Computational Methods Assignment 7 - Markov Chain Monte Carlo Optimisation/Exploration

Martik Aghajanian

1 Question 1

The program written for this assignment was a Markov Chain Monte Carlo function optimiser and mapper. This was implemented as a class *MarkovChain*, which takes a function $f(\vec{x})$ initial starting vector $\vec{x}_0$ and initial standard deviation vector $\vec{\sigma}_0$ for each component of $\vec{x}$. The Metropolis algorithm is implemented for the function to advance to another $\vec{x}$. This involves the use of a proposal function $P(\vec{x})$, which determines how the next step is chosen. Additionally, a 'burn-in' period is included in this algorithm, since statistically, the equilibrium distribution represented by the final result is rapidly approached from any particular initial vector $\vec{x}_0$. This means that a portion of the chain at the start was required to get from the initial point to the final distribution of points, and the number of steps this corresponded to was the 'burn-in' period. Points from this part were not included in the final distribution of the function from the Markov chains. During this period, the proposal function was modified to represent more the current sample of points, whilst prior to the burn-in period terminating where the function values were recorded, the proposal function was fixed. This was because the proposal function represents the (Bayesian) conditional probability of moving to a new point $\vec{x}_{\text{new}}$ given the current point $\vec{x}_{\text{old}}$. If the proposal function is fixed then through Bayes' Theorem, the values of the function $f(\vec{x}_{\text{new}})$ and $f(\vec{x}_{\text{old}})$ correspond to the distribution of points in the Markov Chain, which allows this method to be used to map the function through the distribution of visited points, opposed to just maximising it. It also means that this burn-in time should be long enough so that the correlation between the points at the start and end of the burnin vanish, and that the proposal function converges to something which allows this method to map the function correctly. The Metropolis algorithm used for this is hence as follows:

1. Using proposal function $P(\vec{x})$, select a new vector of independent variables $\vec{x}_{\text{temp}}$.
2. Calculate the ratio $\alpha = f(\vec{x}_{\text{temp}})/f(\vec{x}_{\text{current}})$, where $\vec{x}_{\text{current}}$ is the current $\vec{x}$ vector.
3. If $\alpha \geq 1$ then accept the new point $\vec{x}_{\text{new}} = \vec{x}_{\text{temp}}$.
4. If $\alpha \leq 1$ then draw a uniform deviate β, and set $\vec{x}_{\text{new}} = \vec{x}_{\text{temp}}$ if $\beta \leq \alpha$ (This is effectively accepting the new point with probability α).
5. If $\beta > \alpha$ then stay at current point such that $\vec{x}_{\text{new}} = \vec{x}_{\text{current}}$.
6. If system still in the burn-in period, update $P(\vec{x})$ given current state of knowledge of parameter space.
7. If system is no longer in the burn-in period, test for convergence using some criteria.
8. Repeat this until convergence is attained..

This is the overall procedure, however specific criteria and convergence parameters were required or chosen for this particular test of the offset inverted Rosenbrock function:

$$f(\vec{x}) = 1000 - (1 - x_1)^2 - 100(x_1^2 - x_2)^2 \tag{1}$$

The proposal function used for this was (initially for this part, otherwise see question 3) a product of independent normally distributed random variables. To achieve this, the Box-Muller transform was used to take two uniform deviates X_1 and X_2, and produce two normally distributed deviates with zero means and unit variance:

$$Z_1 = \sqrt{-2\ln(X_1)}\cos(2\pi X_2) \quad \Longrightarrow X_1 = e^{-(Z_1^2+Z_2^2)/2}$$
$$Z_2 = \sqrt{-2\ln(X_1)}\sin(2\pi X_2) \quad \Longrightarrow X_2 = \frac{1}{2\pi}\tan^{-1}\left(\frac{Z_2}{Z_1}\right)$$

