Statistical Tools for Astronomers

PHYS 788 Final Report

Rashaad Reid

University of Waterloo

April 22, 2024

Assignment 1

As part of the two-point statistics project, our first assignment concerned the use of χ^2 distributions, covariance matrices, and the Hartlap factor. We began by generating a set of noisy data vectors based on a noiseless reference model using an analytic covariance matrix. We then computed the χ^2 values for each of these noisy models and produced a normalized histogram of these values, as can be seen in Figure 1. Comparing the resulting distribution to the theoretical χ^2 distribution for a system with 900 degrees of freedom, we see in Figure 1 that the noisy vectors are distributed as predicted. This demonstrates how χ^2 distributions can be used to estimate the degrees of freedom of a system and check that a data set is distributed as expected.

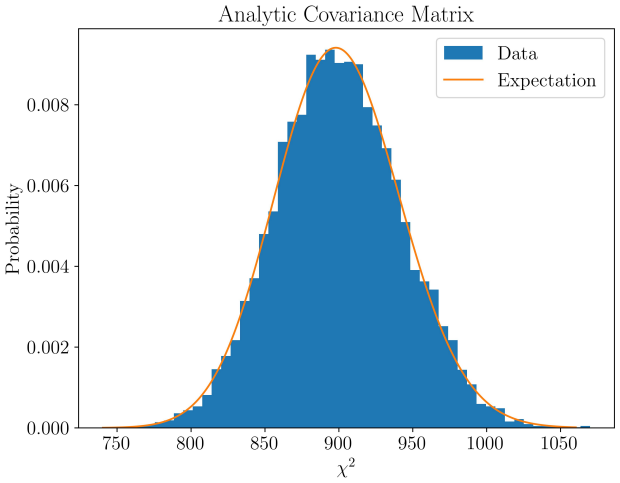

Figure 1: The χ^2 distribution of a set of noisy data vectors generated from a reference model and an analytic covariance matrix for Assignment 1. The blue region is a normalized histogram of the χ^2 values of the noisy vectors, and the orange curve is the theoretical χ^2 distribution for a system with 900 degrees of freedom.

We then performed the same procedure using a numerical covariance matrix. Numerical covariance matrices are useful in contrast to analytic matrices because they can be computed directly from realisations of the data, and are thus typically available when analytic matrices are not. Additionally, they are always invertible if they are produced from more realisations than the dimension of the matrix. The left panel of Figure 2 shows the results for a set of vectors generated from a numerical covariance matrix computed from 5000 data vectors. We see that this matrix produced a greater χ^2 distribution than expected. This shift is more significant when the covariance matrix is generated from fewer data vectors. This shift is due to the numerical covariance matrix being biased when computed from too few model vectors, highlighting a drawback of using numerical matrices. The Hartlap factor is a scalar that can be used to correct this bias in a numerical covariance matrix. The Hartlap factor is defined as

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

where n is the number of realisations used to produce the matrix, and m is its dimension. The right panel of Figure 2 shows the resulting χ^2 distribution after the covariance matrix has been corrected by the Hartlap factor. We see in Figure 2 that the matrix has indeed been debiased and that the result now matches the expected distribution. We make use of the Hartlap factor in future assignments in order to bebias our numerical covariance matrices.

