Markov Chain Monte Carlo for Inverse Problems

David Ochsner

July 15, 2020

Contents

1 Theory

1.1 Papers

1.1.1 Stuart et al: Inverse Problems: A Bayesian Perspective [3]

Theoretical Background

Notation Central equation:

$$y = \mathcal{G}(u) + \eta$$

with:

- $y \in \mathbb{R}^q$: data
- $u \in \mathbb{R}^n$: IC ("input to mathematical model")
- $\mathcal{G}(\cdot) : \mathbb{R}^n \to \mathbb{R}^q$: observation operator
- η: mean zero RV, observational noise (a.s. $\eta \sim \mathcal{N}(0, \mathcal{C})$)

1.1.2 Cotter et al: MCMC for functions [1]

Implementation, MCMC in infinite dimensions

1.1.3 Schneider et al: Earth System Modeling 2.0 [2]

Example for MCMC on ODE

1.2 Small results

1.2.1 Gaussian in infinite dimensions

This section is quite a mess, maybe you could suggest a not-too-technical introduction to infinite dimensional Gaussian measures?

Wiki: Definition of Gaussian measure uses Lesbesgue measure. However, the Lesbesgue-Measure is not defined in an infinite-dimensional space (wiki).

Can still define a measure to be Gaussian if we demand all push-forward measures via a linear functional onto $\mathbb{R}$ to be a Gaussian. (What about the star (E*, L$_*$) in the wiki-article? Are they dual-spaces?) (What would be an example of that? An example for a linear functional on an inf-dims space given on wikipedia is integration. What do we integrate? How does this lead to a Gaussian?)

How does this fit with the description in [1]? -> Karhunenen-Loéve

What would be an example of a covariance operator in infinite dimensions? The Laplace-Operator operates on functions, the eigenfunctions would be *sin*, *cos* (I think? This might not actually be so easy, see Dirichlet Eigenvalues). Are the eigenvalues square-summable?

Anyway, when a inf-dim Gaussian is given as a KL-Expansion, an example of a linear functional given as $f(u) = \langle \phi_i, u \rangle$ for ϕ_i an eigenfunction of $\mathcal{C}$, then I can see the push-forward definition of inf-dim Gaussians satisfied. ($\mathcal{C}$ spd, so ϕ_i s are orthogonal, so we just end up with one of the KH-"components" which is given to be $\mathcal{N}(0,1)$).

The problem is not actually in $\exp\bigl(-1/2x^T\mathcal{C}^{-1}x\bigr)$. What about $\exp\bigl(-1/2\bigl\|\mathcal{C}^{-1/2}x\bigr\|\bigr)$?

What about the terminology in [1]? Absolutely continuous w.r.t a measure for example?

How is the square root of an operator defined? For matrices, there seems to be a freedom in choosing whether $A = BB$ or $A = BB^T$ for $B = A^{1/2}$. The latter definition seems to be more useful when working with Cholesky factorizations (cf. `https://math.stackexchange.com/questions/2767873/why-is-the-square-root-of-cholesky-decomposition-equal-to-the-lower-`
but for example in the wiki-article about the matrix (operator) square root

(`https://en.wikipedia.org/wiki/Square_root_of_a_matrix`): "The Cholesky factorization provides another particular example of square root, which should not be confused with the unique non-negative square root."

1.2.2 Bayes' Formula & Radon-Nikodym Derivative

Bayes' Formula is stated using the Radon-Nikodym Derivative in both [1] and [3]:

$$\frac{\mathrm{d}\mu}{\mathrm{d}\mu_0} \propto \mathrm{L}(u),$$

where $\mathrm{L}(u)$ is the likelihood.

Write the measures as $\mathrm{d}\mu = \rho(u)\mathrm{d}u$ and $\mathrm{d}\mu_0 = \rho_0(u)\mathrm{d}u$ with respect to the standard Lesbesgue measure. Then we have

$$\int f(u)\rho(u)\mathrm{d}u = \int f(u)\mathrm{d}\mu(u) = \int f(u)\frac{\mathrm{d}\mu(u)}{\mathrm{d}\mu_0(u)}\mathrm{d}\mu_0 = \int f(u)\frac{\mathrm{d}\mu(u)}{\mathrm{d}\mu_0(u)}\rho_0(u)\mathrm{d}u,$$

provided that $\mathrm{d}\mu$, $\mathrm{d}\mu_0$ and f are nice enough (which they are since we're working with Gaussians). This holds for all test functions f, so it must hold pointwise:

$$\frac{\mathrm{d}\mu(u)}{\mathrm{d}\mu_0(u)} = \frac{\rho(u)}{\rho_o(u)}.$$

Using this we recover the more familiar formulation of Bayes' formula:

$$\frac{\rho(u)}{\rho_o(u)} \propto \mathrm{L}(u).$$

1.2.3 Acceptance Probability for Metropolis-Hastings

A Markov process with transition probabilities $t(y|x)$ has a stationary distribution $\pi(x)$.

- The existence of $\pi(x)$ follows from *detailed balance*:

  $$\pi(x)t(y|x) = \pi(y)t(x|y).$$

  Detailed balance is sufficient but not necessary for the existence of a stationary distribution.

- Uniqueness of $\pi(x)$ follows from the Ergodicity of the Markov process. For a Markov processto be Ergodic it has to:

  - not return to the same state in a fixed interval

– reach every state from every other state in finite time

The Metropolis-Hastings algorithm constructs transition probabilities $t(y|x)$ such that the two conditions above are satisfied and that $\pi(x) = P(x)$, where $P(x)$ is the distribution we want to sample from.

Rewrite detailed balance as

$$\frac{t(y|x)}{t(x|y)} = \frac{P(y)}{P(x)}.$$

Split up the transition probability into proposal $g(y|x)$ and acceptance $a(y, x)$. Then detailed balance requires

$$\frac{a(y,x)}{a(x,y)} = \frac{P(y)g(x|y)}{P(x)g(y|x)}.$$

Choose

$$a(y,x) = \min\left\{1, \frac{P(y)g(x|y)}{P(x)g(y|x)}\right\}$$

to ensure that detailed balance is always satisfied. Choose $g(y|x)$ such that ergodicity is fulfilled.

If the proposal is symmetric ($g(y|x) = g(x|y)$), then the acceptance takes the simpler form

$$a(y,x) = \min\left\{1, \frac{P(y)}{P(x)}\right\}. \tag{1}$$

Since the target distribution $P(x)$ only appears as a ratio, normalizing factors can be ignored.

1.2.4 Potential for Bayes'-MCMC when sampling from analytic distributions

How can we use formulations of Metropolis-Hastings-MCMC algorithms designed to sample from posteriors when want to sample from probability distribution with an easy analytical expression?

Algorithms for sampling from a posterior sample from

$$\rho(u) \propto \rho_0(u) \exp(-\Phi(u)),$$

where ρ_0 is the prior and $\exp(-\Phi(u))$ is the likelihood. Normally, we have an efficient way to compute the likelihood.

When we have an efficient way to compute the posterior ρ and we want to sample from it, the potential to do that is:

$$\Phi(u) = \ln(\rho_0(u)) - \ln(\rho(u)),$$

where an additive constant from the normalization was omitted since only potential differences are relevant.

When working with a Gaussian prior $\mathcal{N}(0, \mathcal{C})$, the potential takes the form

$$\Phi(u) = -\ln \rho(u) - \frac{1}{2}\left\|\mathcal{C}^{-1/2}u\right\|^2.$$

When inserting this into the acceptance probability for the standard random walk MCMC given in formula (1.2) in [1], the two Gaussian-expressions cancel, as do the logarithm and the exponentiation, leaving the simple acceptance described in 1.

This cancellation does not happen when using the pCN-Acceptance probablity. This could explain the poorer performance of pCN when directly sampling a probablity distribution.

1.2.5 Acceptance Probabilities for different MCMC Proposers

Start from Bayes' formula and rewrite the likelyhood $\mathrm{L}(u)$ as $\exp(-\Phi(u))$ for a positive scalar function Φ called the potential:

$$\frac{\rho(u)}{\rho_o(u)} \propto \exp(\Phi(u)).$$

Assuming our prior to be a Gaussian ($\mu_0 \sim \mathcal{N}(0, \mathcal{C})$).

Then

$$\rho(u) \propto \exp\left(-\Phi(u) + \frac{1}{2}\left\|C^{-1/2}u\right\|^2\right),$$

since $u^T C^{-1} u = (C^{-1/2}u)^T(C^{-1/2}u) = \langle C^{-1/2}u, C^{-1/2}u\rangle = \left\|C^{-1/2}u\right\|^2$, where in the first equality we used C being symmetric.

This is formula (1.2) in [1] and is used in the acceptance probability for the standard random walk (see also Acceptance Probability for Metropolis-Hastings)

$\mathcal{C}^{-1/2}u$ makes problems in infinite dimensions.

Todo: Why exactly is the second term (from the prior) cancelled when doing pCN?

1.2.6 Different formulations of multivariate Gaussians

Is an RV $\xi \sim \mathcal{N}(0, C)$ distributed the same as $C^{1/2}\xi_0$, with $\xi_0 \sim \mathcal{N}(0, \mathcal{I})$?

From wikipedia: Affine transformation $Y = c + BX$ for $X \sim \mathcal{N}(\mu, \Sigma)$ is also a Gaussian $Y \sim \mathcal{N}\left(c + B\mu, B\Sigma B^T\right)$. In our case $X \sim \mathcal{N}(0, \mathcal{I})$, so

$Y \sim \mathcal{N}\left(0, C^{1/2}\mathcal{I}C^{1/2^T}\right) = \mathcal{N}(0, C)$, since the covariance matrix is positive definite, which means it's square root is also positive definite and thus symmetric.

