HEWLETT PACKARD

A Simulation Study of Generalised Processor Sharing

Jamie Wood, Neil O'Connell
Basic Research Institute in the Mathematical Sciences
HP Laboratories Bristol
HPL-BRIMS-96-23
September, 1996

In recent work of the second author, some curious phenomena were observed concerning optimal resource allocation in a resource with two incoming traffic streams with dedicated storage space in a finite buffer, and limited output capacity to be shared between the two streams according to some priority rule. An example is given where there are two independent incoming streams, with identical statistical properties, yet the optimal allocation of resources, from the point of view of minimising the overall frequency of overflow, is to give top priority to either of the two streams and more buffer space to the other. These results were obtained using large deviation theory and as such are only large buffer approximations; also, explicit calculations were only possible for relatively specific traffic types.

In this paper we explore, by simulation, the universality of these phenonema. We find that, not only do the observations persist for a wide variety of traffic types, they are also valid for systems with small buffers. There is a `rotational' effect as the buffer size decreases, leading to optimal policies that give top priority to one stream, as before, but that stream gets an even smaller share of the available buffer space.

A SIMULATION STUDY OF GENERALISED PROCESSOR SHARING

Jamie Wood[1] *and Neil O'Connell*[2]

BRIMS Technical Report HPL-BRIMS-96-XX

Abstract

In recent work of the second author, some curious phenomena were observed concerning optimal resource allocation in a resource with two incoming traffic streams with dedicated storage space in a finite buffer, and limited output capacity to be shared between the two streams according to some priority rule. An example is given where there are two independent incoming streams, with identical statistical properties, yet the optimal allocation of resources, from the point of view of minimising the overall frequency of overflow, is to give top priority to either of the two streams and more buffer space to the other. These results were obtained using large deviation theory and as such are only large buffer approximations; also, explicit calculations were only possible for relatively specific traffic types.

I this paper we explore, by simulation, the universality of these phenonema. We find that, not only do the observations persist for a wide variety of traffic types, they are also valid for systems with small buffers. There is a 'rotational' effect as the buffer size decreases, leading to optimal policies that give top priority to one stream, as before, but that stream gets an even smaller share of the available buffer space.

[1]Department of Mathematics, Cambridge University. This research was carried out while the first author was on a student internship at BRIMS.

[2]BRIMS, Hewlett Packard Laboratories, Filton Road, Stoke Gifford, Bristol BS12 6QZ. Corresponding author.

INTRODUCTION

This paper is the study of a single server buffer by direct simulation. Its main aim is to confirm, and, where possible, expand upon analytic results obtained by the application of large deviation theory to systems of this type. Even in such a simple situation there are still many different QOS (Quality of Service) parameters that it would be useful to investigate by this method. This report is exclusively concerned with minimising the overall frequency of overflow from the buffer, a term more usually described as the rate of cell loss from the system. This minimisation can be achieved by altering the service parameters of the unit. The sheer complexity of the rate function, for even simple distributions, makes simulations of this type a viable alternative.

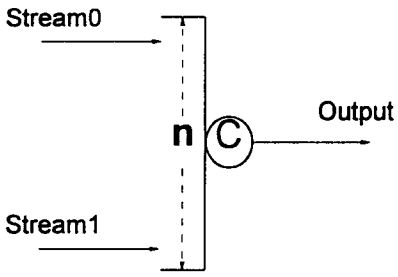


figure 1: A diagram of the single server buffer system.

The unit under simulation is crudely depicted above. It consists of a buffer of a fixed size n which is served by a processor also with a fixed rate of service, c. Two traffic streams are fed into the buffer and a single output stream is produced. The minimisation is concerned with finding the optimal divisions of both the buffer and the service the processor provides. This means the first stream is allocated buffer space an and the processor time cp where $0 \le a, p \le 1$. The corresponding parameters for the second stream are clearly $(1-a)n$ and $(1-p)c$.

SOME LARGE DEVIATION RESULTS

In this section a few simplified results of large deviation theory are presented so that motivation for this study can be seen. These results are all obtained from

Shwartz and Weiss[1] and O'Connell[2]. This study was done with close supervision of Neil O'Connell and continuous reference to the latter paper so results from that paper will be displayed without justification in most cases.

Large deviation theory, as it's name suggests, is a method of estimating the probability of rare events. More specifically it is concerned with the calculation of the following probability for the IID distributions X_i.

$$P\left(\frac{x_1 + x_2 + \ldots + x_n}{n} > \theta\right) = e^{-nI(\theta)}$$

The function $I(\theta)$ is known as the large deviation rate function, or simply the rate function. The study of large deviations is essentially the study of the properties and calculation details of this function. A simplified definition of the function, for a given distribution X, is that $I(\theta)$ will equal the Fenchel-Legendre transform of the cumulant generating function of the distribution, $\lambda_x(t)$, defined as follows:

$$I_x(\theta) = \inf_t \{t\theta - \lambda_x(t)\}$$

Because the definition is in terms of an infinum this result does not always lead to an explicit form of the rate function, complicating calculations based upon it. This is the case even with relatively simple distributions such as the uniform distribution. Fortunately the gaussian distribution does have a simple form, so the initial calculations were made using gaussian inputs.