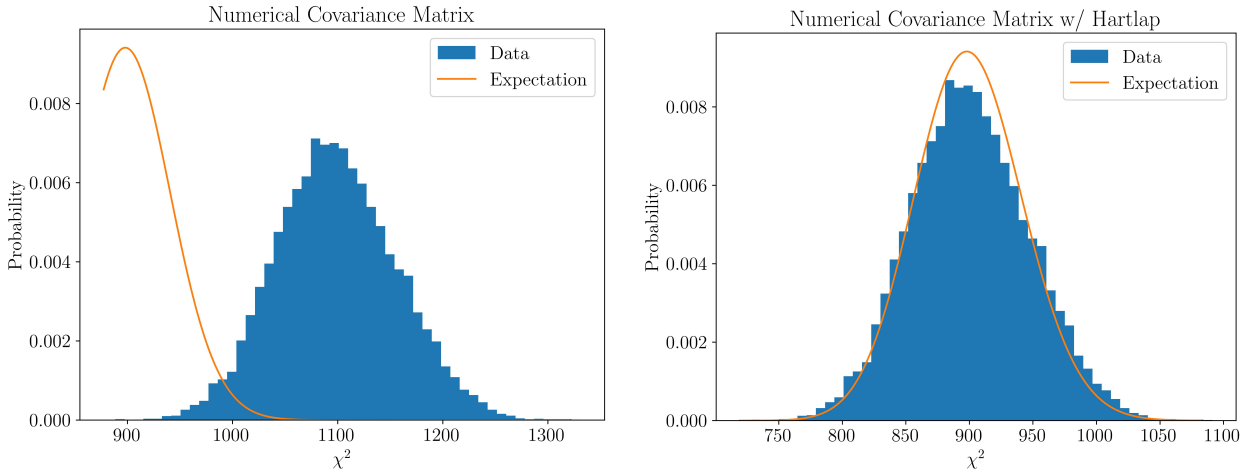

Figure 2: The χ^2 distributions of a set of noisy data vectors generated from a reference model and a numerical covariance matrix computed from 5000 vectors for Assignment 1. The blue regions are normalized histograms of the χ^2 values of the noisy vectors, and the orange curves are the theoretical χ^2 distributions for a system with 900 degrees of freedom. The left panel shows the χ^2 distribution from the biased numerical covariance matrix. The right panel shows the χ^2 distribution for the numerical covariance matrix corrected using the Hartlap factor.

Assignment 2

Assignment 2 was focused on neural network emulators and Principal Component Analysis (PCA). Emulators are a useful tool as they can be used to predict the features of a model without the need for more computationally expensive methods, such as simulation. We trained a neural network emulator on a set of data vectors in order to predict the features of a model from a set of four input parameters. We used a neural network in particular to emulate this data because it's well-suited to complex regression problems, unlike other unsupervised learning techniques, such as decision trees, which are better suited to classification. Neural networks also don't require us to make assumptions about the model, unlike linear regression techniques, which makes them more versatile. While training the emulator, we chose our hyperparameters by first selecting a low number of neurons and a large batch size in order to produce fast results and determine the most effective learning rates. We chose learning rates of 1e-2, 1e-3, 1e-4, 1e-5, and 1e-6, after which the validation loss didn't greatly improve. Then, we increased the number of neurons and decreased the batch size as far as our computational power would allow to 4 layers of 512 neurons, and a batch size of 350. We didn't find significant difference in validation loss between using many neurons in few layers or few neurons in many layers. Once the emulator had been trained, its predictions were compared to a test set of data vectors. Figure 3 shows the differences between the emulated features and the test features. We see from the top panel of the figure that the emulator is able to predict all of the points in 99% of the test models to within 0.1 times the standard deviation of the points, as determined from the analytic covariance matrix. We see in the bottom panel that the mean differences between the emulator and the models at each point are on the order of 0.001σ. This illustrates that our emulator is well trained and able to accurately predict the model features.

