OxWaSP Module 1: Adaptive MCMC

Virginia Aglietti Tamar Loach

October 18, 2016

1 Introduction to Adaptive MCMC - the AM algorithm

MCMC algorithms allow sampling from complicated, high-dimensional distributions. Choice of the proposal distribution (from which samples are taken in an attempt to approximate sampling from the target distribution π) determines the ability of the algorithm to explore the parameter space fully and hence draw a good sample. Adaptive MCMC algorithms tackle this challenge by using samples already generated to learn about the target distribution; they push this knowledge back to the choice of proposal distribution iteratively.

This project explores adaptive MCMC algorithms existing in the literature that use covariance estimators to improve convergence to a target distribution supported on a subset of $\mathbb{R}^d$. In this schema we learn about the target distribution π through estimation of its correlation structure from the MCMC samples. We use this correlation structure to improve our estimate of the target.

We first implement an adaptive MCMC algorithm AM (Haario et al., 2001) which is a modification of the random walk Metropolis-Hastings algorithm. In AM the proposal distribution is updated at time t to be a normal distribution centered on the current point X_{t-1} with covariance $C_t(X_0, ..., X_{t-1})$ that depends on the the whole history of the chain. The use of historic states means the resulting chain is non-markovian, and reversibility conditions are not satisfied. Haario et al show that, with a small update to the usual Metropolis-Hastings acceptance probability, the right ergodic properties and correct simulation of the target distribution none the less remain. The probability with which to accept candidate points in the chain becomes:

$$\alpha(X_{t-1}, Y) = \min\left(1, \frac{\pi(Y)}{\pi(X_{t-1})}\right)$$

With C_t given by:

$$C_t = s_d \text{cov}(X_0, ..., X_{t-1}) + s_d \epsilon I_d$$

Here cov() is the usual empirical covariance matrix, and the parameter $s_d = \frac{2.4^2}{d}$ (Gelman et al., 1996). ϵ is chosen to be very small compared to

the subset of $\mathbb{R}^d$ upon which the target function is supported. The AM algorithm is computationally feasible due to recursive updating of the covariance matrix on acquisition of each new sample through the relation:

$$C_{t+1} = \frac{t-1}{t} C_t + \frac{s_d}{t}(t\bar{X}_{t-1}\bar{X}_{t-1}^T - (t+1)\bar{X}_t\bar{X}_t^T + X_t X_t^T + \epsilon I_d)$$

with the mean calculated recursively by:

$$\bar{X}_{t+1} = \frac{t\bar{X}_t + X_{t+1}}{t+1}$$

Because of the instability of the covariance matrix, to implement the adaptivity we first run the algorithm with no change to the covariance of the proposal distribution. The adaptation starts at a user defined point in time, and until this time the covariance of the proposal is chosen to represent our best knowledge of the target distribution.

