Parameter Estimation of Global 21cm Signal

Aryana Haghjoo

Department of Physics

McGill University

Montréal, Québec, Canada

August 15, 2023

A thesis presented for the degree of Masters of Physics

Abstract

The global 21cm signal has been used to study the period between the end of the cosmic dark ages and the formation of the first stars and galaxies.

In this study, we explore the potential of this signal to reveal non-standard physics by means of providing a new path to test fundamental physical theories. The 21cm signal is sensitive to the density and temperature of neutral hydrogen in the early universe and the presence of the first stars and galaxies. Therefore, any deviation from the predictions of the standard cosmological model of this signal could indicate the presence of new physics beyond the standard model.

We adopt the Markov Chain Monte Carlo (MCMC) method combined with the Levenberg Marquardt (LM) algorithm to estimate the best-fit physical parameters (e.g., clumping factor, star formation efficiency) of the theoretical 21cm curves. We use the Accelerated Reionization Era Simulations (ARES) to generate models for the global 21-cm signal. Our method is flexible to the choice of parameters from ARES. The knowledge of these best-fit parameters will help us to constrain future proposed models and set

theoretical limits for the precision of upcoming experiments to observe non-standard effects.

Abrégé

Le signal global de 21 cm a été utilisé pour étudier la période entre la fin de l'âge sombre cosmique et la formation des premières étoiles et galaxies.

Dans cette étude, nous explorons le potentiel de ce signal pour révéler la physique non standard en fournissant une nouvelle voie pour tester les théories physiques fondamentales. Le signal de 21 cm est sensible à la densité et à la température de l'hydrogène neutre dans l'univers primordial et à la présence des premières étoiles et galaxies. Par conséquent, tout écart par rapport aux prédictions du modèle cosmologique standard de ce signal pourrait indiquer la présence d'une nouvelle physique au-delà du modèle standard.

Nous adoptons la méthode Markov Chain Monte Carlo (MCMC) combinée à l'algorithme de Levenberg Marquardt (LM) pour estimer les paramètres physiques les mieux adaptés (par exemple, facteur d'agrégation, efficacité de formation d'étoiles) des courbes théoriques de 21 cm. Nous utilisons les simulations d'ère de réionisation accélérée (ARES) pour générer des modèles pour le signal global de 21 cm. Notre méthode est flexible au choix des paramètres d'ARES. La connaissance de ces paramètres de meilleur ajustement nous aidera à contraindre

les futurs modèles proposés et à fixer des limites théoriques pour la précision des expériences à venir pour observer les effets non standard.

Acknowledgements

I thank my supervisors, Jonathan Sievers and Oscar Hernandez for their help throughout this project. I thank my dear parents for their ever-lasting support even from thousands of kilometers away. I thank all my new friends in Montreal, with whom I gained a ton of new experiences.

I also want to mention the great help of my therapist at McGill Wellness Hub, Romeo Bidar, whose constant presence was a miracle in my graduate experience journey.

I also do not want to forget the huge impact of help of my two favorite Persian singers, Kamran and Hooman, whose spotify playlist was the reason that I could keep debugging my code.

Last but not least, I want to thank me, for my perseverance and hard work which led to a lot of new experience and a wide insight.

To all the life long doers: you are changing the world every day

Contents

List of Figures

List of Tables

List of Acronyms

ARES	The Accelerated Reionization Era Simulations.
DLS	Damped Least-Squares.
EDGES	Experiment to Detect the Global EoR Signature.
EoR	Epoch of Reionization.
IGM	Intergalactic Medium.
LM	Levenberg-Marquardt.
MCMC	Markov Chain Monte Carlo.
SARAS	The Shaped Antenna to measure the background Radio Spectrum.
SFR	Star Formation Rate.

Chapter 1

Introduction

1.1 Background and motivation

1.2 Research questions and objectives

1.3 Overview of the thesis

Chapter 2

The Global 21cm Signal

2.1 Theoretical basis and physical principles of the Global 21cm Signal

2.2 Simulating the global 21cm signal

2.2.1 The Accelerated Reionization Era Simulations (ARES)

2.2.2 21cmFast

2.3 Effects of non-standard physics on the global 21cm signal

2.3.1 Cosmic Strings

2.3.2 Dark Matter

Chapter 3

Observations of the Global 21cm Signal

3.1 Experiment to Detect the Global EoR Signature (EDGES)

3.1.1 error bars and foreground removal

3.1.2 theoretical model and associated parameters

3.2 The Shaped Antenna to measure the background Radio Spectrum (SARAS)

Chapter 4

Parameter Estimation Methods

Reminder: The chapter preface still needs to improve. The reason for choosing MCMC rather than ML methods is not correct and require revision.

We briefly discussed the parameter estimation and model fitting of global 21cm in 2.4. Various computational algorithms, including machine learning and neural networks (cite relevant sources), have been used to accomplish this task. While neural networks have been shown to be effective, one drawback is the lack of clear physical interpretation of the resulting parameter values. In contrast, sampling algorithms like Markov chain Monte Carlo (MCMC), which preserve the basis vectors of the parameter space, can overcome this limitation. The parameters used in these methods are directly linked to the underlying physical mechanisms of the model.