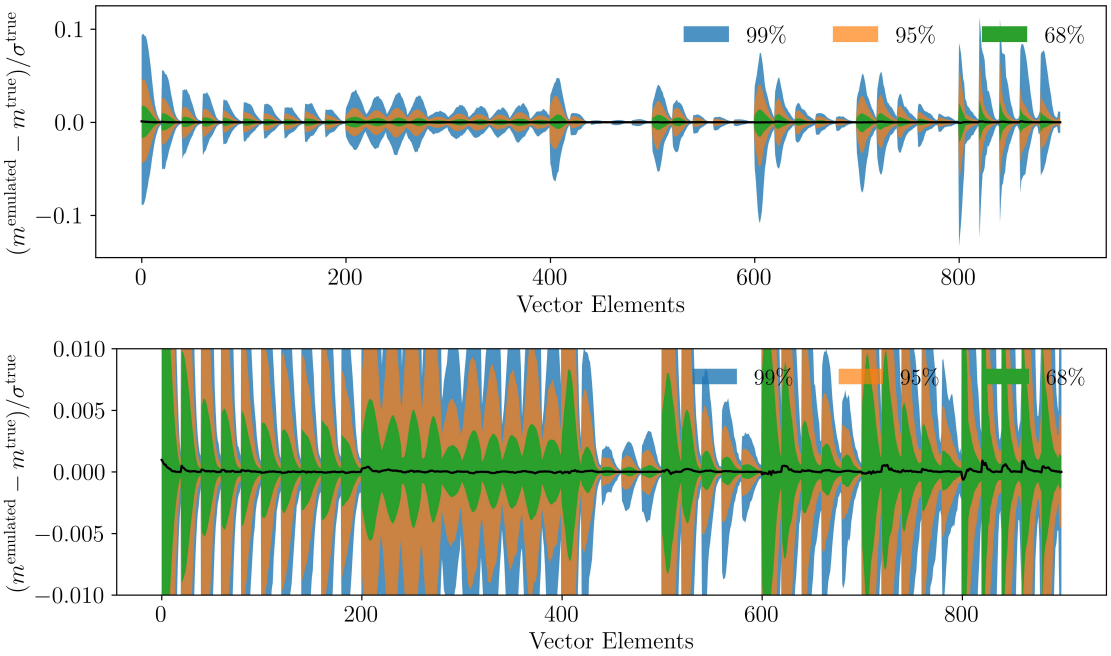

Figure 3: The difference between a set of test data vectors and the features predicted by our emulator for Assignment 2. The difference is scaled by the standard deviation of each point, determined from the analytic covariance matrix. The black line represents the mean difference at each point along the vector, and the shaded contours contain the differences from the indicated fraction of the 3000 test vectors.

After having trained the emulator, Assignment 2 then turned to exploring the efficacy of PCA compression. Principal component analysis is a useful form of compression because, unlike other compression techniques, it can be applied to raw data and doesn't require us to assume any particular structure. Additionally, it explicitly reduces the dimension of the data, unlike other compression algorithms such as Massively Optimised Parameter Estimation and Data compression (MOPED). This dimension reduction makes it easier to work with large data sets and allows us to identify the components of the data that carry the most information. To examine the effectiveness of PCA compression, we first used the Fisher information matrix to determine the constraints on the parameters Ω_m and ω from our full data vectors with 900 points. We then compressed our data vectors using various numbers of PCA elements from the minimum of 1 to the maximum of 900 and computed the same constraints for each case. We found through Figure 4 that the constraints are poorer for fewer PCA elements, but that they converge to the maximum constraining power for more PCA elements. The loosening of constraints for fewer elements is expected since PCA necessarily reduces the information carried by the data when using fewer elements. This highlights a disadvantage of the PCA compression algorithm in terms of data preservation relative to other methods, such as MOPED, which can retain information almost perfectly. However, we see in Figure 4 that the data can be reduced significantly to 591 elements while still getting within 1% of the maximum constraining power for both tested parameters.

