MCMC Assignment 1

(1) Write a simple MCMC routine (preferably in C or Python) to produce N draws $\{x_i\}$ from the scaled/shifted Student-t distribution

$$\pi(x|\nu,\sigma,\mu) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\,\sigma}\left(1+\frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right)^{-(\nu+1)/2}$$

Note that the parameters ν, σ, μ are fixed here, and they define the properties of the distribution. As a concrete example, set $\nu = 3, \mu = 1, \sigma = 1$. For the proposal distribution, use

$$q(y|x) = \frac{1}{\sqrt{2\pi}\,\alpha} e^{-(y-x)^2/2\alpha^2}$$

Note that this proposal distribution is symmetric, $q(y|x) = q(x|y)$, and so the proposal densities will cancel in the Metropolis-Hastings ratio. If coding in C you can use the GNU scientific library which provides the gsl_ran_gaussian and gsl_ran_gaussian_pdf functions.
(a) Run your code trying four different jump size scalings, $\alpha = 0.01$, $\alpha = 0.1$, $\alpha = 1.0$ and $\alpha = 10.0$. Visually inspect the chains. Which one appears to be exploring the distribution most effectively?
(b) Have your code compute the acceptance fraction for the proposed jumps (*i.e.* the fraction of the proposed jumps that are accepted). Does a high acceptance rate necessarily mean efficient exploration of the posterior distribution?
(c) Using the same 4 sets of jumps sizes, run your code for $N = 10,000$ iterations and use the output to produce histograms of the $\{x_i\}$. Plot them against the target distribution $\pi(x|\nu,\sigma,\mu)$. Which distribution looks the best?
(d) Using much longer runs ($N = 1,000,000$ or so), compute the auto-correlation length of the chains for the 4 different jumps sizes. Which jump size produces the shortest correlation length? The sample auto-correlation is defined:

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)}$$

where

$$\gamma(h) = \frac{1}{N-h}\sum_{i=1}^{N-h}(x_{i+h}-\bar{x})(x_i-\bar{x})$$

Note that $\gamma(0)$ is just the variance. Here $\bar{x}$ denotes the mean. The auto-correlation length is defined (roughly) as the h at which $\rho(h)$ drops to ~ 0.01.
(e) Compute the Fisher information matrix (here just a 1 x 1 matrix!)

$$\Gamma_{xx} = -\partial_x\partial_x \ln \pi(x)|_{\max}$$

and use this to select the appropriate size for α in the proposal. Test this choice out for $\nu = 3, \mu = 1, \sigma = 1$ and $\nu = 10, \mu = -2, \sigma = 7$.

MCMC Assignment 2

(1) Use the best mixing version of the code from Assignment 1 to produce a large number of independent samples from the scaled/shifted Student-t distribution with $\nu = 3, \mu = 1, \sigma = 1$. The goal here is going to be trying to infer the parameters ν, μ, σ that produced the data, and to look at model selection.
(a) In this case, the samples $\{x_i\}$ are the data, d, and the model parameters are ν, μ, σ. We wish to produce posterior distribution functions for these parameters. The likelihood is given by

$$p(d|\nu,\mu,\sigma) = \prod_{i=1}^{N} \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\,\sigma}\left(1+\frac{1}{\nu}\left(\frac{x_i-\mu}{\sigma}\right)^2\right)^{-(\nu+1)/2}$$

Take the priors to be uniform in the ranges $\nu \in [0.1, 10]$, $\mu \in [-5, 5]$, $\sigma \in [0.1, 10]$. For the proposal distribution use a multi-variate Gaussian defined by the Fisher information matrix

$$q(y|x) = \frac{1}{(2\pi)^{N/2}\det\Gamma^{1/2}} e^{-\frac{1}{2}\Gamma_{ij}(y^i-x^i)(y^j-x^j)}$$

The Fisher matrix is defined:

$$\Gamma_{ij} = -\langle \partial_i \partial_j \ln p(d|\nu,\mu,\sigma)\rangle|_{\max}$$

