Hamiltonian Monte Carlo sampling in linear and non-linear geophysical systems

Lars Gebraad

August 10, 2017

In this document I will collect my thoughts, findings and other remarks on Hamiltonian Monte Carlo (HMC) sampling of posterior probability distributions. HMC sampling finds it's usefulness in proposing new models; as prior information becomes poorer classical methods of exploring model space suffer from increasing performance loss. For example, Metropolis-Hastings sampling draws new models just from the prior distribution, which, if largely incorrect, will lead to almost no useful exploration of the model space.

It is not understood how well HMC scales with increasing dimension. Performance should decrease quite drastically, as models need to be propagated using Hamiltonian Mechanics, but the acceptance rate will also most likely be affected. One of the aims of this work is to explore how sampling of simple linear models and more complex non-linear systems scales with increasing dimensions. Tuning parameters such as assigning masses and momenta are also investigated.

As of yet, three types of physical models are considered. First os a simple low dimensional arbitrary linear model, where we can control the input and forward model. The second model is based upon a straight ray tomography MATLAB program which is a more complicated, but still linear, system. Lastly, a vertical seismic profile with variable layer speed and thickness is highlighted.

The notation used might be a bit of a mix between Neal et al. [2011] and Andreas Fichtner's preliminary probabilistic inversion document, but I will try to be coherent and assign recognizable variable symbols.

1. Recap of probabilistic inversion and Hamiltonian Mechanics

Probabilistic inversion and classical sampling In probabilistic inversion one does not require regularization, in contrast to deterministic inversion. Instead, one explores the model space by quantifying uncertainties and calculating probabilities for any model. In a sense, this can be considered as the 'complete' solution. It is defined using Bayes' Theorem as follows;

$$p(\mathbf{q}|\mathbf{d}) = \frac{p(\mathbf{d}|\mathbf{q})\ p(\mathbf{q})}{p(\mathbf{d})} \tag{1}$$

Generally, we ignore the data evidence ($p(\mathbf{d})$) as it only provides scaling of the posterior.

If the prior is normally distributed in model space, we can define it by

$$p(\mathbf{q}) = k \cdot \exp\left(-\frac{1}{2}\left[\mathbf{q}-\mathbf{q_0}\right]^T \underline{C}_M^{-1}\left[\mathbf{q}-\mathbf{q_0}\right]\right) \tag{2}$$

where k is a scaling constant (i.e. for normalization), $\mathbf{q_0}$ the mean of the prior distribution and $\underline{C}_M^{-1}$ the inverse parameter covariance matrix. This inverse parameter covariance matrix is given by

$$\left[\underline{C}_M^{-1}\right]_{ij} = r_{ij}\sigma_i\sigma_j \tag{3}$$

with r_{ij} the correlation between parameters q_i and q_j and σ_i the standard deviation of parameter q_i.

The prior in data space, $p(\mathbf{d}|\mathbf{q})$, can be seen as the likelihood to see data based on a selected model. This generally incorporates a forward model, measurement uncertainties and forward modeling uncertainties. Because the quantification of forward modeling uncertainties is generally very hard to estimate usually the actual implementation of this is ignored. With only the forward model and measurement uncertainties (normally distributed) the prior in data space is as follows:

$$p(\mathbf{d}|\mathbf{q}) = k \cdot \exp\left(-\frac{1}{2}\left[G\left(\mathbf{q}\right)-\mathbf{d_0}\right]^T \underline{C}_D^{-1}\left[G\left(\mathbf{q}\right)-\mathbf{d_0}\right]\right) \tag{4}$$

with k again a scaling constant, $\underline{C}_D$ the data covariance matrix, d_0 the observed data and $G\left(\mathbf{q}\right)$ the forward modeled data based on parameters q and forward model G. This forward model G can be any non-linear model, but Hamiltonian Monte Carlo (HMC) inversion algorithms are greatly simplified if they are actually linear systems, as will be illustrated later.

When these prior data functions are combined one obtains the (improperly scaled) posterior. The negative exponent of the posterior is generally called the misfit and has a special role in many inversions. In deterministic inversion, the aim is usually to find the global minimum of this function. In probabilistic inversion,one tries to map the entire misfit by (pseudo) random sampling. The misfit is given by

$$\chi(\mathbf{q}) = \frac{1}{2}\left[\mathbf{q}-\mathbf{q_0}\right]^T \underline{C}_M^{-1}\left[\mathbf{q}-\mathbf{q_0}\right] + \frac{1}{2}\left[G\left(\mathbf{q}\right)-\mathbf{d_0}\right]^T \underline{C}_D^{-1}\left[G\left(\mathbf{q}\right)-\mathbf{d_0}\right]. \tag{5}$$

Exploring this misfit is typically done based on prior information. The Metropolis-Hastings algorithm draws new models entirely from the prior in model space, and accepts either by having a lower misfit or randomly based on the exponential increase in misfit if the proposed model has a larger misfit.

If the prior model is far off or has large uncertainties, classical samplers might have a hard time finding proper acceptable models. An example which will be expanded on in the first section is a linear model where prior data is off by double the forward modeled value, which on 10,000 samples lead to less than 100 accepted models. These acceptance rates are very wasteful of computing power. On the other side, drawing new models is computationally cheap relative to the more intensive HMC sampling.

A rather helpful way to write the misfit for a linear forward model with Gaussian uncertainties allows for easy computation of derivatives, as well as keeping notations clean. I start by rewriting the misfit functional as;

$$\begin{aligned}
\chi(\mathbf{q}) =& \frac{1}{2}\left[\mathbf{q}-\mathbf{q_0}\right]^T \underline{C}_M^{-1}\left[\mathbf{q}-\mathbf{q_0}\right] + \frac{1}{2}\left[\underline{G}\mathbf{q}-\mathbf{d_0}\right]^T \underline{C}_D^{-1}\left[\underline{G}\mathbf{q}-\mathbf{d_0}\right] \\
=& \frac{1}{2}\mathbf{q}^T\underline{C}_M^{-1}\mathbf{q} - \frac{1}{2}\mathbf{q_0}^T\underline{C}_M^{-1}\mathbf{q} - \frac{1}{2}\mathbf{q}^T\underline{C}_M^{-1}\mathbf{q_0} + \frac{1}{2}\mathbf{q_0}^T\underline{C}_M^{-1}\mathbf{q_0} \\
&+ \frac{1}{2}\left(\underline{G}\mathbf{q}\right)^T \underline{C}_D^{-1}\left(\underline{G}\mathbf{q}\right) - \frac{1}{2}\mathbf{d_0}^T\underline{C}_D^{-1}\left(\underline{G}\mathbf{q}\right) \\
&- \frac{1}{2}\left(\underline{G}\mathbf{q}\right)^T \underline{C}_D^{-1}\mathbf{d_0} + \frac{1}{2}\mathbf{d_0}^T\underline{C}_D^{-1}\mathbf{d_0} \\
=& \frac{1}{2}\mathbf{q}^T\underline{C}_M^{-1}\mathbf{q} - \frac{1}{2}\mathbf{q_0}^T\underline{C}_M^{-1}\mathbf{q} - \frac{1}{2}\mathbf{q}^T\underline{C}_M^{-1}\mathbf{q_0} + \frac{1}{2}\mathbf{q_0}^T\underline{C}_M^{-1}\mathbf{q_0} \\
&+ \frac{1}{2}\mathbf{q}^T\underline{G}^T\underline{C}_D^{-1}\underline{G}\mathbf{q} - \frac{1}{2}\mathbf{d_0}^T\underline{C}_D^{-1}\underline{G}\mathbf{q} - \frac{1}{2}\mathbf{q}^T\underline{G}^T\underline{C}_D^{-1}\mathbf{d_0} + \frac{1}{2}\mathbf{d_0}^T\underline{C}_D^{-1}\mathbf{d_0}.
\end{aligned} \tag{6}$$

Now the trick is to realize that all the components can be individually transposed without altering the equation. The fact that $\mathbf{a}^T\mathbf{b} = \mathbf{a}\cdot\mathbf{b} = \mathbf{b}\cdot\mathbf{a} = \mathbf{b}^T\mathbf{a}$ should be enough proof. Also realizing that covariance matrices are always symmetric, we rearrange the equation in second, first and zeroth order terms of $\mathbf{q}$:

$$\begin{aligned}
\chi(\mathbf{q}) =& \frac{1}{2}\mathbf{q}^T\left[\underline{C}_M^{-1} + \underline{G}^T\underline{C}_D^{-1}\underline{G}\right]\mathbf{q} - \left(\mathbf{q_0}^T\underline{C}_M^{-1} + \mathbf{d_0}^T\underline{C}_D^{-1}\underline{G}\right)\mathbf{q} \\
&+ \frac{1}{2}\mathbf{q_0}^T\underline{C}_M^{-1}\mathbf{q_0} + \frac{1}{2}\mathbf{d_0}^T\underline{C}_D^{-1}\mathbf{d_0}.
\end{aligned} \tag{7}$$