In order to use this idea for queuing theory a rather long and complicated amount of theory must be developed, this is all explained in [2]. The end result of which is the following estimate for the overall frequency of cell loss:

$$P(\text{cell loss}) = P(Q_0 > an \text{ or } Q_1 > (1-a)n) \approx e^{-n\delta(a,p)}$$

In this equation Q_0 and Q_1 are the queues in the respective buffers and $\delta(a,p)$ is given by the set of formulas below:

$$\delta(a,p) = (a\delta_1(p)) \wedge ((1-a)\delta_2(p))$$
$$\delta_1(p) = \inf_{\tau \geq 0, x_2 \geq 0} \left\{ \tau \left[\Lambda_1^* \left(\tfrac{1}{\tau} + (pc) \vee (c - x_2) \right) + \Lambda_2^*(x_2) \right] \right\}$$
$$\delta_2(p) = \inf_{\tau \geq 0, x_1 \geq 0} \left\{ \tau \left[\Lambda_2^* \left(\tfrac{1}{\tau} + ((1-p)c) \vee (c - x_1) \right) + \Lambda_1^*(x_1) \right] \right\}$$

This function, $\delta(a,p)$ is clearly very desirable to find, but because of its very non-trivial nature it is difficult to find explicitly except in simplified cases. The following pages contain a few estimates of $\delta(a,p)$ for some different distributions and priority settings.

ESTIMATION OF $\delta(a,p)$ BY SLOPE FITTING

In order to minimize the overall frequency of overflow we need to maximize $\delta(a,p)$ with respect to a and p. To do this we set

$$a = a^*(p) = \frac{\delta_1(p)}{\delta_1(p) + \delta_2(p)}$$

and then inserting this back into the original equation to get:

$$\delta\left(a^*(p), p\right) = \frac{\delta_1(p)\delta_2(p)}{\delta_1(p) + \delta_2(p)}.$$

So to obtain an estimate of $\delta(a,p)$ we need to calculate $\delta_1(p)$ and $\delta_2(p)$ and then use the function given above.

This immediately causes a problem when we come to simulate the buffer as we need to fix both a and p to obtain our estimates of $\delta_1(p)$ and $\delta_2(p)$. One possible solution would be to create a mesh of a and p so that we could see a surface plot of the results. Unfortunately a simulation of this type is unfeasible as we need to put quite a lot of data through the program in order to obtain a reasonable estimate. Creating a mesh in this manner greatly increases the time required for each simulation to run, so

much so that it proved beyond the scope of this investigation. A crude estimate was instead used, being that a range of values of p were fed into the computer and a was calculated to be $1-p$. This is intuitively reasonable as we are simply apportioning the total service equally.

The program works by simulating the traffic arrivals and recording the amount of traffic in each queue at all of the discrete time intervals. This information is then used to construct a histogram and then an estimate of the sequence of probabilities $P(Q>q)$, for all values of q. This information can be used to produce a plot of q against $log(P(Q>q))$. The slope of this watermark will then be approximately equal to the corresponding δ for that queue. For this purpose a sequence of code was written to be able to find an approximate line of best fit for these quite crude graphs. Once the values of δ_1 and δ_2 had been established it is then a simple matter to produce an estimate for $\delta^*(a,p)$.

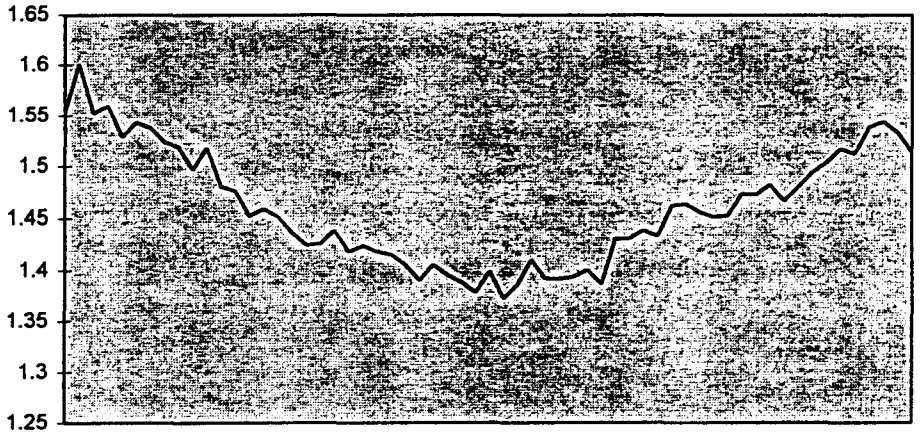


figure 2: Plot of δ(a,p) *against* p *for two identical traffic streams* N(*0.4,0.1*) *with a service capacity,* c, *of 1.*

Figure 2 shows a plot for the symmetric gaussian inputs. For this, and the following simulations the buffer size was set at 6. This size of buffer provides a reasonable approximation to large n for these service statistics. This graph is in good concordance with the graph in [2,figure 1] except for the values all being a little high. This is mainly due to failure of this system to plot for the optimal values of a, which the analytic graph does, and that the watermark plots typically have a slightly higher

gradient than the theory predicts. This is caused by the system not running for a long enough to give accurate estimates for the rarer events.

