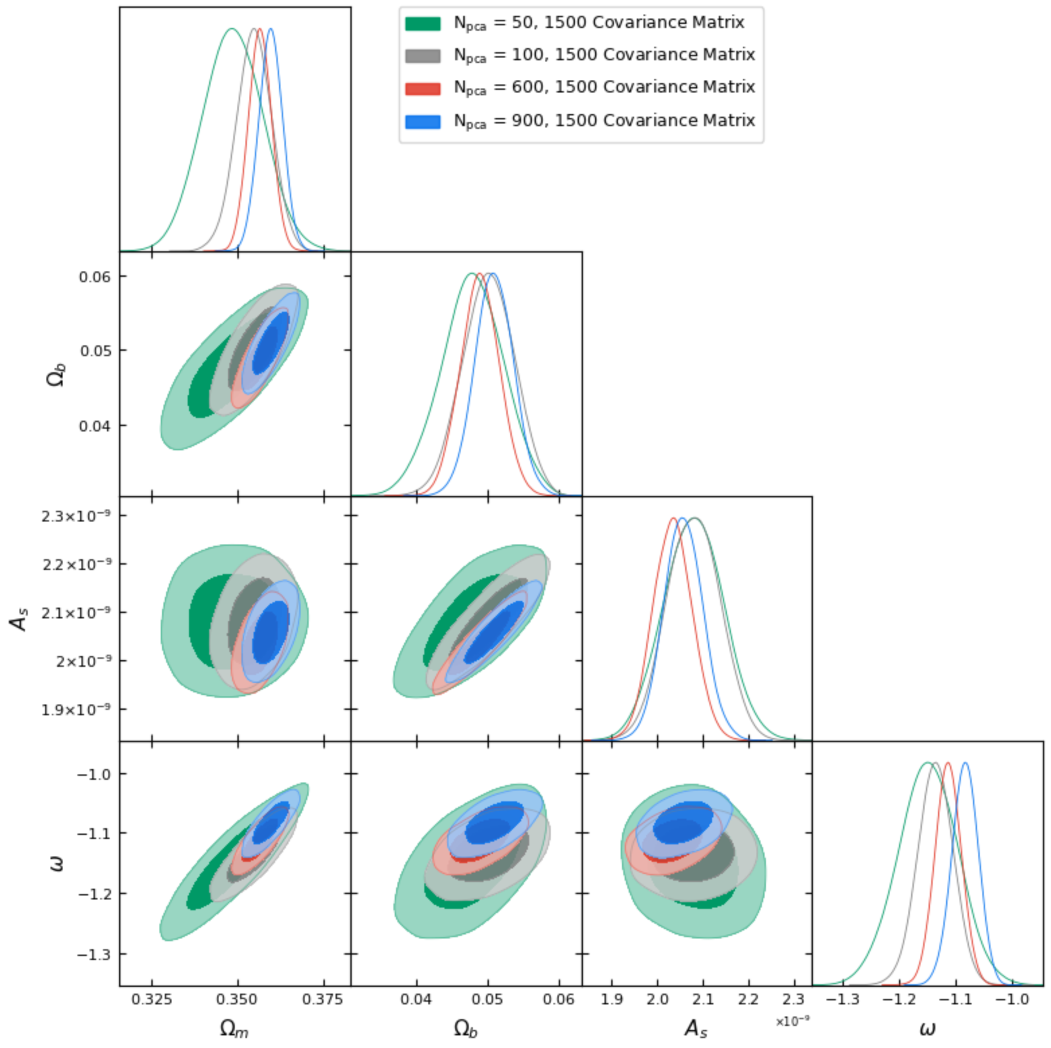
As can be seen from the plots above, the runs with and without Hartlap factor correction estimate the same best-fit values (comparing runs with the same covariance matrix). However, the width of the posteriors are larger when Hartlap factor correction is used. This is especially noticeable for the runs using the 1500 covariance matrix. It makes sense that Hartlap factor correction increases the width of the posteriors since it is some factor greater than 1 multiplied by the numerical covariance matrix. Additionally, the chi-squared distributions are shifted down the x-axis when Hartlap factor correction is used. This result is logical, as the chi-squared value is proportional to the inverse of a covariance matrix, whose elements become smaller with Hartlap factor correction. Lastly, the width of the posteriors is consistent between runs using Hartlap factor correction. All of the results mentioned agree with the findings from Assignment 1. In that assignment, chi-squared distributions were also shifted down the x-axis with Hartlap factor correction. Also, there was less variance in the variance of chi-squared distributions when Hartlap factor correction was used.

For the next task, I ran an MCMC for different numbers of PCA elements (50, 100, 600, and 900) using the noisy reference model. Half of the runs used the analytical covariance matrix, whereas the other half used the 1500 covariance matrix with Hartlap factor correction. Below are the posterior and chi-squared distributions using the analytical covariance matrix:

