Notes on the Trans-Dimensional Metropolis-Hastings (MH) Algorithm

Bill Menke, March-April 2018

0. My Agenda. The goal of the trans-dimensional Metropolis-Hastings (MH) algorithm is to generate a large number of realizations of a trans-dimensional probability distribution $P(X)$, where X is a state variable. My goal is to work through simples examples of trans-dimensional MH, in order to identify issues needing further research.

1. The MH test parameter. For current state X and proposed successor state X', the MR test parameter α is typically stated:

$$\alpha = \frac{P(X'|\mathbf{d}^{\text{obs}})}{P(X|\mathbf{d}^{\text{obs}})} \frac{Q(X|X')}{Q(X'|X)}$$

where $P(x|\mathbf{d}^{\text{obs}})$ is the target probability function for observed data $\mathbf{d}^{\text{obs}}$ and $Q(X'|X)$ is the probability function for the proposed successor state X'given current state X. Suppose now that the current state X has N dimensions and the successor state X' has N' simensions. The equivalent statement for probability density functions of the form $P(X) \equiv p(X)d^N x$ is:

$$\alpha = \frac{p\left(X'|\mathbf{d}^{\text{obs}}\right)d^{N'}x'}{p(X|\mathbf{d}^{\text{obs}})d^N x} \frac{q(X|X')d^N x}{q(X'|X)d^{N'}x'} = \frac{p(X'|\mathbf{d}^{\text{obs}})}{p(X|\mathbf{d}^{\text{obs}})} \frac{q(X|X')}{q(X'|X)}$$

Here x is a individual state variable in the N-dimensional space of X. The probability functions are just replaced by the analogous probability density functions. However, the cancellation of volume elements is between two $(p(.),\ q(.|.))$ pairs and not between one pair of $p(.)$'s and one pair of $q(.|.)$'s. Thus, care must be taken to ensure that the $p(.)$ and $q(.|.)$ in a pair have the same dimension.

Note that if we used Bayes' rule to write:

$$p\left(X, \mathbf{d}^{\text{obs}}\right) d^N x\, d^L d = p\left(X|\mathbf{d}^{\text{obs}}\right) p\left(\mathbf{d}^{\text{obs}}\right) d^L d\, d^N x \ = p_E\left(\mathbf{d}^{\text{obs}}|X\right) p_A(X) d^N x\, d^L d$$

then:

$$p\left(X|\mathbf{d}^{\text{obs}}\right) d^N x \ = \frac{p_E\left(\mathbf{d}^{\text{obs}}|X\right) p_A(X) d^N x}{p(\mathbf{d}^{\text{obs}})}$$

and:

$$\alpha = \frac{p_E\left(\mathbf{d}^{\text{obs}}|X\right) p_A(X) d^{N'}x'}{p_E(\mathbf{d}^{\text{obs}}|X)\, p_A(X) d^N x} \frac{q(X|X')d^N x}{q(X'|X)d^{N'}x'} = \frac{p_E\left(\mathbf{d}^{\text{obs}}|X\right) p_A(X)}{p_E(\mathbf{d}^{\text{obs}}|X')\, p_A(X')} \frac{q(X|X')}{q(X'|X)'}$$

I note that, in the literate, one usually sees a factor of the Jacobian J (with $d^{N'}x' = J\, d^N x$) added to the definition of a, above. I do not understand why it is needed; all the volume elements appear to cancel out.

2. Use of Discrete Variables. In the test programs, I use a state described by a finite number of discrete (integer) parameters, and probability functions $P(X)$ and $Q(X|X')$ that describe actual probabilities of these states. Thus, $P(X) = 0.01$ means that state X occurs 1% of the time. I don't believe that a discrete approximation of a continuous variable poses any special problem, since our ability to resolve the value of variables like layer thickness and slowness is so limited by data quality.

3. Effect of Changing Dimension of Ratio of Q's. Suppose that $P(X'|\mathbf{d}^{\text{obs}})/P(X|\mathbf{d}^{\text{obs}}) = 0.01/0.02$; then all other things being equal, X' would be accepted 50% of the time. Now suppose that X is in a space that has one less dimension than the space of X' (that is $N' = N + 1$) and that the probability of the successor staying in the N-dimensional space is about the same as moving to N'-dimensional (because they are adjacent dimensions). If the Q's are Normal, the number of high-probability elements in the N-dimensional space scales as M^Nelements, and the number in the N'-dimensional space, as $M^{N'}$elements. Thus, $Q(X|X')/Q(X'|X) \sim M$. The probability of acceptance increases by a factor of M. Thus, a jump to a higher dimensional space tends to accepted, unless the decrease in probability is exceedingly low.

4. MH use of Q. The MH algorithm requires that both $Q(X|X')$ and $Q(X'|X)$ be evaluated for arbitrary values of X and X', and furthermore, that a proposed successor state X' be realized from $Q(X'|X)$ for an arbitrary current state X. These two requirements pose practical challenges, which I address in the following way:

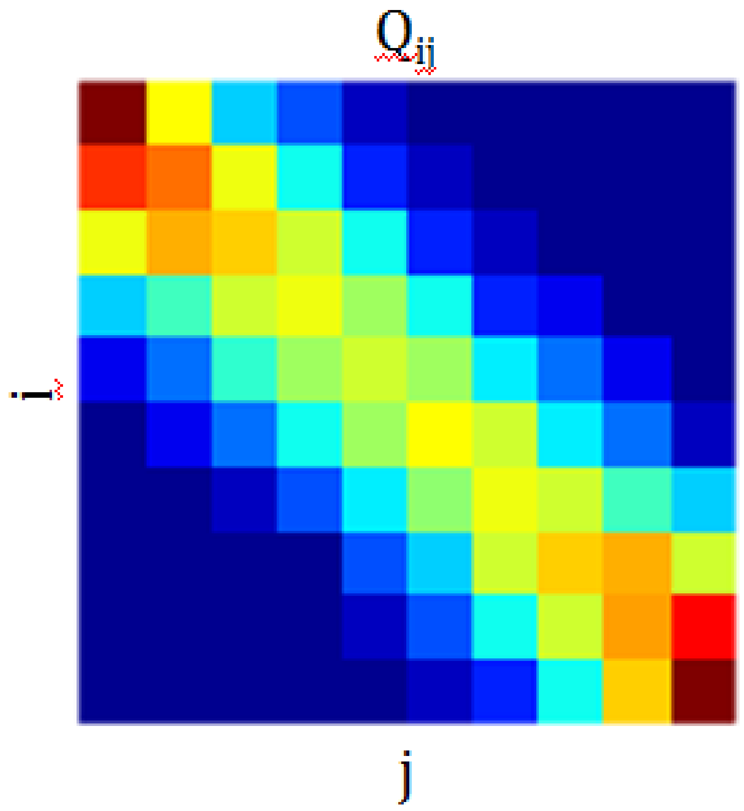
$Q(X|X')$ is built up by assuming that individual state variables x are uncorrelated:

$$Q(X|X') = \prod_x Q(x|x')$$

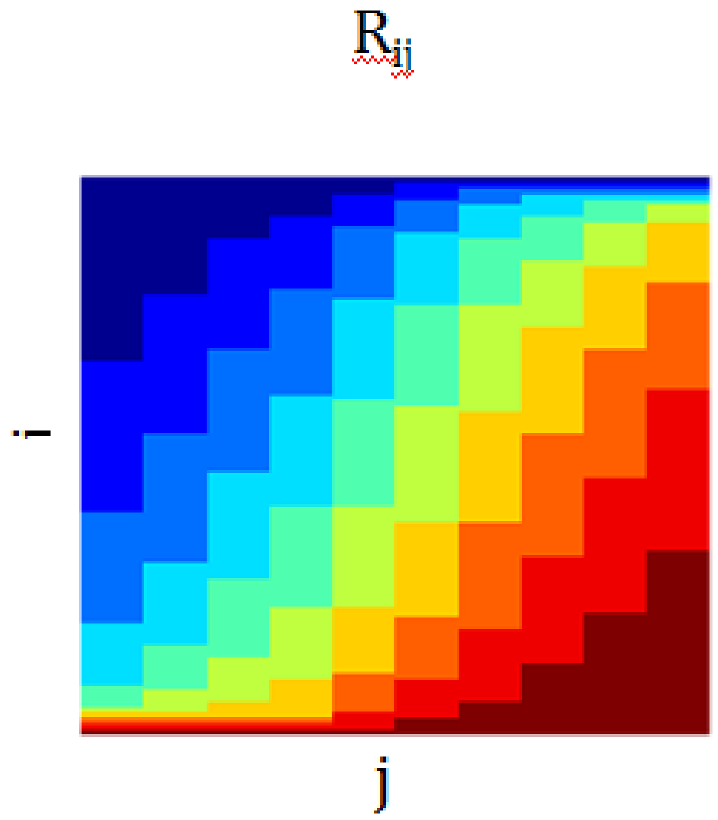
Since both x and x' are a finite number discrete variables, they can be represented by a set of integers in the $1 - K$ range. The function $Q(x|x')$ is therefore a $K \times K$ table $Q_{ij} \equiv Q(x_i|x_j')$ of real values. No element of Q_{ij} may be less than unity, else the MH test parameter α may be singular.

Since MH works for a wide range of $Q(x|x')$'s, Q_{ij} can be merely an approximation to $Q(x_i|x_j')$ Furthermore, the approximation can be chosen in a way that facilitates the processes of creating realizations of $Q(x|x')$. Suppose that $Q(x|x')$, with fixed x', should approximate a known function $f_j(x)$. We choose an approximation that satisfied three conditions: $Q_{ij} \approx f_j(x_i)$; for an specified integer $L \gg K$, every element of LQ_{ij} is an integer; and $\sum_i Q_{ij} = L$. This approximation can be made arbitrary accurate by choosing a sufficiently large L.

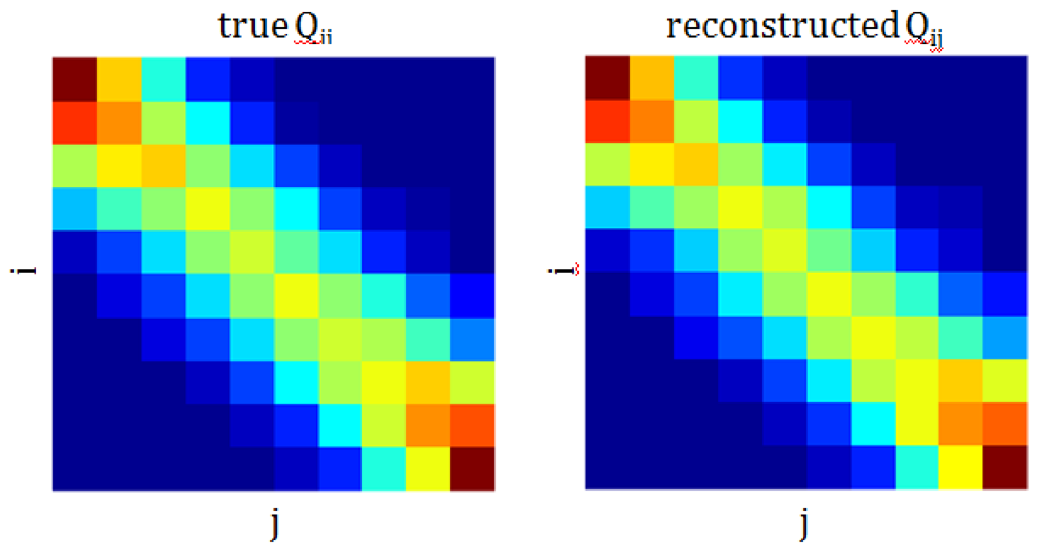
A Q_{ij} that satisfies these three conditions is constructed in the following way: First, set $L \approx 10K$; Second, set $A_{ij} \approx f_j(x_i)$ and define $S = \sum_i A_{ij}$; Third, set $B_{ij} = \text{ceil}\left(LA_{ij}/S\right)$; Fourth, if any B_{ij} is zero, reset it to unity; Fifth, if $\sum_i B_{ij} \neq L$ add/subtract unity from randomly chosen entries until $\sum_i B_{ij} = L$ (but never decreasingB_{ij} below unity); and Sixth, set $Q_{ij} = B_{ij}/\sum_i B_{ij}$. An example of a 10×10 approximately Normal Q_{ij} with a mean of about j and a standard deviation of about $K/4$ is:



A $L \times K$ companion table R_{ij} facilitates the generation of realizations. This table is defined so that each column j has exactly LQ_{1j} occurences of the value 1, exactly LQ_{2j} occurences of 2, etc., with the order of the occurrences being arbitrary. Then, if ℓ is a random integer in the range $1 - L$, $R_{\ell j}$ is approximately a realization of $Q(x|x_j')$. An example of a 100×10 table R_{ij} corresponding to the $Q(x|x')$ above is:



The Q_{ij} reconstructed by binning a set of 10^5 realizations matches the true Q_{ij} very well:



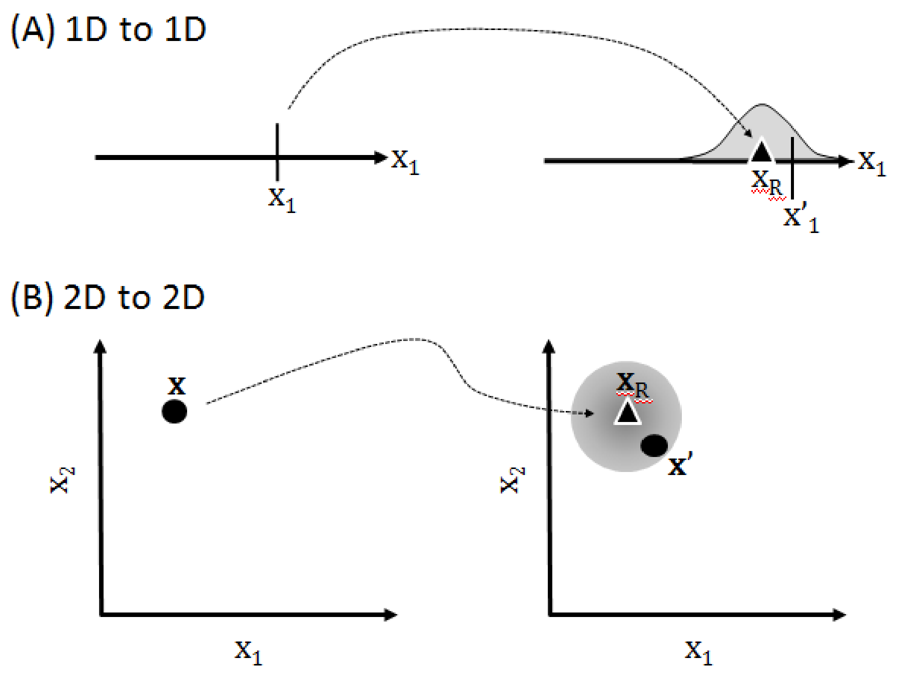
5. Generating a Proposed Successor State. I view the process of generating a proposed successor state X' from a current state X as consisting of three steps:

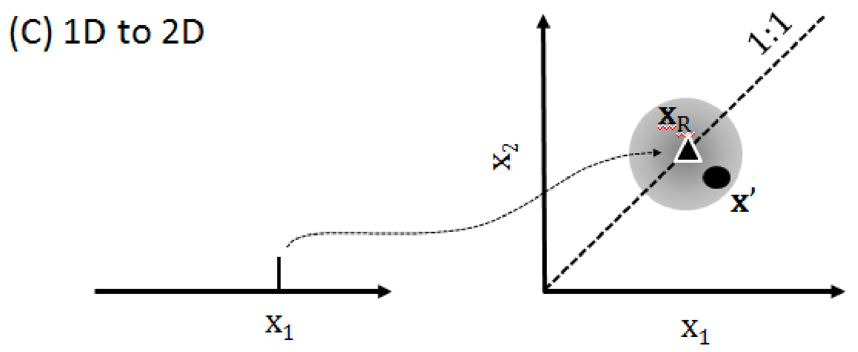
First, the dimension N' of the proposed successor state is generated by realizing $Q(N'|N)$, where N is the dimension of the current state.

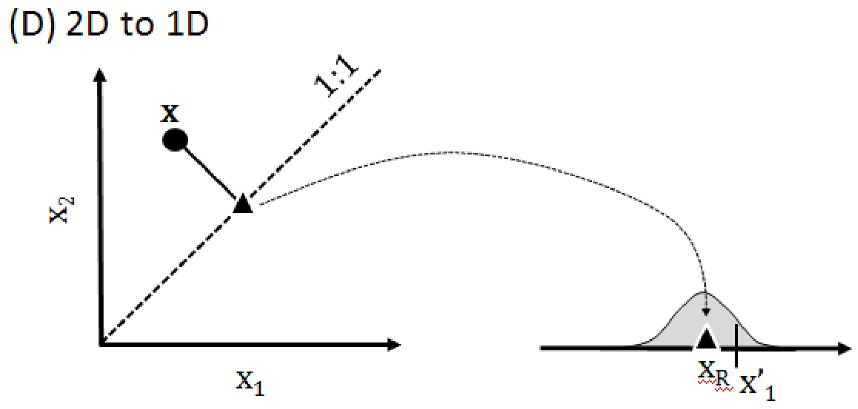
Next, a deterministic reference state X^R is created, with N' dimensions but as close to the current state X as possible.

Finally, the proposed successor state X' is created by realizing $Q(X'|X^R)$ (but with fixed dimension N').

This process is illustrated below for the $N = 1, N' = 2$ case:







In the case of a layered earth model, where each layer is described by a layer thickness and layer slowness, the reference state is defined in different ways, depending on whether the number of layers (that is, the dimension) remains the same, increases or decreases.

Case $N' = N$. The reference state is the current state.

Case $N' < N$ case. The reference state is created by pseudo-randomly choosing a pair of layers and aggregating them, and repeating the process as many times as necessary to achieve the necessary number of layers. Each time a pair of layers is aggregated, the slowness of the resulting single layer is the thickness-weighted average slowness of the two original layers.

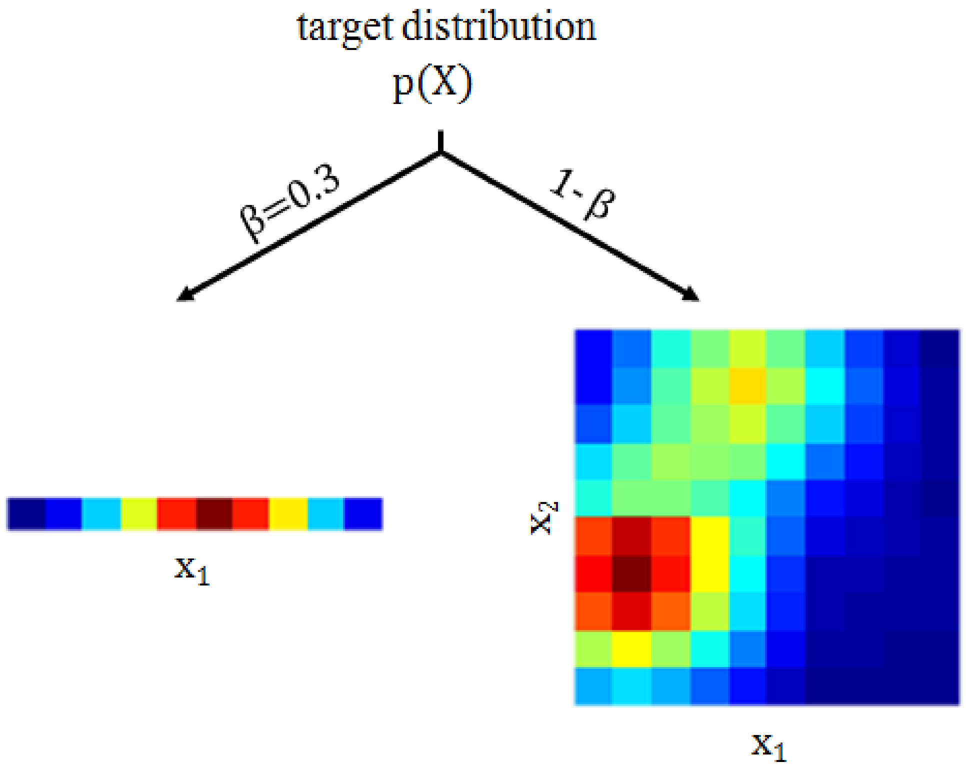
Case $N' > N$. the reference state is created from the current state by pseudo-randomly choosing a layer and splitting it into two layers at a pseudo-randomly choosing a point, and repeating the process as many times as necessary to achieve the necessary number of layers. Each time, two pieces are assigned the same slowness as the original layer.

The pseudo-random number i in the range $1 \leq i \leq N$ is generated using the rule:

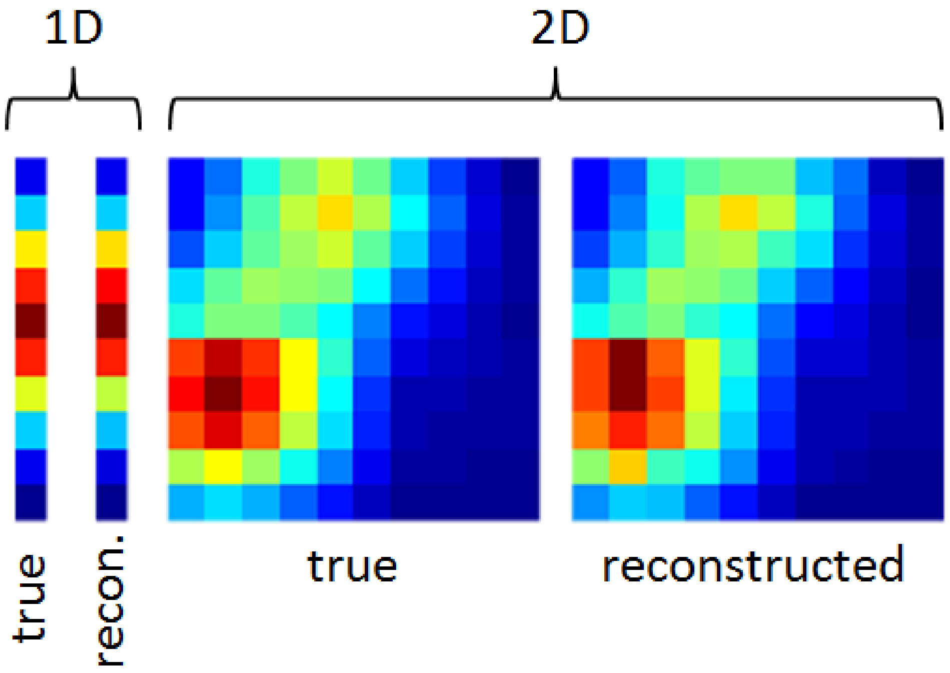
$$i = \text{mod}(\mathbf{p}^{\mathrm{T}}\mathbf{s}, N) + 1$$

where **s** is the slowness vector associated with the state and **p** is a specified vector of prime numbers. The pseudo-random number i is a deterministic function of the state.

6. A Simple Example of a Trans-Dimensional Target Distribution. Here the state X is either one dimensional (with probability $\beta = 0.3$) or two-dimensional (with probability $1 - \beta$). The one dimensional distribution is unimodal in a single variable x_1 and the two-dinensional distribution is bimodal is a pair of variables (x_1, x_2). Each x is a inter in the range $1 - 10$.



7. A Simple Test of Algorithm. MH was used to generate 10^5 realizations of X using the target distribution from 6, and histograms of them were used to reconstruct $P(X)$. The Q's were discrete approximation to Normal distributions, as described above. The reconstruction distribution closely matches the true distribution, with $\beta^{est} \approx 0.30$ and:



8. A More Complicated Test of the Algorithm. The test uses a model that consists of layers over a half-space. The model has N layers, each with a thickness H_i and a slowness s_i, and the half-space has a slowness, so that an N-layer model is represented by $2N + 1$ parameters. All parameters are discrete and bounded, $1 \leq N \leq K_N$ layers, $1 \leq H_i \leq K_H$ layer thicknesses and $1 \leq s_i \leq K_S$ layer and half-space slownesses. In this example, we use $K_N = 10$, $K_H = 50$ and $K_s = 50$.

