Report on MCMC algorithms

Nikolaos Tsikouras

Sampling from a Two–Dimensional, Highly Correlated Function

We are going to sample from the following density on $\mathbb{R}^2$,

$$
\begin{aligned}
f(x,y) &= kexp\left\{-\frac{x^2}{100} - \left(y + \frac{3}{100}x^2 - 3\right)^2\right\} \\
&= kexp\left\{-\frac{x^2}{100}\right\} exp\left\{-\left(y + \frac{3}{100}x^2 - 3\right)^2\right\} \\
&= kexp\left\{-\frac{1}{2}\frac{(x-0)^2}{\sqrt{50}^2}\right\} exp\left\{-\frac{1}{2}2\left(y - (3 - \frac{3}{100}x^2)\right)^2\right\}. \qquad (1)
\end{aligned}
$$

Metropolis–Hastings Algorithm

The MCMC algorithm we are going to use for is a Metropolis-Hastings algorithm with a Multivariate Normal proposal with dependent marginal distributions. We are picking this specific distribution because first, it is very simple to random from and additionally has a simple implementation (it cancels out from the acceptance probability). Furthermore, it generates values covering the whole space, $\mathbb{R}^2$, and by looking at Equation 1 we can see that the target density $f(x,y)$ can be written as $f(x,y) = g(x)h(y|x)$, with $g(\cdot), h(\cdot)$ being density functions of two normal distributions. Note that the former is a $N(0,50)$ distribution while the latter depends on x, thus justifying the reason of using a correlation term $\rho \neq 0$ for the covariance matrix Σ. We provide a pseudocode for a Metropolis-Hastings algorithm below.

Algorithm 1 Metropolis-Hastings algorithm

```
Input: Target density $f$; starting value $(x^{(0)}, y^{(0)})$; $2 \times 2$ positive definite covariance matrix $\Sigma$ of
Multivariate Normal distribution; number of iterations $(n)$.
for $t = 1, 2, \ldots, n$ do
    Let $Y^t = (x^{(t)}, y^{(t)}) \sim N((\mu_1, \mu_2)^T, \Sigma)$
    Let $U \sim U(0,1)$
    if $U \leq min\left(\frac{f(Y^t)q(Y^t, X^{t-1})}{f(X^{t-1})q(X^{t-1}, Y^t)}, 1\right)$ then
        $X^t = Y^t$
    else
        $X^t = X^{t-1}$
Output: A 2-dimensional Markov Chain $\{(x^{(t)}, y^{(t)})\}, t = 1, 2, \ldots$.
```

We tried several values for $\sigma_1^2, \sigma_2^2, \rho$ and initial starting value $(x^{(0)}, y^{(0)})$ to see how the algorithm performs. The results were never good enough and to illustrate this we are going to compare one of the final cases we tried with the improved values we found using the following method.

The way we picked the starting values for $\sigma_1^2, \sigma_2^2, \rho$ is the following. Instead of randomly picking which initial values would perform better we ran a loop 100 times that would eventually allow the data to answer this question for us. For the first iteration we randomly picked starting values for these 3 quantities then for each iteration we generated 5 random starting points (between -15 and 15) and ran a Metropolis-Hastings algorithm for 10000 iterations. After that we updated our values for $\sigma_1^2, \sigma_2^2, \rho$