In this chapter, we describe the main fitting algorithm used in our study, which combines

Levenberg-Marquardt and MCMC. We explain why we used both of these algorithms for this particular fitting challenge and provide a detailed explanation of how we combined them. We also discuss two different tests that we used to measure the quality of the proposed fit.

In chapter 5, we present the results of applying this fitting method to the observed global 21cm curve[1] and its corresponding theoretical model[2].

4.1 Levenberg-Marquardt (LM)

The Levenberg-Marquardt (LM) algorithm, also known as the damped least-squares (DLS) method, is a fitting algorithm used for non-linear least-squares problems. This iterative algorithm is based on another gradient decent method referred to as "Newton's method".

LM is perfectly capable of fitting models with Gaussian-shaped likelihood spaces. However, it's abilities are limited when it comes to more complicated surfaces.

We continue this chapter by deriving the basic analytical definition of this method. In order to do so, we start by defining the matrix form of chi-square. Subsequently, we attempt to calculate the second order derivative of chi-square with respect to the parameters of the model. We will show that this calculation leads to defining the covariance matrix, which will be later used as our basis to generate correlated noise (4.3.1).

[1] released by the EDGES group
[2] generated using ARES

4.1.1 Covariance Matrix

The goal of gradient decent algorithms is to minimize the following sum called the "chi-square":

$$\chi^2 \equiv \sum \frac{(x_i - \mu_i)^2}{\sigma_i^2} \tag{4.1}$$

Where x_i is the observed data, μ_i is the expected value with respect to the theoretical model, and σ is the error associated with each data point. Is it often much easier to calculate the derivatives if we write the above expression in the language of linear algebra. μ can be defined as $\mu_i =< d_i >= A_i(m)$, where A is the model which is dependent on the set of parameters m (the dependency can be nonlinear). We can also define a diagonal noise matrix N, where $N_{i,i} = \sigma_i^2$. Substituting these two in equation 4.1 we get:

$$\chi^2 = (d - A(m))^T N^{-1} (d - A(m)) \tag{4.2}$$

Where d is the array containing the data points.

We continue by calculating the first two derivatives of the above expression, leading to the construction of the gradient decent method.

$$\frac{d\chi^2}{dm} = -(\frac{dA(m)}{dm})^T N^{-1}(d - A(m)) - (d - A(m))^T N^{-1} \frac{dA(m)}{dm} \tag{4.3}$$

Since we know that:

$$(N^{-1})^T = N^{-1} \tag{4.4}$$

$$[\frac{dA(m)}{dm} N^{-1}(d - A(m))]^T = (d - A(m))^T N^{-1} \frac{dA(m)}{dm} \tag{4.5}$$

Substituting in 4.3, we get:

$$\frac{d\chi^2}{dm} = -2(\frac{dA(m)}{dm})^T N^{-1}(d - A(m)) \tag{4.6}$$

$$\tag{4.7}$$

Thus, we can calculate the second derivative:

$$\frac{d^2\chi^2}{dm^2} = -2(\frac{d^2A(m)}{dm^2})^T N^{-1}(d - A(m)) - 2(\frac{dA(m)}{dm})^T N^{-1}(-\frac{d\chi^2}{dm}) \tag{4.8}$$

The first term can simply be neglected due to the following reasons:

1. The $(d - A(m))$ component, which is the residual comparing the model to data, can take both negative and positive values; Thus on average, it will be close to zero.

2. The best-fit model is expected to mimic the behavior of data closely. Therefore, we the residuals must have a small value: $(d - A(m)) \approx 0$

Finally, using the above mentioned logic, we are left with:

$$\frac{d^2\chi^2}{dm^2} = 2(\frac{dA(m)}{dm})^T N^{-1}(\frac{dA(m)}{dm}) \tag{4.9}$$

Which is the definition of the **Covariance Matrix**. The dimensions of this square matrix is the the same as that of m. The diagonal elements of this matrix are simply the inverse of variance on each parameter ($\sigma_{i,i}$), while the off-diagonal elements represent the inverse of covariance, measuring the dependency of a pair of parameters ($\sigma_{i,j}$). If calculated correctly, this matrix is always semi-positive definite[3].

4.1.2 Derivation of the Levenberg-Marquardt Algorithm

In LM method, on each iteration, the set of parameters m is replaced with $m + \delta m$. To find the δm, the function $\chi^2(m + \delta m)$ is approximated by its linearization:

$$\chi^2(m) = (d - A(m))^T N^{-1}(d - A(m)) \tag{4.10}$$

$$\chi^2(m + \delta m) = \chi^2(m) + \frac{d\chi^2}{dm}\delta m \tag{4.11}$$

[3]A positive definite matrix only posses positive eigenvalues. However, for a semi-positive definitive matrix, eigenvalues are non-negative.