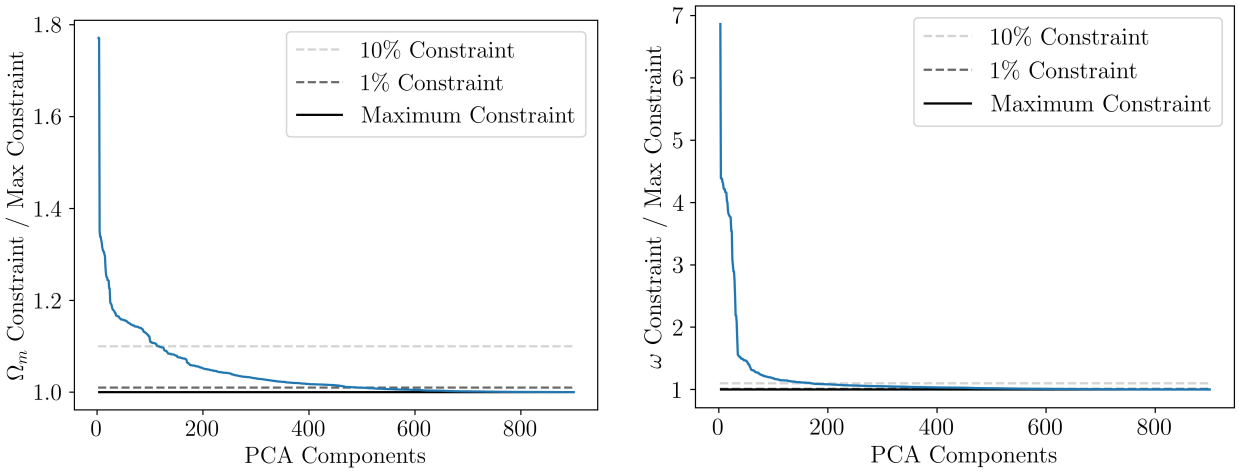

Figure 4: The constraints via Fisher analysis on the values of Ω_m and ω as a function of the number of PCA elements for Assignment 2. The solid black lines represent that maximum constraints from the raw data vectors. The dashed lines represent the thresholds for being within 1% and 10% of the maximum constraint, respectively. The blue curves represent the constraints as a function of the number of PCA elements.

Assignment 3

The final assignment focused on using Markov chain Monte Carlo (MCMC) techniques in order to constrain the model parameters for a given noisy data vector. MCMC algorithms are useful as they allow us to draw random samples from an unknown posterior distribution and efficiently explore areas of interest in our parameter space. We used the Python package "emcee" for our MCMC runs, which uses a modified Metropolis-Hastings algorithm that shares information between walkers when generating a proposal distribution for each step. This means that the algorithm is optimized for using many walkers, as they can gain more information from each other about the posteriors. This algorithm is generally preferable to a standard Metropolis-Hastings method, as it requires far fewer steps due to its short auto-correlation time. However, a drawback of this algorithm is that it is difficult to parallelize, since each walker is dependent on the positions of the others. With all of this in mind, it would have been optimal to use a few hundred walkers and a few hundred steps to explore our parameter space. However, I was not aware of emcee's particular algorithm until after I had run my chains, so I used 50 walkers and 3000 steps. These were chosen as they were at the limit of what my computer could run in a reasonable amount of time, and they generated smooth posteriors and χ^2 distributions, as we will see. I also chose only 100 burn-in steps, as a greater amount didn't seem to affect my parameter constraints and would have thrown away more data than necessary. Lastly, I chose to spawn the walkers at random locations near the correct posteriors, determined from preliminary MCMC chains, so as to avoid walkers getting stuck in distant local minima in the likelihood.

Figure 5 shows the posteriors and χ^2 distribution for our first MCMC run, which used the analytic covariance matrix for the given noisy model. We see that the constraints on the parameters are smooth and Gaussian, as expected. The χ^2 distribution peaks at a value slightly higher than 900, which is higher than may be expected. We would in principle expect the peak of the distribution to be around 896, as the model vector has 900 degrees of freedom, and we're fitting 4 model parameters to it. The discrepancy between the expectation and the result is due to the MCMC walkers jumping around the parameter space and not landing directly on the best fit parameters. In principle, a

smaller step size could reduce this χ^2 distribution if the walkers were still able to efficiently explore the parameter space with a smaller learning rate.