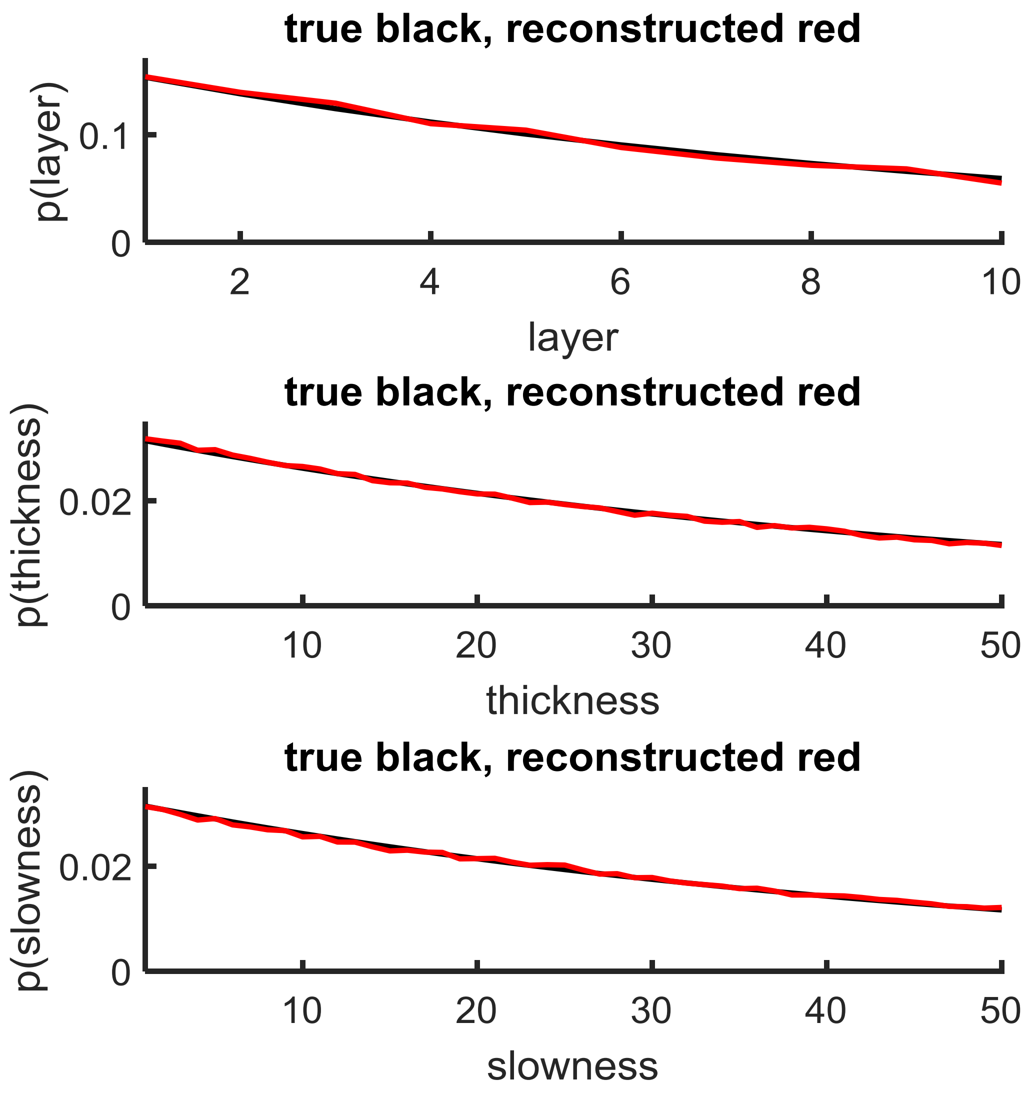
The true (or target) distribution $P(X)$ is the product of truncated geometric distributions for number of layers, layer thicknesses and layer and halfspace slownesses, with "success" parameters of $\nu_L = 1/K_L$, $\nu_H = 1/K_H$ and $\nu_S = 1/K_S$, respectively. The geometrical distribution monotonically decays with x, with the rate being controlled by the success parameter.

I chose the geometric distribution because it is a non-trivial discrete distribution that has a simple analytic formula.

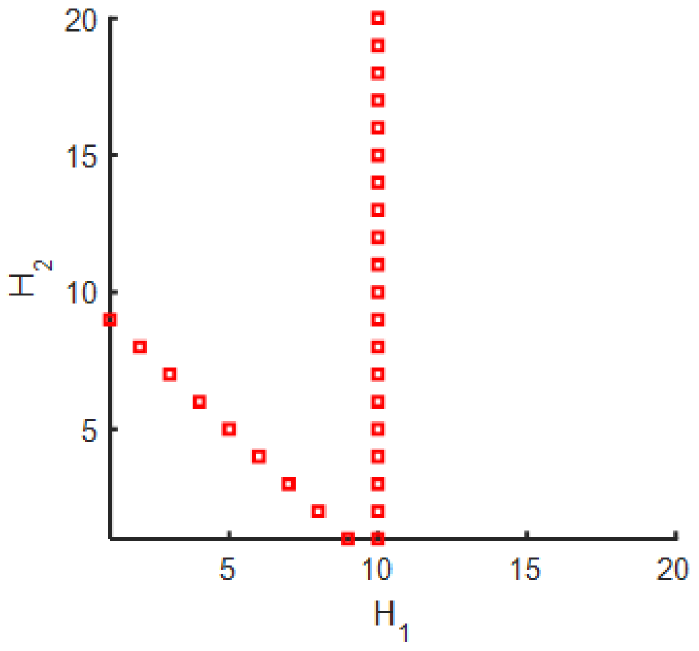
I used MH to generate 10^6 realizations of the model state. I then built histograms of number of layers, thicknesses and slowness and compared them to the geometric distribution. The match is excellent (see Figure). Note, however, that the verification is incomplete, since I checked only three projections of a multivariate distribution with:

$$\sum_{\ell=1}^{10} (2\ell + 1) = 120$$

dimensions. Note also that even 10^6 realizations sample this high dimensional space only very sparsely.



9. Issues associated with the over-fitting the data. A model with a larger number of parameters tends to over-fit the data, relative to a model with a lower number of parameters. This behavior is guaranteed to occur in trans-dimensional layered models, because a model space with N_2 layers *contains* the space of all $N_1 < N_2$ models. It is the subspace in which two adjacent layers have the same slowness. In fact, as is shown in the figure below, it contains many instances of this model, since N_2 layers can be aggregated into N_1 layers in many different ways. Thus, the error of the a best fitting higher dimensional model can never be worse than that of a lower dimensional model, and may by slightly better, since the higher dimensional model may be better able to fit noise.



Consider N_d Normally-distributed observed data d_i^{obs} with prior variance σ_d^2, together with corresponding predicted data d_i^{pre}. The total error:

$$E = \sum_{i=1}^{N_d} \frac{e_i^2}{\sigma_d^2} \quad \text{with} \quad e_i = d_i^{obs} - d_i^{pre}$$

is approximately chi-squared distributed with $\nu = N_d - (2N + 1)$ degrees of freedom, where N is the number of layers.

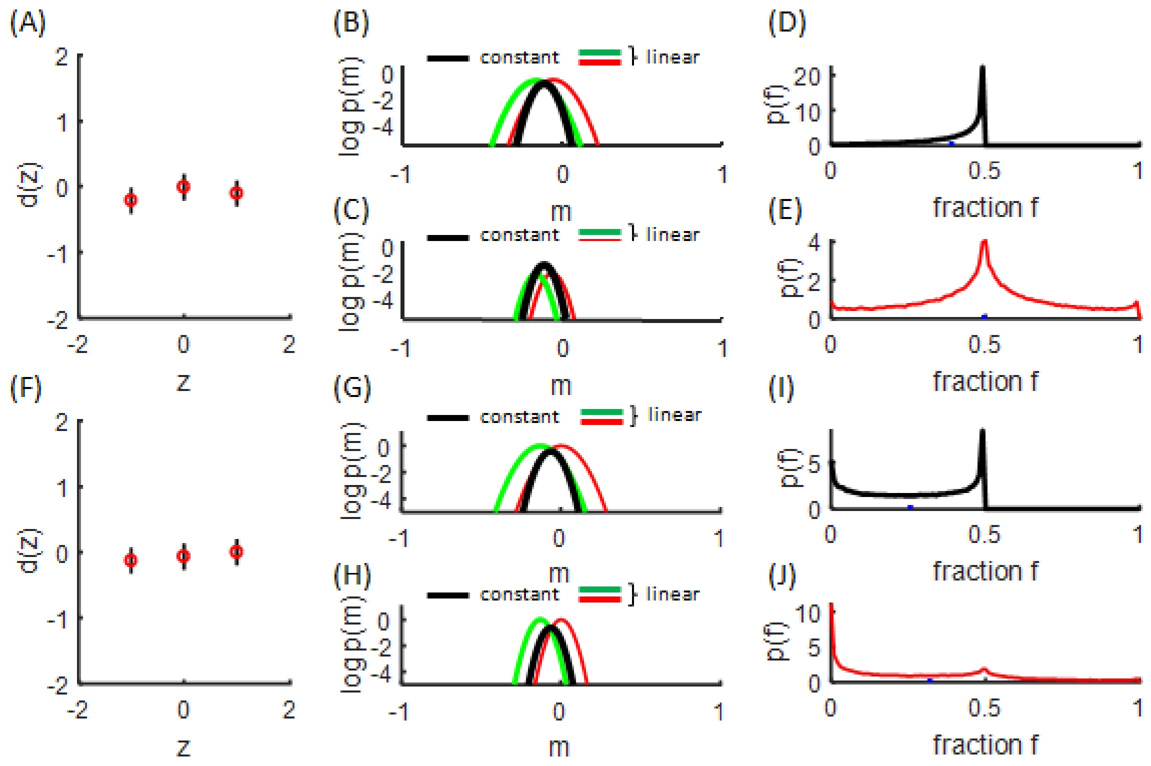
Suppose that the probability of the data given state X is $P_E(\mathbf{d}^{\text{obs}}|X) = p_E(E)$, where $p_E(E) \propto \exp(-½E)$ is a multivariate Normal distribution in the individual errors. Supposing two states, X_1 and X_2, the ratio R of their likelihoods is:

$$R = \frac{p_E(E_1)}{p_E(E_2)}$$

$$\log(R) = -½(E_1 - E_2)$$

This definition of R is problematical in the trans-dimensional case, as is illustrated in the following example. Suppose that $\nu = 100$, so that $\bar{E} = \nu = 100$ and $\sigma_E = \sqrt{2L} \approx 14$ and suppose two experiments, one associated with state X_1 with error $E_1 = 95$ and the other associated with state X_2 with error $E_2 = 105$. These two errors are not significantly different when judged by a chi-squared test, because both fall within one standard deviation of the mean. One cannot rule out the Null Hypothesis that the difference in fit is due to random variation; state X_1 does not fit the data significantly better than state X_2. However, the ratio of likelihoods of the two states is $R = \exp\{5\} \approx 150$. Furthermore, the ratio grows very quickly with the number of data; for example, the comparable ratio is about 9.1×10^6 when $\nu = 1000$. Assuming, for the moment, a uniform prior $P_A(X) = 1$, so that $P(X_1|\mathbf{d}^{\text{obs}})/P(X_2|\mathbf{d}^{\text{obs}}) = R$, we find that X_1 is much more probable than X_2, even though X_1 and X_2 fit the data to similar degrees. This seemingly contradictory result can be resolved in a case where the *number* of models with the same error grows very quickly with E. (This behavior occurs in classical chi-squared analysis, where the number of models with error E scales as the surface area of a ν-dimensional hypersphere of radius E). Then we could understand that yes, while the probability of X_2, relative to that of X_1 is not very high, this is offset by the existence of many alternatives to X_2, all with the same error, so the probability of selecting *one* of them is still high. However, this reasoning is making the assumption that the states span spaces of equal dimension, and is not true when the states have different dimensions.

This issue is illustrated in the following ultra-simplified trans-dimensional model. Consider uncorrelated observed data $\mathbf{d}^{\text{obs}}$ of length $N_d = 3$ and a corresponding auxiliary vector $\mathbf{z} = [-1,0,1]^{\text{T}}$. Now suppose a trans-dimensional model consisting of an $M = 1$ parameter constant model, such that $\mathbf{d}^{\text{pre}} = [m_1, m_1, m_1]^{\text{T}}$; and an $M = 2$ parameter linear model $\mathbf{d}^{\text{pre}} = [m_2, ½(m_2 + m_2), m_3]^{\text{T}}$. Note that the linear model contains the constant model as a special case, so the error of the former can never be less than the error of the latter. The likelihood is defined as $p(\mathbf{d}^{obs}|X) \propto \exp(-½E)$, where $E = \mathbf{e}^{\text{T}}\mathbf{e}/\sigma_d^2$, with $\mathbf{e} = \mathbf{d}^{\text{obs}} - \mathbf{d}^{\text{pre}}$ and $\sigma_d^2 = (0.1)^2$ is the variance in the data. Part A of the figure, below, shows synthetic data, realized for the constant model with $m_1^{true} = 0$. Part B shows the log-likelihood for the constant (black) and linear model spaces (red and green corresponding to two orthogonal slices through the peak of the linear model). In this example, best-fitting constant model fits the data almost (but not quite) as well as the linear model. Part C shows a revised version of the log-likelihood, adjusted for the different degrees of freedom for the two model; that is, E had been multiplied by $N_d/(N_d - M)$. The constant model now has the larger log-likelihood. Now suppose that the prior has been adjusted so that the constant model and the linear model are equally likely when they have the same log-likelihood. The fraction of constant realizations that occur in an ensemble is shown for the uncorrected (Part D) and corrected (Part E) log-likelihood functions. In the uncorrected case, the typical $\mathbf{d}^{\text{obs}}$ is associated with only about 40% constant models, even though the true model is constant. The correction increases the percentage, but only to 50%. Parts F-J of the figure show corresponding results when the true model is linear with $[m_2, m_3]^{\text{T}} = [-0.1, 0.1]^{\text{T}}$. Now linear models are favored in both cases, though not overwhelmingly so.