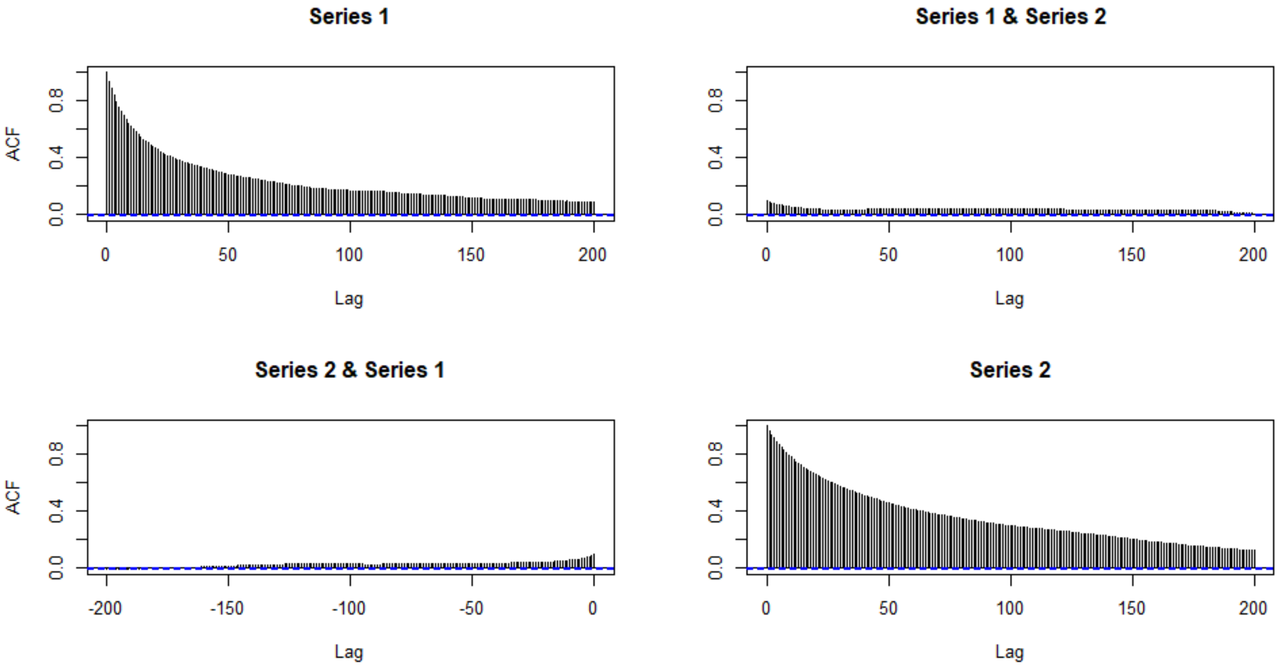

Figure 1: Autocorrelation plot for the worse model. Top left is the plot for X while bottom right is the plot for Y.

	"Bad" Model	"Good" Model
Starting values	(2.5,2.5)	(2.5,2.5)
Number of Iterations	10^5	10^5
Standard Deviation X	7.071	7.348
Standard Deviation Y	3	2.097
Covariance	2.7	-1.2
Effective Size X	1364.879	2403.629
Effective Size Y	738.698	1437.856

Table 1: Table showing the main characteristics for the two models.

based on the generated Markov Chains (combining the values for the 5 chains we found the variances and the correlation of the generated samples). We stored the current value and continued to the next iteration. After some hours we received our results, i.e. 100 pairs of variances and 100 correlations. Finally we found the mean of these values and we got the following results $\sigma_x^2 = 54.007$ (very close to the theoretical Variance, 50) with a standard deviation of 0.984, $\sigma_y^2 = 4.4$ with a standard deviation of 1.03 and covariance $= -1.201$ with a standard deviation of 6.649.

Although the standard deviations are large we still get an improved model. The details of the "bad" and improved models (starting value, number of iterations, Effective Sample Size ,etc.) can be found in Table 1. The autocorrelation plots of the "bad" MCMC can be seen in Figure 1 and the autocorrelation plot for the better MCMC can be seen in Figure 2. It is clear that the better model we used has better mixing, i.e. a better autocorrelation plot since the ACF has a steeper decrease and the Effective Sample Size is higher. Thus for the rest of this exercise we will be using a Multivariate Normal proposal with $\Sigma = \begin{pmatrix} 7.348^2 & -1.2 \\ -1.2 & 2.097^2 \end{pmatrix}$.

Although this MCMC chain is better than everything we tried, it is also not mixing well enough as we can see from the top right, bottom left panels in Figure 2. Even at Lag 200 the ACF is still not inside the threshold. Additionally, considering we had 10^5 iterations the Effective Sample Size is very small for both X and Y, 2403.629 and 1437.856, respectively. Thus, we can conclude that the mixing is poor.

From Figure 3 we can conclude that the chain has reached convergence since for all five pairs of starting values (they were picked at random) and initial Σ, (also randomly picked) it is clear that the