2 An example - testing the AM algorithm

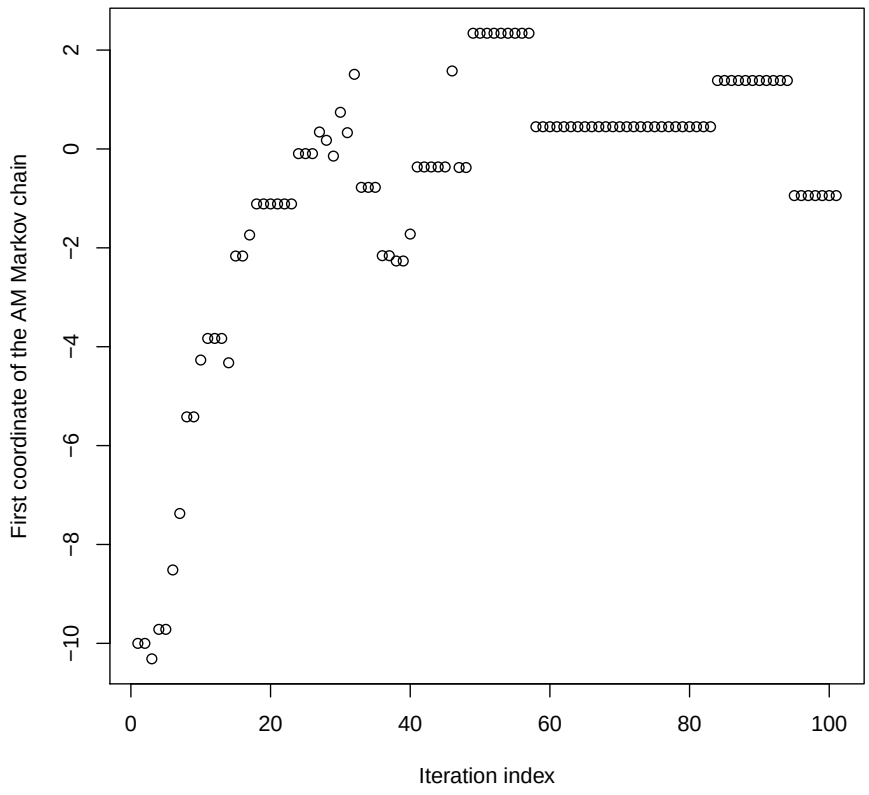
We now numerically test the AM algorithm. We have used two different target distributions: a correlated Gaussian distribution $N(0, \Sigma)$ and a "banana"-shaped distribution ((Roberts and Rosenthal, 2009)) given by:

$$f_B(x_1, ..., x_d) \propto \exp\left[-x_1^2/200 - \frac{1}{2}\left(x_2 + Bx_1^2 - 100B\right)^2 - \frac{1}{2}\left(x_3^2 + x_4^2 + ... + x_d^2\right)\right]$$

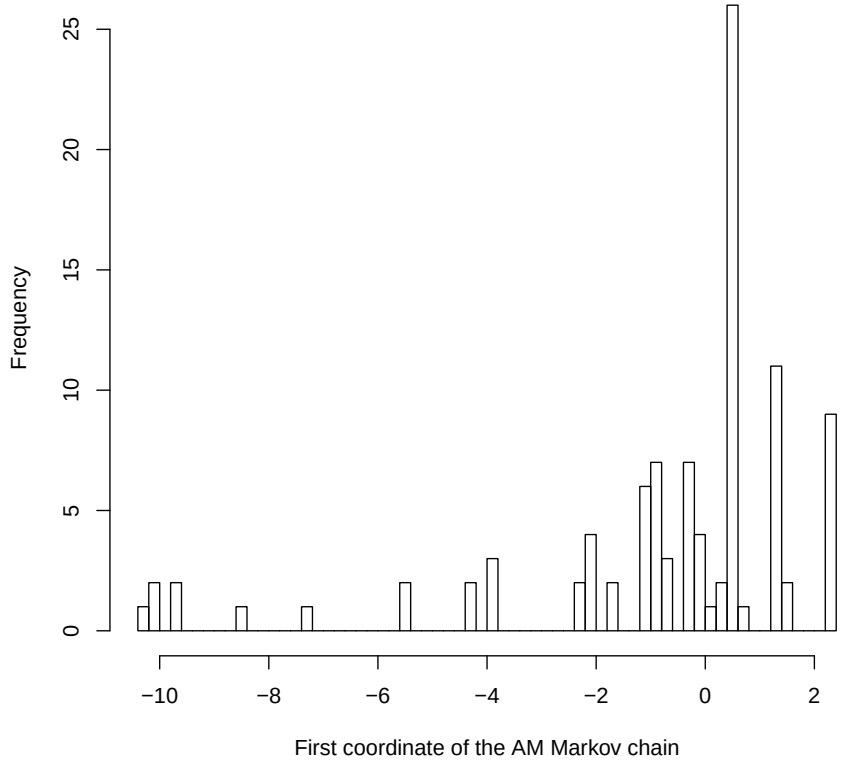
$B > 0$ is the "bananicty" constant (set to 0.1 throughout) and d is the dimension.

```
library(adaptiveMH)

n_iter = 100  #  Total number of iterations
t_adapt = 50  #  Time step at which adaptation begins
x_1 = rep(-10,2) #  Vector of inital values
adapt = "AM"  #  Choose the type of adaptation.

X = mcmc(target = pi_norm,
         n_iter = n_iter,
         x_1 = x_1,
         adapt=adapt,
         t_adapt = t_adapt
        )

## Running MCMC targeting  pi_norm with  100 iterations.
```

```
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.4175606 seconds.
## The acceptance rate was: 0

plot(X[,1],
     xlab="Iteration index",
     ylab="First coordinate of the AM Markov chain"
    )
```