On second thought, it also follows straight from the definition:

$$\mathbf{X} \sim \mathcal{N}(\mu, \Sigma) \Leftrightarrow \exists \mu \in \mathbb{R}^k, A \in \mathbb{R}^{k \times l} \text{ s.t. } \mathbf{X} = \mu + A\mathbf{Z} \text{ with } \mathbf{Z}_n \sim \mathcal{N}(0,1) \text{ i.i.d}$$

where $\Sigma = AA^T$.

1.2.7 Autocorrelation of non-centered distributions

A common definition of the autocorrelation function of a series $\{X_t\}$ is (cite something here?)

$$R(\tau) = \mathbb{E}\left[X_t X_{t+\tau}^*\right], \tag{2}$$

which can be normalized by $\tilde{R}(\tau) = R(\tau)/R(0)$ [1].

For calculating R of a finite series $\{X_t\}_{t=1}^T$, the series can either be zero-padded or the summation limits adjusted accordingly:

$$R(\tau) = \sum_{t=1}^{T-\tau} X_t X_{t+\tau} \text{ for } \tau < T \tag{3}$$

This seems to be the definition that is used in the function `np.correlate`: `c_{av}[k] = sum_n a[n+k] * conj(v[n])` (where we get autocorrelation for `a=v=x`).

This gives the expected result for uniformly random noise in $[-1, 1]$. However, when shifting the same distribution by a constant factor to get uniformly random noise in $[0, 2]$, the autocorrelation decays approximately linearly: 1.

To see why this happens, split up the signal into its mean plus a mean-zero pertubation: $X_t = \overline{X} + \tilde{X}_t$, where $\overline{X} = \mathbb{E}[X]$ and $\mathbb{E}\left[\tilde{X}\right] = 0$. The normalized autocorrelation is then:

[1] Since we're only working with real numbers, the complex conjugate in the definition will be dropped from now on.

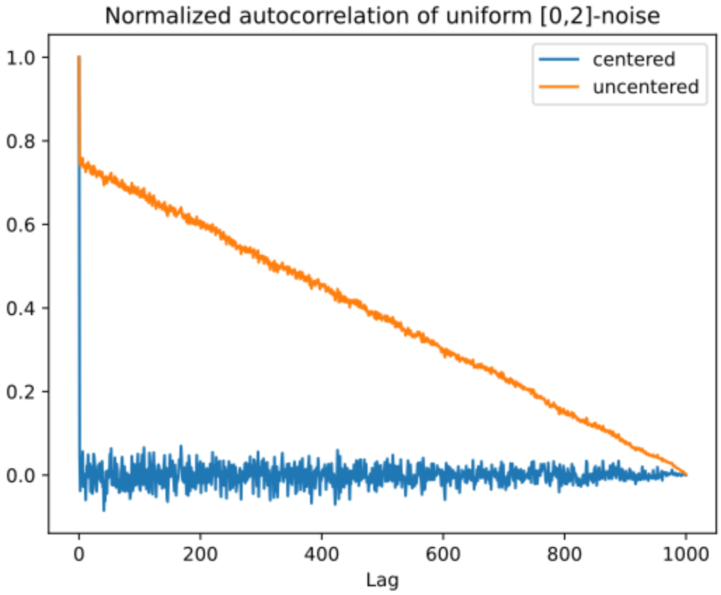

Figure 1: Autocorrelation function of the same signal $\{X_t \sim \mathcal{U}([0,2])\}$, once computed with `np.correlate` on the original series (uncentered), and once for $\{X_t - 1\}$ (centered).

$$\begin{aligned}
\tilde{R}(\tau) =& \frac{\sum_{t=1}^{T-\tau} X_t X_{t+\tau}}{\sum_{t=1}^{T} X_t^2} && (4) \\
=& \frac{\sum_{t=1}^{T-\tau} (\overline{X} + \tilde{X}_t)(\overline{X} + \tilde{X}_{t+\tau})}{\sum_{t=1}^{T} (\overline{X} + \tilde{X}_t)^2} && (5) \\
=& \frac{(T-\tau)\overline{X}^2 + \overline{X} \sum_{t=1}^{T-\tau} (\tilde{X}_t + \tilde{X}_{t+\tau}) + \sum_{t=1}^{T-\tau} \tilde{X}_t \tilde{X}_{t+\tau}}{T\overline{X}^2 + 2\overline{X} \sum_{t=1}^{T} \tilde{X}_t + \sum_{t=1}^{T} \tilde{X}_t^2} && (6) \\
=& \frac{(T-\tau)\overline{X}^2}{T(\overline{X}^2 + \text{var}(X))}, && (7)
\end{aligned}$$

where the last equality holds because

- $|\sum_{t=1}^{T-\tau} (\tilde{X}_t + \tilde{X}_{t+\tau})| < \epsilon$ and $|\sum_{t=1}^{T} \tilde{X}_t| < \epsilon$ when $T >> \tau$ by the weak law of large numbers
- $\sum_{t=1}^{T-\tau} \tilde{X}_t \tilde{X}_{t+\tau} = R(\tau) = 0$ for $\tau \neq 0$ and X "uncorrelated"
- $\sum_{t=1}^{T} \tilde{X}_t^2 = T \cdot \text{var}(\tilde{X}) = T \cdot \text{var}(X)$

This is a linear function in τ, which is what we see in the plots (plus quite some noise).

For $\overline{X} >> \text{var}(X)$, we get $\tilde{R}(\tau) = 1 - \tau/T$, which explains nicely why in the "uncentered" autocorrelation always linearly decays to 0, independently of the signal length T.

Considering these points, the python-function that computes the autocorrelation we're actually interested in [2] looks like this:

```python
def autocorr(x):
    x_centered = x - np.mean(x)
    result = np.correlate(x_centered, x_centered, mode='full')
    # numpy computes the correlation from -\infty to +\infty
    result = result[-len(x):]
    # normalize the result
    return result / result[0]
```

Much of this hassle could be avoided when the expectation value in the definition of the autocorrelation 2 would be computed correctly (not just "a

[2] The function I'm describing here is called auto-covariance function $K_{XX} = \mathbb{E}\left[(X_t - \mu)(X_{t+\tau} - \mu)^*\right]$.

posteriori" in the normalizing step with a much too large factor for bigger values of τ). However, this is not possible while still taking advantage of the huge speedup of doing the convolution operation in Fourier space.

1.2.8 Wasserstein metric

The Wasserstein metric is a distance function between distributions over a space M.

The p -th Wasserstein distance between two distributions μ, ν is given by

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{M \times M} d(x,y)^p \, \mathrm{d}\gamma(x,y) \right)^{\frac{1}{p}}, \tag{8}$$

with:

- don't confuse distance function $d(x, y)$ and measure $\mathrm{d}\gamma(x, y)$
- on wiki: first condition is that for for distributions with finite p-th moment, there is a point to where the "moment" (weighted distance) is finite -> normalizable

  How does that help exactly?
- explain marginal space
  - easy for single diracs

Equivalently, <formulation using expected value>, however this is "straightforward"

Connection to optimal transport (EMD):

- Conservation of mass -> marginals
- Minimizing cost of "reallocation" gives W_1

Linear (?) program formulation from Kjetil's OT library

2 Framework/Package Structure

The framework is designed to support an easy use case:

```
proposer = StandardRWProposer(beta=0.25, dims=1)
accepter = AnalyticAccepter(my_distribution)
rng = np.random.default_rng(42)
```

```
sampler = MCMCSampler(rw_proposer, accepter, rng)

samples = sampler.run(x_0=0, n_samples=1000)
```

There is only one source of randomness, shared among all classes and supplied by the user. This facilitates reproducability.

Tests are done with `pytest`.

2.1 Distributions

A class for implementing probability distributions.

```
class DistributionBase(ABC):
    @abstractmethod
    def sample(self, rng):
        """Return a point sampled from this distribution"""
        ...
```

The most important realisation is the `GaussianDistribution`, used in the proposers.

```
class GaussianDistribution(DistributionBase):
    def __init__(self, mean=0, covariance=1):
        ...

    def sample(self, rng):
        ...

    def apply_covariance(self, x):
        ...

    def apply_sqrt_covariance(self, x):
        ...

    def apply_precision(self, x):
        ...

    def apply_sqrt_precision(self, x):
        ...
```

The design of this class is based on the implementation in muq2. The `precision` / `sqrt_precision` is implemented through a Cholesky decomposition, computed in the constructor. This makes applying them pretty fast ($\mathcal{O}(n^2)$).

At the moment the there is one class for both scalar and multivariate Gaussians. This introduces some overhead as it has to work with both `float` and `np.array`. Maybe two seperate classes would be better.

Also, maybe there is a need to implement a Gaussian using the Karhunen-Loéve-Expansion?

2.2 Potentials

A class for implementing the potential resulting from rewriting the likelihood as

$$\mathrm{L}(u) = \exp(-\Phi(u)).$$

```
class PotentialBase(ABC):
"""
    Potential used to express the likelihood;
    d mu(u; y) / d mu_0(u) \propto L(u; y)
    Write L(u; y) as exp(-potential(u; y))
    """
    @abstractmethod
    def __call__(self, u):
        ...

    @abstractmethod
    def exp_minus_potential(self, u):
        ...
```

The two functions return $\Phi(u)$ and $\exp(-\Phi(u))$ respectively. Depending on the concrete potential, one or the other is easier to compute.

Potentials are used in the accepters to decide the relative weight of different configurations. They use the `__call__`-method to do that. Especially for high-dimensional error-terms, the value of the pdf of the error term can become very small, so it is important to implement this computing the log-pdf directly instead of manually exponentiating and running into issues with floating point number limitations.