Note that here the i, j refer to the model parameters, so you are taking derivatives of the likelihood with respect to ν, μ, σ. The "max" refers to the theoretical maximum of the likelihood, which corresponds to the injected values for the parameters. Note that the Fisher matrix will depend on the size of the data set N. Larger data sets should produce larger Fisher matrix entries, and hence smaller uncertainties in the recovered parameters. I suggest proposing jumps along the eigen-directions of the Fisher matrix, scaled by the inverse of the square root of the eigenvalues. In other words, have the code select at random which eigen-direction to jump along, then propose a jump along that direction. Once again, keep track of the acceptance rates and the auto-correlation length. Produce marginalized posterior distribution functions for the parameters. Also look at 2-d scatter plots of ν versus σ etc to get a sense of the parameter correlations. Do they agree with the predictions from the Fisher matrix? Try running with $N = 100$ and $N = 1000$ data samples. Check that your proposal distributions are obeying detailed balance by setting the likelihood equal to a constant value, $p(d|\nu,\mu,\sigma) = \text{constant}$ and making sure that the posterior distributions for the parameters recovers the prior distributions on the parameters.

Note: The Python packages https://samreay.github.io/ChainConsumer/ and https://corner.readthedocs.io/en/latest/ are useful tools for making histograms and corner plots.

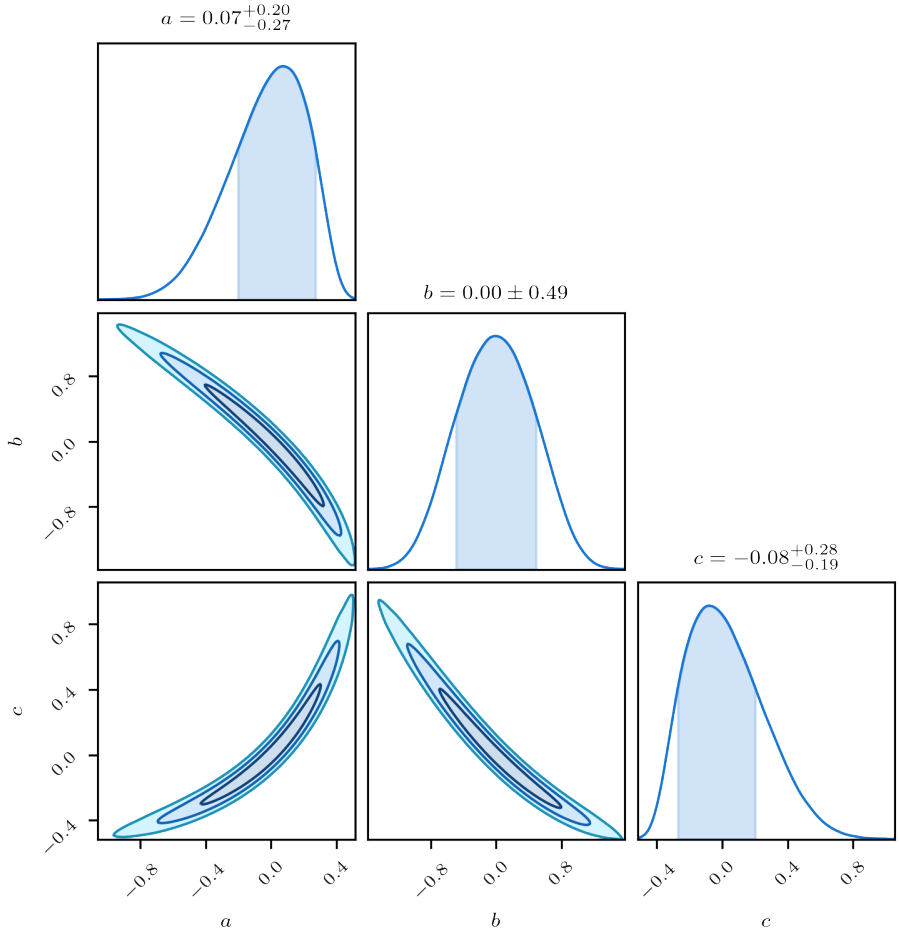
MCMC Assignment 3

This assignment deals with situations where some of the parameters are highly correlated. Jumps along eigenvectors of the Fisher information matrix do not always work in these situations. A powerful method for handling correlations is called *differential evolution*. The method was first introduced by C. J. F. Ter Braak, [Stat. Comput. **16**, 239 (2006)] and developed further in C. J. F. Ter Braak & J. A. Vrugt, [Stat. Comput. **18**, 435 (2008)]. The DE algorithm needs to be used in concert with other proposal distributions. For example, you could use local Gaussian jumps such as were used in Assignment 1, and/or jumps along eigendirections of the Fisher matrix. The Metropolis-Hastings updates are then made using a randomly selected type of proposal distribution. For example, 50% of the time it might try a Fisher matrix based jump, then try a DE jump the other 50%. Any number of proposals can be added to the cocktail. The mix is a matter of taste. Try different mixes and see what you like (For a Margarita I like 2 shots of Petrón Reposado, 1.5 shots of fresh lime, 1 shot of Drillaud Triple Sec and a dash of Agave syrup).

