POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Politecnico di Milano

052499 - Bayesian Statistics

055700 - Computational Statistics

Autumn 2021

Multilevel Markov Chain Monte Carlo

Student:	**Student ID**
Rasmus Tuxen	10844184
Simon Fredriksen	10846824
William Rom	10846436

February 14, 2022

Contents

List of Figures

Chapter 1

Introduction

We often find ourselves in the context of inverse mathematical problems, where we have access to a set of observations, but the causal factors that produced these remain unknown. Deriving some forward model from these observations is a classic statistical problem, and can be solved in different ways.

While the idea of Bayesian statistics is over 250 years old, the Bayesian approach has been considered inferior to the frequentist for most of these. However, with the rapid development of computational power available through computers, Bayesian methods has seen increased attention in the last decades. In this report we dive into one of the most famous classes of computational statistic algorithms, Markov chain Monte Carlo methods. In particular, we are interested in exploring the possibility of increased computational efficiency with the more recent hierarchical technique, multilevel Markov chain Monte Carlo (MLMCMC).
This project report is the result of a group project that formed part of our evaluation of the Bayesian Statistics and Computational Statistics courses taught at Politecnico di Milano by professors A. Guglielmi and A. Manzoni respectively.

Chapter 2

Introduction to Multilevel Markov Chain Monte Carlo methods

2.1 Standard Monte Carlo

Monte Carlo simulation is a computational technique that relies on repeated random sampling to solve problems of mainly three categories: optimization, numerical integration and drawing from a probability distribution. In the context of Bayesian inverse problems, we often wish to compute the expected value of some quantity of interest Q, which indeed employs numerical integration. Based on N samples $Q_1, ..., Q_N$, which again depends on some random element θ, we can compute an approximate expected value of the quantity given by

$$\mathbb{E}[Q] \approx \frac{1}{N} \sum_{i=1}^{N} Q(\theta_i).$$

It can be shown by the application of the law of large numbers and the central limit theorem, that the Monte Carlo approximation of the expected value converges to the true expected value with a rate of $\frac{1}{\sqrt{N}}$.

2.2 Issues with standard Monte Carlo and extending to the multilevel method

In Bayesian inference we depend on data or observations of this quantity of interest in order to produce a more accurate estimate of this quantity. In the Bayesian framework, this is called determining a posterior distribution of Q. More specifically, we are trying to determine posterior distributions of the parameters which are used in the forward model F that the quantity of interest depends on. Very often, this forward model is a differential equation with potentially many parameters that we wish to make inference on. In order to compute a Monte Carlo estimate of the expected value of Q, we rely on the numerical solution of a differential equation N times, and with the slow convergence rate of Monte Carlo, and potentially expensive forward model computation, estimating this expected value for a given accuracy level can quickly become computationally prohibitive. The multi level Monte Carlo idea proposes a divide-and-conquer strategy to this problem.

We introduce a dividing of the quantity of interest into l different levels for $l = 0, ..., L$, ranging from a low ground level 0 that corresponds to a coarse estimate, to a high level L that corresponds to a finer and more accurate estimate. This means that we make inference on the parameters used in the forward model on $L + 1$ different levels, which requires the constructions of $L+1$ individual Monte Carlo estimators, where the computational cost of each estimator increases proportionally with the increase of level. To be specific, coarseness of the forward model can be related to the grid resolution of the numerical solution of the differential equation, meaning that a lower resolution could represent the forward model at a low (coarse) level. Coarseness can also be introduced by the degree of noise in the observations we make of the dynamical system or natural phenomena we are considering. In this project we consider both of these variants of coarseness. In the multi level Monte Carlo estimator, we exploit the linearity property of expectation, and write

$$\mathbb{E}[Q_L] = \mathbb{E}[Q_0] + \sum_{\ell=1}^{L} \mathbb{E}[Q_\ell - Q_{\ell-1}] \tag{2.1}$$

which results in the multi level estimator[1]

$$\hat{Q}^{MLMC}_{\{N_\ell\},L} = \frac{1}{N_0}\sum_{i=1}^{N_0} Q_0(\theta^{(i,0)}) + \sum_{\ell=1}^{L}\frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\left(Q_\ell(\theta^{(i,\ell)}) - Q_{\ell-1}(\theta^{(i,\ell)})\right). \tag{2.2}$$

The first term in this sum is the ordinary Monte Carlo estimator at the lowest level. Then, we sum up the Monte Carlo estimators of the differences between two consecutive levels ℓ and $\ell - 1$. It may seem that because of the fact that we now have to compute $L + 1$ individual Monte Carlo estimators, that it is not feasible compared to standard Monte Carlo. However, the key observation here is that the low level estimators are computationally inexpensive. If the estimator converges as $\ell \to \infty$, i.e. $Q_\ell \to Q$, then $Q_\ell - Q_{\ell-1}$ tends to zero in mean-square. This convergence is exponential in ℓ, which implies a variance reduction on the higher levels. The mean-square error (MSE) of a sample mean is proportional to the variance of the sample. This means that the number of samples produced at level ℓ, N_ℓ, can be chosen to decrease exponentially as level increase, since we need less samples in order to obtain an estimator with the same variance. A requirement to obtain this desired variance reduction at higher levels is that we use the same sample $\theta^{(i,\ell)}$ on two consecutive levels.

2.3 MLMCMC

The use of general state space Markov chains combined with the discussed multi level version of the Monte Carlo method forms the basis for the MLMCMC algorithm. We now view the domain of the inferred parameters in the forward model as the state space of Markov chains. In Bayesian inference we are often required to sample from the posterior probability distribution of, in our case, the parameters used in the forward model. This is what allows us to compute estimates of the parameters θ, and thus the quantity of interest Q. This posterior density is often known only up to a normalizing constant, consisting of the likelihood of the observations and a prior probability distribution over the parameters. We therefore rely on repeated sampling from a Markov chain whose limiting distribution is indeed the posterior distribution of the parameters. The Metropolis algorithm can be seen as a special case of MLMCMC with just one "level". The general principle in this algorithm is the following[2]:

- Choose an initial state θ_0. I.e. a point in the parameter space. This can for instance be determined by optimizing some kind of loss function which measures the discrepancy between observations and the solution of the forward model. This makes the least-square estimate of the parameters, $\hat{\theta_{LS}}$ a natural choice.
- For $k = 0, 1, ...$: Draw a proposed next state θ^* from a proposal distribution $q(\cdot \mid \theta_n)$ conditioned on the previous state. This proposal distribution should be "close" to the target distribution to ensure efficient sampling. This is covered in detail in chapter 3.
- Accept the proposed next state with probability $\alpha = \min\left\{1, \frac{\pi(\theta^*)}{\pi(\theta_{k-1})}\right\}$ This is the mechanism that allows the chain to explore the state space and characterize the posterior distribution over the parameter domain. Here π is a density that is easy to sample from, which is computed from the likelihood of observations given θ^* and θ_{k-1} and prior over the parameters θ^* and θ_{k-1}.
- If rejected, set $\theta^* = \theta_{k-1}$, and return to the top of the loop.