The Jacobian of this transformation was the product of two independent zero-centred unit-variance Gaussian distributions so that by putting two independent uniform deivates into the transform, the resulting output are two independent normal deviates. For a mean μ_i and standard deviation σ_i, this can be rescaled and shifted to give the desired distribution of points $x_i\tilde{N}(\mu_i,\, \sigma_i)$ through $x_i = \mu_i + \sigma_i Z_1$.Initially, the user supplies a trial standard deviation for the first N_σ steps of the Markov chain, which was chosen to be $\vec{\sigma}_0 = (5,\, 5)$, which from testing showed that this was of the correct order of the standard deviation that the method converged towards. Following N_σ steps in the Markov chain, the variance $\vec{\sigma}$ of the current sample of $\vec{x}$ vector in the chain was estimated. This became the new standard deviation of the proposal function $P(\vec{x})$, and was updated each time by having the mean of each independent normal distribution to be the current position along that axis and the standard deviation of that distribution to be the standard deviation of the most recent N_σ *accepted* points.

The number of steps for which it was reasonable to terminate the burn-in period was determined by calculating the kth autocorrelation function for each variable:

$$r_i(k) = \frac{\sum_{i=1}^{N_{points}-k}(x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{N_{points}}(x_i - \bar{x})^2}$$

where N_{points} gave the number of steps currently taken. This measured the degree of correlation between x_i values separated by k steps, and can be positive or negative, representing correlation and anti-correlation respecively. The closer the absolute value of this was to zero, the closer to no correlation these random variables separated by k steps were. For all independent variables (components) in $\vec{x}$, the absolute value of $r_i(k)$ was required to drop below a given threshold (chosen reasonably as 0.05) for the burn-in period to end. To avoid checking this every step, the autocorrelation function was checked every N_{check} steps (chosen as 100) for all components for all k smaller than the number of steps taken. This meant that the burn-in period N_{burn} was an integer multiple of N_{check}.

Following the termination of the burn-in period, the proposal function $P(\vec{x})$ was fixed (for this part, otherwise see question 2), and the burn-in points were stored separately from the following points so that they could be identified as being not suitable for determining the function. The Metropolis algorithm was iterated until the standard deviation of the the function points over a the most recent $N_\sigma(=100)$ accepted steps converged to some value. This was an indicator of the function remaining in a particular region of the variable space and also tended to some true value. It was expected that the spread would converge to a value, so a tolerance $\epsilon = 10^{-8}$ was included to account for this sensitive change, as the function in this case was particularly insensitive to the change in $\vec{x}$.
Primarily, it was noted that the Metropolis algorithm will run into problems for regions of the independent variable space for which $f(\vec{x}) < 0$. To avoid this, the function is set to zero for all regions where this occurs. While this may cause a higher number of rejected steps, it does not affect the mapping of the function. This is because the Markov Chain Monte Carlo aims to produce a distribution of points in this parameter space of $\vec{x}$ for which areas where the function is higher in value should be visited more often. Thus the points of interest for the Rosenbrock function are almost entirely in the regions where the function is positive (in principle, all points should be visited if the chain was left to run for long enough, but most of them would be positive). So that the function started in a region where this augmented Rosenbrock function was non-zero, the starting positions were chosen randomly within a square of side length 2 centred on the origin (corners on $\pm(1,\, 1)$, $\pm(1,\, -1)$) where the Rosenbrock function was positive. This meant that regions which were not positive (and hence set to zero) were visited much less frequently. To increase the amount of the function being mapped, $N_{\text{chains}}(=12)$ Markov Chains were simulated with randomised starting 2-dimensional vectors in this initial region, for a single run of the algorithm. This reduced the risk of missing regions of importance in this function and allowed the maximum to be picked out of a handful of N_{chains} runs.