The DE updates require a history array $\mathbf{h}_i$. This array is a collection of past samples. Typically $N \sim 1000$ or so past samples is sufficient. The array is initialized with draws from the prior. Samples are added to the history array after every $M \sim 10$ iterations of the MCMC. A counter i keeps track of how many samples have been added. The new sample added is $\mathbf{h}_{i(\mathrm{mod}N}$. That is, when i reaches N the $\mathbf{h}_0$ entry gets replaced. The history array thus gets fully re-freshed every NM iterations. The DE proposal is made as follows:
Draw two samples from the history array: $j \sim U[0, N-1]$, $k \sim U[0, N-1]$. Propose the move from $\mathbf{x}$ to $\mathbf{y}$:

$$\mathbf{y} = \mathbf{x} + \gamma(\mathbf{h_j} - \mathbf{h_k})\,. \tag{1}$$

Here we draw γ from a Gaussian of width $2.38/\sqrt{2d}$ for 90% of the DE updates, where d is number of parameters, and set $\gamma = 1$ for the rest. The proposal is symmetric, so the proposal densities cancel in the Metropolis-Hastings ratio. The DE proposal takes a while to kick in - the improved mixing starts to be seen after a few times NM iterations of the MCMC. Technically, the DE proposal violates detailed balance since it uses past samples in the draw. But it can be shown that the DE proposal is asymptotically Markovian. In other words, run for long enough and it will be ok.



Write a MCMC to draw samples from the target distribution

$$\pi(a, b, c) = A\, e^{-\left(\alpha(a+b+c)^2+\alpha(a-c+ac)^2+a^2+b^2+c^2\right)}$$

with $\alpha = 10^4$. A corner plot showing samples drawn from this distribution is displayed above. Start with a basic MCMC that just uses independent Gaussian updates in each parameter. Try and find standard deviations σ_a, σ_b, σ_c for this proposal that yield decent exploration of the target. Next, see if you can get a multivariate Fisher matrix based proposal to work (you might run into some issues with singular eigenvalues, see if you can figure out a way around this problem). Finally, taking the best MCMC algorithm you were able to get working with independent Gaussian and/or Fisher jumps, add DE updates to the mix and see what happens. Look at trace plots of the parameters a, b, c. Can you see when the DE proposal kicks in? Compare the autocorrelation length of the chains with and without the DE proposal.

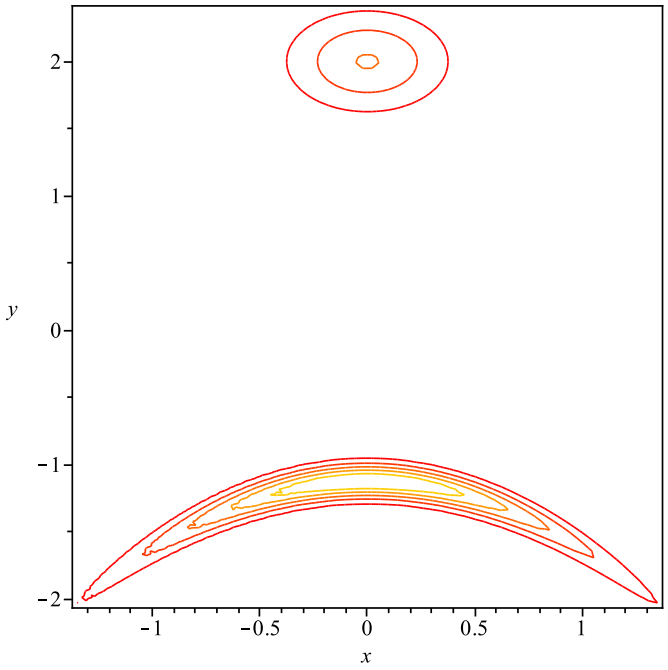
MCMC Assignment 4

This assignment deals with multi-modal posterior distributions. Various techniques for exploring multi-model posteriors will be compared. One will be parallel tempering with Fisher jumps, the other will be Langevin MCMC (aka MALA), which is closely related to the Hamiltonian MCMC algorithm.