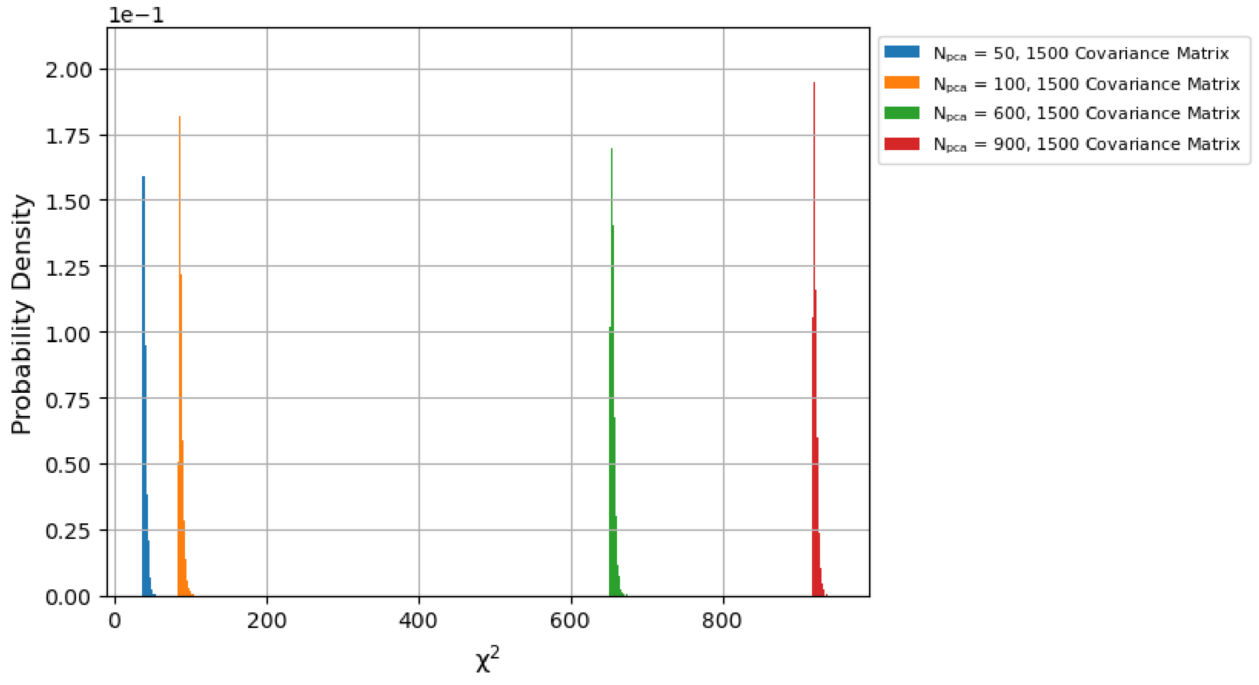




Posterior and chi-squared distributions using the 1500 covariance matrix:

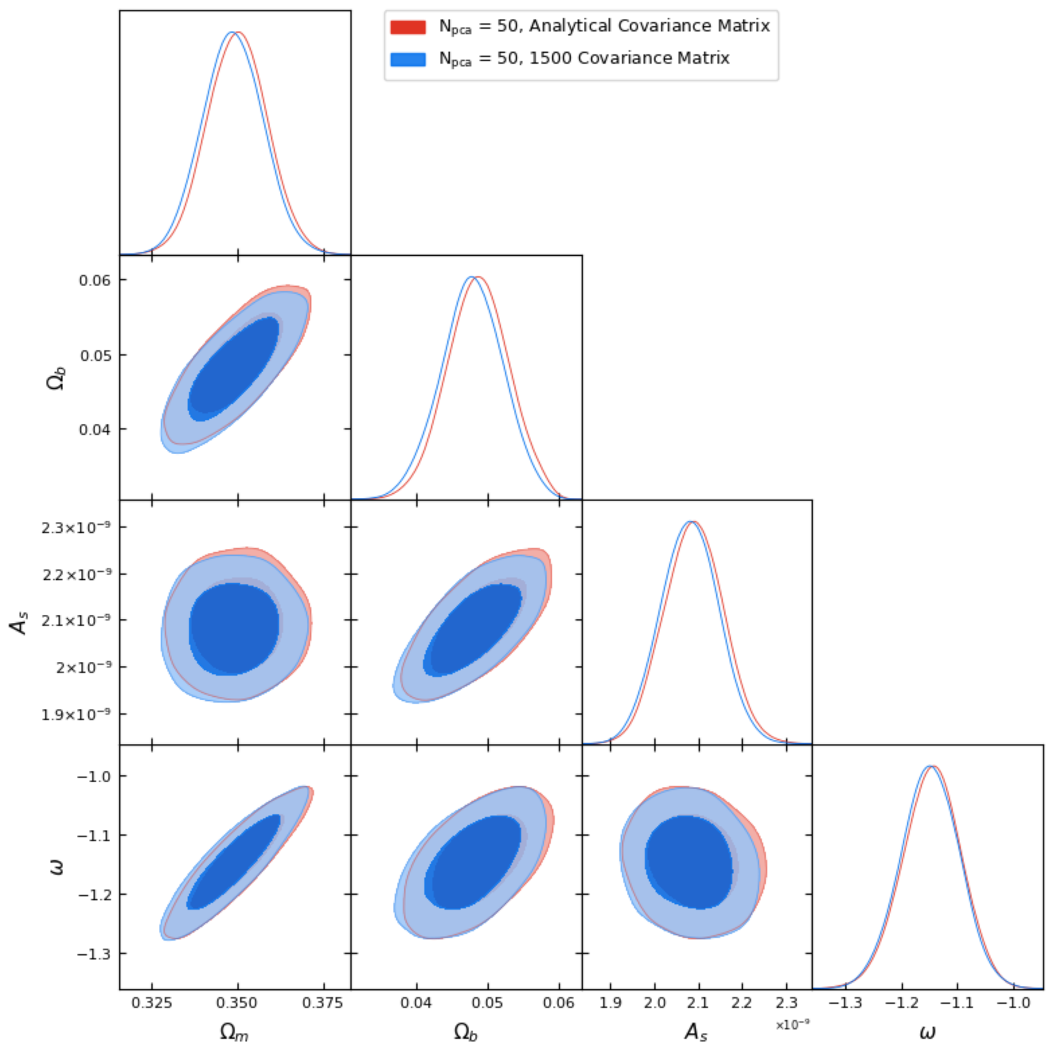


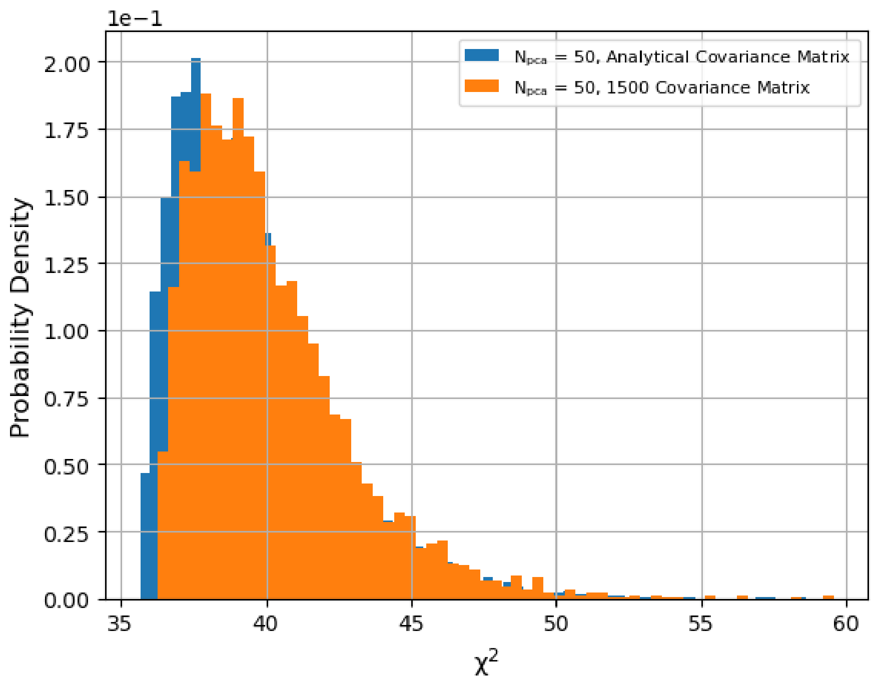
N_pca = 50, 1500 Covariance Matrix
N_pca = 100, 1500 Covariance Matrix
N_pca = 600, 1500 Covariance Matrix
N_pca = 900, 1500 Covariance Matrix
Ω_b
A_s
ω
Ω_m
0.325 0.350 0.375
0.04 0.05 0.06
1.9 2.0 2.1 2.2 2.3 ×10⁻⁹
−1.3 −1.2 −1.1 −1.0



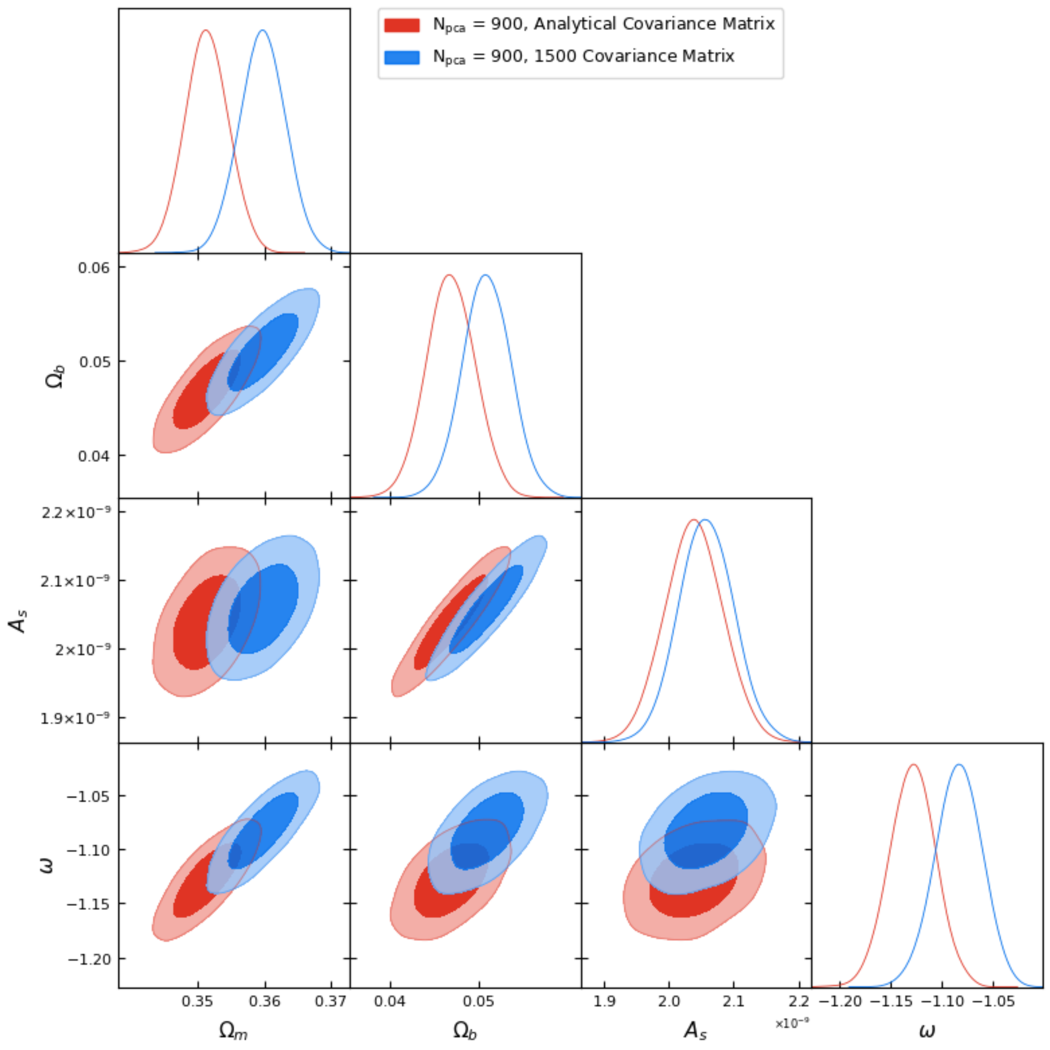
1e−1
Probability Density
χ²
N_pca = 50, 1500 Covariance Matrix
N_pca = 100, 1500 Covariance Matrix
N_pca = 600, 1500 Covariance Matrix
N_pca = 900, 1500 Covariance Matrix
0 200 400 600 800
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

Posterior and chi-squared distributions for $N_{pca} = 50$:

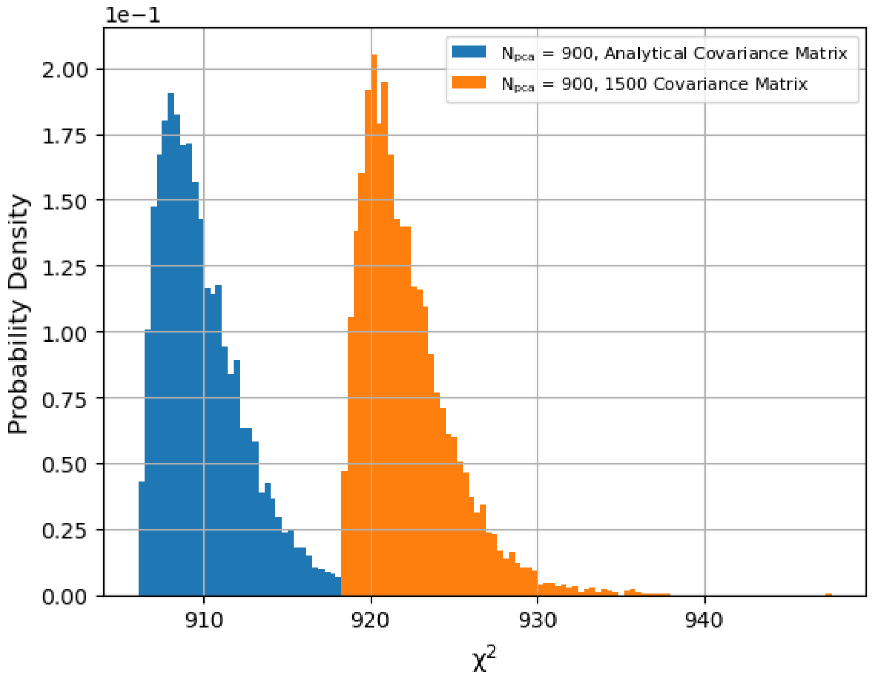




Posterior and chi-squared distributions for $N_{\text{pca}} = 900$:



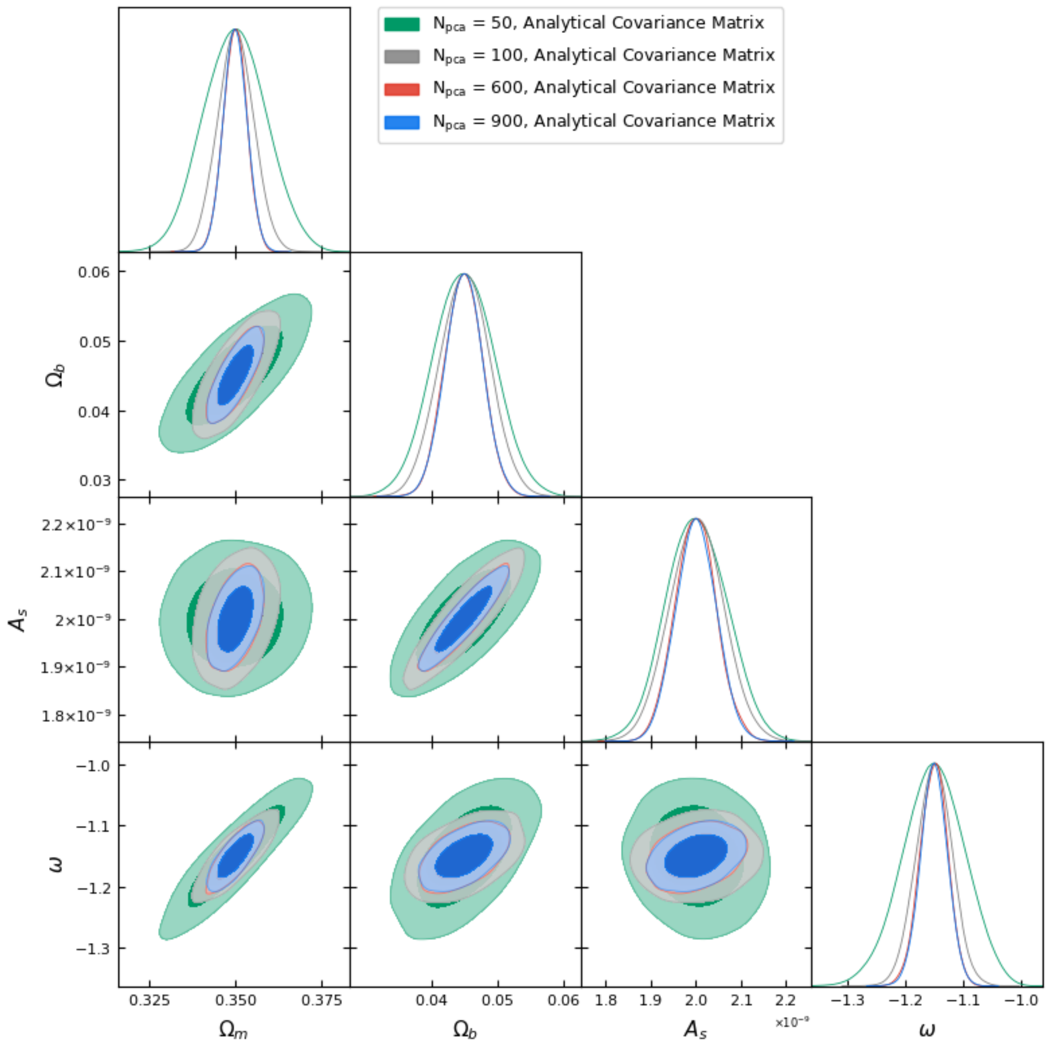
N_pca = 900, Analytical Covariance Matrix
N_pca = 900, 1500 Covariance Matrix
Ω_b
A_s
ω
Ω_m
0.06
0.05
0.04
2.2×10⁻⁹
2.1×10⁻⁹
2×10⁻⁹
1.9×10⁻⁹
−1.05
−1.10
−1.15
−1.20
0.35
0.36
0.37
0.04
0.05
1.9
2.0
2.1
2.2
×10⁻⁹
−1.20
−1.15
−1.10
−1.05