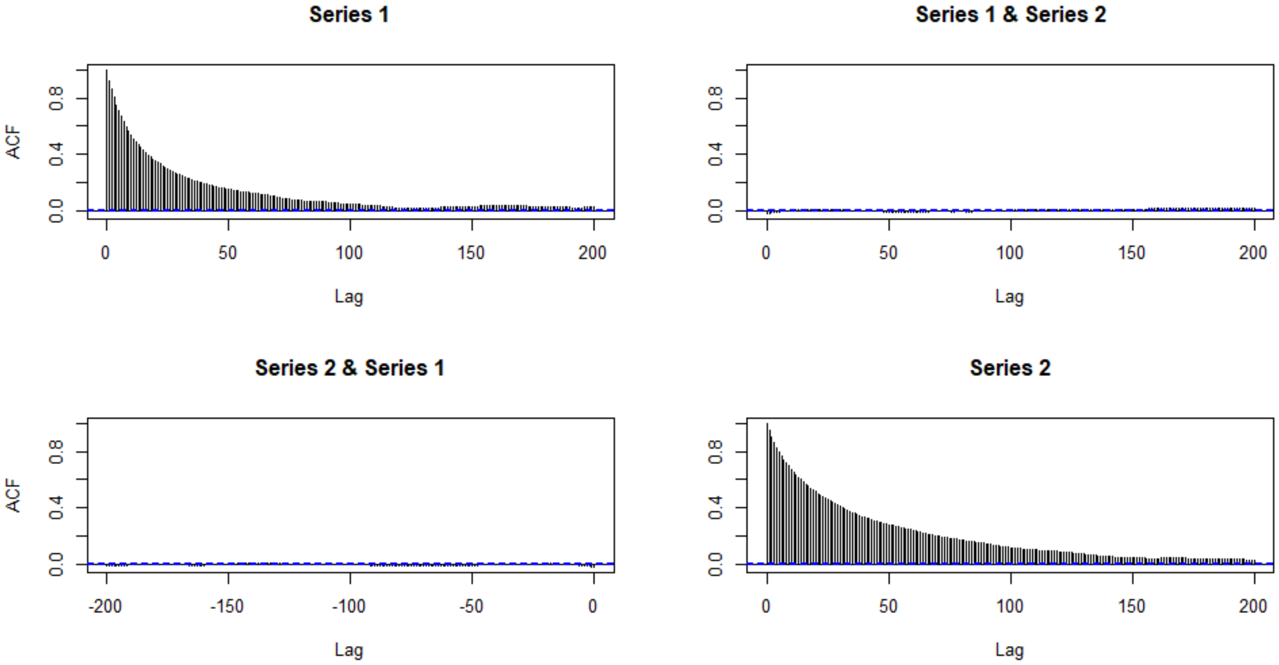

Figure 2: Autocorrelation plot for the better model. Top left is the plot for X while bottom right is the plot for Y.

respective chains seem to be in the same area. We also computed the Gelman Rubin diagnostic and obtained a point estimate and an upper confidence interval equal to 1 for each of the two parameters. These two diagnostics both suggest that the MCMC algorithm has reached convergence. We have to note here that we can not be certain that this is true, these are just positive indications.

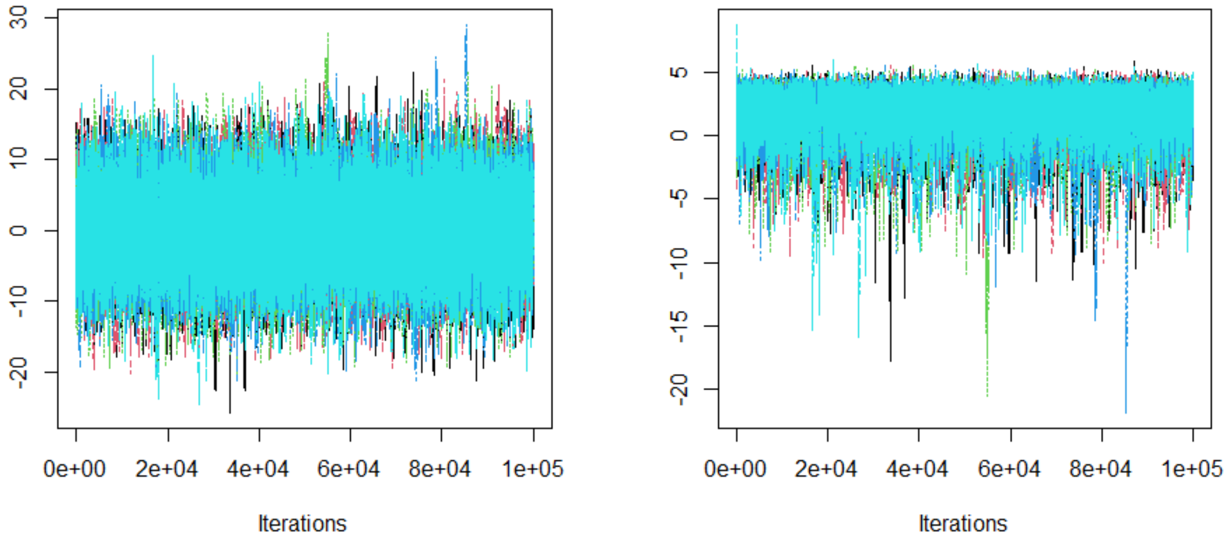

Figure 3: Traceplots of the MCMC chain for different starting values. The left traceplot is for X while the right traceplot is for Y.