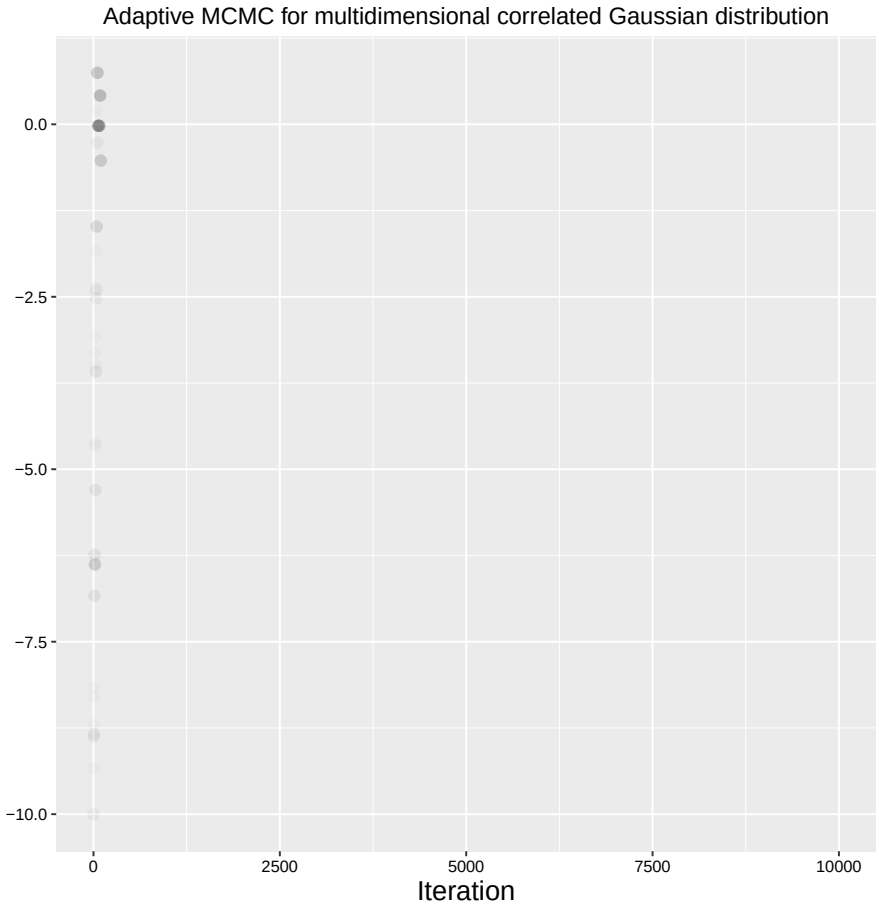



```
hist(X[,1],
     xlab="First coordinate of the AM Markov chain",
     ylab="Frequency",
     breaks=50,
     main=""
    )
```




```
library(ggplot2)

iterations<-c(seq(from= 1, to=n_iter+1, by=1))

plot<-qplot(iterations,
            X[,2],
            geom="point",
            main="Adaptive MCMC for multidimensional correlated Gaussian distribution",
            xlab="Iteration",
            ylab="",
            xlim=c(0,10000),
            alpha = I(1/50),
            size=I(2.5)) +  theme(axis.title.x = element_text(size = 15),
            title = element_text(colour='black'),
            axis.text.x=element_text(colour="black"),
            axis.text.y=element_text(colour="black"),
```

```
            axis.line = element_line(colour="black",
                                     size = 1,
                                     linetype = "solid"
                                    )
                                   )+
                                   theme(axis.line = element_line(colour = "darkblue", size = 1
plot
```