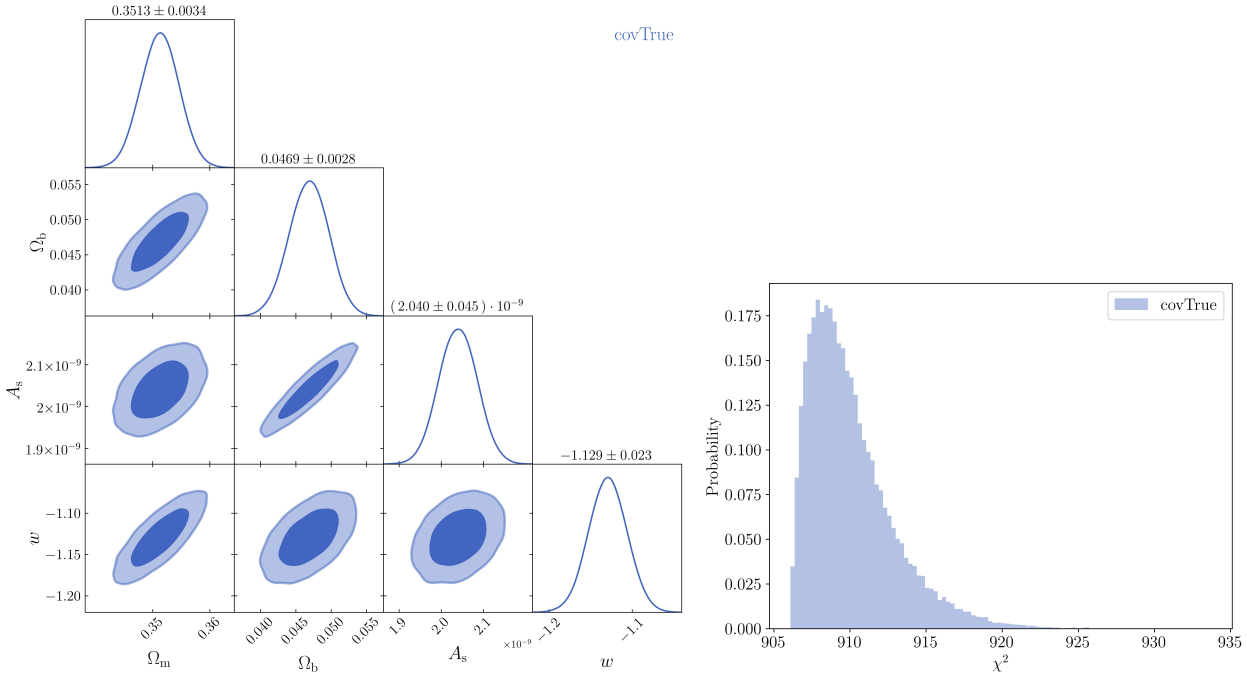

Figure 5: The posteriors for the four model parameters and the χ^2 distribution from an MCMC chain run on a noisy reference model using the analytic covariance matrix for Assignment 3.

Having run an initial MCMC using the analytic covariance matrix, a range of other MCMC chains were run for various covariance matrices and PCA compressions. The results of many of these chains are summarized in Figure 6. In the top left panel of Figure 6, we see that when using numerical covariance matrices with Hartlap corrections, the posteriors shift depending on the number of realisations used to produce the matrices. This indicates that MCMC runs are sensitive to noise in the covariance matrix, despite being debiased by Hartlap. In the top right panel, we see that using more PCA elements results in tighter constraints. This is expected since we're including more information with more elements. In the bottom left panel, we see that the posteriors shift when using a numerical covariance matrix and different numbers of PCA elements. This suggests that PCA compression is less reliable when using noisy numerical covariance matrices. Lastly, in the bottom right panel, we see that PCA compression is extremely effective for a noiseless model, even with a numerical covariance matrix. This is clear since the posteriors are all concentric and converge quickly for few PCA elements.

