Chapter 5:
Bayesian Techniques for Statistical Inference

Danial Faghihi

March 30, 2022

Contents

1 Bayesian Inference:

The Bayesian inference differs significantly from the frequentist perspective summarized in the previous chapter. In the context of parameter estimation, the Bayesian approach is considered as a statistical inverse problem in which model parameters $\boldsymbol{\theta}$ are random variables, and their associated densities incorporate known information (prior) or information obtained from observational data (posterior). In particular, converting the Bayes' rule $P(A|B) = P(B|A)P(A)/P(B)$ into probability densities and letting B represent the observational data $\mathbf{D}$ and A represent the model parameters $\boldsymbol{\theta}$, we obtain the *Bayes's theorem of inverse problems* as [9, 8],

$$\pi_{post}(\boldsymbol{\theta}|\mathbf{D}) = \frac{\pi_{like}(\mathbf{D}|\boldsymbol{\theta})\pi_{prior}(\boldsymbol{\theta})}{\pi_{evid}(\mathbf{D})}. \tag{1}$$

where $\pi_{prior}(\boldsymbol{\theta})$ is called prior PDF, reflecting any prior knowledge that may be known about the parameters before the observational data is taken into account. This could come from previous similar experiments or models. If such prior information is not available (total ignorance), one can use *noninformative* prior as uniform distribution posed on the parameter support. In (1), $\pi_{like}(\mathbf{D}|\boldsymbol{\theta})$ is the likelihood of observing data $\mathbf{D}$ given parameter

realization $\boldsymbol{\theta}$. Hence, if we let $\pi(\boldsymbol{\theta}, \mathbf{D})$ be the joint density of parameters and data, then the likelihood,

$$\pi_{like}(\mathbf{D}|\boldsymbol{\theta}) = \frac{\pi(\boldsymbol{\theta}, \mathbf{D})}{\pi_{prior}(\boldsymbol{\theta})}.$$

As described in Chapter 4, the specification of the likelihood function depends on the noise model, i.e., the assumption made regarding the distributions of data and modeling errors. The solution of Bayesian inference is the posterior PDF $\pi_{post}(\boldsymbol{\theta}|\mathbf{D})$, indicating the updated parameters given observational data. In (1), $\pi_{evid}(\mathbf{D})$ is known as evidence PDF and is a normalization factor. Since $\int_\Theta \pi_{post}(\boldsymbol{\theta}|\mathbf{D})d\boldsymbol{\theta} = 1$ for all $\mathbf{D}$ in the space of observational data, then the evidence must be the marginalization of the numerator,

$$\pi_{evid}(\mathbf{D}) = \int_\Theta \pi_{like}(\mathbf{D}|\boldsymbol{\theta})\pi_{prior}(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{2}$$

Maximum a Posteriori (MAP) estimate. The posterior PDF $\pi_{post}(\boldsymbol{\theta}|\mathbf{D})$, provides the complete distribution of $\boldsymbol{\theta}$ based on the observations $\mathbf{D}$. A point estimates for the parameter value is the one that maximizes the posterior and is termed as the MAP estimate,

$$\boldsymbol{\theta}_{MAP} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}}\ \pi_{post}(\boldsymbol{\theta}|\mathbf{D}). \tag{3}$$

For a uniform prior MAP estimate is equivalent to Maximum Likelihood Estimate (MLE), $\boldsymbol{\theta}_{MAP} = \hat{\boldsymbol{\theta}}$.

1.1 Relation to frequentists approach

Here we provide a standard but fundamentally important example of calculating posterior in comparison with the frequentist approach. Recall the additive noise model,

$$\mathbf{D} = \mathbf{y}(\boldsymbol{\theta}) + \boldsymbol{\eta},$$

where $\mathbf{D}$ is data with size of n, $\mathbf{y}(\boldsymbol{\theta})$ is the model output, and $\boldsymbol{\eta}$ is the total error (or data noise in absence of modeling error). Let the error be Gaussian with zero mean $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ being the $n \times n$ covariance matrix. The likelihood function is then given by:

$$\pi_{like}(\mathbf{D}|\boldsymbol{\theta}) \propto \exp\left\{-\frac{1}{2}\left\|\mathbf{D} - \mathbf{y}(\boldsymbol{\theta})\right\|^2_{\boldsymbol{\Sigma}}\right\} \tag{4}$$

where $\|\cdot\|^2_{\boldsymbol{\Sigma}} = (\ \cdot\ )^T\boldsymbol{\Sigma}^{-1}(\ \cdot\ )$ and the proportional symbol $\propto$ is used to simplify the notation and avoid showing the associated constants. We note that if $\mathbf{y}(\boldsymbol{\theta})$ is linear in $\boldsymbol{\theta}$, the likelihood in (4) is Gaussian; otherwise, it is non-Gaussian with respect to $\boldsymbol{\theta}$.

Now let the covariance matrix be diagonal $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$ and the model be linear, i.e., $\mathbf{y}(\boldsymbol{\theta}) = \mathbf{G}\boldsymbol{\theta}$, with $\mathbf{G}$ being a $n \times p$ matrix. Then the MLE is

$$\hat{\boldsymbol{\theta}} = (\mathbf{G}^T\sigma^{-1}\mathbf{G})^{-1}\mathbf{G}^T\sigma^{-1}\mathbf{D}. \tag{5}$$

If the prior $\pi_{prior}(\boldsymbol{\theta})$ is also Gaussian, the posterior becomes

$$\pi_{post}(\boldsymbol{\theta}|\mathbf{y}) \propto \exp\left\{-\frac{1}{2}\left(\|\mathbf{D} - \mathbf{y}(\boldsymbol{\theta})\|^2_{\boldsymbol{\Sigma}} + \|\boldsymbol{\theta} - \boldsymbol{\theta}_{pr}\|^2_{\boldsymbol{\Sigma}_{\text{prior}}}\right)\right\}, \tag{6}$$

where $\boldsymbol{\Sigma}_{\text{prior}}$ being the covariance matrix of the prior density, $\boldsymbol{\theta}_{pr}$ being the mean of the prior, and $\|\boldsymbol{\theta} - \boldsymbol{\theta}_{pr}\|^2_{\boldsymbol{\Sigma}_{\text{prior}}} = (\boldsymbol{\theta} - \boldsymbol{\theta}_{pr})^T\boldsymbol{\Sigma}^{-1}_{\text{prior}}(\boldsymbol{\theta} - \boldsymbol{\theta}_{pr})$. The likelihood corresponds to the misfit function (sum of squares error in OLS) while prior is seen to provide a Tikhonov-type regularization.