The Rosenbrock function was mapped and explored, providing the distribution in Fig 1, with the burn-in data shown separately in Fig 2. It can be seen that the function stretches far into regions where is might be expected to be negative, however, this locus mapped out by the $N_{chains} = 12$ Markov chains shows exactly where the function is non-zero. This predicted a maximum of 1000 at $\vec{x}$ =(1.00135, 1.00271), requiring 1733497 accepted steps and 26956614 function evaluations, over 12 chains. This gives an average of 144458 accepted steps per chain and 2246384 function evalulations per chain, and an overall success rate of 6.43%. This relatively poor efficiency is due to the evaluation of the function to zero, though potentially made better by only starting initially close to the calculated maximum. A slightly more efficient approach could have been to shift the function and negative values and then search for the minimum. The burn-in data in Fig 2 shows relatively strong adherance to the shape of the distribution found in Fig 1, indicating that this burn-in period on the order 10^2 is sufficient to capture the points which do not initially follow the distribution. This small required burn-in period is related, again, to starting the algorithm in the region specified earlier.

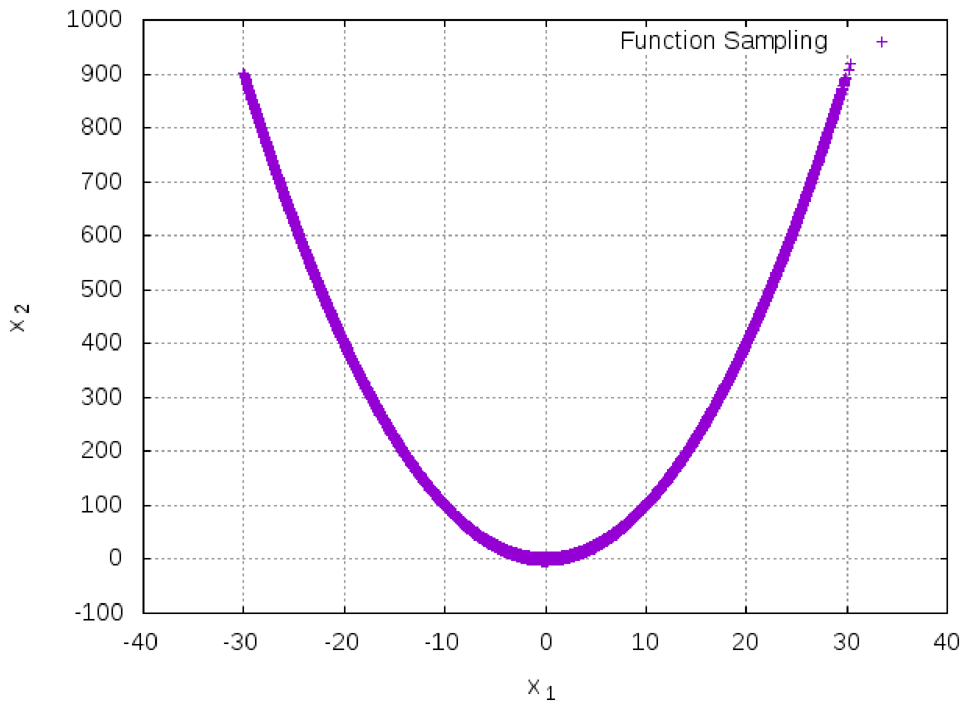

Figure 1: Distribution of the Rosenbrock function via Markov Chain Monte Carlo sampling in the $x_1 - x_2$ plane, for non-burn-in period points.

2 Question 2

Now the *MarkovChain* class was adapted so that the proposal function continued to update after the burn-in period had been exceeded. The results for the number of accepts and require function evaluations over 12 chains each for the same test conditions are shown in Table 1.

It can be seen from Table 1 that the variable proposal function and the fixed proposal function regimes, under the same convergence parameters and region of initial points, have very different levels of efficiency. While it can be seen that both regimes require a number of accepted steps with similar orders of magnitude, the number of function evaluations is an order of magnitude higher when the proposal function is allowed to vary. This is because while the number of accepted points in the region of interest are sampled, a higher number of points venture out into regions where the function is negative and hence points for which $f(\vec{x})$ is set to zero. When the proposal function is allowed to vary, the statistical properties of the Markov Chain are less