Or, substituting the different components;

$$\chi(\mathbf{q}) = \frac{1}{2}\mathbf{q}^T\underline{A}\,\mathbf{q} - \mathbf{b}\mathbf{q} + c, \tag{8}$$

where

$$\underline{A} = \underline{C}_M^{-1} + \underline{G}^T \underline{C}_D^{-1} \underline{G} \tag{9}$$

$$\mathbf{b} = \mathbf{q_0}^T \underline{C}_M^{-1} + \mathbf{d_0}^T \underline{C}_D^{-1} \underline{G} \tag{10}$$

$$c = \frac{1}{2} \mathbf{q_0}^T \underline{C}_M^{-1} \mathbf{q_0} + \frac{1}{2} \mathbf{d_0}^T \underline{C}_D^{-1} \mathbf{d_0}. \tag{11}$$

Note that $\mathbf{b}$ is a row vector, so $\mathbf{bq}$ is actually the dot product between $\mathbf{b}^T$ and $\mathbf{q}$.

Now using this notation, one can easily compute the gradient of the misfit as

$$\frac{\partial \chi}{\partial q_i} = A_{ij} q_j - b_i. \tag{12}$$

Note that repeated indices are summed. Note that using these quantities (which are precomputed every inversion) sped up propagation by 10 times.

Hamiltonian Mechanics and sampling One way to propose more acceptable models is to use HMC sampling. In this algorithm, the model is considered as a particle in n-dimensional space, where n denotes the number of inversion parameters. By not assigning new parameters but momenta in each dimension and propagating the model for a certain amount of time one proposes new models.

The propagation is based on Hamilton's Equations. These two equations interrelate energy of a system to it's position and momentum. Hamilton's equations for a trajectory of a particle are given in this n-dimensional space as;

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \tag{13}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}. \tag{14}$$

In these equations, q_i is the position in dimension i, p_i is the momentum in dimension i (given by $p_i = \mu_i \frac{dq_i}{dt}$). H stands for the Hamiltonian, which in physical Hamiltonian Mechanics is the sum of potential en kinetic energy.

The trick to HMC sampling is to consider the misfit functional χ as a gravitational potential. This allows us to propagate our model over model space as if it were a particle moving under the influence of gravity defined by the posterior distribution. The actual definitions for potential and kinetic energy then become:

$$K(\mathbf{p}) = \frac{1}{2} \sum_{i=1}^{n} \frac{p_i^2}{\mu_i}, \tag{15}$$

$$U(\mathbf{q}) = \chi(\mathbf{q}). \tag{16}$$

There are already many interesting options, remarks and conclusions to draw from this framework, but as it is usually nicely illustrated using a few examples, we will come

to that later. Noteworthy however that a simplification is done on the calculation of momenta. In a later analysis, I extend this definition. See also the note about equation (17).

One thing which is useful to note now is that the derivatives on the right hand sides of equations (13) and (14) now simplify, for the Hamiltonian derivative with respect to momenta only depends on kinetic energy, while the derivative with respect to position only depends on potential energy. The simplified representation is:

$$\frac{dq_i}{dt} = \frac{p_i}{\mu_i}, \tag{17}$$

$$\frac{dp_i}{dt} = -\left[\nabla_q \chi\right]_i. \tag{18}$$

An extension on this with a non-diagonal mass matrix will allows us to 'link' parameters together in the propagation. This will be analyzed later on. Typically, however, for simple cases one chooses a diagonal positive definite mass matrix, which leads to equation (17).

The propagation of models is done using a leapfrog scheme, in which one splits up each time step in three separate calculations. First the momentum is propagated half an original time step, then the model parameters are propagated a full step, after which the momentum catches up again. Since I will not alter much on this side of the algorithm, for specifics I refer to section 5.2.3.3 of Neal et al. [2011].

Acceptance of a new proposition works almost equal with HMC as it does with the Metropolis-Hastings algorithm. The difference is that one does not compare misfit magnitudes, but the actual Hamiltonian, or energy of the system. Mathematically, this can be expressed as

$$\min\left(1, \frac{\exp\left[-H(\mathbf{q}_\tau, \mathbf{p}_\tau)\right]}{\exp\left[-H(\mathbf{q}, \mathbf{p})\right]}\right). \tag{19}$$

I like this better in words. As we explore model space, we assign new momenta in each iteration. This will result in a different energy of the system. If the Hamiltonian (the system's energy) has decreased from the previous sample, than it is accepted unconditionally (because $\exp(H - H_\tau) > 1$). If, however, energy has increased, it has a chance of $\exp\left(H - H_\tau\right)$ to be accepted. This can be expressed as exploration of energy levels.

What is noteworthy is that through the conservation of energy the Hamiltonian will not change over the course of the trajectory. This allows two things; a quality control to ensure that propagation of the model is performed correctly. Moreover, the chance of accepting a model is completely defined as soon as the momentum is assigned. This is very important, as it now also follows that propagation time doesn't influence the acceptance rate *at all*. As one will see, trajectory length is more a tuning parameter which determines the algorithm's measure of exploration.

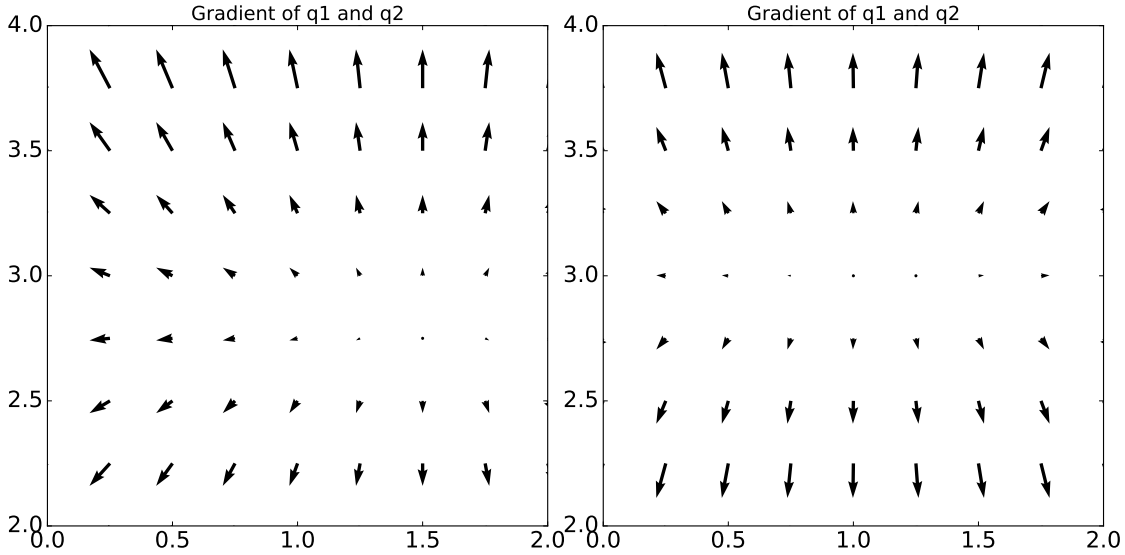
Mass matrices can be chosen in two ways. A standard, non-optimized option would be to choose the unit matrix. A simple analysis given in Andreas Fichtner's reader also reveal that parameters with different forward model derivatives oscillate non-equally

during a trajectory. A mitigation is to assign the mass matrix according to the forward model. The mass matrix that would result in equal oscillations would be

$$\underline{M} = \underline{G}^T \underline{G} \tag{20}$$

where just taking the trace of this matrix would 'unlink' the parameters again and make oscillations roughly equal. We'll see later that this assumption is not fully correct.

Momenta are drawn from the mass matrix diagonal (correlation is assumed to be non-existent). By using the square root of the mass as standard deviation and zero mean, n momenta are drawn corresponding to n parameters.



(a) Gradient with standard deviation $\sigma = 1$ (b) Gradient with standard deviation $\sigma = 5$

Figure 1: Normalized gradient of 2D misfit functionals with differently confined prior information. The means of the prior are for both parameters 2. It is obvious that decreasing the standard deviation for all parameters means that we increase the importance of the prior over the data, 'pulling' the minimal gradient more towards the prior mean. Similar effects can be attained with increasing the magnitude of the data covariance matrix. Together, these matrices finely tune the minimum of the misfit function. Choosing well informed data and parameter covariace matrices is essential to meaningful HMC sampling. In this example, the data covariance was uniformly 0.5.

