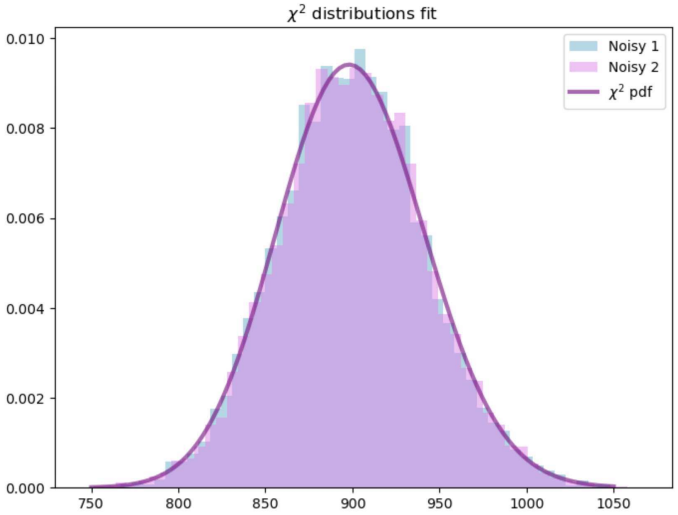
Statistical tools for astronomers - Final report

Sofia Chiarenza

April 2024

1 Assignment 1

In assignment 1 we investigated the covariance matrices, particularly focusing on the ones obtained from numerical simulations. We were given a 3×2 point statistic data vector with 900 degrees of freedom, and an analytical covariance matrix. Starting from the reference model, we generated two sets of 10000 data vectors containing some Gaussian noise and verified that they follow a theoretical χ^2 distribution, as shown in Figure 1



In the second part of the assignment, we looked into how numerical covariance matrices impact the χ^2 distributions. When the covariance is built with too few realizations, it becomes non-invertible and unreliable. However, with more realizations and employing the Hartlap factor correction, we are able to recover the expected distribution.

2 Assignment 2

In assignment 2, we built an emulator using the `cosmopower` software and implemented a PCA compression. The neural network emulator is trained on 10000 models with different cosmologies: I used 80% of them as a training sample, 10% of which became the validation sample. The other 20% constitutes the test sample. The model was trained on a normalized dataset in order to make the training more stable and improve accuracy. In particular, I used a fully connected neural network with 4 hidden layers, each with 512 neurons. The default `cosmopower` settings give a model with 3 hidden layers, but I found that the extra layer delivered more accurate results. I varied the learning rates between 10^{-2} and 10^{-6}, used a batch size of 500, and a patience value of 100. This is the number of epochs to wait before decreasing the learning rate if the loss function doesn't show improvement. To check the accuracy of the emulator, I used the test sample and

checked how many predictions are inside the 68%, 95% and 99% confidence intervals. I did that with two different metrics. One are the residuals, defined as:

$$\frac{m^{emu} - m^{true}}{m^{true}} \tag{1}$$

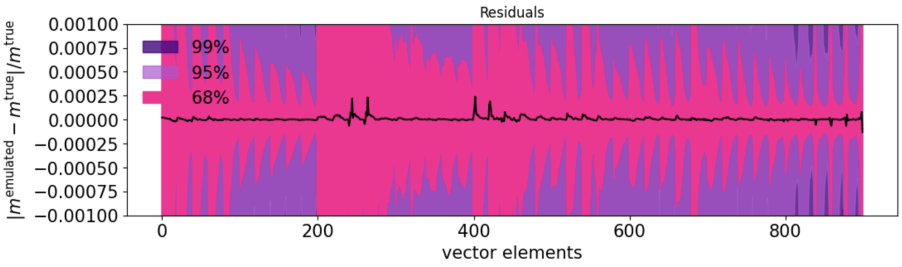


Figure 1: Residuals between the emulated models and the true ones. The 68%, 95% and 99% intervals are shown.