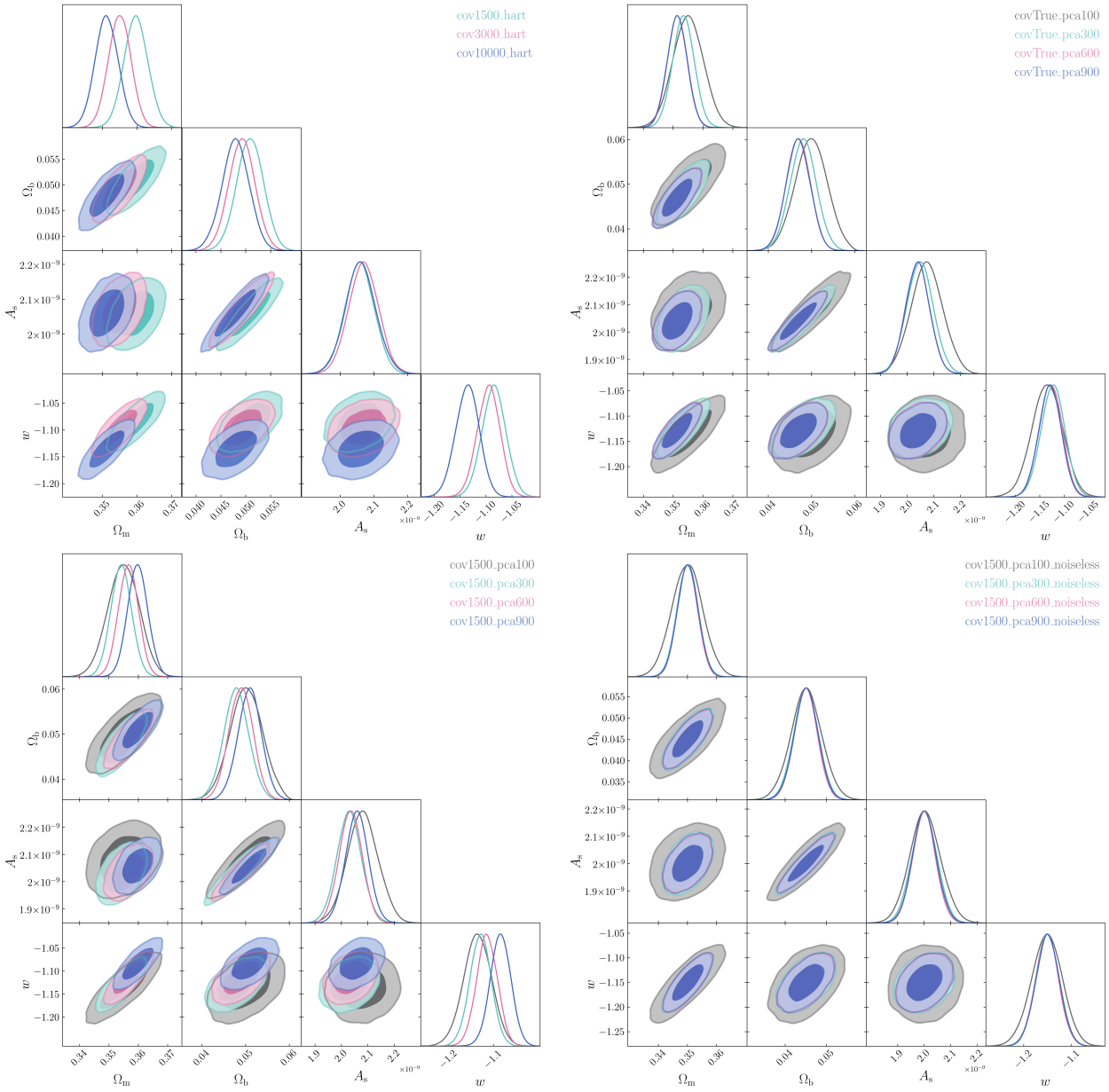

Figure 6: The posteriors for the four model parameters from various MCMC runs with different covariance matrices, reference vectors and PCA elements for Assignment 3. The top left panel shows the posteriors for numerical covariance matrices generated from different numbers of data vectors, each corrected by the Hartlap factor. The top right panel shows posteriors from an analytic covariance matrix with various numbers of PCA elements. The bottom left panel shows posteriors from a numerical covariance matrix with various numbers of PCA elements. The bottom right panel shows posteriors for a noiseless reference model from a numerical covariance matrix with various numbers of PCA elements.

Taking a closer look at PCA compression for MCMC chains, Figure 7 shows the constraints from some MCMC runs on Ω_m and ω as a function of the number of PCA elements used. The constraints generally tighten for more PCA elements, similarly to in Figure 4. However, there is a slight minimum in the constraints around approximately 500 PCA elements. This is because the constraints would typically be tighter for more PCA elements due to the inclusion of more infor-

mation, but the Hartlap factor provides a stronger correction for larger covariance matrices. This can be seen in Equation 1, where h deviates further from unity for large dimension m. So, these two competing factors result in a maximum constraint for a middling number of PCA elements. After comparing these results for a variety of MCMC chains with the same parameters, we found that the constraints had a standard deviation of less than 1% between runs, confirming that the results from the MCMC chains and PCA compression are reliable.