Unfortunately this plot was perhaps the best result achieved with the use of this method. The slope fitting algorithm proved to be too unreliable to be able to consistently give graphs of the accuracy required so the use of it to produce plots for later distributions, where no analytic plot existed, would have been inconclusive. The plots did give useful qualitative validation of the existing results however. *Figure 3* below shows a plot for gaussian streams with different means.

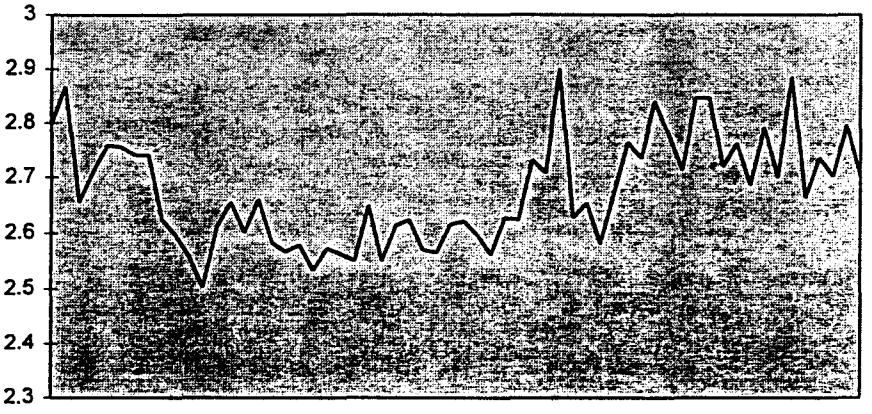


figure 3: plot of $\delta(a,p)$ against p for stream0 with N(0.2,0.1) and stream1 with N(0.4,0.1). The service rate was again 1.

OPTIMISATION OF *A* AND *P* BY SIMULATION

In order to find the optimal conditions on a and p for a wider variety of distributions, a far more direct method of simulation was decided upon. This was to input the traffic as before but to only count the number of times the buffer actually overflowed and not try to find more detailed information about the distribution of the queue lengths. The program for this purpose was considerably simpler and ran a lot faster so a full surface plot of $\delta(a,p)$ was created for each different simulation. This enabled a much clearer picture of the qualitative features of $\delta(a,p)$ to be determined.

An estimate of $\delta(a,p)$ was obtained by the following approximate formula:

$$\delta(a,p) \approx \frac{1}{buffersize} \log\left\{\frac{\#overflows}{\#iterations}\right\}$$

These simulations were run for three different distributions; the gaussian distribution, the uniform distribution and a simple two-state markov chain. In each case the size of the buffer used was progressively altered to see how this effected the shape of the plot. All the plots referred to in the text below are presented as colour printouts with this document. For the gaussian and the uniform distribution the colours go from deep blue for lots of overflows through green and then red and finally to black for very few overflows. In the markov case a different colour pattern was chosen so that the banding effect could be seen more clearly. The plots show p varying on the vertical scale and a on the horizontal scale with the point *(0,0)* in the bottom left hand corner.

- **GAUSSIAN INPUTS**

The gaussian inputs produced results largely as expected. The graphs presented in [2] can be roughly confirmed using these plots as they simply refer to the best result obtained with respect to a for any given value of p. The analytic result predicted that $p=1$ and $a=0.13$ for the symmetric case, and in the plots this is confirmed for the large values of the buffer. But as the buffer size decreases it is interesting to note that the best priority still remains at *1* but the optimum value for a decreases with the buffer size. What this is saying is that the buffer space allocated to the stream with maximum priority is something of a luxury and as the total amount of space decreases this space can be used more profitably.

The results obtained for the gaussian with different input streams produced the same qualitative results and also confirmed the values predicted analytically. The next test was to see if these basic results were a feature of the gaussian distribution or of the system as a whole.

- **UNIFORM INPUTS**

With the uniform inputs it there was an immediate problem in that the mean and the variance could not be set independently. So the mean was fixed at 0.4, and this meant the variance could not go any higher than 0.05 without allowing for the possibility of negative input which was highly undesirable. For the input streams distributed between $[0,0.8]$ it was clear that the important aspect was to set the priority to a high value for one stream and give it none of the available buffer space. This method would insure that the cell frequency would be minimum but would clearly cause significant delay in the other stream. In a further study it would be interesting to see how taking into account the mean cell delay would affect these plots.

In the asymmetric case the second stream was distributed between $[0.2,0.6]$, reducing the variance to ≈ 0.13. As with the gaussian case, the way to optimize the higher varying stream with respect to a steady one is to give the former as much buffer space as possible and concentrate on serving the steadier stream.

What is noticeable with all the uniform inputs is the very sharp drop off when the buffer is set to 0 for that stream and when the priority is gradually reduced. This is explained by the fact that the uniform distribution has a maximum which is less than the service size so the priority can be set to serve all the traffic from one stream. This is indeed the case as the drop-offs are at $p = 0.8$ and $p = 0.6$ respectively. This phenomenon is even more apparent in the following case.