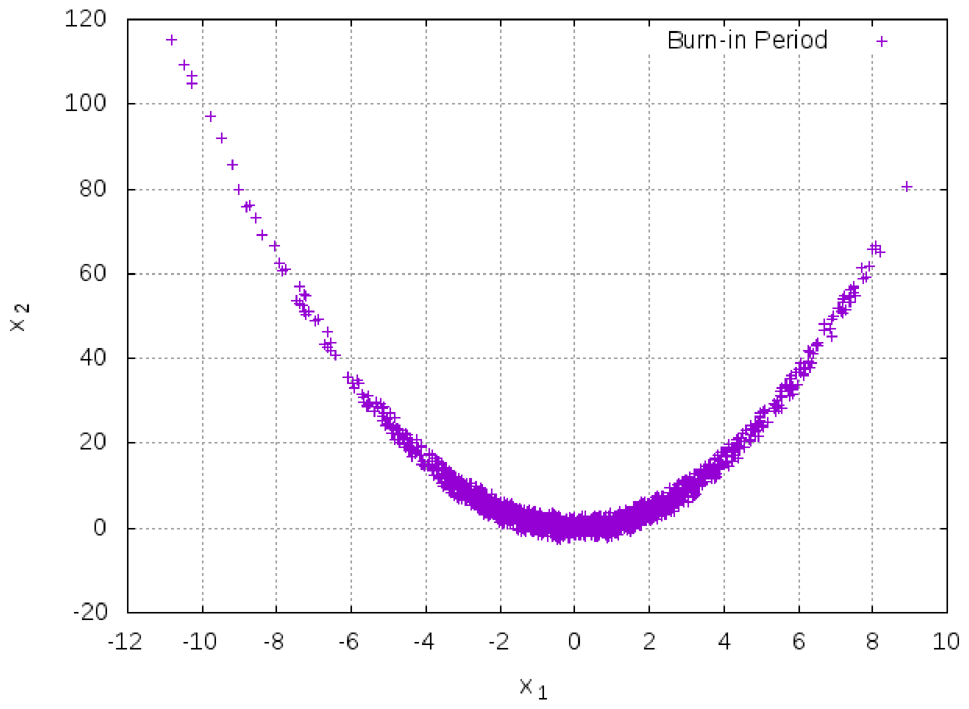

Figure 2: Markov Chain Monte Carlo samples of the Rosenbrock function in the $x_1 - x_2$ plane, for burn-in period points.

coherence and subsequently lost. More specifically, Bayes' Theorem that $P(A|B)P(B) = P(B|A)P(A)$ is not fulfilled for the individual steps, since the conditional probability $P(\vec{x}_2|\vec{x}_1)$ (which is the proposal function) changes at each step such. As a result, the function evaluations are no longer a good representative of the distribution of points or effective probability of obtaining those points and we obscure our mapping of the function. This causes the algorithm to sample more points outside the region where the function is negative. Also, this causes more samples to be taken for convergence, leading to the slightly higher number of accepted steps in a variable proposal function compared to a fixed proposal function after burn-in period.

	Fixed Prop. Functon	Variable Prop. Function
Maximum Estimated	1000	1000
Vector of Max	(1.00529, 1.00947)	(1.00408, 1.00712)
Total required steps	1.617193×10^6	2.447460×10^6
Steps per Chain	1.34766×10^5	2.03955×10^5
Steps of Chain with Max	2.12755×10^5	3.50830×10^5
Total required func. evaluations	3.0224216×10^7	1.31551014×10^8
Func. evals per Chain	2.518685×10^6	1.0962584×10^7
Func. evals of Chain with Max	1.886486×10^6	1.9689452×10^7

Table 1: Results of comparison of Markov Chain Monte Carlo function exploration between the case of fixing the proposal function, and the case of allowing it to vary after the burn-in period .

