From Covariance to Compression: Astrostats is Cool!!

Darshak Patel | PHYS 788

Outline

1.0 Covariance, χ^2 Distributions and Hartlap Factor

Assignment 1

2.0 Neural Network/Emulator & PCA

Assignment 2

3.0 MCMC + PCA + Covariance

Assignment 3

Assignment #1

- Given the true covariance matrix of a reference model, from which we generate two sets of 10,000 noisy Gaussian data vectors and calculate the chi2 values and plotted the distribution
  - Need to convince yourself that these were Chi-2 distributions = to do this we look at the mean and variance
- Generated four sets of numerical covariance matrices from one of the noisy data set
  - Compared the correlation matrices of the num cov matrices
  - Tested to see if the covariances were positive semi-definite
- Calculated the chi2 distribution once again
- Applied Hartlap factor and redid the chi2 distributions
- Chi2 against the same dataset that made the num cov

What’s a covariance?

In the 2D case:

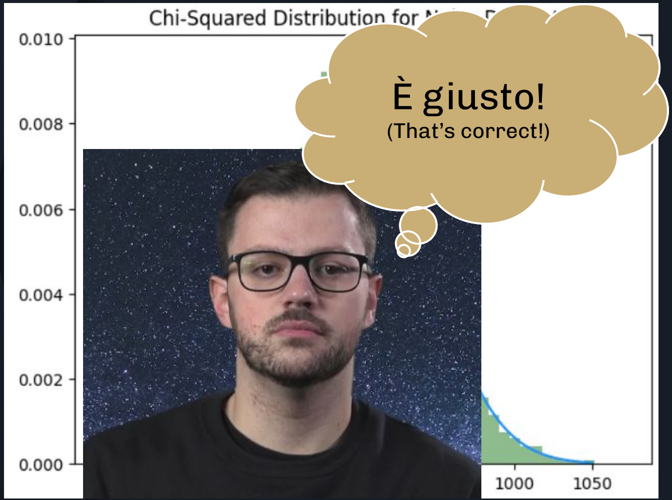
$$Cov(X,Y) = \begin{pmatrix} Var(X) & Var(Y,X) \\ Var(X,Y) & Var(Y) \end{pmatrix}$$

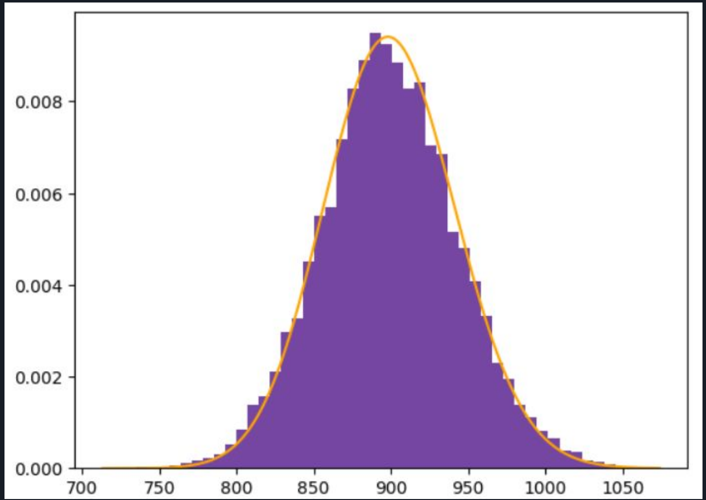
$$Var(X,Y) \approx \frac{1}{n-1}\sum_{i}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

What power do covariance matrices have statistically?

- Obtain errors for your data and analysis
- Insight on underlying correlation between variables
  - Example of where knowing the correlation between variables is important?!

Covariance to χ^2 Distributions to Hartlap





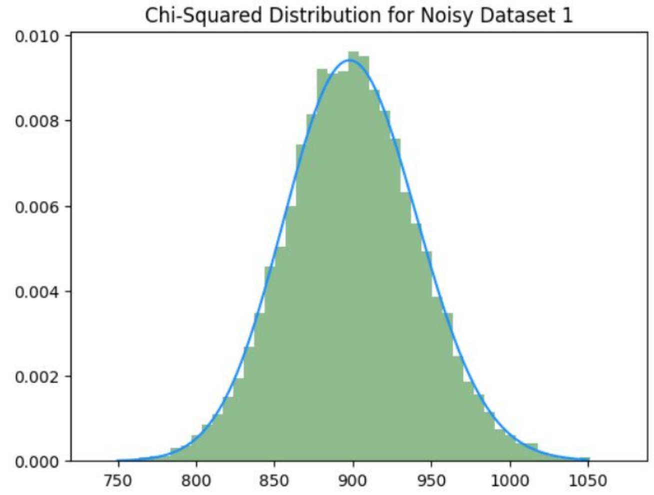
Mean = k | Variance = 2k

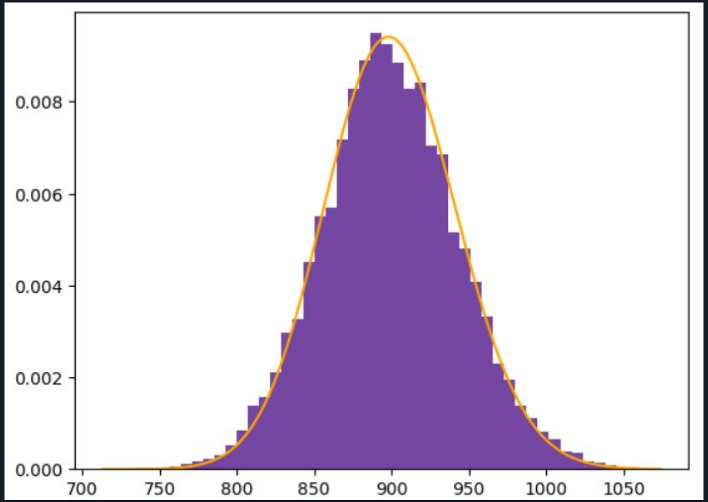
Mean = 899.96 | Variance = 1795.49 ✅

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

Hartlap Factor

Covariance to χ^2 Distributions to Hartlap





Mean = k | Variance = 2k

Mean = 899.96 | Variance = 1795.49 ✅

+

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

Hartlap Factor

A1 - Key Takeaways

1. Check that your covariance matrices are invertible and positive semi-definite
2. Always apply the Hartlap factor when using numerical covariance matrices

Covariance to χ^2 Distributions

$$\chi^2 = D^T C^{-1} D$$

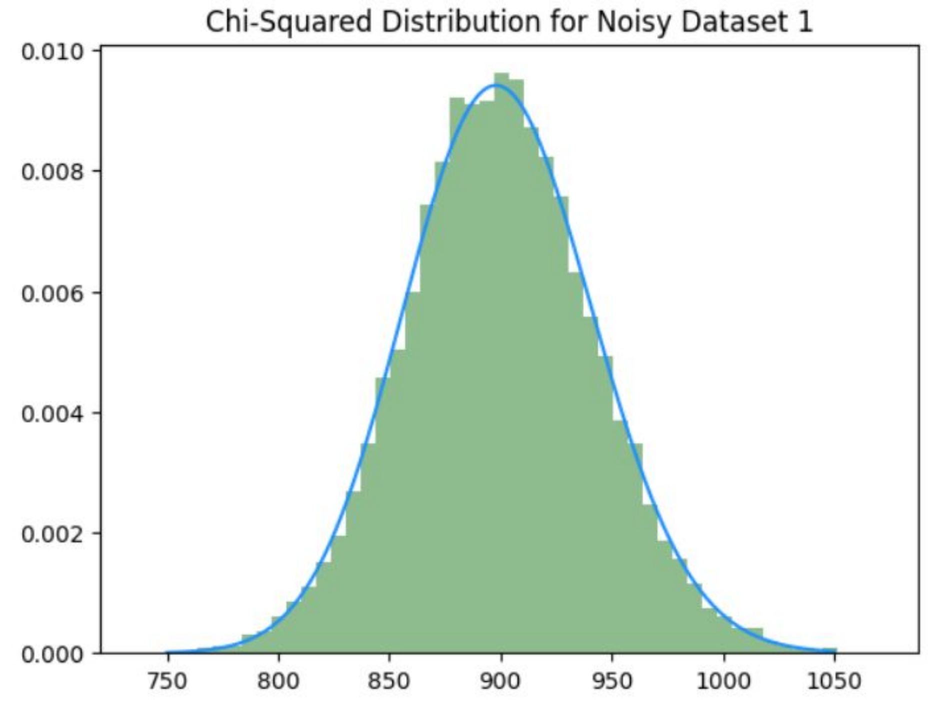
At high degrees of freedom (k) the χ^2 distribution tends to become Gaussian:

Mean = k | Variance = 2k

For noisy dataset #1:
Mean = 899.96 | Variance = 1795.49 ✅

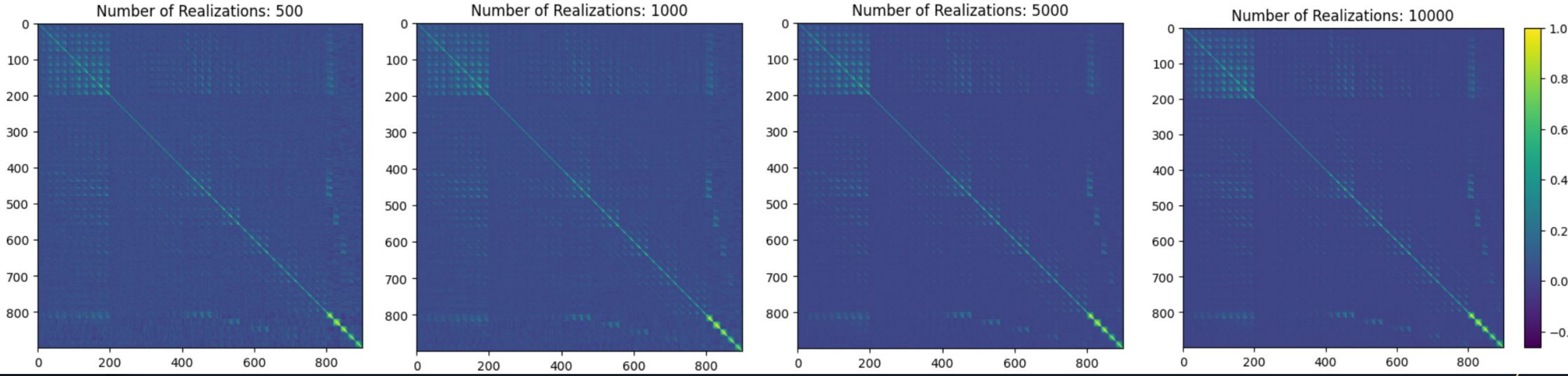
For noisy dataset #2:
Mean = 900.38 | Variance = 1800.64 ✅

Note: not fitting to data so dof = n_{dim}

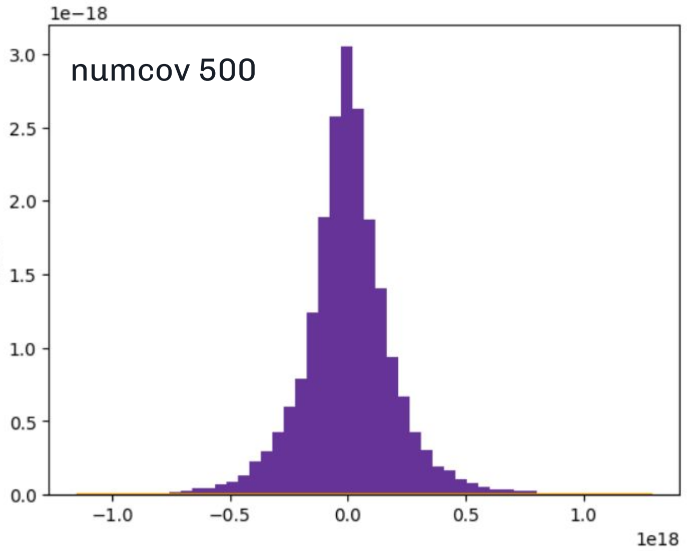


Numerical Covariance Matrix from Simulations

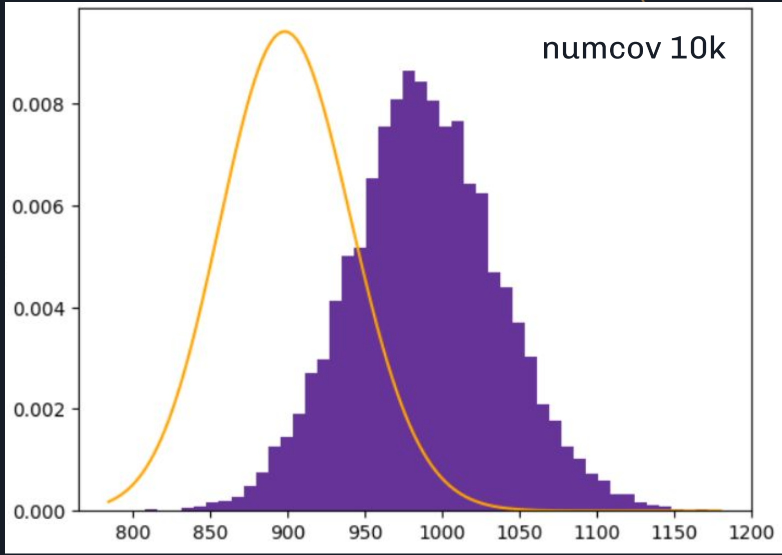
- Numerical covariance matrices: 500, 1000, 5000, and 10,000
- Calculated the correlation matrix
- Checked if positive-semi definite = no negative eigenvalues



χ^2 Distribution: Something is off...



Mean: 3.74e15
Variance: 3.92e34



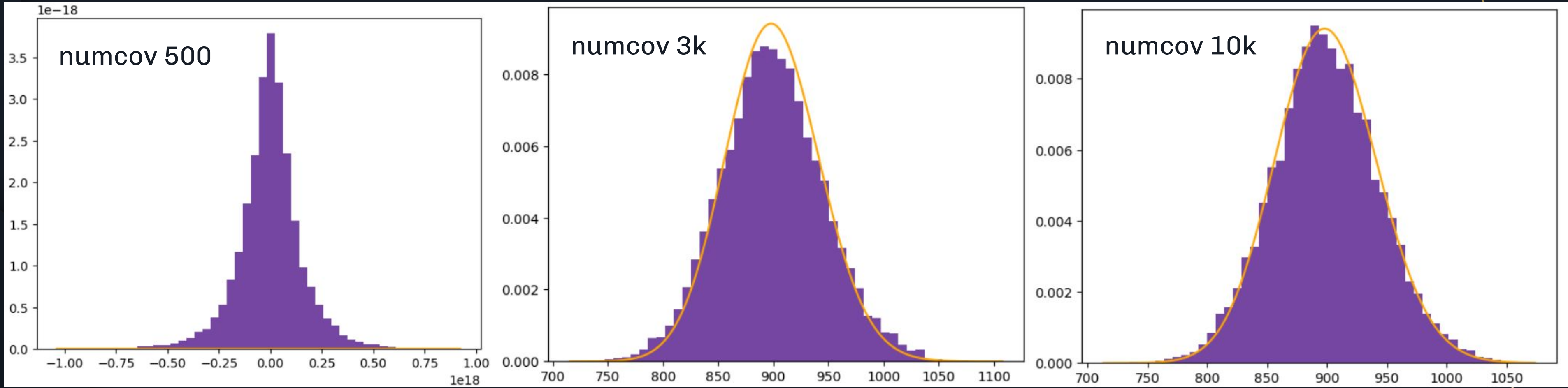
Mean: 988.54
Variance: 2372.37

The numerical covariance matrix itself is a random variable. We need to debias the inverse covariance matrix with the Hartlap factor!

χ^2 Distribution + Hartlap Factor

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

$$\tilde{C}^{-1} = \frac{C^{-1}}{h}$$



Mean: -3.02e16
Variance: 2.54e34

Mean: 899.52
Variance: 2192.16

Mean: 899.46
Variance: 1964.09

A1 - Key Takeaways

1. Check that your covariance matrices are invertible and positive semi-definite
2. Always apply the Hartlap factor when using numerical covariance matrices

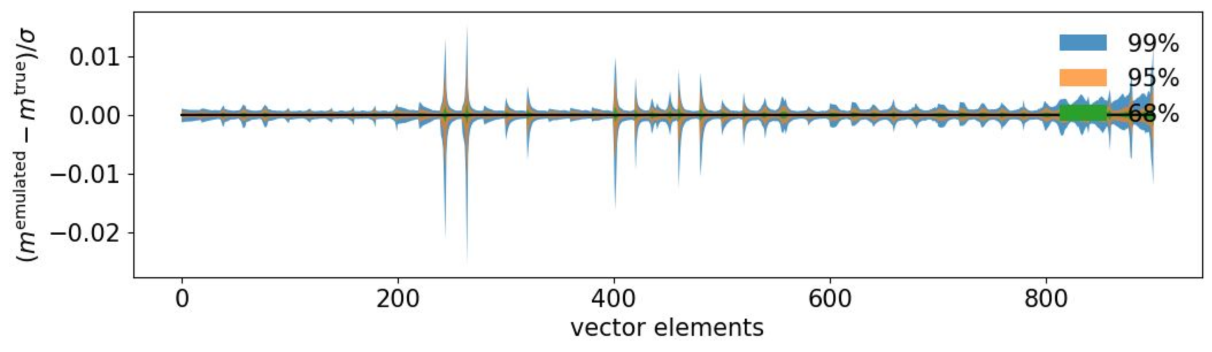
Assignment #2

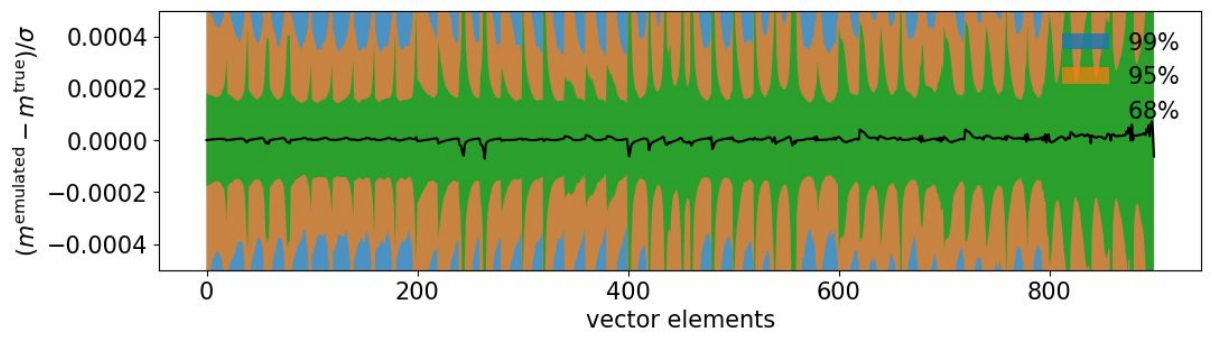
- Training Emulator
  - Normalizing the training features vs. not normalizing
  - Accuracy plots
  - What do the hyper parameters do? How did I choose them?
- Fisher Analysis to get max parameter constraints
- PCA data compression
  - How many PCA to get 10% and 1% max constraining power