I consider the preceding discussion a strong argument for correcting the error for the degrees of freedom; that is replacing E with:

$$\frac{N}{\nu}E$$

This change leads to no change in relative likelihood when comparing two models having same dimension, but it offsets the tendency for a high-dimensional case to over-fit the data.

As an alternative, we might consider writing:

$$P\left(X\middle|E^{\text{obs}}\right) \propto P_{\chi^2}\left(E^{\text{obs}}\middle|X\right)\, P_A(X)$$

where $p_{\chi^2}(.)$ is the chi-squared distribution. The ratio R of likelihoods is then:

$$R = \frac{p_{\chi^2}(E_2, \nu_2)}{p_{\chi^2}(E_1, \nu_1)}$$

$$\log(R) = -\log\left[2^{\nu_2/2}\Gamma(\nu_2/2)\right] + (\nu_2/2 - 1)\log(\mathrm{E}_2) - \mathrm{E}_2/2$$

$$+\log\left[2^{\nu_1/2}\Gamma(\nu_1/2)\right] - (\nu_1/2 - 1)\log(\mathrm{E}_1) + \mathrm{E}_1/2$$

where $\Gamma(.)$ is the gamma function. When this version of the likelihood is used, two models that fit the data similarly well, when judged against the chi-squared distribution, will have similar probabilities of acceptance. One might argue that this choice of R gives too much weight to states with large error, since $p_{\chi^2}(E,\nu)$ falls off much more slowly with E than $\exp(-½E)$. However, the criticism is only valid when $\mathbf{d}^{\text{pre}}(\mathbf{m})$ spans the full space of the data, so that the number of models with error E grows very quickly with E (as they do in classical chi-squared analysis). In cases where the d_{i}^{pre}'s are very highly correlated with one another, the e_i's may not span the whole space and the alternative assumption embodied in above definition of R may be appropriate.

I note that the calculation of $\log(R)$ can be expedited by tabulating the values of $\log\left[2^{\nu_1/2}\Gamma(\nu_1/2)\right]$, for there are only K_N of them. I also note that R is best calculated as $\exp(\log(R))$ and not as $p_{\chi^2}(E_2,\nu_2)/p_{\chi^2}(E_1,\nu_1)$, for the division in the latter is very unstable at points where $p_{\chi^2}(E_1,\nu_1) \ll 1$.

10. Issues associated with the model-space volume and the choice of the prior distribution.

One possible choice of the prior is that every possible configuration has equal probability (the *configuration-uniform* prior). A N-layer model has a total of $Z_N = (K_H)^N(K_s)^{N+1}$ configurations, so that the total prior probability of N-layer models is:

$$P_A(X) = 1 \quad \text{and} \quad P_A(N) = \frac{Z_N}{\sum_{i=1}^{K_N} Z_N}$$

Another possibility is to define the prior so that models of any number of layers have equal probability (the *layer-uniform* prior). In this case, the prior probability of a configuration must decrease with its number of layers:

$$P_A(X) = \frac{[Z_N]^{-1}}{\sum_{i=1}^{K_N}[Z_N]^{-1}} \quad \text{and} \quad P_A(N) = 1$$

Both priors are plausible, but they can lead to different outcomes.

First, consider the case of the configuration-uniform prior. Suppose that the best-fitting model with N_2 layers fits the data as well as a model with $N_1 < N_2$ layers, so that the ratio of likelihoods is unity. Even so, the total probability $P_{N_2}^T$ of all N_2 models will typically be much higher than the total probability $P_{N_1}^T$ of all N_1 models, because the model space for the former has the higher number of dimensions. For example, consider a likelihood function that, for fixed number of layers N, is Gaussian:

$$P(X|\mathbf{d}^{\text{obs}}) = P_E(\mathbf{d}^{\text{obs}}|X)P_A(X) \quad \text{with} \quad P_A(X) = 1 \quad \text{and}$$

$$P_E\left(\mathbf{d}^{\mathrm{obs}}|X\right) = P_{max}^{(N)} \exp\left\{-½\Delta\mathbf{m}^{(\mathrm{N})\mathrm{T}}\left[\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}\right]^{-1}\Delta\mathbf{m}^{(\mathrm{N})}\right\}$$

where $\mathbf{m}^{(\mathrm{N})}$ are the $2N+1$ model parameters associate with X, $\Delta\mathbf{m}^{(\mathrm{N})}$are their deviations from the mean, and $\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}$ is a covariance matrix. At fixed N, the total probability is:

$$P_N^T = A^{(N)}P_{max}^{(N)} \quad \text{with} \quad A^{(N)} = (2\pi)^{(2N+1)/2}\left|\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}\right|^{1/2}$$

The ratio $A^{(N+1)}/A^{(N)}$ is, in general a strong function of covariance. Consider, for instance, the special case $\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})} = \sigma_m^2\mathbf{I}$, it is $A^{(N+1)}/A^{(N)} = 2\pi\sigma_m^2$. Now suppose that $P_{max}^{(N+1)} = P_{max}^{(N)} = 1$ and $\sigma_m = 2$ grid nodes. Then the ratio of total probability is about a factor of 25; that is, models with $(N+1)$ layers occur twenty-five times as often in a set of realizations than do models with N layers, even though both fit the data equally well.

Now consider the case of the layer-uniform prior. The ratio of total probability now includes the ratio of priors:

$$\frac{P_A(X_{N+1})}{P_A(X_N)} = \frac{Z_N}{Z_{N+1}} = \frac{(K_H)^N(K_s)^{N+1}}{(K_H)^{N+1}(K_s)^{N+2}} = (K_HK_s)^{-1}$$

Typically, the full range of model parameters are chosen to be much larger than the variance, so the ratio of priors $(K_HK_s)^{-1}$ overcorrects the factor of $2\pi\sigma_m^2$, leading to a situation where realization with $(N+1)$ layers are under-represented when compared to models with N layers. Furthermore, the results depend strongly on the choices of K_H and K_s, whereas we would have preferred it to be insensitive to them, as long as they were chosen to be large enough that model volume completely enclosed the region of high probability.

Most authors use layer-uniform priors. However, I propose using an *area-uniform* prior, defined as:

$$P_A(N) = \frac{\left[A^{(N)}\right]^{-1}}{\sum_{i=1}^{K_N}[A^{(i)}]^{-1}}$$

The area-uniform prior corrects for the tendency of the model volume to grow with dimension, but does not depend upon the arbitrary choice of bounds. However, it can be criticized because it depends on the data, whereas a fully-consistent prior should not. However, the data enters only weakly; $P_A(N)$ depends exclusively on the behavior the covariance $\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}$, and that is controlled by the mapping from state X to slowness function $s(z_j)$ and the functional relationship between slowness and the predicted data d_i^{pre}. When these two relationships are linear, the prior is data-independent (except it scales with the variance of the data). The *value* of the data enters only when the relationships are non-linear. Given enough experience with a particular class of problems, I imagine that it would be possible to choose $\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}$ *a priori*, in which case it could be understood as a prior covariance of the model.

In general, the covariance $\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}$ is not known, and must be estimated on-the-fly. I use a two-step process: I first realize $P_E(X|\mathbf{d}^{\mathrm{obs}})$ or $P_E(X|E^{\mathrm{obs}})$ with a uniform prior, estimate $\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}$ from the realizations, and then realize it again with an area-uniform prior. I estimate mean and covariance by performing the sums:

$$K_i^{(N)} = \sum_{k=1}^{N_n} \begin{cases} 1 \text{ if the model has N layers} \\ 0 \text{ otherwise} \end{cases} \quad \text{and}$$

$$S_i^{(N)} = \sum_{k=1}^{N_n} \left[m_i^{(N)}\right]_k \quad \text{and} \quad W_{i,j}^{(N)} = \sum_{k=1}^{N_n} \left[m_i^{(N)}\right]_k \left[m_j^{(N)}\right]_k$$

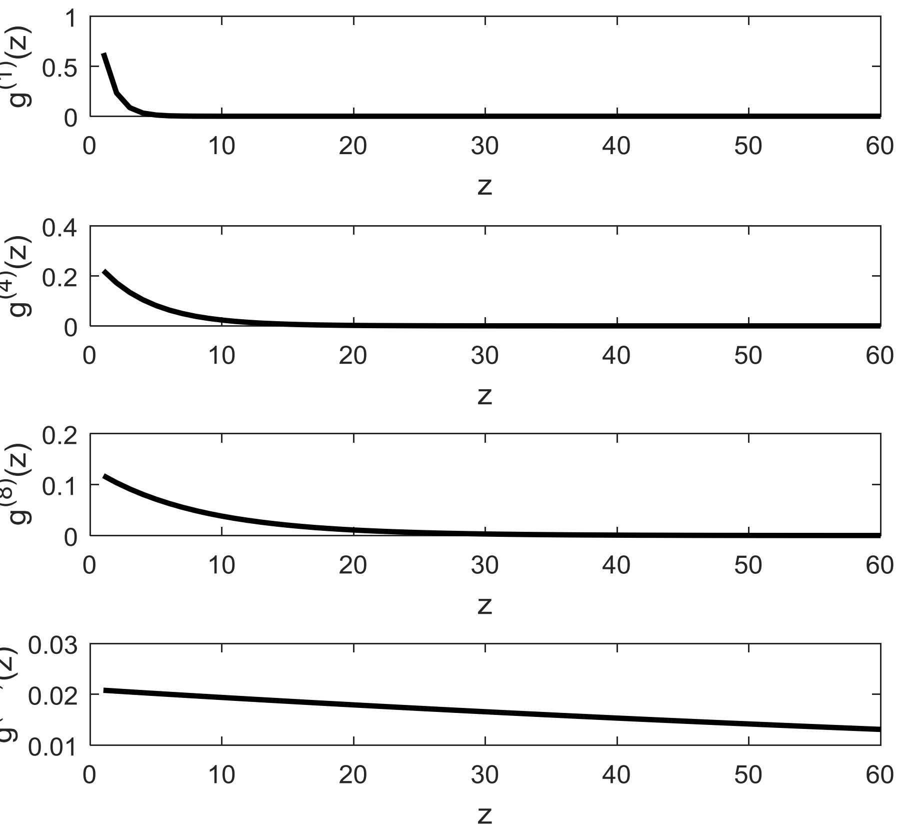
for all $1 \leq N \leq K_N$, $1 \leq i \leq (2N+1)$ and $1 \leq j \leq (2N+1)$ during the loop over the $1 \leq k \leq N_r$ realizations. Then:

$$\bar{m}_i^{(N)} \approx S_i^{(N)} / K_i^{(N)}$$

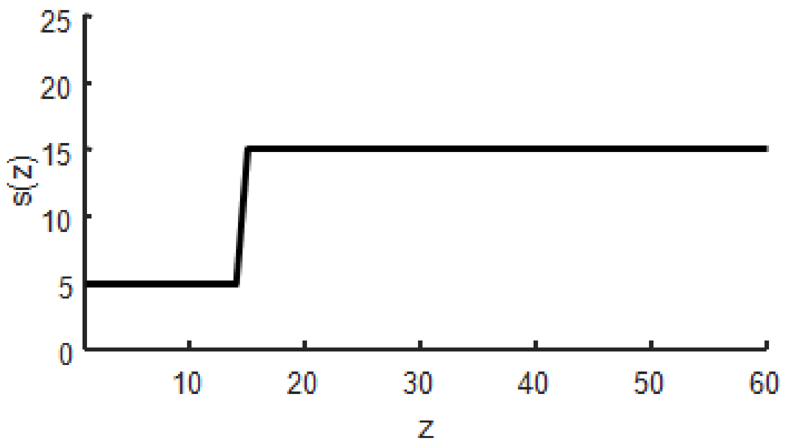
$$C_{i,j}^{(N)} \approx \left(W_{i,j}^{(N)} - S_i^{(N)}\bar{m}_j^{(N)} - \bar{m}_i^{(N)}S_j^{(N)} + \bar{m}_i^{(N)}\bar{m}_j^{(N)}\right) / \left(K_i^{(N)} - 1\right)$$