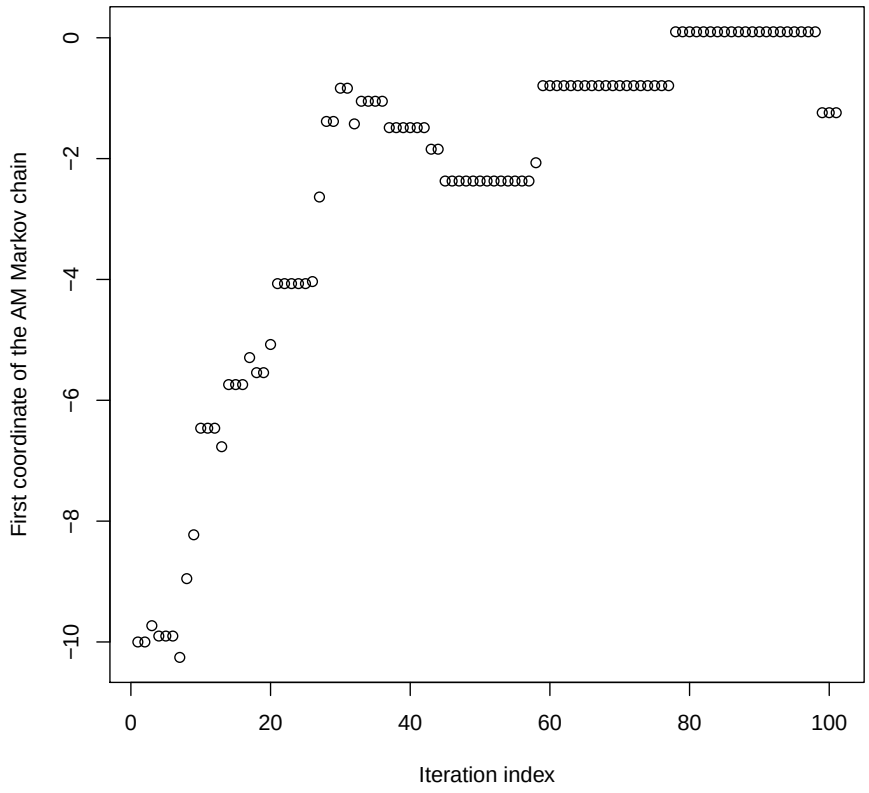


In order to test the increased performance of the AM algorithm, we have compared it with the classical Metropolis-Hastings algorithm. We found the AM algorithm to be faster and to perform better in terms of acceptance rate. Again for $10,000$ iterations of the MCMC and for a dimension $d = 2$, but this time with no adaptation:

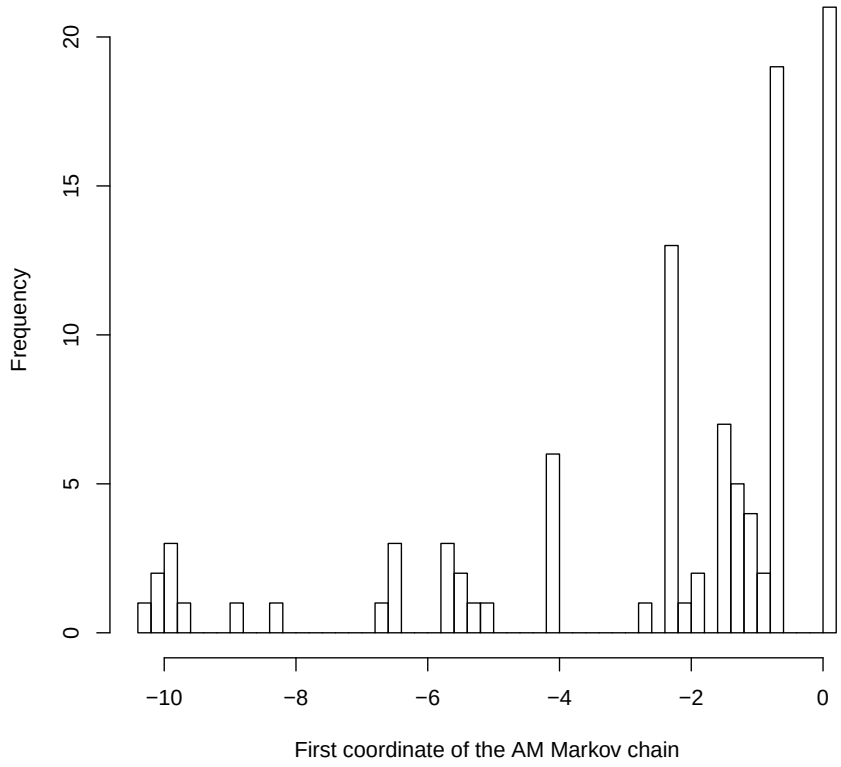
```
adapt = "None"  #  Choose the type of adaptation. "AM" or "None" currently.

X = mcmc(target = pi_norm, n_iter = n_iter, x_1 = x_1, adapt=adapt, t_adapt = t_adapt)
```

```
## Running MCMC targeting  pi_norm with  100 iterations.
## Adaptation algorithm is:  None .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.03240347 seconds.
## The acceptance rate was: 0
```

```
plot(X[,1],xlab="Iteration index", ylab="First coordinate of the AM Markov chain")
```




```
hist(X[,1], xlab="First coordinate of the AM Markov chain", ylab="Frequency", breaks=50, mai
```



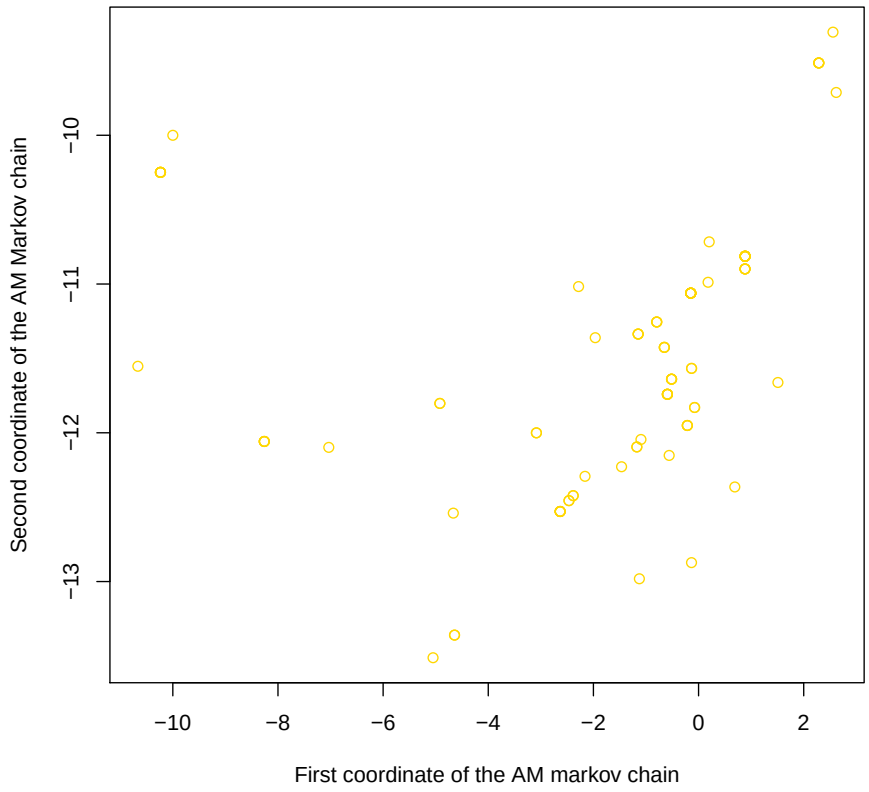
3 A Bit About knitr

Whilst this demonstrates that the algorithms both approximately sample from the correct target distribution, there is no improvement in acceptance rate due to the adaptation steps. We require a less regular shaped target distribution to test whether adaptation improves our acceptance rate towards the optimal 0.234. For this we follow Haario et al in using a banana-shaped distribution. The following results show a significantly lower acceptance rate for the adaptive version of the algorithm.