1.2 Bayesian Central Limit Theorem

We now discuss the asymptotic behavior of Bayesian methods in cases in which the number n of data samples becomes arbitrarily large. One of the most interesting aspects of the Bayesian approach is the so-called *Bayesian learning*. Suppose a single observation $\mathbf{D}_1$ is recorded. Beginning with a prior $\pi(\boldsymbol{\theta})$, we update $\boldsymbol{\theta}$ throughout the calculation of the posterior:

$$\pi(\boldsymbol{\theta}|\mathbf{D}_1) \propto \pi(\mathbf{D}_1|\boldsymbol{\theta})\pi(\boldsymbol{\theta}). \tag{7}$$

When new additional data $\mathbf{D}_2$ is attained, the posterior becomes the prior for another posterior update:

$$\pi(\boldsymbol{\theta}|\mathbf{D}_1, \mathbf{D}_2) \propto \pi(\mathbf{D}_2|\boldsymbol{\theta}, \mathbf{D}_1)\pi(\boldsymbol{\theta}|\mathbf{D}_1). \tag{8}$$

Continuing in this way, we "learn" more about $\boldsymbol{\theta}$ with each new observation (see Figure 1),

$$\pi(\boldsymbol{\theta}|\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_n) \quad \propto \quad \left(\prod_{i=1}^{n} \pi(\mathbf{D}_i|\boldsymbol{\theta})\right) \pi(\boldsymbol{\theta}) \tag{9}$$

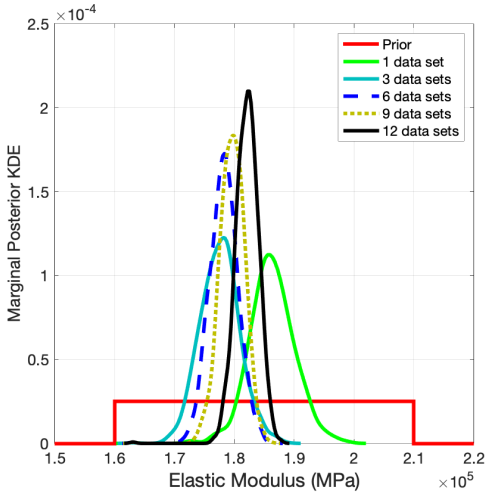


Figure 1: Prior and posterior distributions of Young's modulus parameter of linear elasticity model, inferred from a different number of stress-strain measurements.

A fundamental question that arises is what is the limiting posterior distribution attained as $n \to \infty$? The answer to this question is captured by Bernstein–von Mises theorem for the case in which the model is correctly specified [2, 14]). Suppose that $\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_n$ are i.i.d. samples from a probability density, prior density is $\pi(\boldsymbol{\theta})$, and likelihood density is $\pi(\mathbf{D}^n|\boldsymbol{\theta}) = \prod_{i=1}^{n} \pi(\mathbf{D}_i|\boldsymbol{\theta})$. Let both $\pi(\boldsymbol{\theta})$ and $\pi(\mathbf{D}^n|\boldsymbol{\theta})$ be twice differentiable near the MLE $\hat{\boldsymbol{\theta}}$. Then, as $n \to \infty$, the posterior density converges in distribution, to the normal distribution

$$\pi(\boldsymbol{\theta}|\mathbf{D}^n) = \frac{\pi(\mathbf{D}^n|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\pi(\mathbf{D}^n)} \xrightarrow{D} \mathcal{N}(\hat{\boldsymbol{\theta}}, \mathbf{I}(\hat{\boldsymbol{\theta}})^{-1}), \tag{10}$$

where $\mathbf{I}(\hat{\boldsymbol{\theta}})$ is the Fisher information matrix.

2 Markov Chain Monte Carlo (MCMC) methods

Bayesian inference requires computing the posterior distribution as the solution of the statistical inverse problem, given the parameters priors and the likelihood PDF. The leading numerical method for the Bayesian inference solution is the Markov Chain Monte Carlo (MCMC) approach that uses attributes of the posterior density to specify parameter values that adequately explore the geometry of the distribution.

Monte Carlo Methods. The Monte Carlo methods were introduced by Nicholes Metropolis and Stanislaw Ulam in 1949, Metropolis et al. in 1953 [10], and with further improvements by Wilfred Hastings in 1970 [7][1]. The basic idea involves solving deterministic problems using a probabilistic analogy. The most common Monte-Carlo (MC) algorithm is that associated with integration of a function $f(x)$ over a domain Ω, $\int_\Omega f(x)dx$. There the idea is to introduce a set X of randomly selected points $X = \{x_1, x_2, ..., x_n\}$ in Ω and approximate the integral with the sum,

$$\int_\Omega f(x)dx \approx \frac{|\Omega|}{n}\sum_{i=1}^{n} f(x_i) \ , \tag{11}$$

$|\Omega|$ being the volume of Ω and the approximation error is $O(\frac{1}{\sqrt{n}})$. In practical applications, it is sufficient to employ a pseudo-random set of sample points, that are statistically random, despite having been produced by a completely deterministic and repeatable process.

Instead of choosing x_i via pseudo-random numbers, it is more efficient to use a low-discrepancy sequence such as the Halton sequence or the Sobol sequence. This is known as quasi-Monte Carlo methods that have rates of convergence close to $O(\frac{1}{n})$.

Markov chain. A stochastic process satisfies the Markov property if the probability of future states is dependent only on the present state rather than the sequence of past events that precede it. A Markov chain is a sequence of random variables $X_k, k = 0, 1, \cdots, m$ that satisfy the Markov property that X_{k+1} depends only on X_k; that is

$$P(X_{k+1} = x_{k+1}|X_0 = x_0, \cdots, X_k = x_k) = P(X_{k+1} = x_{k+1}|X_k = x_k),$$

that shows the *memoryless-ness* property of a Markov chain. A distribution is said to be invariant or stationary w.r.t a Markov chain if the transition function of that chain leaves that distribution unchanged.

[1] Ulam invented the Monte Carlo method while he was working on nuclear weapons projects at the Los Alamos National Laboratory. Immediately after Ulam'sUlam's breakthrough, John von Neumann understood its importance in studying neutron diffusion in the core of a nuclear weapon in which deterministic mathematical methods were unable to solve the problem. Being secret, the work of von Neumann and Ulam required a code name. Their colleague, Nicholas Metropolis, suggested using the name Monte Carlo, which refers to the Monte Carlo Casino in Monaco where Ulam'sUlam's uncle would borrow money from relatives to gamble. Monte Carlo methods were central to the simulations required for the Manhattan Project. von Neumann and Metropolis and others programmed the ENIAC computer to perform the first fully automated Monte Carlo calculations, of a fission weapon core, in the spring of 1948. In the 1950s Monte Carlo methods were used at Los Alamos for the development of the hydrogen bomb and became popularized in other fields of physics.

Bayes'Bayes' equation can be solved by constructing Markov chains whose stationary (equilibrium) distribution is the posterior density. By evaluating realizations of the chain, one thus samples the posterior and hence obtains a density for the parameter values based on observed measurements. The general strategy of most MCMC methods to sample posterior density is:

1. Consider the parameter $\boldsymbol{\theta}^{(k)}$ to be specified and take it as the current chain realization.
2. Propose a new value $\boldsymbol{\theta}^* \sim J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)})$, where J is called proposal or jumping distribution. The notation indicates that J specifies $\boldsymbol{\theta}^*$ based on the previous value $\boldsymbol{\theta}^{(k-1)}$ and $J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)})$ should not be interpret as a conditional density.
3. With probability $\alpha(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k-1)})$, determined by the properties of the likelihood function and prior, accept $\boldsymbol{\theta}^*$; otherwise take $\boldsymbol{\theta}^{(k-1)}$.
4. Establish that the posterior density is the stationary distribution for the chain.

