# Project 3: Exact and approximate Bayesian inference

Due Tuesday 6 December 2022

## 1 Bayesian inverse problem via MCMC

The estimation of spatially distributed properties from indirect data is important to a host of science and engineering applications, ranging from hydrology to materials processing. A classic example, arising in subsurface modeling, involves estimating a spatially varying permeability field $k(x)$ from limited and noisy observations of pressure. In this project, you will develop a Bayesian approach to this inverse problem. Your Bayesian solution will, in the context of a few practical assumptions, yield quantitative descriptions of uncertainty in the permeability field conditioned on data and of uncertainty in subsequent model predictions.

### 1.1 Model

#### 1.1.1 Inverse problem definition

Consider *(once again)* an elliptic equation on a one-dimensional spatial domain:

$$\frac{\partial}{\partial x}\left(k(x)\frac{\partial u(x)}{\partial x}\right) = -s(x), \qquad x \in D = [0,1], \tag{1}$$

with a deterministic source term $s(x)$, a deterministic Dirichlet boundary condition at $x = 1$,

$$u(1) = u_r,$$

and a deterministic Neumann condition at $x = 0$,

$$k(x)\frac{\partial u}{\partial x}\bigg|_{x=0} = -F.$$

Note that this model is similar (but not completely identical) to that used in Project #2.

The exact permeability field $k(x)$ is unknown. We will endow the log-permeability $Y(x) \equiv \log k(x)$ with a Gaussian process prior that describes our knowledge of the field before collecting the present data.

The **inverse problem** consists of inferring $k(x)$ from noisy observations of $u$ at $N$ distinct spatial locations $x_i$:

$$y_i = u(x_i) + \epsilon_i, \qquad i = 1 \ldots N. \tag{2}$$

Here the observational errors $\epsilon_i$ are i.i.d. Gaussian with mean zero and variance $\sigma_\epsilon^2$, i.e., $\epsilon_i \sim N(0, \sigma_\epsilon^2)$. Moreover, these errors are assumed to be independent of $k(x)$.

### 1.1.2 Parameters and data

Parameters and boundary conditions for equation (1) are specified as follows:

- The Neumann flux is $F = -1.0$.
- The Dirichlet datum is $u_r = 1$.
- The source term $s(x)$ mimics $M = 4$ "injection wells":

$$s(x) = \sum_{i=1}^{M} \frac{\theta}{\delta\sqrt{2\pi}} \exp\left(-\frac{(x - m_i)^2}{2\delta^2}\right).$$

  The wells are distributed evenly through the interior of the domain: $(m_1, \ldots, m_4) = (0.2, 0.4, 0.6, 0.8)$. The source strength is $\theta = 0.8$ and the source width is $\delta = 0.05$.

**Data** for inference are provided in the file `inferencedata.mat`. This file contains two variables: `xobserved`, indicating the coordinate $x_i$ of each noisy pressure measurement $y_i$, $i = 1 \ldots N$, and `Uobserved`, indicating the actual values $y_i$. There are $N = 10$ measurements.

For reference, the "truth" permeability field used to generate the data is provided in `ktrue.mat`. This field is *not* to be used when solving the inverse problem. You can use it afterwards for comparison to your posterior samples and statistics.

### 1.1.3 Priors

We endow the log-permeability $Y(x) = \log k(x)$ with a stationary Gaussian process prior. In other words, $Y \sim \mathcal{GP}(\mu, C)$ with constant mean $\mu(x) = \mu_Y$ and exponential covariance kernel

$$C(x_1, x_2) = \sigma_Y^2 \exp\left(-\frac{|x_1 - x_2|}{L}\right).$$

Use the following parameter values: mean $\mu_Y = 1.0$, correlation length $L = 0.3$, and variance $\sigma_Y^2 = 0.3$.

## 1.2 Questions

(a) Given the data in `inferencedata.mat`, infer the permeability. More specifically: **develop and implement an MCMC scheme** to characterize the posterior distribution of $k(x)$ conditioned on the data. In this part of the project, you should assume that the data are observed with a noise variance $\sigma_\epsilon^2 = 10^{-4}$.