2. Simple linear systems

2.1. Introduction with a 2 parameter system

In a first analysis a simple linear system is considered, which allows us to get a grip on what is happening behind the scenes. A first analysis will be done on a two dimensional system given by the following linear system;

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}. \tag{21}$$

For our synthetic data I choose $q_1 = 1$ and $q_2 = 3$. Prior information and measurement uncertainty all have an impact on posterior exploration, but for now I will set some arbitrary values. Our prior I synthetically set to have all means of 2 and standard deviations of 1. Measurement errors are chosen to be uncorrelated and having a standard deviation of 0.5. The influence of the two covariance matrices on the misfit functional are given in Figure 1.

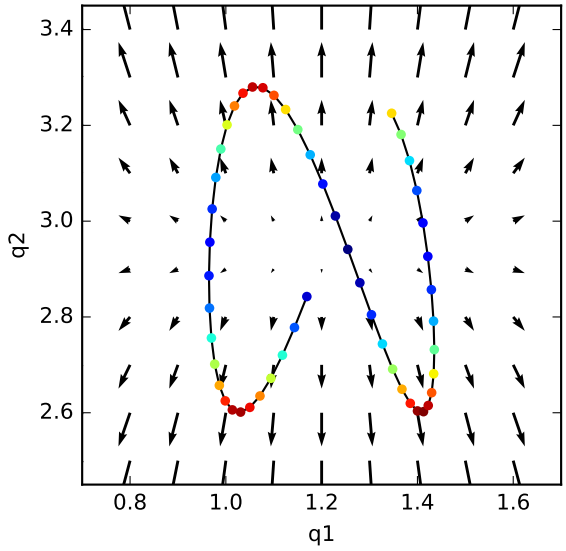


Figure 2: Untuned trajectory of the simple linear system described in Equation (21). The color of the trajectory points represents the normalized misfit, with red being the highest value, and dark blue being the lowest. Note that the direction of the trajectory can not be determined upon inspection of the trajectory itself; Hamilton's equation are invariant under time reversal. A particle would traverse exactly the same trajectory in reverse if at the end it's momentum would be reversed. The trajectory is superimposed on the gradient of the misfit functional χ, which acts as the direction of the largest increase in gravitational potential. A particle starting without momentum would start to roll in opposite direction of these vectors, eventually orbiting the point of zero gradient (without energy loss it can never reach a steady state in the point of minimum energy).

For illustration purposes, 50 time steps were taken with a stepsize of 0.05 in time units (actual units of time might be a bit meaningless, without defining the units of length and mass). Using the unit mass matrix now generates trajectories like the one described in Figure 2.

Mass matrix optimization There are some undesirable characteristics of this trajectory. Due to the mass matrix being a diagonal unit matrix, but the misfit functional being elongated along the dimension of parameter q_1 the oscillations in either dimension do not have the same period. Using a mass matrix based on Equation (20) (which is still diagonal with the current forward model) will equalize oscillations. The reason this behavior is desired is that this way we see as many different energy levels in every dimension. The result of assigning the mass matrix based on the trace of $\underline{G}^T\underline{G}$ is seen

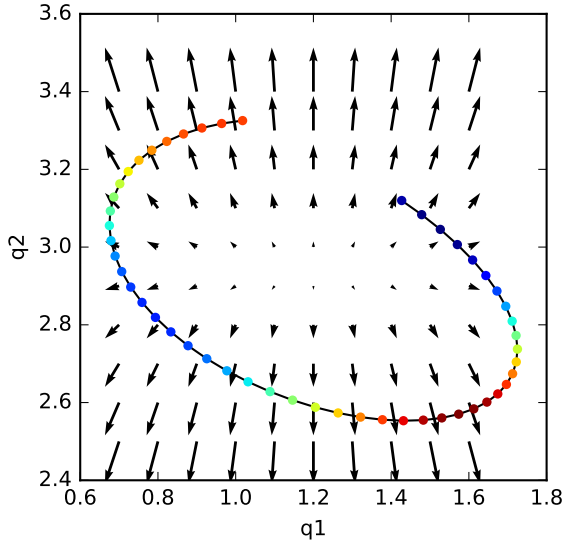


Figure 3: Trajectory tuned with mass matrix, according to the simple linear system described in Equation (21). The color of the trajectory points represents the normalized misfit, with red being the highest value, and dark blue being the lowest. With this augmented mass matrix, oscillations are equal in duration for each dimension.

in Figure 3. The resulting mass matrix is

$$\underline{M} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}. \tag{22}$$

No U-Turn Criterion Visible now in Figure 3 is another characteristic of untuned trajectories. What would be ideal is that the algorithm explores the model space as efficiently as possible. The trajectory depicted in the figure traverses the model space around the minimum fully, but also start to come back to the original position. This so called 'U-Turn' behavior can be mitigated by terminating the trajectory as soon as one detects the propagated model coming closer to the initial model. This point is reached as

$$\mathbf{v}(t) \cdot [\mathbf{m}(t) - \mathbf{m}_0] < 0, \quad \text{and} \quad \mathbf{v}_0 \cdot [\mathbf{m}_0 - \mathbf{m}(t)] < 0. \tag{23}$$

This means that as soon as the momenta vectors for both the beginning and end of the trajectory both make an angle of less than 90 degrees with the vector connecting the points, so if the two models are 'moving towards' each other, the trajectory is terminated. An illustration of a terminated trajectory is given in Figure 4. By limiting the trajectories using this criterion, and additionally increasing the step size so only a few samples are needed to traverse to model space will effectively reduce computational costs of proposing new models.

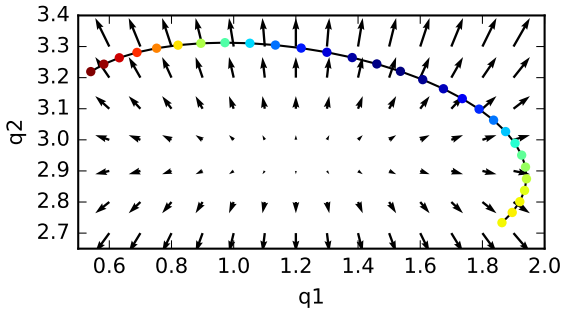


Figure 4: Trajectory tuned with the **No U-Turn Criterion** and forward model based mass matrix. The color of the trajectory points represents the normalized misfit, with red being the highest value, and dark blue being the lowest. As soon as the momentum of the initial point and the last point of the trajectory point 'towards' eachother the trajectory is terminated. In this case, 29 samples were made before the trajectory was terminated.

Number of timesteps	50
Length of timestep	0.05
Predicted stability	0.5

Table 1: Inversion parameters for 'naive' trajectory settings.

Stepsize of the trajectories The step size is another important tuning parameter. Wasting many computation on many steps is wasteful if the 'particle' will end up in the same place, as was done in previous illustrations. According to Neal et al. [2011] the maximum step size is defined by the minimum standard deviation of momentum, or square root of the mass matrix.

Results Using optimized parameters, given in Table 1, multiple inversions were performed using HMC. The used stepsize is well below the upper limit of stability predicted by the mass matrix, according to Neal et al. [2011]. As can be seen in Figure 5, over many iterations the amount of accepted models evens out to approximately 66%. I'll call this method 'naive' for now, for reasons which will become apparent later.

In Figure 6 one sees what we'd expect to see with a linear model with Gaussian uncertainties. After more and more iterations the marginal posterior probability density functions approaches a normal distribution. It's actual mean in the last iteration is $\mu_1 = 2.941$, close to the original parameter. This value is reached to within 1% as soon as 500 iterations. The actual development of the value is given in Figure 7. This all seems like ideal behaviour, our sampler is able to get meaningful statistic from our dataset after only 500 iterations, and after 50,000 iterations we have a very nice looking posterior.