	Independent Gaussian Product	Multivariate Gaussian
Maximum Estimated	1000	1000
Vector of Max	(1.00529, 1.00947)	(1.00821, 1.01632)
Total required steps	1.617193×10^6	1.788733×10^6
Steps per Chain	1.34766×10^5	1.49062×10^5
Steps of Chain with Max	2.12755×10^5	3.02561×10^5
Total required func. evaluations	3.0224216×10^7	2.7528506×10^7
Func. evals per Chain	2.518685×10^6	2.294042×10^6
Func. evals of Chain with Max	1.886486×10^6	2.600982×10^6

Table 2: Results of comparison of Markov Chain Monte Carlo function exploration between using a product of independent Gaussians as the proposal distribution, and a multivariate Gaussian distribution .

3 Question 3

The *MarkovChain* class was further adapted to sample random variables from an $n-$dimensional multivariate Gaussian distribution. This required several extra algorithms to be implented. This distribution has a mean vector $\vec{mu}$ as before, but instead of decoupled variances in a vector $\vec{\sigma}$, the use of a covariance matrix C was required. So that the distribution took the form:

$$P_{MV}(\vec{x}_{MV}|\vec{\mu}, C) = \frac{1}{(\sqrt{(2\pi)})^n\sqrt{\det(C)}}\exp\left(-\frac{1}{2}(\vec{x}_{MV}-\vec{\mu})C^{-1}(\vec{x}_{MV}-\vec{\mu})\right)$$

To obtain this, a method of matrix decomposition called *Cholesky Decomposition* was directly implemented into the class, which produced a lower triangular matrix L for which $C = LL^T$, such that L is effectively the square root of the covariance matrix. From a vector $\vec{x}_N$ of one-dimensional normally distributed variables (still obtained via the Box-Muller transform) with zero mean and unit variance, the $n-$dimensional multivariate Gaussian distributed vector $\vec{x}_{MV}$ was obtained via $\vec{x}_{MV} = \vec{\mu} + L\vec{x}_N$. The mean $\vec{\mu}$ was it current position vector, and the covariance matrix was calculated such that $C_{ij} = (1/N_\sigma)\sum_k(x_i^{(k)} - \bar{x}_i)(x_j^{(k)} - \bar{x}_j)$, where $\bar{x}_i$ is the mean value of the ith component of the $\vec{x}$ vector over the last N_σ accepted steps, and $x_i^{(k)}$ is the kth entry of the N_σ sample for the ith component. This was then used to compare with original regime of a proposal function of a product of inpendent normal distributions, for the same convergence paramters and initial setup, and is shown in Table 2.

Table 2 shows a similar number of accepted steps, whilst using the multivariate gaussian has higher relative accepted steps per function evaluations, indicating it is a proposal function which better represents the conditional probability of going from one point to another in the Markov Chain. The mapping of the function in the $x_1 - x_2$ plane along with the burn-in points are shown in Fig 3 and Fig 4 respectively. This shows that for a multivariate gaussian proposal function, the burn-in points in Fig 4 are more concentrated and vary in the $x_1 - x_2$ plane, due to correlated variables in either dimension. This explain why the rate of accepted steps is higher, as this higher localisation of points from the multivariate Gaussian results in the Markov chain venturing less into regions which are not of interest and providing a more close-packed mapping of the function, seen in Fig 1.