Estimating Expectations

The expectations required here can be estimated from the Ergodic Theorem (assuming all the assumptions hold) from an output $(x^{(t)}, y^{(t)})_t$ as follows,

$$E(X) = \int xf(x)dx \approx \frac{1}{n - n_0} \sum_{t=n_0+1}^{n} x^{(t)},$$

where n_0 is the burn-in period. By considering a burn-in period of 1000 we obtain the following $E(X) \approx 0.0862$, $E(Y) \approx 1.5722$.

Adaptive Proposal Algorithm

We observed from Equation 1 that X, Y seem to be correlated. Thus, from [1] it makes sense that we were struggling to tune our proposal. Haario et al. [1] proposed a new method in order to avoid the difficulties of tuning a Metropolis-Hastings algorithm, the Adaptive Proposal (AP). The main idea of the method is that instead of having a fixed covariance matrix we could update the matrix every H steps. For our example, our target distribution $f(x, y)$ is a d=2-dimensional distribution. Suppose we have sampled at least H points at iteration t $\{X_1, \dots, X_{t-H+1}, \dots, X_t\}$, then the proposal distribution q_t for sampling proposal state Y is selected as follows,

$$q_t(\cdot|X_1, \dots, X_t) \sim N(X_t, c_d^2 R_t),$$

where R_t is calculated by collecting the points $X_{t-H+1}, \dots, X_t$ into a $H \times 2$ matrix K, where each row is a generated sample. Then,

$$R_t = \frac{1}{H-1} \tilde{K}^T \tilde{K},$$

where $\tilde{K}$ denotes the centered matrix $K - \mathbb{E}(K)$. Thus, to propose the next state we sample from a Multivariate Normal distribution $N(X_t, \frac{c_d^2}{H-1} \tilde{K}^T \tilde{K})$.

The basic choice for c_d is $c_d = \frac{2.4}{\sqrt{2}}$ [1], which corresponds to theoretically optimal mixing properties of the Metropolis-Hastings search in the case of Gaussian targets and Gaussian proposals, which is what we will be using here.

Algorithm 2 Adaptive-Proposal algorithm

```
Input: Target density $f$; starting value $(x^{(0)}, y^{(0)})$; $2 \times 2$ covariance matrix $\Sigma$ of Multivariate Normal distribution; number of iterations $(n)$; Memory parameter H; Update frequency L; Scaling parameter c.
for $t = 1, 2, \dots, L$ do
    Let $Y^t = (x^{(t)}, y^{(t)}) \sim N((x_{t-1}, y_{t-1})^T, \Sigma)$
    Let $U \sim U(0,1)$
    if $U \leq min\left(\frac{f(Y^t)q(Y^t, X^{t-1})}{f(X^{t-1})q(X^{t-1}, Y^t)}, 1\right)$ then
        $X^t = Y^t$
    else
        $X^t = X^{t-1}$
```

```
for $t = L+1, \ldots, n$ do
    if $\mod(t, L) = 1$ then
        $K = (x_i, y_i)_{i=t-H+1}^{i=t}$
        $\tilde{K} = K - \mathbb{E}(K)$
        $\Sigma = \left(\frac{2.4}{\sqrt{2}}\right)^2 \frac{1}{H-1}\tilde{K}^T\tilde{K}$
    else
        Let $Y^t = (x^{(t)}, y^{(t)}) \sim N((x_{t-1}, y_{t-1})^T, \Sigma)$
        Let $U \sim U(0,1)$
        if $U \leq min\left(\frac{f(Y^t)q(Y^t, X^{t-1})}{f(X^{t-1})q(X^{t-1}, Y^t)}, 1\right)$ then
            $X^t = Y^t$
        else
            $X^t = X^{t-1}$
Output: A 2-dimensional Markov Chain $\{(x^{(t)}, y^{(t)})\}, t = 1, 2, \ldots$
```

Note, that we do not update at every step but rather update R_t every L steps. AP solves the issue of tuning a Metropolis-Hastings MCMC with highly correlated parameters, since it automatically updates and adapts to the target distribution thus keeping the search effective all the time. Furthermore, AP is superior to the MCMC with Gaussian Multinormal proposal (modification of the single component Metropolis-Hastings algorithm) or a Metropolis-Hastings algorithm where the proposal is not properly tuned and it is not *essentially inferior* to the Metropolis-Hastings algorithm with an optimal proposal distribution [1]. Thus, even if our MCMC before is properly tuned we are going to get similar results in less time, with an easier implementation by avoiding the troublesome tuning process of the proposals.

