Multi-Level Monte Carlo Markov Chain

Joint Project

Bayesian Statistics - Computational Statistics

Politecnico di Milano, Mathematical Engineering

Andrea Boselli, Carlo Ghiglione, Eleonora Spizzi,
Erica Manfrin, Randeep Singh

Tutors:
Prof. Alessandra Guglielmi, Prof. Andrea Manzoni

15 February 2022

Contents

1 Abstract

Many natural phenomena and engineering applications are described by *Differential Models* expressed in form of *Partial Differential Equations* (PDE) or *Ordinary Differential Equations* (ODE). In several scenarios, models depend on unknown parameters. A problem of applicative interest is to estimate them from a noisy and partial observation of the solution.
The *Parameter Estimation* problem can be tackled within a *Bayesian framework* since it naturally provides probability distributions that can be propagated through models. Assuming an additive noise with Gaussian distribution, a likelihood can be assigned to the observed data. The Bayesian framework is completed adding a prior for the parameters, allowing to update the knowledge about them computing the posterior.
However, the computation of the likelihood requires the exact solution of the differential problem, that is usually impossible to have in closed form. Numerical solvers provide accurate approximations but, when the complexity of the model and the required accuracy increase, the computation can become extremely time consuming, making standard *Monte Carlo Markov Chain* (MCMC) procedure unfeasible.
In our project, we explore the *Multi Level Monte Carlo Markov Chain* (MLMCMC) method, that has been recently proposed as a possible solution [1]. It applies the MCMC procedure solving the differential problem at different levels of accuracy, so that the *coarse levels* (faster but less accurate) propose samples to the *fine levels* (slower but more accurate), hoping to achieve more efficient and less correlated samplings and a reduction of the variance when estimating expected values.
Our project is structured in two main parts: in the first, we focus on a PDE and in the second on a ODE. In particular:

1. we introduce the theoretical *Bayesian Parameter Estimation* framework and MLMCMC procedure;
2. we apply MLMCMC to an Advection-Diffusion PDE, adopting nested levels of accuracy of the numerical solver and tuning the likelihood variance;
3. we introduce a Neural Networks-based *Surrogate Model* at coarse level and study the impact of prior settings, data collection scheme and hyper-parameters tuning;
4. we apply MLMCMC to a Compartmental ODE model for Epidemiology adopting nested levels of refinements; we tune the multilevel parameters and study the impact of prior settings;
5. we then estimate the main features of a new partially observed epidemic informing the prior on historical data;
6. we finally adopt a simpler partially overlapping epidemiological model in the coarse level; we study the performance and the applicability range.

In our analysis, we compare the performance of MLMCMC to standard MCMC methods. We rely on the implementation of Python *PyMC3* library, namely on the *Multi Level Delayed Acceptance* (MLDA) algorithm.

2 Background on Multi Level MCMC

In this chapter, we give a theoretical background regarding the method under analysis and show how it is useful to study *Parameter Estimation* problems, especially in the context of differential models (like the PDE and ODE examined in the following chapters).

2.1 Bayesian Parameter Estimation

Differential Models are defined in a physical domain $\Omega \subset \mathbb{R}^d$ ($d \in \mathbb{N}$) and depend on parameters $\underline{\theta} \in \Theta \subset \mathbb{R}^p$, ($p \in \mathbb{N}$). Formally, their solution is defined as:

$$\begin{aligned} u : \Theta \times \Omega &\to \mathbb{R} \\ (\underline{\theta}, \underline{x}) &\mapsto u(\underline{\theta}, \underline{x}). \end{aligned}$$

Suppose we collect some noisy measurements $\underline{y} := \{y_i\}_{i=1}^N$ of the solution at some points $\mathcal{D} := \{\underline{x}_i\}_{i=1}^N$ of the physical domain:

$$y_i = u(\underline{\theta}, \underline{x}_i) + \epsilon_i$$

with $\underline{x}_i \in \Omega$ and ϵ_i denoting random noise.
To retrieve the Bayesian framework, we model the noise with a Gaussian distribution $\epsilon_i|\tau \overset{\text{iid}}{\sim} \mathcal{N}(0, \tau^2)$, so that a likelihood $L(\mathcal{D}, \underline{y}|\underline{\theta})$ can be assigned to the observed data:

$$y_i|\underline{\theta}, \tau \overset{\text{ind}}{\sim} \mathcal{N}(u(\underline{\theta}, \underline{x}_i), \tau^2) \text{ with } \tau^2 > 0.$$

The *Bayesian Parameter Estimation* setting is completed assigning a prior distribution $\pi(\underline{\theta})$ to $\underline{\theta}$ with the aim of computing the posterior distribution $\pi(\underline{\theta}|\mathcal{D}, \underline{y})$ of $\underline{\theta}$.
However, the likelihood definition $L(\mathcal{D}, \underline{y}|\underline{\theta})$ requires the solution of the differential problem $u(\underline{\theta}, \underline{x})$ that is usually unavailable in closed form and is then replaced by its approximation $u_h(\underline{\theta}, \underline{x})$ computed with a numerical solver.

2.2 Multi Level Monte Carlo

Consider a differential problem, depending on parameters $\underline{\theta}$, and let $u(\underline{\theta}, \cdot)$ be its solution. A typical task is approximating the expected value of a *Quantity of Interest* $Q(\underline{\theta}) := f(u(\underline{\theta}, \cdot))$, where f is an arbitrary function.
Suppose that we can sample from $\pi(\underline{\theta})$, the distribution of $\underline{\theta}$, while $u(\underline{\theta}, \cdot)$ cannot be computed exactly. Hence, from a sequence of numerical solutions $\{u_\ell(\underline{\theta}, \cdot)\}_{\ell=0}^\infty$, whose accuracy increases with the *level* ℓ, we define a hierarchy of approximations $\{Q_\ell\}_{\ell=0}^\infty$ of Q, with $Q_\ell(\underline{\theta}) := f(u_\ell(\underline{\theta}, \cdot))$.
A typical improvement to classical *Monte Carlo* (MC) method is provided by the *Multi*

Level Monte Carlo (MLMC) method, relying on the use of multiple *Control Variates*. In particular, choosing L large enough:

$$\begin{aligned}\mathbb{E}[Q] \simeq \mathbb{E}[Q_L] &= \mathbb{E}[Q_0] + \sum_{\ell=1}^{L} \mathbb{E}[Q_\ell - Q_{\ell-1}] \\ &\simeq \frac{1}{N_0}\sum_{i=1}^{N_0} Q_0(\underline{\theta}^{(i,0)}) + \sum_{\ell=1}^{L}\frac{1}{N_\ell}\sum_{i=1}^{N_\ell}[Q_\ell(\underline{\theta}^{(i,\ell)}) - Q_{\ell-1}(\underline{\theta}^{(i,\ell)})]\end{aligned}$$

where $\underline{\theta}^{(i,\ell)} \overset{\text{iid}}{\sim} \pi(\underline{\theta})$. A proper choice of $\{N_\ell\}_{\ell=0}^{L}$ ensures better performances than classical MC method, in particular the estimator can achieve a strong *variance reduction*. Unfortunately, we cannot sample from $\pi(\underline{\theta})$ in most cases, since it is unknown. Indeed, $\pi(\underline{\theta})$ is usually the posterior of $\underline{\theta}$, whose likelihood evaluation typically requires the unknown solution $u(\underline{\theta}, \cdot)$. However, we rely on the numerical solutions $\{u_\ell(\underline{\theta}, \cdot)\}_{\ell=0}^{\infty}$ to compute a likelihood at each level, thus obtaining a hierarchy of approximations of the posterior $\{\pi^\ell(\underline{\theta})\}_{\ell=0}^{\infty}$ such that $\pi^\ell(\underline{\theta}) \to \pi(\underline{\theta})$ as $\ell \to \infty$. So for L big enough the expression above becomes:

$$\mathbb{E}[Q] \simeq \mathbb{E}_{\pi^L}[Q_L] = \mathbb{E}_{\pi^0}[Q_0] + \sum_{\ell=1}^{L}(\mathbb{E}_{\pi^\ell}[Q_\ell] - \mathbb{E}_{\pi^{\ell-1}}[Q_{\ell-1}]) \tag{2.1}$$

Note that, although now we know $\pi^\ell(\underline{\theta})$, it is not obvious how we can sample from them. So here comes into play *Multi Level Monte Carlo Markov Chain* (MLMCMC).

2.3 Multi Level Monte Carlo Markov Chain

The idea of MLMCMC [1] is to estimate each term on the RHS of (2.1) separately. In particular, we estimate:

- $\mathbb{E}_{\pi^0}[Q_0]$ through standard MCMC (for instance, using the *Metropolis-Hastings* algorithm);
- each $Y_\ell := (\mathbb{E}_{\pi^\ell}[Q_\ell] - \mathbb{E}_{\pi^{\ell-1}}[Q_{\ell-1}])$ from $\{\underline{\theta}_\ell^n\}_{n\in\mathbb{N}}$ and $\{\underline{\Theta}_{\ell-1}^n\}_{n\in\mathbb{N}}$ employing 2 Markov chains. If N_ℓ denotes the number of samples on level ℓ:

$$\hat{Y}_{\ell,N_\ell} = \frac{1}{N_\ell}\sum_{n=1}^{N_\ell} Y_\ell^n = \frac{1}{N_\ell}\sum_{n=1}^{N_\ell}[Q_\ell(\underline{\theta}_\ell^n) - Q_{\ell-1}(\underline{\Theta}_{\ell-1}^n)]. \tag{2.2}$$

The biggest advantages of this method consist in the significative *reduction of the computational cost* and in the *variance reduction*. We mainly focus on the first one, which is due to mainly two reasons:

- The cost of the samples of Q_ℓ is less than the one of Q_L for $\ell < L$, so that on the coarser levels the estimators are cheaper;
- If $Var(Y_\ell^n) = Var(Q_\ell(\underline{\theta}_\ell^n) - Q_{\ell-1}(\underline{\Theta}_{\ell-1}^n)) \to 0$ as $\ell \to \infty$, we need a smaller number of samples to get a quite accurate estimate of Y_ℓ on the finer levels, so that the computational cost is reduced on the finer levels too.

However, to practically obtain the advantages above, the two Markov chains $\{\underline{\theta}_\ell^n\}$ and $\{\underline{\Theta}_{\ell-1}^n\}$ must be chosen wisely and there are several MLMCMC algorithms that achieve this goal. In this project, we use the *Multi Level Delayed Acceptance* (MLDA) algorithm. The main idea is to use samples generated at each level as proposals for the level above.
In particular, let us assume to have just two levels ($L = 1$), which is indeed the case we will consider throughout our project. Let us denote with $\{\underline{\Theta}_0^n\}_{n\in\mathbb{N}} \sim \pi^0(\underline{\theta})$ the coarse chain ($\ell = 0$) and with $\{\underline{\theta}_1^n\}_{n\in\mathbb{N}} \sim \pi^1(\underline{\theta})$ the fine chain ($\ell = 1$). The MLDA algorithm reads:

Algorithm: MLDA
Set the initial states $\underline{\Theta}_0^1, \underline{\theta}_1^1 = \underline{\Theta}_0^1$; fix T (number of total draws) and n_{subs} (subsampling rate).