Similar to the procedure done in section 4.1, we calculate the derivative of 4.11:

$$\frac{d\chi^2(m+\delta m)}{dm} = \frac{d}{dm}(\chi^2) + \frac{d}{dm}(\frac{d\chi^2}{dm}\delta m) \tag{4.12}$$

We already have the expression for the first order derivative of chi-square in 4.3. Therefore:

$$\frac{d\chi^2(m+\delta m)}{dm} = -2(\frac{dA(m)}{dm})^T N^{-1}(d - A(m)) + \frac{d^2\chi^2}{dm^2}\delta m + \frac{d\chi^2}{dm}\frac{d}{dm}(\delta m) \tag{4.13}$$

Where the last term equals zero since δm does not have any fundamental dependencies on m. Looking at the second term, it is simply inferred that we have already found the expression for second derivative of chi-square in 4.9. Thus, we are left with:

$$\frac{d\chi^2(m+\delta m)}{dm} = -2(\frac{dA(m)}{dm})^T N^{-1}(d - A(m)) + 2(\frac{dA(m)}{dm})^T N^{-1}(\frac{dA(m)}{dm}) \tag{4.14}$$

We define $d - A(m) \equiv r$, and $\frac{dA(m)}{dm} \equiv A'$, so the expression takes the following form:

$$A'^T N^{-1} A' \delta m = A'^T N^{-1} r \tag{4.15}$$

$$\delta m = (A'^T N^{-1} A')^{-1} A'^T N^{-1} r \tag{4.16}$$

Equation 4.16 represents the basis for **Newton's method**. However, as previously mentioned, this method suffers from convergence issues, especially on complicated

likelihood surfaces. To overcome this obstacle, we add a new term to the left-hand side of 4.16. This term includes a control parameter Λ that is updated on each iteration depending on the quality of fit.

$$(A'^T N^{-1} A' + \Lambda I)\delta m = A'^T N^{-1} r \tag{4.17}$$

$$\delta m = (A'^T N^{-1} A' + \Lambda I)^{-1} A'^T N^{-1} r \tag{4.18}$$

Now the basic idea is apparent: On each iteration, a set of parameters m will be replaced by $m + \delta m$ and the chi-square is calculated based on the perturbed parameters. Subsequently, the new chi-square is compared to its value in the last step. If we encounter a higher value, the Λ will be multiplied to a constant arbitrary number (> 1). Otherwise, it will be divided with another constant value (> 1). For practical purposes, if Λ takes a value lower than a constant small arbitrary number, we set it equal to zero. If the Λ is zero, and the chi-square is less than an arbitrary threshold value, we declare the **convergence** of the algorithm.

A flow chart summary of this method is shown in 4.1. The Python code for LM is also available in appendix 7.1.1.

4.2 Markov Chain Monte Carlo (MCMC)

Given that Levenberg-Marquardt (LM) is not always effective in finding the best-fit point for complex likelihood spaces, a more powerful algorithm such as Markov Chain Monte Carlo (MCMC) is necessary in many real-life applications.

MCMC is particularly effective in fitting non-Gaussian likelihood spaces and has a guaranteed convergence, regardless of the starting point in parameter space. However, the algorithm is computationally heavy due to its iterative process, which is based on the evaluation of chi-square on each step. Initially, an arbitrary point in parameter space is given to MCMC as its initial trial step and the associated chi-square is calculated. Then, a random point is drawn from a Gaussian distribution, where the mean is set at the last point in the chain [4]. Subsequently, the new chi-square is compared to the previous one from the last trial step. If the new chi-square is lower, the new trial point is accepted in the chain. If the new chi-square is higher, the trial point is accepted with a probability determined by a specific criterion.

The above-mentioned probability threshold is typically defined as:

$$Probability = e^{\frac{-1}{2}(\chi^2_{new} - \chi^2)} \tag{4.19}$$

Which again has Gaussian insights.

[4] In Python applications, the numpy.randn function is used to serve this purpose.

As a measure of MCMC performance, the acceptance ratio is used to determine the fraction of trial steps that end up getting accepted into the chain. An ideal MCMC would typically have an acceptance ratio of 25 percent. However, even with a lower acceptance ratio, the MCMC can still converge, but it will require more trial steps.

A visual summary of MCMC algorithm is shown in 4.2, and the Python code is available at 7.1.2.

4.2.1 Convergence Test

The MCMC algorithm is designed to explore different regions of the parameter space in order to reach convergence. Various methods have been developed to ensure that the MCMC has converged, one of which is to check the power spectrum.

The power spectrum represents the distribution of power at different frequencies in the MCMC chain. A converged MCMC chain must have the behaviour of a white noise, with power uniformly distributed among all frequencies. On the other hand, an unconverged chain will show more power at lower frequencies compared to higher ones. Therefore, the criterion for checking the convergence of an MCMC chain is the flatness of the power spectrum in low frequencies when plotted on a log-log scale. Figure 4.3 and 4.4 illustrates the difference between a converged and an unconverged chain.