- **MARKOV INPUTS**

The plots for the markov inputs produced strikingly different results to the above cases. In all the plots the frequency of overflow clearly occurs in a series of strong bands. What is more remarkable is that these bands are strongly dependent on the size of the buffer, a fact not predicted by the calculation of $\delta(a,p)$. The size of the buffer though does not have any effect on the optimum values of a and p. The distributions used had the same mean and variance as the gaussian case so that a direct comparison could be made. In each of the markov chains there are three variables which can be altered, the size of the traffic produced and the activation and

deactivation probabilities. In fixing the mean and variance the size of the traffic produced was kept constant and the two probabilities were constrained by a simple ratio. The probabilities were kept reasonably low, in order to produce bursty traffic, but surprisingly the alteration of these probabilities had little effect on the plots produced.

Some of the banding can be explained by the realisation that what we are looking at here is a far more discrete process. There are infact only four possible events at any given traffic arrival. This gives rise, in the high priority cases, of a random walk process which is analogous to the classic gamblers ruin problem.

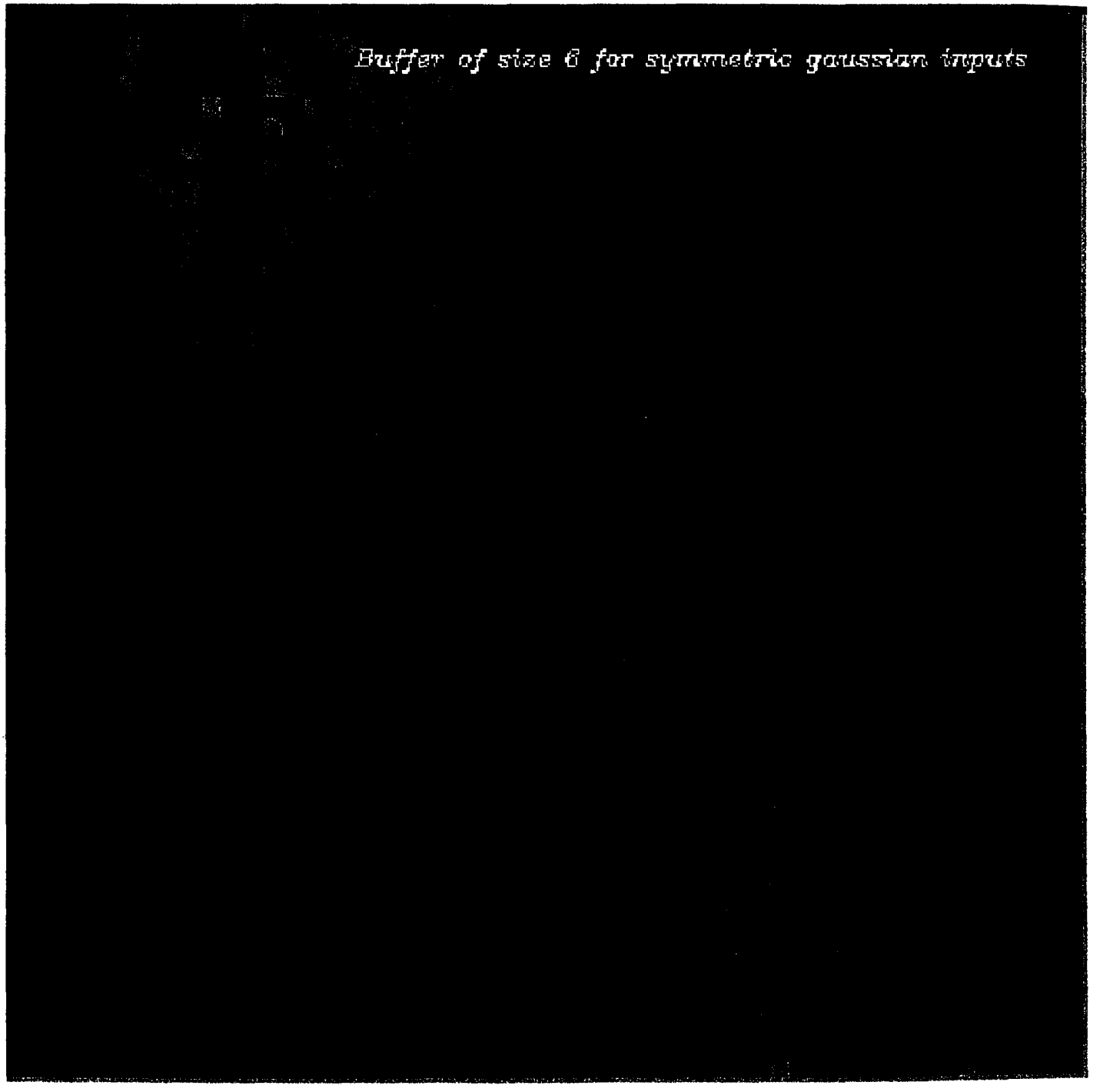
It would be interesting to discover if the large deviation predictions were in agreement with the gamblers ruin problem in this case, but unfortunately it was beyond the scope of this study to encompass such a detailed analytic viewpoint.