- **On coarse level**:
  1. Generate $n_{subs} \times T$ samples from $\pi^0(\underline{\theta})$ using an MCMC sampler (e.g. *Metropolis-Hastings* or *DEMetropolisZ*)
  2. Subsample the obtained chain each n_{subs} values, to get the final coarse chain $\{\underline{\Theta}_0^j\}_{j=1}^{j=T}$
- **On fine level**: for $j = 2, .., T$
  1. compute
$$\alpha_{\mathrm{ML}}(\underline{\Theta}_0^j | \underline{\theta}_1^{j-1}) = \min\left\{1, \frac{\pi^1(\underline{\Theta}_0^j)\pi^0(\underline{\theta}_1^{j-1})}{\pi^1(\underline{\theta}_1^{j-1})\pi^0(\underline{\Theta}_0^j)}\right\}$$
  2. Set $\underline{\theta}_1^j = \underline{\Theta}_0^j$ with probability α_{ML} and $\underline{\theta}_1^j = \underline{\theta}_1^{j-1}$ otherwise.

Why do we need the subsampling on the coarse chain? Essentially, the MLDA sampler is based on the assumption that the coarse proposal samples (i.e. the samples proposed from the coarse chain to the fine one) are independent (or almost independent) from each other. In order to generate independent samples, it is necessary to run the coarse chain for an adequate number of iterations to get rid of *Autocorrelation*. Therefore, the higher the autocorrelation in the coarse chain, the more iterations are needed and the larger the subsampling rate should be.

In this work, we test MLDA as a sampling algorithm, rather than as a variance reduction technique for the evaluation of *Quantities of Interest*. For the practical implementation of MLDA, we rely on Python *PyMC3* library. In particular, we compare the performance of MLDA with adaptive *Metropolis-Hastings* (MH) and *DEMetropolisZ* (DEMZ). The latter denotes a further adaptive algorithm, included in the benchmark because it is employed as sampling method at level 0 of MLDA, in *PyMC3* implementation. Further information about DEMZ can be found in [2].
As for MLDA, after examining the "standard" version, we focus on studying the impact of using problem-specific *Surrogate Models* at coarse level (respectively: a *Neural Network* in the PDE case of Chapter 3 and the SIR model in the ODE case of Chapter 4).

3 Application to a PDE problem: the "Comet" Equation

In this chapter, we use MLDA algorithm to perform *Bayesian inference* on the parameters of a PDE. In particular, to compute the likelihood at the coarse level, we compare the use of a classical numerical solver with a *Neural Network* used as *Surrogate Model*.

3.1 Mathematical Model

Our first case study is a linear advection-diffusion PDE, which we refer to as *"Comet" Equation*. We consider a Dirichlet boundary value problem, that reads as follows:

$$\begin{cases} -\mu\Delta u + 10(\cos\theta, \sin\theta)\cdot\nabla u = 10\mathrm{e}^{-50\|\underline{x}-\underline{x}_0\|_2} & \underline{x}\in\Omega=[0,1]^2 \\ u = 0 & \underline{x}\in\partial\Omega \end{cases} \tag{3.1}$$

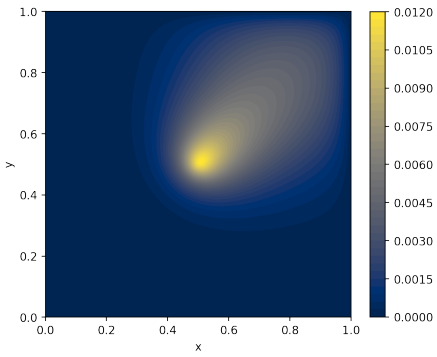


Figure 3.1: Solution of Comet Equation for $(\mu,\theta)=(0.5,\frac{\pi}{4})$, with its peculiar shape

where $\underline{x}_0=(0.5,0.5)$. Parameters have a clear interpretation: the *diffusion parameter* $\mu\in(0,+\infty)$ affects the flatness of the solution u, while the *angle of advection* $\theta\in(0,2\pi)$ controls the direction where u is pushed the most.
The aim of this chapter is to perform *Bayesian inference* on both μ and θ, given some noisy observations of u, available on a finite set $\mathcal{D}\subset\Omega$.

3.1.1 Solution of Comet Equation

The solution u of Comet Equation is usually defined in the *Sobolev space* $V=H_0^1(\Omega)$, but it cannot be computed analytically. Hence, we rely on numerical methods, in particular on the *Finite Element Method (FEM)*: an approximation u_h of the solution u is computed in the space $\dot{X}_h^r=\{v_h\in C^0(\overline{\Omega}) : v_h|_K\in\mathbb{P}_r\ \forall K\in\mathcal{T}_h; v_h|_{\partial\Omega}=0\}$, where $\mathcal{T}_h$ is a triangulation of the domain Ω (usually called *mesh*) and $\mathbb{P}_r$ is the space of polynomials of degree $\leq r$ [3]. The quality of the approximation, together with the computational cost, increases as the polynomial degree r and the triangulation

refinement increase. In this work, problem (3.1) is solved through the FEM method using piecewise quadratic polynomials ($r = 2$), exploiting the Python *FEniCS* library [9].

3.1.2 Data Generation

Data are generated computing the FEM solution for fixed and known (μ^*, θ^*) on a set of points of the domain, to which a Gaussian noise is added. More specifically:

1. Consider a finite set $\mathcal{D} \subset \Omega$ of points of the domain, and set $(\mu^*, \theta^*) = (2, \pi)$;

2. Compute the FEM solution $u_h^*((\mu^*, \theta^*), \underline{x})$ for Comet Equation on a structured mesh made by triangular elements, with mesh size $h = 2^{-8}$. Being the chosen triangulation very refined, FEM yields a very accurate approximation u_h^* of u;

3. Considered data are $\{(\underline{x}_i, y_i)\}_{\underline{x}_i \in \mathcal{D}}$, with $y_i = u_h^*((\mu^*, \theta^*), \underline{x}_i) + \epsilon_i$ and $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. Being $\max_{\underline{x}} u_h^*((\mu^*, \theta^*), \underline{x}) \simeq 5.086 \cdot 10^{-3}$, the noise scale is fixed at the reasonable value $\sigma = 1 \cdot 10^{-4}$.

3.1.3 Priors and Likelihoods

For the rest of the chapter, unless differently specified, we set *Uniform* independent priors for μ and θ:

$$\pi(\mu, \theta) = \pi(\mu)\pi(\theta): \quad \mu \sim \mathcal{U}(0.1, 5) \quad \theta \sim \mathcal{U}(0, 2\pi).$$

For what concerns the likelihood, we set:

$$y_i | \mu, \theta \overset{\text{ind}}{\sim} N(u_h((\mu, \theta), \underline{x}_i), \tau^2),$$

where $u_h((\mu, \theta), \underline{x}_i)$ is a numerical solution (computed, for instance, through FEM method), while τ is a fixed noise scale. In particular, τ is chosen to be greater than σ, to keep into account both the Gaussian noise and the numerical error. The latter is due to the different accuracy between FEM for data generation and for likelihood computation, since data are generated from u_h^* with $h = 2^{-8}$, while any choice of $h \leq 2^{-6}$ for the numerical solution u_h in the likelihood yields computationally unfeasible MCMC runs.

3.2 Approach 1: Coarse and Fine Model Differ in the FEM Triangulation

In this section, MLDA with subsampling rate $n_{subs} = 2$ and MH are compared. In particular:

- For MH and both levels of MLDA, u_h is computed through FEM;
- Both MH and the fine level of MLDA feature a triangulation with $h = 2^{-5}$, while for the coarse level of MLDA we set $h = 2^{-4}$;

- Both MH and the fine level of MLDA feature $\tau = 5 \cdot 10^{-4}$, while different values of τ ($\tau \in \{5 \cdot 10^{-4}, 7.5 \cdot 10^{-4}, 1 \cdot 10^{-3}\}$) for the coarse level of MLDA are compared.

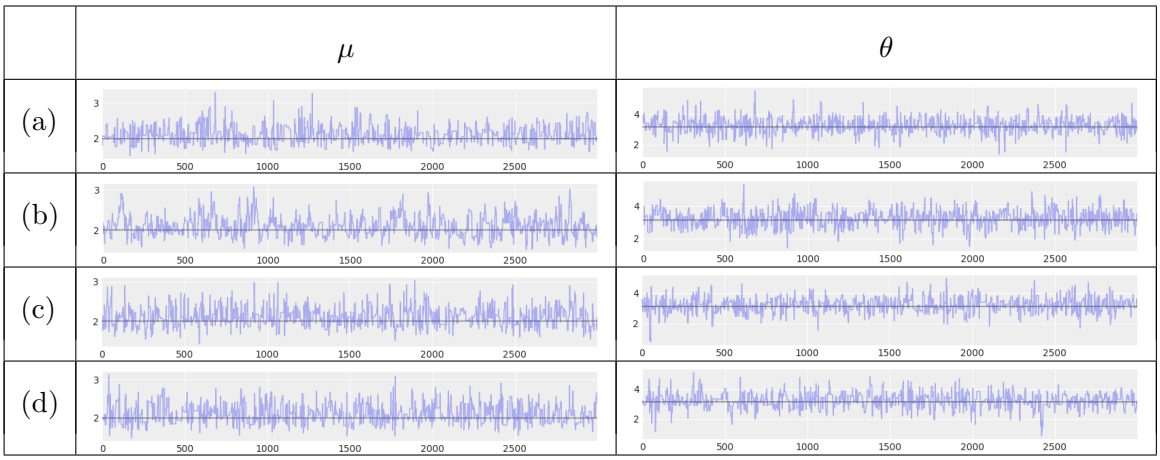


Figure 3.2: Traces for μ and θ using samplers (a) MH, (b) MLDA with $\tau = 5 \cdot 10^{-4}$, (c) MLDA with $\tau = 7.5 \cdot 10^{-4}$, (d) MLDA with $\tau = 1 \cdot 10^{-3}$

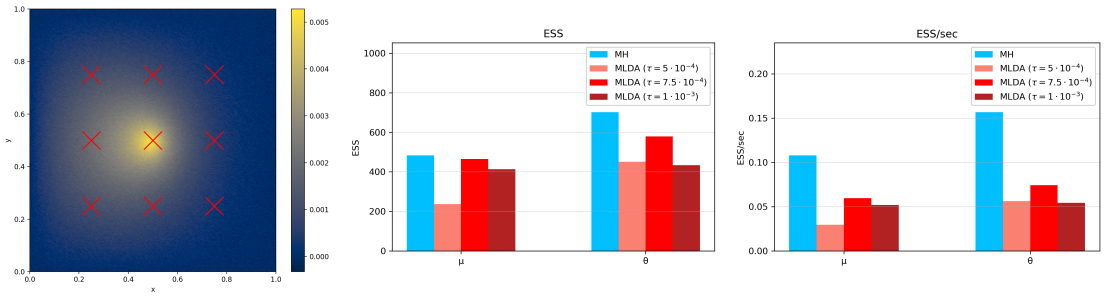


Figure 3.3: Left: Set $\mathcal{D}$ of points where data is collected; Center and Right: ESS and ESS/sec of the considered methods

First of all, as parameters are sampled in reasonably narrow intervals, we conclude that the chosen data-points set $\mathcal{D}$ conveys sufficient information about them.

Concerning the choice of τ for coarse level, the worst performance is achieved for $\tau = 5 \cdot 10^{-4}$, the same of fine level: the problem is that, fixing the same error scales, the different solver accuracies in the two levels are not kept into account.