When one however examines not only the marginal posterior of parameter 2, but also the marginal posterior of parameter 1 (Figure 8) and the full two dimensional posterior

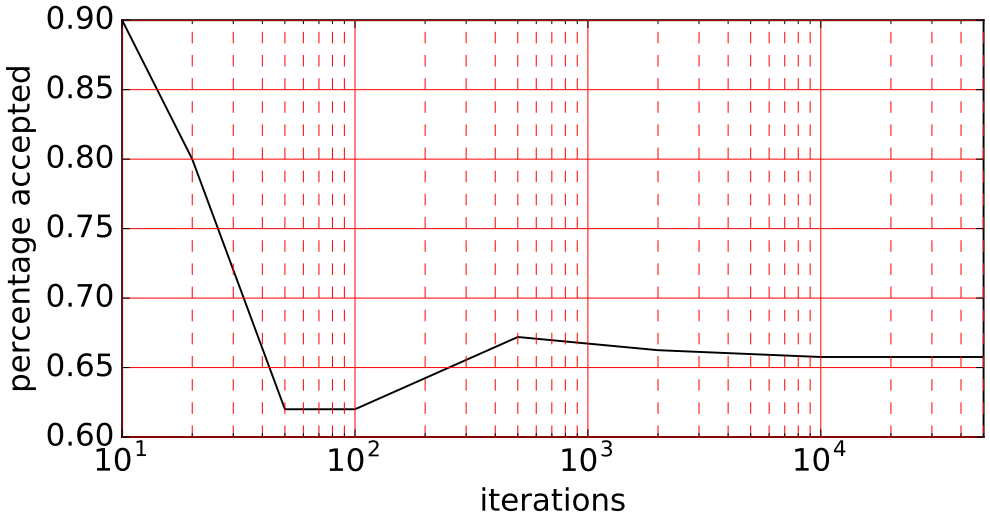


Figure 5: Acceptance rates of the 'naive' sampling.

(Figure 9) unexpected characteristics are found. It seems that parameter 1 has a bimodal distribution, while none of our input prior information nor forward model suggests such behaviour.

Upon closer inspection of the random walk trajectory done after 50,000 iterations (Figure 10), there seems to be some bias in proposing new models. What happened in this case which spefically leads to the bimodal distribution is theorized as follows; the prior information always leads to propose the first model close to $\mathbf{q} = (2, 2)^T$. This starting point coupled with the rather long trajectory length of 50 samples and still rather large stepsize, result in every iteration traversing almost the complete model space.

This, in addition with the No U-Turn Criterion, makes these trajectories 'resonate', always exploring the same portion of the model space. If a model is launched away from the misfit minimum, it wil return towards its original position and then be killed by the No U-Turn Criterion. If it is launched in a relatively beneficial direction. it will traverse a lot of space and turn at the opposite side of the model space. I would not go as far as to call it hysteresis, but rather a fundamental result of the tuning parameters.

There's two ways to test this hypothesis. First, the random walk using a different prior is analysed. Now the prior means are chosen at exactly the values of the parameters used for the synthetics. The result, given in Figure 11, is that sampling is a little bit improved. The sampling is more uniform in the 2D space, but there is still some kind of 'orbiting' behaviour, the sampler never reaches the minimum potential. This is because all trajectories passing through this point are not yet terminated. Making trajectories even longer won't work, because the No U-Turn Criterion will terminate them anyway. This leads us to testing another method of avoiding biased sampling; shorter trajectories.

I think that this is a better solution. By decreasing total trajectory length, one does

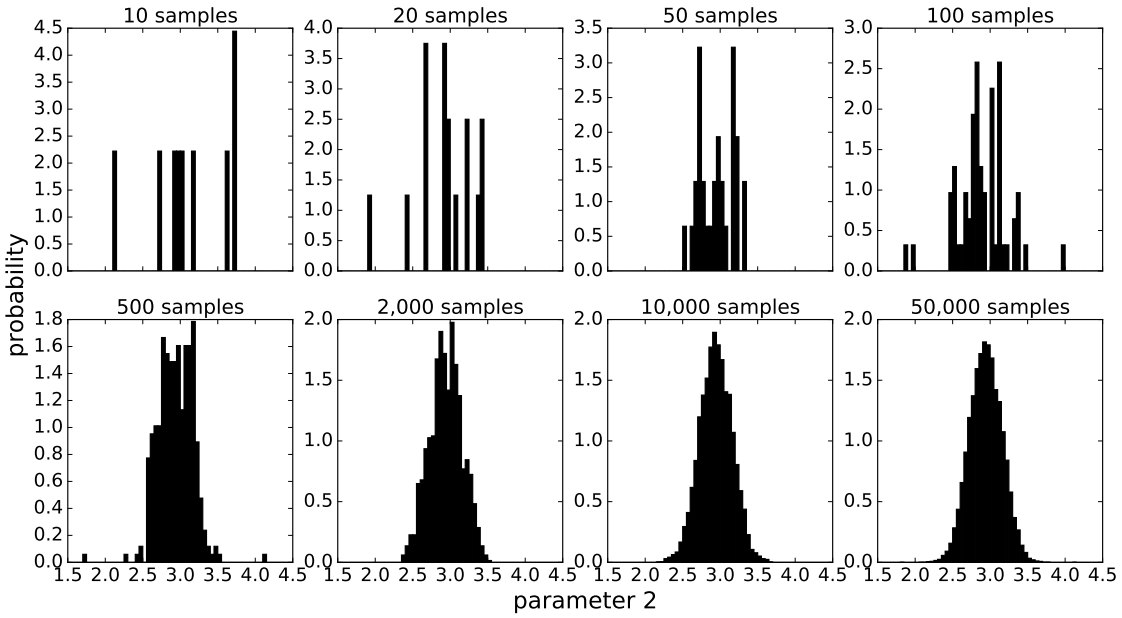


Figure 6: Histogram for marginal probability density of parameter 2 after 'naive' HMC sampling using different amount of samples.

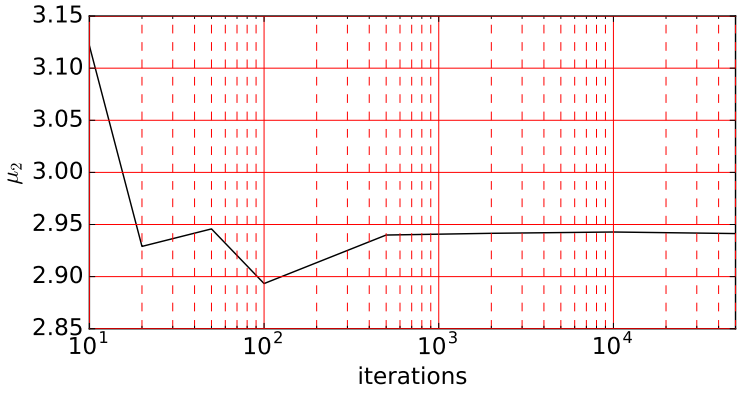


Figure 7: Evolution of mean of parameter 2 as the number of samples increases during the 'naive' sampling.

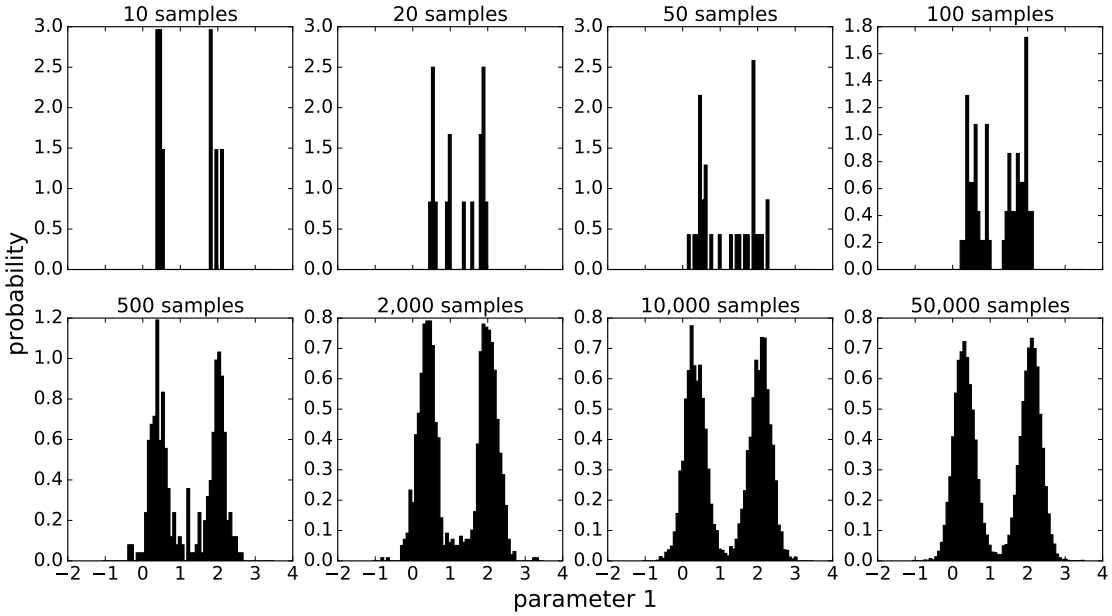


Figure 8: Histogram for marginal probability density of parameter 1 after 'naive' HMC sampling using different amount of samples.