The AM algorithm with banana target:

```
adapt = "AM"  #  Choose the type of adaptation. "AM" or "None" currently.

X = mcmc(target = pi_banana, n_iter = n_iter, x_1 = x_1, adapt=adapt, t_adapt = t_adapt)
## Running MCMC targeting  pi_banana with  100 iterations.
```

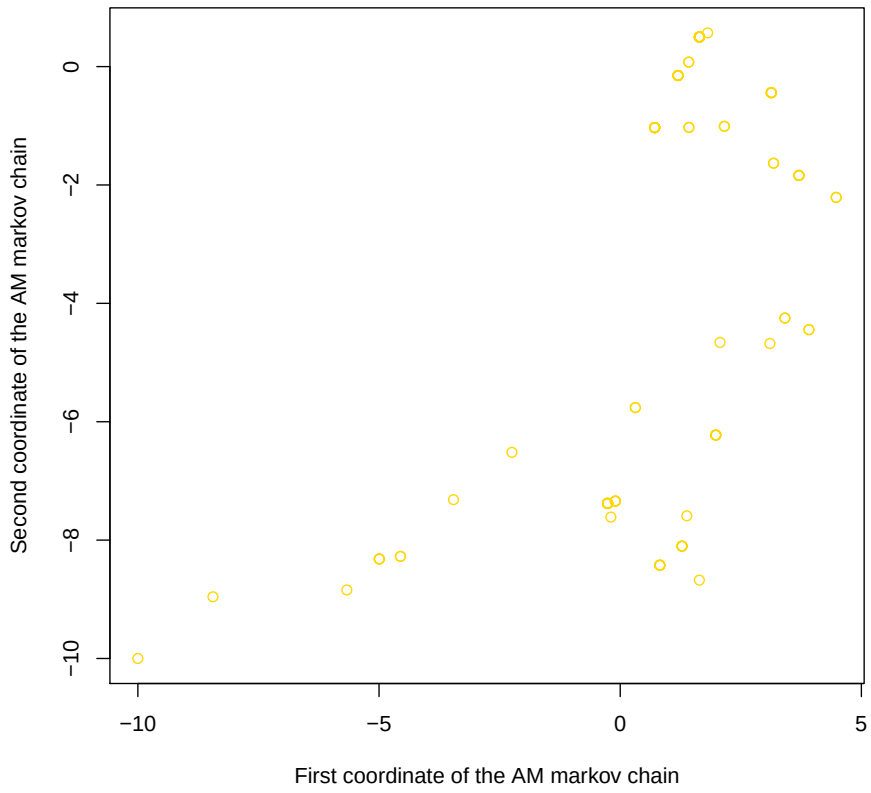
```
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.5627069 seconds.
## The acceptance rate was: 0

#plot(X[,1],xlab="Iteration index", ylab="First coordinate of the AM Markov chain")
#hist(X[,1], xlab="First coordinate of the AM Markov chain", ylab="Frequency", breaks=50, m
plot(X[,2],X[,1],col='gold', xlab="First coordinate of the AM markov chain",  ylab="Second c
```



The MH algorithm with banana target:

```
adapt = "None"  #  Choose the type of adaptation. "AM" or "None" currently.

X = mcmc(target = pi_banana, n_iter = n_iter, x_1 = x_1, adapt=adapt, t_adapt = t_adapt)

## Running MCMC targeting  pi_banana with  100 iterations.
## Adaptation algorithm is:  None .
```

```
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.01443243 seconds.
## The acceptance rate was: 0

#plot(X[,1],xlab="Iteration index", ylab="First coordinate of the AM Markov chain")
#hist(X[,1], xlab="First coordinate of the AM Markov chain", ylab="Frequency", breaks=50, m
plot(X[,2],X[,1],col='gold', xlab="First coordinate of the AM markov chain",  ylab="Second c
```