Anyway, traceplots and *Effective Sample Size* (ESS) barplots suggest no advantage in using MLDA rather than MH, regardless of the chosen τ. This is probably due to the too small subsampling rate n_{subs} chosen for MLDA.

Moreover, ESS/sec barplots point out that MH sampler is much faster than MLDA, due to the slowness of FEM solver employed in the coarse level of MLDA. To increase the speed of the solver for such level, in the next section we introduce a *Surrogate Model* for the efficient approximation of the PDE problem.

3.3 Approach 2: Coarse level features a NN as a surrogate model

We introduce a *Surrogate Model* for the coarse level, that is, an *emulator* of a classical PDE solver. Built with a data-driven approach, it is able to compute accurate approximations of the solutions of the PDE in a specific range of the parameters with a much lower computational effort than a traditional FEM solver.
A widely used type of *Surrogate Model* in simulating PDEs are *Neural Networks (NNs)* [7], very flexible tools that, at a high *off-line phase* cost (due to their *training*), are able to accurately reconstruct the manifold of the solutions (*i.e.* the map associating the parameter to the corresponding solution of the PDE). If designed properly, they can mimic with high precision a FEM solution with faster evaluation (*on-line phase*).
Formally, the *Neural Network* associates to each geometric point $\underline{x} \in [0,1]^2$ and couple of parameters $(\mu, \theta) \in \mathbb{R} \times [0, 2\pi]$ $u_{\text{NN}}((\mu,\theta), \underline{x})$, an approximation of the corresponding solution $u((\mu,\theta), \underline{x})$ of the PDE:

$$\begin{aligned} u_{\text{NN}}: \quad & \mathbb{R}^+ \times \mathbb{R} \times [0,1]^2 \to \mathbb{R} \\ & (\mu, \theta, \underline{x}) \mapsto u_{\text{NN}}((\mu,\theta), \underline{x}) \end{aligned}$$

The implementation of such architecture follows these steps:

1. collection of a *training dataset*;
2. definition of a *loss function*, measuring the accuracy of the output w.r.t. our data;
3. definition of the *Neural Network* architecture;
4. minimization of the loss (*training*) through optimization techniques;
5. evaluation of the output (*validation*).

The *NN* is implemented with *Keras*, a Python library for *Artificial Intelligence*.

3.3.1 Surrogate model implementation

The training dataset $\Delta = \{u_i\}_{i=1}^N$ is built collecting the solutions of the PDE provided by a classical FEM solver featuring a triangulation $\mathcal{T}_h$ with $h = 2^{-5}$ for parameters μ and θ ranging in $[0.1, 5]$ and $[0, 2\pi]$ respectively. Solutions are computed for all the couples of the grid defined by 30 different values of $\{\mu_j\}_{j=1}^{30}$ and $\{\theta_k\}_{k=1}^{30}$ uniformly distributed in their ranges:

$$u_i(\underline{x}) = u_{\text{FEM}}((\mu_j, \theta_k), \underline{x}), \quad j = 1, ...30, \quad k = 1, ...30, \quad i = 30(j-1) + k \in \{1, ..., 900\}$$

so that the training dataset contains solutions of the PDE for $N = 900$ different couples of the parameters evaluated on a grid $\mathcal{P} \subset [0,1]^2$ of points of the domain. Note that, once the *NN* is trained, u_{NN} can be evaluated on any point of the domain. The solutions are standardized in [0,1] for numerical stability of the training.

The loss function is defined as the mean quadratic distance of the output of the *NN* from the solution of the dataset:

$$\mathcal{L} = \frac{1}{|\mathcal{P}|N} \sum_{i=1}^{N} \sum_{\underline{x} \in \mathcal{P}} [u_i(\underline{x}) - u_{\text{NN}}((\mu, \theta)_i, \underline{x})]^2$$

The architecture of the *NN* is composed of 4 fully connected layers featuring 50 neurons each with *hyperbolic tangent* activation function.
The network is trained setting a random training/validation split of 90%, batch-size of 64 and performing 100 epochs with *Adam* optimizer with $learning - rate = 10^{-3}$ and 50 epochs with *Stochastic Gradient Descent* with $learning - rate = 10^{-2}$ and $momentum = 0.9$.

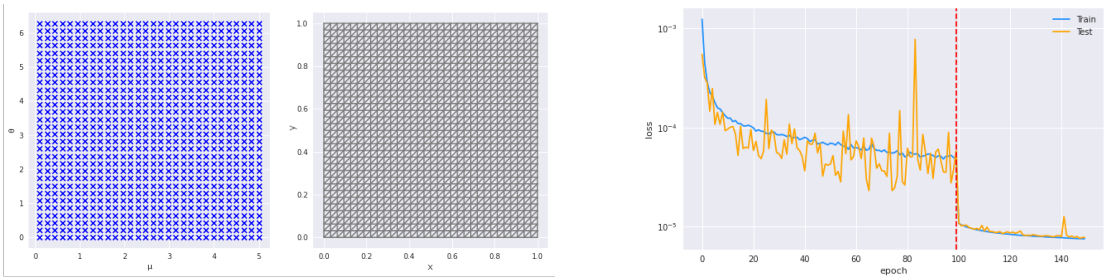


Figure 3.4: Left: Grid of parameters μ and θ for the training dataset; Center: Triangulation with $h = 2^{-5}$; Right: Training logs of the Neural Network.

The emulator reaches validation accuracy levels at the order of 10^{-5}, showing no *overfitting* issues during the training during 3913 seconds in total.
The high accuracy of the *Surrogate Model* to reconstruct the manifold of the solutions in the given range can be observed graphically comparing some solutions (see figure A.1 in Appendix A). Note that the FEM solver takes on average 3.5 seconds to compute a solution for a given pair of (μ, θ), whereas the *Surrogate Model* just takes on average 0.30 seconds, being more that 10x faster.

3.3.2 Subsampling rate

The replacement of the FEM solver in the coarse level with the *Surrogate Model* allows to set higher values of the subsampling rate n_{subs} hyper-parameter, still keeping the simulation time limited, thanks to the much faster computation of the likelihood.
In particular, MLDA performances are compared to MH and DEMZ algorithms setting $n_{subs} = 2, 5, 10, 20$. Data are collected on the data-points set $\mathcal{D}$ (see figure 3.3). All the other settings are equal: $\tau = 5 \cdot 10^{-4}$ for MH, DEMZ and fine level of MLDA, $\tau = 7.5 \cdot 10^{-4}$ for coarse level of MLDA, priors $\mu \sim \mathcal{U}(0.1, 5)$ and $\theta \sim \mathcal{U}(0, 2\pi)$ and same initial guess for the parameters. A simulation of 1000 burn-in + 3000 iterations is run for all the methods, providing the results in figures 3.5 and A.3.
As expected from the theory, high values of n_{subs} increase the *Acceptance Rate*, the ESS and the efficiency of the algorithm, providing better performances with respect to MH and DEMZ. Thanks to the *Surrogate Model*, the computational cost of MLDA is highly decreased w.r.t. Section 3.2, being more efficient than MH and DEMZ in terms

of ESS/sec, in particular for high values of n_{subs} (10, 20). See figure A.2 in Appendix A for the *Autocorrelation* plots.

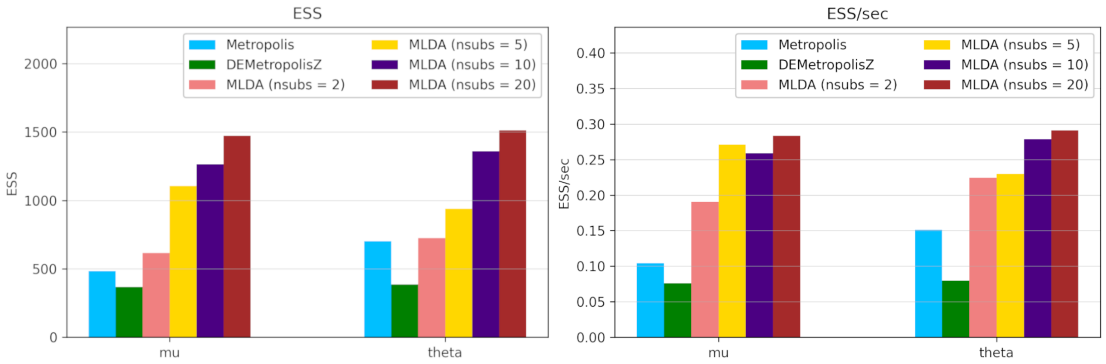


Figure 3.5: ESS (left) and ESS/sec (right) for the six methods.

From the traceplots, it is evident that MLDA with high values of n_{subs} provides less correlated sampling from the posterior, providing posterior distributions of the parameters with a shape more regular and concentrated around the target value with respect to MH and DEMZ methods. For this reason, in the following experiments the subsampling rate is set to 20.

3.3.3 Data collection grid

As we can now rely on a more powerful sampler for parameters in the case of the Comet Equation, we can show the effect of the choice of the data-points set $\mathcal{D} \subset \Omega$ on the likelihood and the resulting posterior, detecting the most informative points of Ω for each parameter. This finds application in the context of *data scarcity*, where data are supposed to be collected through expensive sensors, so that we aim at minimizing their quantity.
Inference is performed through MLDA with 3 different sets $\mathcal{D}$ and the standard settings chosen for section 3.3. For each set $\mathcal{D}$, likelihoods for both levels of MLDA, traceplots and posteriors for (μ, θ) are shown in table A.1 in Appendix A.

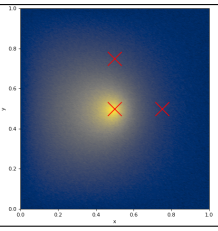
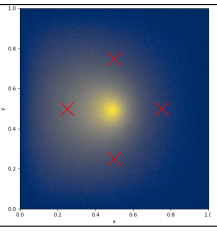
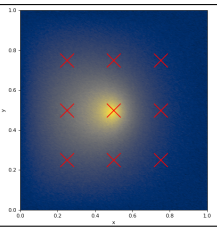
	3 Points	4 Points	9 Points
(a)			

Table 3.1: The 3 considered sets $\mathcal{D}$.

The set with 3 points, encompassing $\underline{x}_0 = (0.5, 0.5)$ central point of Ω, yields a narrow

posterior for μ, but not for θ. Conversely, the set with 4 points, encompassing more points around $\underline{x}_0$, but not $\underline{x}_0$, yields a more concentrated posterior for θ, but not for μ. We conclude that including $\underline{x}_0$ conveys relevant information about μ, while including some points around $\underline{x}_0$, for instance 4, is necessary to accurately infer θ. Consequently, the set $\mathcal{D}$ considered in most of our analyses, featuring 9 points including $\underline{x}_0$ and points around $\underline{x}_0$, is sufficient to perform Bayesian inference for both parameters (μ, θ).

3.3.4 Prior for the parameters

In this section, we try three different settings for the prior of μ, in order to understand whether there is a remarkable effect on the estimation of the posterior distribution, assuming different a priori knowledge scenarios of the diffusion parameter.

Three possibilities are tested: (a) $\mu \sim \mathcal{U}(0.1, 5)$, (b) $\mu \sim \Gamma(2, 1)$ and (c) $\mu \sim \Gamma(10, 5)$.