We use MCMC to compute approximations of properties of the posterior distribution, like its expected value or variance. For ordinary MCMC, we use the standard Monte Carlo approach described in section 2.1. But as we have discussed, this might lead to an unfeasible computational cost due to costly likelihood evaluations in every iteration because of the need to solve a differential problem on a sufficiently fine numerical grid. So we employ the multi level Monte Carlo idea from section 2.2. The challenge now is to design Markov chains that ensures variance reduction at higher levels of the Monte Carlo estimator in order to reduce sample sizes at higher levels. Denote by F_ℓ and P_ℓ the forward model and posterior distribution respectively at level ℓ for $\ell \in \{1, ..., L\}$. We now again consider the estimator of (2.2). The first term is determined by means of standard MCMC with N_0 samples. This term calculates the "bulk" shape of the posterior, and is cheap to compute. The remaining terms can be viewed as "corrections" of this first rough estimate of the posterior. So, at level ℓ, for each sample $\theta^{(i,\ell)}$ in that level where $i = 1, ..., N_\ell$, we use the sample $\theta^{(s+R,\ell-1)}$ from level $\ell - 1$, which is a random sample in the batch of K samples produced on level $\ell - 1$ to propose the next state of the chain in level ℓ, being s the starting point of the batch. Meaning, that K is the subsampling rate at level ℓ and R, ranging from 1 to K, is the index of the randomly picked sample. From this, we can calculate an estimate of the difference in Q between two consecutive levels

$$Y_i^\ell = Q_\ell(F_\ell(\theta^{(i,\ell)})) - Q_{\ell-1}(F_{\ell-1}(\theta^{(s+R,\ell-1)})).$$

We then sum up all of these estimates and make an average

$$\mathbb{E}_{P_\ell}[Q_\ell] - \mathbb{E}_{P_{\ell-1}}[Q_{\ell-1}] = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} Y_i^\ell$$

Finally, we sum up all of these differences up to level L, and obtain the complete estimator at level L for Q as described in (2.2).

Chapter 3

Numerical experiments with MLMCMC framework

In our project we have chosen to focus on two dynamical systems described by ODE's, namely the oscillating spring system and the Lotka-Volterra predatory-prey system. The overall goal of both MLMCMC and the Metropolis method is to make inference about some random variable that in our case will be the parameters in the ODE's describing the dynamical systems. The following sections will on the basis of numerous numerical experiments concern the discussion of the applicability, performance and accuracy of the MLMCMC method described in the previous chapter, compared to the Metropolis algorithm.

In the following examples we will consider a set of true parameters θ^* from which a number of observations in time can be derived using the analytical solution $y = f(t, \theta^*)$.
We will generally consider two scenarios for parameter estimation. In the first scenario we make direct inference on the model parameters by considering the analytical solution, $\tilde{y} = f(t, \tilde{\theta})$. In the second scenario we consider the case where we don't have access to the exact analytical solution and instead have to rely on a numerical approximation, $\tilde{y} = \tilde{f}(t, \tilde{\theta})$. In this case, we model the system behavior as a random process, and perform inference on the parameters that characterizes this random process.

3.1 Defining a target distribution

In the Markov Chain framework we need to specify a target distribution, which will produce a likelihood for a given realization of the inference parameters, $\tilde{\theta}$.
Given a set of observations $y = f(\theta^*)$ then the negative log-likelihood of a Gaussian can be used as target distribution:

$$L = -\frac{1}{2\sigma^2} \sum (y - f(\tilde{\theta}))^2$$

Where f is either an analytical or numerical solver.

3.2 Efficiency and Accuracy

In the following experiments we define two ways of assessing the performance of a given sampler:
The efficiency of a Markov Chain can be measured by the `Effective Sample Size (ESS)` which is an estimate of the sample size required to achieve the same level of precision if that sample was a simple random sample.
The accuracy is measured using the `Mean Squared Error (MSE)` between the observations y and the model predictions using the mean of the parameters in the Markov Chain $f(\tilde{\theta})$.

These two performance statistics of efficiency and accuracy are not bound to follow each other and are thus both necessary for an estimation of a given models performance.

3.3 The oscillating spring system

We consider the spring model

$$\begin{cases} u'' + Cu' + Ku = 0 \quad t > 0 \\ u(0) = 2 \\ u'(0) = -C \end{cases}$$

Which has the analytical solution

$$u(t) = 2e^{\frac{-Ct}{2}} cos\left(\sqrt{K - \frac{C^2}{4}}t\right).$$

The parameters C and K are in real life subject to uncertainty, and this uncertainty should be quantified and dealt with, by means of Bayesian inference. For all of the following experiments the true parameters for data generation are:

$$C_{true} = 1.5$$
$$K_{true} = 20.5$$
$$T_{max} = 5$$

In figure 3.1 the log-likelihood of different values of C and K evaluated on 300 data points can be seen. Here we can see that the K parameter seems to have a much wider range of values with a low negative log-likelihood than the C parameter does.

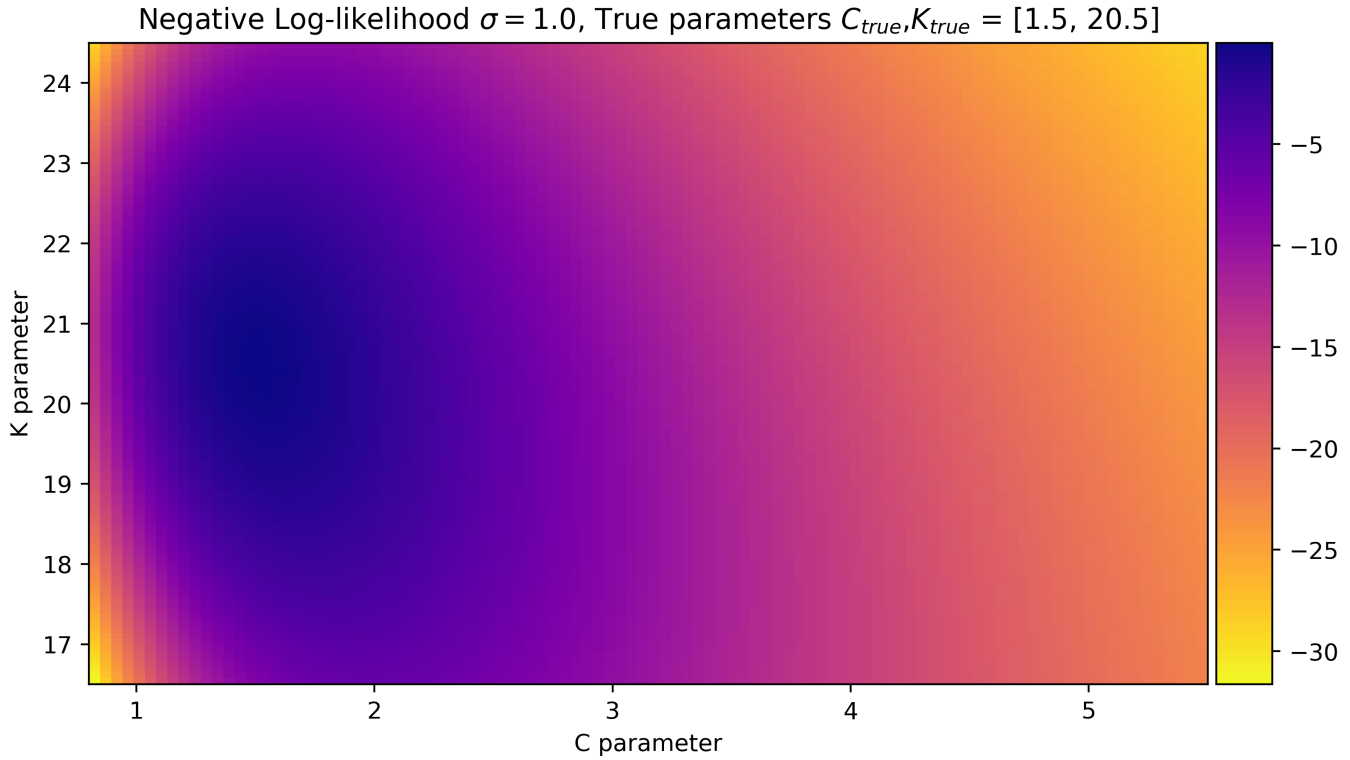


Figure 3.1: **Likelihood function for a range of input parameters:** The likelihood of a model with certain values of C and K (higher is more likely) evaluated on a 200×200 grid. The true parameters (with the highest likelihood) are $[C_{true}, K_{true}] = [1.5, 20.5]$.