I also compared the results with the measurement error:

$$\frac{m^{emu} - m^{true}}{\sigma} \tag{2}$$

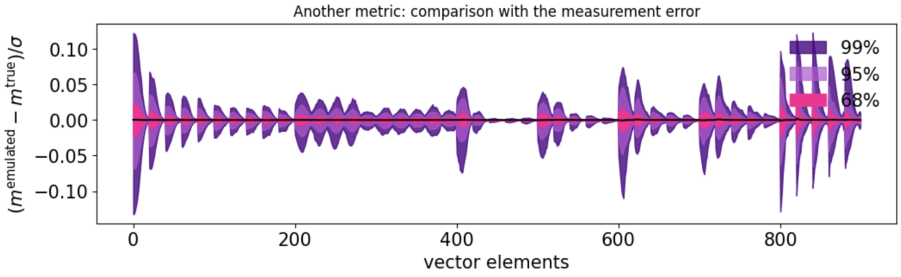


Figure 2: The figure shows the comparison of the differences between true and emulated models to the measurement error. The 68%, 95% and 99% intervals are shown.

Figure 1 shows very small residuals, on the order of magnitude of $\approx 10^{-4}$. This tells us that the emulator is indeed very accurate. From Figure 2 we can see that the emulator error is always below 10% of the measurement error, suggesting that the emulator can be used and is reliable in a real analysis.

In the second part we performed a Fisher matrix analysis to constrain the error bars of two cosmological parameters, Ω_m and w. The goal was to test how those errors are impacted by a PCA data compression. The results for the error bars are showed in Figure 3.

As expected, when the information is compressed the error bars on the parameters are larger. Figure 3 shows that 10% constraining power ($\frac{\sigma}{\sigma_{best}} = 1.1$) is lost when keeping ≈ 100 PCA elements, while the loss is of only 1% when keeping ≈ 400 elements.

3 Assignment 3

In assignment 3, we finally use the emulator to run Monte Carlo Markov Chains (MCMC) and constraint the cosmological parameters. In order to run a MCMC, we need to define the priors and the likelihood. The priors I used are flat, but restricted in the training range of the emulator. For the likelihood, I'm using the assumption of a Gaussian likelihood:

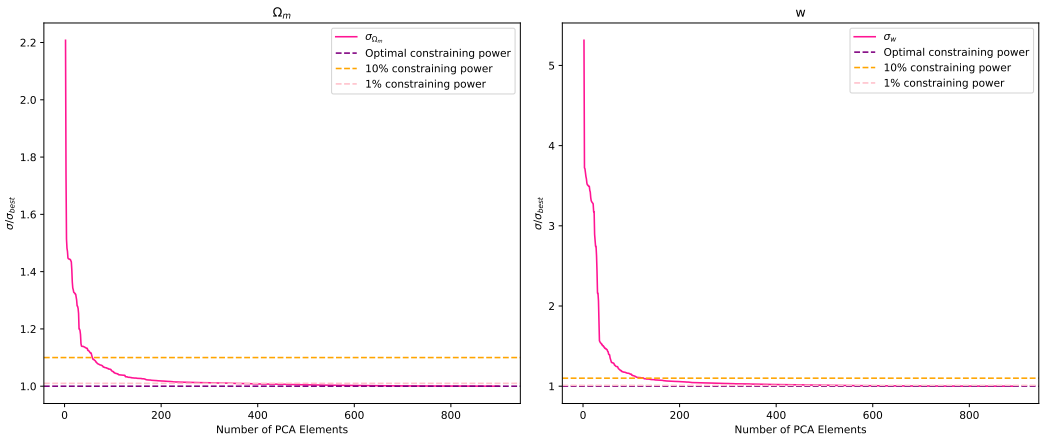


Figure 3: The figure shows the error bars for Ω_m and w as a function of the number of PCA elements maintained. The horizontal lines represent the optimal constraining power (this is the case when no data compression is performed), the 1% and the 10% constraining power.

$$log\mathcal{L}(\theta|d) \propto -\frac{1}{2}[X_d - X_t(\theta)]^T C^{-1}[X_d - X_t(\theta)] \tag{3}$$

For the assignment, two reference model were given: one of them has some Gaussian noise in it. We were also given 4 covariance matrices, one is analytical and three are numerical, made with different numbers of realizations.