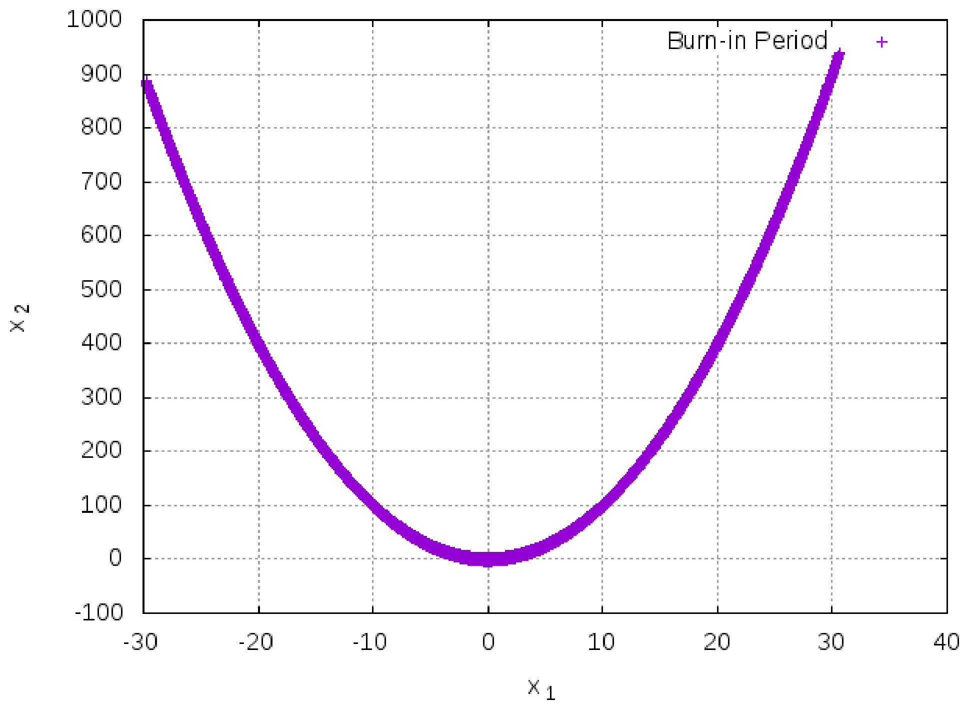

Figure 3: Distribution of the Rosenbrock function via Markov Chain Monte Carlo sampling using a multivariate gaussian proposal function in the $x_1 - x_2$ plane, for non-burn-in period points.

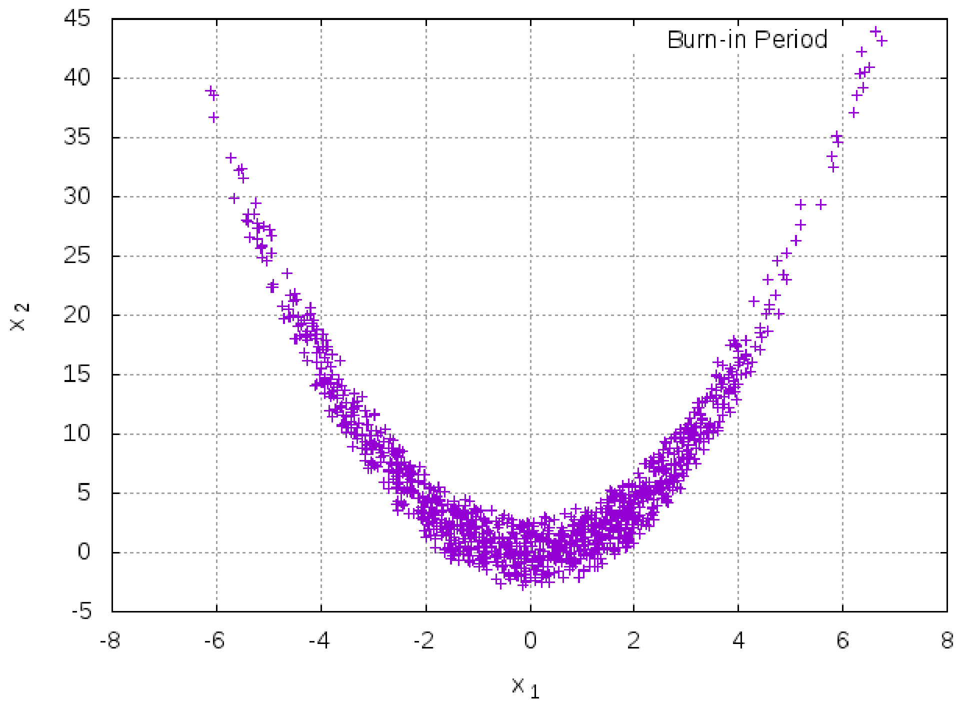

Figure 4: Markov Chain Monte Carlo samples of the Rosenbrock function using a multivariate gaussian proposal function in the $x_1 - x_2$ plane, for burn-in period points.