2.1 Random Walk Metropolis Algorithm

The proposal density can be of any form that is easy to sample, such as multivariate Gaussian with a mean equal to the current sample $\boldsymbol{\theta}^{(k-1)}$. We consider the proposal distribution to be symmetric in the sense that $J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)}) = J(\boldsymbol{\theta}^{(k-1)}|\boldsymbol{\theta}^*)$, such as

$$J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)}) = \mathcal{N}(\boldsymbol{\theta}^{(k-1)}, \mathbf{V}),$$

where $\mathbf{V}$ is the covariance matrix for $\boldsymbol{\theta}$. This choice of proposal results in the Metropolis algorithm summarized as follows.

Random Walk Metropolis Algorithm

1. Initialization: choose an initial value of the parameter $\boldsymbol{\theta}^{(0)}$.
2. For $k = 1, \cdots, M$

   (i) For $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$, construct the candidate

   $$\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(k-1)} + \mathbf{z},$$

   This ensures $\boldsymbol{\theta}^* \sim \mathcal{N}(\boldsymbol{\theta}^{(k-1)}, \mathbf{V})$. Because the construction of $\boldsymbol{\theta}^*$ takes into account $\boldsymbol{\theta}^{(k-1)}$, this is termed a *random walk*.

   (ii) Compute the ratio

   $$r(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)}) = \frac{\pi(\boldsymbol{\theta}^*|\mathbf{D})}{\pi(\boldsymbol{\theta}^{(k-1)}|\mathbf{D})} = \frac{\pi_{like}(\mathbf{D}|\boldsymbol{\theta}^*)\pi_{prior}(\boldsymbol{\theta}^*)}{\pi_{like}(\mathbf{D}|\boldsymbol{\theta}^{(k-1)})\pi_{prior}(\boldsymbol{\theta}^{(k-1)})}$$

   (iii) Set

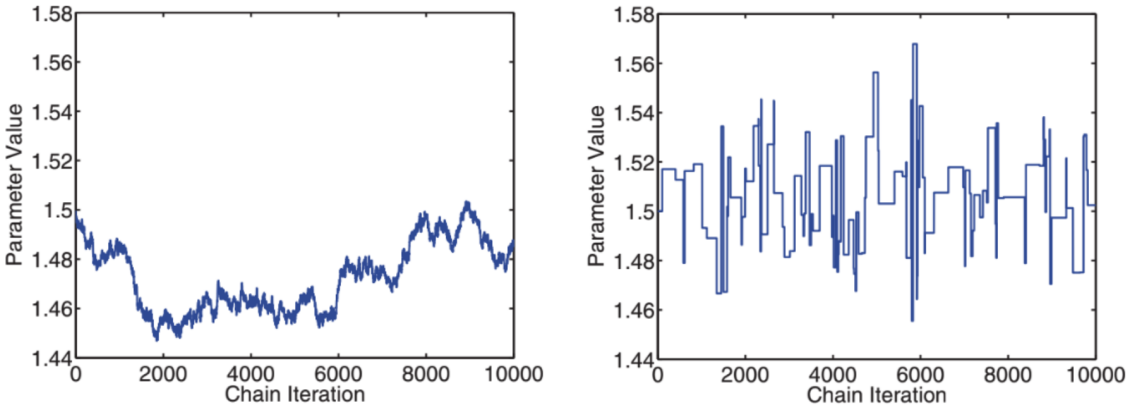
   $$\boldsymbol{\theta}^{(k)} = \begin{cases} \boldsymbol{\theta}^* & , \quad \text{with probability } \alpha = \min(1, r) \\ \boldsymbol{\theta}^{(k-1)} & , \quad \text{else.} \end{cases}$$

   That is, we accept $\boldsymbol{\theta}^*$ with probability 1 if $r \geq 1$ and we accept it with probability r if $r < 1$.

Remarks:

- The choice of acceptance criteria indicates that if a candidate $\boldsymbol{\theta}^*$ that yields $\pi_{like}(\mathbf{D}|\boldsymbol{\theta}^*) > \pi_{like}(\mathbf{D}|\boldsymbol{\theta}^{(k-1)})$ this candidate is accepted with probability of one. However, if $\pi_{like}(\mathbf{D}|\boldsymbol{\theta}^*) < \pi_{like}(\mathbf{D}|\boldsymbol{\theta}^{(k-1)})$, we accept the candidate $\boldsymbol{\theta}^*$ with probability $\alpha = r$. This ensures that samples are not concentrated in the high likelihood regions and the tails of the posterior are also sampled.

- The properties of the proposal function and how they affect mixing are illustrated in Figure 2. If the variance is too large, a large percentage of the candidates will be rejected since they will have a smaller likelihood, and hence the chain will stagnate for long periods. The acceptance ratio will be high if the variance is small, but the algorithm will be slow to explore the parameter space. Interactive visualization of the random walk Metropolis and other MCMC methods is provided in `http://chi-feng.github.io/mcmc-demo/`.



(a) (b)

Figure 2: Chains resulting from proposal distribution with the variance that are (a) too small and (b) too large.

- For models in which the parameter scales vary by several orders of magnitude, one typically employs the scaled parameter to efficiently explore highly anisotropic posterior via an isotropic proposal distribution.

- To improve the efficiency of the MCMC algorithm, one may use deterministic parameter inference from data to assign initial values of the parameters $\boldsymbol{\theta}^{(0)}$.

Noise model and error variance. As indicated in the previous chapter, the choice of noise model directly influence the inference process. Let observational data $\mathbf{D}$ be n i.i.d. samples. If we ignore the model inadequacy, then the additive noise model only includes data uncertainty, such that

$$\mathbf{D}_i = \mathbf{y}_i(\boldsymbol{\theta}) + \boldsymbol{\epsilon}_i, \quad i = 1, 2, \cdots, n.$$

Under the assumption $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma_i^2),\ i = 1, 2, \cdots, n$, the llikelihood function becomes,

$$\pi_{like}(\boldsymbol{\theta}|D) = \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \right) \exp\left(-\sum_{i=1}^{n} \frac{(\mathbf{D}_i - \mathbf{y}_i)^2}{2\sigma_i^2} \right),$$

where σ_{data} may be associated with the error bars of the measurement data. Now consider the noise model, including the modeling error,

$$\mathbf{D}_i = \mathbf{y}_i(\boldsymbol{\theta}) + \boldsymbol{\epsilon}_i + \boldsymbol{\gamma}_i.$$