4.3 Combination of MCMC and LM

As mentioned before, MCMC is a computationally heavy algorithm due to its iterative nature. If calculating the chi-square takes a long time on each step, the MCMC itself will have a rather long run time before reaching the converged state. Different methods have been proposed to deal with this issue and help the MCMC to converge faster, one of which is to use the insights from running LM.

We previously discussed that parameters of a model might be correlated (4.1.1). During a simple MCMC, we are drawing random samples from a gaussian distribution. This samples do not take the possible correlations into account. However, if we generate samples with such characteristics, the probability of them getting accepted into the actual chain is higher. Therefore, this approach (feeding the MCMC with a posterior distribution) will eventually assist the MCMC to explore more efficient regions of parameter space, and converge faster.

Since the covariance matrix of these samples is already in hand, we are able to easily generate a set of correlated noise samples. The procedure to do so is described in the following section.

4.3.1 Generating correlated noise

As discussed before, the off-diagonal elements of a covariance matrix correspond to the inverse of covariance between each pair of parameters. Thus, if we draw samples from the inverse of covariance matrix (which needs to be calculated at the point of ”best-fit”), you

are essentially sampling from the multivariate normal distribution with the deviation values describing the uncertainties in the parameters.

The equivalent methods can be used to generate correlated noise: Cholesky and eigenvalue decomposition[5]. For practical reasons, we prefer to use the eigenvalue decomposition for 21cm applications. The procedure is a s follows: A matrix of normal gaussian-drawn random variables is constructed in the desired shape ($n \times m$, corresponding to the number of samples and number of parameters respectively). Then, it is multiplied by the eigenvalue matrix and scaled by the square root of the eigenvalues. The transpose of the product will give us the correlated samples.

To gain more precision, is it possible to use the eigenvalues decomposition of the normalized covariance matrix (where diagonal samples all equal unity). The normalization process is done by multiplying the covariance matrix with its own diagonal. Eventually, the drawn samples need to be scaled by the square root of the diagonal matrix.

The Python implementation of the above-mentioned procedure is given in 7.1.3, and the flowchart is shown in 4.5.

4.4 Testing the Algorithm

The complicated algorithm described in the previous sections, does not always behave as expected. Thus, naturally, one seeks options to weigh the output. In sections 4.4.1 and

[5]Normally, calculating the Cholesky decomposition takes a shorter amount of time

4.4.2, we introduce two methods to measure the overall quality of the model fitting.

4.4.1 The chi-square test

This method is more focused on inspecting the output of LM and drawn samples. A large number of samples are drawn from the covariance matrix and the corresponding chi-squares are calculated. Since the covariance matrix describes the uncertainties in the parameters, we expect that the average difference in the chi-square statistics for two different samples should be of order unity per each parameter. This comes from the fact that the chi-square statistic scales with the uncertainties in the data and model, which are typically of order 1.

Figure 4.6 shows an example of the distribution of difference between the chi-square values.

4.4.2 Chi-Square vs Parameters Plots

Another method to verify the results is to plot the chi-square values of drawn samples versus the each of the parameters. According to the definition of chi-square 4.2 (the non-linear dependency of chi-square on model parameters) we expect to observe a parabolic behaviour.

Figure 4.7 demonstrates chi-square vs parameters plots for the same model as 4.6.