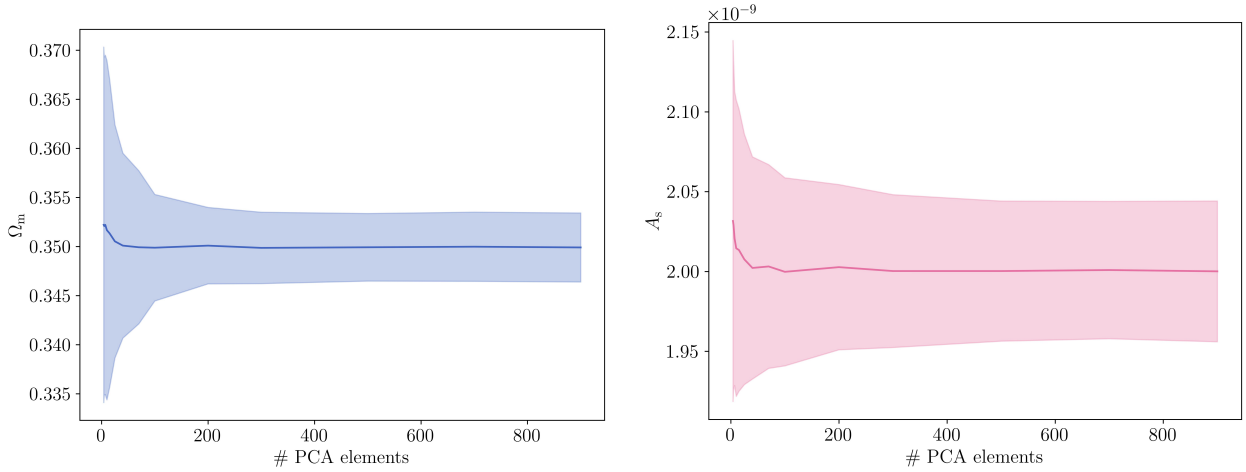

Figure 7: The standard deviations of Ω_m and ω from MCMC chains, as a function of the number of PCA elements used, for Assignment 3. The MCMC chains were run using a numerical covariance matrix generated from 1500 data vectors. The lowest number of PCA elements used for this plot is 4, and the greatest is 900.

Research Applications

Some of the techniques introduced in this course could be applied to improve the efficiency of my research. For example, a part of my work involves running computationally expensive simulations in order to produce data vectors for a variety of different input parameters. An example of such a data vector is shown by the errorbars in Figure 8. The neural network emulation technique that was used in Assignment 2 could be implemented here in order to produce data vectors for given input parameters without the need for laborious simulations. After having analysed these simulations, I then fit an analytic model with multiple free parameters to the data vector. An example of such a fit is shown by the smooth curve in Figure 8. The MCMC techniques used in Assignment 3 could be used to fit these parameters and determine their constraints much more efficiently than varying the parameters by hand, as I have done so far. These techniques, along with the general understanding of χ^2 distributions and covariances gained from Assignment 1, have significantly contributed to my ability to apply statistical methods to my astrophysics research.

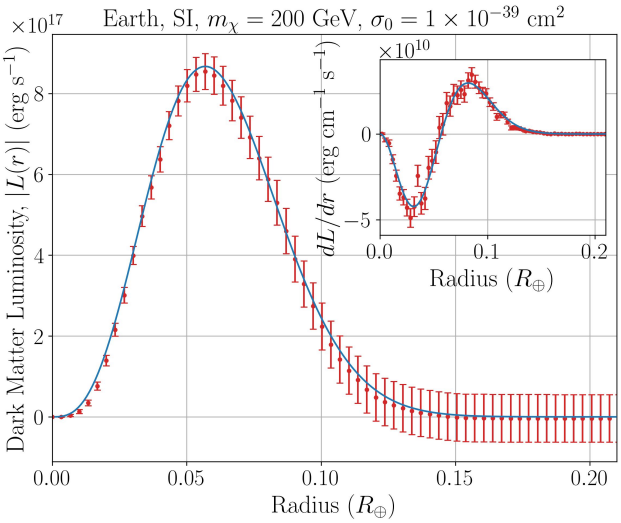

Figure 8: A data vector computed from simulation and a fitted analytic model from my own research. The red points with error bars represent the model vector generated from simulation, while the blue curve represents an analytic model with free parameters fitted to the data. This data represents the theoretical energy carried by a population of dark matter particles in the core of the Earth.