Assuming zero mean Gaussian distributions for both data uncertainty $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2_{data}\mathbf{I})$ and modeling error $\boldsymbol{\gamma}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2_{model}\mathbf{I})$, one can write the total error as,

$$\boldsymbol{\epsilon}_i + \boldsymbol{\gamma}_i = \boldsymbol{\eta}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}),$$

where $\sigma^2 = \sigma^2_{model} + \sigma^2_{data}$. In this case, the log-likelihood function is

$$\pi_{like}(\boldsymbol{\theta}|\mathbf{D}) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\sum_{i=1}^{n}(\mathbf{D}_i - \mathbf{y}_i)^2}{2\sigma^2}\right), \tag{12}$$

where $\mathbf{D} = [\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_n]$ and σ^2 is usually unknown and called hyper-parameter. One can treat this hyper-parameter as an additional random parameter whose density is sampled through realizations of the Markov chain. We note that the likelihood (12) is in the inverse-gamma family, so the conjugate prior is

$$\pi_{prior}(\sigma^2) \propto (\sigma^2)^{-(\alpha+1)} \exp(\beta/\sigma^2),$$

where α and β are additional hyper-parameters that can be tuned for a particular application. The resulting posterior is

$$\pi_{post}(\sigma^2|\boldsymbol{\theta}, \mathbf{D}) \propto (\sigma^2)^{-(\alpha+1+n/2)} \exp\left(-\frac{\beta + 0.5\sum_{i=1}^{n}(\mathbf{D}_i - \mathbf{y}_i)^2}{\sigma^2}\right),$$

so that

$$\sigma^2 \sim Inv - gamma\left(\alpha + \frac{n}{2}, \beta + \frac{\sum_{i=1}^{n}(\mathbf{D}_i - \mathbf{y}_i)^2}{2}\right).$$

According to [3], 2α can be interpreted as representing the number of observations that provide the information encoded in the prior, whereas β/α represents the mean square error of the observations. In practice, one often takes 2α =0.01 to 1, which corresponds to noninformative prior.

Convergence criteria. It can be shown (see, e.g., [13]) that if MCMC chains are run sufficiently long, they will produce samples from the posterior density. But how long chains must be run to converge to and adequately sample from the posterior? Unfortunately, an analytical convergence theory and stopping criteria are lacking for MCMC. However, there are various diagnostics (tests) that can be used to establish confidence in MCMC simulations.

1 - The most direct method for assessing convergence is to visually or statistically monitor the marginal path associated with each parameter. The initial period during which means appear to transition is often termed the *burn-in period*, and these values are excluded when computing posterior PDFs since they are not sampled from the stationary or posterior distribution.

2 - The *acceptance ratio*, is the percentage of accepted candidates and can be used to quantify whether or not the chain is adequately sampling from the posterior. The optimal acceptance ratio depends on the geometry of the posterior, but values between 0.1

and 0.5 are often considered acceptable. The acceptance ratio can also be used to tune the proposal density to improve chain mixing, i.e., moving through the support of the posterior distribution rapidly.

3 - A commonly employed statistical test is the *autocorrelation*, between components in the same chain that are k iterations apart,

$$R(k) = \frac{cov(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(i+k)})}{Var(\boldsymbol{\theta}^{(i)})}.$$

Because adjacent components are likely correlated, this test can be used to establish that the chain is producing iid samples from the posterior. Low autocorrelation is often indicative of fast mixing of the chain and efficient sampling.

4 - Another MCMC test is the multivariate potential scale reduction factor (MPSRF), which is a multivariate extension of the potential scale reduction factor (PSRF) defined by the square root of the ratio between the pooled variance estimate and the within-sequence variance [4, 3]. A MPSRF close to one indicates that chains have fully explored the parameter space over the support of the posterior distribution.

Example. In this example, we consider 1D plasticity model in which stress T and strain e, are related through

$$e = \frac{T}{E} + 0.002\left(\frac{T}{T_y}\right)^{3.5},$$

where E is the Young'sYoung's modulus and T_y is the yield stress. We would like to estimate the parameters $\boldsymbol{\theta} = [E, Ty]$. Here, we use synthetic data generated by the model. To this end, we take the true parameters $E = 150 \times 10^3$ and $T_y = 310$ and use the forward model to generate n=10 data points, $e_i^{data}, T_i^{data},\ i = 1, 2 \cdots, n$. The 10 stress-strain data set $\mathbf{D}$ is then created by adding the random noise $\eta \sim \mathcal{N}(\mathbf{0}, \sigma_{data}^2\mathbf{I})$ to the strain values with $\sigma_{data} = 0.001$. Figure 3 indicates the generation of synthetic data from true model parameters.

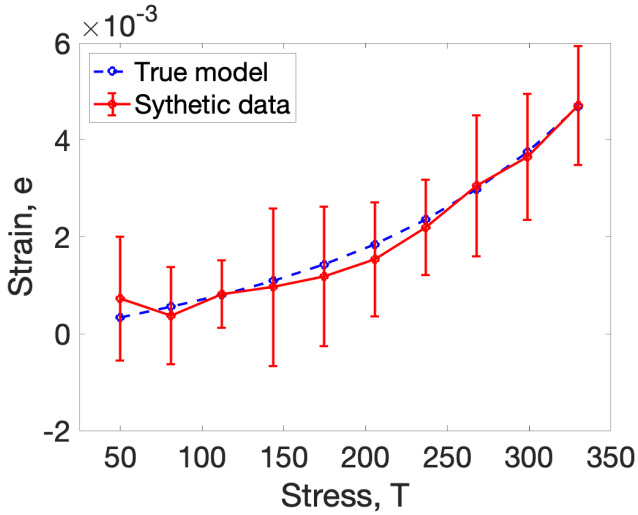


Figure 3: Synthetic data generated from the forward model was contaminated with noise with a standard deviation of 0.001. Error bars indicate the mean and variance of 10 synthetic stress-strain curves.

The inverse crime. *The inverse crime occurs when the same theoretical ingredients are employed to synthesize as well as to invert data in an inverse problem. However, the inversion scheme is often tested using the synthetic data generated from the same model.*

In this example, we take Gaussian priors for the parameters $\pi_{prior}(E) = \mathcal{N}(150 \times 10^3, (6.7 \times 10^4)^2)$ and $\pi_{prior}(T_y) = \mathcal{N}\left(300, (34)^2\right)$. Since the noise level is known, we can write the likelihood function as

$$\pi_{like}(\mathbf{D}|\boldsymbol{\theta}) \propto -\frac{1}{2}\sum_{i=1}^{n}\left[-\frac{e_i^{data} - e_i^{model}}{\sigma_{data}}\right]^2,$$

where the strains evaluated by the model are,

$$e_i^{model} = \frac{T_i^{data}}{E} + 0.002\left(\frac{T_i^{data}}{T_y}\right)^{3.5}, \quad i = 1, 2\cdots, n.$$

We then estimate the posterior distributions $\pi_{post}(\boldsymbol{\theta}|\mathbf{D})$ using the random walk Metropolis algorithm (see the MATLAB code `MH_plasticity.m`). Figures 4 and 5 shows the marginal path (trace) of the chain and marginal posterior distributions of parameters. Note that due to high data uncertainty, the prior is dominated the inference process, specifically for the parameter E. Matlab function `cov` calculates the covariance matrix for a data matrix (where each column represents a separate quantity), and `corr` calculates the correlation matrix. We also used Matlab function `autocorr` to plot the autocorrelation of the parameters in Figure 6.