REFERENCES

[1] Adam Shwartz and Alan Weiss (1995). Large deviations for performance analysis.

[2] Neil O'Connell. Queue lengths and departures at single-server resources.

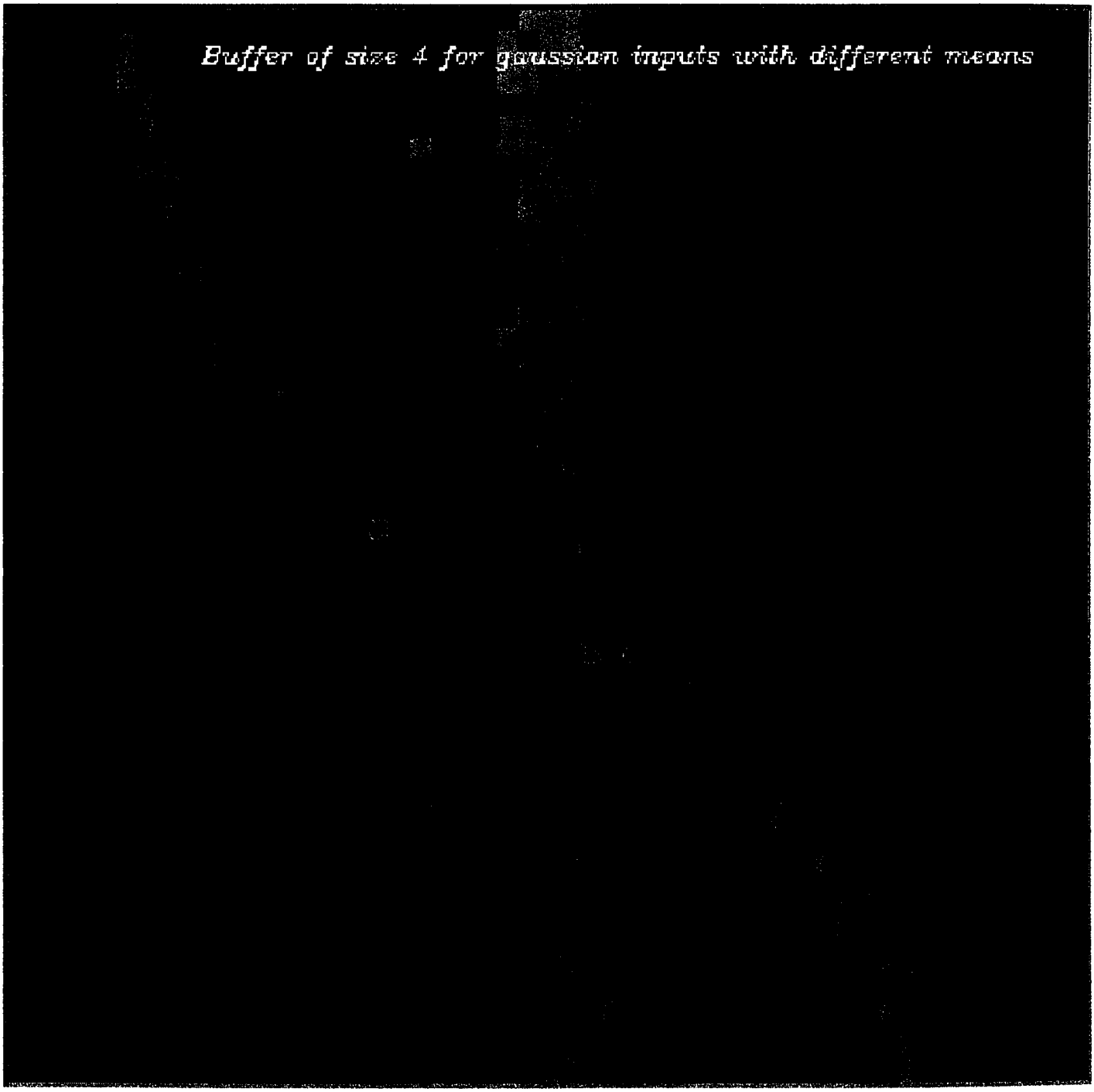
Thanks to Neil O'Connell for his considerable help with this study.