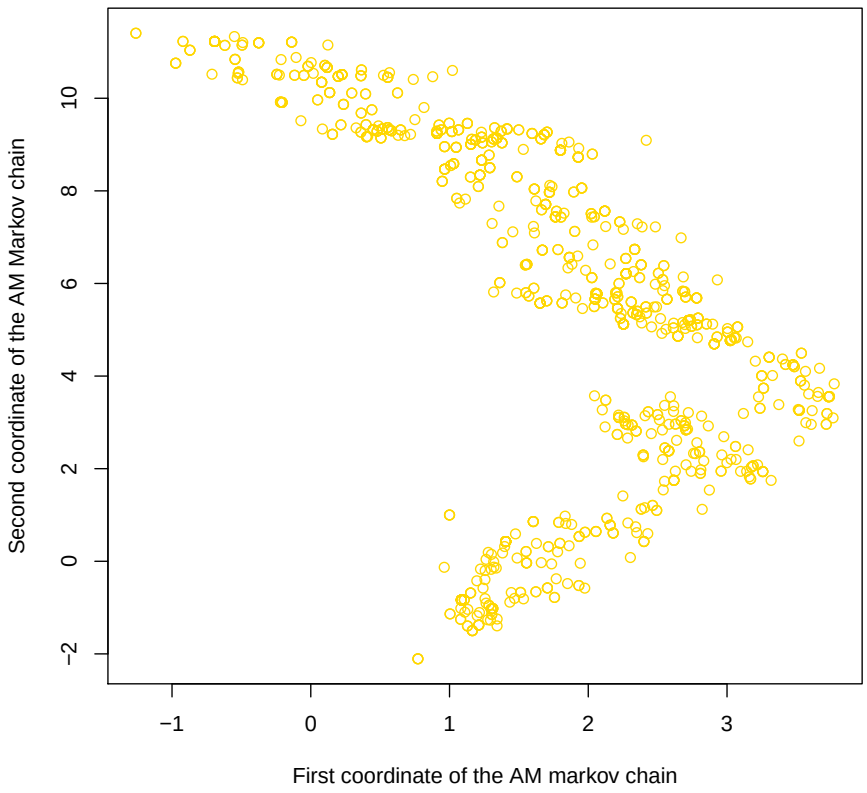
We can see that the acceptance probability is closer to the optimal 0.234 with adaptation according to Haario et al.

We now implement an adaptation scheme that uses stochastic stabilisation rather than numerical. This algorithm (AM2), from Roberts and Rosenthal (2009), differs from AM by using a mixture of Gaussians as the proposal distribution. In proportion β a normal uncorrelated distribution is used, and this is mixed with a correlated normal distribution.