The target distribution that we wish to sample from has the form

$$\pi(x, y) = \frac{16}{3\pi}\left(e^{-x^2-(9+4x^2+8y)^2} + \frac{1}{2}e^{-8x^2-8(y-2)^2}\right)$$

The challenge here is that the target distribution is multi-modal, and in addition, one of the modes is "banana shaped", as can be seen in the contour plot below



Parallel Tempering

Exploring multi-modal posteriors can be a serious challenge for many MCMC algorithms, which tend to get stuck exploring a single mode of the posterior. While the Metropolis-Hastings transition probability allows for "downhill steps", it is highly unlikely that a standard MCMC chain will "walk" all the way down from one mode, cross the valley and climb up a second mode. There are a host of techniques that have been developed to tackle this type of problem. One that works fairly well is parallel tempering, which is a form of population MCMC (PTMCMC is also known as the replica exchange method and Metropolis Coupled MCMC or MCMCMC). The idea is that while

exploring a multi-modal posterior $\pi(\vec{x}|d) = p(d|\vec{x})p(\vec{x})$ may be challenging when the likelihood $p(d|\vec{x})$ has multiple peaks, it is much easier to explore the modified posterior $\pi_T(\vec{x}|d) = p(d|\vec{x})^{1/T}p(\vec{x})$ in the limit of large "temperature" T. The idea is to run multiple chains in parallel, each with a different temperature T, and by allowing exchanges between the chains we effectively end up using the high temperature chains as proposal densities for the low temperature chains. The high temperature chains freely explore the entire space and locate all the modes, while the low temperature chains map out the peaks in greater detail. Only samples from the $T = 1$ chain correspond to samples from the target distribution. Samples from the other chains are discarded. If properly implemented, a PTMCMC algorithm with M parallel chains run for N iterations each will converge to the target distribution far more quickly than a single chain run for $M \times N$ iterations.

The key to getting a PTMCMC algorithm to work efficiently is selecting the spacings of the temperature ladder. If the spacing is too large swaps between chains will never be accepted. If the spacing is too small it takes a huge number of chains to reach a high enough maximum temperature. The hottest chain should have a temperature $T_{\max}$ large enough to ensure that this chain moves freely around the space. If $\Delta \log p(d|\vec{x})$ is the difference in the log likelihood of the highest peak and the lowest valley, then $T_{\max} \sim \Delta \log p(d|\vec{x})/25$. In gravitational wave applications a good choice is $T_{\max} \sim \mathrm{SNR}^2/25$, where SNR is the signal to noise ratio of the signal. This choice means that the hottest chain sees a signal with an effective SNR of ~ 5.

A good choice of temperature spacing for most applications is given by the geometric progression $T_{i+1} = cT_i$ with $c \sim 1.2 \to 2$. Geometric spicing is known to be suboptimal in many cases, but there are no general theorems to provide guidance on better choices. Once the maximum temperature and temperature spacing have been chosen the total number of chains M is determined, and we are ready to proceed. The PTMCMC algorithm works by independently evolving each of the M chains using some standard MCMC algorithm to explore the $= \pi_{T_i}(\vec{x}|d)$, and then once every so often "swaps" are proposed with an inter-chain transition probability

$$H = \min\left(1, \frac{\pi_{T_i}(\vec{x}_{i+1}|d)\pi_{T_{i+1}}(\vec{x}_i|d)}{\pi_{T_i}(\vec{x}_i|d)\pi_{T_{i+1}}(\vec{x}_{i+1}|d)}\right) \tag{2}$$

If the swap is accepted, the parameters of the chain at temperature T_i get swapped with those at temperature T_{i+1} (not the entire past history, just the current location). The regular MCMC moves at each temperature then proceed for a while until another inter-chain swap is proposed. Selecting the right ratio of within temperature proposals to chain swap proposals is more art than science. Typically a 3:1 ratio works pretty well.