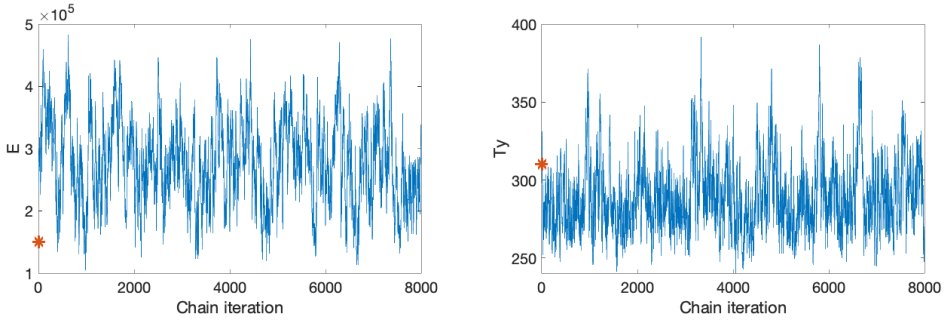


Figure 4: Marginal paths of parameters E and T_y of the plasticity model.

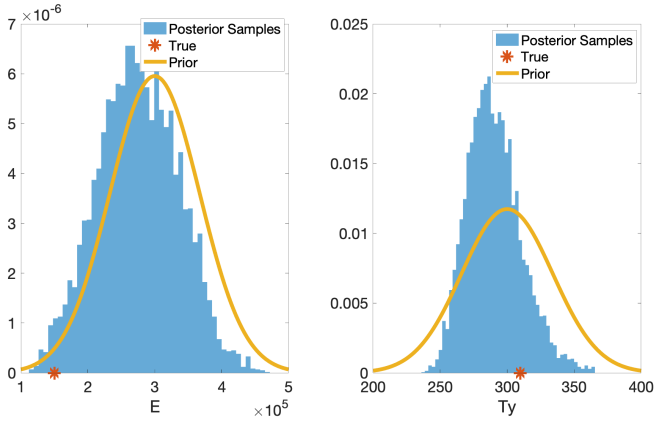
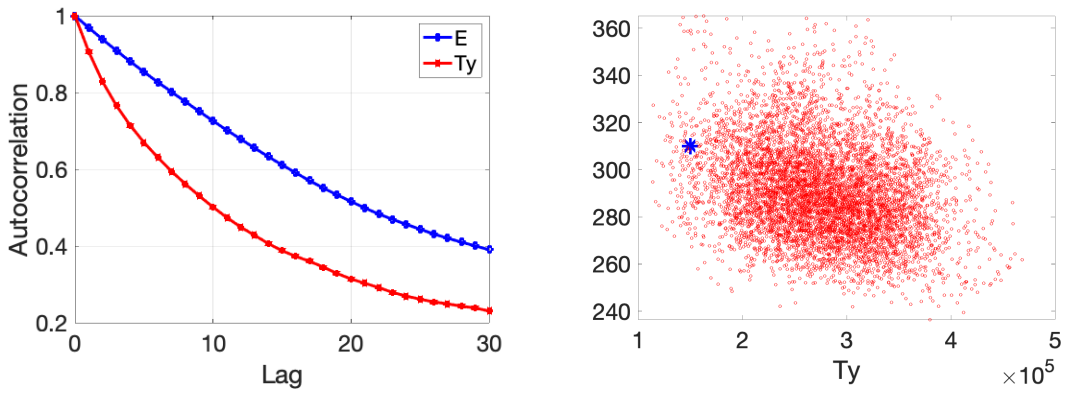


Figure 5: Marginal posterior distributions of parameters E and T_y of the plasticity model.



(a) (b)

Figure 6: Chain autocorrelation and posterior samples from MH algorithm with initial chain at the parameters mean.

Next, we propagate the parametric uncertainty through the forward model and using the posterior samples. Figure 7 shows the comparison of the calibrated model and the stress-strain data.

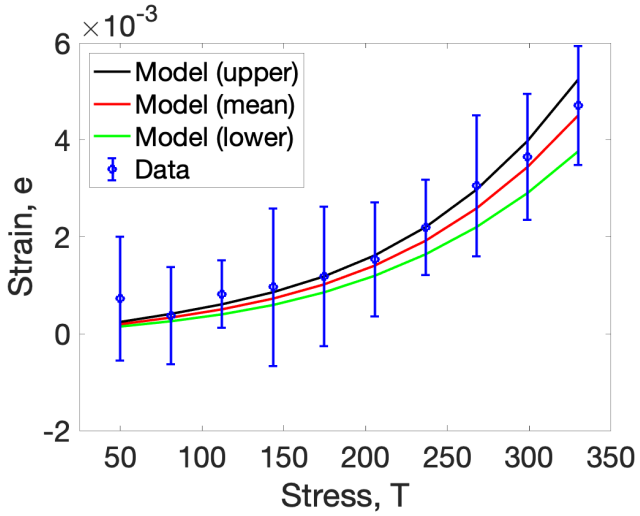


Figure 7: Comparing the plasticity model prediction and data.

You can leverage the Matlab code of this example and explore the effect of data noise σ_{data}, Metropolis step size and initial chain position in the chain mixing, rejection ratio, and posterior distributions

Matlab code for Metropolis algorithm for the example problem.

```
%% Metropolis algorithm
% This code illustrates the implementation of the Metropolis algorithm for
% plastic deformation model, e: strain, T: stress
%
%     e = T/E + 0.002*(T/Ty)^3.5, where E and Ty are model parameters
%
% Here we are sampling the Young's modulus and yield stress
% parameters theta = [E,Ty]