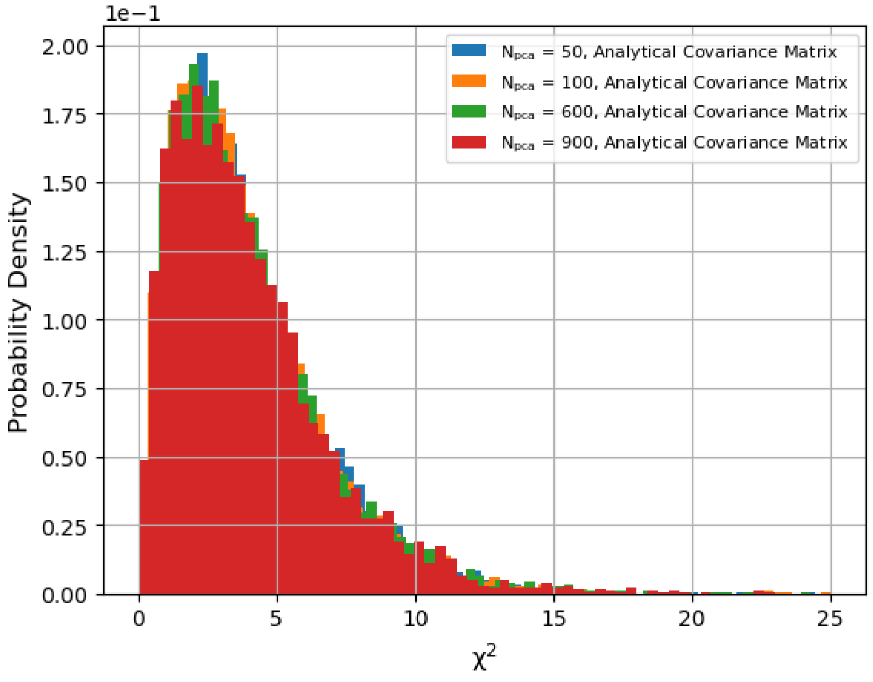


1e−1
N_pca = 900, Analytical Covariance Matrix
N_pca = 900, 1500 Covariance Matrix
Probability Density
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
910
920
930
940
χ²

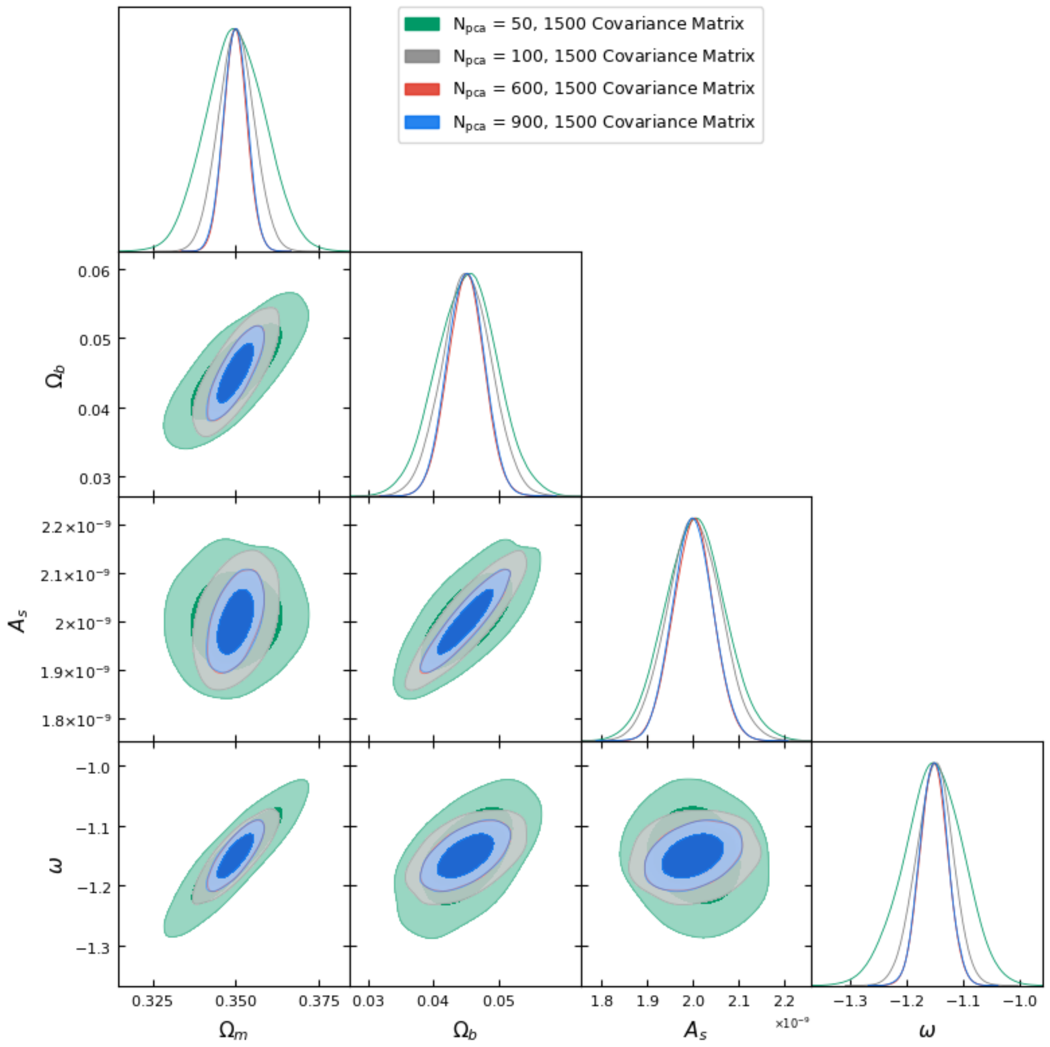
Looking at the plots above, it is clear that a larger number of PCA elements results in smaller error bars/tighter parameter constraints. Also, the larger $\mathrm{N_{pca}}$ runs (600 and 900) that used the analytical covariance matrix produced essentially the same posteriors. Interestingly, when $\mathrm{N_{pca}}$ is small (50 or 100), the analytical and 1500 covariance matrices result in very similar posterior and chi-squared distributions. Note that the 1500 covariance matrix is not as accurate as the analytical covariance matrix. In Assignment 2, I saw that when $\mathrm{N_{pca}}$ was very small, I had much less constraining power compared to the original Fisher analysis. Essentially, there is a point where $\mathrm{N_{pca}}$ is too small and, as a result, too much information is lost. So, it is not unreasonable that when $\mathrm{N_{pca}}$ is small, I get similarly bad results between the analytical and 1500 covariance matrices. One final remark is that smaller $\mathrm{N_{pca}}$ produce chi-squared distributions that are closer to zero. The reason is that fewer PCA elements means fewer degrees of freedom, which shifts the chi-squared distribution closer to zero.