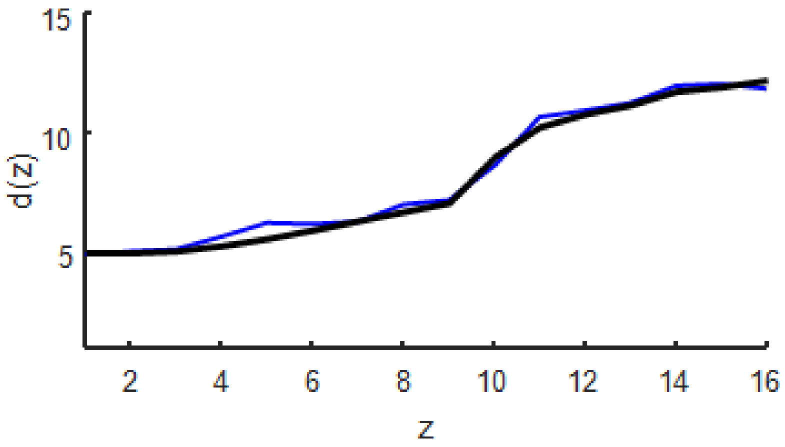
Since I am using integer variables, I reset each diagonal element of $\mathbf{C}_{\mathrm{m}}^{(\mathrm{N})}$ to $1/4$ if it falls below that threshold.

11. A sample distribution with a configuration-uniform prior. In this example, I use $K_N = 10$, $K_H = 20$ and $K_s = 20$, so that the full trans-dimensional model space has about 3.2×10^6 grid nodes. A slowness vector **s** with elements $s(z_i)$ consisting of the slowness evaluated at integer depth $1 < z_i < (K_N + 1)K_H$ is calculated for model X. Each of the $N_g = 16$ data d_i is a weighted average of $s(z_j)$, with an averaging kernel $g^{(i)}(z_j)$. I choose this kernel so that it has unit area and elements that exponentially decay with depth. The g's are ordered so that the decay rate decreases with i:

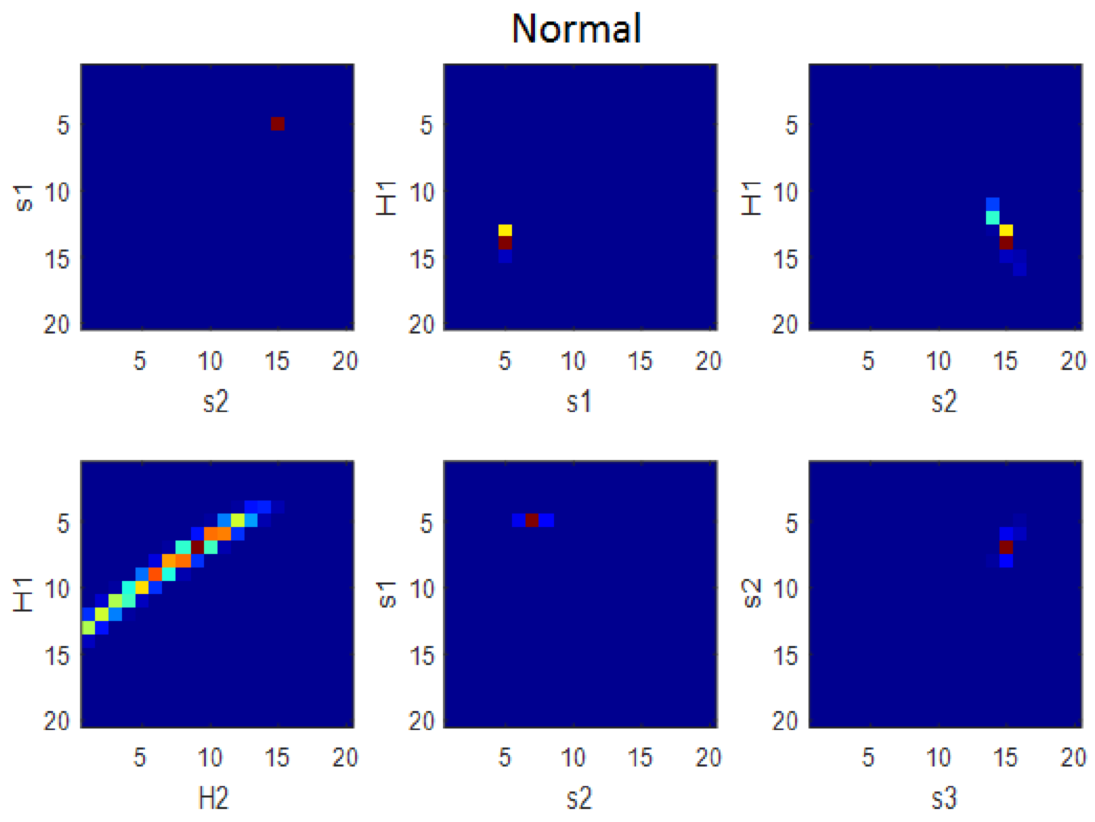


The true model has one low-slowness layer. The observed data are the true data plus uncorrelated Normally-distributed random noise with variance $\sigma_d^2 = (0.4)^2$.

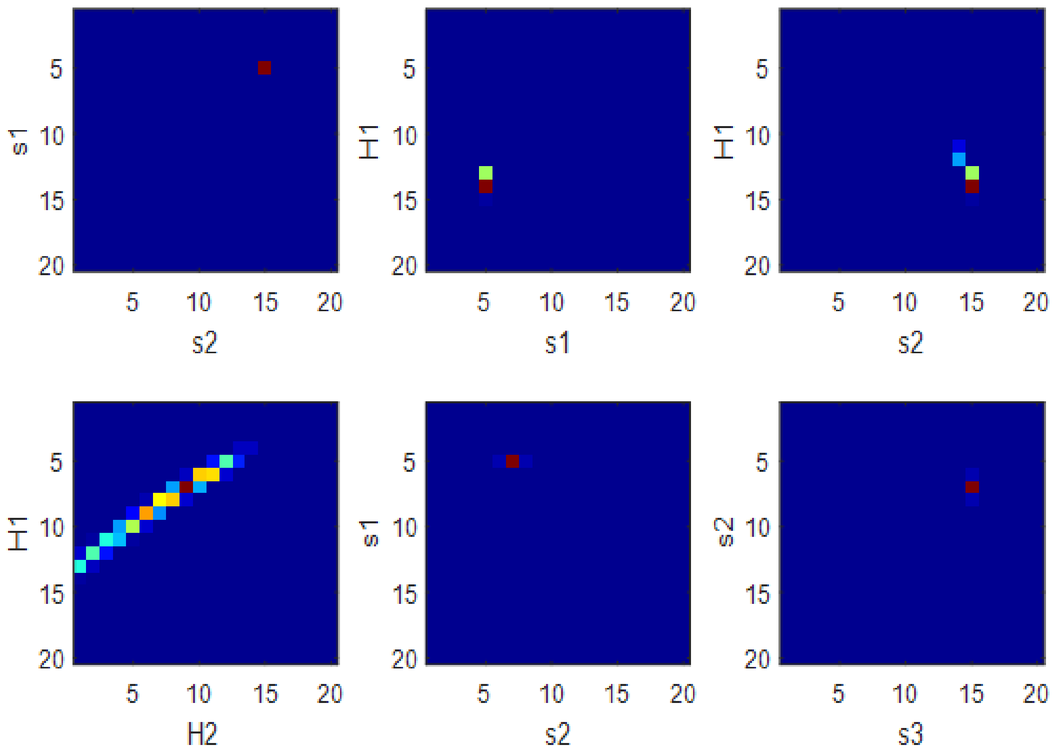




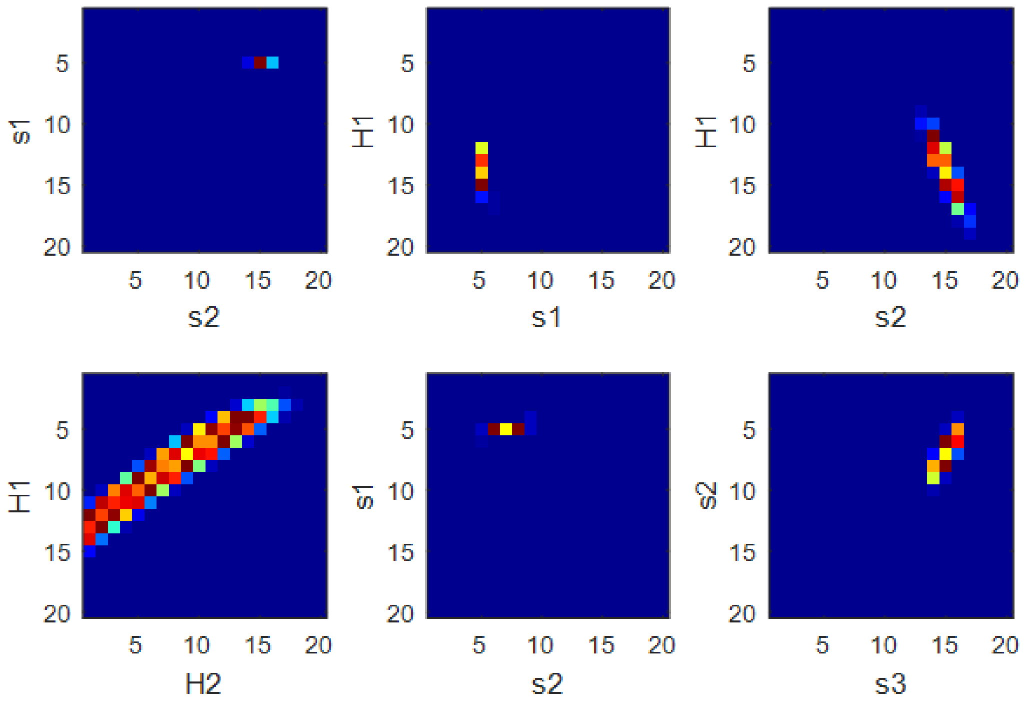
I compute the $P(X, \mathbf{d}^{\text{obs}})$ and $P(X, E^{\text{obs}})$ distributions with configuration-uniform priors. They are evaluated on a one-layer $20 \times 20 \times 20$ (H_1, s_1, s_2) grid and a two-layer $20 \times 20 \times 20 \times 20 \times 20$ $(H_1, H_2, s_1, s_2, s_3)$ grid. Slices through them are shown below, with the top row for the one-layer grid and the bottom row for the two-layer grid. The Normal, Normal-corrected and chi-squared likelihood functions are shown below (for a configuration uniform prior).


Normal
s1
s2
H1
s1
H1
s2
H1
H2
s1
s2
s2
s3
5
10
15
20

Normal-corrected

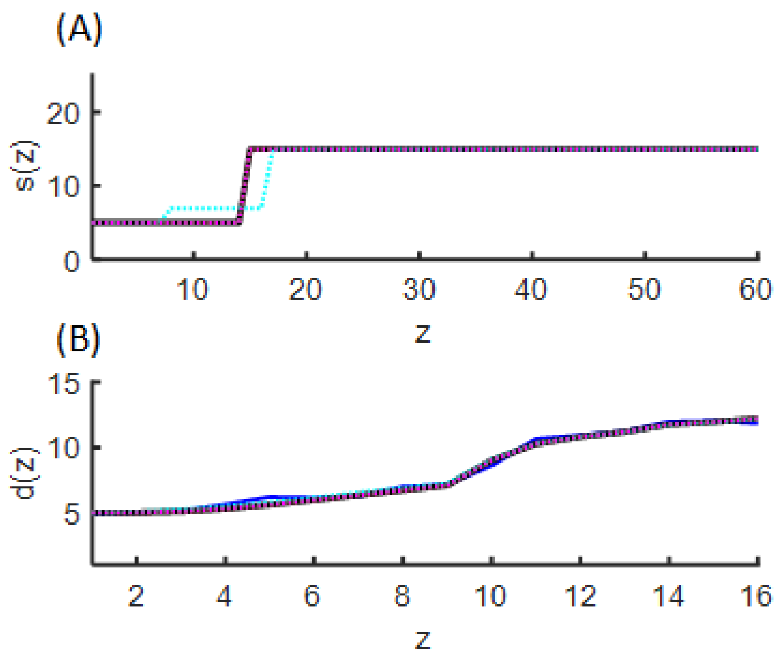


chi-squared



As expected, the chi-squared likelihood function has the widest peak.