We performed numerical experiments in the context of making inference on the mentioned parameters, by creating 5 different models (one of which corresponds to the standard Metropolis method), that are characterized by different features. The experiments are the following:
Testing the impact on performance, effective sample size (ESS), mean square error (MSE) and acceptance rate by

- Varying the amount of data observations used in the coarser levels of each model.
- Varying the degree of zero mean Gaussian noise in the data at different levels of each model
- Applying different prior distributions for each model.

- Varying the entire covariance matrix in the Gaussian proposal distributions, i.e. introducing covariance between the two parameters.
- Varying the variance of the Gaussian proposal distributions
- Changing the sub-sampling rate of the coupled chains in the multi-level models.

All of these tests were performed using 6000 draws in three Markov Chains where the first 1000 are used as burn samples, which from initial experimentation seemed sufficient for convergence. The sub-sample rate between levels were $[3, 5]$ such that three draws would be made in the first coarse level before sampling in the fine level etc.
We have only included a discussion about the three first experiments, since these were the ones that gave results we considered useful and interesting when learning about the properties of multilevel MCMC.

3.3.1 Analytical Scenario

3.3.1.1 Coarsity experiment

In this experiment we created the following model configurations:

- One standard one-level model with 400 data points, using the Metropolis method.
- Two 2-level models, one with more data points in the coarse level than the other.
- Two 3-level models, one with more data points in the coarser levels than the other.

Figure 3.2 shows the system behavior in the first 5 seconds, in all the different model configurations. We include the data plot only in this case, as the model configurations in the other experiments are similar, and a brief explanation of the setup will suffice.

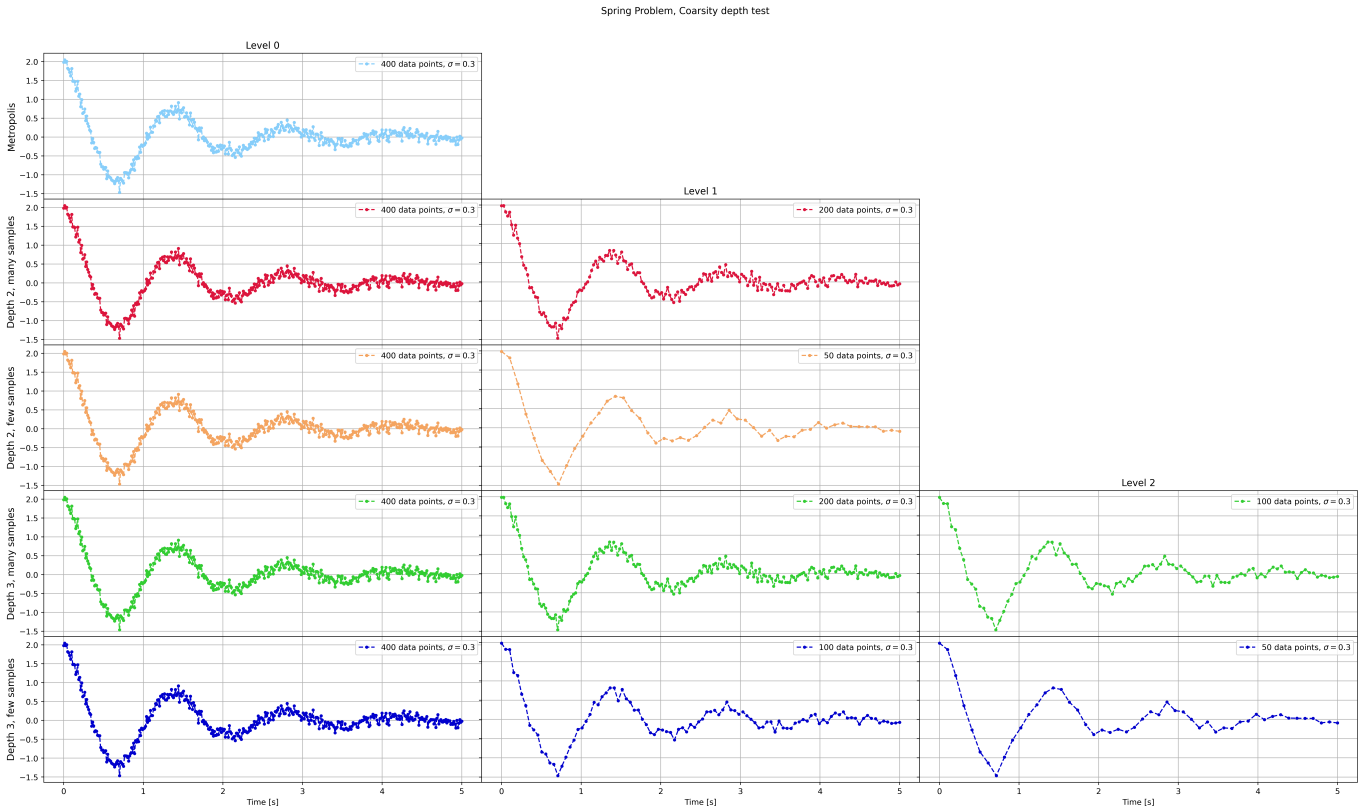
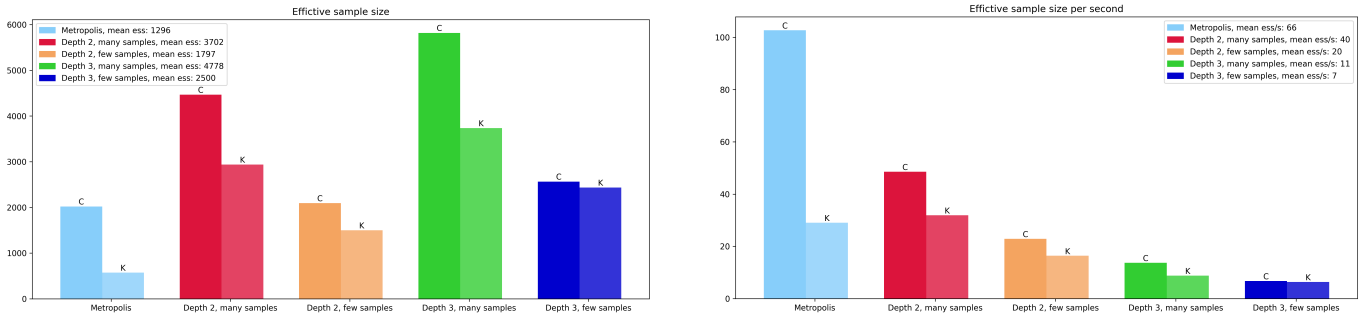


Figure 3.2: **System behaviour in different model configurations:** The five models are presented row wise, where the first one corresponds to the one-level Metropolis method, the two following the 2-level models, and the two last ones are the 3-level models.