```

```
clear all; close all; clc


%% Define forward model
% plastic deformation model is defined as an anonymous function

forward = @(T,parameters)  T./parameters(1) + 0.002*(T./parameters(2)).^3.5
    ;



%% Denerate synthetic data from forward model

% true parameters
E = 150e3;
Ty = 310;

% true stress-strain points
n = 10;
T_true = linspace(50, 330, n);
e_true = forward(T_true,[E, Ty]);

% adding fixed data noise to the true strain
num_exprmt = 10; % number of stress-strain curves
data_stDev = 0.001;

e_exprmts=zeros(num_exprmt,length(e_true));
for i = 1:num_exprmt
    e_exprmts(i,:) = e_true + normrnd(0,data_stDev, size(e_true));
end
e_data_mean = mean(e_exprmts);
e_data_stDev = std(e_exprmts);

% plot synthetic data
figure()
plot(T_true, e_true, '--bo', 'LineWidth', 2); hold on;
errorbar(T_true, e_data_mean, e_data_stDev, '-ro', 'LineWidth', 2);
xlim([25,350])
xlabel('Stress, T')
ylabel('Strain, e')
legend('True model','Sythetic data','location','northwest')
set(gca,'FontSize',24)


%% Define Gaussian prior

% parameter1: E
E_min  = 100E3;          % lowerbound
E_max  = 500E3;          % upperbound
E_prior_mean  = 300E3;   % prior mean
E_prior_stDev = 6.7E4;   % prior standard deviation

% parameter1: Ty
Ty_min  = 200;          % lowerbound
Ty_max  = 400;          % upperbound
Ty_prior_mean  = 300;   % prior mean
Ty_prior_stDev = 34; % prior standard deviation

% rearrange prior infos
```

```
prior_mean = [E_prior_mean Ty_prior_mean];
prior_var  = diag([E_prior_stDev Ty_prior_stDev].^2);
lobounds   = [E_min Ty_min];
upbounds   = [E_max Ty_max];


%% Define inputs of Metropolis: initial point and proposal variance

% start at prior means by default
MH_initial = [E_prior_mean  Ty_prior_mean];

% MH step size, assumed 1% of prior range here
proposal_variance = [(E_max-E_min)*0.05   (Ty_max-Ty_min)*0.05 ].^2;


%% Define log-likelihood function

misfit = @(sample) (forward(T_true,sample) - e_data_mean).^2 ./ (e_data_stDev.^2) ;
loglike = @(sample) -sum( 0.5*misfit(sample) );


%% Construct a Metropolis chain of length N

N = 10000;
Nparam = length(MH_initial);

% initilize parameter samples
samples = zeros(N,Nparam)*nan;

% initial sample at MH start
samples(1,:) = MH_initial;

loglike_current = loglike(samples(1,:));
logprior_current = log(  mvnpdf(samples(1,:), prior_mean, prior_var)  );

for i = 2:N

    % proposed a parameter sample
    % resample if the sample is not in prior bounds
    in_range = 0;
    while ~in_range
        proposed_sample = samples(i-1,:) + mvnrnd( [0,0],  diag( proposal_variance)  );
        if sum(  (proposed_sample>upbounds) + (proposed_sample<lobounds) ) == 0
            in_range = 1;
        end
    end

    loglike_proposal = loglike(proposed_sample);
    logprior_proposal = log(  mvnpdf(proposed_sample, prior_mean, prior_var)  );


    % acceptance ratio
    accept =  exp(  (loglike_proposal + logprior_proposal) - ( loglike_current + logprior_current)  );

```

```
    if  rand(1) < min(1,accept)
        % accept the proposed sample and update the current infos
        samples(i,:) = proposed_sample;
        loglike_current  = loglike_proposal;
        logprior_current = logprior_proposal;

    else
        % reject proposal, stay at the same sample
        samples(i,:) = samples(i-1,:);

    end
end

% burn-in period: 20% of the samples
burn_in = round(N*0.2);


%% Plot MH results


% marginal paths of the chain
figure()
title('Marginal paths')
subplot(1,2,1)
plot(samples(burn_in+1:end,1)); hold on;
plot(0,E, '*', 'Markersize', 15,'LineWidth',3);
xlabel('Chain iteration')
ylabel('E')
set(gca,'FontSize',20)

subplot(1,2,2)
plot(samples(burn_in+1:end,2)); hold on;
plot(0,Ty, '*', 'Markersize', 15,'LineWidth',3);
xlabel('Chain iteration')
ylabel('Ty')
set(gca,'FontSize',20)

% marginal posterior distribution
figure()
title('Marginal posterior distributions of parameters')
subplot(1,2,1)
histogram(samples(burn_in+1:end,1),50,'edgecolor','none','Normalization','
    pdf'); hold on;
plot(E,0,'*', 'Markersize', 15,'LineWidth',3);
Elocs = linspace(lobounds(1),upbounds(1),200);
plot(Elocs,normpdf(Elocs, prior_mean(1), sqrt(prior_var(1,1))  ),'LineWidth
    ',5);
xlim([E_min, E_max])
xlabel('E')
legend('Posterior Samples','True','Prior','location','northwest')
set(gca,'FontSize',20)

subplot(1,2,2)
histogram(samples(burn_in+1:end,2),50,'edgecolor','none','Normalization','
    pdf'); hold on;
plot(Ty,0,'*', 'Markersize', 15,'LineWidth',3);
Tylocs = linspace(lobounds(2),upbounds(2),200);
plot(Tylocs,normpdf(Tylocs, prior_mean(2), sqrt(prior_var(2,2))  ),'
    LineWidth',5);
```

```
xlim([Ty_min, Ty_max])
xlabel('Ty')
legend('Posterior Samples','True','Prior','location','northwest')
set(gca,'FontSize',20)

% posterior samples
figure
plot(E, Ty, 'b*', 'Markersize', 15,'LineWidth',3)
hold on
plot(samples(burn_in+1:end,1), samples(burn_in+1:end,2), 'ro', 'Markersize'
    , 3);
xlabel('E'); xlabel('Ty')
set(gca,'FontSize',24)


% autocorelation
figure
nlags =30;
[ACF_E,lags,bounds] = autocorr(samples(1:end,1), nlags, 0);
[ACF_Ty,lags,bounds]= autocorr(samples(1:end,2), nlags, 0);
plot(lags,ACF_E,'bo-',lags,ACF_Ty,'r*-','linewidth',3);
ylabel('Autocorrelation');
xlabel('Lag');
grid on;
legend('E','Ty');
set(gca,'FontSize',24)


% Covariance and Correlation Matrices
cov_matrix  = cov(samples)
corr_matrix = corr(samples)



%% Forward uncertianty propagation

fwds = [];
for i = burn_in+1:N
    fwds(end+1,:) = forward(T_true,samples(i,:));
end

fwd_mean = mean(fwds);
fwd_stDev  = std(fwds);


% plot data and model prediction
figure()
plot(T_true,fwd_mean+fwd_stDev, 'k', 'LineWidth', 2);hold on;
plot(T_true,fwd_mean, 'r', 'LineWidth', 2);
plot(T_true,fwd_mean-fwd_stDev, 'g', 'LineWidth', 2);
errorbar(T_true, e_data_mean, e_data_stDev, 'ob', 'LineWidth', 2);
xlim([25,350])
xlabel('Stress, T')
ylabel('Strain, e')
legend('Model (upper)','Model (mean)', 'Model (lower)',...
    'Data','location','northwest')
set(gca,'FontSize',24)
```

2.2 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm generalizes the Metropolis algorithm to include non-symmetric proposal function, such as chi-squared distributions,

$$J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)}) = \kappa(\boldsymbol{\theta}^*)^{\frac{k}{2}-1} \exp\left(\frac{\boldsymbol{\theta}^*}{2}\right).$$

In this case, candidates are accepted with probability $\alpha = \min(1, r)$, where the acceptance ration is

$$r(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)}) = \frac{\pi(\boldsymbol{\theta}^*|\mathbf{D})/J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)})}{\pi(\boldsymbol{\theta}^{(k-1)}|\mathbf{D})/J(\boldsymbol{\theta}^{(k-1)}|\boldsymbol{\theta}^*)} = \frac{\pi_{like}(\mathbf{D}|\boldsymbol{\theta}^*)\pi_{prior}(\boldsymbol{\theta}^*)J(\boldsymbol{\theta}^{(k-1)}|\boldsymbol{\theta}^*)}{\pi_{like}(\mathbf{D}|\boldsymbol{\theta}^{(k-1)})\pi_{prior}(\boldsymbol{\theta}^{(k-1)})J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)})}.$$

We refer to [4] for details regarding the Metropolis-Hastings algorithm.