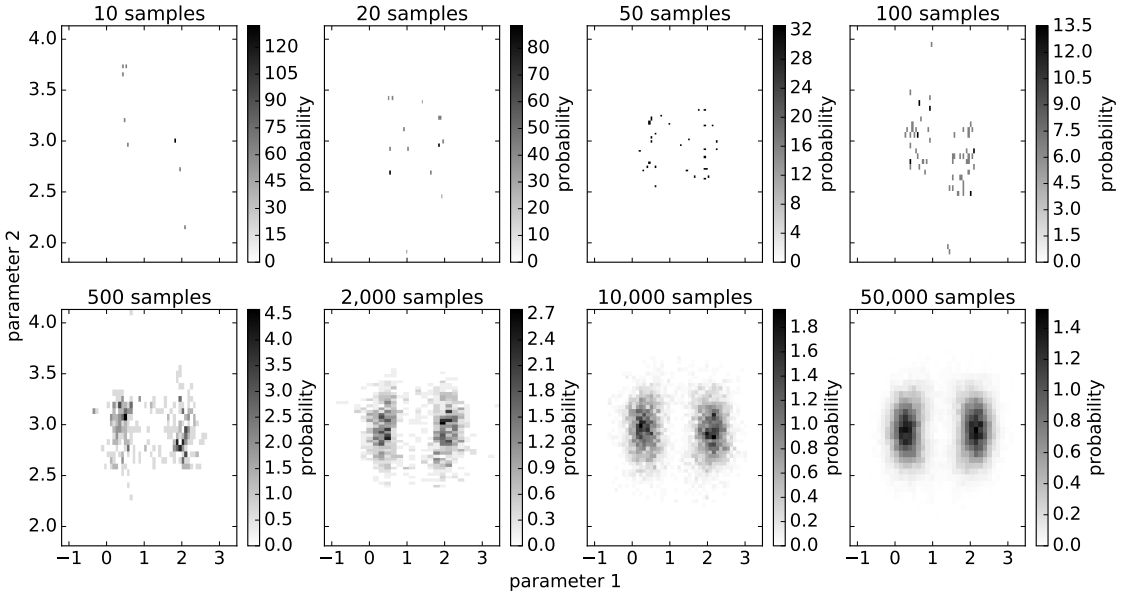


Figure 9: Full 2D marginal probability density after 'naive' HMC sampling using different amount of samples.

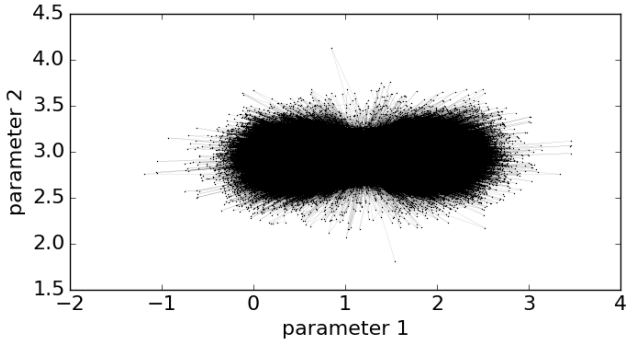


Figure 10: Random walk during 50,000 'naive' HMC samples.

not need to tune actual physical intuition or information (the prior) to the needs of the algorithm. Recognizing this behaviour in non-linear or high dimensional systems might prove very challenging upon inspection of marginal probability distributions. One quality check one could do, even during sampling, is to monitor how often the No U-Turn Criterion triggers the termination of a trajectory, relative to total models ran. The promising result is given in Figure 12.

2.2. Performance of higher dimensional systems

In this section I want to look at the influence of dimensionality; how it affects computation time and acceptance rates. A general forward model is presented for n-dimensions;

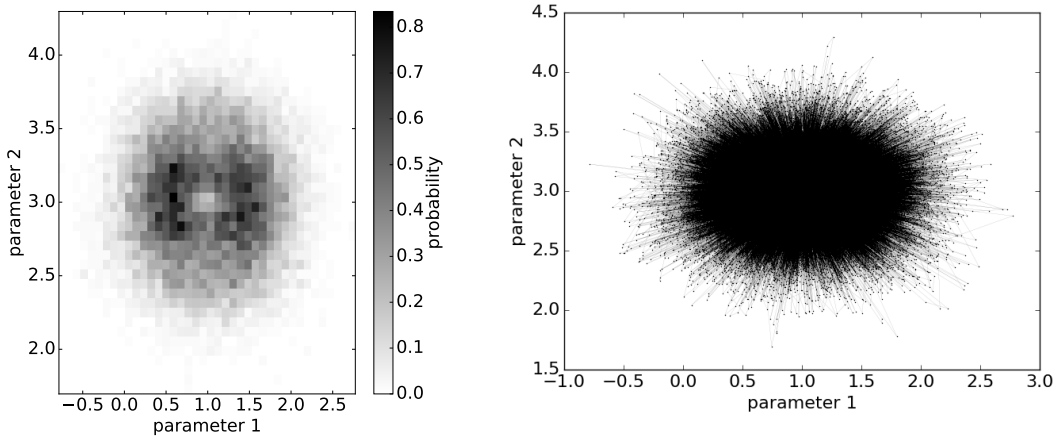
$$\underline{G} = \begin{bmatrix} 1 \cdot 1 & & & \\ & 2 \cdot 2 & & \\ & & \ddots & \\ & & & n \cdot n \end{bmatrix} \tag{24}$$

A note on the mass matrix The mass matrix was initially again based upon the trace of $\underline{G}^T\underline{G}$. This **did not** produce equally oscillating trajectories with models larger than 2 dimension and varying forward models. After some trial and error, and revisiting the theory, an extension upon the mass matrix is made. In the aforementioned reader, the covariance matrices were not analyzed in the general solution. By substituting the 'new' $\underline{A}$ and $\mathbf{b}$ from equations (9) and (10), we see that a more optimized mass matrix could be constructed from (the trace of) $\underline{A}$ like so;

$$\underline{M} = \underline{C}_M^{-1} + \underline{G}^T\underline{C}_D^{-1}\underline{G}. \tag{25}$$

This deficit was not apparent in the analysis of the 2 dimensional system probably due to the small difference in forward matrix components and parameters.

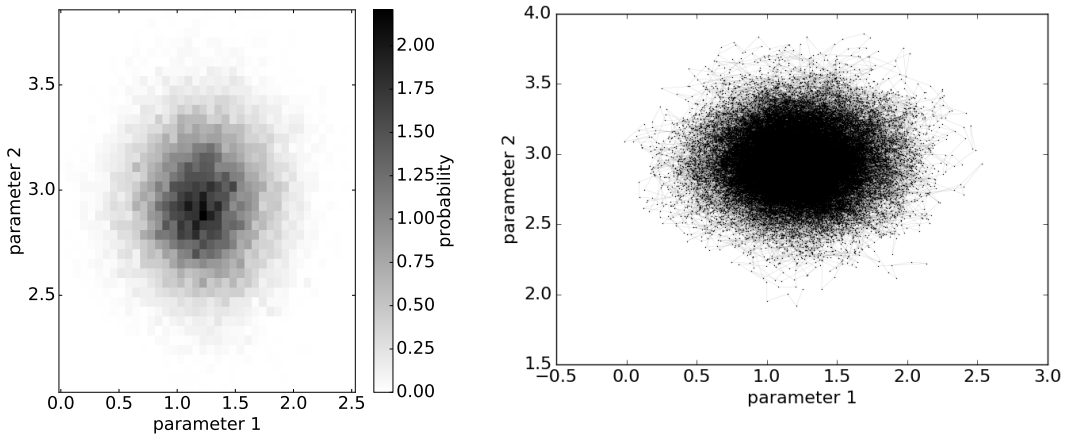
One important note is that this matrix is required to be positive definite to allow us to sample the full model space. If the $n \times n$ matrix $\underline{M}$ is not of full rank, the column



(a) Full 2D posterior

(b) Random walk sequence

Figure 11: 50,000 samples of the same 2D posterior as Figure 9, tuned with a different starting prior. Note that the random walk still traverse most of the model space in each proposal, leading again to 'orbiting' of the minimum.



(a) Full 2D posterior

(b) Random walk sequence

Figure 12: 50,000 samples of the same 2D posterior as Figure 9, tuned with a much smaller trajectory length. Note that now each new model only has a small distance from the previous, but that the model space is much more evenly sampled.

space is not completely $\mathbb{R}^n$, and one does not explore the 'interesting' part of our model space **I want to refine this wording, be a bit more specific**. Now, one does not generally now much about the properties $\underline{G}^T$ if the system is very large. We'll also see later that if the mass matrix is not positive definite, the suggested code won't run since the required matrix decomposition does not exist.