2.2.1 AnalyticPotential

This potential is used when sampling from an analytically computable probability distribution, i.e. a known posterior. In this case

$$\exp(-\Phi(u)) = \frac{\rho(u)}{\rho_0(u)},$$

see `theory.org`

2.2.2 EvolutionPotential

This potential results when sampling from the model-equation

$$y = \mathcal{G}\left(u\right) + \eta,$$

with $\eta \sim \rho$. The resulting potential can be computed as

$$\exp(-\Phi(u)) = \rho(y - \mathcal{G}\left(u\right)).$$

2.3 Proposers

Propose a new state v based on the current one u.

```
class ProposerBase(ABC):
    @abstractmethod
    def __call__(self, u, rng):
        ...
```

2.3.1 StandardRWProposer

Propose a new state as

$$v = u + \sqrt{2\delta}\xi,$$

with either $\xi \sim \mathcal{N}\left(0, \mathcal{I}\right)$ or $\xi \sim \mathcal{N}\left(0, \mathcal{C}\right)$ (see section 4.2 in [1]).

This leads to a well-defined algorithm in finite dimensions. This is not the case when working on functions (as described in section 6.3 in [1])

2.3.2 pCNProposer

Propose a new state as

$$v = \sqrt{1 - \beta^2}u + \beta\xi,$$

with $\xi \sim \mathcal{N}\left(0, \mathcal{C}\right)$ and $\beta = \frac{8\delta}{(2+\delta)^2} \in [0, 1]$ (see formula (4.8) in [1]).

This approach leads to an improved algorithm (quicker decorrelation in finite dimensions, nicer properties for infinite dimensions)(see sections 6.2 + 6.3 in [1]).

The wikipedia-article on the Cholesky-factorization mentions the use-case of obtaining a correlated sample from an uncorrelated one by the Cholesky-factor. This is not implemented here.

2.4 Accepters

Given a current state u and a proposed state v, decide if the new state is accepted or rejected.

For sampling from a distribution $P(x)$, the acceptance probability for a symmetric proposal is $a = \min\{1, \frac{P(v)}{P(u)}\}$ (see `theory.org`)

```
class ProbabilisticAccepter(AccepterBase):
    def __call__(self, u, v, rng):
        """Return True if v is accepted"""
        a = self.accept_probability(u, v)
        return a > rng.random()

    @abstractmethod
    def accept_probability(self, u, v):
        ...
```

2.4.1 AnalyticAccepter

Used when there is an analytic expression of the desired distribution.

```
class AnalyticAccepter(ProbabilisticAccepter):
    def accept_probability(self, u, v):
        return self.rho(v) / self.rho(u)
```

2.4.2 StandardRWAccepter

Based on formula (1.2) in [1]:

$$a = \min\{1, \exp(I(u) - I(v))\},$$

with

$$I(u) = \Phi(u) + \frac{1}{2}\left\|\mathcal{C}^{-1/2}u\right\|^2$$

.

See also `theory.org`.

2.4.3 pCNAccepter

Works together with the pCNProposer to achieve the simpler expression for the acceptance

$$a = \min\{1, \exp(\Phi(u) - \Phi(v))\}.$$

2.4.4 CountedAccepter

Stores and forwards calls to an "actual" accepter. Counts calls and accepts and is used for calculating the acceptance ratio.

2.5 Sampler

The structure of the sampler is quite simple, since it can rely heavily on the functionality provided by the Proposers and Accepters.

```python
class MCMCSampler:
    def __init__(self, proposal, acceptance, rng):
        ...

    def run(self, u_0, n_samples, burn_in=1000, sample_interval=200):
        ...

    def _step(self, u, rng):
        ...
```

3 Results

3.1 Analytic sampling from a bimodal Gaussian

3.1.1 Setup

Attempting to recreate the "Computational Illustration" from [1]. They use, among other algorithms, pCN to sample from a 1-D bimodal Gaussian

$$\rho \propto (\mathcal{N}(3,1) + \mathcal{N}(-3,1))\mathbb{1}_{[-10,10]}.$$

Since the density estimation framework for a known distribution is not quite clear to me from the paper, I don't expect to perfectly replicate their results.

They use a formulation of the prior based on the Karhunen-Loéve Expansion that doesn't make sense to me in the 1-D setting (how do I sum infinite eigenfunctions of a scalar?).

The potential for density estimation described in section is also not clear to me (maybe for a similar reason? What is u in the density estimate case?).

I ended up using a normal $\mathcal{N}(0, 1)$ as a prior and the potential described before, and compared the following samplers:

- (1) `StandardRWProposer` ($\delta = 0.25$) + `AnalyticAccepter`
- (2) `StandardRWProposer` ($\delta = 0.25$) + `StandardRWAccepter`
- (3) `pCNProposer` ($\beta = 0.25$) + `pCNAccepter`

The code is in `analytic.py`.

3.1.2 Result

All three samplers are able to reproduce the target density 2

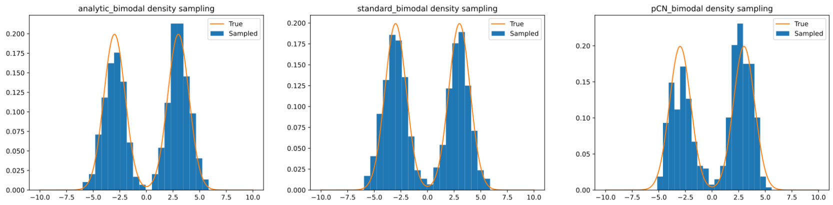

Figure 2: Burn-in: 1000, sample-interval: 200, samples: 500

The autocorrelation of the pCN sampler doesn't decay nearly as well as expected. This could be the consequence of the awkward formulation of the potential or a bad prior.

3.2 Bayesian inverse problem for $\mathcal{G}(u) = \langle g, u \rangle$

For $\mathcal{G}(u) = \langle g, u \rangle$ the resulting posterior under a Gaussian prior is again a Gaussian. The model equation is

$$y = \mathcal{G}(u) + \eta$$

with:

- $y \in \mathbb{R}$
- $u \in \mathbb{R}^n$
- $\eta \sim \mathcal{N}\left(0, \gamma^2\right)$ for $\gamma \in \mathbb{R}$

A concrete realization with scalar u:

- Ground truth $u^* = 1$
- $g = 1$
- $\gamma = 0.1$
- $y = 1.012$
- prior $\mathcal{N}\left(0, \Sigma_0 = (0.5)^2\right)$

leads to a posterior givne by 3.

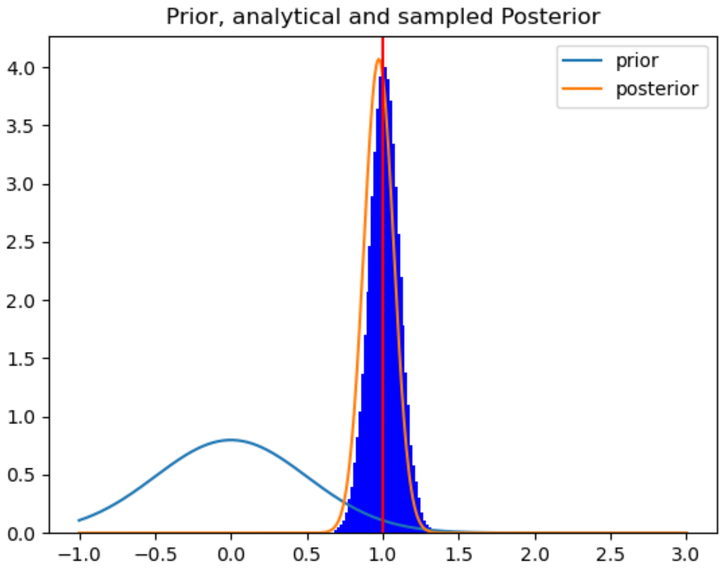

Figure 3: Posterior sampled from a chain with length 1'000'000, using a standard random walk sampler.

3.2.1 Posterior mean is off

According to Stuart, the posterior is a Gaussian with

- mean: $m = \frac{(\Sigma_0 g) y}{\gamma^2 + \langle g, \Sigma_0 g \rangle}$

- covariance: $\Sigma = \Sigma_0 - \frac{(\Sigma_0 g)(\Sigma_0 g)^*}{\gamma^2 + \langle g, \Sigma_0 g \rangle}$.

This is (according to Stuart) in the case of $q = 1$ ($\rightarrow y \in \mathbb{R}$) and $n \in \mathbb{N}$ ($\rightarrow g, u \in \mathbb{R}^n$).

I chose to work with $n = 1$, since then it's easy to visualize the posterior and the chain.

However, when $q = n = 1$, I'm doubtful whether the expressions given for the posterior are actually correct, since when Stuart discusses the zero-noise limit in the case of $n = q$, he proves that then the posterior is independent of the prior (which for $q = n$ in this example is not the case).

(Theorem 2.3 pg 463 [3])

3.2.2 Wasserstein convergence

As noted above, we have an analytical expression for the posterior. The steady state of the Markov Chain will sample from this distribution, so we expect the distributions sampled from the steady state to converge to the posterior when the sample length is increased (citation needed).

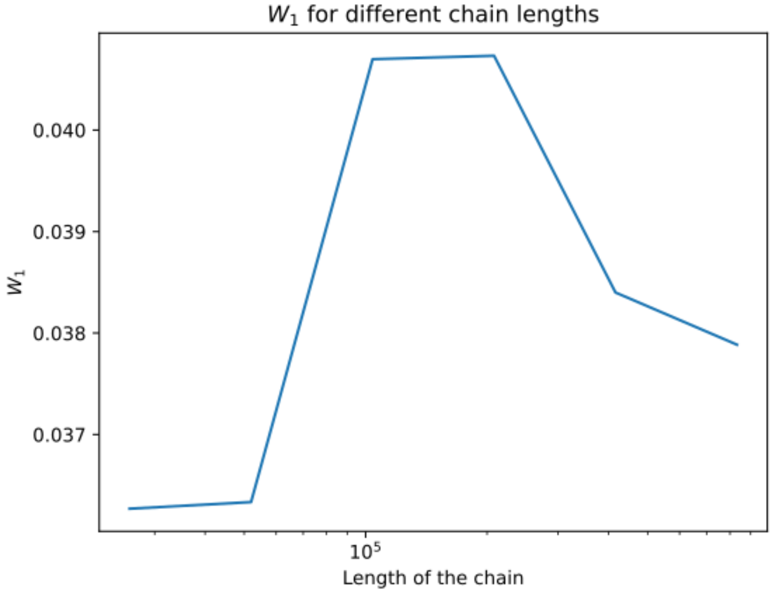

Figure 4: Wasserstein distance between the sample distributions and the analyical posterior. The same chain as above is used.