2.3 Delayed Rejection Adaptive Metropolis (DRAM)

Random walk Metropolis algorithm with Gaussian proposal distribution involves a proposal at each step and accepts or rejects this proposal based on the ratio between proposal and prior likelihoods. As indicated previously, should the proposal variance be set too high, many proposals will be rejected. This is undesirable, as it increases the autocorrelation of the chain. Furthermore, should the target distribution deviate greatly from the proposal distribution, the proposal will not match the local shape of the distribution, resulting in poor sampling. This motivated developing various adaptive algorithms, including DRAM [5], in attempts to circumvent these issues. *Delayed rejection* consists of before rejecting a sample, a series of backup proposals, each with successively smaller jumps in state space, are pushed through the Metropolis-Hasting acceptance probability rejection. They are tested in order of decreasing jump size, and if one of them is accepted, the sampler continues. If they are all rejected, the sampler rejects the sample and starts again. Conversely, when the proposal variance in the Metropolis algorithm is too small, the sampler will randomly walk through regions of higher likelihood in the posterior distribution without efficiently sampling the tails. This results in a high acceptance rate. In order to mitigate this, *Adaptive Metropolis* sampling continuously adapts the proposal covariance. This is accomplished by using the covariance of the history of the chain as the proposal covariance matrix of the Gaussian proposal distribution instead of the arbitrary proposal covariance imposed at the start. Adapting the proposal to match the posterior covariance structure results in a better chain performance than a static proposal covariance. We note that because adaptive algorithms employ part of the chain history to update the proposal function, they are no longer Markovian processes. Hence ergodicity properties must be established to guarantee convergence to posterior density including *diminishing adaptation* and *bounded convergence conditions* detailed in [6].

Adaptive Metropolis (AM). Starting from initial covariance, $\mathbf{V}_0$, the updated chain covariance matrix at k-th step is taken from the chain generated so far such that

$$\mathbf{V}_k = s_p \, cov(\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \cdots, \boldsymbol{\theta}^{(k-1)}) + \varepsilon \mathbf{I}_p,$$

where the scaling factor s_p depends on the dimension p of the parameter space and $s_p = 2.38^2/p$ is standard optimal choice for Gaussian targets [5]. The term $\varepsilon \mathbf{I}_p$ with the small

number $\varepsilon \geq 0$ and $\mathbf{I}_p$ is the p-dimensional identity matrix, ensures that $\mathbf{V}_k$ is positive definite, i.e., the term prevents the sample covariance matrix from becoming singular. In practice, an efficient way of computing $\mathbf{V}_{k+1}$ is the following recursive relation

$$\mathbf{V}_{k+1} = \frac{k-1}{k}\mathbf{V}_k + \frac{s_p}{k}[k\bar{\boldsymbol{\theta}}^{(k-1)}(\bar{\boldsymbol{\theta}}^{(k-1)})^T - (k+1)\bar{\boldsymbol{\theta}}^{(k)}(\bar{\boldsymbol{\theta}}^{(k)})^T + \boldsymbol{\theta}^{(k)}(\boldsymbol{\theta}^{(k)})^T + \varepsilon\mathbf{I}_p],$$

where $\bar{\boldsymbol{\theta}}^{(k)} = \frac{1}{k}\sum_{i=0}^{k-1}\boldsymbol{\theta}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ are column vectors. The sample mean can also be computed recursively as

$$\bar{\boldsymbol{\theta}}^{(k+1)} = \frac{k}{k+1}\bar{\boldsymbol{\theta}}^{(k)} + \frac{1}{k+1}\boldsymbol{\theta}^{(k)}$$

Delay Rejection (DR). The DR algorithm provides a mechanism for constructing alternative candidates $\boldsymbol{\theta}^{*(j)}$ if $\boldsymbol{\theta}^*$ is rejected rather than initially retaining the previous value as in the standard Metropolis algorithm. In particular, the second-stage candidate $\boldsymbol{\theta}^{*(2)}$ is chosen using the proposal function

$$J_2(\boldsymbol{\theta}^{*(2)}|\boldsymbol{\theta}^{(k-1)}, \boldsymbol{\theta}^*) = \mathcal{N}(\boldsymbol{\theta}^{(k-1)}, \gamma^2\mathbf{V}_k),$$

where $J_2(\boldsymbol{\theta}^{*(2)}|\boldsymbol{\theta}^{(k-1)}, \boldsymbol{\theta}^*)$ indicates that we are proposing $\boldsymbol{\theta}^{*(2)}$ having started at $\boldsymbol{\theta}^{(k-1)}$ and rejected $\boldsymbol{\theta}^*$. Since $\gamma < 1$, the second-stage proposal is narrower than the original, which increases mixing. The probability of accepting the second-stage candidate, having started at $\boldsymbol{\theta}^{(k-1)}$ and rejected $\boldsymbol{\theta}^*$, due to symmetry of J_2, is

$$\alpha_2(\boldsymbol{\theta}^{*(2)}|\boldsymbol{\theta}^{(k-1)}, \boldsymbol{\theta}^*) = \min\left(1, \frac{\pi(\boldsymbol{\theta}^{*(2)}|\mathbf{D})J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{*(2)})[1-\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{*(2)})]}{\pi(\boldsymbol{\theta}^{(k-1)}|\mathbf{D})J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(k-1)})[1-\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{*(k-1)})]}\right).$$

The above condition guarantees that the detailed balance condition is satisfied and that $\alpha \leq 1$ (see [5] for more details). If $\boldsymbol{\theta}^{*(2)}$ is rejected, a third-stage candidate and acceptance condition can be constructed, and recursive relations to construct j-th stage candidate $\boldsymbol{\theta}^{*(j)}$ and probabilities $\alpha_i(\boldsymbol{\theta}^{*(j)}|\boldsymbol{\theta}^{*(j-1)}, \cdots, \boldsymbol{\theta}^{*(2)}, \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k-1)})$ are provided in [5].

In combination, DRAM provides two complementary mechanisms to modify the proposal function. The AM provides feedback in the sense that information learned about the posterior through accepted chain candidates is used to update the proposal via the chain covariance matrix. The DR is an open-loop mechanism that alters the proposal function in a predetermined manner to improve mixing. The modifications from DR are temporary and have the goal of stimulating mixing, whereas the AM mechanism enacts permanent changes that reflect information learned about the posterior.