We are now going to compare these two algorithms in our example. Note that we use 10^5 iterations for both algorithms, memory parameter $H = 200$, update frequency $L = 200$, $c_d = \frac{2.4}{\sqrt{2}}$, starting vector $(x^{(0)}, y^{(0)}) = (0, 0)$, initial covariance matrix for AP, $\Sigma = \begin{pmatrix} 50 & 1 \\ 1 & 2.5 \end{pmatrix}$. For the classic MCMC approach we used a starting value of $(x^{(0)}, y^{(0)}) = (2.5, 2.5)$, and a covariance matrix as we have defined above, $\Sigma = \begin{pmatrix} 7.348^2 & -1.2 \\ -1.2 & 2.097^2 \end{pmatrix}$.

The results show what we would expect from the article; while both methods converge to the target distribution, AP has better mixing as we can see from the autocorrelation plots in Figures 4 (classic Metropolis-Hastings approach) and 5 (AP approach), especially the plots for Y (bottom right) seem to be decreasing much sharper for the AP approach. Additionally, the effective sample size is higher for AP as we can see from Table 2

	Classic MCMC	Adaptive Proposal
Effective Sample Size X	2085.736	2533.853
Effective Sample Size Y	1205.126	1569.136

Table 2: Table showing the Effective Sample Size for X and Y for both approaches.

For completeness we verify that both methods sample from the same target distribution (banana-shape plot) as we can see from Figure 6. The black dots are the samples generated by AP, while the transparent green are the samples generated by the classic Metropolis-Hastings algorithm.

Sampling from a Bimodal Distribution

We are going to use the following target density to sample from,

$$g(\theta) = k\left(\frac{4}{10}exp\{-(3-\theta)^2\} + \frac{6}{10}exp\{-(30-\theta)^2\}\right) \tag{2}$$

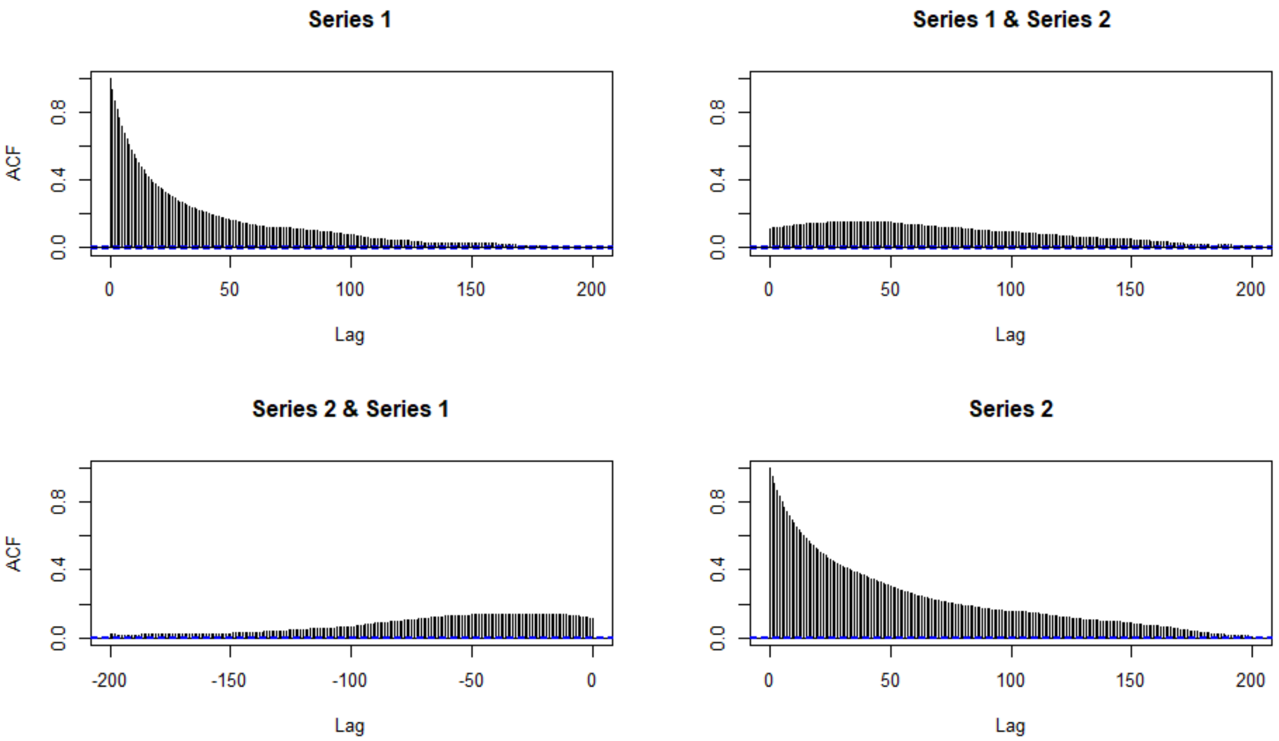

Figure 4: Autocorrelation plots for the classic Metropolis-Hastings approach. Top left is the plot for X while bottom right is the plot for Y.