The 100 walkers are initialized with random positions inside emulator training range. Looking back, initializing the walkers in a tighter "ball" around the true cosmology as I did in the last parts of the assignment would have helped the convergence of the chains. In fact, I found that a few walkers, typically the ones that started further away from the true cosmology, got stuck in local minima. In this case, that is easy to do since the true cosmology that we want to recover is known. In general, if I didn't have any previous knowledge, I would run multiple chains as a burn-in phase, and then initialize the walkers near the point of maximum log probability found between all chains. This would ensure a faster convergence of the final chain. After a burn-in phase of 200 steps, the chains were run with 2000 steps. Since the autocorrelation length is ≈ 40 steps, this gives ≈ 5000 independent samples.

The most interesting results are the ones that show links with the 2 previous assignments. In particular in task 2 we saw directly how the covariance matrix, alongside the Hartlap correction, has a very strong impact on the constraints. In total, I run 6 different chains: one for each covariance matrix (with 1500, 3000, and 10000 realizations) and with/without Hartlap correction. The constraints are shown in Figure 4 and 5.

The only case in which the MCMC is able to recover the true cosmology within 1σ is for the $10k$ realization covariance, when the Hartlap correction factor is applied to its inverse. This is in perfect agreement with what we learned in assignment 1.

In task 3, we explored the impact of PCA data compression on the posterior distributions. I run chains compressing the data by different amounts (aka keeping different numbers of PCA elements). As observed in assignment 2, the more elements are kept, the tighter the error bars. This statement holds true when using both the analytical and numerical covariance matrices, as can be seen in Figures 6 and 7. When keeping more than 500 elements, the error bars are basically the same: this suggests that most of the information in the data has been captured. Despite difficult to see from Figure 7, the constraints for 500 PCA elements are tighter than the ones for 900. This interesting behaviour is given by an interplay between the effect of information compression and the Hartlap factor, which depends on the number of PCA elements. In particular, the information content increases with the number of PCA elements, while the Hartlap factor has the following behaviour:

$$h = \frac{N - N_{pca} - 2}{N - 1} \tag{4}$$

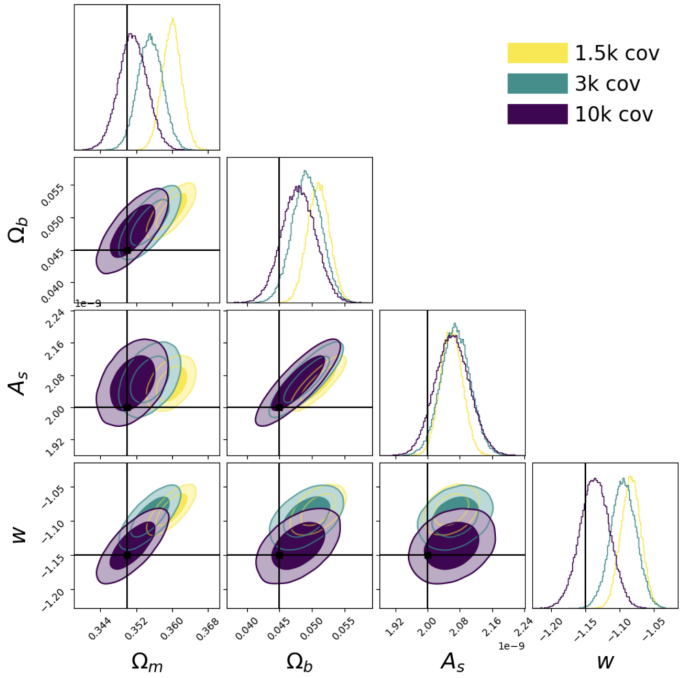


Figure 4: Constraints for the cosmological parameters obtained with different covariance matrices, whose inverses were not Hartlap corrected.