3.3 Bayesian inverse problem for $\mathcal{G}(u) = g(u + \beta u^3)$

Since the observation operator is not linear anymore, the resulting posterior is not Gaussian in general. However, since the dimension of the input u is 1, it can still be plotted.

The concrete realization with:

- $g = [3, 1]$
- $u = 0.5$
- $\beta = 0$
- $y = [1.672, 0.91]$
- $\gamma = 0.5$
- $\eta \sim \mathcal{N}\left(0, \gamma^2 I\right)$
- prior $\mathcal{N}(0, \Sigma_0 = 1)$

however leads to a Gaussian thanks to $\beta = 0$. The mean is $\mu = \frac{\langle g,y \rangle}{\gamma^2 + |g|^2} \approx 0.58$. Plot: 5

The pCN-Sampler with $\beta = 0.25$ (different beta) had an acceptance rate of 0.576.

For $\beta \neq 0$, the resulting posterior is not a Gaussian. Still $n = 1$, so it can be plotted. Just numerically normalize the analytical expression of the posterior?

3.4 Lorenz96 model

3.4.1 Model

Based on: Properly cite this!

Lorenz, E. N. (1996). Predictability—A problem partly solved. In Reprinted in T. N. Palmer & R. Hagedorn (Eds.), Proceedings Seminar on Predictability, Predictability of Weather and Climate, Cambridge UP (2006) (Vol. 1, pp. 1–18). Reading, Berkshire, UK: ECMWF.

Equation A system of ODEs, representing the coupling between slow variables X and fast, subgrid variables Y. The system is used in [2] to illustrate different algorithms for earth system modelling.

$$\frac{\mathrm{d}X_k}{\mathrm{d}t} = -X_{k-1}(X_{k-2} - X_{k+1}) - X_k + F - hc\bar{Y}_k \tag{9}$$

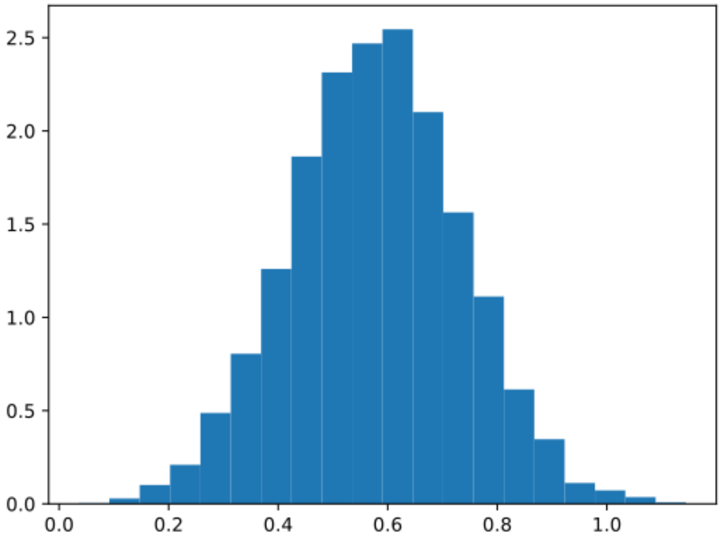

Figure 5: $N = 5000, \mu \approx 0.58$

$$\frac{1}{c}\frac{\mathrm{d}Y_{j,k}}{\mathrm{d}t} = -bY_{j+1,k}(Y_{j+2,k} - Y_{j-1,k}) - Y_{j,k} + \frac{h}{J}X_k \tag{10}$$

- $X = [X_0, ..., X_{K-1}] \in \mathbb{R}^K$
- $Y = [Y_{j,0}|...|Y_{j,K-1}] \in \mathbb{R}^{J\times K}$
  $Y_{j,k} = [Y_{0,k}, ..., Y_{J-1,k}] \in \mathbb{R}^J$
- $\bar{Y}_k = \frac{1}{J}\sum_j Y_{j,k}$
- periodic: $X_K = X_0$, $Y_{J,k} = Y_{0,k}$
- Parameters $\Theta = [F, h, c, b]$
- h: coupling strength
- c: relative damping
- F: external forcing of the slow variables (large scale forcing)
- b: scale of non-linear interaction of fast variables
- $t = 1 \Leftrightarrow 1$ day (simulation duration is given in days)

b or J? In the original paper, the equations are given in a different form, namely all explicit occurences of J above (in the fast-slow interaction) are replaced by b. Since in both concrete realizations (1996 & 2017) are identical and conviniently have $b = J = 10$, the difference doesn't lead to different results for that setup.

"Looking ahead" vs. "Looking back" Comparing nonlinearity terms

$$-X_{k-1}(X_{k-2} - X_{k+1})$$

$$-bY_{j+1,k}(Y_{j+2,k} - Y_{j-1,k})$$

for a given Y_k, does the "direction" of the $Z_{k\pm1}Z_{k\pm2}$ $(Z = X, Y)$ matter?

I don't think so, since the interaction with the other variable is only via point-value and average, and the nonlinearity is periodic.

A bit more formally: The PDE is invariant under "reversing" of the numbering: $Y_{j,k} \to Y_{J-j,k}$ which is the same as switching $+ \leftrightarrow -$ in the only "asymmetric" term.

Addendum 2 days later: Need to define more clearly what it means for the direction to matter. In the original paper on page 12, it is described how "active areas [..] propagate slowly eastward", while "convective activity tends to propagate westward within the active areas" (rephrased from paper). The paper also explicitly mentions the signs of the subscripts in that context. So some characteristics of the solution are definitely affected. What about the stuff we care about (statistical properties, chaotic behaviour)?

Properties For $K = 36$, $J = 10$ and $\Theta = [F, h, c, b] = [10, 1, 10, 10]$ there is chaotic behaviour.

Energy The nonlinearities conserve the energies within a subsystem:

- $E_X = \sum_k X_k^2$
  $\frac{1}{2}\frac{\mathrm{d}(\sum_k X_k^2)}{\mathrm{d}t} = \sum_k X_k \frac{\mathrm{d}X_k}{\mathrm{d}t} = -\sum_k(X_kX_{k-1}X_{k-2} - X_{k-1}X_kX_{k+1}) = 0$,
  where the last equality follows from telescoping + periodicity
- $E_{Y_k} = \sum_j Y_{j,k}^2$
  which follows analogously to the X -case

The interaction between fast and slow variables conserves the total energy:

- $E_T = \sum_k (X_k^2 + \sum_j Y_{j,k}^2)$
  $\frac{1}{2}\frac{\mathrm{d}E_T}{\mathrm{d}t} = \sum_k X_k \frac{\mathrm{d}X_k}{\mathrm{d}t} + \sum_j Y_{j,k} \frac{\mathrm{dY}_{j,k}}{\mathrm{d}t} = \sum_k X_k(-\frac{hc}{J}\sum_j Y_{j,k}) + \sum_j Y_{j,k}(\frac{hc}{J}X_k) = \sum_k -\frac{hc}{J}X_k(\sum_j Y_{j,k} + \frac{hc}{J}X_k(\sum_j Y_{j,k})) = 0$

In the statistical steady state, the external forcing F (as long as its positive) balances the dampling of the linear terms.

Averaged quantities

$$\langle \phi \rangle = \frac{1}{T}\int_{t_0}^{t_0+T} \phi(t)\,\mathrm{d}t$$

(or a sum over discrete values)

Long-term time-average in the statistical steady state: $\langle \cdot \rangle_\infty$

- $$\left\langle X_k^2 \right\rangle_\infty = F\left\langle X_k \right\rangle_\infty - hc\left\langle X_k \bar{Y_k} \right\rangle_\infty \forall k \tag{11}$$
  (multiply X -equation by X, all X_k s are statistically equivalent, $\frac{\mathrm{d}\langle X \rangle}{\mathrm{d}t} = 0$ in steady state)

- $$\left\langle \bar{Y_k^2} \right\rangle_\infty = \frac{h}{J}\left\langle X_k \bar{Y_k} \right\rangle_\infty \forall k \tag{12}$$

(Quasi) Ergodicity Whether chaotic regions of the phase space of a system are ergodic seems not be an easy question to answer (citation needed probably) [3] Are there any inaccessible regions in phase space for the Lorenz system? I can't think of any. However, there seem to be "traps" that take the system out of it's chaotic behaviour ($X_i = c$, $Y_i = a$). These somehow destroy ergodicity. Are they somehow "measure 0" or something?)/. However, for the purposes of this section (which deals with finite time anyway), it is enough to assert that

for the Lorenz system, for sufficiently long times, the time-average converges to the "space-average" over phase-space:

$$\langle f \rangle_\infty = \lim_{T\to\infty}\int_0^T f(Z(t))\,\mathrm{d}t = \int_{\mathbb{R}^{K(J+1)}} f(x)\rho(x)\,\mathrm{d}x \tag{13}$$

[3] Are there any inaccessible regions in phase space for the Lorenz system? I can't think of any. However, there seem to be "traps" that take the system out of it's chaotic behaviour ($X_i = c$, $Y_i = a$). These destroy ergodicity. Are they somehow "measure 0" or something?

where $Z(t)$ is a phase space trajectory of the system and $\rho(x)$ is the probaility of the system in the statistical steady state to be in state x.

One sufficiently long simulation of the system gives information about all accessible [4] initial conditions. As a consequence, as long as the integration time of the system is "long enough", the chosen initial condition is meaningless and can even vary without changing the behaviour of the observation operator.