Usually, we can see that a system is under-determined or mixed-determined, but it is not easily verifiable if the system is over-determined (i.e. full rank). In this sense, the prior information in parameter space acts as a stabilizer.

Positive definiteness is defined by;

$$\mathbf{x}^T \underline{A}\, \mathbf{x} > 0, \quad \forall\, \mathbf{x} \neq \mathbf{0} \tag{26}$$

We first assume that our prior covariance for both the data and parameters are positive definite. I think for data covariances this describes independence of measured values, and the same goes for parameter covariances. Positive definiteness ensures that matrices can be inverted, and that their inverse is also positive definite.

Now, we can easily show that the second term of the right hand side of Equation (25) is at least positive semi-definite;

$$\mathbf{x}^T \underline{G}^T \underline{C}_D^{-1} \underline{G}\mathbf{x} = \ (\underline{G}\mathbf{x})^T \underline{C}_D^{-1} \underline{G}\mathbf{x} \geq 0, \quad \forall\, \mathbf{x} \neq \mathbf{0} \tag{27}$$

The reason this semi-definiteness creeps in is that one can't be sure that $\underline{G}$ doesn't map $\mathbf{x}$ to $\mathbf{0}$, since we don't know the dimension of it's null space. This is directly related to a system being under- or mixed determined, as $\text{rank}(\underline{G}) + \text{nullity}(\underline{G}) = n$. If a system is under-determined, the second term is positive semi-definite. If one knows that $\underline{G}$ is of full rank, then the resulting second term is positive definite.

This is also where our prior knowledge comes in, and acts as a stabilizer when the data prior is not sufficient for an inversion. If given two matrices for which hold $\mathbf{x}^T \underline{A}\, \mathbf{x} > 0, \quad \forall\, \mathbf{x} \neq \mathbf{0}$ and $\mathbf{x}^T \underline{B}\, \mathbf{x} \geq 0, \quad \forall\, \mathbf{x} \neq \mathbf{0}$ then the following holds:

$$\mathbf{x}^T \left(\underline{A} + \underline{B}\right) \mathbf{x} = \ \mathbf{x}^T \underline{A}\mathbf{x} + \mathbf{x}^T \underline{B}\mathbf{x} > 0, \quad \forall\, \mathbf{x} \neq \mathbf{0}, \tag{28}$$

thus, proving that the mass matrix is always positive definite if C_M^{-1} and C_D^{-1} are too.

Performance increase At this stage I also realized that there are two main points on which we can increase performance. The code at this stage was sped up threefold by compiling using any of the -O# g++ flags. Any specific compiler flag (-O1 through -O4) didn't noticeably alter performance, mostly invisible due to the random nature of the sampling. I advise therefore to use -O2, the most tested mode, which alters your code the least. Also, RAM usage can be easily decreased by limiting the amount of std::vector<> copies. Every time one of those objects is passed and used only once, (near) perfect forwarding can be achieved using std::move().

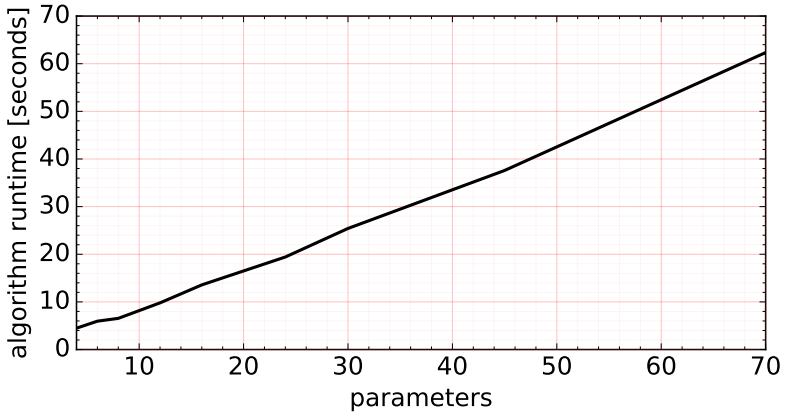


Figure 13: Computation time of 50,000 samples for a varying amount of parameters.

Inversion characteristics The step size of any model is fixed at 0.05, since the first dimension provides the upper bound to the stability. For the trajectory, 10 different iterations are done, or whenever the U-Turn criterion is met. In total, 50,000 samples are drawn. Prior information for every parameter is set at 10, with a standard deviation of 5. This means that as the dimensionality increases, the prior becomes worse with respect to the actual parameter. Data covariance is fixed at 10 percent of the observed value, as to negate it's influence on the algorithm as much as possible.

Results The computation time versus number of parameters exhibits an almost linear relationship, as seen in Figure 13. The acceptance rate seems to suffer from the increasing dimensions, appearing inversely proportional to the amount of parameters (Figure 14). The results are rather good, but decreasing in quality as the number of accepted models decreases. This performance might be improved by actually drawing correlated samples from the mass matrix (I advise Cholesky decomposition).

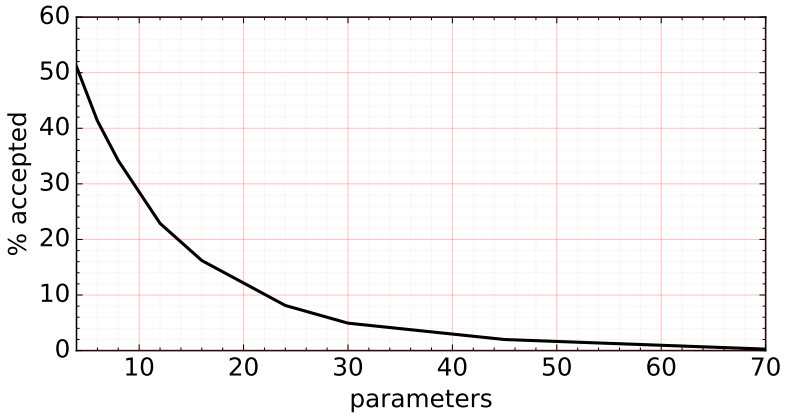
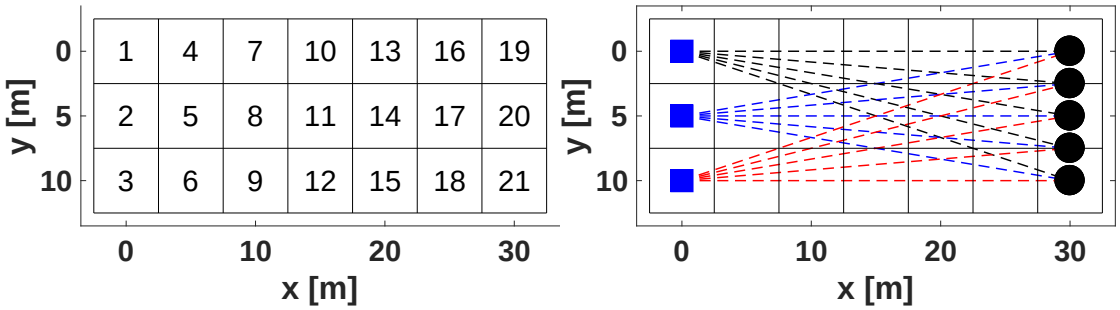


Figure 14: Acceptance rate of 50,000 samples for a varying amount of parameters.



(a) Parameters of model

(b) Ray coverage of model

Figure 15: Linear tomography model used in the first part of Section 3, with 21 dimensions. The ray coverage is variable per cell, and can be better analysed in Figure 16

3. Linear tomography

In this section the HMC sampler is applied to a linear two dimensional tomography problem. These kind of problems might be high dimensional, and I will eveluate two different sizes of models with varying velocity models. With the aid of model and data resolution matrices we are able to link the quality of the inversions to the actual models and measurement configurations.