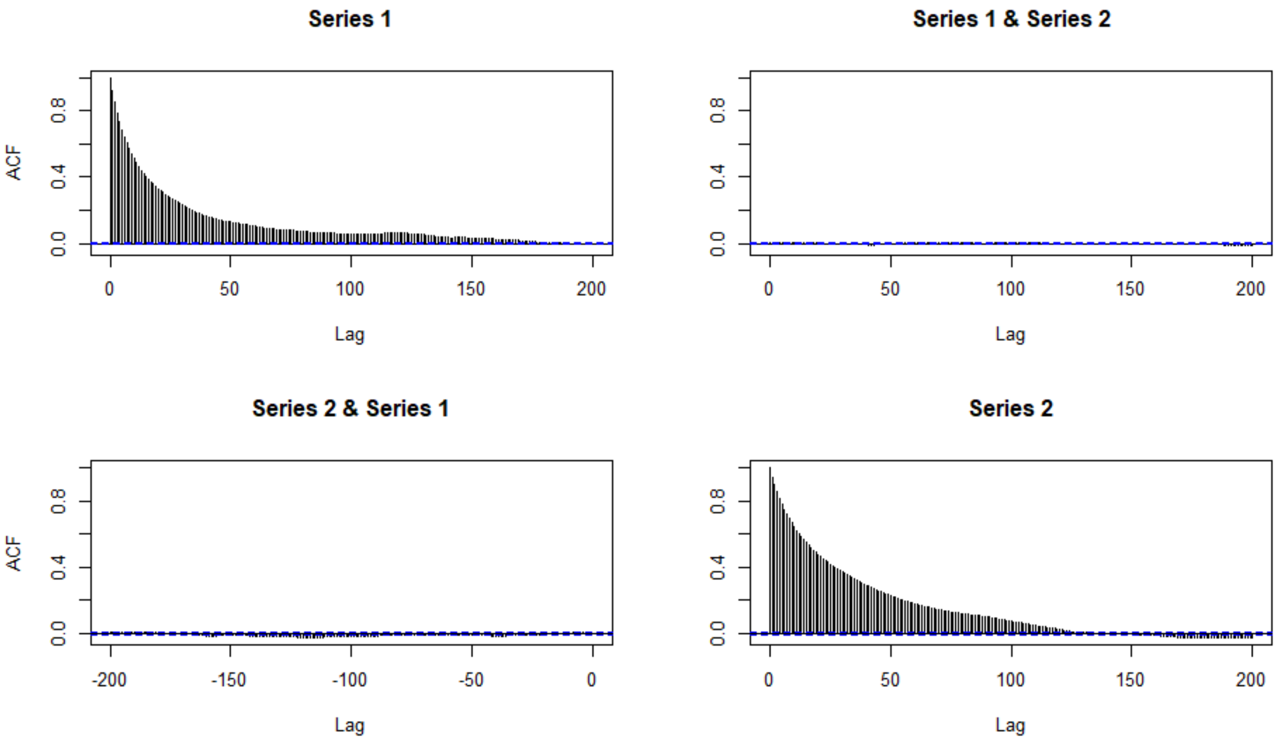

Figure 5: Autocorrelation plot for the AP approach. Top left is the plot for X while bottom right is the plot for Y.

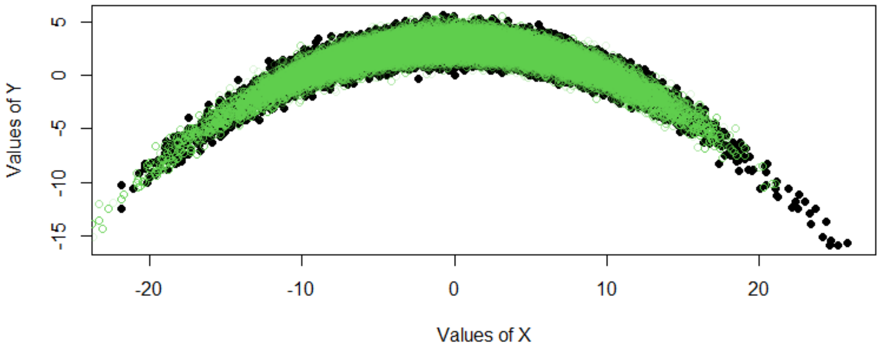

Figure 6: Plot showing the samples generated from both methods. The black dots are the samples generated by the AP approach while the transparent green dots are the samples generated by the classic MCMC algorithm.