3.4.2 Model implementation

Implementing the model in python and using a locally 5-th order RK solver yields the following results (inital conditions are just uniformly random numbers in $[0, 1)$ since they don't matter for the long-term evolution of the chaotic system):

Reproducting the results of the original paper Running the setup with $K = 36, J = 10, (F, h, c, b) = (10, 1, 10, 10)$ gives the following states 6, which qualitatively agree with the results from Lorenz.

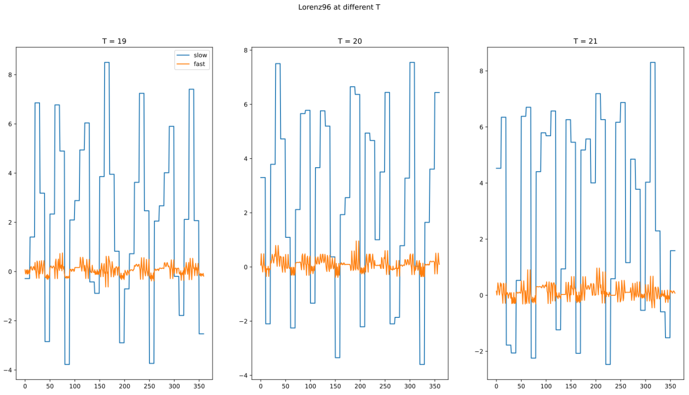

Figure 6: System around $T = 20$

The decay of the linear term and the forcing of the slow variables balance out after reaching the steady state, however there is a much bigger fluctuation in the energy than expected 7.

[4] Here a more precise definition of ergodicity of the system would help out. What I mean is "all sensible initial conditions".

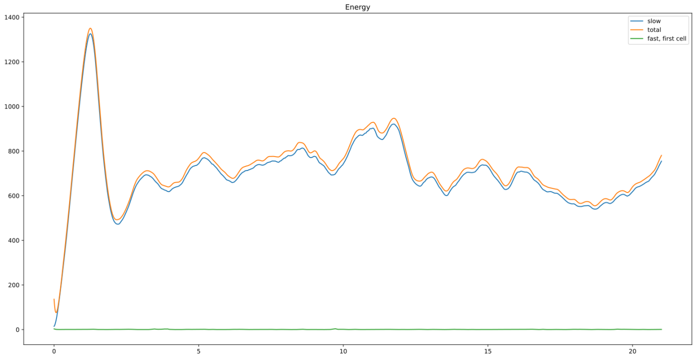

Figure 7: Energies in the system. $E_X >> E_{Y_k} > 0$

Equilibrium averages Analysis suggests certain long-term averages to be equal in the equilibrium.

3.4.3 MCMC

General point: The `RK45` method uses a predictor/corrector step and thus does non-uniform timesteps. However, in the following I compute time-averages with a simple `np.mean`, ignoring the different length of timesteps. It would be not impossible to write my own `time_average(y, t)`-function that takes the non-uniform timesteps into account. However I'm not sure how necessary this is, considering a forward-integration takes (2000) timesteps, so I suspect that differences are washed out a bit?

Setup Denote the Lorenz-96 system 9, 10 with parameters $\tilde{u} = [F, h, c, b]$ as $\mathcal{M}[\tilde{u}]$. It acts on the initial condition $z_0 = [X_0, Y_0] \in \mathbb{R}^{K(J+1)}$ to evolve the system for N_t timesteps and generate the phase space trajectory $Z = [{}^{X_1}_{Y_1}| \cdots |{}^{X_{N_t}}_{Y_{N_t}}] \in \mathbb{R}^{K(J+1) \times N_t}$:

$$Z = \mathcal{M}[\tilde{u}] z_0$$

Define the "moment function" $f(z) : \mathbb{R}^{K(J+1)} \to \mathbb{R}^{5K}$

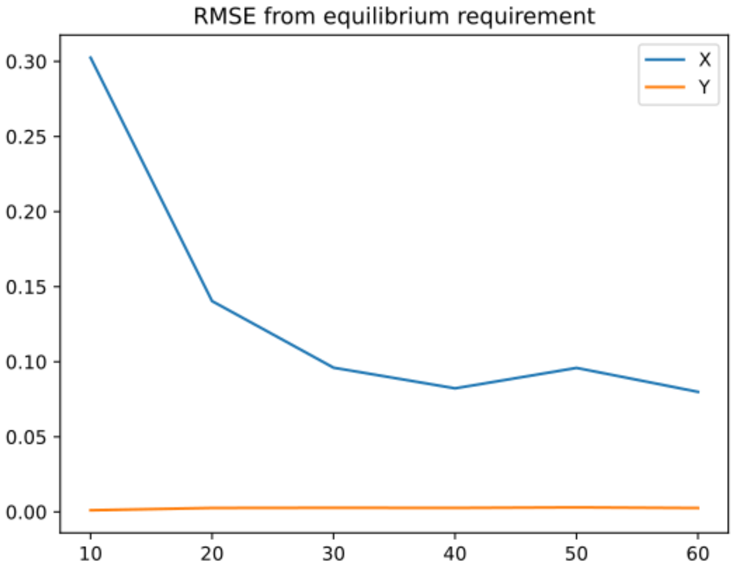

Figure 8: RMSE for long-term averages 11 and 12. Averaged over 10 runs

$$f(z) = \begin{bmatrix} X \\ \bar{Y} \\ X^2 \\ X\bar{Y} \\ \overline{Y^2} \end{bmatrix} \tag{14}$$

The MCMC-Algorithm then samples based on:

$$\langle f \rangle_\infty = \langle f \rangle_T (u) + \eta$$

with:

- $\langle f \rangle_\infty \approx \langle f \rangle_{T_r}$ with $T_r >> T$ over a simulation $\mathcal{M}[u^*]z_0$
- $\langle f \rangle_T (u)$ the time average over a simulation $\mathcal{M}[u_p + u]z_0$
- Due to the ergodic properties of $\mathcal{M}$ 3.4.1 , it doesn't really matter what z_0 is
- The parameter vector comes in many different variations:

  - $u^* \in \mathbb{R}^4_{\geq 0}$: true underlying parameters, used to compute the "data"
  - $u_p \in \mathbb{R}^4_{\geq 0}$: mean of the prior
  - $u \in \mathbb{R}^4$: pertubations to the prior mean, the actual input to the observation operator

- $\eta \sim \mathcal{N}(0, \Sigma)$, where $\Sigma = r^2[\text{var}(f)_{T_r}]$, where $r \in \mathbb{R}$ is the "noise level"

1. The parameter vector u

   The theoretical background assumes the prior to be a centered Gaussian ($\mu = 0$). Specifically, it matters during the proposal step, where the step is taken either with scaled sample from the prior or from a centered Gaussian with the covariance of the prior. A compromise would be to just ignore a nonzero prior-mean in the proposal, however I'm not sure if such a prior has other effects that invalidate the algorithm.

2. "Noise level"

   r is scaling of covariance matrix of noise term. This in turn is just step-width in proposal.

   TODO: Verify by checking acceptance rate for different noise levels

Concrete parameters The MCMC-Simulation was carried out with the following parameters:

- $K = 6, J = 4$
- Reference Simulation to get $\langle f \rangle_\infty$ and Σ:
  - $u^* = [F^*, h^*, c^*, b^*] = [10, 10, 1, 10]$
  - $T_r = 500$
- Noise $\eta \sim \mathcal{N}(0, \Sigma)$ with $\Sigma = r^2\text{diag}(\text{var}(f_{T_r})) \in \mathbb{R}^{5K \times 5K}$
- Noise level $r = 0.5$
- Prior $\mathcal{N}(u_p, \Sigma_0)$
  - $u_p = [F_0, h_0, b_0] = [12, 8, 9]$
  - $\Sigma_0 = \text{diag}([10, 1, 10])$
  - c was excluded from the sampling since it is very hard to estimate ("bad statistics")

  - The prior was chosen closer to the true value to make the job of the algorithm easier

- Sampling with pCN proposer and acccepter with $\beta = 0.25$
  - Evaluating the observation operator with a model-simulation of $T = 20$
  - Start sampling very close to true value: $u_0 = [-1.9, 1.9, 0.9]$ so that $u^* \approx u_p + u_0$
  - This means we can use a short burn-in of 100
  - Sample $N = 2500$ with a sample-interval of 2
  - The sample interval of 2 is very short, especially considering the long correlation time see below. But 2 is also what they used in the ESM paper.

Result

Density plot for posterior The resulting density plots show a improvement from the prior towards the true value 9. The estimation of the parameter F seems to be easier than b, where the prior and the posterior seem pretty much identical.

This slight improvement is however not unexpected, as the simulations I've done are much shorter than the ones in [2] $(K, J) = (6, 4)$ vs $(36, 10)$, $T = 20$ vs 100, $T_r = 500$ vs 46,416)

Should I do some more analysis here, like reporting sample means and covariances to compare posterior/prior not just visually?

ACF The autocorrelation decays for all three variables. As expected from the accuracy of the posteriors, the autocorrelation of F decays much faster than that of b. This simulation was done with a value of $\beta = 0.5$, which controls the "step size" of the proposer, and resulted in an acceptance rate of around 0.6. The value for β can now be tuned in such a way to get the fastest decay of the autocorrelation, which happens when the steps taken during sampling are big enough to quickly decorrelate the chain, while not being so big that the accepter declines too many of the steps.