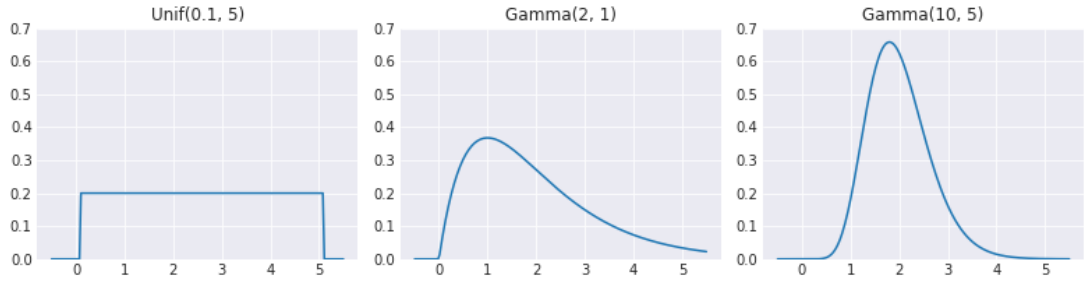


Figure 3.6: Priors for μ for cases (a), (b) and (c) respectively.

The priors have the following values of mean and variance, noting that (a) and (b) have the same variance and (b) and (c) the same mean (equal to the target value):

Prior	Mean	Variance
(a)	2.45	2.00
(b)	2.00	2.00
(c)	2.00	0.40

Note that the case (c) distribution provides the strongest prior information about μ, whereas (a) the weakest one.
See figure A.4 in Appendix A for the traces and posteriors.
In the following table, the main features of the estimated posteriors of μ are reported:

μ	Mean	Std. Dev.	HDI 94%
(a)	2.105	0.227	[1.685; 2.533]
(b)	2.071	0.227	[1.681; 2.517]
(c)	2.066	0.212	[1.679; 2.449]

We remark that estimations are very similar under different chosen priors, so the effect of the latter is very small. We gather that data are sufficient to generate a likelihood which overwhelms any information that the prior provides, despite the set $\mathcal{D}$ consists of only 9 measurements.

4 Application to an ODE problem: SEIR Epidemiological Model

In this chapter, we use MLDA algorithm to perform *Bayesian inference* on the parameters of an ODE, the SEIR model. Moreover, we compare the use of the SIR and of the SEIR at the coarse level.

4.1 Mathematical Model

Our second case study are ODEs, focusing on two Compartmental Epidemiological models: SIR and SEIR. Epidemiological models are used to model the spread of infectious diseases, but they are also useful to analyze phenomena in widely different fields, like the adoption of a product or the diffusion of opinions/information.
SIR is the simplest model, composed by 3 compartments:

- *Susceptible (S)*: number of people who are still unaffected by the disease;
- *Infectious (I)*: number of people who are ill at time t (and thus can contribute to the diffusion of the epidemics);
- *Recovered (R)*: number of the people who have recovered or died because of the infection.

SEIR is another compartmental model, often used as a backbone in the analysis of infectious diseases, where we add a fourth compartment:

- *Exposed (E)*: number of people who have contracted the infection but are still not infective (i.e. people that are in their incubation period for the disease);

The systems of equations describing SIR and SEIR models are, respectively:

$$\begin{cases} S'(t) = -\beta SI \\ I'(t) = \beta SI - \gamma I \\ R'(t) = \gamma I \\ S(0) = S_0, I(0) = I_0, R(0) = R_0 \end{cases} \qquad \begin{cases} S'(t) = -\beta SI \\ E'(t) = \beta SI - \sigma E \\ I'(t) = \sigma E - \gamma I \\ R'(t) = \gamma I \\ S(0) = S_0, E(0) = E_0, I(0) = I_0, R(0) = R_0 \end{cases}$$

$$S(t)+I(t)+R(t) = 1 \qquad S(t)+E(t)+I(t)+R(t) = 1$$

where $\beta \in (0, \infty)$ is the *Infection rate*, $\sigma \in (0, \infty)$ is the *Incubation rate*, $\gamma \in (0, \infty)$ is the *Recovery rate*. The three rates and the variance of the noise of the observations τ^2, differently from the PDE case where τ^2 was fixed, are the four parameters we want to estimate.

4.2 Approach 1: Coarse and fine model differ in the solver time-step

As a first approach, we decide to use the same model (SEIR) both at coarse and fine level but with different time steps, replicating a situation with data separated by a long time interval in which we want an estimate of the process more frequently.
In this section, MH, DEMZ and MLDA are compared. As baseline, we set MLDA featuring 2 levels and $n_{subs} = 5$.

4.2.1 Data Generation and initial settings

Data are generated setting β=0.27, σ=0.2, γ=0.1, that are the values estimated from the Covid-19 infection in India [4], adding Gaussian noise of variance $\tau^2 = 0.01$.
We normalize the population and we set the initial proportion of the four compartments as follows: I(0)=0.1, E(0)=0.1 and R(0)=0. An instance of SEIR solution with Gaussian noise is shown in figure 4.1. Note that the added noise can push some observations outside the [0,1] normalization interval.

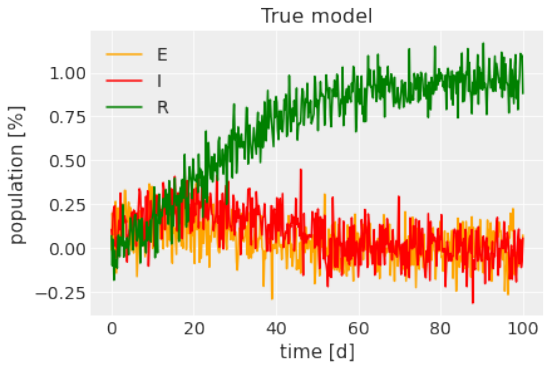


Figure 4.1: Data generated with noise

For the rest of the chapter, unless differently specified, we set *Uniform* independent priors for β, σ and γ:

$$\pi(\beta) \sim \mathcal{U}(0.05, 0.5)$$
$$\pi(\sigma) \sim \mathcal{U}(0.05, 0.4)$$
$$\pi(\gamma) \sim \mathcal{U}(0.05, 0.8)$$
$$\pi(\tau^2) \sim \mathcal{HC}(0, 1).$$

Note that here the numerical solution of the differential system is a vector:

$$\underline{u}_h(\underline{\theta}, t) = [E_h(\underline{\theta}, t), I_h(\underline{\theta}, t), R_h(\underline{\theta}, t)]$$
$$\underline{\theta} = (\beta, \sigma, \gamma), \quad t \in [0, T]$$

The data, generated through simulation, are $(t_i, \underline{y}_i)$, where:

$$t_i \in [0, T]$$
$$\underline{y}_i = \underline{u}_h(\underline{\theta}^*, t_i) + \underline{\epsilon}_i, \quad \underline{\theta}^* = (0.27, 0.2, 0.1), \quad \underline{\epsilon}_i \sim \mathcal{N}(0, \tau^2 \mathbb{I}_3)$$

So that the likelihood reads as:

$$\underline{y_i}|\underline{\theta}, \tau \sim \mathcal{N}(\underline{u}_h(\underline{\theta}, t_i), \tau^2 \mathbb{I}_3)$$

4.2.2 Subsampling rate

Firstly, we tune the subsampling rate, trying with $n_{subs} = 2,\ 5,\ 10,\ 20$. We find that the best performances in terms of ESS/sec are obtained with $n_{subs} = 10$ that is the optimal trade-off between the quality of the sampling (ESS) and the computational cost. So, we set this value as baseline.

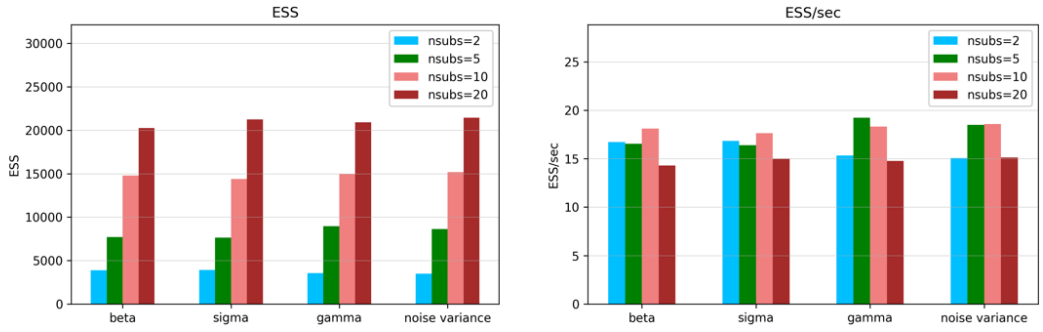


Figure 4.2: ESS and ESS/sec of MLDA with different subsampling rates.

4.2.3 Time grid refinement

We then change the time step of the coarse model to understand how a different amount of available data impacts on the MLDA performance. We fix the time step of the fine level $T_f = 0.2$ and in the coarse we set it as a multiple of T_f, in particular $T_c = 2T_f, 5T_f, 10T_f, 20T_f$. The best choices are not surprisingly $T_c = 2T_f$ and $T_c = 5T_f$, since these are the cases with more available data.

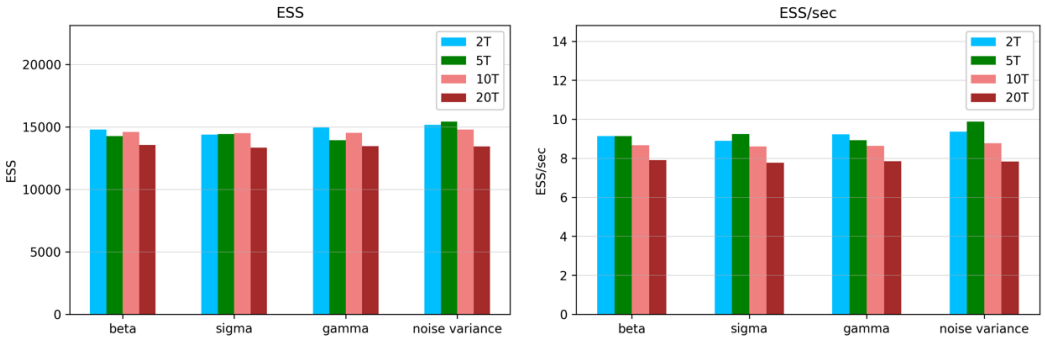


Figure 4.3: ESS and ESS/sec of MLDA with different time steps.