Lastly, the previous task was repeated using the noise-free reference model. The following are the posterior and chi-squared distributions using the analytical covariance matrix:

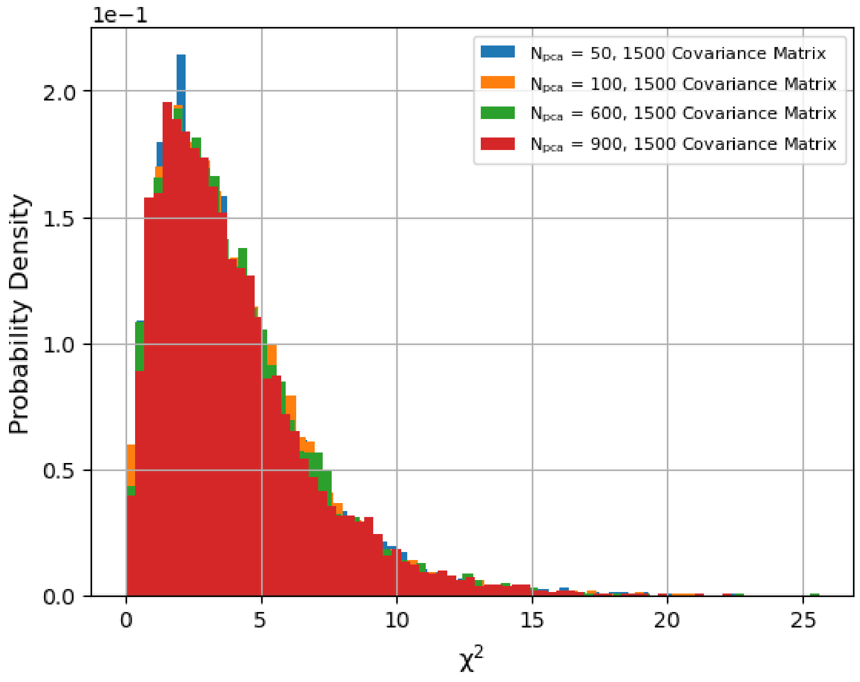




Posterior and chi-squared distributions using the 1500 covariance matrix:

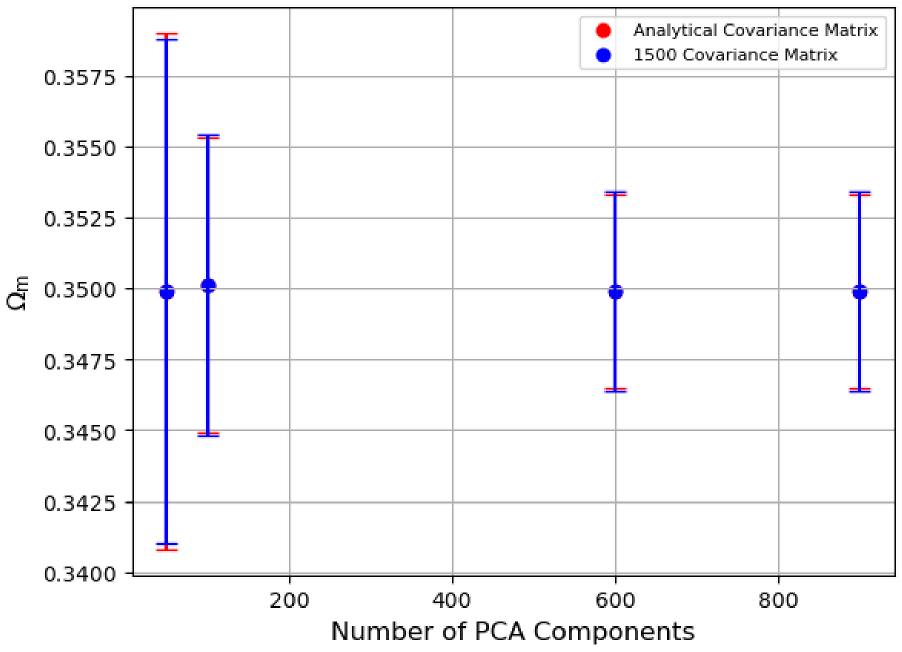


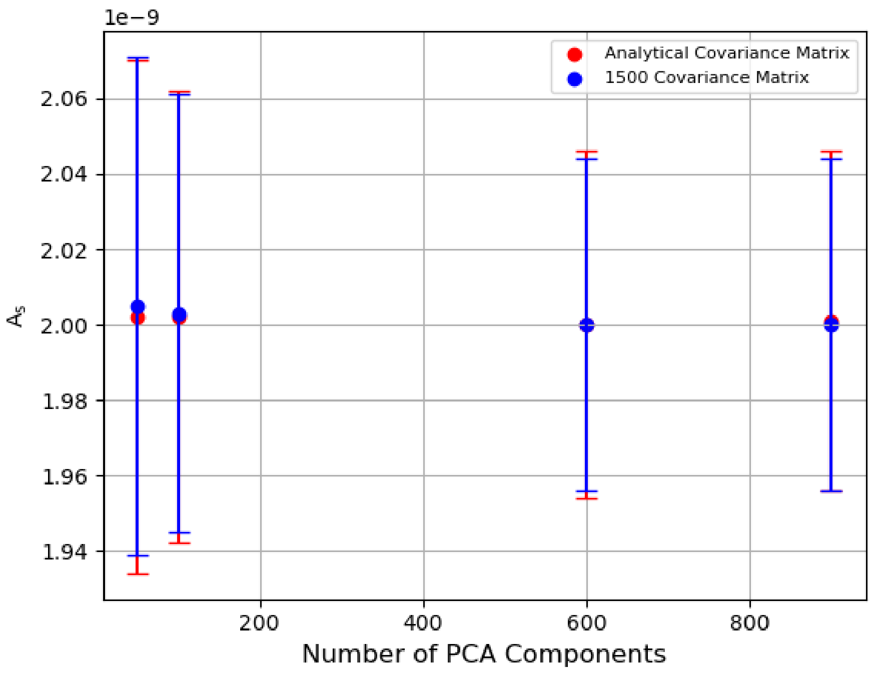
N$_{pca}$ = 50, 1500 Covariance Matrix
N$_{pca}$ = 100, 1500 Covariance Matrix
N$_{pca}$ = 600, 1500 Covariance Matrix
N$_{pca}$ = 900, 1500 Covariance Matrix
Ω_b
A_s
ω
Ω_m
0.325 0.350 0.375
0.03 0.04 0.05 0.06
1.8 1.9 2.0 2.1 2.2 $\times10^{-9}$
−1.3 −1.2 −1.1 −1.0