What we can observe from figures 3.3 and 3.4, is that the deeper models, i.e. models with more levels generally samples much more effectively from the posteriors than Metropolis, especially if they are given more data. This is because the finest chain is sampling from a proposal distribution that is given by the chain at the level below, and is

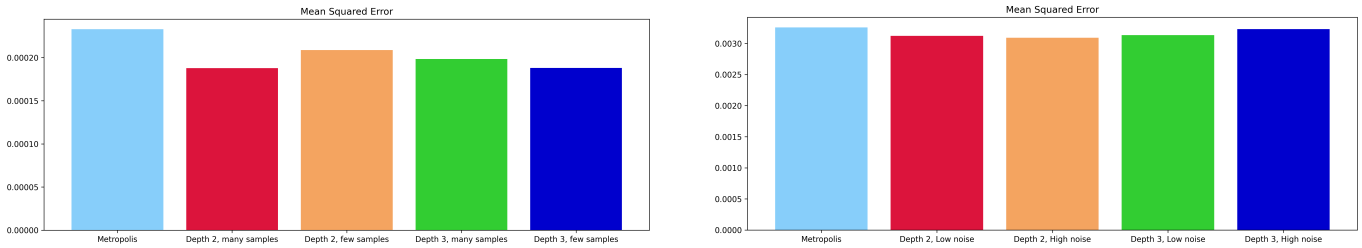
therefore sampling from a distribution that ideally is really close to the actual posterior, leading to a higher acceptance rate and ESS. This is not guaranteed for the Metropolis method. However when it comes to performance (ESS per second), we see that the Metropolis algorithm outperforms all the other models. This is expected since this is a quite simple inference problem, and it is not given that it is feasible to apply MLMCMC in this case. Afterall, MLMCMC is expected to be useful when the parameter space is high dimensional, and it can outperform one level methods both in computational efficiency and accuracy. It surges a trade-off between computational efficiency and accuracy, and it seems in this case that the Metropolis method can provide better performance and equal accuracy than the other multilevel methods. When it comes to accuracy, we see that all of the models obtain quite low MSEs, with negligible differences between them.
In figure 3.5 we observe that the (marginal) posterior distributions of C and K have somehow different shapes. This can be explained by observing the likelihood function in figure 3.1. We see that for a much larger range of K's we can obtain a similar likelihood estimate, which means that a larger range of values for K will be accepted as transitions in the Markov chain leading to the posterior of K. The likelihood of the C parameter is much more concentrated, and therefore the posterior of K will be more heavily tailed than the one of C. This non-uniformity in likelihood also explains that the ESS are different for C and K, and the most frequent case is that the ESS for C is higher.



(a) Effective sample size

(b) Effective sample size per second

Figure 3.3: **ESS and performance in coarsity experiment**:



(a) Coarsity experiment

(b) Data noise experiment

Figure 3.4: **Mean squared errors in coarsity and data noise experiments**

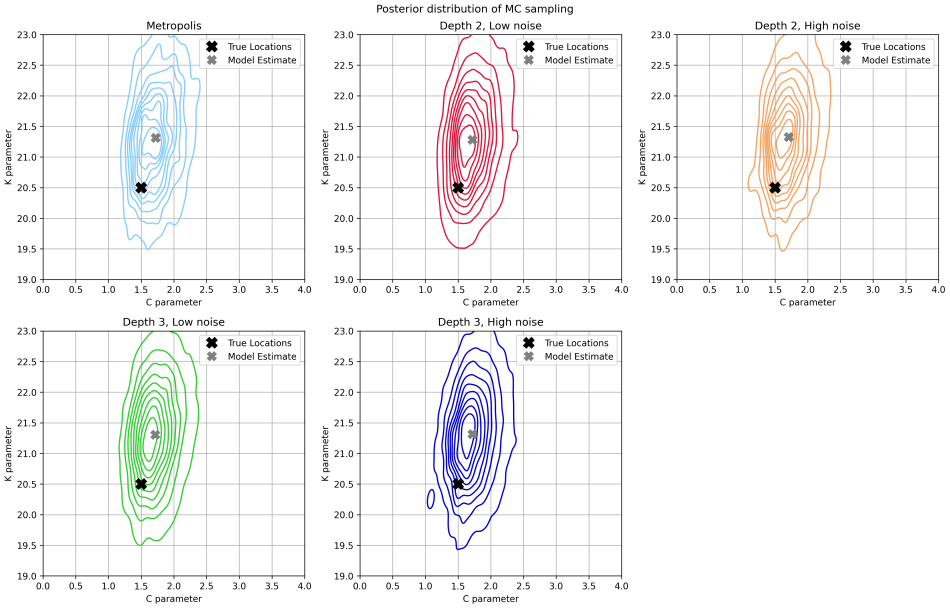


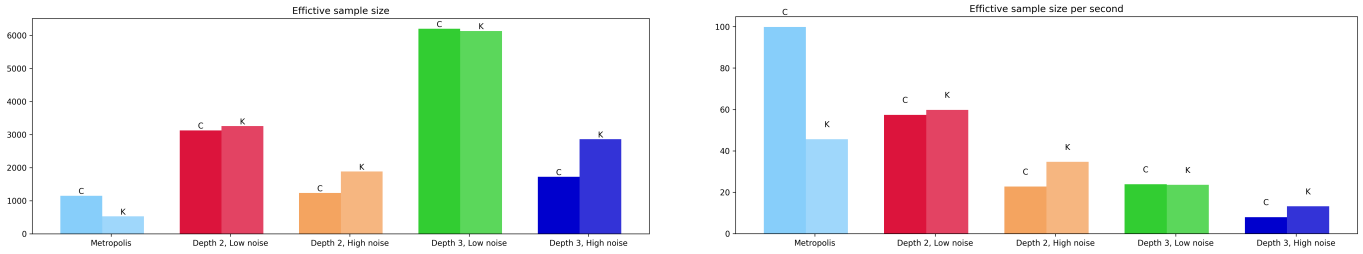
Figure 3.5: **Posteriors from coarsity experiment**

3.3.1.2 Data noise experiment

Another way of interpreting the concept of increased coarsity in deeper levels is to imagine that the inherent measurement noise increases as the levels increase.
In this experiment we created the following model configurations:

- One standard one-level model with 400 data points, using the Metropolis method.
- Two 2-level models, all levels with 400 data points. One model has more noisy data in the coarse level and the other has less noise.
- Two 3-level models, all levels with 400 data points. One model has more noisy data in the coarser levels, and the other has less noise.

Now the data measurements are less accurate, but we have the same amount of data points in all levels. This means that we cannot expect the multilevel methods to be better than Metropolis in terms of runtime, but we can still obtain a better ESS for the deeper models, which is exactly what we found, as illustrated in figure 3.6. The models with low-noise data samples more efficiently from the posteriors compared to high-noise models. All the models obtain a quite low MSE also in this experiment.



(a) Effective sample size

(b) Effective sample size per second

Figure 3.6: **ESS and performance in data noise experiment**

Additionally we see from figure 3.7 that the MLMCMC model are able to produce a much higher acceptance rate than the standard Metropolis method.

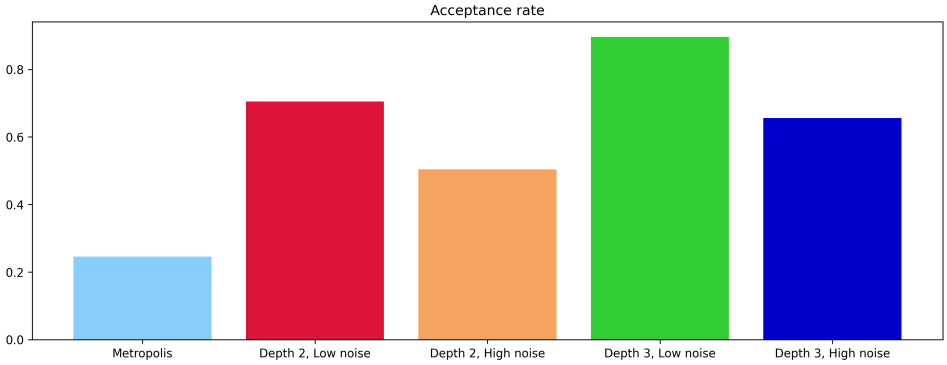


Figure 3.7: **Acceptance rate for the models in the Data noise experiment**

3.3.1.3 The impact of prior distributions

We investigated the behavior of a model when exposed to a variety of prior distributions for the parameters. The configuration of the model we used is the following: 2 levels where the fine level has 500 data points and noise $\sigma = 0.2$, and the coarse level has 250 data points and noise $\sigma = 0.3$. Figure 3.8 shows the prior distributions we have considered in the experiment.