4.2.4 Prior for the parameters

From the literature about SEIR Epidemiological model [5], it is suggested that β and σ follow a *Lognormal* distribution whereas γ follows a *Weibull*. Since in this part we are mostly interested in comparing the performance of MH, DEMZ and MLDA, we set the priors for the parameters in a favourable case. In particular, we impose the means equal to the true values, and the variances to 10^{-4}:

$$\pi(\beta) \sim LogNormal(\mu = -1.31, \sigma = 0.037)$$
$$\pi(\sigma) \sim LogNormal(\mu = -1.61, \sigma = 0.04997)$$
$$\pi(\gamma) \sim Weibull(\alpha = 2, \beta = 0.12).$$

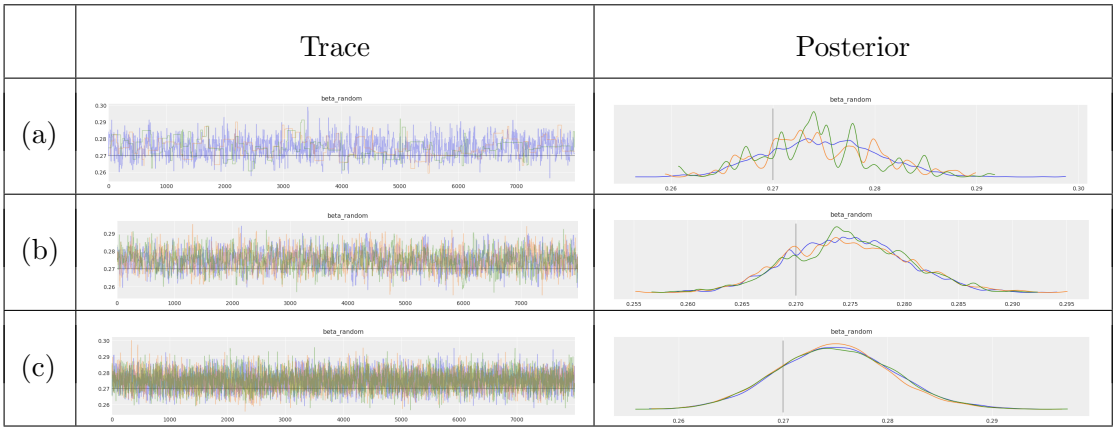


Figure 4.4: Traces (left) and posteriors (right) for β with (a) MH, (b) DEMZ, (c) MLDA; the gray line is the target value.

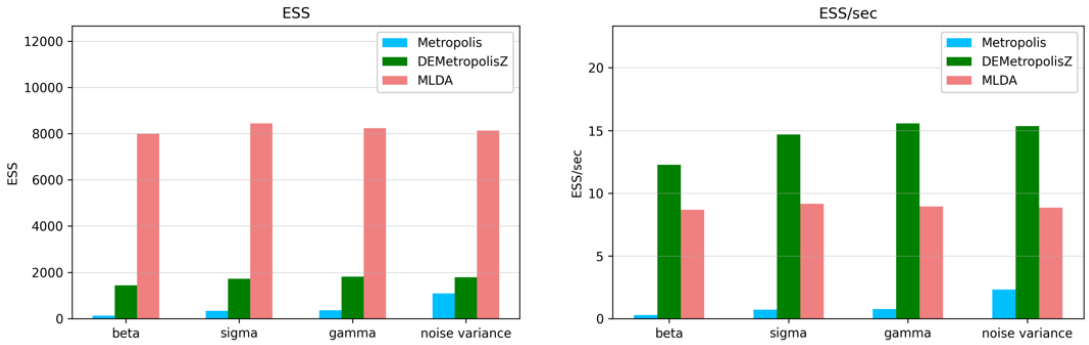


Figure 4.5: ESS and ESS/sec of the three different methods

We observe that, having fixed the same number of iterations, MLDA algorithm outperforms MH and DEMZ, providing more uncorrelated sampling (*Autocorrelation* plots in figure A.5 in Appendix A) with higher ESS and much more regular posteriors. Nevertheless, the performances are lower than DEMZ in terms of ESS/sec.

4.2.5 Prior informed on historical data

Having selected the optimal prior and hyper-parameters configuration, we apply the model in a real case scenario. In particular, we test the capacity of the method to estimate the time of the peak of the *Infectious* just from the early stages of the epidemic, using prior info coming from previously observed phenomena.
To simulate this context, the mean and variance of the posteriors of the parameters β, σ, γ are estimated with MLDA technique from a previously observed epidemic. Then, parameter estimation is performed on data coming from the early stages of a new epidemic with different observed infection rate. Priors are set from parametric family of Section 4.2.4 with hyper-parameters such that parameters means coincide with the ones of the posteriors estimated from historical data and the variances analogously, up to a scaling factor.
The new epidemic has $\beta = 0.27$, $\sigma = 0.2$ and $\gamma = 0.1$, with an *Infectious* peak at time 20.12, whereas the historical one has infection rate 25% higher (σ and γ are the same). The method is applied in three cases when the new epidemic is observed up to time $T = 10, 20, 50$ respectively.

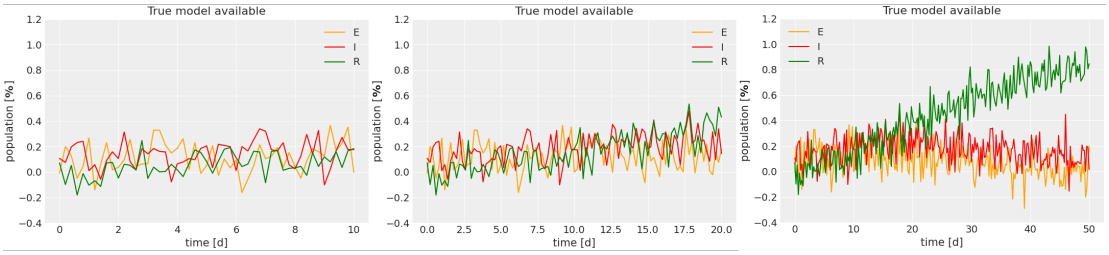


Figure 4.6: Observed data from new epidemic up to time 10 (left), 20 (center) and 50 (right).

Parameter estimation is performed running 2000 burn-in + 8000 iterations with 3 parallel chains of MLDA algorithm, setting $n_{subs} = 5$ and time step ratio between coarse and fine equal to 2.
For all the three cases, the sampling has very good performances both in terms of ESS and efficiency, having a very high acceptance rate of 0.835, 0.882 and 0.878 respectively. The good sampling performance of MLDA method is confirmed by the traceplots not showing correlation issues and the posteriors that have a very regular shape, being centered very close to the target value too. Traces and posteriors are shown in figure A.6 in Appendix A.
We report the expected value and variance of the peak time and value coming from the Bayesian estimate of the model parameters, knowing that the true values are 20.12 and 0.225 respectively.

T	Peak time	SE	Peak value	SE
10	19.94	$7.47 \cdot 10^{-3}$	0.243	$2.33 \cdot 10^{-4}$
20	19.81	$5.45 \cdot 10^{-3}$	0.245	$1.34 \cdot 10^{-4}$
50	20.33	$3.25 \cdot 10^{-3}$	0.233	$8.56 \cdot 10^{-5}$

As expected, as the time horizon increases, the *Standard Error* (SE) of the estimate decreases. Moreover, even from the early stages of the epidemic, the predicted peak

time and value are very close to the real values, meaning that the method is able to predict well the main features.
We can further note that, using the same number of iterations, for the same time horizon ($T = 10$) MLDA provides much more precise estimates with respect to MH and DEMZ methods in terms of SE of the posteriors.

Method	Peak time SE	Peak value SE
MH	$1.86 \cdot 10^{-2}$	$4.63 \cdot 10^{-4}$
DEMZ	$1.165 \cdot 10^{-2}$	$5.05 \cdot 10^{-4}$
MLDA	$7.47 \cdot 10^{-3}$	$2.33 \cdot 10^{-4}$

4.3 Approach 2: Coarse level is the SIR model

As further development, we implement a simpler mathematical model at the coarse level of MLDA, namely the SIR model, hoping to improve the efficiency of the method thanks to the lower computational cost it requires.
However, its usage is not straightforward since SEIR model has an additional parameter, the *incubation rate* σ, whose effect must be taken into account in such a way that the SIR and SEIR solutions are as much close as possible. To achieve this, we perform some analyses on the semi-analytical approximated solutions of the models that can be retrieved in [6]. First, we observe that the curves of *all* SEIR models are simply stretched or compressed relative to one another by the factor:

$$\alpha = \frac{\sigma}{\sigma + \gamma}. \tag{4.1}$$

Noting that the SIR model is a special case with $\alpha = 1$, obtained for $\sigma \to +\infty$, we can gather that this approach can work only if σ is not too small, so that α is not too close to zero.
Now, consider the SIR indicating with $(\tilde{\beta},\tilde{\gamma})$ its parameters and with $(\tilde{S}(t), \tilde{I}(t), \tilde{R}(t))$ its states. If $\tilde{R}_0 = 0$, then:

$$\tilde{R} = \frac{\tilde{\gamma}}{\tilde{\beta}} \ln\left(\frac{\tilde{S}}{\tilde{S}_0}\right). \tag{4.2}$$

Moreover, indicating with $\widetilde{\Delta t}$ the peak time of $\tilde{I}$, under the hypothesis $\tilde{I}_0 \ll 1$, we can express:

$$\tilde{\gamma}\widetilde{\Delta t} \approx \int_{\tilde{S}_0}^{\tilde{\gamma}/\tilde{\beta}} \frac{1}{s\left[\frac{\tilde{\beta}}{\tilde{\gamma}}(s - \tilde{S}_0) - \ln\left(\frac{s}{\tilde{S}_0}\right)\right]} ds. \tag{4.3}$$

Similarly, for the SEIR model, if $R_0 = 0$:

$$R = \frac{\gamma}{\beta} \ln\left(\frac{S}{S_0}\right) \tag{4.4}$$

and the expression of the peak time Δt of I to peak, under $I_0 \ll 1$, is:

$$\alpha\gamma\Delta t \approx \int_{S_0}^{\gamma/\beta} \frac{1}{s\left[\frac{\beta}{\gamma}(s - S_0) - \ln\left(\frac{s}{S_0}\right)\right]} ds. \tag{4.5}$$

From equations (4.2) and (4.4), assuming:

$$\tilde{S}_0 = S_0, \tag{4.6}$$

$$\frac{\tilde{\gamma}}{\tilde{\beta}} = \frac{\gamma}{\beta} \tag{4.7}$$

we impose the same behaviour of R and $\tilde{R}$ in function of S and $\tilde{S}$ respectively so that the final values of *Susceptible* and *Recovered* coincide in the two models. Indeed, at final time $t = T$ for the SIR model:

$$\tilde{S}(T) + \tilde{I}(T) + \tilde{R}(T) = 1$$

but, since $\tilde{I}(T) = 0$ (no infectious people at the end of the pandemic),:

$$\begin{cases} \tilde{S}(T) + \frac{\tilde{\gamma}}{\tilde{\beta}} \ln\left(\frac{\tilde{S}(T)}{\tilde{S}_0}\right) = 1 \\ \tilde{R}(T) = 1 - \tilde{S}(T) \end{cases}$$

so both $\tilde{S}(T)$ and $\tilde{R}(T)$ depend only on $\tilde{\gamma}/\tilde{\beta}$ and $\tilde{S}_0$. Repeating the procedure for the SEIR model, where $E(T) = 0$ as well, we obtain the same dependence of $S(T)$ and $R(T)$ from γ/β and S_0. Imposing (4.6) and (4.7) we obtain that:

$$\tilde{S}(T) = S(T), \quad \tilde{R}(T) = R(T).$$

To establish the same infection peak time in the two models, we impose the equality between equations (4.3) and (4.5) and, taking into account (4.6) and (4.7), we obtain:

$$\tilde{\gamma} = \alpha\gamma. \tag{4.8}$$

Combining (4.8), (4.7) and (4.1), we finally get:

$$\tilde{\gamma} = \frac{\sigma}{\sigma + \gamma}\gamma, \tag{4.9}$$

$$\tilde{\beta} = \frac{\sigma}{\sigma + \gamma}\beta. \tag{4.10}$$

Given the SEIR parameters, the parameters computed with the formulas above generate the SIR model that best approximates the SEIR solution in terms of compartments evolution, peak times and final values.
Notice that, if the approximation is accurate, $\tilde{I}$ in SIR model should mimic $E + I$ in the corresponing SEIR model. Consequently, at coarse level we set the likelihood:

$$\begin{bmatrix} R_i \\ I_i + E_i \end{bmatrix} \mid \underline{\theta} \overset{\text{ind}}{\sim} \mathcal{N}\left(\begin{bmatrix} \tilde{R}(\underline{\tilde{\theta}}, t_i) \\ \tilde{I}(\underline{\tilde{\theta}}, t_i) \end{bmatrix}, \quad \tau^2 \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \right) \tag{4.11}$$

where $\underline{\tilde{\theta}} = (\tilde{\beta}, \tilde{\gamma})$ is computed from $\underline{\theta}$ through formulas (4.9) and (4.10), while $(\tilde{R}, \tilde{I})$ is the SIR solution computed for parameters $\underline{\tilde{\theta}}$ and initial conditions:

$$\begin{cases} \tilde{S}_0 = S_0 \\ \tilde{I}_0 = I_0 + E_0 \\ \tilde{R}_0 = R_0 = 0 \end{cases} \tag{4.12}$$

Priors for (β, γ, σ) are set as *Uniform*, while we fix τ^2 to its true value.