Training a Neural Network Emulator

- Step #1 - Split your data into a training and testing sample (70% | 30%)
- Step #2 - Manipulate training features (Normalize? Rescale by σ?)
- Step #3 - Choose hyper parameters
  - Learning Rate, Batch Size, Max Epochs, Number of Neurons
- Step #4 - Train Emulator and check accuracy
- Step #5 - Repeat #3 and #4 till desired accuracy is achieved

My Best Trained Emulator





The Parameters:

- 5 layers
- LRs: 1e-2, 1e-3, 1e-4, 1e-5, 1e-6
- Batch Size: 500
- Normalized Train features

Loss: 7.296e-5

A2 - Key Takeaways

1. Hyper parameter choices must be well-thought out
2. Normalizing training features improves emulator accuracy
3. PCA data compression is a powerful tool (extra slides)

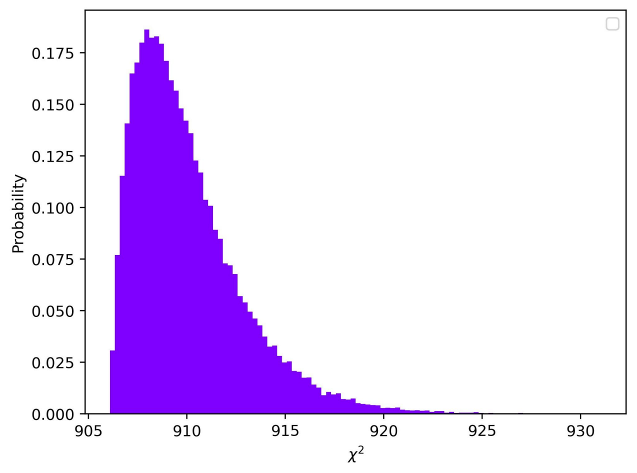
Assignment #3

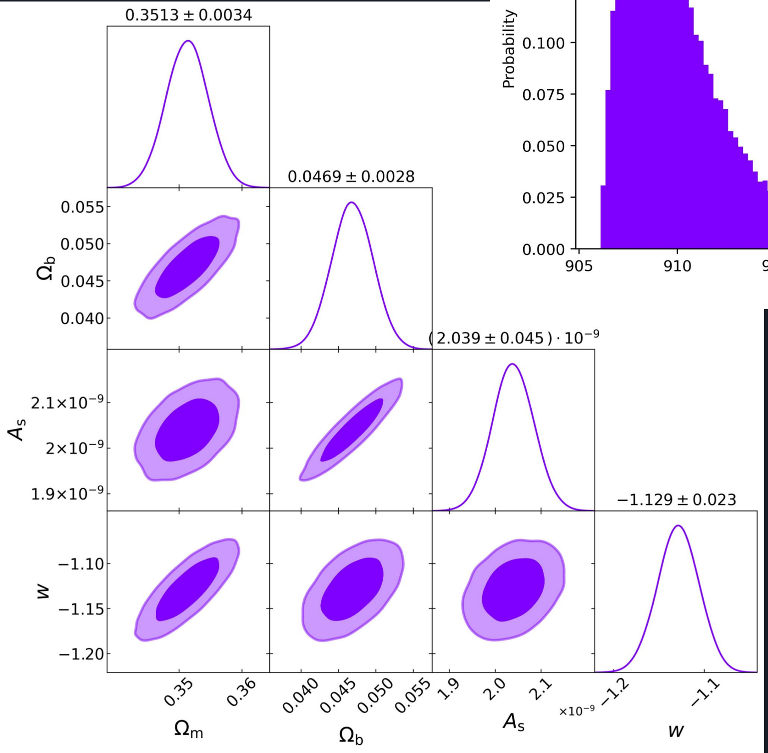
- Running MCMC
  - Hyperparameters and why I choose what I did
    - Like how I used lower params while working on it but then went to higher higher walkers and steps when I ran them overnight
  - What does a MCMC do? How does it work?
  - Example Posterior distribution from task 1
  - Example chi2 distribution
- Using Num Cov and how Hartlap effects the posterior distributions
- Effects of PCA analysis on the posterior distributions
- Noise-free model vs. Noisy-model on MCMC

Monte-Carlo Markov Chain (MCMC)

- MCMC via. Emcee to find parameter constraints for:
  Ω_m, Ω_b, As, w

- Hyper parameters:
  - Total steps
  - Burning steps
  - Number of walkers



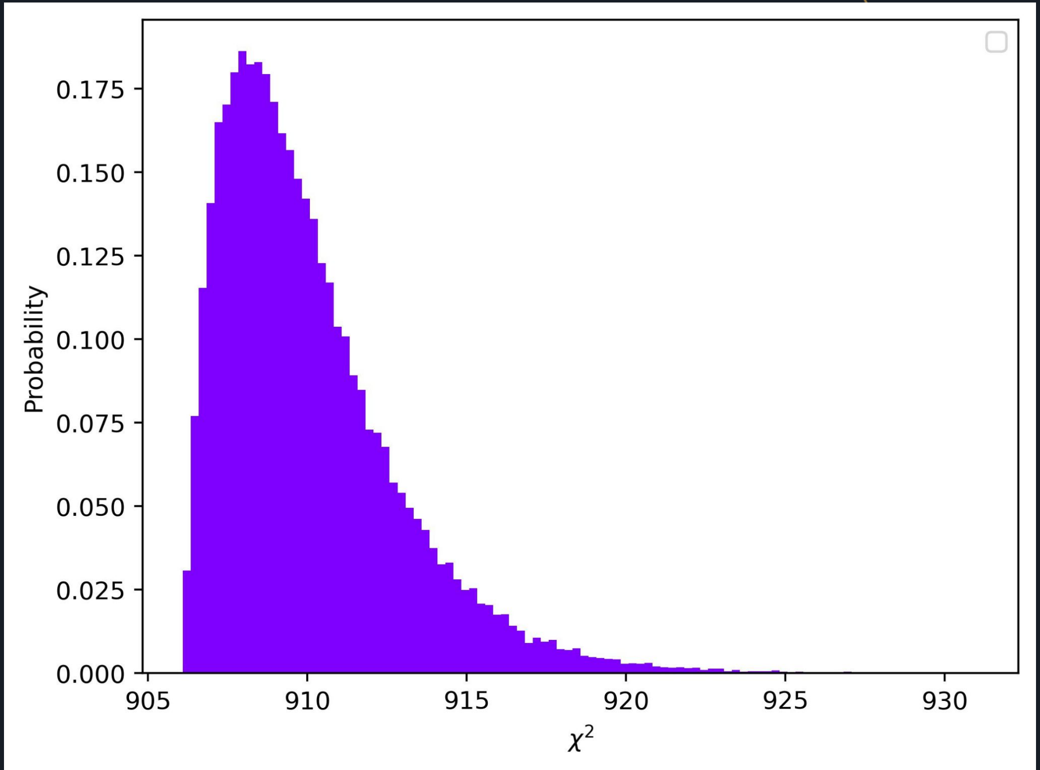


Monte-Carlo Markov Chain (MCMC)

- Probabilistic based sampling technique useful for complex distributions, or high dimensionality

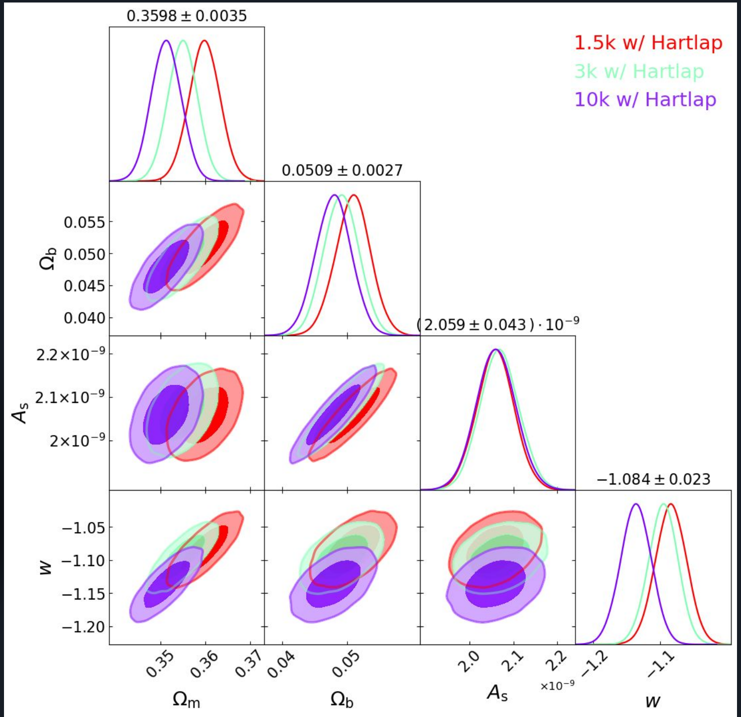
- MCMC via. Cosmopower to find parameter constraints for: Ω_m, Ω_b, As, w