Compare the performance of a single chain MCMC sampler with a multi-chain PTMCMC sampler. For simplicity use Fisher matrix proposals for the within-temperature steps. Start the chains out at the locations $(x, y) = (0, 0)$, $(x, y) = (1, 2)$ and $(x, y) = (0, -2)$ and produce plots showing the path taken by the chain(s) for the first 50 iterations for each sampler. For the PTMCMC sampler show the paths for both the $T = 1$ and the $T_{\rm max}$ chains. What did you choose for $T_{\rm max}$? Run the PTMCMC sampler for a large number of iterations. Produce 2-d histograms of the recovered posterior distributions for the $T = 1$ and the $T_{\rm max}$ chains. Play around a little with the choice of the geometric scaling parameter c and the maximum temperature $T_{\rm max}$. Produce a plot of the inter-chain swap acceptance fraction as a function of c with c between 1 and 2 in increments of 0.1.

Langevin MCMC

In the first two assignments your learnt that Gaussian scaled jumps along randomly selected eigendirections $\hat{e}_{(i)}$ of the Fisher information matrix, scaled by the inverse square root of the corresponding eigenvalue λ_i were quite effective:

$$\vec{y} = \vec{x} + \frac{\delta}{\sqrt{\lambda_i}}\,\hat{e}_{(i)} \tag{3}$$

with δ a unit Normal deviate. There is actually some interesting theory behind all this - in 1945 Rao showed that the "expected Fisher information" matrix EFI defines a natural Riemannian metric on spaces defined by parameterized probability density functions. The likelihood defined in Assignment 2 is an example of a parameterized density function $p(d|\vec{\theta})$, where d is the data and $\vec{\theta}$ are the parameters. The EFI is then defined as

$$\begin{aligned} \Gamma_{ij} &= -\langle \partial_i \partial_j \log p(d|\vec{\theta})\rangle = \mathrm{cov}(\partial_i \log p(d|\vec{\theta}), \partial_j \log p(d|\vec{\theta})) \\ &= \langle \partial_i \log p(d|\vec{\theta}) \partial_j \log p(d|\vec{\theta})\rangle - \langle \partial_i \log p(d|\vec{\theta})\rangle \langle \partial_j \log p(d|\vec{\theta})\rangle \end{aligned} \tag{4}$$

In the mathematical literature you will see this written as

$$\Gamma = -E_{\boldsymbol{\theta}|d}\left[\frac{\partial^2}{\partial \boldsymbol{\theta}^2} \log p(d|\boldsymbol{\theta})\right] = \mathrm{cov}\left[\frac{\partial}{\partial \boldsymbol{\theta}} \log p(d|\boldsymbol{\theta})\right] . \tag{5}$$

Here the EFI is evaluated at the current location $\vec{\theta}$, and the expectation is taken over the data distribution. In frequentists statistics it is more common to see the "observed Fisher information" matrix OFI,

$$\Gamma = -\frac{\partial^2}{\partial \boldsymbol{\theta}^2} \log p(d|\boldsymbol{\theta})\big|_{\boldsymbol{\theta}_{ML}} , \tag{6}$$

where the matrix is evaluated at the maximum likelihood estimate for the parameters. In the current assignment we don't have any data per se, so the Fisher matrix is taken to be the negative Hessian of the log probability:

$$\Gamma_{ij} = -\partial_i \partial_j \log \pi(\vec{x}) \,. \tag{7}$$

Note that Γ is not constant, so additional care has to be taken since the eigenvalues and vectors will vary from location to location, leading to more complicated expressions in the Hasting's ratio since $q(\vec{x}|\vec{y}) \neq q(\vec{y}|\vec{x})$.

Taking the idea of the Fisher matrix as a metric one step further leads to the "Riemann Manifold Langevin" MCMC algorithm. This approach is very good at finding local maxima of complicated multi-modal posteriors. The basic idea is to use gradient information to help guide the jumps. Rather than use the full RML algorithm, I'll have you try the Manifold Metropolis Adjusted Langevin Algorithm (MMALA) without the corrections needed to handle non-constant curvature. The proposals take the form

$$\vec{y} = \vec{x} + \frac{\epsilon^2}{2}\mathbf{C}(\vec{x})\nabla \log \pi(\vec{x}) + \epsilon\sqrt{\mathbf{C}(\vec{x})}\vec{z}\,, \tag{8}$$

where the covariance matrix $\mathbf{C}(\vec{x})$ is the inverse of the Fisher matrix $\mathbf{\Gamma}(\vec{x})$ define in (7). The square root of covariance matrix is computed using a Cholesky decomposition. The vector $\vec{z}$ is drawn from a multivariate Normal distribution with zero mean and covariance matrix $\mathbf{I}$, such that $\vec{z} \sim \mathcal{N}(\vec{z}|\mathbf{0}, \mathbf{I})$. Defining $\mathbf{S}(\vec{x}) = \sqrt{\mathbf{C}(\vec{x})}$, the component form of (8) looks like

$$y^i = x^i + \frac{\epsilon^2}{2}C^{ij}\nabla_j \log \pi(\vec{x}) + \epsilon S^{ij} z_j \,, \tag{9}$$