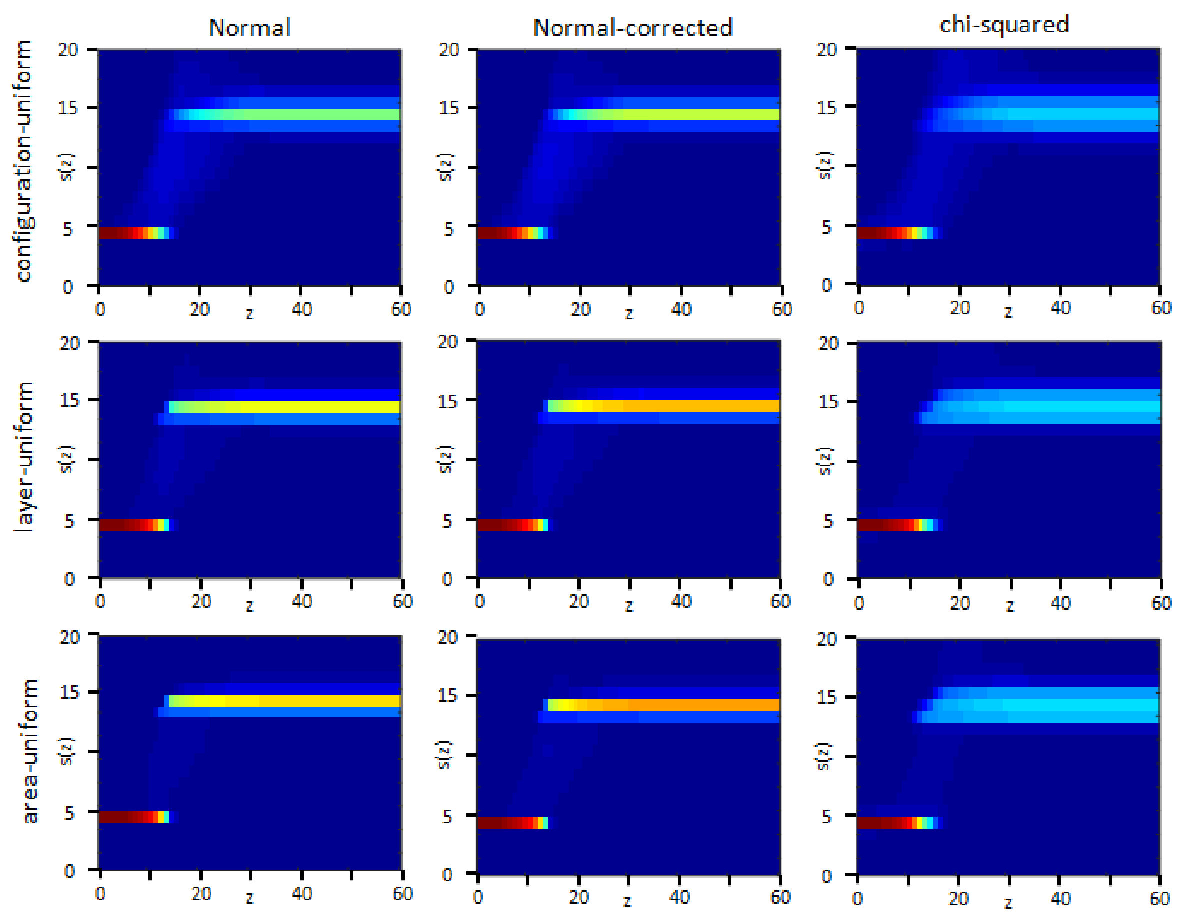
The best-fitting one-layer (red) and two layer (cyan) models are compared to the true model (back) in the Part A of the figure below. The corresponding fit to the data is shown in Part B.



The uncorrected error of the best-fitting one-layer solution of $E_1^{min} = 5.94$ (with $\nu_1 = 16 - 3 = 13$) is larger than the error of the best-fitting two-layer solution of $E_2^{min} = 5.21$ ($\nu_2 = 16 - 5 = 11$). Consequently, the likelihood of the best-fitting two layer model is about twice that of the best-fitting one layer model. This difference in error is not significant at the 95% confidence level under an F-test. And, indeed, the predicted data (magenta, cyan) appears to fit the observed data (black) equally well. When corrected for the differing number of degrees of freedom, the error becomes $E_1^{min} = 7.35$ and $E_2^{min} = 7.59$.

12. Inversion results. I used MH to create 10^7 realizations of $P(X|\mathbf{d}^{\text{obs}})$ and $P(X|E^{\text{obs}})$ for all nine combinations of likelihood functions and priors. MH does a good job reconstructing all the distributions. However, the approximation involving covariance that is used to estimate $A^{(N)}$ can be off by as much as 20%, so the area-uniform prior can be off by a similar amount.

Histograms of three sets of slowness profiles are shown below:



As expected, the chi-squared distributions tend to be wider, reflecting the wider likelihood function. The percentage of 1-layer models is given below:

	Normal	Normal-corrected	chi-squared
configuration-uniform	0.23%	0.27%	0.29%
layer-uniform	52.5%	49.1%	51.7%
area-uniform	66.9%	60.35%	58.7%

As expected, the configuration-uniform priors lead to a set of realizations that contain very few instances of 1-layer models.