To solve this problem, first think about a *finite-dimensional* representation of the permeability field $k(x)$. (Such a representation will yield a slightly different prior—an approximation of the original prior distribution—at least to some degree. That is okay.) Many parameterizations are feasible and will yield similar results, but for

computational convenience one would like a parameterization that is relatively low-dimensional and that minimizes correlations (at least under the prior distribution). To that end, we suggest a truncated Karhunen-Loève (K-L) representation of the Gaussian field $Y(x)$, since there is a bijection between $Y(x)$ and $k(x)$. Random coefficients of the K-L expansion will then become the direct objects of inference.

To communicate and validate your MCMC results, please make the following plots and carry out the associated analysis:

- How well is your MCMC chain mixing? Plot time traces for several components of the MCMC chain. For each component, compute and plot the autocorrelation along the chain as a function of lag.
- Plot the posterior densities of the parameters you are using to describe the permeability $k(x)$. (Hint: use MATLAB's `plotmatrix` command, or some Julia/python equivalent, to easily generate one-parameter histograms and scatterplots of pairs of parameters. Or use kernel density estimation to visualize the corresponding densities.)
- Plot the marginal posterior mean and standard deviation of the log-permeability as a function of the spatial coordinate $x$. Is the posterior stationary? Compare the posterior mean to the "true" log-permeability in `ktrue.mat`. Comment on the extent of the agreement/disagreement.
- Plot the posterior covariance of the log-permeability field. Compare it to the prior covariance.

(b) Now suppose that the variance $\sigma_\epsilon^2$ of the observational error is unknown. Treat this variance as a hyperparameter: endow it with an appropriate hyperprior and now jointly infer $k(x)$ and $\sigma_\epsilon^2$. In other words, construct an MCMC sampler that efficiently explores the joint posterior distribution of $k(x)$ and $\sigma_\epsilon^2$.

Plot the marginal posterior density of $\sigma_\epsilon^2$ and compare it with the hyperprior density.

As in part (a), plot the posterior mean and standard deviation of the log-permeability versus the spatial coordinate $x$. How do the marginal distributions of log-permeability differ from those obtained in part (a)?

(c) Plot the *posterior predictive density* of $u(x = 0)$. First, do this assuming a fixed observational error variance $\sigma_\epsilon^2 = 10^{-4}$ as in part (a). Next plot posterior predictive density of $u(0)$ for the case where the noise magnitude is inferred from data, as in part (b).

(d) Now suppose that you can observe the pressure only at *one* spatial location rather than at ten. Where should your one pressure monitor/well be placed in order to minimize the spatially averaged posterior variance of $k(x)$?

### 1.3 Hints and Suggestions

1. Try a few different truncations of the K-L expansion of $Y(x)$ to evaluate their impact on the posterior (and on the mixing of your MCMC chain).

2. In part (a), a simple random-walk Metropolis scheme should work well, but will require some tuning of the proposal covariance. It is not much more effort to implement an **adaptive Metropolis** scheme that adapts the proposal covariance from past samples. We recommend trying this algorithm! Note that one can use a cheap *recursive* update of the sample covariance in this algorithm.

3. In part (b), choose a *conjugate prior* for the noise variance and think about the form of the full conditional distribution of $\sigma_\epsilon^2$.

4. The parameters of the noise variance prior in part (b) can have a dramatic effect on the posterior. To set these parameters, begin by creating a prior centered tightly on the nominal value of the noise variance $\sigma_\epsilon^2$ from part (a). Then *gradually* relax and broaden this prior and watch its effect on the posterior.

5. To avoid numerical underflow (e.g., in the accept-reject step of a Metropolis–Hastings algorithm, among other settings) one should almost always work with the *logarithm* of the posterior density, rather than the value of the density itself.

6. Since your MCMC chains are of finite length, don't forget about the practical notion of "burn-in." In other words, look at the time-traces of your MCMC chains and throw away an initial interval of samples where mixing is poor or where the chain appears far from stationarity. Use only the post burn-in samples to estimate any posterior expectations or densities. Doing so may reduce the bias of your estimates.

7. A completely rigorous approach to part (d) is beyond the intended scope of this project. Some trial and error is perfectly acceptable here.

8. Throughout this project, the prior covariance of $k$ is assumed fixed. More sophisticated approaches might involve inferring aspects of the prior covariance itself from data, in a hierarchical Bayesian approach. A simple example of this would involve treating $L$ or $\sigma_Y^2$ as hyperparameters.

## 2 Bayesian inverse problem via the ensemble Kalman update

The MCMC method you developed in the previous part of the project is asymptotically *exact*; while it has a non-zero bias and variance after any finite number of MCMC steps, the bias and variance both go to zero as the number of steps goes to infinity. The bias in

particular decays quite rapidly, becoming negligible in practice for a sufficiently long and well-mixing chain.