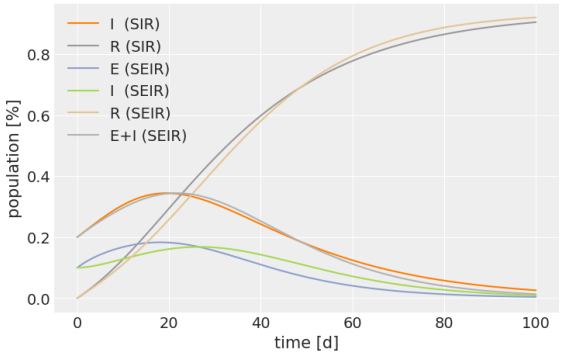


Figure 4.7: SEIR and approximated SIR curves.

4.3.1 Applicability range

The above computations hold only when the parameters respect some precise conditions. Note that, besides the assumptions in (4.12), there are other two hypotheses that are implicitly true, namely:

1. σ not too small (so that α is high enough);
2. $I_0, \tilde{I}_0 \ll 1$.

We further investigate these two assumptions, carrying out some experiments to understand the effect of σ. Unless differently specified, we set the parameters as:

β_T	γ_T	I_0	E_0	T
0.27	0.1	0.1	0.1	100

and we make different runs with the following values of σ_T:

Experiment	(1a)	(1b)	(1c)	(1d)	(1e)	(1f)
σ_T	0.2	0.1	0.05	0.035	0.35	0.5
α	0.66	0.50	0.33	0.26	0.77	0.83

As we can see from figure A.7, in the experiments *(1a), (1b)* the method seems to work well, with very uncorrelated trace-plots and regular posterior densities. In the experiment *(1c)*, the estimations are acceptable, although MLDA starts suffering a bit; indeed, we see that the three chains are quite different from one another, highlighting a problem in the convergence of the method. In *(1d)*, the sampling fails, confirming our expectations: if σ is too small, the method does not work since SIR solution is very different from the SEIR solution.
For experiments *(1e), (1f)*, an important observation comes out. Although α is sufficiently large and MLDA works well, we remark that the posterior density of σ has a very large variability, thus worsening the estimation of the real value of the incubation rate. Recalling that the curves of SEIR model are stretched by α that has an horizontal asymptote, different solutions with high σ are very similar one to the other, being α almost the same. So, the higher the true value of σ_T, the more similar the SEIR solutions

with the sampled σ, making difficult to infer the real value of the parameter. From here we can gather a trade-off: if σ_T is too small since, the method would not work; if σ_T is too big, it would be difficult to estimate it.

Regarding the second assumption, the analysis is more straightforward. We set the parameters as before with $\sigma_T = 0.1$ and the experiments for:

Experiment	(2a)	(2b)	(2c)	(2d)
I_0	0.10	0.15	0.25	0.40
$\tilde{I}_0$	0.20	0.25	0.35	0.50

As expected, from figure A.8 it is clear that MLDA performances drop as I_0 and $\tilde{I}_0$ increase. Indeed, the approximation SIR-SEIR is less accurate as the hypotheses in formulas (4.3) and (4.5) are not satisfied.

4.3.2 SIR-SEIR vs SEIR-SEIR

The main reason we develop the SIR-SEIR approach is to mimic the use of a *Surrogate model* in the coarse level, as done for the Comet Equation case, in order to gain in computational efficiency. However, this solution does not provide the same benefits as it does in the PDE case.
We run both SEIR-SEIR and SIR-SEIR models under same conditions of the true parameters, number of draws, prior distributions, and ground truth data. As we can see, it is evident that SEIR-SEIR is the clear winner in terms of ESS and ESS/sec, as regards MLDA.

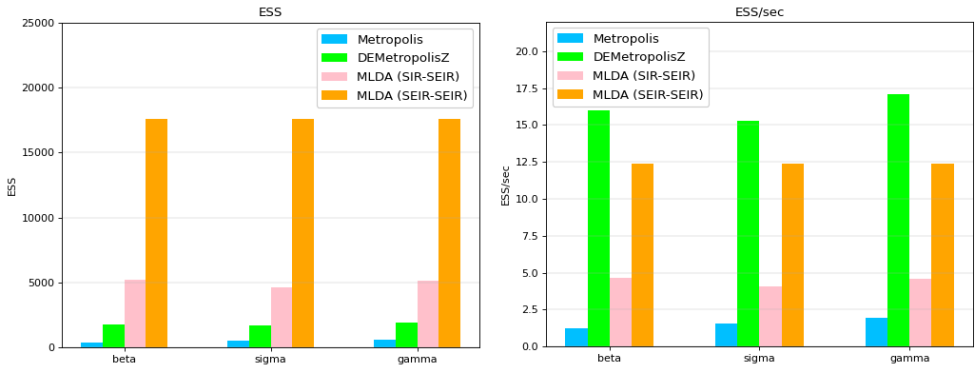


Figure 4.8: Comparing performance of SIR-SEIR versus SEIR-SEIR

We explain this behaviour with the fact that the computational gain of using SIR model instead of the SEIR in the coarse level is very small. Indeed, the *Scipy* library used to solve the differential problems is really efficient, and there is not a significant difference in solving an ODE system of three or four states. This little computational gain can not compensate the loss in the accuracy of the samples proposed by the coarse chain to the fine chain, resulting in a less precise and efficient method.
Eventually, we can say that the use of the same ODE model at both coarse and fine level is better that trying to use a surrogate low fidelity model for the coarse level, given that the performances of the ODE solver do not dramatically change in the two cases.

5 Conclusions

In this work, *Parameter Estimation* is performed through MLDA algorithm in the contexts of a PDE and an ODE, testing in both cases the use of *Surrogate Models*.
Our analysis shows that MLDA outdoes MH and DEMZ in terms of correlation between samples and ESS, thanks to the subsampling performed at coarse level. However, this usually comes at the cost of higher computational time, resulting in poorer overall performance of MLDA. To ensure overall better performance for MLDA, the computational cost of coarse level needs to be sufficiently smaller than the one of the fine. As seen, using the same solver with different degrees of accuracy for the two levels does not provide a sufficient saving in computational time, so that we employ *Surrogate Models* at coarse level.
In the case of Comet Equation, MLDA proves to be a success, thanks to the huge difference in computational time between the highly demanding FEM solver and the almost immediate *Neural Network*. On the other hand, the use of SIR model is not equally successful. Indeed, the solver at fine level is already fast, and the approximation provided by the SIR model is not accurate enough to ensure the similarity between the posteriors of coarse and fine level.
Finally, we notice that both implemented *Surrogate Models* require the true values of the parameters to be in a given *applicability range*. So, a remarkable prior knowledge about the parameters is required, in order to employ such models.

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] Cajo J. F. Ter Braak, *A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: easy Bayesian computing for real parameter spaces*, Stat Comput 16, 239–249, 2006.

[3] Quarteroni, Alfio, and Silvia Quarteroni. Numerical models for differential problems. Vol. 2. Milan: Springer, 2009.

[4] Parth Vipul Shah, *Prediction of the Peak, Effect of Intervention, and Total Infected by COVID-19 in India*, Cambridge University Press, 09 September 2020

[5] Linton NM, Kobayashi T, Yang Y, et al. Incubation period and other epidemiological characteristics of 2019 novel coronavirus infections with right truncation: A statistical analysis of publicly available case data. J Clin Med. 2020;9(2):538.

[6] Heng, K., Althaus, C.L. *The approximately universal shapes of epidemic curves in the Susceptible-Exposed-Infectious-Recovered (SEIR) model*, Sci Rep 10, 19365, 2020

[7] Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G. E. *Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators*, Nature Machine Intelligence, 3(3), 218-229, 2021

[8] Python PyMC3 package (providing an implementation for MLMCMC and a few examples)

[9] FEniCS (providing an easy implementation for the discretization and solution of PDEs through Finite Element Methods (FEM))

A Further plots

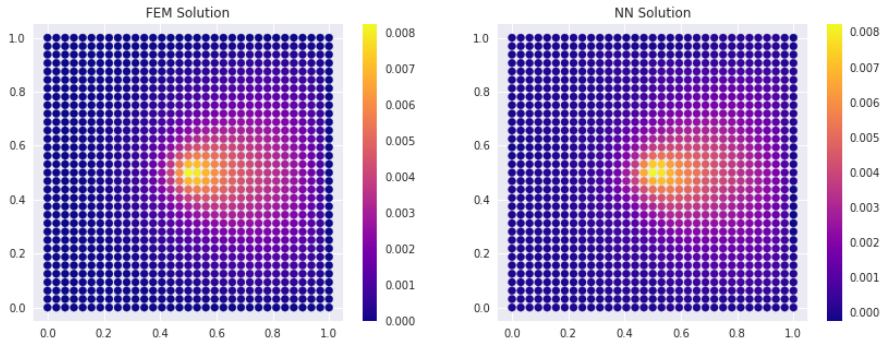
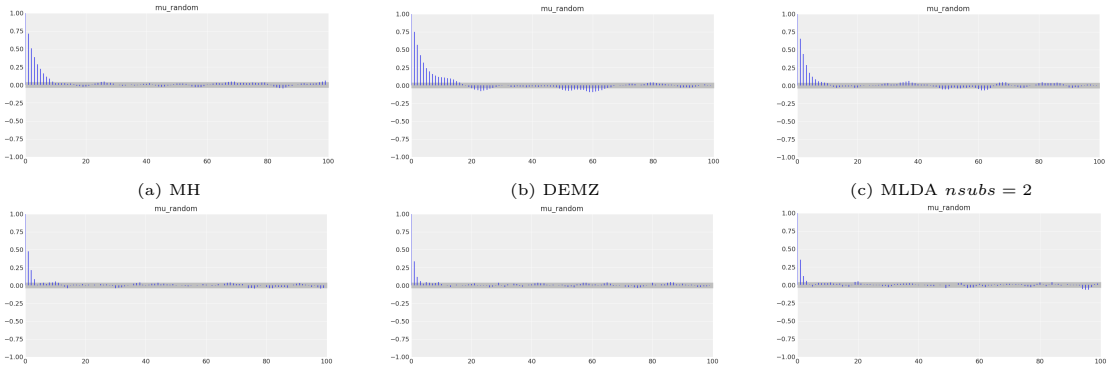


Figure A.1: FEM solution (left) and Neural Network reconstruction (right) for $\mu = 0.945$ and $\theta = 0$.