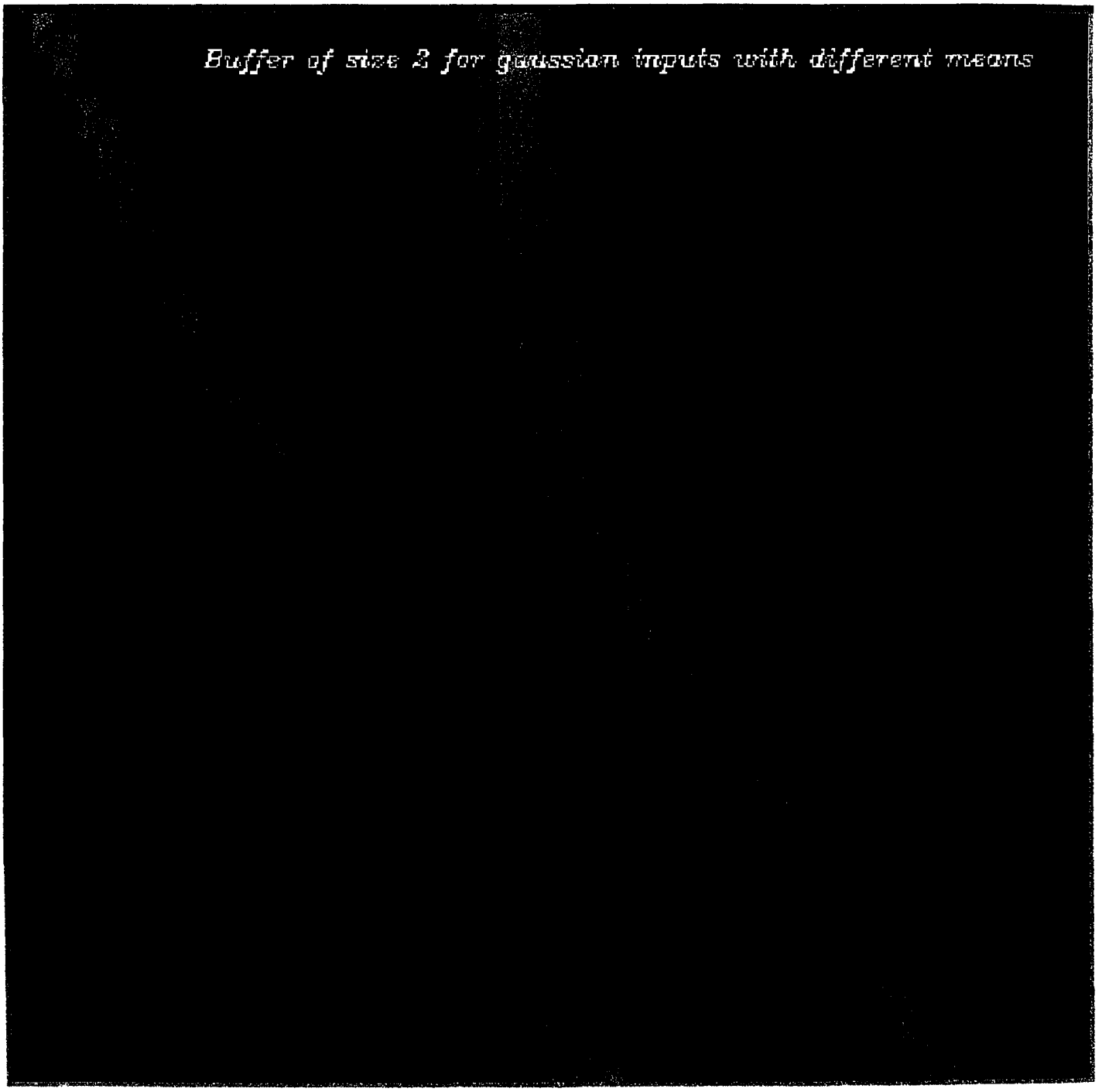

Buffer of size 6 for symmetric gaussian inputs

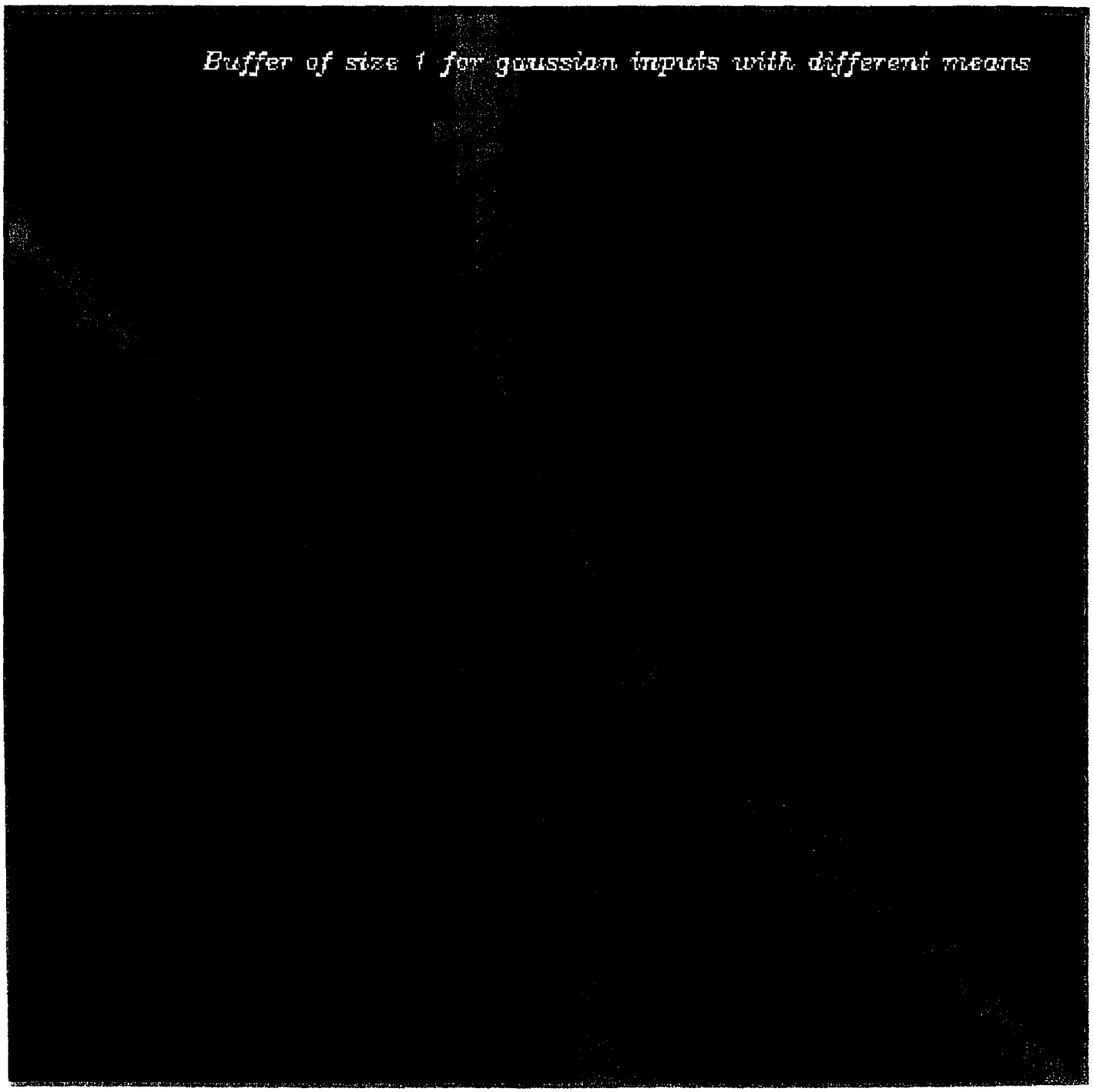