The proposal density corresponds to a multi-variate Normal distribution with mean $\boldsymbol{\mu}(\vec{x}, \epsilon) = \vec{x} + \frac{\epsilon^2}{2}\mathbf{C}(\vec{x})\nabla \log \pi(\vec{x})$ and covariance matrix $\mathbf{C}(\vec{x})$. Note that this is precisely the same as the "Fisher matrix proposals" we have used before, except that now the mean has been shifted by the gradient term. In practice we have found it convenient to work with an eigenbasis for the Fisher matrix (just as we did with the Fisher jumps). This saves us from having to perform an explicit Cholesky factorization. The MALA then looks like

$$\vec{y} = \vec{x} + \hat{e}_{(i)}\,\frac{\epsilon^2 \hat{e}_{(i)} \cdot \nabla \log \pi}{2\lambda_{(i)}} + \frac{\epsilon\delta}{\sqrt{\lambda_i}}\,\hat{e}_{(i)} \tag{10}$$

Here we have the gradient of the log posterior evaluated along the eigendirection - in other words, the directional derivative. The proposal distribution is

$$q(\vec{y}|\vec{x}) = \mathcal{N}\left\{\vec{x}|\boldsymbol{\mu}(\vec{x}, \epsilon), \epsilon^2\mathbf{C}(\vec{x})\right\} \,. \tag{11}$$

Be careful in your Hastings ratios since $q(\vec{x}|\vec{y}) \neq q(\vec{y}|\vec{x})$. In most applications, values of ϵ around unity have worked well, so just set $\epsilon = 1$.

Compare the performance of MCMC samplers that use the proposals based on (3) and (10). Start the chains out at the locations $(x, y) = (0, 0)$, $(x, y) = (1, 2)$ and $(x, y) = (0, -2)$ and produce plots showing the path taken by the chain for the first 50 iterations for each sampler. Which one converges on a maxima first? Do they always find the same maxima? Let the chains run for a large number of iterations and produce corner plots of the chains. Did either of the algorithms manage to map out both modes?

MCMC Assignment 5

In the previous assignments we used a variety of MCMC techniques to sample from distributions and produce estimates for the posterior distributions. We now go one step further and consider a space of possible models and use a "trans-model" MCMC to explore the model space. This approach is often called "trans-dimensional MCMC" since the models may have different dimensions. The standard algorithm for implementing the model exploration is the "Reversible Jump MCMC" or RJMCMC. The RJMCMC automatically implements Bayesian model selection. The fraction of the total iterations that algorithm spends in each model yields the probability that each model is correct. If we have just two models the ratio of these probabilities is the Bayesian odds ratio.

(a) Here we return to consider the data sets considered in Assignment 2, which were generated from the Student-t distribution, and introduce an alternative Gaussian model for the data with parameters α, β and likelihood

$$p(d|\alpha,\beta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\,\alpha} e^{-(x_i-\beta)^2/2\alpha^2}$$

We know that this model is "wrong" since the data samples were drawn from a different distribution, but the Gaussian model has less parameters, and we might find that it is favored over the Student-t model in certain situations.

Use uniform priors with the ranges $\alpha \in [0.1, 10]$, $\beta \in [-5, 5]$. Note that the parameter β should correspond to μ, and to a fair approximation, the parameter α should correspond to the parameter combination $\sigma\sqrt{\nu/(\nu+1)}$. Use a Fisher matrix based proposal distribution for the Gaussian model. Produce marginalized posterior distributions for α, β using data sets with $N = 100$ and $N = 1000$ samples (the same data sets used in part a of Assignment 2). Is there anything that stands out in the distributions that gives you any hint that you might be using the wrong model to analyze the data?