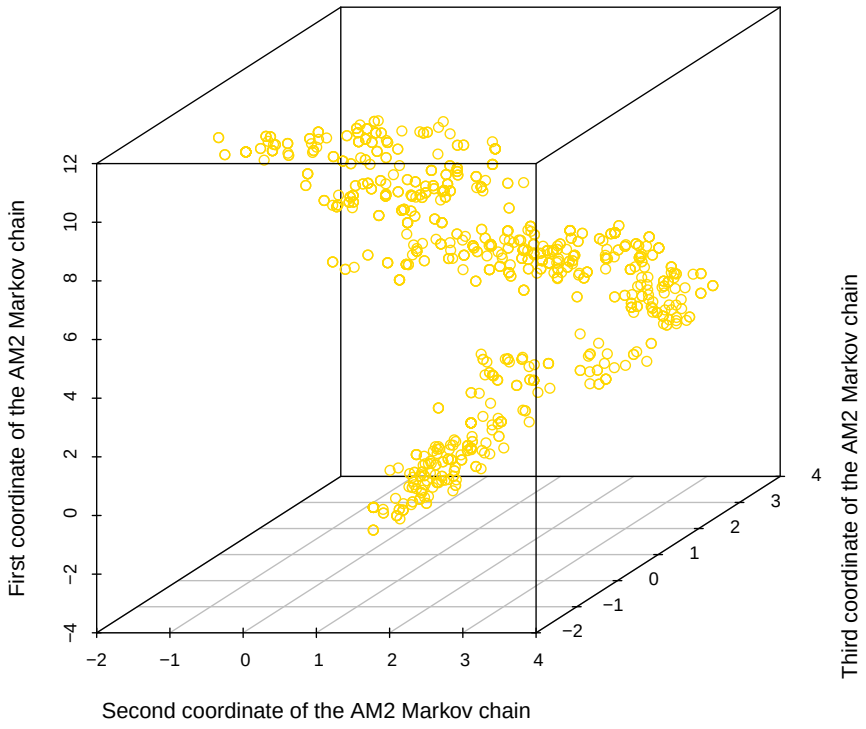
$$Q_n(x,.) = (1-\beta)N(x, s_d\Sigma_n) + \beta(N(x,(0.1^2)I_d/d)$$

The results for our implementation of AM2 are as follows. We have used a higher dimensional normal distribution for the proposal 20 and the banana target 3. Notice that the point at which we start adapting is determined by d, the dimension of the target distribution. The algorithm performs well when this d is high; we believe this is due to the point at which the adaptation starts. We have thus approximated a 3-dimensional banana with a 20-dimensional normal gaussian distribution. This forces the adaptation to start later. We would like to further investigate this effect, and explore various dimensional spaces.

```
n_iter = 1000 #  Total number of iterations
x_1 = rep(1,20) #  Vector of inital values
adapt = "AM2"  #  Choose the type of adaptation. "AM" or "None" currently.

X = mcmc(target = pi_banana3, n_iter = n_iter, x_1 = x_1, adapt=adapt, t_adapt = t_adapt)

## Running MCMC targeting  pi_banana3 with  1000 iterations.
## Adaptation algorithm is:  AM2 .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 10.80043 seconds.
## The acceptance rate was: 0

#plot(X[,1],xlab="Iteration index", ylab="First coordinate of the AM Markov chain")
#hist(X[,1], xlab="First coordinate of the AM Markov chain", ylab="Frequency", breaks=50, m
plot(X[,2],X[,1],col='gold', xlab="First coordinate of the AM markov chain",  ylab="Second c
```




```
library(scatterplot3d)
scatterplot3d(x=X[,2],y=X[,3],z=X[,1], xlab="Second coordinate of the AM2 Markov chain", yla
```



4 Less naive covariance estimation

Naive empirical covariance estimators are unstable for high-dimensional problems with little data; the literature .

Following these two adaptive MH implementations, we are now looking to use something that is less naive than the vanilla estimator for the covariance matrix. In particular we have began testing the shrinkage and thresholding estimators as modifications to AM2. We will now test whether these estimators translate into better convergence by looking at their trajectories. We then plan to include burn in, and compare all algorithms in terms of the suboptimality factor following Roberts and Rosenthal (2009).

5 LaTeX

LaTeXitself is complicated if you've never used it before, but I'm sure you'll pick it up quickly: there are a lot of guides on the web. I recommend using the `align` environment (in the `amsmath` package) for displayed equations:

$$f(x) = x^3 - x - 1$$
$$g(y) = y^4 + 2y$$

You can cite in two ways using the `natbib` package: (Haario et al., 2001) and Haario et al. (2001).

References

A Gelman, GO Roberts, and WR Gilks. Efficient metropolis jumping rules. 1996.

Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001. URL `http://projecteuclid.org/euclid.bj/1080222083`.

Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349367, 2009. doi: 10.1198/jcgs.2009.06134. URL `http://dx.doi.org/10.1198/jcgs.2009.06134`.