Approximate inference algorithms, on the other hand, make simplifying assumptions at the expense of exactness. Here we will consider one type of approximate inference algorithm, based on the analysis/inference step of the ensemble Kalman filter. This approach makes a Gaussian approximation to the joint distribution of parameters and observations to construct a simple—in particular, an affine—prior-to-posterior transformation.

## 2.1 Questions

Consider the inverse problem from Question 1.2(a) with known noise variance. Let the parameters of the inference problem (e.g., coefficients of the K-L expansion of $Y(x)$) be denoted $\boldsymbol{\theta} \coloneqq (\theta_1, \ldots, \theta_n)$. And let the observations be denoted $\boldsymbol{y} \coloneqq (y_1, \ldots, y_N)$. Now suppose you generate a collection (equivalently called an "ensemble") of prior samples $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^{M}$ via direct Monte Carlo.

(a) **Derive an affine prior-to-posterior transformation via the Kalman update formula** and apply it to each prior sample $\boldsymbol{\theta}^{(j)}$ in order to generate an approximate posterior ensemble. Generate the corresponding posterior ensemble of $k(x)$ realizations too.

(b) Compare these ensembles to posterior samples you generated via MCMC in Question 1.2(a), for equivalent problem settings. Comment on the degree to which various moments and marginal distributions coincide.

## 2.2 Hints and suggestions

1. The Kalman update contains covariance and cross-covariance terms which, in this problem, we may not know *a priori*. Instead, we will need to *estimate* them from samples. To that end, you can generate synthetic realizations of the observation vector $\boldsymbol{y}$ via equation (2).

2. Without introducing regularization methods, the size of the ensemble $M$ should not be smaller than the relevant dimensions $n$ or $N$. Moreover, increasing it far beyond these minimums will improve stability. Experiment with the ensemble size in your results.

# 3 Exact and approximate Bayesian filtering

The Lorenz-63 model is a three-dimensional nonlinear dynamical system (first introduced by Edward Lorenz in a famous 1963 paper) that describes the natural convection of a heated

fluid. The state at time $t$ is a three-dimensional vector $\boldsymbol{Z}(t) = (Z_1(t), Z_2(t), Z_3(t))$ whose dynamics are given by the ODE system,

$$\begin{aligned}\frac{dZ_1}{dt} &= -\sigma Z_1 + \sigma Z_2,\\ \frac{dZ_2}{dt} &= -Z_1 Z_3 + \rho Z_1 - Z_2,\\ \frac{dZ_3}{dt} &= Z_1 Z_2 - \beta Z_3,\end{aligned}$$

where $(\beta, \rho, \sigma)$ are fixed parameters. We will set $\beta = 8/3$, $\rho = 28$, and $\sigma = 10$, which produces chaotic solutions tending to the well-known Lorenz attractor. To make notation more compact, we will write the ODE system above simply as

$$\frac{d\boldsymbol{Z}}{dt} = f(\boldsymbol{Z}). \tag{3}$$

In this problem, we will experiment with two different classes of algorithms for *nonlinear filtering*. To set up the filtering problem, we will fix an assimilation time interval $\Delta t_{\text{obs}}$ and index the state at discrete observation times as $(\boldsymbol{Z}_k)_{k\geq 1}$, where $\boldsymbol{Z}_k = \boldsymbol{Z}(k\Delta t_{\text{obs}})$. For any observation step $k$, we will observe the entire state (i.e., all three components) perturbed with additive Gaussian noise. The observation model is thus $\boldsymbol{Y}_k \sim \mathcal{N}(\boldsymbol{Z}_k, \sigma^2 \boldsymbol{I}_3)$. Set $\sigma^2 = 4$.

To assess the performance of the filtering algorithms developed below, we will use the so-called "identical twin experiment" setting: a single trajectory $\boldsymbol{z}^*(t)$ of the system, starting from an arbitrarily chosen initial condition, is observed every $\Delta t_{\text{obs}}$ time units and perturbed with noise (according to the observation model) to create a sequence of synthetic observations $\{\boldsymbol{y}_k^*\}_{k\geq 1}$. These observations, along with an initial ensemble at time $t = 0$, serve as inputs to the filtering algorithm. The filter then produces sample approximations to the sequence of posterior distributions $\{p(\boldsymbol{Z}_k|\boldsymbol{y}_{1:k}^*)\}_{k\geq 1}$. Depending on the form of the filter, this sample approximation might involve non-uniform weights (as in a particle filter) or no weights (effectively, uniform weights, as in the ensemble Kalman filter).