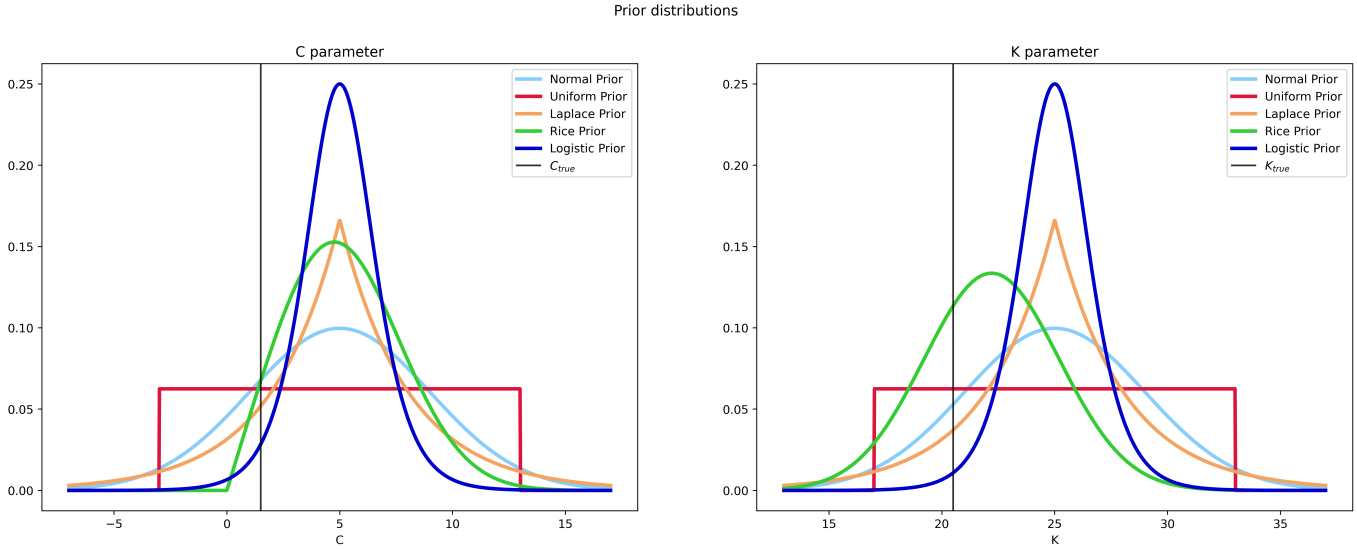
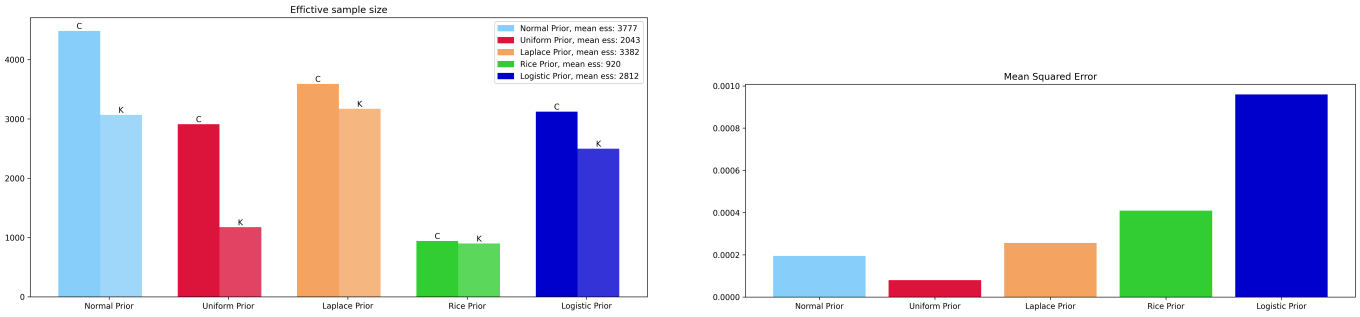


Figure 3.8: **Prior distributions for C and K**

The prior distribution has great influence on the shape of the posterior distribution, especially if few observations are available for the likelihood calculation. Therefore it is natural to believe that using different priors will lead to variations in the posterior distributions and thus differences in terms of accuracy and effective sampling. Figure 3.9 demonstrates that the most efficient samplers come from the normal prior, Laplace prior and the logistic prior while the highest accuracy is achieved by the uniform prior.

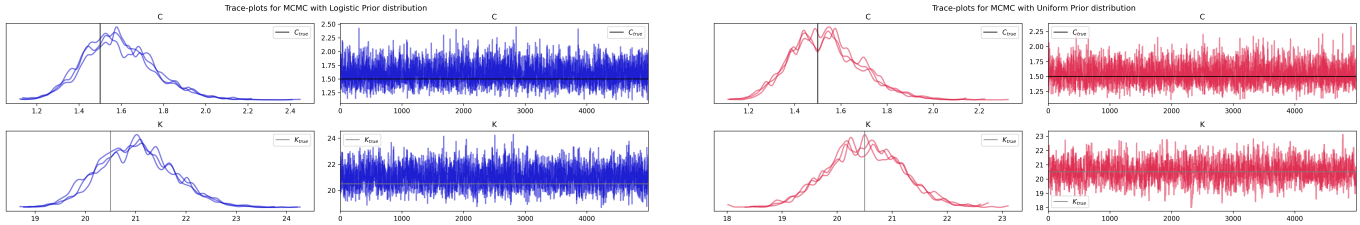


(a) Effective sample size

(b) Effective sample size per second

Figure 3.9: **ESS and MSE for different prior distributions.**

Even though all the models seem to achieve a similarly low MSE there is still a noticeable difference in the resulting parameters as can be seen in figure 3.10. Here it is observed that the while the uniform prior hits the true parameters the logistic is actually quite far off - especially for the K parameter.



(a) Trace for Logistic Prior

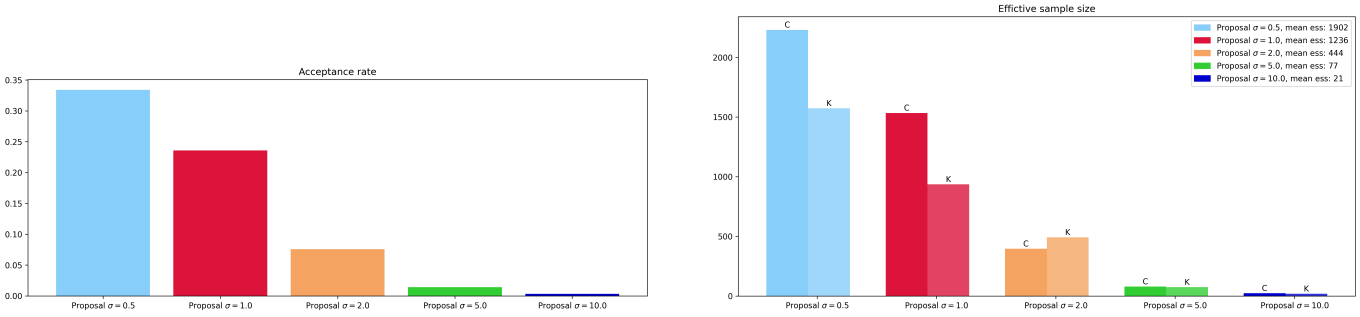
(b) Trace for Uniform Prior

Figure 3.10: **Trace plots for two of the tested priors**

From these results there is a case to be made for either using a Normal prior or a Uniform prior as these achieve the most efficient sampling and highest accuracy respectively.

3.3.1.4 Impact of Proposal variance

Another knob in the MLMCMC framework is the proposal variance. It is by default a Gaussian with $\sigma = 1$, but it is possible to adjust this parameter. This will effectively lead to increasing the step size in the chains.
For this test the same data as in the `Prior distribution test` was used. In figure 3.11 the acceptance rate and ESS can be seen. Here we observe that unsurprisingly a lower proposal variance leads to a higher acceptance rate. Which in this case also seems to lead to a higher effective sample size.