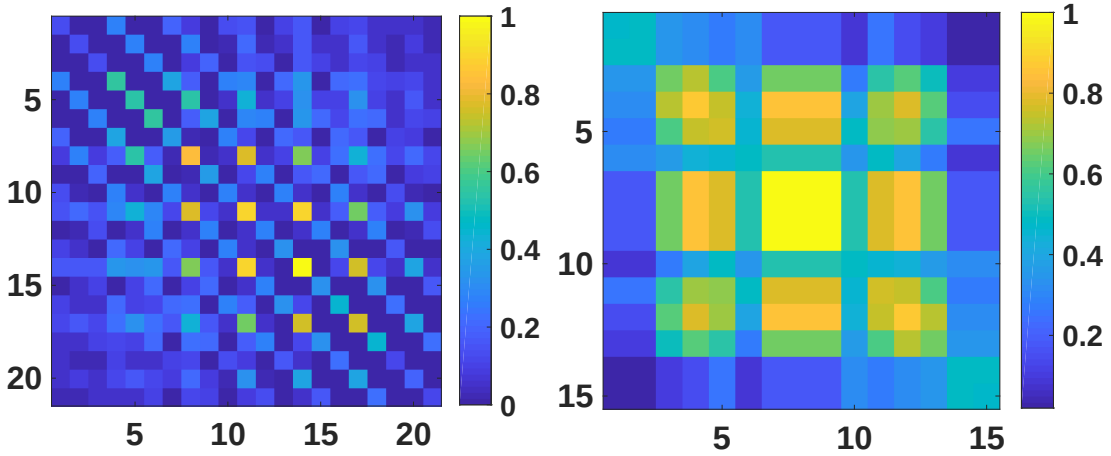
3.1. Straight ray tomography and a first model

In this part, extensive use is made of the MATLAB package written by Naiara Korta Martiartu and Christian Böhm (Computational Seismology Group (CSE), ETH Zürich). I will leave out the details of this algorithm, as it is perfectly explained in the accompanying manual. The important bit is that the algorithm produces the linear system matrix and synthetic data.

Straight ray tomography is especially suited to test HMC for it's forward model is completely linear, and it usually encompasses some interesting correlated parameters and mixed-determined systems.

We'll start by looking at a relatively simple model of 21 dimensions. The region of interest is composed out of 7x3 blocks of 5 meters wide, as depicted in Figure 15a. In this region, 3 sources and 5 receivers are present, set up as in Figure 15b. This results in $3 \cdot 5 = 15$ data points. The resulting forward matrix $\underline{G}$ is therefore of size 15x21. Each block in the model has rays passing through it in a different way, and may be differently resolved then the next block.

Note, I actually started on this thinking these were resolution matrices, but I read Hansruedi's script wrong. Still, my analysis stands and I wonder what these matrices actually represent. To visualize this, we employ the parameter (again, not really the) resolution matrix, for equal data covariance given by $\underline{G}^T\underline{G}$ (which, interestingly enough, is equal to the original proposal of the mass matrix in HMC). In



(a) Parameter resolution matrix

(b) Data resolution matrix

Figure 16: Normalized 'half' resolution matrices for the model depicted in Figure 15.

this matrix, the diagonal elements define how well resolved each parameter is relatively, while the rows (as well as the columns, the matrix is symmetric) define averaging kernels.

As seen in Figure 16a, some parameters are properly resolved, while others are not. Well resolved parameters have higher indices (note that the matrix is normalized, so we can't say anything about absolutely quality). Some parameters also have high off-diagonal elements, which indicates correlation between parameters. This means that if one of these parameters is high, it influences our perception of the other parameter.

Figure 16b shows the data resolution matrix, given by $\underline{G}\,\underline{R}\,\underline{G}^T$ for equal data covariance. This matrix helps us identify which data is relatively important in the resolution of parameters, i.e. which have a large impact on the inversion. For example, data points 7 through 9 have the highest data importance; they relate the parameter well to the observed value.

The actual resolution matrices for linear systems are identity matrices, providing not so much insight.

Velocity model and inversions The velocity model used for the synthetic data is a simple homogeneous model with acoustic speed constant at $20000 m/s$. Inversion characteristics are given in Table 2. The parameter which is inverted for is slowness, and the prior means is taking as 1.0/1500.0, with a standard deviation of 0.00025 seconds per meter. A consequence of the system being underdetermined is that less resolved parameters will diverge from the real model in the sense that some will tend towards the prior information, which changes correlated parameters to higher values. The means after inversion are given in Figure 17 and show this characteristic quite strongly.

Influence of data dimension on performance Before we saw the influence of the amount of parameters on the inversion performance. Another possible detrimental fac-

Number of Samples	1,000,000
Number of timesteps	10
Length of timestep	0.1
Runtime	118.324 s
Accepted samples	108215
Number of U-Turns	0

Table 2: Inversion parameters for first tomography inversions.

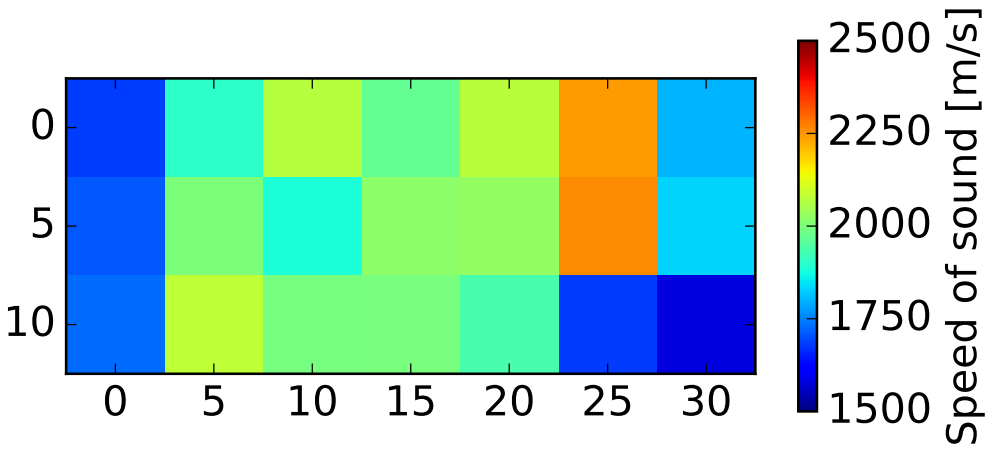


Figure 17: Means after the initial inversion of approximately 100,000 accepted samples. The parameters at the very horizontal ends of the model are relatively low, probably due to the small entries in the pseudo-resolution matrix. These parameters are not influenced by actual data that much and therefor are pulled more towards the prior informations. At the same time, to compensate for relatively low speeds, parameters in other parts are excessively high. The actual speed in most elements (columns 5 through 20) is however close to the actual forward model of 2000 m/s.

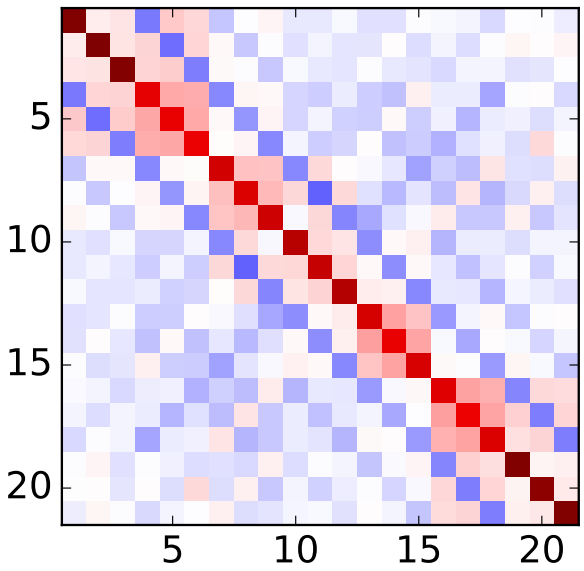


Figure 18: Parameter covariance of the approximately 100,000 inversion samples. The band of negative correlation of entries σ_i, j and $\sigma_{i,j\pm3}$ or $\sigma_{i\pm3,j}$ is generated by blocks that lie next to each other; as one speed goes up, the next and previous horizontal blocks compensate by lowering their speed. This behaviour is upon close inspection also visible for the ±6 bands. Entries in the same model column also experience a kind of grouping, but with positive correlation. The maximum covariance is $3.6 \cdot 10^{-8}$.

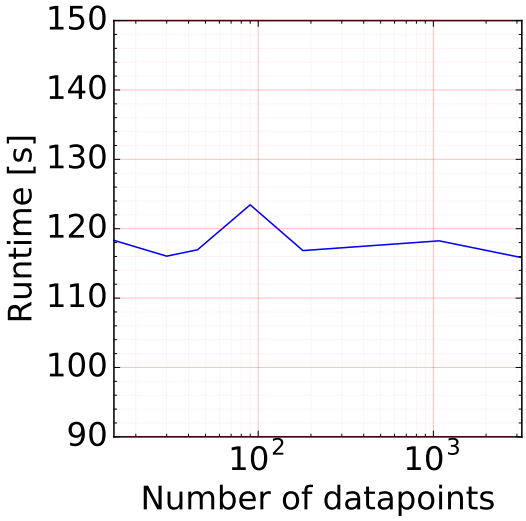


Figure 19: Algorithm runtime for increasing amount of datapoints. There seems to be no impact on the algorithm performance with an increasing amount of datapoints.

tor might be the amount of data, which also alters some computational requirements. The predicted influence is not so big, as in the model propagation the dimension is not increase. The additional computational cost will be mostly present in evaluating misfits and their gradients, but as these numbers are definitely precomputed the influence might be weak.