None of the other six combinations gives a really high probability to 1-layer models (even though the true model has one layer), though the area-uniform prior does best, with probabilities of $60-65\%$. I think that this behavior is due to the larger-dimension models containing the lower-dimensional ones. I think that it caused probability to be distributed evenly between dimensions equal to or greater than the true dimension. This should be an area of future research.

```
% state X, the trans-dimensional model structure:
% X.Nlayers: number of layers
% X.thickness: vector of layer thicknesses
% X.slowness: vector of layer and halfspace slownesses

function [ d, b, bc ] = Xtod( X, Klayers, Kthicknesses )
% converts state X to slowness vector d
% input:
% X: state
% Klayers: maximum number of layers
% Kthicknesses: maximum thickness
% output:
% d: vector of slownesses, s(z)
% b: vectors of 0's and 1's, with 1's at boundaries
% bc: vector of 0's and +/-1's, with 1's at boundaries with
%     positive slowness jump, -1's at boundaries with negative
%     jump
threshold = 5;
Nd = (Klayers+1)*Kthicknesses;
Nlayers = X.Nlayers;
d = [];
b = [];
bc = [];
for i=[1:Nlayers]
    d = [d; X.slowness(i)*ones(X.thickness(i),1) ];
    Ds = abs(X.slowness(i+1)-X.slowness(i));
    b = [b; zeros(X.thickness(i)-1,1); 1 ];
    bc = [b; zeros(X.thickness(i)-1,1); sign(Ds) ];
end
i = Nd-length(d);
d = [d; X.slowness(Nlayers+1)*ones(i,1) ];
i = Nd-length(b);
b = [b; zeros(i,1) ];
i = Nd-length(bc);
bc = [bc; zeros(i,1) ];

end
```

```
function [ P ] = makeP( p, L )
% input
% p
% KxK matrix that gives p(i;j), 1<=i<=K, 1<=j<=K
% the probability of i for case j
% (typically, when the maximum is at j)
% L
% an integer parameter L>>K that will be used
% to create realizations of p(i;j)
% output
% P.K
% the value of K
% P.L
% the value of L
% P.p
% a modified version of p(i;j), in which
% L*P.p(i;j) is exactly and integer in the
% range 1-L.
% P.logp
% log of P.p
% P.r
% a L by K table such that P.r(:,j),
% with 1<=j<=K, is an array of length L that has exactly L*P.p(i;j)
% occurences of each integer in the range 1<=j<=K. A realization
% of p for case j is then
% i_realization = P.r(unidrnd(L),j)

[K, i] = size(p);
P.K = K;
P.L = L;

for j=[1:K]
    pa = p(:,j);
    sump = sum(pa);
    pd = ceil(L*pa/sump);
    sumpd = sum(pd);
    % insure has exactly 100 area
    while( sumpd>L ) % if area is too big, reduce it
        i = unidrnd(K);
        if( pd(i) == 1 )
            continue;
        end
        pd(i) = pd(i)-1;
        sumpd = sum(pd);
    end
    while( sumpd<L ) % if area is too small, increase it
        i = unidrnd(K);
        pd(i) = pd(i)+1;
        sumpd = sum(pd);
    end
    pa = pd/sumpd;
    jl=1;
    r = zeros(L,1);
    for i=[1:K]
        jr=jl+pd(i)-1;
        r(jl:jr)=i;
        jl=jr+1;
    end
    P.p(:,j)=pa;
    P.r(:,j)=r;
end

P.logp=log(P.p);

end
```

```
function [ X0 ] = refstate( X1, N, H, S, Nlayers )

% create the reference state X0 which has the same number
% of layers as the new state, with other attributes chosen
% to be close to the original state.  This section chooses
% some critical parameters deterministically based on the
% values of the slownesses in X1, but pseudo-randomly.
% output:
% X0: reference state
% input:
% X1: ols state
% N: probability structure for number of layers
% H: probability structure for layer thicnnesses
% S: probability structure for layer slownesses
% Nlayers: number of layers in reference state

% table of primes; must be at least of length N.K
p = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71]';
if( length(p)<N.K )
    fprintf('Error: table of primes too short\n');
    xxx
end

if( X1.Nlayers == Nlayers ) % no change in dimension
    % reference state is old state
    X0 = X1;
    cs = 1;
elseif( X1.Nlayers > Nlayers ) % dimension goes down
    % aggregate a pseudo-randomly chosen pair of adjacent layers
    X0 = X1;
    while ( X0.Nlayers > Nlayers )
        t = X0.thickness;
        s = X0.slowness;
        N0 = X0.Nlayers;
        n1 = mod(sum(p(1:N0).*s(1:N0)),N0-1)+1; % pick pseudo-random layer
        tt=t(n1)+t(n1+1); % aggregated thickness
        if( tt>H.K ) % don't let aggregated thickness exceed limit
            tt = H.K;
        end
        % weighted average of the slownesses
        sw = t(n1)*s(n1)+t(n1+1)*s(n1+1);
        sw = floor(sw/tt);
        if( sw<1 ) % don't let average stray out of allowed range
            sw=1;
        elseif( sw>S.K )
            sw = S.K;
        end
        t2 = [ t(1:n1-1); tt; t(n1+2:X0.Nlayers) ];
        s2 = [ t(1:n1-1); sw; s(n1+2:X0.Nlayers+1) ];
        X0.thickness = t2;
        X0.slowness = s2;
        X0.Nlayers = X0.Nlayers-1;
    end
    cs = 2;
else
    % dimension goes up, split a pseudo-randomly chosen layer
    % a a pseudo-randomly chosen place point

    X0 = X1;
    while ( X0.Nlayers < Nlayers )
        t = X0.thickness;
        s = X0.slowness;
        N0 = X0.Nlayers;
        n1 = mod(sum(p(1:N0).*s(1:N0)),N0)+1; % pick pseudo-random layer
        n2 = t(n1); % thickness of layer
        if( n2 == 1 )
            ta=1;
            tb=1;
        else
            ta = mod(sum(p(2:N0+1).*s(1:N0)),n2-1)+1; % pick pseudo-random
subdivision
            if( ta<1 )
                ta=1;
            end
            tb = n2-ta;
            if( tb<1 )
                tb=1;
            end
        end
        t2 = [ t(1:n1-1); ta; tb; t(n1+1:N0) ];
        s2 = [ s(1:n1-1); s(n1); s(n1); s(n1+1:N0+1) ];
        X0.thickness = t2;
        X0.slowness = s2;
        X0.Nlayers = X0.Nlayers+1;
    end
    cs=3;
end

if( 0 ) % for debugging purposes, check consistency of X0
    if( length(X0.thickness) ~= X0.Nlayers )
        fprintf('Error: thickness length mismatch, case %d\n', cs );
        xxx;
    end
    for i=[1:X0.Nlayers]
        j = X0.thickness(i);
        if( (j<1) || (j>X0.Kthicknesses)  )
            fprintf('Error: thickness bounds, case %d layer %d value %d\n',
cs, i, j );
            xxx;
        end
    end
    if( length(X0.slowness) ~= (X0.Nlayers+1) )
         fprintf('Error: slowness length mismatch, case %d\n', cs );
         xxx;
    end
    for i=[1:(X0.Nlayers+1)]
        j = X0.slowness(i);
        if( (j<1) || (j>X0.Kslownesses)  )
            fprintf('Error: slowness bounds, case %d layer %d value %d\n',
cs, i, j );
```

```
            xxx;
        end
    end
end

end
```

```
function [X2] = randomstate( X1, N, H, S )
% make a random state X2 that is "close to" X1

% start another random state close to the first one, but with
% a possibly different number of layers
X2.Klayers = X1.Klayers;
X2.Kthicknesses = X1.Kthicknesses;
X2.Kslownesses = X1.Kslownesses;
X2.Nlayers = N.r(unidrnd(N.L),X1.Nlayers);

% make the refernence state with this number of layers
X0 = refstate( X1, N, H, S, X2.Nlayers );

% random generation of the new state based on the reference state
X2.thickness = zeros( X2.Nlayers, 1 );
for i=[1:X2.Nlayers]
    X2.thickness(i) = H.r(unidrnd(H.L),X0.thickness(i));
end
X2.slowness = zeros( X2.Nlayers, 1 );
for i=[1:X2.Nlayers+1]
        X2.slowness(i) = S.r(unidrnd(H.L),X0.slowness(i));
end

end
```

```
function [ logP, d, b, bc ] = logPE2ofd( X, Klayers, Kthicknesses )
% log likelihood logP of a state X, by lookup into PE2 structure
% input:
% X: the state
% Klayers: maximum number of layers
% Kthicknesses: maximum number of thicknesses
% output:
% logP: log likelihood
% d: slowness vector
% b: interface vector, with 1's at interfaces
% bc: interface vector, with +/- 1's at interfaces,
%                       depending upon the sign of the jump

global PE2

% transform state to data
[d,b,bc]=Xtod(X,Klayers,Kthicknesses);

if( X.Nlayers == 1 )
    H1 = X.thickness(1);
    s1 = X.slowness(1);
    s2 = X.slowness(2);
    logP = PE2.logP1(H1,s1,s2);
else
    H1 = X.thickness(1);
    H2 = X.thickness(2);
    s1 = X.slowness(1);
    s2 = X.slowness(2);
    s3 = X.slowness(3);
    logP = PE2.logP2(H1,H2,s1,s2,s3);
end

end
```

```
function [ logp ] = logcondprob( X2, X1, N, H, S )
% input:
% X2: state 2
% X1: state 1
% N: layer probability structure
% H: thickness probability structure
% S: slowness probability structure
% output
% logp: log probability of state X2, given state X1

% probability associated with layers
logp = N.logp( X2.Nlayers, X1.Nlayers );

% reference state corresponding to X1
X0 = refstate( X1, N, H, S, X2.Nlayers );

% probability associated with thicknesses
for i=[1:X0.Nlayers]
    logp = logp + H.logp( X2.thickness(i), X0.thickness(i) );
end

% probability associated with slownesses
for i=[1:(X0.Nlayers+1)]
    logp = logp + S.logp( X2.slowness(i), X0.slowness(i) );
end

end
```

```
% makePEK_G G for Gaussian, no correction for degrees of freedom

clear all;

load('mydata.mat');  % use the same realization of data

% set up for 2 layers
Nlayers=2;
PE2.Nlayers = Nlayers;
K = 20;
PE2.K = K;
Nd = (Nlayers+1)*K;
PE2.Nd = Nd;

% exponentially-decaying kernel G, such that d=Gs with s computed from X
k = [1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, 1/16, 1/25, 1/32, 1/40, 1/64,
1/80, 1/128]';
NG = length(k);
G = zeros( NG, Nd );
for i=[1:NG]
    g = exp(-k(i)*[1:Nd]);
    g = g/sum(g);
    G(i,:) = g;
end

% degrees of freedom
nu = zeros(Nlayers,1);
for i=[1:Nlayers]
    nu(i) = NG-(2*i+1);
end

PE2.NG = NG;
PE2.k = k;
PE.G = G;

sd=0.4;
PE2.sd = sd;

Htrue = [7, 7]';
strue = [5, 5, 15]';
dtrue = [];
for i=[1: Nlayers]
   dtrue = [dtrue; strue(i)*ones(Htrue(i),1) ];
end
i = Nd-length(dtrue);
dtrue = [dtrue; strue(Nlayers+1)*ones(i,1) ];
PE2.dtrue = dtrue;
Dtrue = G*dtrue;
PE2.Dtrue = Dtrue;
% Dobs = Dtrue + random('Normal',0,sd,NG,1);
PE2.Dobs = Dobs;

figure(10);
clf;
```

```
% plot model
subplot(2,1,1);
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
hold on;
xlabel('z');
ylabel('s(z)');
axis( [1, Nd, 0, K+5] );
plot( [1:Nd]', dtrue, 'k-', 'LineWidth', 3 );

% plot slowness
subplot(2,1,2);
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
hold on;
xlabel('z');
ylabel('d(z)');
axis( [1, NG, 1, 15] );
plot( [1:NG]', Dobs, 'b-', 'LineWidth', 2 );
plot( [1:NG]', Dtrue, 'k-', 'LineWidth', 3 );

E2 = zeros(K,K,K,K,K);
P2 = zeros(K,K,K,K,K);
logP2 = zeros(K,K,K,K,K);
Nlayers2=2;
first=1;
for Hi=[1:K]
for Hj=[1:K]
    Hpre = [Hi, Hj]';
for si=[1:K]
for sj=[1:K]
for sk=[1:K]
    spre = [si,sj,sk]';

    dpre = [];
    for i=[1:Nlayers2]
       dpre = [dpre; spre(i)*ones(Hpre(i),1) ];
    end
    i = Nd-length(dpre);
    dpre = [dpre; spre(Nlayers2+1)*ones(i,1) ];
    Dpre = G*dpre;
    e = (Dobs - Dpre)/sd;
    myE = e'*e;
    E2(Hi, Hj, si, sj, sk) = myE;
    logP2(Hi, Hj, si, sj, sk) = -myE/2;

    if( first==1 )
        Himin2=1; Hjmin2=1; simin2=1; sjmin2=1; skmin2=1;
        dmin2 = dpre;
        Dmin2 = Dpre;
        Emin2 = myE;
        first=0;
    elseif( myE<Emin2 )
        Himin2=Hi; Hjmin2=Hj; simin2=si; sjmin2=sj; skmin2=sk;
        dmin2 = dpre;
        Dmin2 = Dpre;
        Emin2 = myE;
    end

    if( (Hi==Htrue(1)) && (Hj==Htrue(2)) && (si==strue(1)) && (sj==strue(2))
&& (sk==strue(3)) )
        figure(10);
        subplot(2,1,1);
        plot( [1:Nd]', dpre, 'r:', 'LineWidth', 2 );
        subplot(2,1,2);
        plot( [1:NG]', Dpre, 'r:', 'LineWidth', 2 );
    end
end
end
end
end
end

P2 = exp(logP2);

figure(10);
subplot(2,1,1);
plot( [1:Nd]', dmin2, 'c:', 'LineWidth', 2 );
subplot(2,1,2);
plot( [1:NG]', Dmin2, 'c:', 'LineWidth', 2 );

Nlayers1 = 1;
E1 = zeros(K,K,K);
P1 = zeros(K,K,K);
first=1;
for Hi=[1:K]
    Hpre = [Hi]';
for si=[1:K]
for sj=[1:K]
    spre = [si,sj]';
    dpre = [];
    for i=[1:Nlayers1]
       dpre = [dpre; spre(i)*ones(Hpre(i),1) ];
    end
    i = Nd-length(dpre);