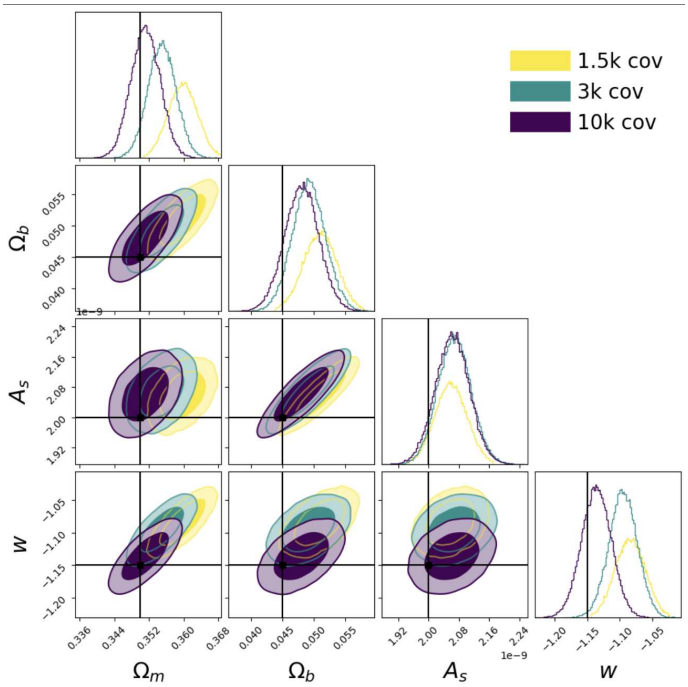


Figure 5: Constraints for the cosmological parameters obtained with different covariance matrices, whose inverses were Hartlap corrected.

The dependence on the number of PCA elements is stronger when using the $1.5k$ covariance, that is, when $N = 1500$. This is the case where I could observe this behaviour.

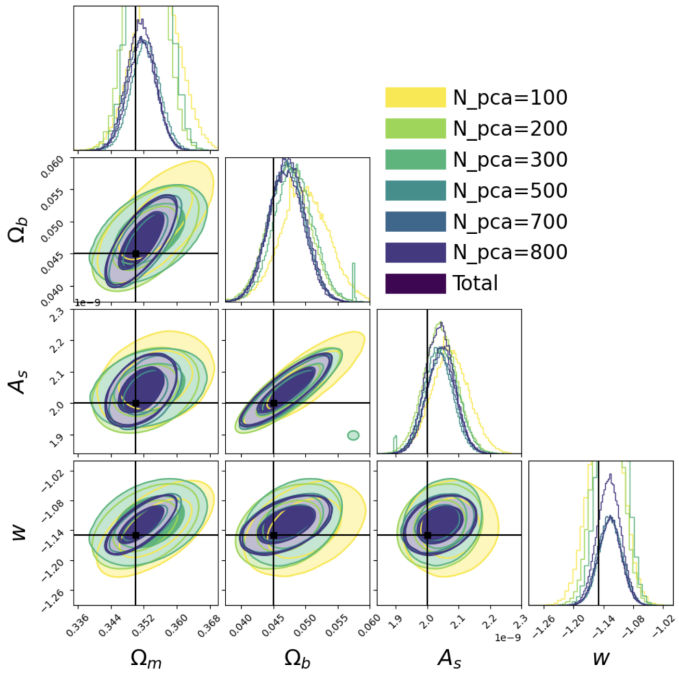


Figure 6: Constraints for the cosmological parameters obtained with the analytical covariance matrix, for different numbers of PCA elements.

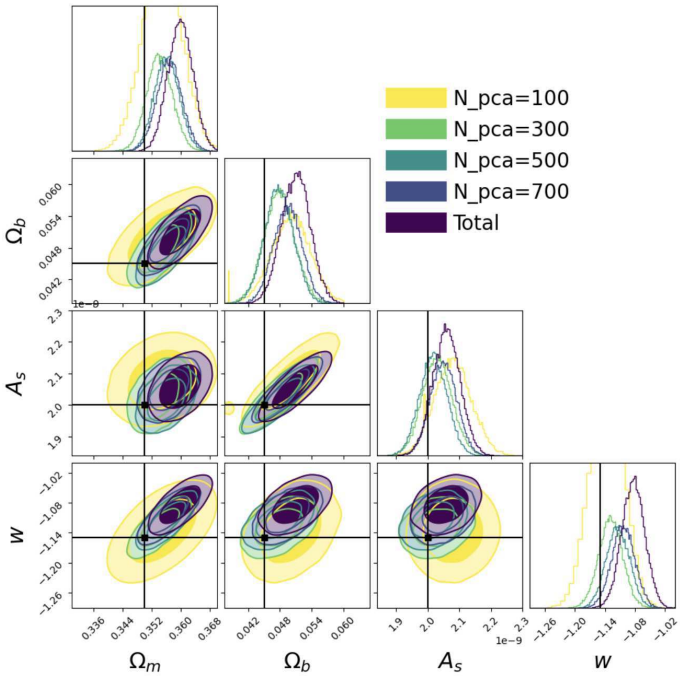


Figure 7: Constraints for the cosmological parameters obtained with the $1.5k$ numerical covariance matrix, for different numbers of PCA elements. Because of the covariance matrix employed, the estimates are biased.