(a) Acceptance rate

(b) Effective sample size

Figure 3.11: **Acceptance rante and ESS for different proposal variances**

3.3.1.5 Other experiments

We performed other experiments that are not discussed in this report, but the numerical results and plots can be consulted in the github repository related to the project: `https://github.com/RTuxen/MLMCMC`.

3.3.2 Numerical Scenario

Now we will turn to the case in which we do not know the exact solution of the model problem, and for the sake of simplicity, we will just pretend not to know the analytical solution of the spring problem. Now, our goal is to represent the data itself as random variables to make inference on, and not the parameters used in the model to generate these data, as has been the case until now. However, making inference on hundreds of parameters (gridpoint solutions) is unfeasible, so we must settle for an approximation. This collection of numerical solution points can be represented as a random field, and more specifically as a finite sum of uncorrelated random variables, by the use of the truncated Karhunen-Loeve expansion. So, our parameters of interest are now the modes of the KL-expansion of a Gaussian process, that now can be considered the "prior" knowledge we have about the numerical solution of the system. The goal is now to compute posterior distributions of these random variables.

First, it is necessary to run a number of tests in order to determine the most optimal model configuration. We must determine

- A given number of parameters from the KL expansion to include for inference.
- What type of covariance kernel that gives the most reasonable results.
- Properties of the covariance kernel like the optimal kernel variance and lengthscale.

According to our numerical tests, we found that

- 10 Karhunen-Loeve modes
- Squared exponential covariance kernel
- Kernel variance $\sigma^2 = 4$
- Lengthscale $\ell = 0.5$

was the optimal configuration for further numerical experiments for comparison between MLMCMC and Metropolis. We refer the reader to the github repository associated with the project for the numerical results and plots.

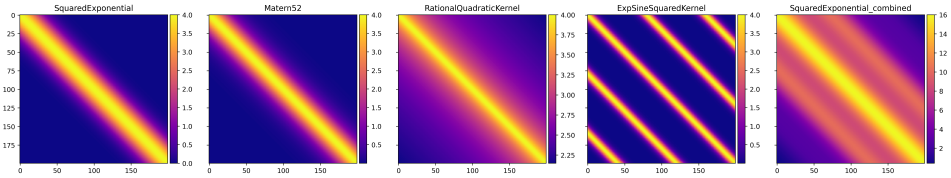


Figure 3.12: **Examples of kernel priors tested for the selection of parameters**

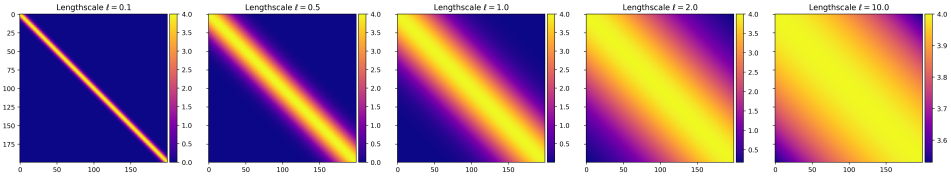


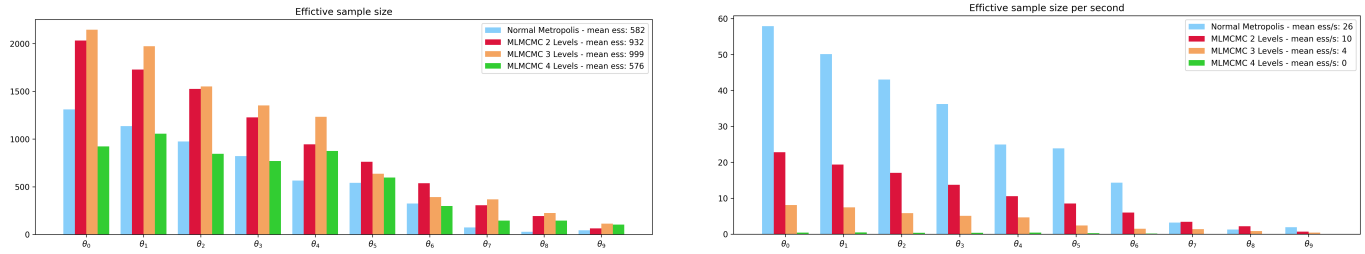
Figure 3.13: **Examples of length-scales for the Squared Exponential kernel tested for the selection of parameters**

3.3.2.1 Comparison of MLMCMC and Metropolis

With the given Gaussian process configuration, we performed inference on the KL-modes for four different models:

- Standard one-level model with 200 data points, using the Metropolis method.
- 2-level model with 200 data points at finest level, and half as many in the coarse level.
- 3-level model with 200 data points at finest level, and half as many in the coarser levels.
- 4-level model with 200 data points at finest level, and half as many in the coarser levels.

All the models had $\sigma = 0.3$ as noise level in the data. From figure 3.14 it is observed that the MLMCMC models with 2 and 3 levels respectively are able to sample more efficiently than standard Metropolis and MLMCMC with 4 levels which are about equal. This additional efficiency comes at the cost of runtime as the standard Metropolis is still the best in terms of ESS/s.
From figure 3.15 we observe that all models achieve a somewhat similar accuracy with the MLMCMC with 3 levels being slightly better than the others. This can also be verified by looking at the predictions of the Gaussian processes in figure 3.16.



(a) Effective sample size (b) Effective sample size per second

Figure 3.14: **ESS and performance for Metropolis and 2,3 and 4-level MCMC models**

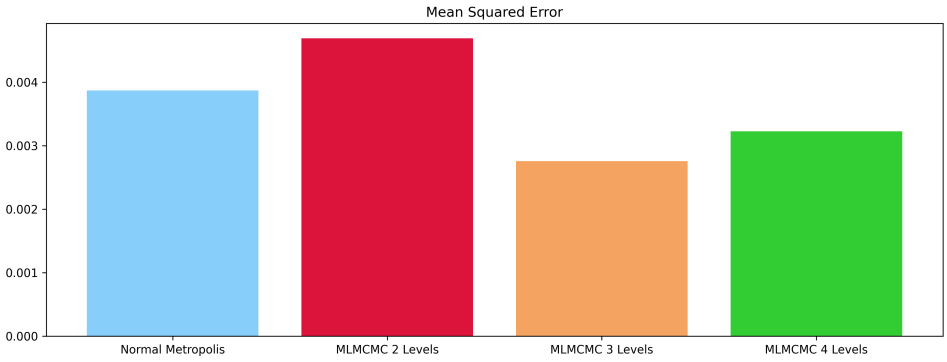


Figure 3.15: **Mean Square Error for Metropolis, 2-, 3- and 4-level MCMC models**

The resulting predictions in data space can be seen in figure 3.16, where all the resulting curves fit the target pretty well considering the noise added to the original data.