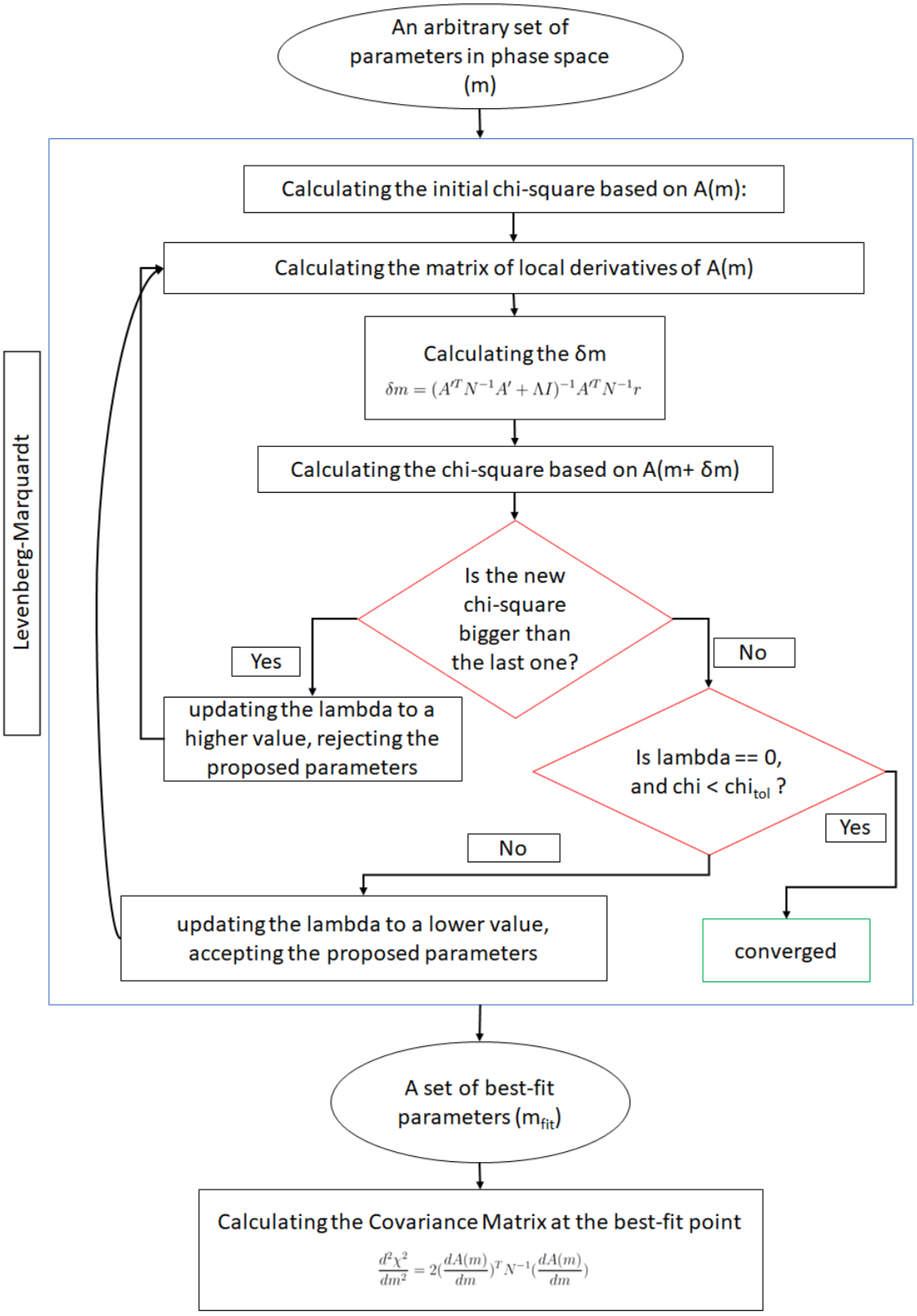

Figure 4.1: Flow chart of Levenberg-Marquardt algorithm

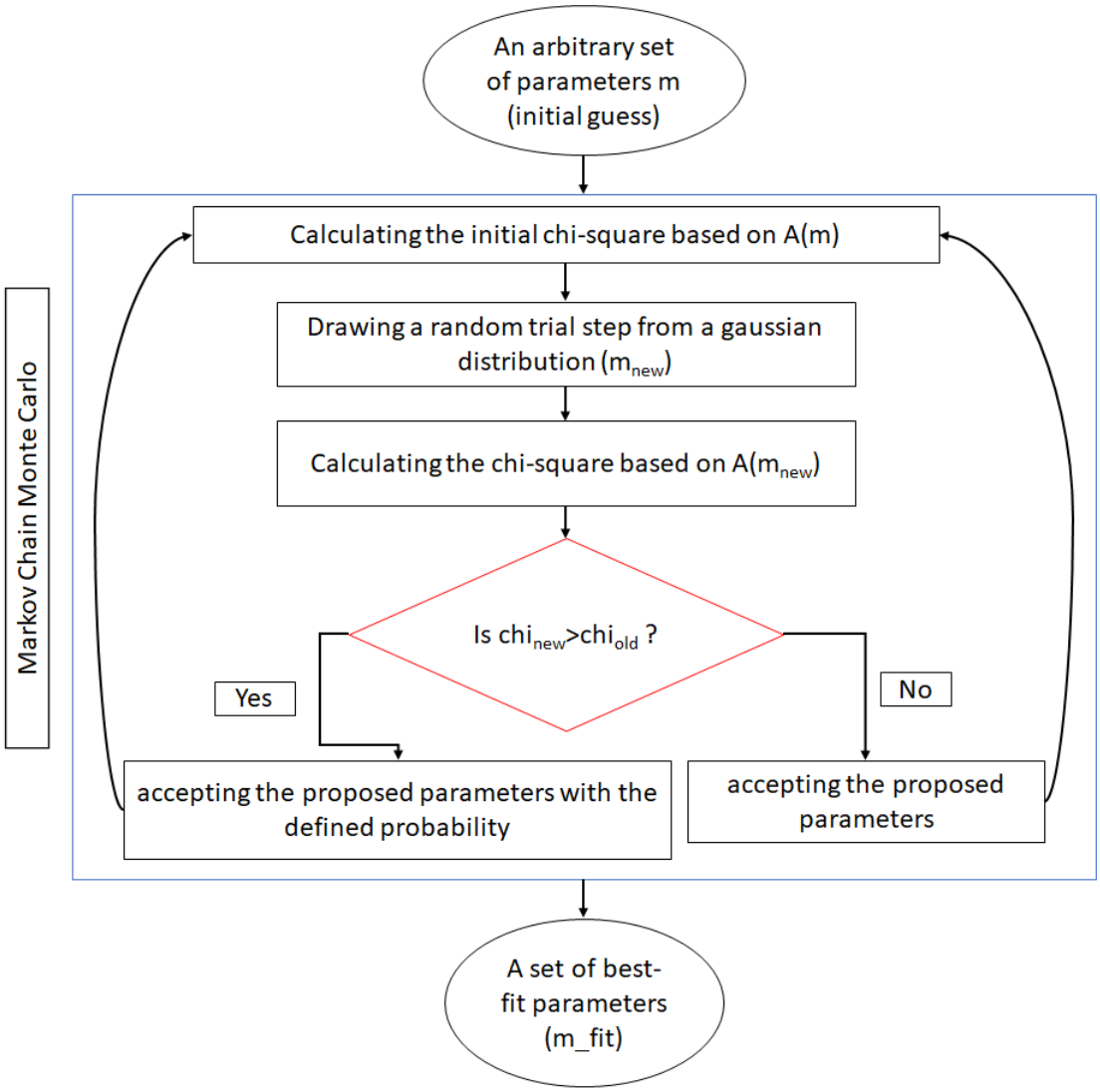

Figure 4.2: Flow chart of MCMC algorithm

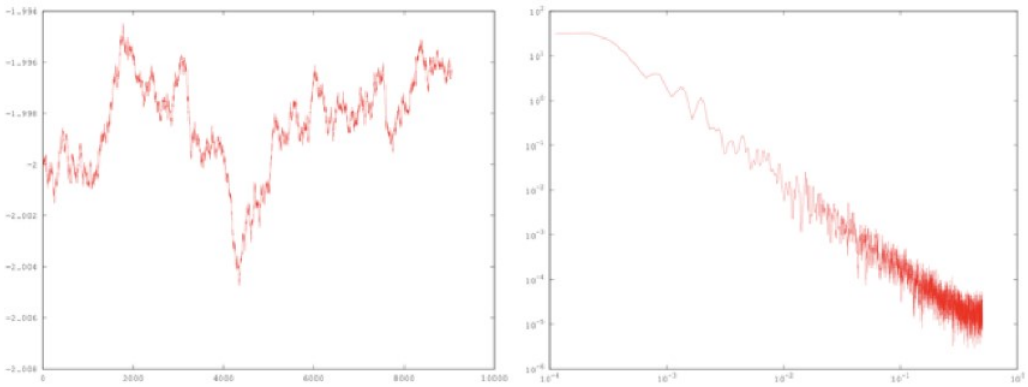

Figure 4.3: An unconverged MCMC chain (left panel) and its power spectrum (right panel): The power