As an alternative to the `emcee` sampler, I used `zeus`, which is convenient because it has the same API as `emcee` but a different underlying algorithm, the ensemble slice sampling. In `emcee`, proposals for walker positions are based on the positions of other walkers in the ensemble, ensuring efficiency. The move performed

(called "stretch move") has the property of being affine invariant. This ensures that it performs equally well under all linear transformations and is not sensitive to covariances between parameters, making it very efficient.

`zeus`, on the other hand, uses the technique of ensemble slice sampling. Slice sampling is based on the idea that sampling from a distribution with density $P(x)$ is equivalent to uniform sampling from the area under the plot of $f(x) \propto P(x)$. To this end, we introduce an auxiliary variable y, called height. To sample from the marginal distribution $P(x)$, we first sample from $P(x, y)$ and then we marginalise by dropping the y value of each sample. The idea for the `zeus` algorithm is to move the walker X_k based on two randomly chosen walkers X_l and X_m, whose position define the direction vector. Then, the new sample is obtained with the slice sampling method previously described. This "differential move" is also affine invariant and naturally exploits correlations between parameters for a more efficient sampling.

As expected, the two samplers give consistent results as shown in Figure 8. The mean autocorrelation time for `zeus` is, however, of ≈ 7 steps, while it was ≈ 40 for `emcee`: `zeus` seems to be more effective in generating independent samples, which is why I run that chain with a fewer number of steps. In this case, initializing the walkers with more localized initial positions is very beneficial in reducing the burn-in phase.

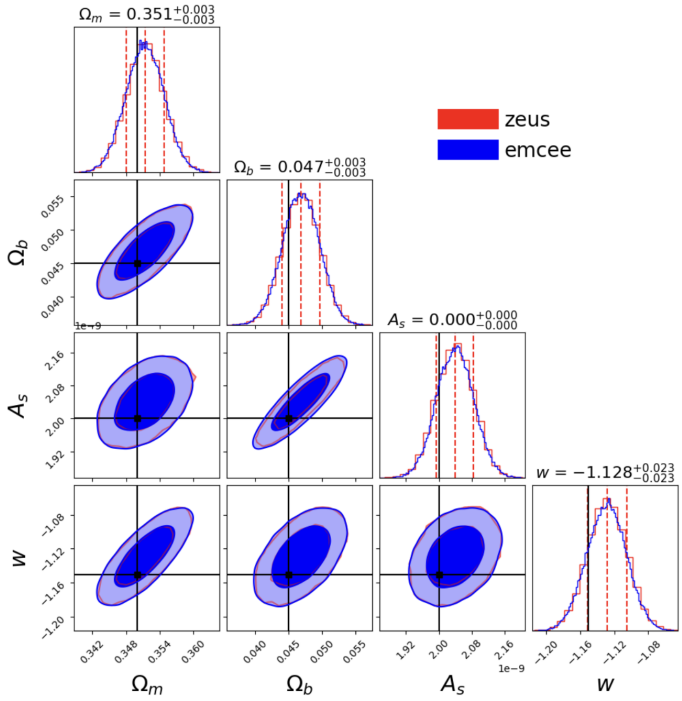


Figure 8: Comparison of the `emcee` and `zeus` ensemble samplers. The analytical covariance matrix and the noisy reference model were used in constructing the likelihood.

Familiarizing myself with the MCMC pipeline is something that will be very helpful for my future work: at the moment I am working on developing a fast and efficient algorithm to measure 3×2 point statistic from data. Once the code is ready, I will apply it to Euclid data, with the goal to constrain cosmological parameters through a likelihood analysis. In conclusion, gaining knowledge about the importance of the covariance matrix, learning more about the effects of data compression, and exploring different sampling algorithms will be very helpful for my research in the near future.