From outputs of a filtering algorithm, we can estimate the posterior mean $\mathbb{E}[\boldsymbol{Z}_k|\boldsymbol{y}_{1:k}^*]$, the posterior covariance $\mathbb{C}\text{ov}(\boldsymbol{Z}_k|\boldsymbol{y}_{1:k}^*)$, or any other expectations at each step $k$. For instance, the (possibly weighted) ensemble mean $\bar{\boldsymbol{z}}_k \coloneqq \sum_{i=1}^M w_k^{(i)} \boldsymbol{z}_k^{(i)} \approx \mathbb{E}[\boldsymbol{Z}_k|\boldsymbol{y}_{1:k}^*]$ serves as a convenient point estimate of the true state $\boldsymbol{Z}_k^*$. One can then evaluate the "tracking" performance of such an estimator via the root-mean-squared error (RMSE), defined at any assimilation step $k$ as $\text{RMSE}_k \coloneqq \|\bar{\boldsymbol{z}}_k - \boldsymbol{z}_k^*\|_2/\sqrt{3}$. A useful result to report is then $\text{RMSE}_k$ averaged over a *large* number of assimilation steps (e.g., several thousand), following a spin-up period (see below). This provides an overall quantification of how well the ensemble/sample mean tracks the true state.

## 3.1 Questions

(a) Implement an ensemble Kalman filter (EnKF) for state estimation and apply it to the Lorenz-63 setup described above. You can implement either a "stochastic EnKF" with perturbed observations, or a deterministic "square-root" EnKF. Test your filter over a range of ensemble sizes $M$, and plot time-averaged RMSE versus ensemble size. How does the RMSE compare with the standard deviation of the observation noise?

(b) Now implement a particle (SIR) filter for the same problem. Explain what proposal distribution you are using for your particle filter. Again, evaluate performance over a range of ensemble sizes $M$ and plot time-averaged RMSE versus ensemble size. How large an ensemble is needed for the particle filter to produce lower time-averaged RMSE than the EnKF?

How large of an $M$ is needed for estimates of the posterior covariance to stabilize? Compare your posterior covariance estimates to those obtained via EnKF, noting that the covariance estimate produced by the particle filter is consistent (i.e., converges in probability to the true covariance as $M \to \infty$) while the covariance estimate produced by the EnKF is not.

(c) If you have time: experiment with how the time-averaged RMSE of each filter varies as you change the assimilation time interval $\Delta t_{\text{obs}}$.

## 3.2 Hints and Suggestions

1. To integrate (3) over the time interval $\Delta t_{\text{obs}}$ you may use any deterministic integration scheme you like, but note that the integration timestep should be somewhat smaller than $\Delta t_{\text{obs}}$; for instance, you might use an RK4 scheme with an integration step of $\Delta t = 0.05$.

2. To make the particle filter more stable, it is useful to add a small amount of stochasticity to the dynamics. The proper way to do this is to rewrite (3) as a stochastic differential equation (SDE) system and to use a proper SDE integration scheme, which appropriately scales the magnitude of the noise perturbations with the timestep. For the present purposes, however, you can add an independent increment of isotropic Gaussian noise to the state at the end of each integration step; a noise variance of $10^{-4}$ should work well.

3. To create a good ensemble and remove transient effects from measures of assimilation quality, it is helpful to introduce a "spin-up" phase. First, draw $M$ i.i.d. random samples of the initial condition, $\mathcal{N}(\mathbf{0}, \mathbf{I}_3)$. Then run your filtering algorithm for a large number of steps (e.g., several thousand), but assess the assimilation quality only using the last half of the assimilation cycles. Another possible experiment is to create your initial filtering ensemble by first running an ensemble of so-called "free model runs," i.e., sufficiently long simulations of (3) starting from random initial conditions

but without filtering; this will create an initial ensemble distributed over the Lorenz attractor. WIth such an ensemble, there will still be an initial transient following the start of filtering, which you should remove when assessing RMSE.

4. Since this system has only a three-dimensional state, no localization or other regularization tricks are needed for the EnKF or particle filter.