tend to increase in lower frequencies, the chain itself does not indicate the behaviour of white noise, plot from Jonathan Sievers

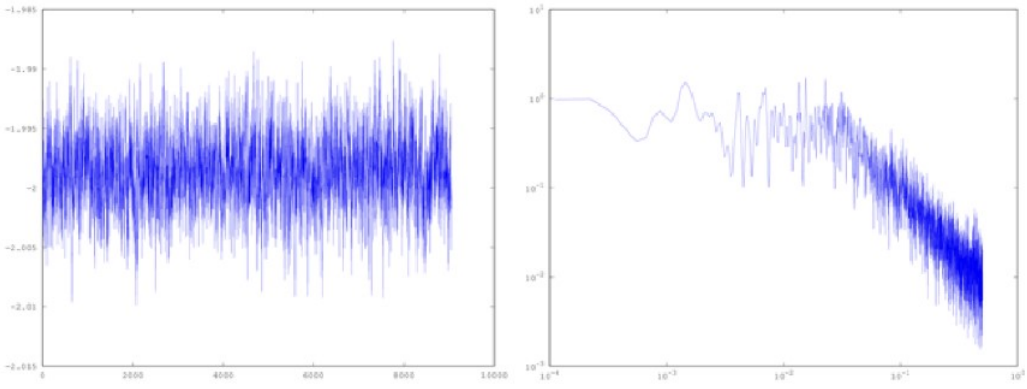

Figure 4.4: A converged MCMC chain (left panel) and its power spectrum (right panel): Note the flatness on low frequencies, plot from Jonathan Sievers