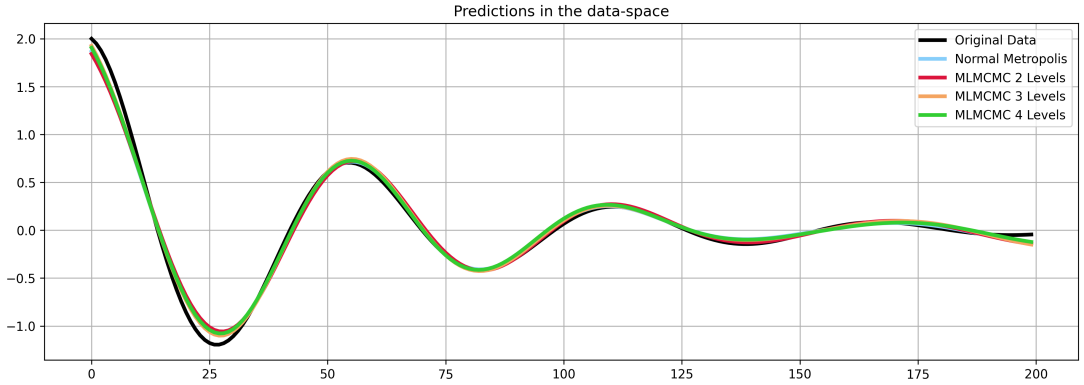


Figure 3.16: **Predictions of the spring system made by the Metropolis, 2-, 3- and 4-level MCMC models**

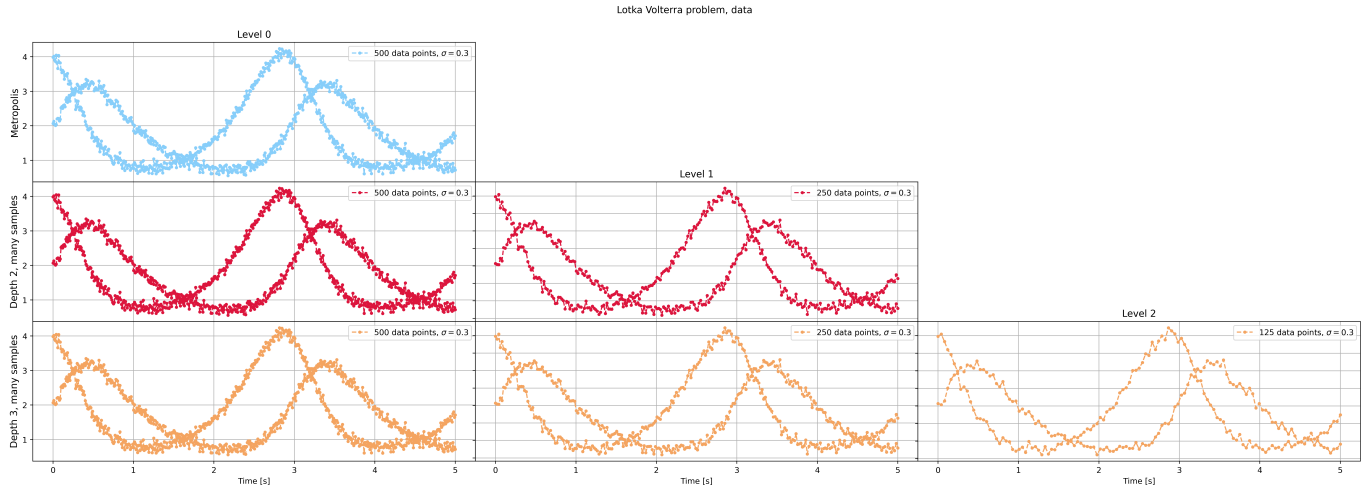


Figure 3.17: **System behaviour in different model configurations:** The three models are represented row wise, where the first one corresponds to the one-level Metropolis method, the second is the 2-level model, and the last one is the 3-level model.

3.4 The Lotka-Volterra system

The Lotka-Volterra equations are a two-dimensional system of non-linear first order differential equations that models the dynamical behavior of two interacting species, one of which is prey and the other a predator. Let

$$X := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

denote the two populations. The predator-prey dynamical system is governed by the ODE

$$\dot{X} = \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = f(X)$$

where

$$f(X) = \begin{bmatrix} x_1(\gamma_1 - \gamma_{12}x_2) \\ x_2(-\gamma_2 + \gamma_{21}x_1) \end{bmatrix}.$$

Note that the numerical solution of this problem is stable only if the rate parameters all are positive. We wish to make inference on the four parameters γ_1, γ_{12}, γ_2 and γ_{21}.

3.4.1 Likelihood of the problem

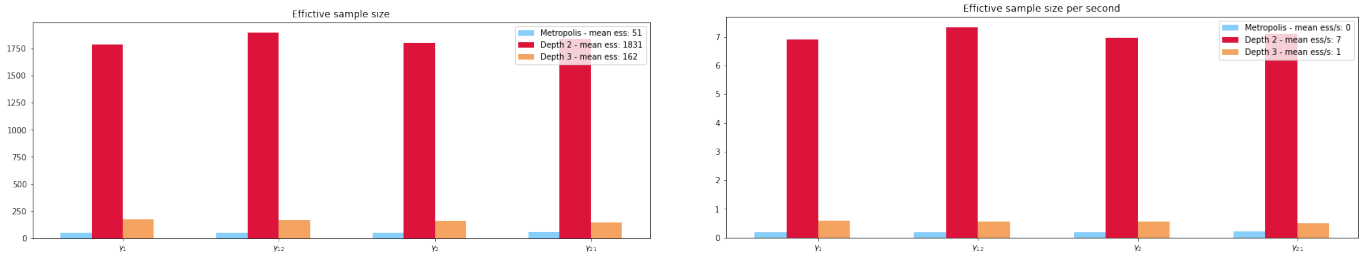
From numerous experiments we have discovered that the behavior of the predator-prey system is highly sensitive to changes in the rate parameters, especially if these parameters are close to zero. This will lead to a likelihood function over the possible values of parameters that has extremely large fluctuations. Say we fix a configuration of true parameter values. The likelihood of a model with slightly different parameter values can potentially be extremely small.

3.4.2 Metropolis vs. MLMCMC: Configurations

Using the previously gained knowledge of the multilevel method, we executed tests to compare Metropolis and MLMCMC on the Lotka-Volterra system. Efficiency of the methods have been compared by keeping the run-times of each simulation equal, by varying the amount of samples drawn. We used three models whose configurations are shown in figure 3.17. The first model has one level consisting of 500 data points, the second is a 2-level model with 500 points in the fine, and 250 in the coarse. The third model has 500, 250, 125 points in the levels respectively. The noise level is here 0.3. A multivariate Gaussian prior and a subsampling rate of 5 is used for all levels.

3.4.3 Metropolis vs. MLMCMC: Results

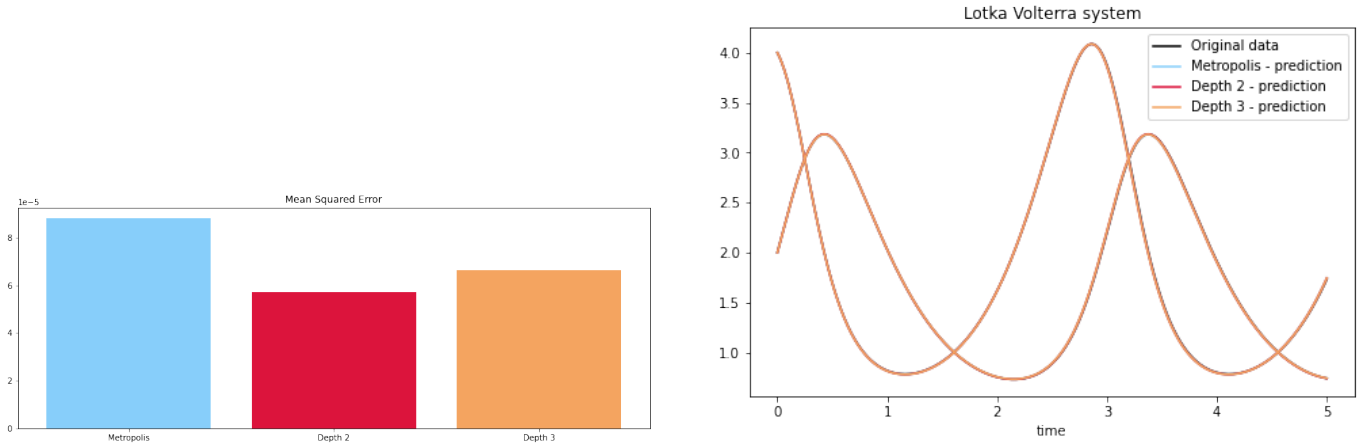
From figures 3.18 and 3.19 we see that both the ESS and ESS/s is highest for the two-level method. Combining this with the information about the MSE in figure 3.19 we see that the two-level method is the superior method. The trace



(a) Effective sample size

(b) Effective sample size per second

Figure 3.18: **ESS and performance for Metropolis and 2 and 3-level MCMC models**



(a) Mean square errors

(b) Predictions of the models

Figure 3.19: **MSE and predictions of the three models**

plots in figure 3.20 looks the best for the two-level method, and supports this conclusion. By increasing the noise from 0.3 to 0.4 in the data, another interesting result is achieved. As seen in figure 3.21 we now achieve a better accuracy for the same runtime for both the 2-level method and the 3-level one. Here, another dataset is included to show that including even more levels just reduces performance. This suggests that there exists some optimal number of levels for a specific problem with a specific amount of noise.