2.4 Parallel Adaptive Multilevel (Parallel Tempering) MCMC

The solution of the Bayesian problem is computationally challenging as the posterior distribution may be a very complex object with multiple modes. The algorithms discussed so far might stuck in a single mode and miss exploring some of the posterior PDF modes. Additionally, we will discuss accurate computation of the evidence PDF in the Bayes theorem for model selection. Accurate calculation of such a high-dimensional integral may not be possible with naive MCMC. To overcome these challenges, Prudencio [11] proposed a parallel adaptive multilevel sampling algorithm while leveraging the parallel environment and efficiently utilizing high-performance computing resources. This sampling algorithm improves the chances of exploring all existing modes of the posterior, and thus accurate

estimation of evidence. The idea is to sample increasingly difficult intermediate distributions, until the posterior distribution is sampled. Possible intermediate distributions are given by:

$$\pi_{\text{int}}^{(\ell)}(\boldsymbol{\theta}|\mathbf{D}) = [\pi_{\text{like}}(\mathbf{D}|\boldsymbol{\theta})]^{\alpha_\ell} .\pi_{\text{prior}}(\boldsymbol{\theta}), \quad \ell = 0, 1, \cdots, L, \tag{13}$$

for a given $L > 0$ and $\alpha_0 = 0$ and $\alpha_L = 1$ denote the prior and posterior distribution, respectively. Therefore as ℓ increases, the distribution transitions from the initial prior to the posterior.

Computing evidence using the Monte Carlo method results in the high variance of the integral estimator. However, using the intermediate distributions, a so-called tempering sequence is developed as

$$\begin{aligned} \pi_{\text{evid}}(\mathbf{D}) &= \int_\Theta \pi_{\text{like}}(\mathbf{D}|\boldsymbol{\theta})\pi_{\text{prior}}(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} \\ &= \int_\Theta [\pi_{\text{like}}(\cdot|\cdot,\cdot)]^{(1-\alpha_{L-1})} [\pi_{\text{like}}(\cdot|\cdot,\cdot)]^{(\alpha_{L-1}-\alpha_{L-2})} \cdots [\pi_{\text{like}}(\cdot|\cdot,\cdot)]^{(\alpha_2-\alpha_1)} \\ &\quad [\pi_{\text{like}}(\cdot|\cdot,\cdot)]^{\alpha_1} \pi_{\text{prior}}(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} \end{aligned} \tag{14}$$

The above integral can be computed by series of Monte Carlo estimations of simpler integrals, considering α_1, $\alpha_2 - \alpha_1, \cdots$ are small enough. This leads to

$$\pi_{\text{evid}}(\mathbf{D}) = \int_\Theta \pi_{\text{like}}(\mathbf{D}|\boldsymbol{\theta})\pi_{\text{prior}}(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} = c_L c_{L-1} c_{L-2} \cdots c_2 c_1, \tag{15}$$

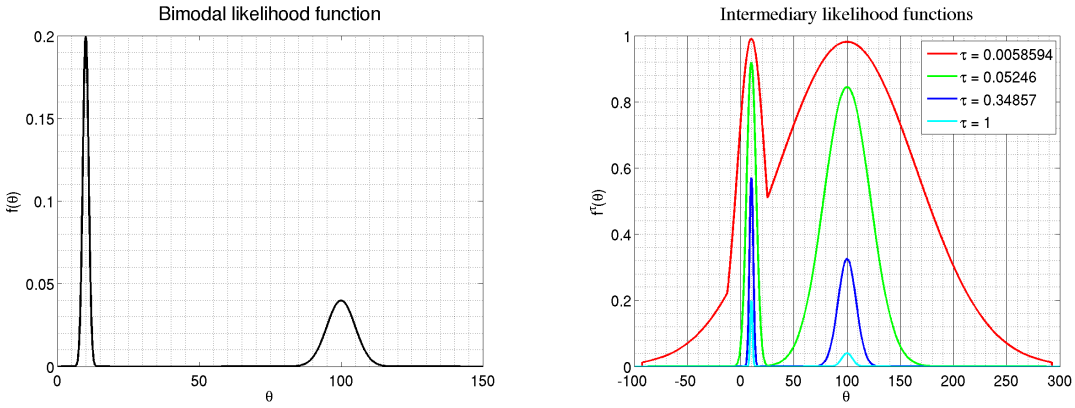
where Monte Carlo can be efficiently applied to calculate

$$c_1 = \int [\pi_{\text{like}}(\cdot|\cdot,\cdot)]^{\alpha_1} \pi_{\text{prior}}\mathrm{d}\boldsymbol{\theta}_i, \qquad c_2 = \int [\pi_{\text{like}}(\cdot|\cdot,\cdot)]^{\alpha_2-\alpha_1} h_1(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}, \qquad \cdots \tag{16}$$

The accuracy of each individual estimator c_ℓ can be controlled by determining the rate of increase of the α_ℓ adaptively, such that the likelihoods at each level be very close to the previous level. In this regard, the number of levels L is not predefined and adaptively determined throughout the simulation. Such a sampling approach can significantly benefit from parallel computing, where at each level, many computing nodes can be employed to sample the parameter collectively. We refer to [11] for more details on computational load balancing and efficiency of this adaptive multilevel MCMC. An example of intermediary likelihood functions for bimodal likelihood functions with four levels is shown in Figure 2.4.

2.5 Existing computational toolboxes.

Several computational toolboxes in different programming languages currently implement variants of MCMC algorithms. Example includes the Python implementation of the random walk Metropolis algorithm in `https://github.com/Joseph94m/MCMC/blob/master/MCMC.ipynb` and the MCMC toolbox for Matlab `https://mjlaine.github.io/mcmcstat/`. The Matlab implementation of DRAM algorithm can be found in `http://helios.fmi.fi/~lainema/dram/` and MATDRAM `https://www.mathworks.com/matlabcentral/fileexchange/80866-matdram-delayed-rejection-adaptive-metropolis-mcmc`. More comprehensive C++ libraries are QUESO [12] and Sandia National Labs' DAKOTA [1].



(a) (b)

Figure 8: (a) Likelihood function given in Section 4.1 A 1D Problem of [11]. (b) Intermediary likelihood functions, where τ_i is the exponent computed at the i-th level. The cyan-colored curve (exponent $\tau = 1$) is the same curve as part (a). The example is adopted from the QUESO users manual [12].

References

[1] B. Adams, W. Bohnhoff, K. Dalbey, M. Ebeida, J. Eddy, M. Eldred, R. Hooper, P. Hough, K. Hu, J. Jakeman, et al. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.13 user's manual. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.

[2] S. N. Bernstein. *Theory of Probability*. Moscow, 1927.

[3] S. Brooks and A. Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7:434–456, 1998.

[4] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[5] H. Haario, M. Laine, A. Mira, and E. Saksman. DRAM: efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354, 2006.

[6] H. Haario, E. Saksman, and J. Tamminen. An adaptive metropolis algorithm. *Bernoulli*, pages 223–242, 2001.

[7] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[8] E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University press, 2003.

[9] J. Kaipio and E. Somersalo. *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media, 2006.

[10] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[11] E. Prudencio and S. H. Cheung. Parallel adaptive multilevel sampling algorithms for the bayesian analysis of mathematical models. *International Journal for Uncertainty Quantification*, 2(3), 2012.

[12] E. E. Prudencio and K. W. Schulz. The parallel c++ statistical library 'queso': Quantification of uncertainty for estimation, simulation and optimization. In *European Conference on Parallel Processing*, pages 398–407. Springer, 2011.

[13] R. C. Smith. *Uncertainty quantification: theory, implementation, and applications*, volume 12. Siam, 2013.

[14] R. von Mises. *Wahrscheinlichkeitsrechnung*. Springer, 1931.