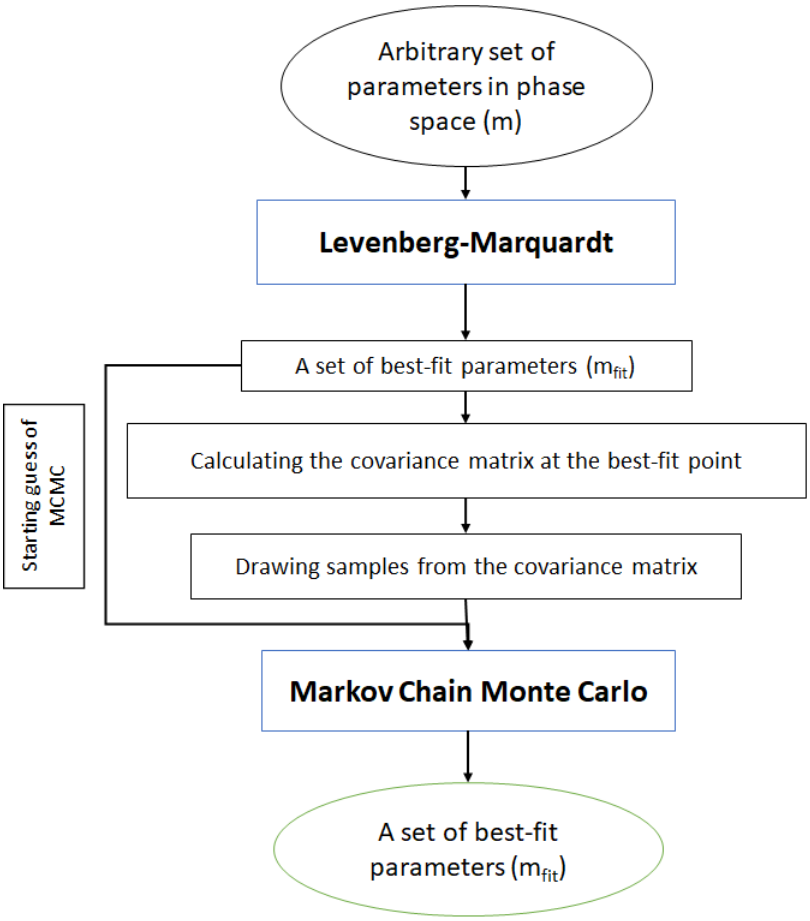

Figure 4.5: Flow chart of the procedure to combine MCMC and LM

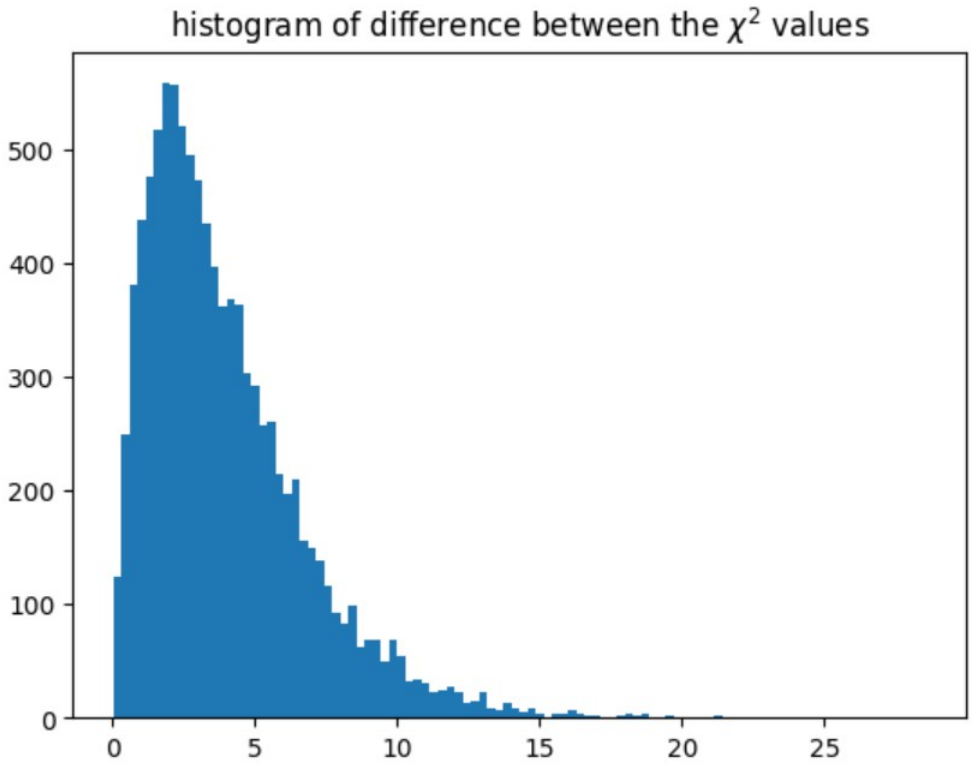

Figure 4.6: Histogram illustrating the distribution of the values of difference between the chi-squares of drawn samples for a four variable model. The average is 3.97 and the standard deviation is 2.83, in agreement with our expectations. The best-fit parameters is considered as a "good fit".