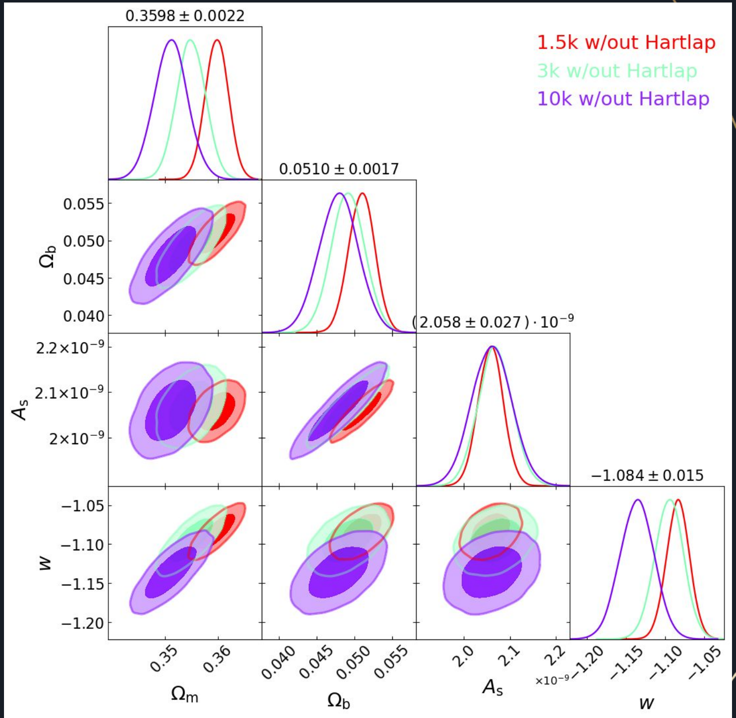
- Hyper parameters:
  - Total steps
  - Burning steps
  - Number of walkers



MCMC: Numerical Covariance + Hartlap Factor

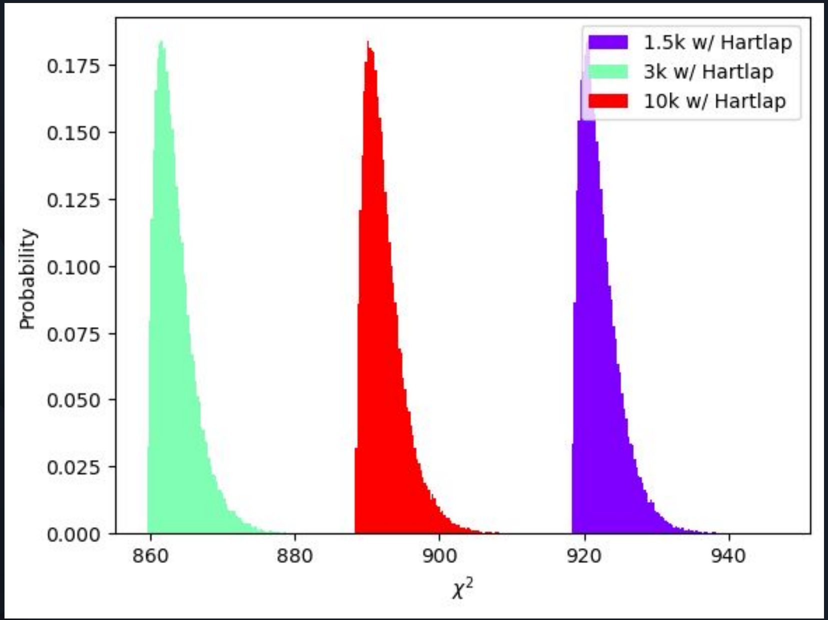
- MCMC posteriors and χ^2 distribution: 1.5k, 3k and 10k numerical covariance matrix

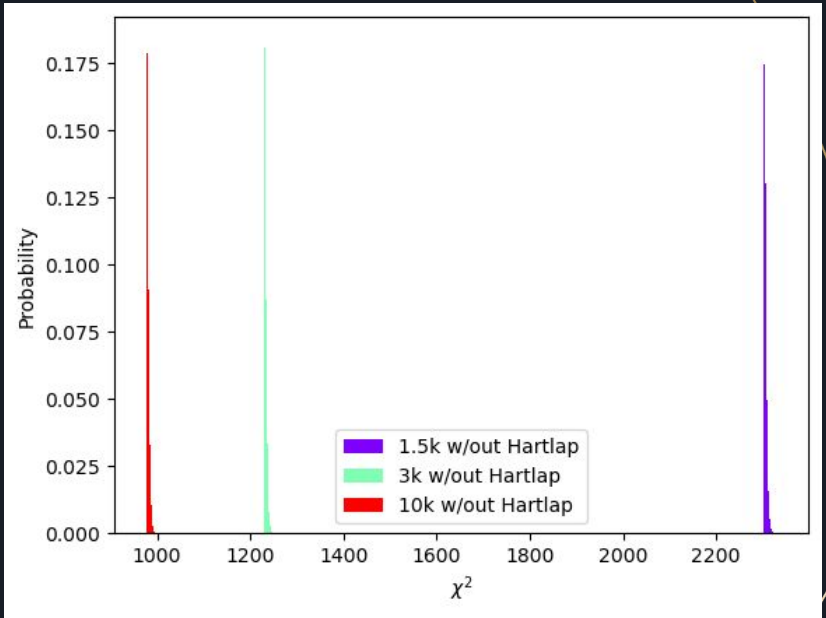




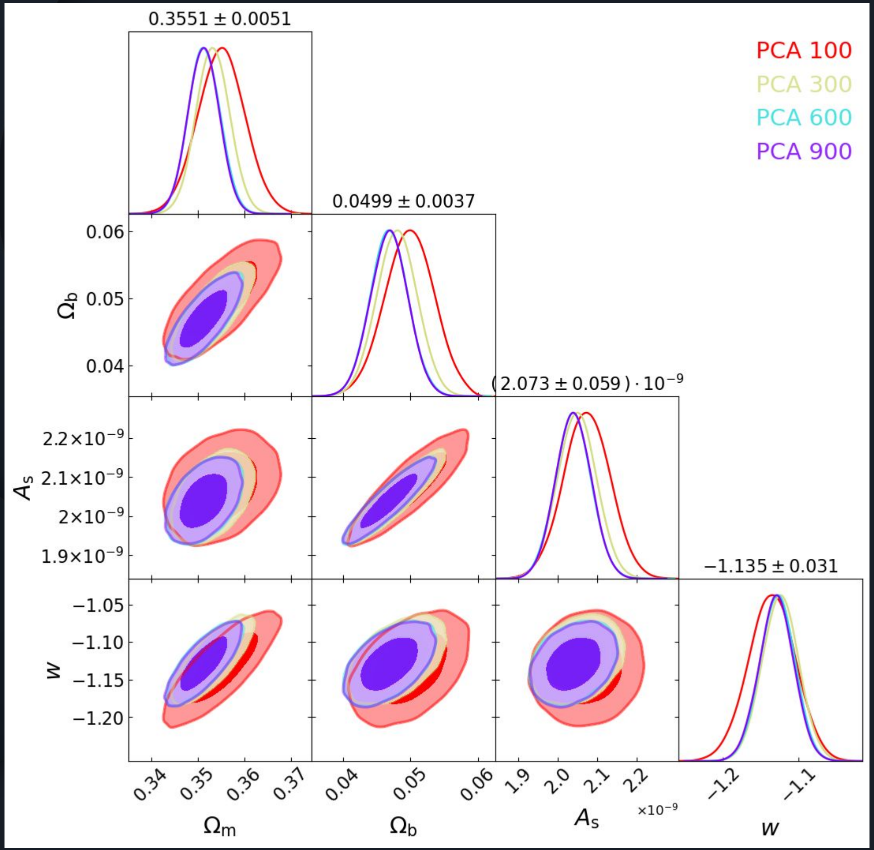
MCMC: Numerical Covariance + Hartlap Factor

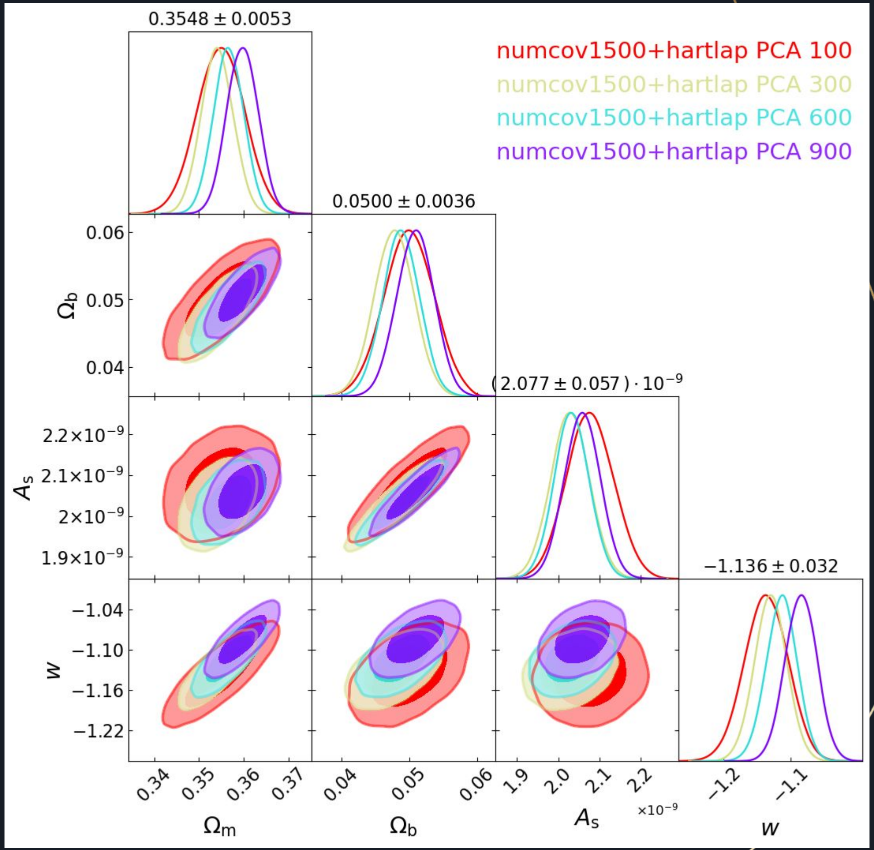
- MCMC posteriors and χ^2 distribution: 1.5k, 3k and 10k numerical covariance matrix