(a) MH (b) DEMZ (c) MLDA $nsubs = 2$

(d) MLDA $nsubs = 5$ (e) MLDA $nsubs = 10$ (f) MLDA $nsubs = 20$

Figure A.2: *Autocorrelation* plots of parameter μ for the different cases.

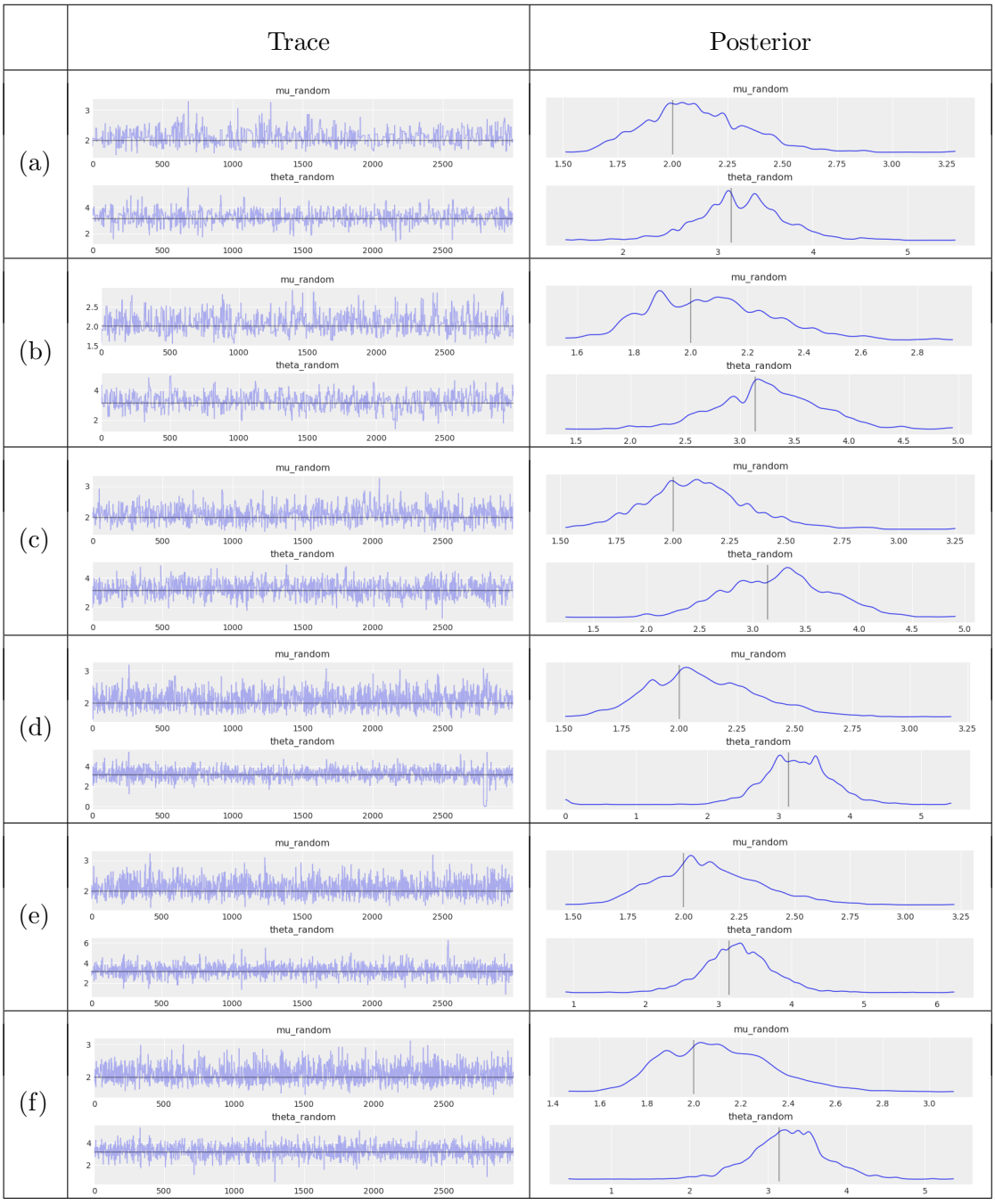


Figure A.3: Traces (left) and posteriors (right) for μ and θ using samplers (a) MH, (b) DEMZ, (c) MLDA with $nsubs = 2$, (d) MLDA with $nsubs = 5$, (e) MLDA with $nsubs = 10$, (f) MLDA with $nsubs = 20$; the gray line is the target value.

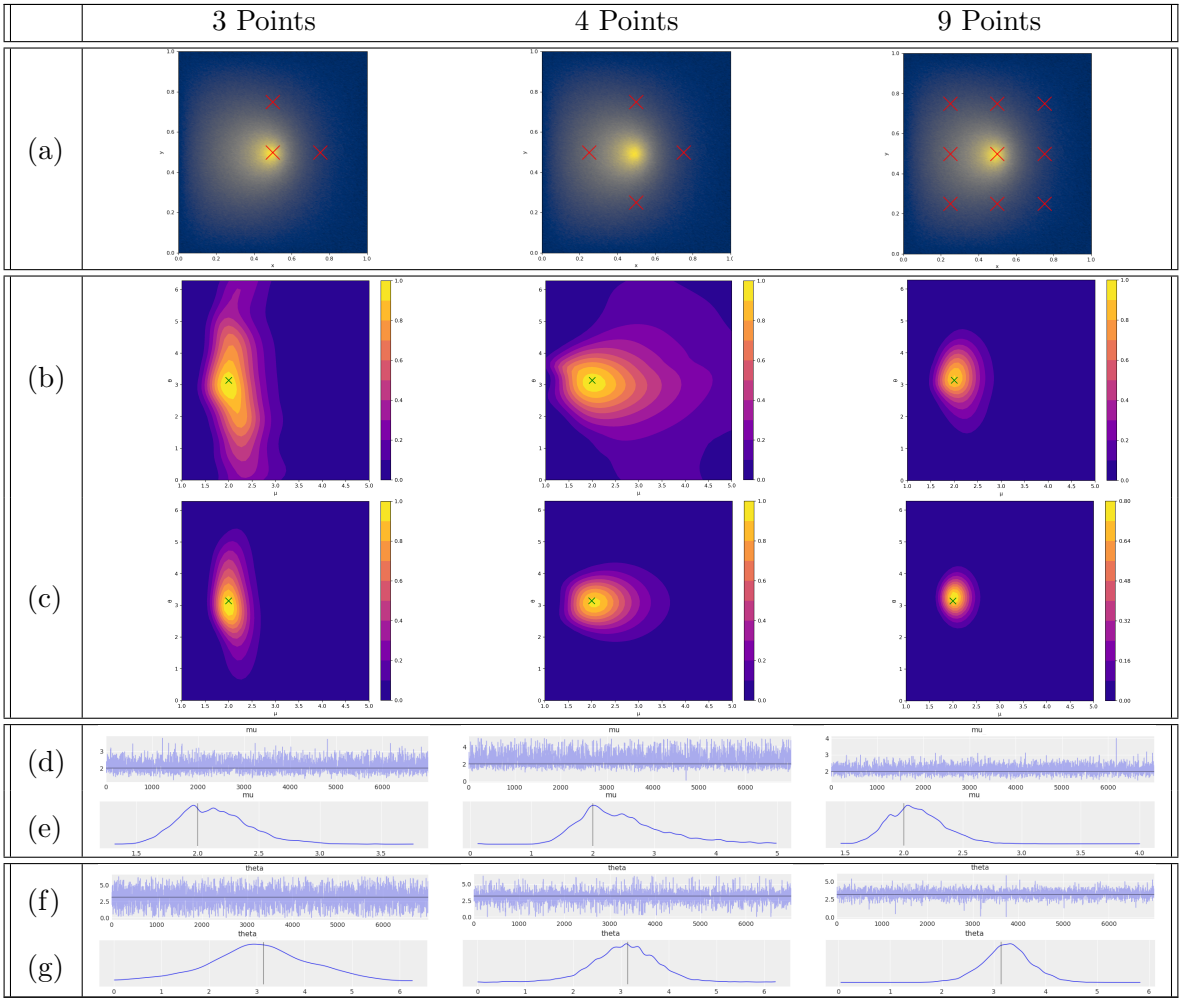


Table A.1: Comparison between 3 sets $\mathcal{D}$: (a) sets $\mathcal{D}$; (b) likelihoods of coarse level of MLDA; (c) likelihoods of fine level; (d) traceplots of μ using MLDA sampler; (e) posteriors of μ; (f) traceplots of θ; (g) posteriors of θ

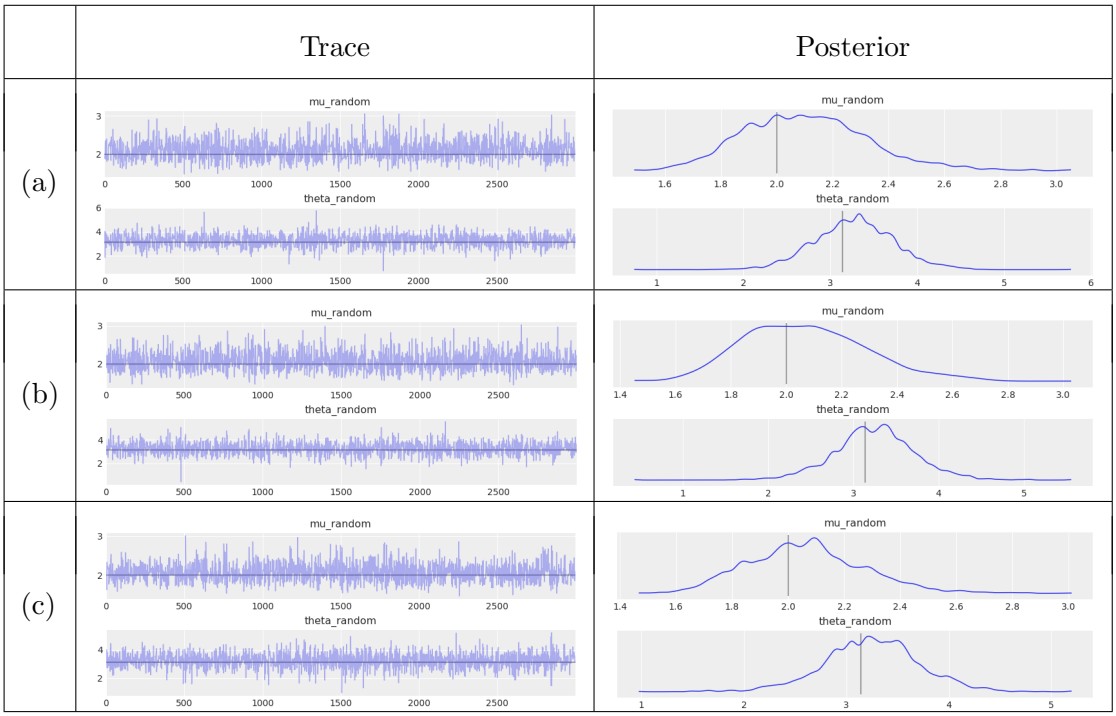


Figure A.4: Traces (left) and posteriors (right) for μ and θ setting as priors for μ $\mathcal{U}(0.1, 5)$ (a), $\Gamma(2, 1)$ (b) and $\Gamma(10, 5)$ (c) respectively; the gray line is the target value.