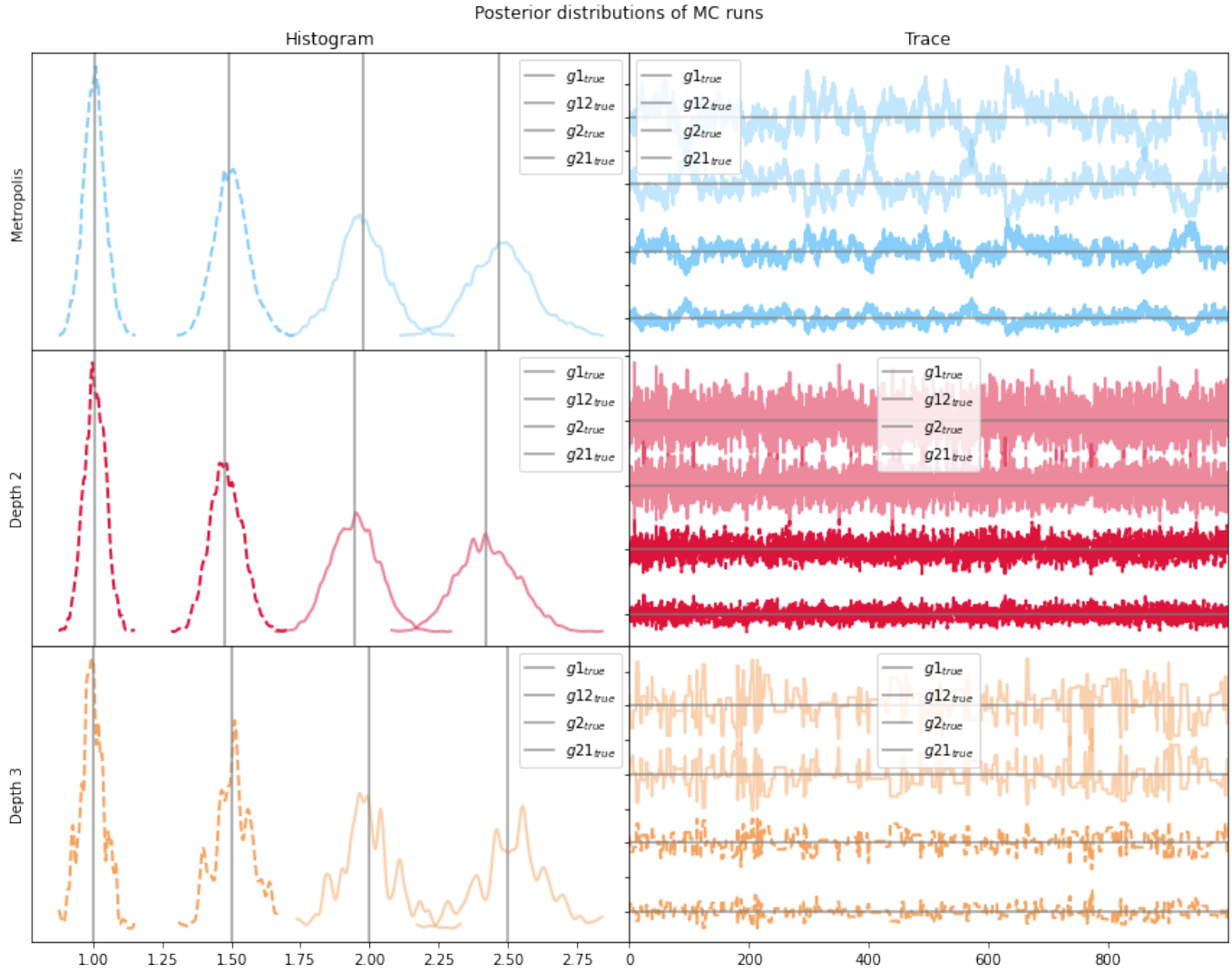
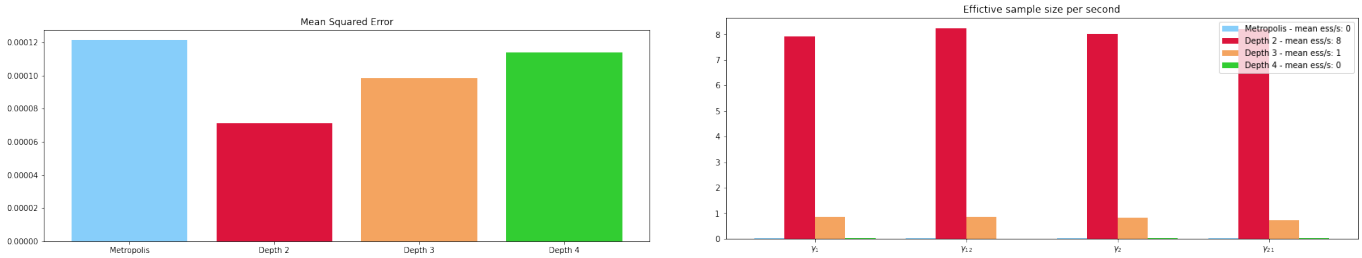


Figure 3.20: **Posterior distributions and trace plots of the parameters.**



(a) Mean square error

(b) Effective sample size per second

Figure 3.21: **MSE and ESS/s for the test with extra noise** From left to right in plot of MSE, the amount of coarse levels is increased from 0 to 3

Chapter 4

Conclusion

We have investigated the different properties of the MLMCMC method through the use of numerical experiments, and have been able to form an opinion as to what kind of sampler we should construct in order to obtain optimal parameter estimations. That is, by learning from the numerical experiments, we found out which method is more suitable in each of the two model problems discussed. In the spring problem, for instance, we saw that data-rich multilevel models samples more effectively from the posteriors of the parameters, and all the models provided quite similar MSEs. However, this is at the cost of a longer runtime. Thus in the case of the simple spring problem, the standard Metropolis approach is often preferred. Working with the Lotka-Volterra system has let us know that the theorized increase in efficiency is already taking place for a problem of this complexity. The multilevel approach performs better than the standard metropolis in all metrics, and is the preferred method for the Lotka-Volterra problem. Adding more noise to the data has shown that additional levels can be beneficial, and tells us that for a given problem there is some optimal number of level which is largely influenced by the noise in the data.

Bibliography

[1] T. J. Dodwell, C. Ketelsen, R. Scheichl, and A. L. Teckentrup, Multilevel Markov Chain Monte Carlo. SIAM Review 61(3), pp. 509–545, 2019.

[2] Juan Pablo Madrigal-Cianci, Fabio Nobile, Raul Tempone, Analysis of a class of Multi-Level Markov Chain Monte Carlo algorithms based on Independent Metropolis-Hastings, may 2021.

[3] PyMC development team, Multilevel gravity survey with MLDA, 2021.
https://docs.pymc.io/en/stable/pymc-examples/examples/samplers/MLDA_gravity_surveying.html

[4] A. Manzoni, Chapter 9: Statistical inverse problems and parameter estimation, 2020.
file:///C:/Users/simon/MTFYMA%204%20klasse%20(POLIMI)/Computational%20Statistics/ClassNotes_Chapter9%20(4).pdf

GitHub repository of the project:
https://github.com/RTuxen/MLMCMC