MCMC: PCA + Analytical/Numerical Cov





A3 - Key Takeaways

1. MCMC’s are a (for the most part) very efficient way of exploring a larger parameter space and determine parameters of best-fit

2. Hartlap factor corrections and PCA data compression together have opposing effects

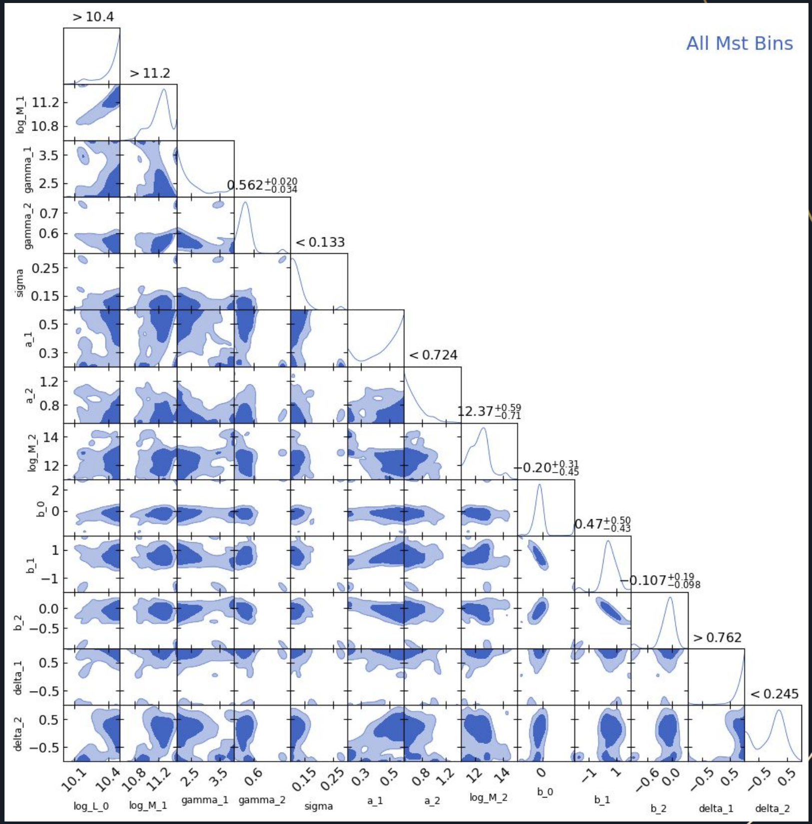
3. Noise within models play a significant role in the accuracy of MCMCs (extra)

Application to my Research:

- Covariance matrices from Jackknife

- Training an Emulator for dsigma

- MCMC to find parameters of best fit for HOD model



Thank you

Backup Slides

All Key Takeaways:

Assignment #1:

- Check that your covariance matrices are invertible and positive semi-definite
- Always apply the Hartlap factor when using numerical covariance matrices
- Do not measure the Chi2-distribution on the data you used to make the numerical covariance matrices

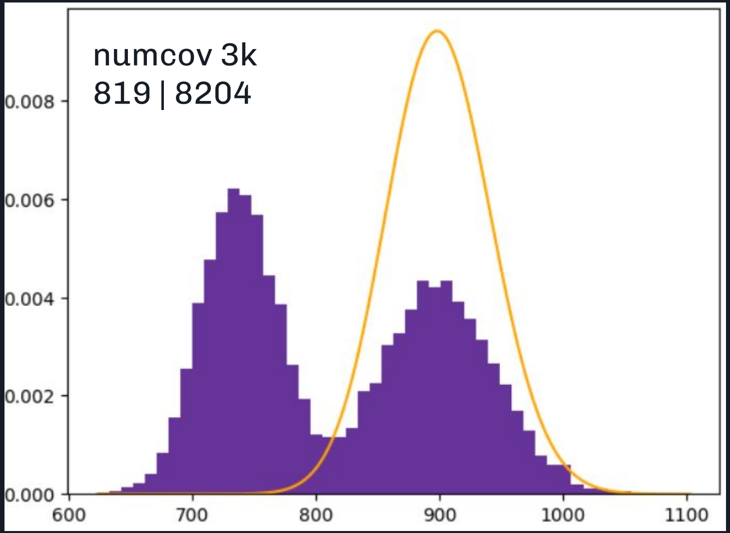
Assignment #2:

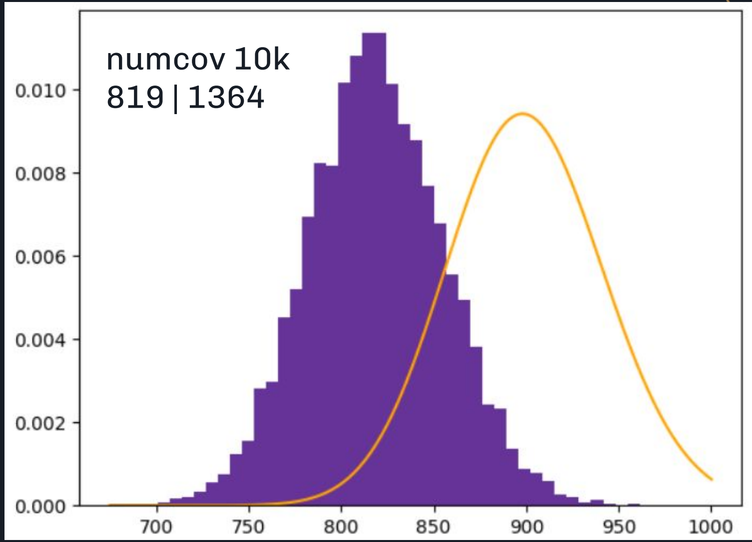
- Hyper parameter choices must be well-thought out
- Normalizing training features improves emulator accuracy
- PCA data compression is a powerful tool

Assignment #3:

- MCMC’s are a (for the most part) very efficient way of exploring a larger parameter space and determine parameters of best-fit
- Hartlap factor corrections and PCA data compression together have opposing effects
- Noise within models play a significant role in the accuracy of MCMCs

Measuring χ^2 against the same dataset as the Numerical Covariance





- The lower peak contains the data vectors used to make the numerical covariance
- If n is VERY high then we can get more reasonable means/variances (80k, 100k etc.)

The Hyper Parameters

Learning Rate:

- Determines how big of “jumps” the NN takes during each step
- Impacts the loss significantly

Batch Size:

- How many elements of the training set is thrown into the NN at once
- How many models you average over so too many and you wash out information, too few and you’re too sensitive
- Impacts the speed and loss

Max Epochs:

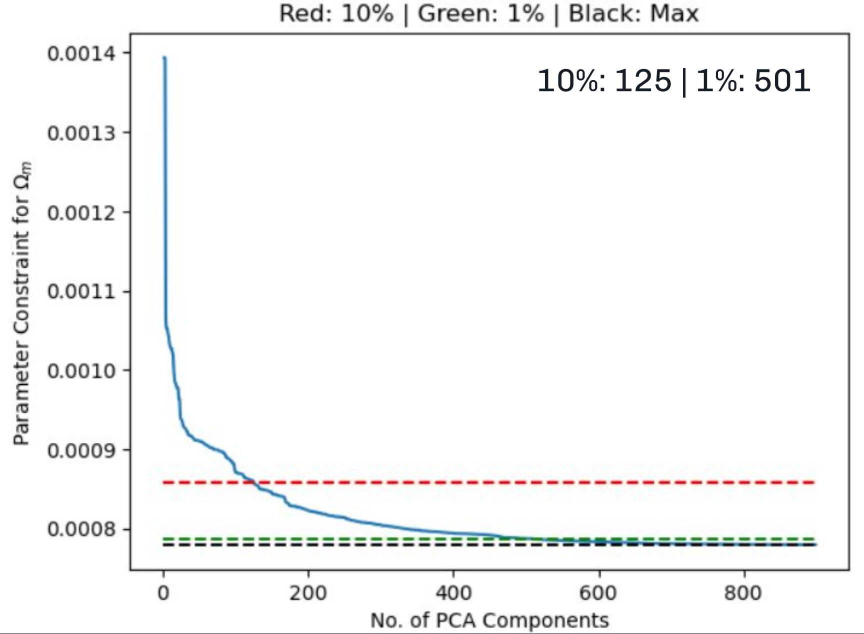
- Max number of epochs before NN moves to next learning rate
- Impacts the speed

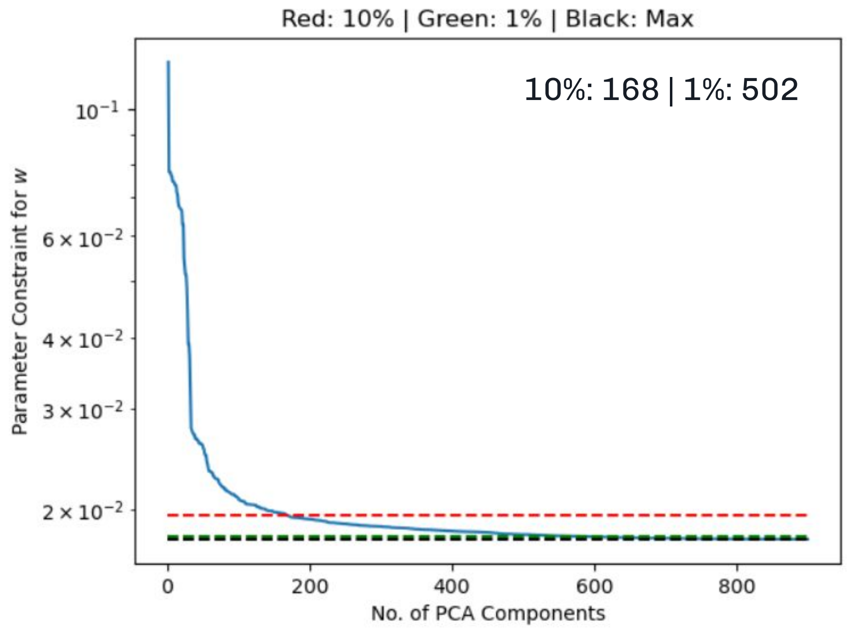
Number of Neurons:

- Main parameter which dictates the performance of the emulator
- Problem of overfitting or underfitting usually originates here!