3.5 Perturbed Riemann problem for Burgers' equation

3.5.1 Model

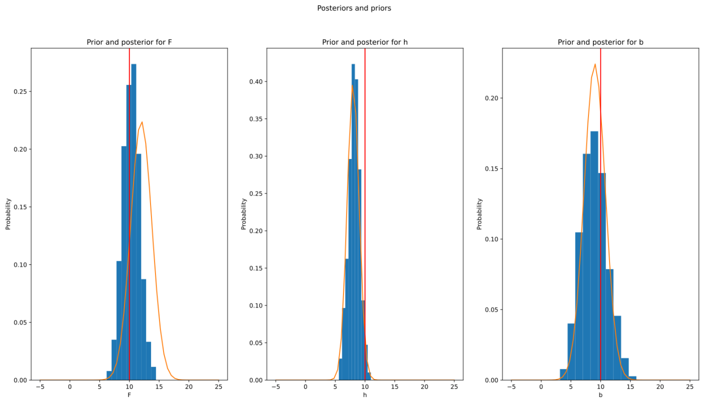

Figure 9: Prior and Posteriors after a 5000 sample MCMC run

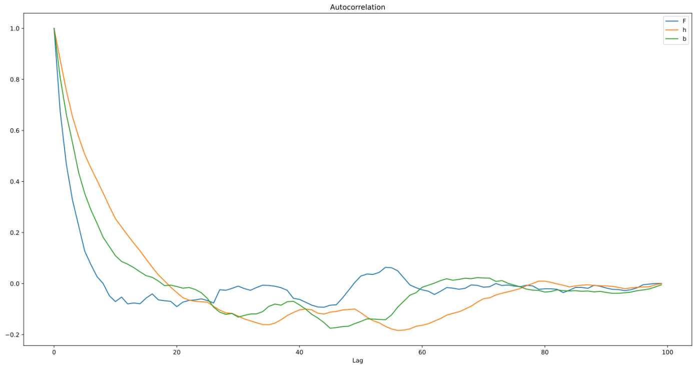

Figure 10: Autocorrelation of during the MCMC sampling. The functions are averaged over ten distinct parts of the chain

Burgers' Equation We consider the Burgers' equation

$$w_t + \left(\frac{w^2}{2}\right)_x = 0 \tag{15}$$

in the domain $(x,t) \in [-1,1] \times [0,1]$.

Riemann Problem The following family of initial conditions:

$$w(x,0) = \begin{cases} 1 + \delta_1 & \text{if } x < \sigma_0 \\ \delta_2 & \text{if } x > \sigma_0 \end{cases} \tag{16}$$

can be parametrized by the vector $u = [\delta_1, \delta_2, \sigma_0] \in \mathbb{R}^3$. As long as $\|u\| \ll 1$ we can expect w to behave very similarly to the usual 1,0-Riemann problem (in particular the location of the shock at $t = 1$ will be close to $x = 0.5$).

Discretization Blabla

3.5.2 MCMC

Hyperparameter tuning To optimize hyperparameters, namely choose the step-size (pCN: β, RW: δ) such that for a given chain length as many samples as possible can be used for estimation, the properties of the chain should be well defined and computable.

The two important characteristics are the burn-in b and the decorrelation time τ_0. Given these values for a chain of length N, the number of usable samples M is

$$M = \frac{N - b}{\tau_0}$$

Burn-In b The most fruitful approach seems to be to visually inspect the evolution of the parameter values and roughly decide when a steady-state is reached. This works nicely as long as the step-size is not too small and we actually reach a steady state.

A more formal approach would require to actually define a criterion for the parameter evolution in the chain that indicates when the steady state is reached. This is challenging, especially when no knowledge of the underlying values (ground truth) is used, and when the criterion should be valid for a wide range of step-sizes.

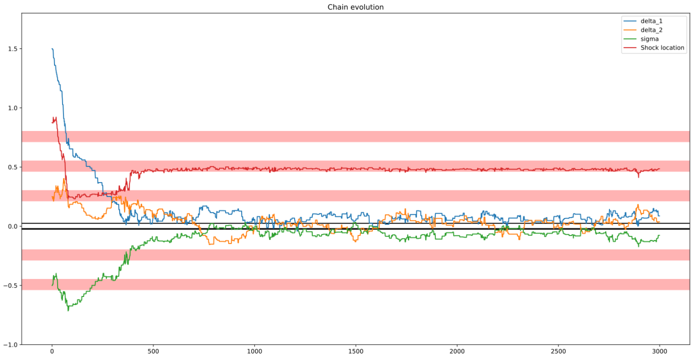

Figure 11: Chain evolution during sampling with $\beta = 0.1$. Visually it seems the the steady-state is reached after around 1000 samples.

Decorrelation time τ_0 After the burn-in is discarded from the original chain, the lag where the autocorrelation function first equals 0 gives the number of samples after which they become decorrelated [5].

Since after burn-in the chain is in the statistical steady state, the autocorrelation function is the same, regardless of which interval of the chain is investigated (this is not the case before discarding the burn-in).

Step size β Given a way to compute b and τ_0, the optimal β^* can be found as

$$\beta^* = \operatorname{argmax} M(\beta)$$

for a given chain length N (which is usually constrained by computational resources).

Generally, a bigger value of β will result in bigger steps proposed during the MCMC steps. This results in a shorter burn-in at the expense of more declined steps during the steady state, which results in longer decorrelation times.

Everything here also applies to δ, the step-size for the random-walk-MCMC algorithm. β and δ are related through $\beta^2 = \frac{8\delta}{(2+\delta)^2}$.

[5] A different, more involved criterion would be to define the decorrelation time as the integral over the autocorrelation function; Θ

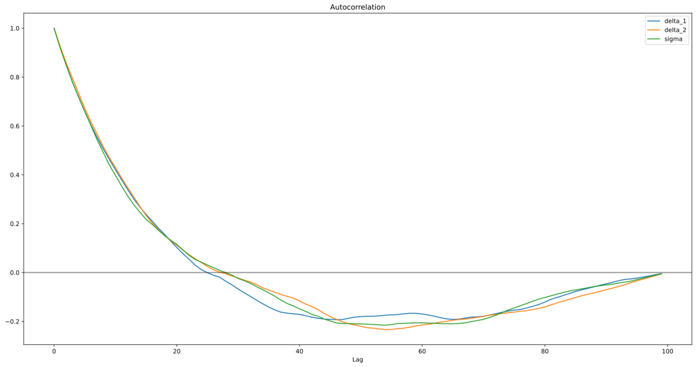

Figure 12: Autocorrelation function during sampling with $\beta = 0.1$. For this chain, $\tau_0 \approx 30$.

Setup As usual, we sample based on the equation

$$y = \mathcal{G}\left(u\right) + \eta$$

with:

- $y \in \mathbb{R}^q$: measurements obtained from a simulation of the ground truth
- $u \in \mathbb{R}^n$: vector parametrizing the pertubations to the Riemann initial conditions
- $\mathcal{G}\left(\cdot\right) : \mathbb{R}^n \to \mathbb{R}^q$: observation operator, measurements on the final state of the Riemann problem
- $\eta \sim \mathcal{N}\left(0, \gamma^2 \mathcal{I}_q\right)$: assumed observational noise [6]

Stuart et. al. describe some cases in [3] (Theorem 2.17) for overdetermined problems ($q > n$), where the posterior converges to a Dirac measure when $\gamma \to 0$. This however only applies to linear invertible observational maps, which is definitely not the case here. However for well-placed measurements we can definitely expect a sharp posterior.

[6] I took the liberty of renaming variables to match more closely Stuart's notation [3] and avoid collisions such as multiple occurences of β.

Observation operator $\mathcal{G}(u)$ We use the FVM to evolve the Riemann intial conditions 16 $w_u(x,0)$ until $T=1$ and then measure the resulting state around certain measurement points:

$$L_i(w) = 10 \int_{x_i-0.05}^{x_i+0.05} w(x,1)\mathrm{d}x \tag{17}$$

with $1 \leq i \leq 5$ and $x_1 = -0.5$, $x_2 = -0.25$, $x_3 = 0.25$, $x_4 = 0.5$, $x_5 = 0.75$.

The observation operator is then:

$$yeahhowdoyouwritethisoutlol$$

1. Placement of measurements

   The choice of the x_i s is crucial. If the shock is not contained in the measurement interval around and x_i, the Markov chain has no chance of determining the initial shock location σ_0 any more accurately than the spacing between measurements.

   Conversely, if the measurement interval is large enough, a single measurement around the shock gives enough information to determine all three parameters $\delta_1, \delta_2, \sigma_0$ simultaneously, provided the Markov chain "finds" to correct parameter configuration to place the shock in the measurement interval.

Ground truth measurements y y is obtained by applying the observation operator to the ground truth u^*.

$$u^* = [\delta_1^*, \delta_2^*, \sigma_0^*] = [0.025, -0.025, -0.02]$$

Noise $\eta \sim \mathcal{N}\left(0, \gamma^2 \mathcal{I}_5\right)$ with $\gamma = 0.05$.

Prior $\nu \sim \mathcal{N}\left(u_p, \varphi^2 \mathcal{I}_3\right)$, with

- $u_p = [1.5, 0.25, -0.5]$, which corresponds to
  - $\delta_1^p = 1.5$
  - $\delta_2^p = 0.25$
  - $\sigma_0^p = -0.5$
- $\varphi = 0.25$

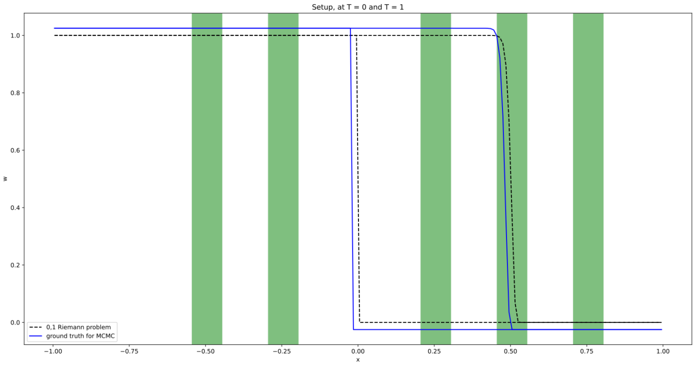

Figure 13: Setup for the MCMC experiment. The values for w at $T = 1$, once for the unperturbed Riemann problem, once for the ground truth of the simulation u^*. The green rectangles are the measurement intvervals of the observation operator : $\int_{x_i-0.05}^{x_i+0.05} w(x,1)\mathrm{d}x$, $x_i \in \{-0.5, -0.25, 0.25, 0.5, 0.75\}$.