Buffer of size 4 for symmetric gaussian inputs


Buffer of size 2 for symmetric gaussian inputs

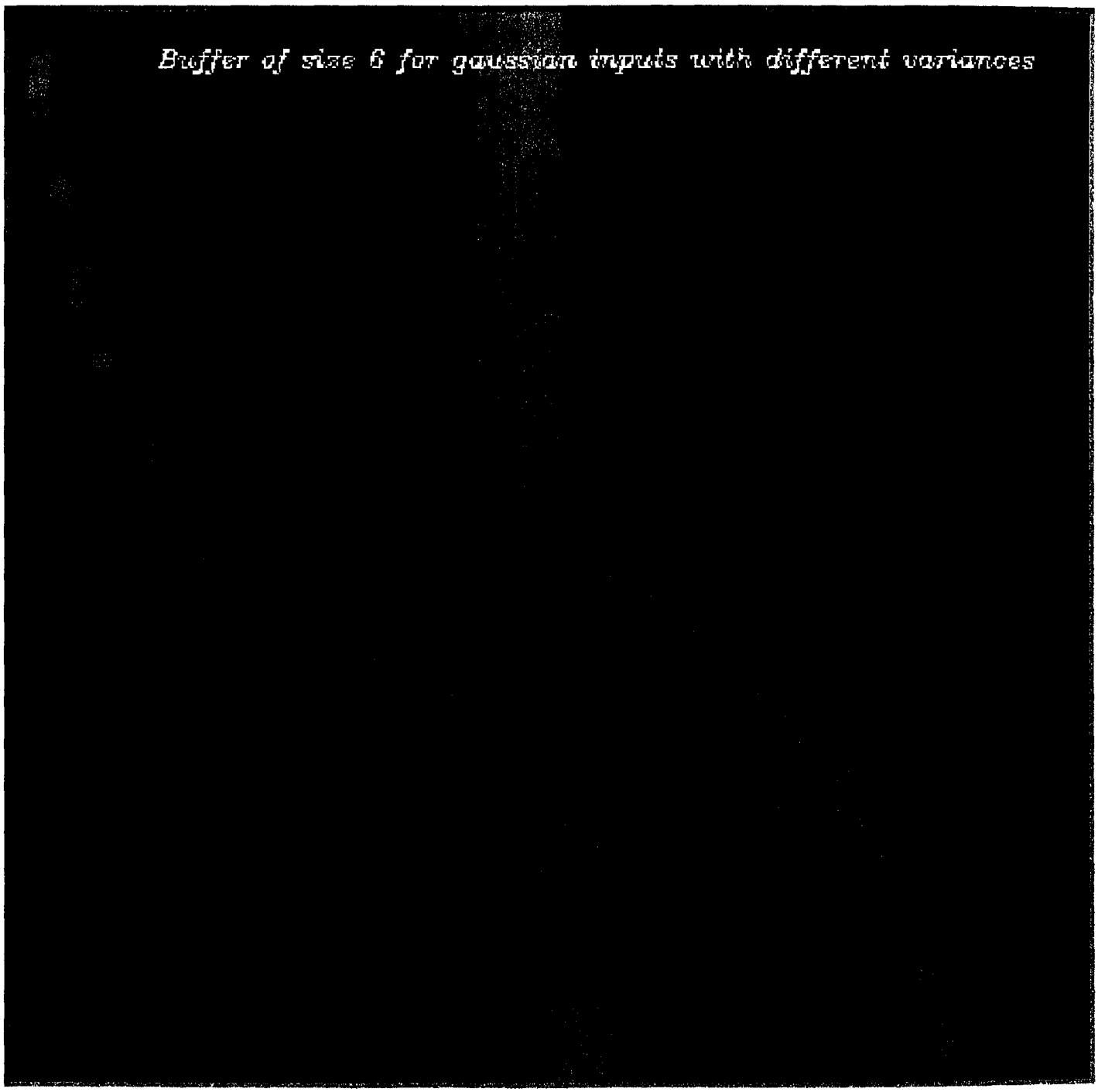