A2: Principal Component Analysis (PCA)

- Form of data compression - decreasing the dimensionality of the original data, but preserving the same information/features





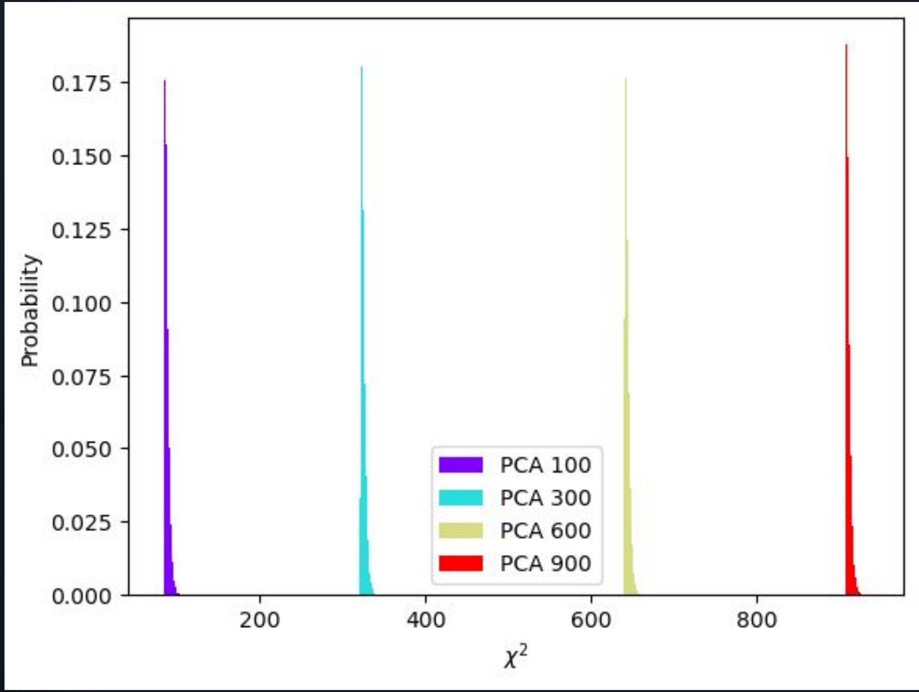
Parameter Covariance via. Fisher Matrix

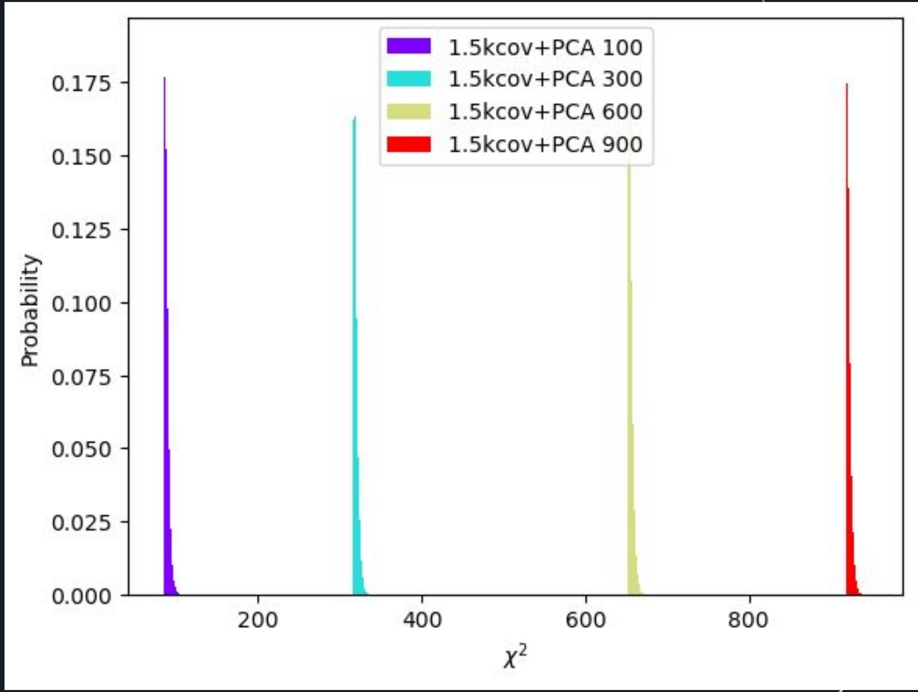
$$F_{ij} = \left(\frac{\partial m(\Theta)}{d\Theta_i} \right)^T C^{-1} \left(\frac{\partial m(\Theta)}{d\Theta_i} \right)$$

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

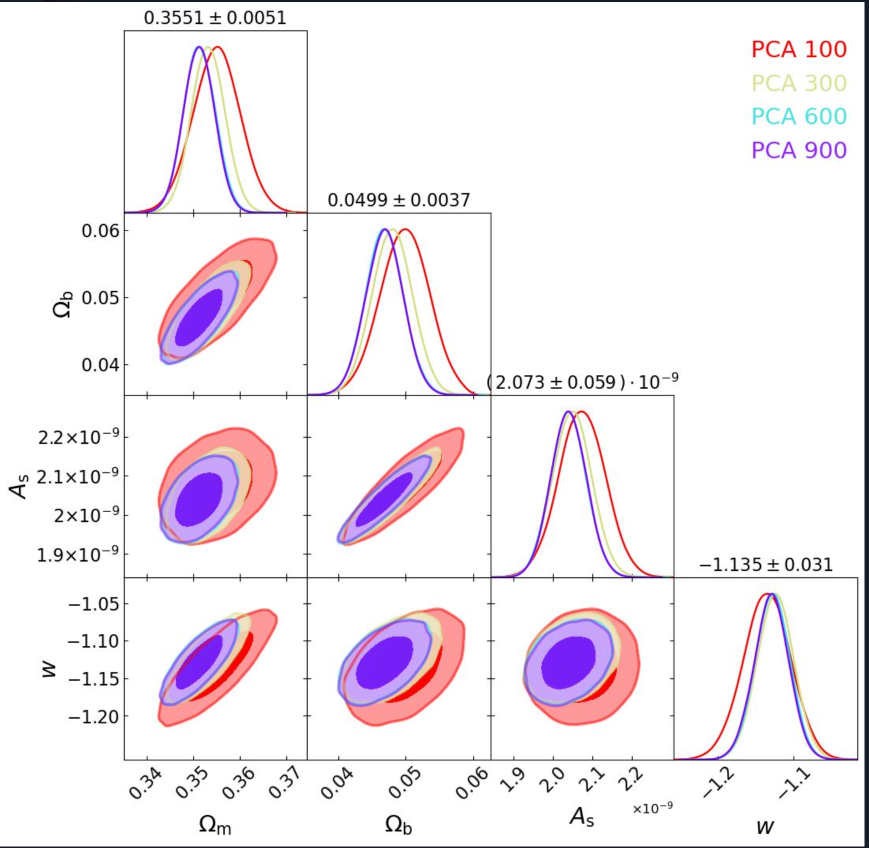
- Parameter covariance = inverse of Fisher matrix
- Max constraints is the square-root of the diagonal terms of the parameter covariance matrix