Result

Investigating concrete values of β Three concrete values for β are investigated closer; $\beta_1 = 0.01$, $\beta_2 = 0.15$ and $\beta_3 = 0.5$. These values were chosen since they correspond to three significantly different behaviours of the Markov chain.

The pCN-proposer computes prospective new states as

$$v = \sqrt{1-\beta^2}u + \beta\xi$$

with $\xi \sim \mathcal{N}(0, \Sigma_0)$, where Σ_0 is the covariance of the prior. Ignoring the scaling of the current state, a characteristic step-size can be said to be $s = \beta\Sigma_0^{-\frac{1}{2}}$, which in the case of $\Sigma_0 = \gamma^2\mathcal{I}_q$ takes the simpler form

$$s = \beta\gamma \tag{18}$$

It is interesting to compare this value to other numbers in the system.

Comparing s to the distance between the prior-mean and the ground truth (namely for δ_1, for which this distance is largest) gives us a rough idea of the length of the burn-in we can expect.

Conversely, the ratio betwenn s and the measurement interval can indicate how high the acceptance ratio in the steady state might be. The idea is that if the stepsize is much larger than the measurement interval, proposed states will likely move the shock outside of the measurement interval and are thus often rejected. (This relationship is admittedly not so simple, since a large change in σ_0 can be compensated by an adjustment in a δ)

1. $\beta = 0.01$

   This very small value of beta gives a characteristic step size $s = 0.0025$. Moving uniformly from the prior-mean $\delta_1^p = 1.5$ to the ground truth $\delta_1^* = 0.025$ is expected to take around 600 steps.

   What we see in the actual chain evolution is quite different, the steps taken by are so small that the chain gets stuck in a local minimum and places the shock in the wrong measurement interval, even after 5000 steps. It can be argued that this is all part of the burn-in, and indeed also chains with a larger β sometimes spend some iterations with the shock-value in the completely wrong location.

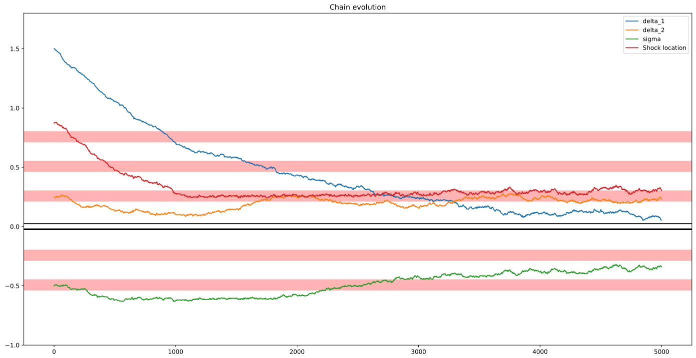

Figure 14: Evolution of the chain with $\beta = 0.01$. The small step size results in getting stuck in a local minimum, placing the around x=0.25 instead of x=0.5.

2. $\beta = 0.5$

   This large value of β results in stagnant behaviour in the steady state. Only very few moves are accepted, so the sampling interval has to be chosen very large to get adequately decorrelated samples (the autocorrelation function doesn't reach 0 until well after 100 samples). This is not too surprising when comparing the measurement interval of 0.1 around $x = 0.5$ with the step-size $s = 0.125$.

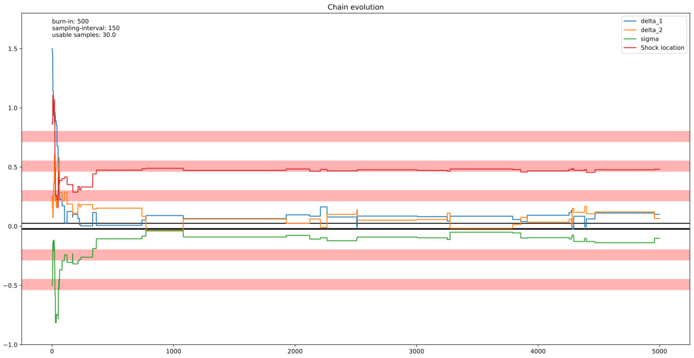

Figure 15: Evolution of the chain with $\beta = 0.5$. After the burn-in, very few moves are accepted, resulting in a long decorrelation time (even longer than written on the figure).

3. $\beta = 0.15$

   With this value of β we get a "healthy" behaviour of the chain: the steps are large enough to finish the burn-in in a reasonable time, while still being small enough to explore phase-space around a favourable state. The characteristic step size $s = 0.375$ reflects that fact.

   However, the region which we explore in the steady-state is still quite large, result in not very sharp posteriors. If sharper posteriors are needed, the value of β should be decreased, while making sure the burn-in doesn't take too long. An adaptive (decreasing whith chain length) value of β could help here.

4. Variable δ

   The idea to have a variable step-size (usually monotonically decreasing) to reap the benefits of both worlds (short burn-in and quick decorrelation in the steady state) is frequently used in optimization. There it is called *simulated annealing*, based on an analogy to tempering metals. The ground state (minimizing the free energy) of the system has favorable mechanical properties and is reached by letting the metal cool slowly. This process is "simulated" by decreasing the step-size of the Markov chain, which in a physical system corresponds to lowering the temperature. This procedure can be very successful at finding global minima of challenging objective functions.

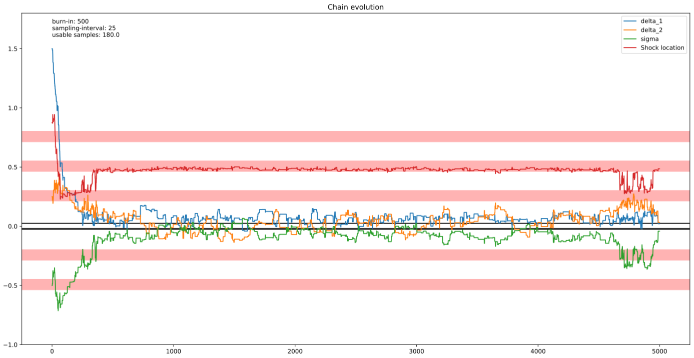

Figure 16: Evolution of the chain with $\beta = 0.15$. After the burn-in, the phase space around the ground-truth is explored nicely. Interesting is the small "excursion" around step 4800.

> Here, we chose a linearly decreasing step-size during burn-in, which is kept constant after. The results look promising and result in the best-performing chain.

pCN vs ordinary random Walk The pCN proposer generates new states as

$$v = \sqrt{1-\beta^2}u + \beta\xi,$$

while the ordinary random walk proposer does

$$x = u + \sqrt{2\delta}\xi$$

with $\xi \sim \mathcal{N}\left(0, \Sigma_0\right)$.

Equating the stepsize s gives $\delta = 0.01125$ being equivalent to $\beta = 0.15$. The chain seems pretty comparable, but the burn-in is noticably shorter. This can be attributed to the scaling of the current state $\sqrt{1-\beta^2}$, which "pulls" the proposed state towards the prior mean.

Posterior estimates (Just some histograms scattered around, see the captions)

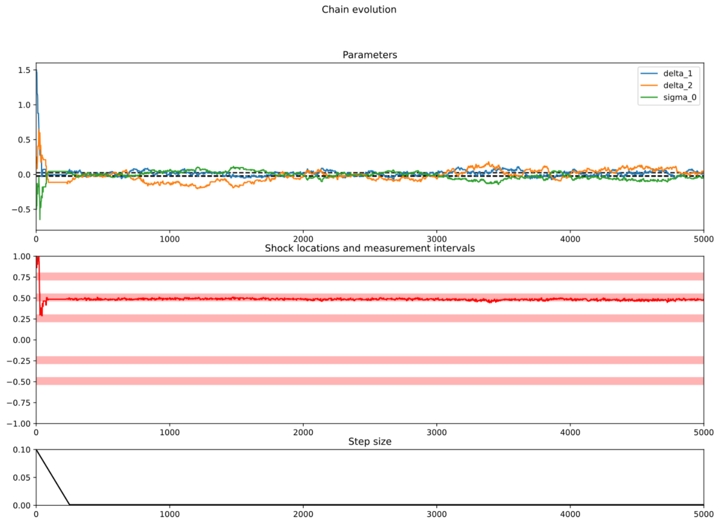

Figure 17: Evolution of the chain with a random walk proposal and a piecewise-linear δ, starting at $\delta_s = 0.1$ and decreasing to $\delta_e = 0.001$ during burn-in.

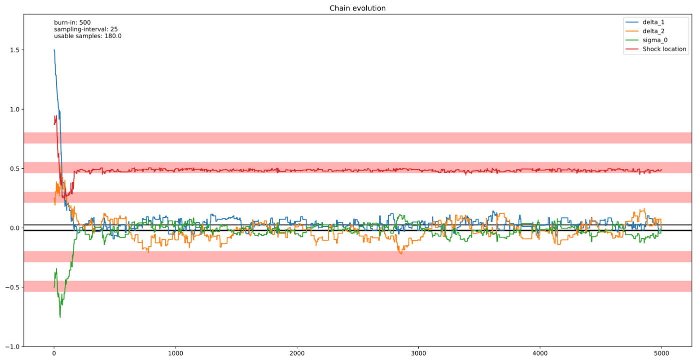

Figure 18: Evolution of the chain with a random walk proposal and $\delta = 0.01125$

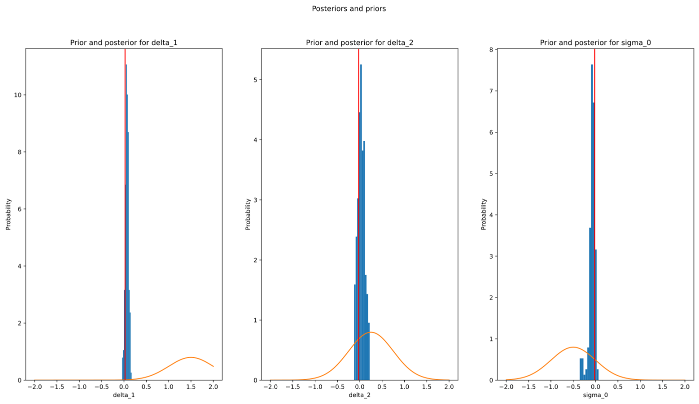

Figure 19: Posterior densities, taken from the pCN-chain shown above with $\beta = 0.15$, burn-in 500 and sampling interval 25.