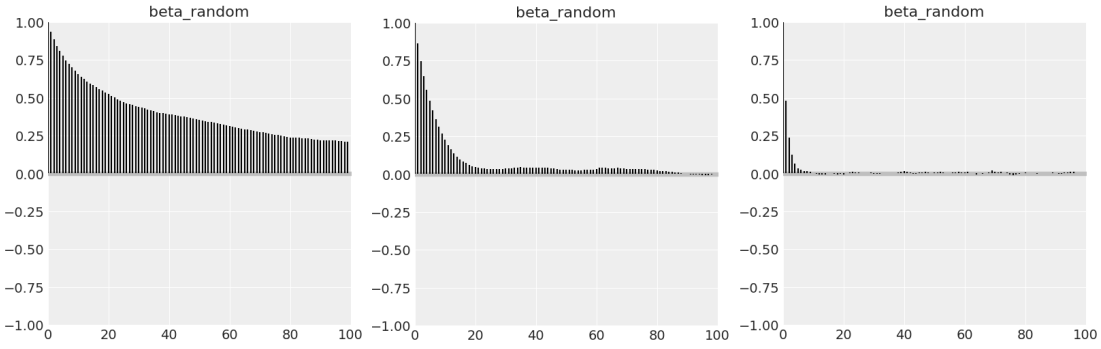


Figure A.5: Autocorrelation plots for MH (left), DEMZ (center), MLDA (right) for parameter β.

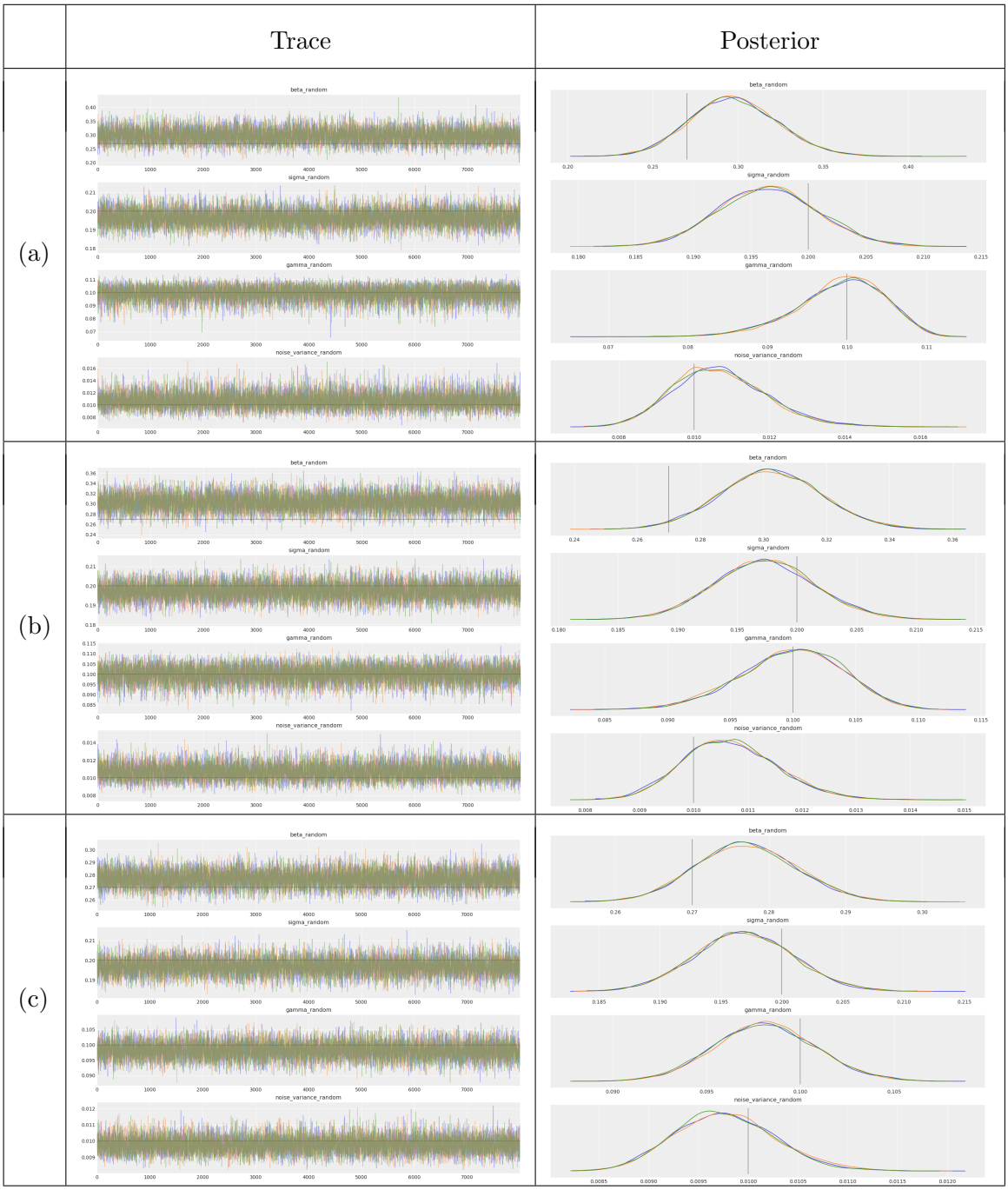


Figure A.6: Traces and posteriors for β, σ, γ and variance computed by MLDA for time horizon 10 (a), 20 (b) and 50 (c); the gray line is the target value.

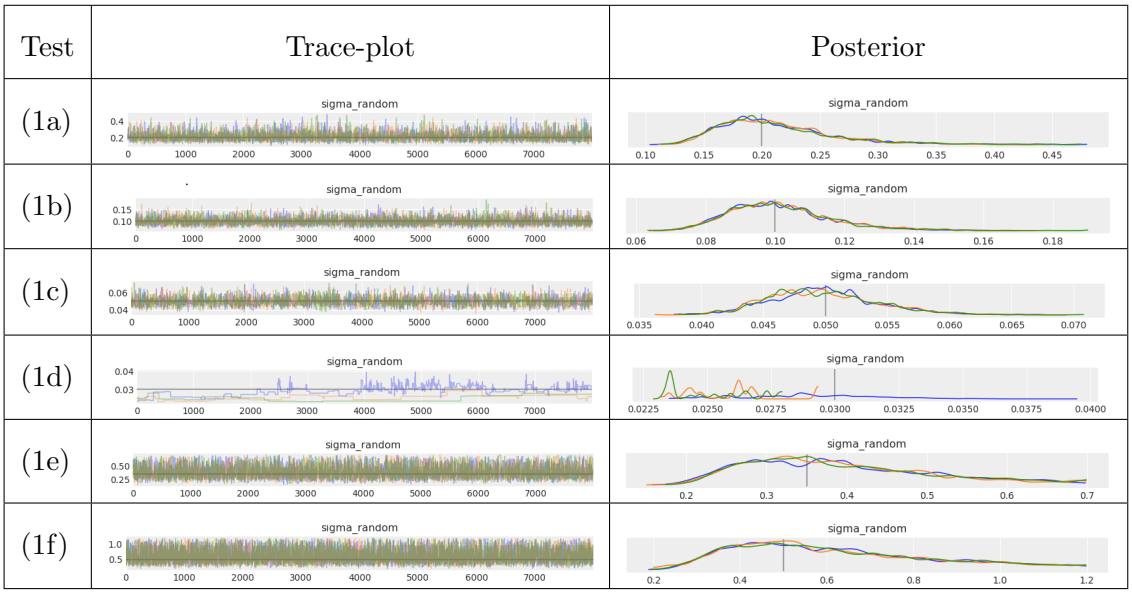


Figure A.7: Effect of different values of σ_T on the estimation of the incubation rate.

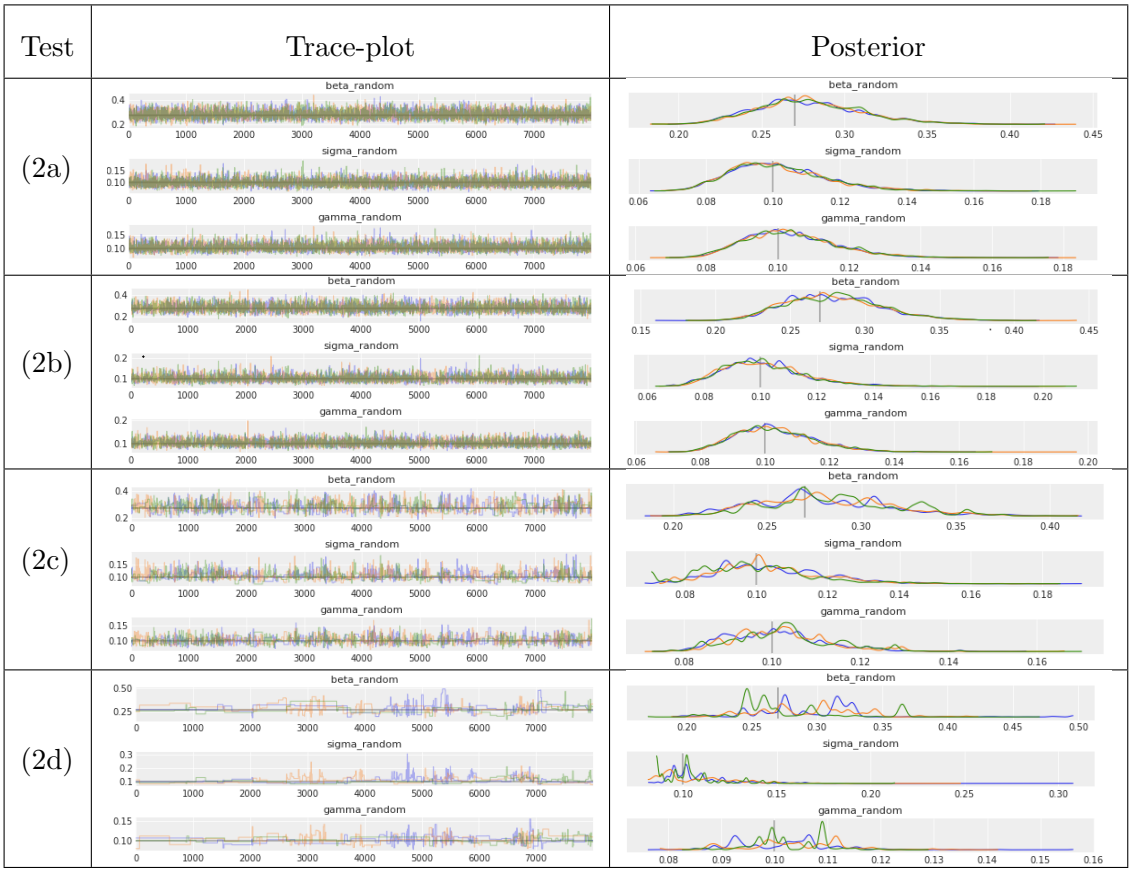


Figure A.8: Effect of different values of I_0 and $\tilde{I}_0$ on the estimation of the parameters.