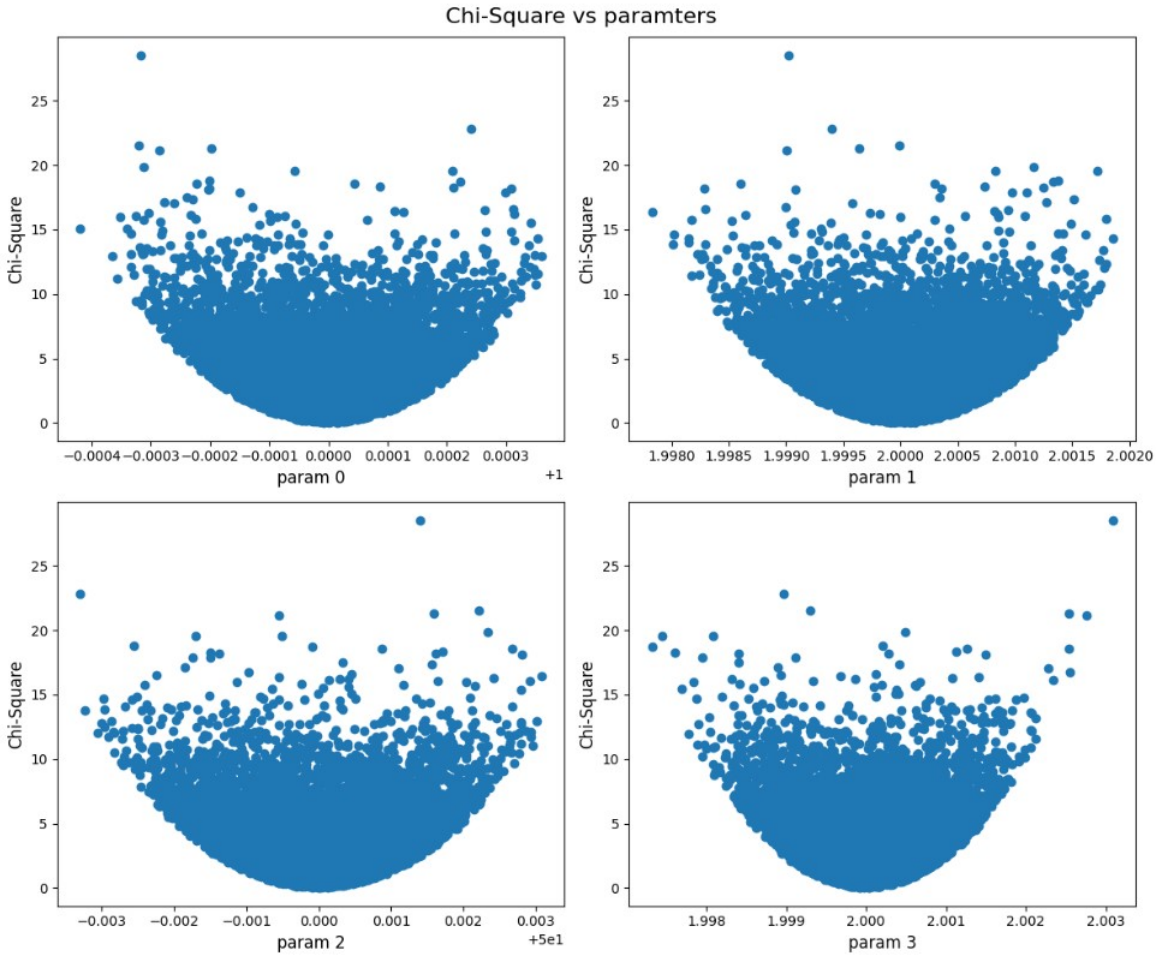

Figure 4.7: Plots of the values of chi-square of drawn samples versus the values of parameters for a four variable model. The parabolic behaviour is obvious.

Chapter 5

Results and Analysis

5.1 Parameter estimates of EDGES data and uncertainties

Talk about the choice of parameters talk about the sensivity of the model due to each param

5.2 Comparison with previous studies and observations

Chapter 6

Discussion and Conclusion

6.1 Interpretation of the results

6.2 Summary of the main findings

6.3 Contributions and significance of the research

6.4 Limitations and future work

Chapter 7

Appendices

7.1 Code snippets and scripts

7.1.1 Levenberg-Marquardt

7.1.2 Markov Chain Monte Carlo

7.1.3 drawing samples from covariance matrix

Listing 7.1: Python example

```
def draw_samples(covariance_matrix, nset):
"""
covariance_matrix: covariance matrix
```

```
    nset:the number of samples

    returns: a matrix of samples

    This function calculates a series of correlated samples based on the pres

    covariance matrix and the number of samples.

    The shape of the output is (nset, m) where m comes from the shape of

    covariance matrix and it typically shows the number of parameters in the

    """

    #normalizing the covariance matrix

    D = np.diag(np.diag(covariance_matrix)) #diagonal matrix of covariance m

    D_sqrt = np.sqrt(D)

    D_inv_sqrt = np.linalg.pinv(D_sqrt)

    #normalized covariance matrix

    covariance_matrix_normalized = D_inv_sqrt @ covariance_matrix @ D_inv_sqr


    e,v = np.linalg.eigh(covariance_matrix_normalized)

    e[e<0]=0 #omitting any negative eigenvalues due to roundoff

    n = len(e)


    #make gaussian random variables

    g=np.random.randn(n, nset)
```

```
#scaling by the square root of the eigenvalues
rte=np.sqrt(e)
for i in range(nset):
    g[:,i]=g[:,i]*rte

#calculating the samples
samples = (v@g).T
#denormalizing the samples
samples_denormalized = samples @ D_sqrt
return samples_denormalized
```

Bibliography