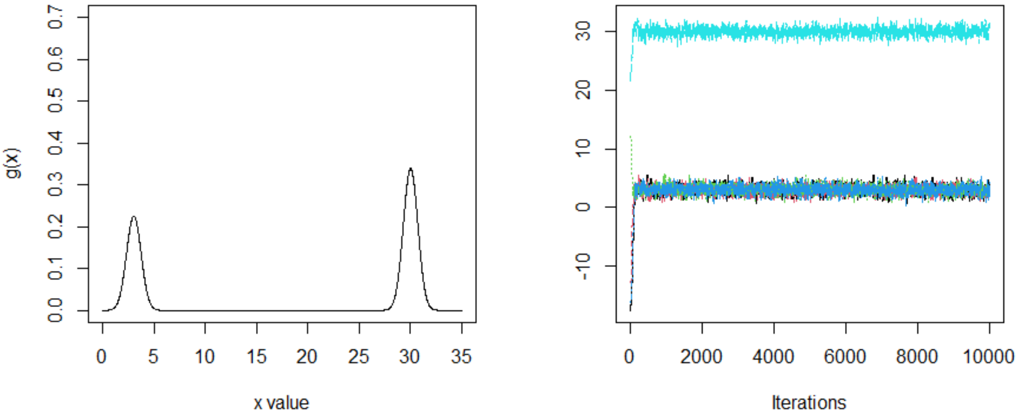

Figure 7: Plot showing the density of Equation 2 on the left. Traceplots of 4 different chains with different starting values and different values for the variance of the Normal Random Walk proposal.

Failures of plain Metropolis Hastings Algorithm

To show that simple random walk Metropolis-Hastings algorithms will not explore the full space well, we observe that the distribution is bimodal, one mode is at 3 while the other is at 30. Since they are well-separated this means that simple Metropolis Hastings will face difficulties. We can see the two modes in the left panel of Figure 7. Simple Normal Random Walks Metropolis-Hastings algorithms would be very efficient and accurate in sampling one of the two modes, as we can see from the right panel of Figure 7. The other solution is to use a big variance for the proposal since this would allow the chain to “jump” from one mode to the other. This would be very inefficient because many values will get rejected. One final diagnostic is the Gelman-Rubin diagnostic, which in this example gives a point estimation of 22.2 and an upper confidence interval of 37.1, which is larger than 1. Thus, we can safely conclude that in this specific example we have not reached convergence after 10^4 iterations. We would need at least 2 chains to explore the full space well. For completeness we state here that we used 5 Metropolis-Hastings algorithms for 10^4 iterations with a Normal Random Walk proposal with random variances and random starting values, the former drawn from a $U(-3, 3)$ distribution while the latter were drawn from a $U(-20, 40)$ distribution.

Parallel Tempering Algorithm

Parallel tempering is a multiple chain method that returns a specific construction of the distributions $\pi_1, \dots, \pi_m$. The target density has to be written in the form $\pi_1(x) \propto exp\{-H(x)\}$ and the rest are $\pi_m(x) \propto exp\{-H(x)/T_m\} = \pi^{1/T_m}(x)$, where $1 = T_1 < T_2 < \cdots < T_M$ are series of temperatures. We provide the pseudocode for parallel tempering below.

Algorithm 3 Parallel Tempering algorithm

```
Input: target density $\pi_1$; between-chain exchange probability $q(l, m)$; starting value $(x^{(0)}, y^{(0)})$;
variances for every proposal; number of iterations $(n)$; $M$ temperatures $T_1, T_2, \dots, T_M$; empty matrix $X$.
for $t = 1, 2, \dots, M$ do
    Let $\pi_t = \pi_1^{1/T_t}$
for $t = 1, 2, \dots, n$ do
    Run a Metropolis-Hastings algorithm for the $t^{th}$ chain. Store the values of the $t^{th}$ iteration in the
    $t^{th}$ row of matrix $X$.
    for $j = 1, \dots$ $M$, do
        Let $U \sim U(0, 1)$
        if $U < q(j, j-1)$ then
            $\alpha = \min\left(\frac{\pi_j(X_{t,j-1})\pi_{j-1}(X_{t,j})}{\pi_{j-1}(X_{t,j-1})\pi_j(X_{t,j})}\frac{q(j-1,j)}{q(j,j-1)}, 1\right)$
            Let $U' \sim U(0, 1)$
            if $U' < \alpha$ then
                Swap $X_{t,j}$ with $X_{t,j-1}$
        else
            $\alpha = \min\left(\frac{\pi_j(X_{t,j+1})\pi_{j+1}(X_{t,j})}{\pi_{j+1}(X_{t,j+1})\pi_j(X_{t,j})}\frac{q(j+1,j)}{q(j,j+1)}, 1\right)$
            Let $U' \sim U(0, 1)$
            if $U' < \alpha$ then
                Swap $X_{t,j}$ with $X_{t,j+1}$
Output: A M-dimensional Markov Chain $\{(X_{1,t}, X_{2,t}, \dots X_{M,t}\}$, $t = 1, 2, \dots, n$
```