1e−1
N$_{pca}$ = 50, 1500 Covariance Matrix
N$_{pca}$ = 100, 1500 Covariance Matrix
N$_{pca}$ = 600, 1500 Covariance Matrix
N$_{pca}$ = 900, 1500 Covariance Matrix
Probability Density
χ^2
0.0 0.5 1.0 1.5 2.0
0 5 10 15 20 25

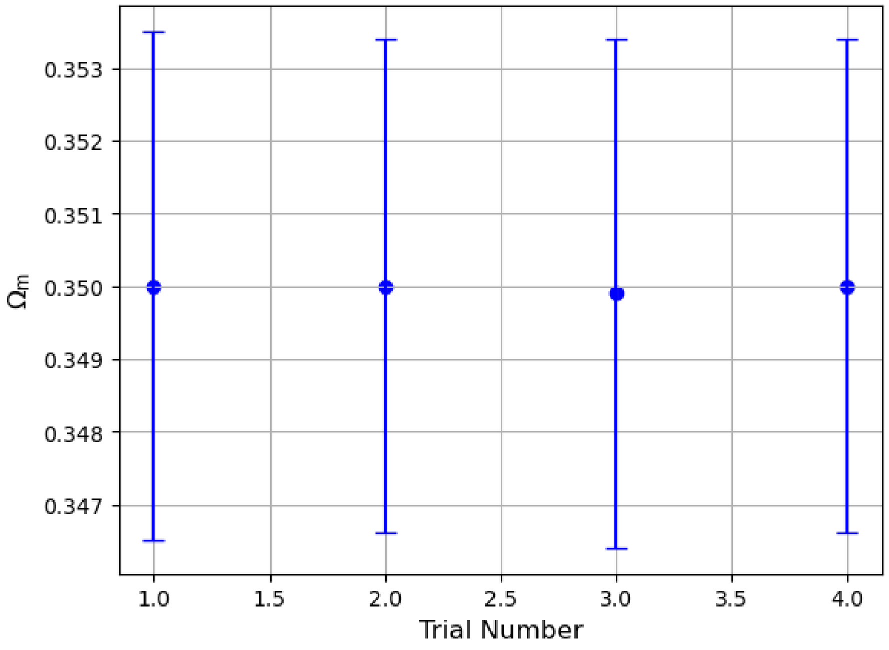
When the noise-free reference model is used, all of the posterior distributions are centered at the same parameter values, and the chi-squared distributions overlay each other. Also, we can still see the trend where, as N_{pca} increases, the posteriors narrow. Additionally, since a noise-free reference model was used, the degrees of freedom came from varying the four cosmological parameters. Thus, the chi-squared distributions are *significantly* shifted down the x-axis. Some other plots that were made during this task were the marginalized constraints on Ω_m and A_s versus N_{pca}:

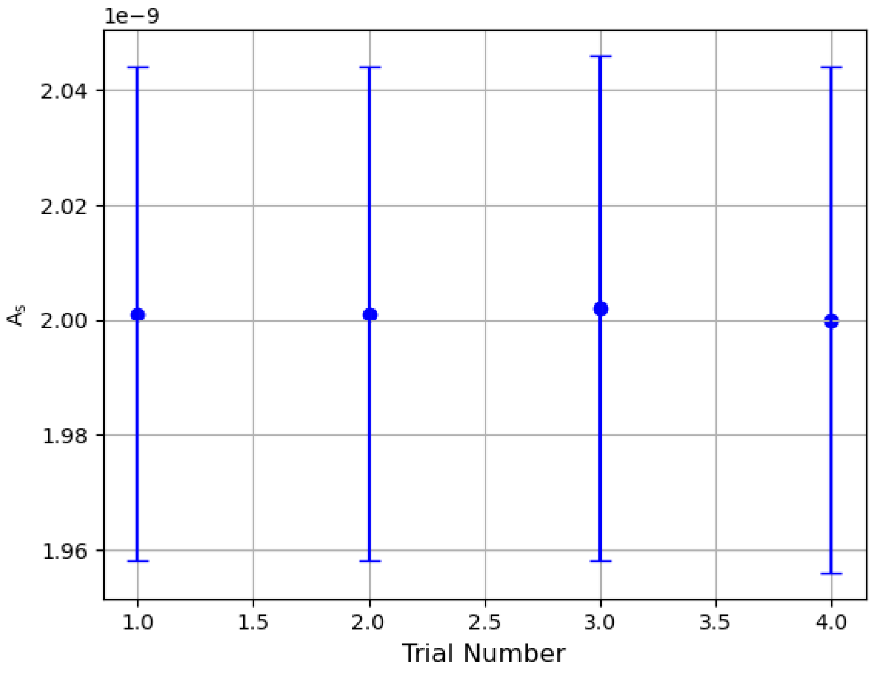




From the plots above, we see that the best-fit parameter values from the analytical and 1500 covariance matrices are very close to each other, and these values do not really change as N_{pca}

increases. However, we can see that the error bars get smaller as N_{pca} increases, which is precisely what we have observed in previous posterior distributions. That being said, there is some point where regardless of how large N_{pca} is, the error bars remain essentially the same size. This seems to occur around $N_{pca} = 600$. To ensure that the trend of decreasing error bars with increasing N_{pca} was real, I ran many identical MCMC chains. For these runs, I used the 1500 covariance matrix, 600 PCA elements, and the noise-free reference model. I then plotted the marginalized constraints on Ω_m and A_s versus trial number:

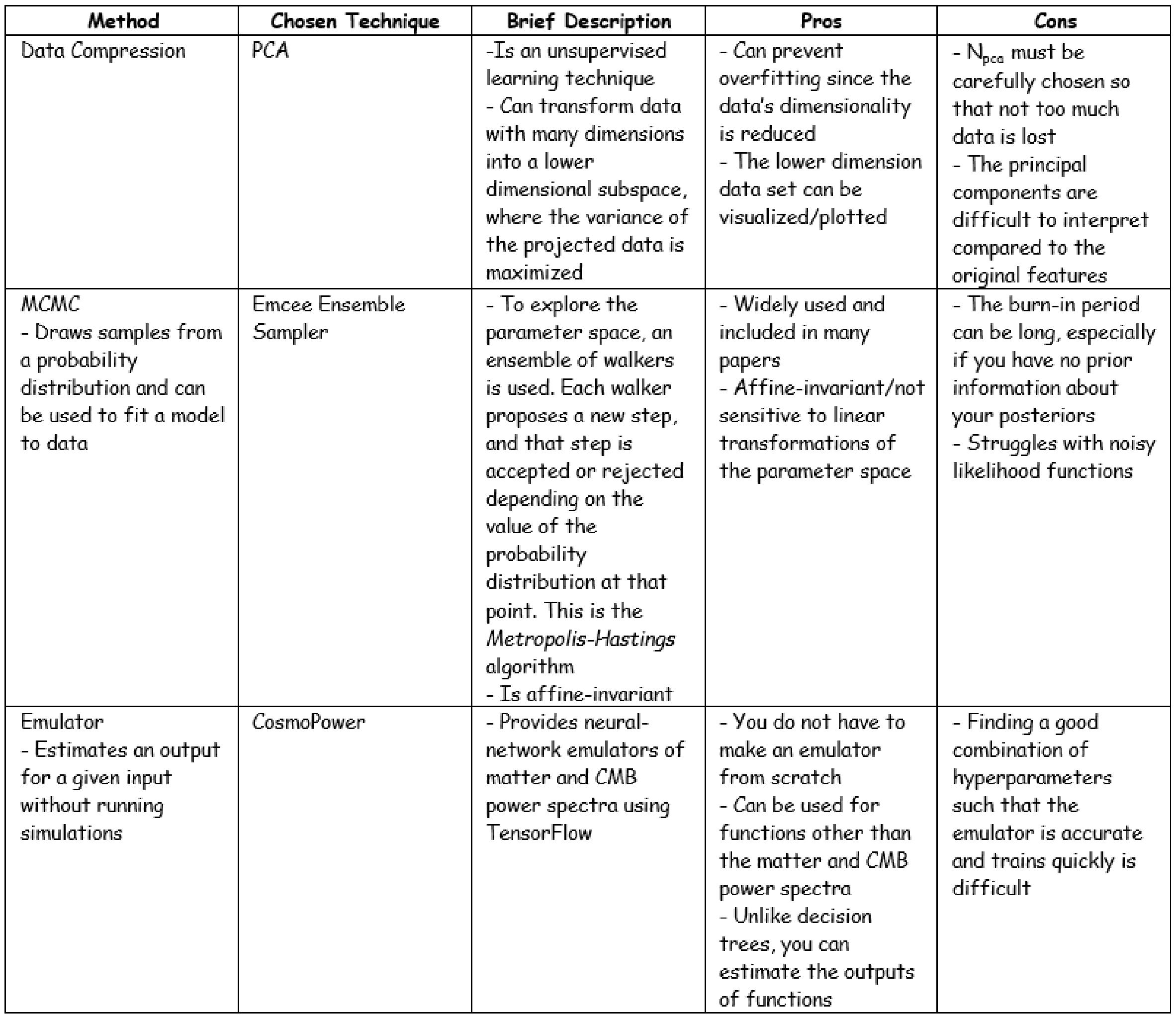




Luckily, the parameter constraints remained consistent across different trials. Also, the variances in the errors shown above are about three orders of magnitude smaller than those for the plots of marginalized constraints versus N_{pca}. Thus, the trends previously mentioned are real, and not just artifacts of MCMC producing different results for different trials.

2 Methods

The table below summarizes the main methods used throughout the assignments:



Method	Chosen Technique	Brief Description	Pros	Cons
Data Compression	PCA	-Is an unsupervised learning technique - Can transform data with many dimensions into a lower dimensional subspace, where the variance of the projected data is maximized	- Can prevent overfitting since the data's dimensionality is reduced - The lower dimension data set can be visualized/plotted	- N_{pca} must be carefully chosen so that not too much data is lost - The principal components are difficult to interpret compared to the original features
MCMC - Draws samples from a probability distribution and can be used to fit a model to data	Emcee Ensemble Sampler	- To explore the parameter space, an ensemble of walkers is used. Each walker proposes a new step, and that step is accepted or rejected depending on the value of the probability distribution at that point. This is the *Metropolis-Hastings* algorithm - Is affine-invariant	- Widely used and included in many papers - Affine-invariant/not sensitive to linear transformations of the parameter space	- The burn-in period can be long, especially if you have no prior information about your posteriors - Struggles with noisy likelihood functions
Emulator - Estimates an output for a given input without running simulations	CosmoPower	- Provides neural-network emulators of matter and CMB power spectra using TensorFlow	- You do not have to make an emulator from scratch - Can be used for functions other than the matter and CMB power spectra - Unlike decision trees, you can estimate the outputs of functions	- Finding a good combination of hyperparameters such that the emulator is accurate and trains quickly is difficult

The techniques above were chosen as they are easy to implement, well-documented, and are widely used in academia. Note that the choices and motivations for hyperparameters can be found in the **Assignments Overview**.

3 Research Application

All of the techniques from these assignments can be used in my research. Previously, I attempted to run MCMC chains that compute the projected correlation function, $\mathrm{w_p}$, within the likelihood function. However, the projected correlation function can take a long time to evaluate (up to minutes). So, the chains never completed. As a workaround, I am currently building a neural-network emulator that will provide the projected correlation function for a given set of parameters. The emulator should output the projected correlation function in fractions of a second, greatly speeding-up the likelihood function. To further reduce the likelihood function's computation time, I can also use PCA to compress data. The likelihood function is $\frac{-\chi^2}{2}$, where χ^2 is computed via matrix multiplication involving a data vector and an inverse covariance matrix. Thus, using PCA to reduce the dimensionality of the data vector and inverse covariance matrix would speed-up the matrix multiplication.