    dpre = [dpre; spre(Nlayers1+1)*ones(i,1) ];
    Dpre = G*dpre;
    e = (Dobs - Dpre)/sd;
    myE = e'*e;
    E1(Hi, si, sj) = myE;
    logP1(Hi, si, sj) = -myE/2;

    if( first==1 )
        Himin1=1; simin1=1; sjmin1=1;
        dmin1 = dpre;
        Dmin1 = Dpre;
        Emin1 = myE;
        first=0;
    elseif( myE<Emin1 )
        Himin1=Hi; simin1=si; sjmin1=sj;
        dmin1 = dpre;
        Dmin1 = Dpre;
        Emin1 = myE;
    end
    if( (Hi==(Htrue(1)+Htrue(2))) && (si==floor((strue(1)+strue(2))/2)) &&
(sj==strue(3)) )
        figure(10);
        subplot(2,1,1);
        plot( [1:Nd]', dpre, 'g:', 'LineWidth', 2 );
        subplot(2,1,2);
        plot( [1:NG]', Dpre, 'g:', 'LineWidth', 2 );
    end
end
end
end

P1 = exp(logP1);

figure(10);
subplot(2,1,1);
plot( [1:Nd]', dmin1, 'm:', 'LineWidth', 2 );
subplot(2,1,2);
plot( [1:NG]', Dmin1, 'm:', 'LineWidth', 2 );

TP1 = sum(P1(:));
TP2 = sum(P2(:));
TP = TP1 + TP2;
P1 = P1/TP;
P2 = P2/TP;

PE2.E1 = E1;
PE2.E2 = E2;
PE2.P1 = P1;
PE2.P2 = P2;
PE2.logP1 = log(P1);
PE2.logP2 = log(P2);

minE1 = min(E1(:));
minE2 = min(E2(:));
maxP1 = max(P1(:));
maxP2 = max(P2(:));
vol1=length(find(P1(:)>(0.1*maxP1)));
vol2=length(find(P2(:)>(0.1*maxP2)));
TP1 = sum(P1(:));
TP2 = sum(P2(:));
TP = TP1 + TP2;
fprintf('Gaussian\n\n');
fprintf('Degrees of freedom %d %d \n', nu(1), nu(2));
fprintf('Minimum Error %f %f\n', minE1, minE2 );
fprintf('Minimum Error over nu %f %f\n', minE1/nu(1), minE2/nu(2) );
fprintf('Total probability %f %f %f\n', TP1, TP2, TP1+TP2 );
fprintf('Maximum Probability %f %f\n', maxP1, maxP2 );
fprintf('Volume %d %d\n', vol1, vol2 );
fprintf('Total probability normalized by volume %f %f \n', TP1/vol1,
TP2/vol2);

figure(11);
clf;
colormap('jet');
subplot( 3, 3, 1 );
```

```matlab
imagesc( squeeze(P1(Himin1,:,:)) );
ylabel('s1');
xlabel('s2');
subplot( 3, 3, 2 );
imagesc( squeeze(P1(:,:,sjmin1)) );
ylabel('H1');
xlabel('s1');
subplot( 3, 3, 3 );
imagesc( squeeze(P1(:,simin1,:)) );
ylabel('H1');
xlabel('s2');

subplot( 3, 3, 4 );
imagesc( squeeze(P2(:,:,simin2,sjmin2,skmin2)) );
ylabel('H1');
xlabel('H2');
subplot( 3, 3, 5 );
imagesc( squeeze(P2(Himin2,Hjmin2,:,:,skmin2)) );
ylabel('s1');
xlabel('s2');
subplot( 3, 3, 6 );
imagesc( squeeze(P2(Himin2,Hjmin2,simin2,:,:)) );
ylabel('s2');
xlabel('s3');

PE2.Himin1=Himin1;
PE2.simin1=simin1;
PE2.sjmin1=sjmin1;

PE2.Himin2=Himin2;
PE2.Hjmin2=Hjmin2;
PE2.simin2=simin2;
PE2.sjmin2=sjmin2;
PE2.skmin2=skmin2;

save('PE2K_G.mat','PE2');
```

```
% MH_G_SU, Metropolis-Hastings, Gaussian Likelihood, Layer-Uniform Prior
% Note: adds covariance to output
clear all;

global PE2

load('PE2K_G.mat');

Klayers = 2;
Llayers = 10*Klayers;

Kthicknesses = 20;
Lthicknesses = 10*Kthicknesses;
sthicknesses = 2;

Kslownesses = 20;
Lslownesses = 10*Kthicknesses;
sslownesses = 2;

% define observed data
Nd = PE2.Nd;
dtrue = PE2.dtrue;

PAnorm = 0;
for i=[1:Klayers]
    configs = (Kthicknesses^i)*(Kslownesses^(i+1));
    PAnorm=PAnorm+1/configs;
end
logPA = zeros(Klayers,1);
for i=[1:Klayers]
    logPA(i)= -i*log(Kthicknesses)-(i+1)*log(Kslownesses)-log(PAnorm);
end

% probability N(i;j)
% probability of of i layers given j layers
% based on a Normal distributon with mean j and standard deviation s
K=Klayers;  % maximum number of layers
L=Llayers; % used in realization method
p = [0.75, 0.25; 0.25, 0.75];
N = makeP( p, L );
% gda_draw(N.p,'caption L');

% probability H(i;j)
% probability of of thickness i given thickness j
% based on a Normal distributon with mean j and standard deviation s
K=Kthicknesses;  % maximum number of layers
L=Lthicknesses; % used in realization method
i = [1:K]'; % all possible means
s = sthicknesses; % standard deviation;
p = zeros(K,K);
for j=[1:K]
    pa = exp(-(i-j).^2/(2*s*s))/(sqrt(2*pi)*s);  % normal distributon
centered at m0
    p(:,j)=pa;
end
H = makeP( p, L );
% gda_draw(H.p,'caption H');
% probability S(i:j)
% probability of of slowness i given slowness j
% based on a Normal distributon with mean j and standard deviation s
K=Kthicknesses;  % maximum number of layers
L=Lthicknesses; % used in realization method
i = [1:K]'; % all possible means
s = sslownesses; % standard deviation;
p = zeros(K,K);
for j=[1:K]
    pa = exp(-(i-j).^2/(2*s*s))/(sqrt(2*pi)*s);  % normal distributon
centered at m0
    p(:,j)=pa;
end
S = makeP( p, L );
% gda_draw(V.p,'caption V');

% make a random starting state
X1.Klayers = Klayers;
X1.Kthicknesses = Kthicknesses;
X1.Kslownesses = Kslownesses;
X1.Nlayers = N.r(unidrnd(N.L),floor(Klayers/2));
X1.thickness = zeros( X1.Nlayers, 1 );
for i=[1:X1.Nlayers]
    X1.thickness(i) = H.r(unidrnd(H.L),floor(Kthicknesses/2));
end
X1.slowness = zeros( X1.Nlayers + 1, 1 );
for i=[1:(X1.Nlayers+1)]
    X1.slowness(i) = H.r(unidrnd(S.L),floor(Kslownesses/2));
end

% for summary statistics
dsum  = zeros(Nd,1);
dsum2 = zeros(Nd,1);
bsum  = zeros(Nd,1);
bcsum  = zeros(Nd,1);

msum1 = zeros(3,1);
msum2 = zeros(5,1);
msum1sq = zeros(3,1);
msum2sq = zeros(5,1);
mprod1 = zeros(3,3);
mprod2 = zeros(5,5);
count1 = zeros(3,1);
count2 = zeros(5,1);

Hist1 = zeros(K,K,K);
Hist2 = zeros(K,K,K,K,K);
dimage = zeros( Kslownesses, Nd );

% Standard Metropolis-Hastings.
Nr = 10000000;
Nadopts = 0;
Hlayers = zeros(Klayers,1);
logp1max = (-1e6)*ones(Klayers,1);

for i=[1:Nr]
```

```
    % X1 is the current state; X2 is the successor state

    oldN = X1.Nlayers;

    % part 1, involving target distribution
    % A1 = p_successor/p_ccurrent;
    X2 = randomstate( X1, N, H, S );

    % the target distribution is the product of
    %    a Normal distribution in the Error with variance vard
    %    a truncated geometric distribution in the number of layers
    %       with success parameter v (this is the prior information)

    [logp1, d1, b1, bc1] = logPE2ofd( X1, Klayers, Kthicknesses );
    [logp2, d2, b2, bc2] = logPE2ofd( X2, Klayers, Kthicknesses );
    logA1 = logp2 - logp1; % no prior + logPA(X2.Nlayers) -
logPA(X1.Nlayers);

    % part 2 involving perturbing distribution
    % A2 = p_current_given_successor / p_successor_given_current;
    logpx1x2 = logcondprob( X1, X2, N, H, S );
    logpx2x1 = logcondprob( X2, X1, N, H, S );
    logA2 = logpx1x2 - logpx2x1;

    % total test parameter
    logA = logA1 + logA2;
    A = exp(logA);

    % MH test for adopting the successor
    adopt = 0;
    if( A>=1 )
        adopt=1;
    else
        Ap = random('uniform',0,1,1,1);
        if( A>Ap )
            adopt=1;
        end
    end
    if( adopt )
        X1 = X2;
        d1 = d2;
        b1 = b2;
        bc1 = bc2;
        logp1 = logp2;
        Nadopts = Nadopts+1;
    else
        % X1 = X1
        ;
    end

    % Z(i) = X1;

    % collect summary statistics
    dsum = dsum+d1;
    dsum2 = dsum2+(d1.^2);
    bsum = bsum+b1;
    bcsum = bcsum+bc1;
    Hlayers(X1.Nlayers) = Hlayers(X1.Nlayers)+1;

    if( X1.Nlayers == 1)
        H1=X1.thickness(1);
        s1=X1.slowness(1);
        s2=X1.slowness(2);

        Hist1( H1, s1, s2 ) = Hist1(H1, s1, s2)+1;
        count1(1)=count1(1)+1; msum1(1) = msum1(1) + H1; msum1sq(1) =
msum1sq(1) + H1^2;
        count1(2)=count1(2)+1; msum1(2) = msum1(2) + s1; msum1sq(2) =
msum1sq(2) + s1^2;
        count1(3)=count1(3)+1; msum1(3) = msum1(3) + s2; msum1sq(3) =
msum1sq(3) + s2^2;

        mym = [H1, s1, s2];
        for iii = [1:3]
        for jjj = [1:3]
            mprod1(iii,jjj) = mprod1(iii,jjj) + mym(iii)*mym(jjj);
        end
        end

        if( logp1 > logp1max(1) )
            logp1max(1) = logp1;
        end

    else
        H1=X1.thickness(1);
        H2=X1.thickness(2);
        s1=X1.slowness(1);
        s2=X1.slowness(2);
        s3=X1.slowness(3);
        Hist2(H1, H2, s1, s2, s3) = Hist2(H1, H2, s1, s2, s3)+1;
        count2(1)=count2(1)+1; msum2(1) = msum2(1) + H1; msum2sq(1) =
msum2sq(1) + H1^2;
        count2(2)=count2(2)+1; msum2(2) = msum2(2) + H2; msum2sq(2) =
msum2sq(2) + H2^2;
        count2(3)=count2(3)+1; msum2(3) = msum2(3) + s1; msum2sq(3) =
msum2sq(3) + s1^2;
        count2(4)=count2(4)+1; msum2(4) = msum2(4) + s2; msum2sq(4) =
msum2sq(4) + s2^2;
        count2(5)=count2(5)+1; msum2(5) = msum2(5) + s3; msum2sq(5) =
msum2sq(5) + s3^2;

        mym = [H1, H2, s1, s2, s3];
        for iii = [1:5]
        for jjj = [1:5]
            mprod2(iii,jjj) = mprod2(iii,jjj) + mym(iii)*mym(jjj);
        end
        end

        if( logp1 > logp1max(2) )
            logp1max(2) = logp1;
        end
    end

    for i=[1:Nd]
```

```
        dimage( d1(i), i ) = dimage( d1(i), i )+1;
    end

end

mean1 = msum1./count1;
mstd1 = sqrt( (count1 .* msum1sq - (msum1.^2)) ./ (count1.*(count1-1)) );
mean2 = msum2./count2;
mstd2 = sqrt( (count2 .* msum2sq - (msum2.^2)) ./ (count2.*(count2-1)) );

Cm1 = zeros(3,3);
for iii = [1:3]
for jjj = [1:3]
     Cm1(iii,jjj) = (mprod1(iii,jjj)-msum1(iii)*mean1(jjj)-
mean1(iii)*msum1(jjj)+count1(1)*mean1(iii)*mean1(jjj))/(count1(1)-1);
end
end

Cm2 = zeros(5,5);
for iii = [1:5]
for jjj = [1:5]
     Cm2(iii,jjj) = (mprod2(iii,jjj)-msum2(iii)*mean2(jjj)-
mean2(iii)*msum2(jjj)+count2(1)*mean2(iii)*mean2(jjj))/(count2(1)-1);
end
end

fprintf('Gaussian State Uniform Nadopts %d Hlayer1 %d Nr %d\n', Nadopts,
Hlayers(1), Nr );

dmean = dsum/Nr;
dvar = dsum2/Nr - dmean.^2;
dstd = sqrt( dvar );
bmean = bsum/Nr;
bcmean = bcsum/Nr;
Hlayers = Hlayers/sum(Hlayers);

% don't let std dev get small than half a grid node
Dm = 1;
Dm2 = (Dm/2)^2;
Cm1c = Cm1 - diag(diag(Cm1)) + diag(diag(Cm1).*(diag(Cm1)>=Dm2)) +
diag(Dm2*(diag(Cm1)<Dm2));
Cm2c = Cm2 - diag(diag(Cm2)) + diag(diag(Cm2).*(diag(Cm2)>=Dm2)) +
diag(Dm2*(diag(Cm2)<Dm2));

% area calculation
A = zeros(Klayers,1);
detC = [det(Cm1c), det(Cm2c)];
for i=[1:Klayers]
    j=2*i+1;
    A(i) = ((2*pi)^(j/2))*sqrt(detC(i));
end

PE2.mean1 = mean1;
PE2.mstd1 = mstd1;
PE2.Cm1 = Cm1;
PE2.mean2 = mean2;
PE2.mstd2 = mstd2;
```

```
PE2.Cm2 = Cm2;
PE2.dmean = dmean;
PE2.bmean = bmean;
PE2.bcmean = bcmean;
PE2.Hlayers = Hlayers;
PE2.Nr = Nr;
PE2.logp1max = logp1max;
PE2.Dm2 = Dm2;
PE2.Cm1c = Cm1c;
PE2.Cm2c = Cm2c;
PE2.A = A;

% prior
PA = [ 1/A(1), 1/A(2)]' / ( 1/A(1) + 1/A(2) );
PE2.PA = PA;
PE2.logPA = log(PA);

% plot
figure(1);
clf;
subplot(3,1,1);
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
hold on;
xlabel('z');
ylabel('s(z)');
plot( [1:Nd]', dtrue, 'r-', 'LineWidth', 3 );
plot( [1:Nd]', dmean+dstd, 'k:', 'LineWidth', 2 );
plot( [1:Nd]', dmean, 'k-', 'LineWidth', 2 );
plot( [1:Nd]', dmean-dstd, 'k:', 'LineWidth', 2 );
subplot(3,1,2);
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
hold on;
xlabel('z');
ylabel('P(b)');
plot( [1:Nd]', bmean, 'k-', 'LineWidth', 2 );
subplot(3,1,3);
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
hold on;
xlabel('z');
ylabel('P(b signed)');
plot( [1:Nd]', bcmean, 'k-', 'LineWidth', 2 );

figure(2);
clf;
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
hold on;
xlabel('layers');
ylabel('P(layers)');
plot( [1:Klayers]', Hlayers, 'k-', 'LineWidth', 3 );

figure(3);
clf;
colormap('jet');
```

```
subplot( 3, 3, 1 );
imagesc( squeeze(Hist1(PE2.Himin1,:,:)) );
ylabel('s1');
xlabel('s2');
subplot( 3, 3, 2 );
imagesc( squeeze(Hist1(:,:,PE2.sjmin1)) );
ylabel('H1');
xlabel('s1');
subplot( 3, 3, 3 );
imagesc( squeeze(Hist1(:,PE2.simin1,:)) );
ylabel('H1');
xlabel('s2');

subplot( 3, 3, 4 );
imagesc( squeeze(Hist2(:,:,PE2.simin2,PE2.sjmin2,PE2.skmin2)) );
ylabel('H1');
xlabel('H2');
subplot( 3, 3, 5 );
imagesc( squeeze(Hist2(PE2.Himin2,PE2.Hjmin2,:,:,PE2.skmin2)) );
ylabel('s1');
xlabel('s2');
subplot( 3, 3, 6 );
imagesc( squeeze(Hist2(PE2.Himin2,PE2.Hjmin2,PE2.simin2,:,:)) );
ylabel('s2');
xlabel('s3');

figure(4);
clf;
colormap('jet');
imagesc( dimage );
ylabel('s(z)');
xlabel('z');

save('PE2K_G_C.mat','PE2');  % C for contains covariance
```