(b) Construct a Reversible Jump MCMC that jumps between the student-t and the Gaussian model. I'll leave you to think up what to use as the proposal distribution for the transdimensional jumps. These are not nested models (they share no common parameters), but we know that there is some similarity between the parameters in the two models (the mean and the variance). This insight should help in designing effective proposals. Keep track of the acceptance rates of both the trans-dimensional proposals (those that propose jumps from model 1 to model 2), and the within model proposals. Check that your trans-dimensional proposals satisfy detailed balance by running with the two likelihoods set equal to the same constant value. In this case the Bayes factors should be unity. While it requires some extra work, it is a very good idea to use parallel tempering in concert with the RJMCMC algorithm. The chain exchanges are capable of changing the model held in the the $T = 1$ chain. Indeed, we often find that it is the PT moves that produce most of the between dimension exchanges. It is really hard to get good inter-model acceptance rates using just RJMCMC moves. Once you have the code running efficiently, produce a plot of the Bayes factor as a function of the size of the data set N

for data generated from the student-t with $\nu = 3, \mu = 1, \sigma = 1$. At what N does the (true) student-t model model become favored relative to the (incorrect, yet simpler) Gaussian model? Selecting a fixed $N = 100$, use your best code from assignment 1 to generate samples with $\mu = 0$ and $\sigma = 1$ for integer values of ν from 1 and 10. Plot the Bayes factor as a function of ν. At which ν does the student-t model become preferred?

MCMC Assignment 6

The most important feature of the `BayesWave` algorithm, and the global fit algorithm we use to resolve thousands of galactic binaries in simulated LISA data, is trans dimensional modeling. In both instances the dimension of the model is not fixed. Wavelets or galactic binaries can be added or removed from the model. The data determines the model with the optimal number of parameters. Here we will use a trans-dimensional MCMC to fit a curve to some simulated data. To keep things simple the noise that is added to the curve will have fixed (yet unknown) variance σ^2 and zero mean, and the noise in each data sample with be uncorrelated. In other words, the noise is i.i.d. $\mathcal{N}(0, \sigma)$. Start by writing a code to produce simulated data of the form

$$d(t_i) = y_M(t_i) + n(t_i)$$

where $t_i = idt$ with $i \in [0, N-1]$ and $dt = 1$. Here $y_M(t_i)$ is a degree M polynomial and the $n(t_i)$ are drawn from a Gaussian distribution with zero mean and variance $\sigma^2 = 1$. We can generate a random degree M polynomial as

$$y_M(t_i) = \sum_{j=0}^{M} \alpha_M \left(\frac{t_i}{Ndt}\right)^j$$

with the α_M drawn from a Gaussian distribution with zero mean and variance β^2. You can generate a variety of different data sets. For example, data set A might have $N = 20$ data points with a polynomial of degree $M = 4$ generated with $\beta = 0.5$, and data set B might have $N = 100$ data points with a polynomial of degree $M = 10$ generated with $\beta = 0.1$. Make plots of your two data sets showing $y_M(t_i)$ superimposed on $d(t_i)$.

The model we are going to use is a sum of Chebyshev polynomials of the first kind $T_n(x)$. Note that the $T_n(x)$ are define over the interval $x \in [-1, 1]$, so you have to map your data, which runs from $t = 0..(N-1)dt$, into the interval $[-1, 1]$. Our models for the data are of the form

$$z_P(x) = \sum_{k=0}^{P} a_k T_k(x)$$

The likelihood is

$$p(\mathbf{d}|P, \mathbf{a}, \bar{\sigma}) = \frac{1}{(2\pi\bar{\sigma}^2)^{N/2}} \prod_{i=0}^{N-1} e^{(d(x_i)-z_P(x_i))^2/(2\bar{\sigma}^2)}$$

I'll leave it up to you to set priors on the parameters $P, \mathbf{a}, \bar{\sigma}$. Your task is to put together a trans-dimensional RJMCMC algorithm to simultaneously explore the model dimension P and the model parameters $\mathbf{a}, \bar{\sigma}$. The choice of proposals *etc* are up to you. Parallel tempering will help with convergence as always. Make plots of the model dimension posterior for data sets A and B, and plots of the $\bar{\sigma}$ posterior. It makes less sense to make posterior

plots of the $\mathbf{a}$ as the number of them changes with dimension. Instead, make "Bayesograms" of the kind that we use for waveform reconstructions by making histograms of $z_P(t_i)$ at each i and displaying them as a color map. Compare the median and mean of the $z_P(t_i)$ to the original simulated function $y_M(t_i)$ and verify that the original function is recovered within the 90% credible interval at each time t_i.