Note that, in the above pseudocode we have to use the following between-chain exchange probability. The probability to swap chain l with chain m, when chain l is selected is,

$$q(l, m) = \begin{cases} 1/2 & \text{if } 2 \leq l \leq M, m = l-1, l+1 \\ 1 & \text{if } l = 1 \text{ and } m = 2 \text{ or } l = M \text{ and } m = M-1 \\ 0 & \text{otherwise} \end{cases}$$

We are now going to implement our own Parallel Tempering algorithm. We ran for 10^4 iterations, we used the following 7 temperatures (1, 2, 5, 10, 30, 50, 150), the following starting values (30, 6, 27, 8, 15 ,32 ,28) and the following variances (0.001, 0.5, 0.6, 1, 5, 7, 10). The reason we picked these specific values is that after many tries of tuning the algorithm we noticed that the density of Equation 2 has very narrow modes. Thus, we picked values close to the modes and adjusted the variances, then as the temperatures increased we increased the variance so as to allow for exchange of information from the whole sample space. The densities of all these functions together with the generated sample can be found at the left panel of Figure 8. Note that the final temperature (Red) is almost flat. Additionally, the green line (generated sample) is almost identical to the black line (actual density), this combined with the trace plot from the right panel of Figure 8 can verify graphically that the sampler is exploring the full space correctly.

Probability Estimation via MCMC

From the density plot at the left panel of Figure 8 it is clear that the estimation of $P(\theta > 20)$ should be around 0.6. The estimated value we obtained, using the ergodic theorem again, with the same specifications for temperatures, starting values, variances as above and a burn-in period of 200 samples, is 0.6016327 which is very close to the value we get by using numerical integration, 0.6.

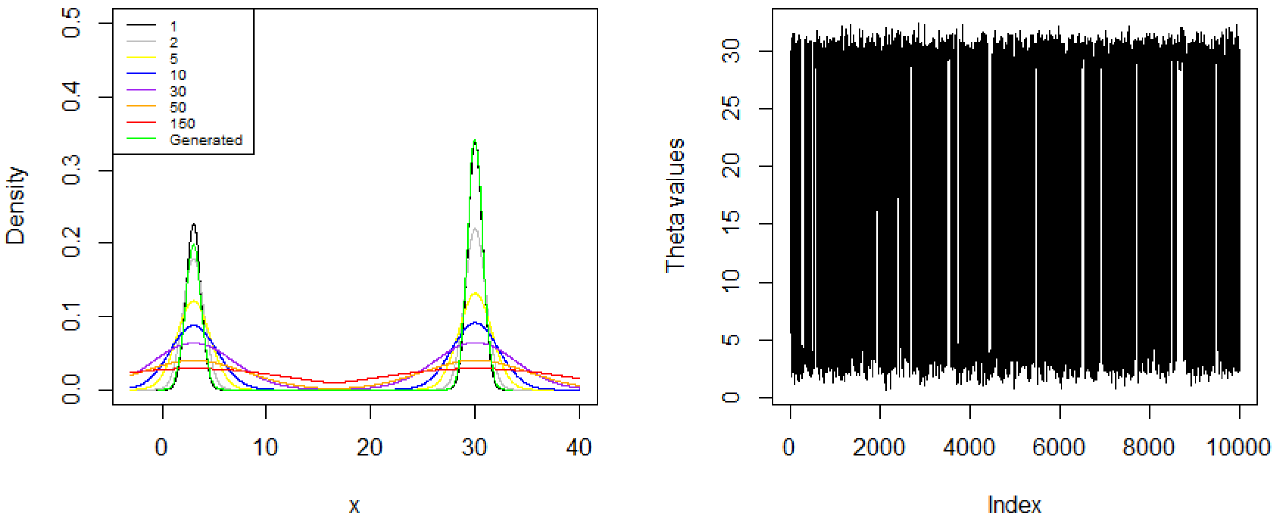

Figure 8: Plot of the density functions of 7 temperatures with the generated sample from parallel tempering (left). Traceplot of the generated sample (right).

We note here that the standard numerical integration method did not work for $\alpha = 3$, but worked for other values of α. This issue was resolved eventually by splitting $(-\infty, \infty)$ to $(-\infty, 3)$, $(3, 30)$, $(30, \infty)$.

References

[1] Heikki Haario, Eero Saksman, Johanna Tamminen, et al. "An adaptive Metropolis algorithm". In: *Bernoulli* 7.2 (2001), pp. 223–242.