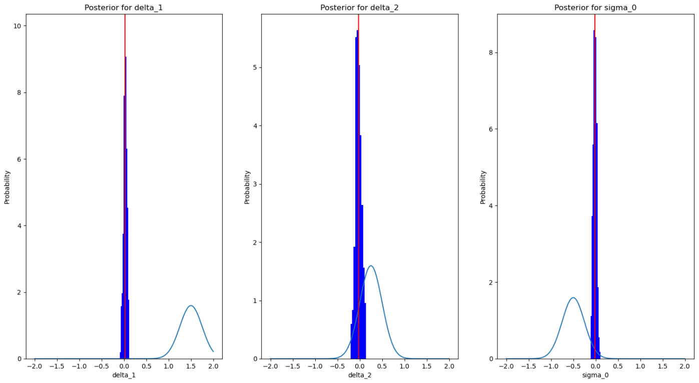

Figure 20: Posterior densities, taken from the RW-chain shown above with $\delta = 0.01125$, burn-in 500 and sampling interval 25.

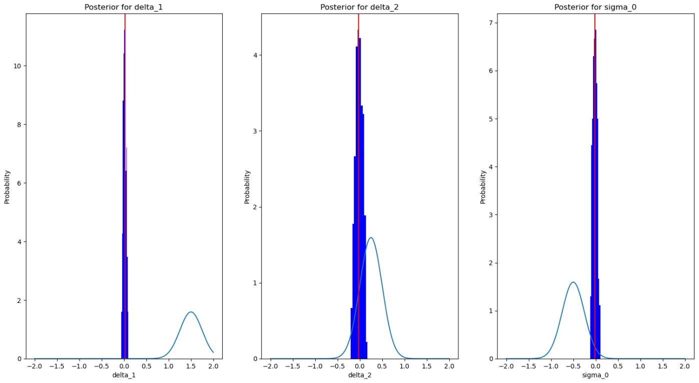

Figure 21: Posterior densities, taken from the RW-chain shown above with piecewise-linear $\delta = 0.1 \to 0.001$, burn-in 250 and sampling interval 20.

Convergence results

1. Convergence over chain length

   Idea: The posterior distributions of chains with increasing length should converge to a delta function located at the ground truth in the Wasserstein distance.

   Setup:

   - Create a *long* chain with the following characteristics:
     - $N = 100'000$
     - $\delta = \mathrm{PWL}(0.05 \to 0.001)$
     - other parameters as usual
   - Repeatedly bisect the chain, keeping the "right half" for the analysis and continuing with the "left half", repeat 4 times;

     get (unique) chains of length 50'000, 25'000, 12'500, 6'250.
   - Remove correlated states. Based on the AC-plot, every 20th state is uncorrelated;

     get 2500, 1250, 625, 312 uncorrelated samples.
   - Create a normalized histogram with 20 bins along each dimension for each sample-set

     (either in 3D for the whole parameter space or in 1D, just using the δ_1-marginal).

- Create a "normalized histogram" (actually just a {0,1}-array) corresponding to the ground truth.
- Compute the Wasserstein-distance ($p = 1$, Euclidean distance) between each of the sample-histograms and the ground truth

Result: Not really promising.

Improvement: Looking at the evolution of the chain, the correlation plot and the posteriors, I would have really expected to see some kind of convergence. The natural solution in the stochastic setting of the MCMC algorithm would be to work with averages. However, what exactly should be averaged?

- Take multiple chains of the same length and average their states before doing the histogram:

  Seems like nonsense, that would just decrease the variance of the resulting sample set and rougly correspond to a chain taking smaller steps (citation needed)
- Take multiple chains and average the densitites they produce: This seems like the most sensible approach to me, however:
  - Since the chains are in the steady state with the same hyperparameters, this just corresponds to taking all chains in the example above to be longer, but doesn't change anything else.
  - When increasing the length of the "base-chain" from 10'000 to 100'000, it didn't qualitatively change anything
- Average the Wasserstein distances of multiple chains of the same length:
  - I don't expect this to improve things, since the non-negative distances are unlikely to compensate the bigger values for the longer chains

2. Convergence over cell-size

   Idea: The posterior distributions of chains using a decreasing cell size in the underlying simulation should converge to a delta function located at the ground truth in the Wasserstein distance.

   Setup:

   - Create chains with the following characteristics:
     - $N = 10'000$

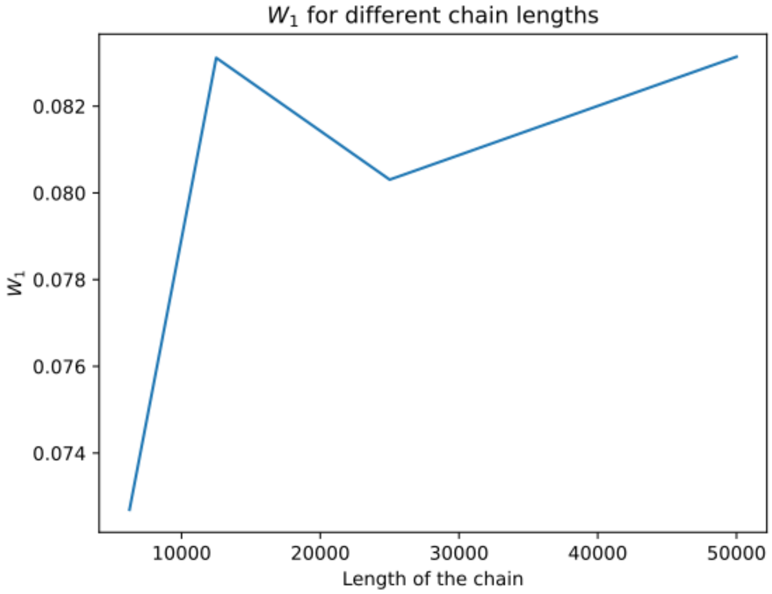

Figure 22: Wasserstein distance between the posterior and a delta-peak at the ground truth for different lengths of the chain

  - $\delta = \text{PWL}(0.05 \rightarrow 0.001)$
  - grid-spacing in the underlying simulation:
    split the $[-1, 1]$ domain into 32, 64, 128, 256 cells

- Remove correlated states and a burn-in of 250 steps. Based on the AC-plot, every 20th state is uncorrelated;

  get 4 sets of ~ 500 uncorrelated samples.

- Create a normalized histogram with 20 bins along each dimension for each sample-set.

- Create a "normalized histogram" (actually just a {0,1}-array) corresponding to the ground truth.

- Compute the Wasserstein-distance ($p = 1$, Euclidean distance) between each of the sample-histograms and the ground truth.

Result: Not really promising.

Improvement: Here it seems pretty clear what should be done to reduce the variance of the result and so hopefully get convergence: Average the densities over multiple chains / make the chains longer [7]

Further work: It might be interesting to compare the characteristic step size to the grid-size to get an idea of a lower bound of hoping to get useful results (when the gridspacing is bigger than steps taken in σ_0 the MCMC will have problems)

[7] In this case, since the global optimum is clearly attained in every chain, longer chains or averaging over multiple chains is equivalent. When it is not straight forward to verify that the chain reaches it's steady state (it might get stuck in a local optimum for an extended amount of time), it can be safer to use multiple chains instead of one long one.

References

[1] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster. *Statistical Science*, 28(3):424–446, August 2013. Publisher: Institute of Mathematical Statistics.

[2] Tapio Schneider, Shiwei Lan, Andrew Stuart, and João Teixeira. Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations. *Geophysical Research Letters*, 44(24):12,396–12,417, 2017. _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2017GL076101.

[3] A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, May 2010.

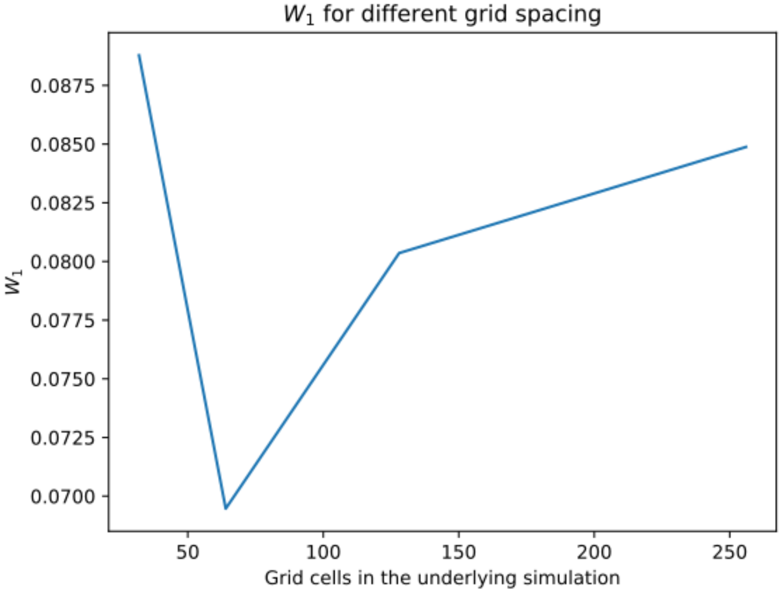

Figure 23: Wasserstein distance between the posterior and a delta-peak at the ground truth for different grid sizes