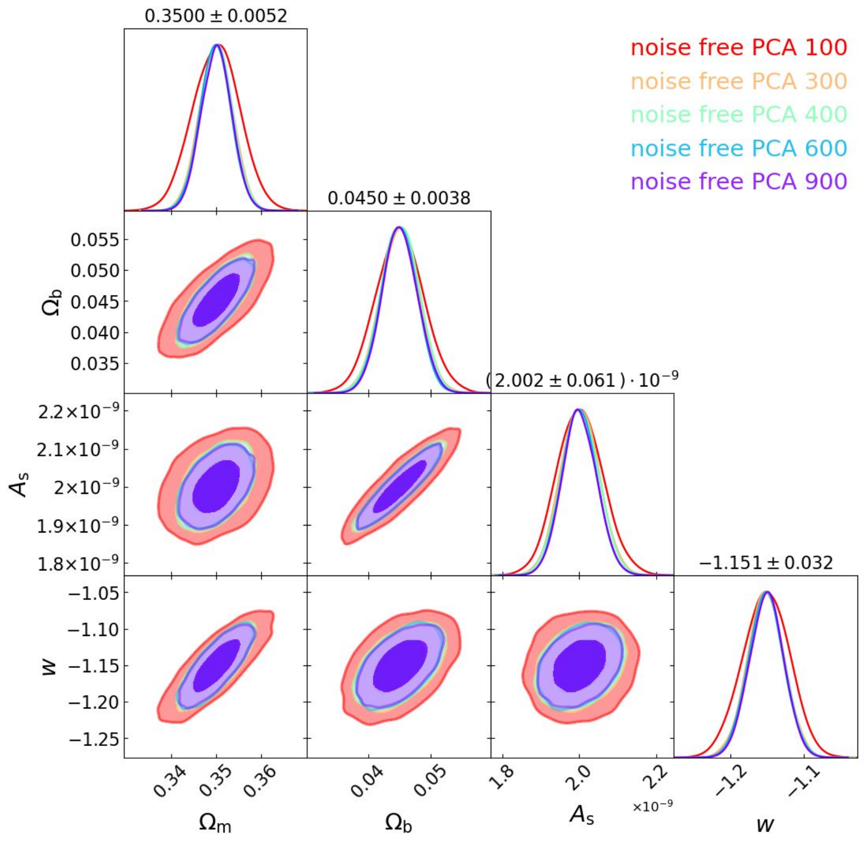
MCMC: PCA + Analytical/Numerical Cov



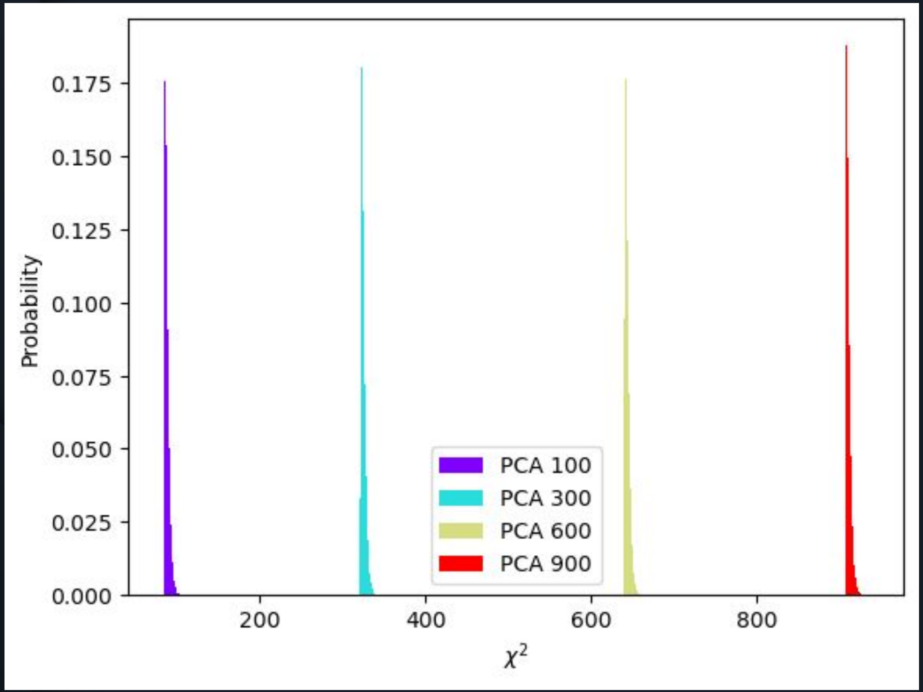


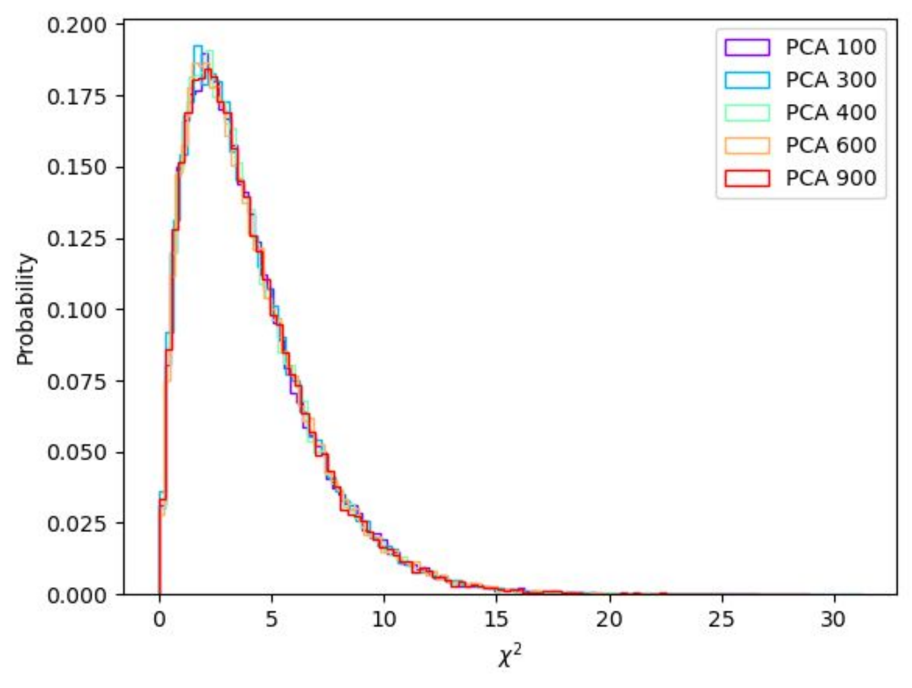
MCMC: Noise-Free vs. Noisy Model + PCA



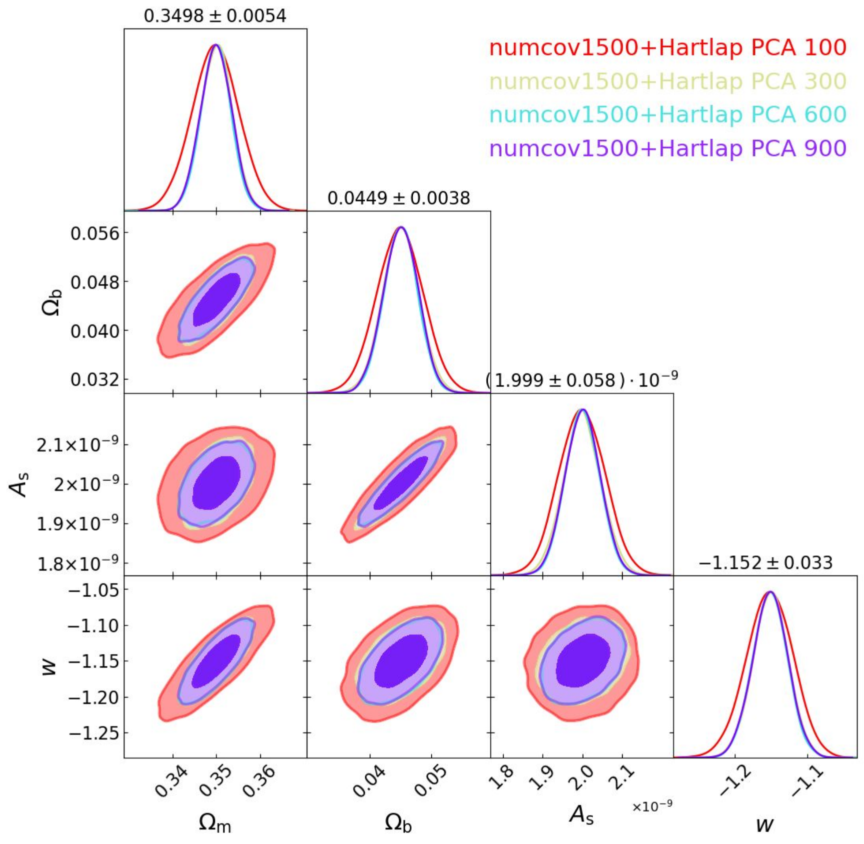


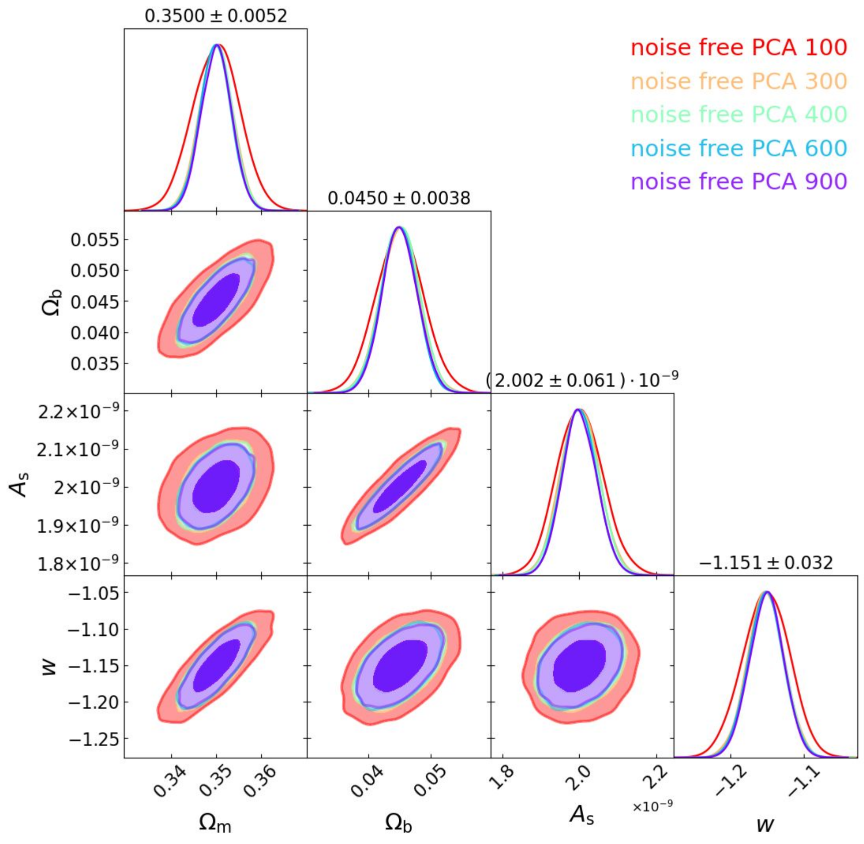
MCMC: Noise-Free vs. Noisy Model + PCA



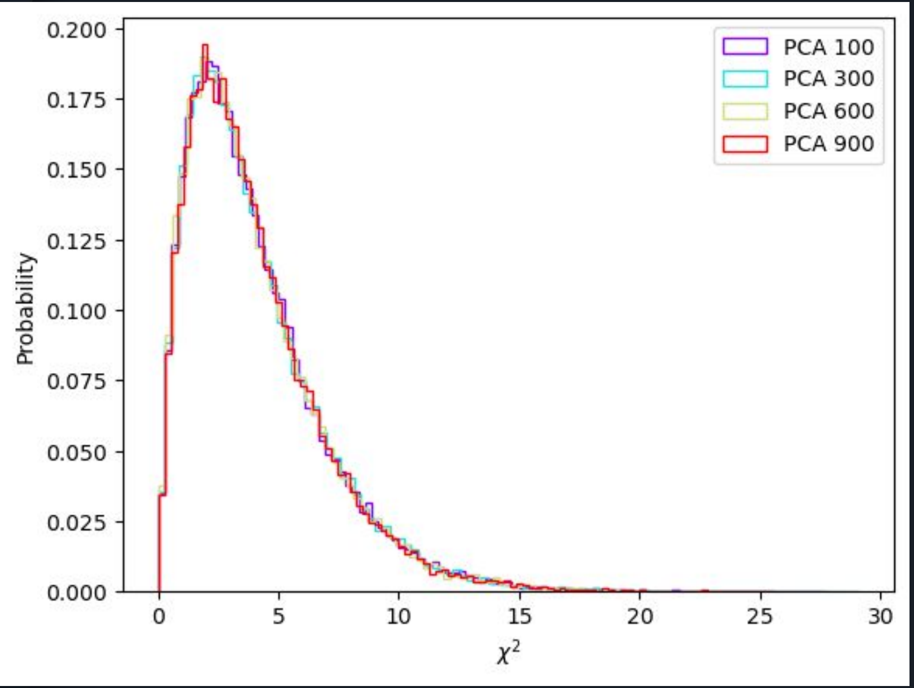


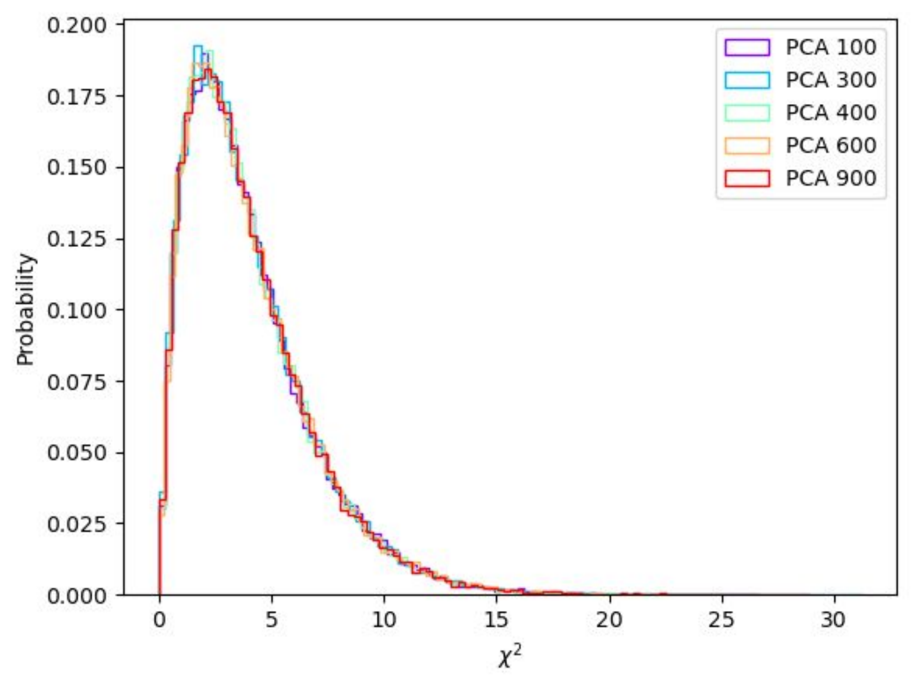
MCMC: Noise-Free + PCA + Analytic/Num Cov



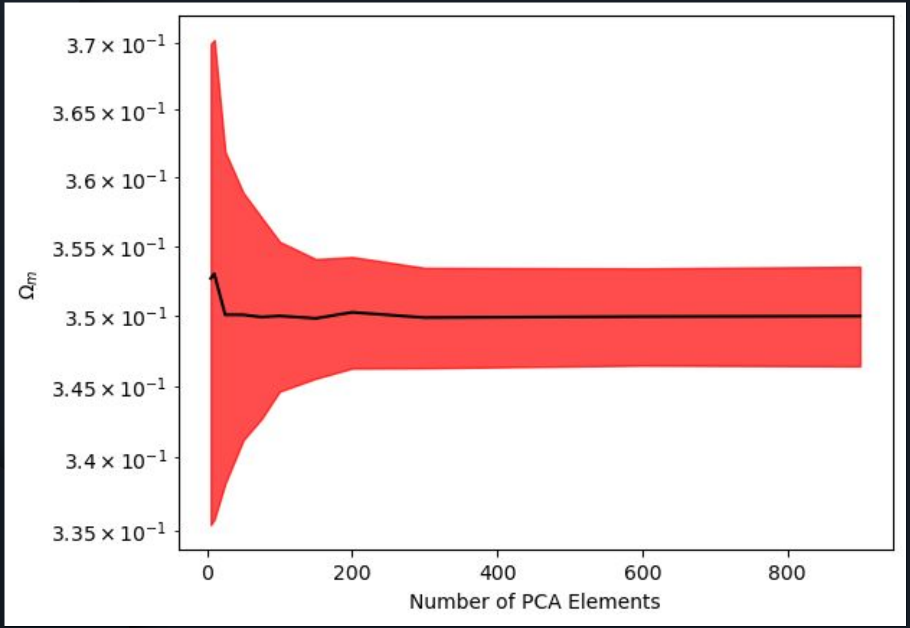


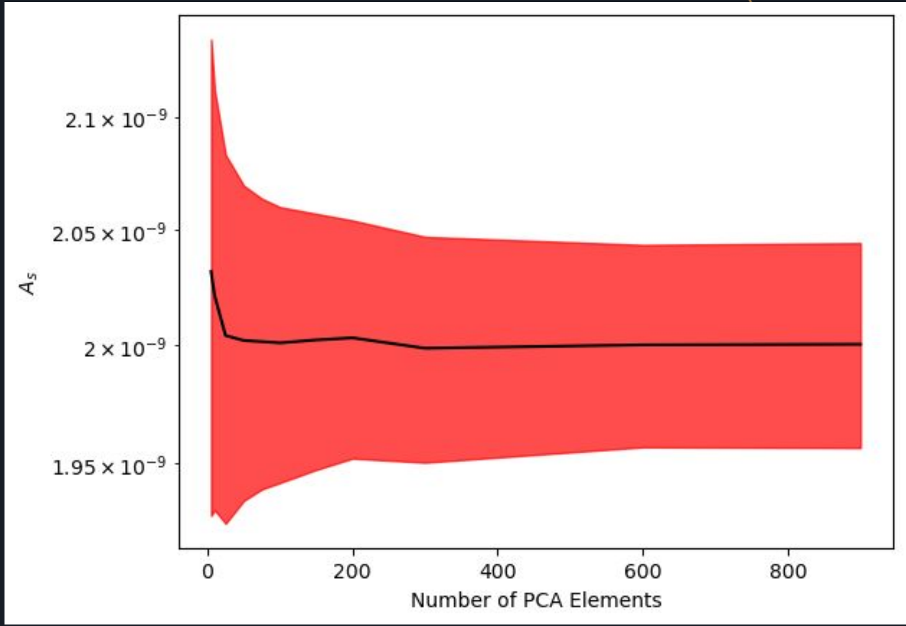
MCMC: Noise-Free + PCA + Analytic/Num Cov





A3: Marginalized Constraints on Ω_m and As





Hard to tell but the constraints slightly increasing after hitting a minimum ~550-600. This minimum is the “sweet spot” where Hartlap and PCA balance out.

Why should we normalize?

- Flattens out the data so that all features are equally learnt by the emulator

Let me draw you a picture!

Data Compression Alternatives:

Method:	Pros:	Cons:
Principal Component Analysis	• Deals with multicollinearity • Easier to implement • Easier to interpret	• Struggles to capture/preserve nonlinear relationships • Sensitive to outliers
Massively Optimised Parameter Estimation and Data compression (MOPED)	• Handles non-linearity better • Significantly less data loss	• More computationally expensive • More complicated to implement

MCMC Alternatives:

Running the MCMC: Emcee vs. Tensorflow vs. OTHER!

Algorithms:

- Metropolis-Hastings
  - Emcee using a modified version of this
- Gibbs Sampling
- Hamiltonian
- etc...

Assignment #1:

- Numerical Stability of the Covariance Matrix
  - Why use/care about covariance matrices?
  - What power do they have statistically?
  - Equation for getting covariance from a data set of 2
- Chi2 Distributions + Hartlap Factor
  - Quick sentence on what chi2 values and its distribution tells us
  - Equation for Hartlap Factor
  - Example plots of Chi2 distribution with and with Hartlap Factor
  - What does the Hartlap Factor do

Assignment #2:

- Training a Neural Network emulator
  - Show the effects of normalizing vs. not normalizing the training features
- PCA data compression
  - What exactly it does? And Why we should care?
  - Show how the data compressed cov matrices look
  - Show 10% and 1% constraint plots as proof

Assignment #3:

- What is a MCMC and how does it work?
- Result from task 1 of assignment
- Effect of Hartlap factor on posterior distribution
- PCA analysis with MCMC
- Noise-free vs. Noisy model with MCMC

Backup Slides:

- Computations of Covariance Matrices
  - Numerical Cov - Pros and Cons
- Singular Value Decomposition
- Test for Gaussianity
- Types of Monte Carlo Sampling: Importance, Rejection
- Gelman-Rubin Statistic
- Types of Priors: Conjugat, Imprope, Jeffreys