To test this, we change the amount of receivers in the previously used homogeneous model. Specifically, we add 5 receivers centered in the model blocks starting from the column at $x = 25$, working our way to the column $x = 5$. This will add 15 data points each step, increasing all the way to 90. Other inversion characteristics remain unchanged.

After each data increase, the model parameters will be better retrieved. Only shown is the last covariance matrix, with 3240 data points. It has relatively less large entries with respect to the largest one. Also the maximum covariance ($2.2 \cdot 10^{-9}$) is lower than the original ($3.6 \cdot 10^{-8}$) for the inversion with 15 data points. The actual runtime is almost not influenced by the amount of data points (number of rows of the forward model).

3.2. Generalized momentum

At this point I wondered about the influence of implementing the generalized momentum. Implementing this however requires the development of some additional theory. To calculate the generalized momentum we need to invert the mass matrix, which is already an interesting problem. Additionally, we'd like to propose momenta that are correlated according to the mass matrix, so we need to sample from a multivariate correlated Gaussian. I'll start with the second problem, as the result of that simplifies our first

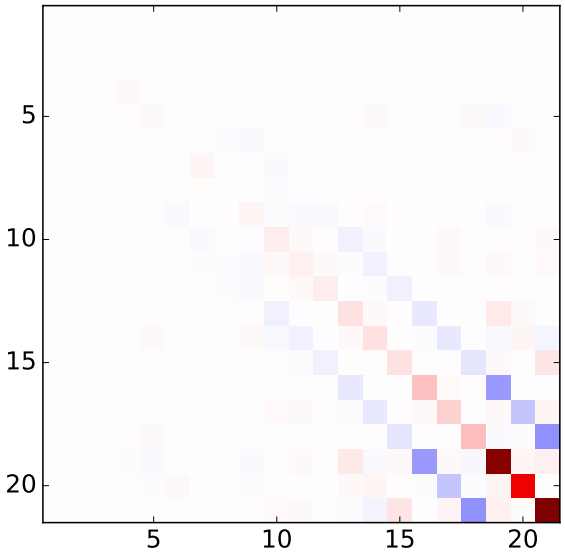


Figure 20: Normalized covariance matrix of the last inversion from Figure 19 with 3240 datapoints. Maximum parameter covariance is $2.2 \cdot 10^{-9}$.

problem as we'll see later.

Sampling from an n-dimensional Gaussian Up to now, we've proposed new momenta by regarding the diagonal of the mass matrix as the variance of each mass. By taking the square root of the inverse of the diagonal we obtained standard deviations of each mass, and by setting the mean to zero we drew new momenta using the Box-Müller transform.

An extension of this mass matrix interpretation is to regard the off diagonal elements as the covariance between parameters. This yields for an $n \times n$ matrix an n-dimensional correlated Gaussian.

First we establish the definition of an affine transformation and it's effect on a normal distribution. The distribution $\mathbf{X} \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ represent a normal distribution of means $[\boldsymbol{\mu}]_i = \mu_i$ and covariances $[\boldsymbol{\Sigma}]_{ij} = \sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$ (covariance σ_{ij}, standard deviation σ_i and correlation ρ_{ij}). Now, if $\mathbf{Y} = \boldsymbol{L}\mathbf{X} + \boldsymbol{c}$ is an affine transformation (that is to say, the transform acts linearly on a vector, which it does for matrix-vector multiplication) then the resulting distribution $\mathbf{Y}$ has means $\boldsymbol{c} + \boldsymbol{L}\boldsymbol{\mu}$ and variance $\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T$ (the proof is given in Appendix A).

If we now sample $\mathbf{X} = \mathcal{N}_n(\mathbf{0}, \mathbf{I})$ and transform it using $\boldsymbol{L}\mathbf{X}$ such that $\boldsymbol{L}\boldsymbol{I}\boldsymbol{L}^T = \boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{\Sigma}$ we have effectively sampled $\mathbf{Y} = \mathcal{N}_n(\mathbf{0}, \boldsymbol{\Sigma})$. Changing the means is as easy as adding the mean vector.

Equipped with this we can now draw from a correlated Gaussian in the following way;

- Draw n uncorrelated samples with mean $\mu_i = 0$ and standard deviation $\sigma_i = 1$;
- Multiple this samples vector with a matrix M $(n \times n)$ which transforms $\mathbf{I}$ to $\mathbf{\Sigma}$ during the affine transform;
- Add the wanted mean.

The second step, finding an appropriate matrix, is the most complicated part of the process. Luckily, we have proven already that our mass matrix is positive definite. These types of matrices, given that they are self adjoint/Hermitian, are representable by a Cholesky decomposition. The Hermitian property is easily validated; covariances are symmetric and real. The Cholesky decomposition produces a lower triangular matrix $\mathbf{L}$ from a positive definite Hermitian matrix $\mathbf{A}$ with the property:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^{\dagger}. \tag{29}$$

Here, $\mathbf{L}^{\dagger}$ stands for the conjugate transpose of $\mathbf{L}$. For the real matrices we are working with, this simplifies to transposition only. Finding the appropriate affine transform is now reduced to finding the appropriate Cholesky decomposition of $\mathbf{\Sigma}$. To achieve this, the Cholesky-Banachiewicz algorithm is implemented.

Evaluating the generalized momentum The generalized kinetic energy, extended from Equation 15 according to Neal et al. [2011], is given as

$$K(\mathbf{p}) = \frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}. \tag{30}$$

The difficulty in evaluating this expression is in calculating the matrix inverse of the mass matrix, which for large systems would result in quite some processing time. But since one already has acquired the Cholesky decomposition we can easily show that:

$$\mathbf{A}^{-1} = \left(\mathbf{L}\mathbf{L}^T\right)^{-1} = \left(\mathbf{L}^T\right)^{-1}\left(\mathbf{L}\right)^{-1}. \tag{31}$$

Now, we're still left with a matrix inverse, but inverting a lower triangular matrix is easily done by forward and back substitution algorithms [1] , obtaining every element of a column row by row when inverting $\mathbf{L}\mathbf{L}^{-1} = \mathbf{I}$. Following that we simply use $(\mathbf{L}^T)^{-1} = (\mathbf{L}^{-1})^T$ and multiply the two inverse matrices to obtain $\mathbf{A}^{-1}$.

[1] https://en.wikipedia.org/wiki/Triangular_matrix#Forward_and_back_substitution

References

R. M. Neal et al. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

Acronyms

HMC Hamiltonian Monte Carlo. 1–5, 7, 10, 14, 19

Index

Appendix A Affine transformations of normal distributions

Proving the relationship's of the means and covariances between affine transformations is easily done from the definition of linear transformations and covariances.

For the means $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ of respectively the normal distributions $\mathbf{X}$ and $\mathbf{Y}$ we use the definition of the transformation of $\mathbf{X}$ to $\mathbf{Y}$. $E\{a\}$ is the expected value of a.

$$\begin{aligned}\overline{\mathbf{y}} &= E\{\mathbf{y}\} \\ &= E\{\mathbf{L}\mathbf{x} + \boldsymbol{c}\} \\ &= \mathbf{L}E\{\mathbf{x}\} + \boldsymbol{c} \\ &= \mathbf{L}\overline{\mathbf{x}} + \boldsymbol{c}\end{aligned} \tag{32}$$

□

For the covariances we use it's definition:

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{y}} &\triangleq E\left\{(\mathbf{y} - \overline{\mathbf{y}})\,(\mathbf{y} - \overline{\mathbf{y}})^T\right\} \\ &= E\left\{(\mathbf{L}\mathbf{x} + \boldsymbol{c} - \mathbf{L}\overline{\mathbf{x}} + \boldsymbol{c})\,(\mathbf{L}\mathbf{x} + \boldsymbol{c} - \mathbf{L}\overline{\mathbf{x}} + \boldsymbol{c})^T\right\} \\ &= E\left\{(\mathbf{L}\mathbf{x} - \mathbf{L}\overline{\mathbf{x}})\,(\mathbf{L}\mathbf{x} - \mathbf{L}\overline{\mathbf{x}})^T\right\} \\ &= E\left\{\mathbf{L}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^T\mathbf{L}^T\right\} \\ &= \mathbf{L}\,E\left\{(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^T\right\}\mathbf{L}^T \\ &= \mathbf{L}\,\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{L}^T\end{aligned} \tag{33}$$

□