Buffer of size 1 for symmetric gaussian inputs

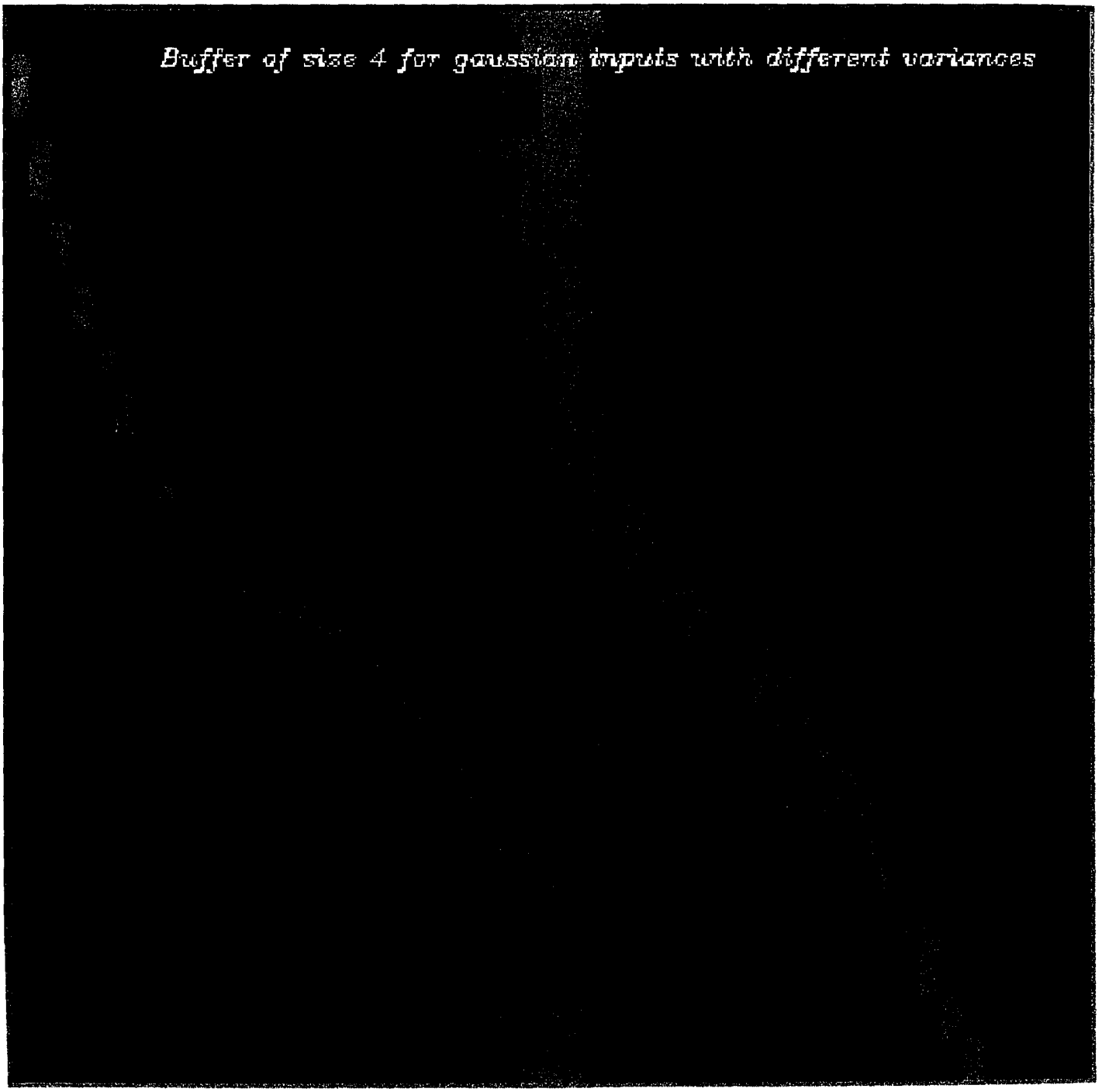

Buffer of size 4 for gaussian inputs with different means

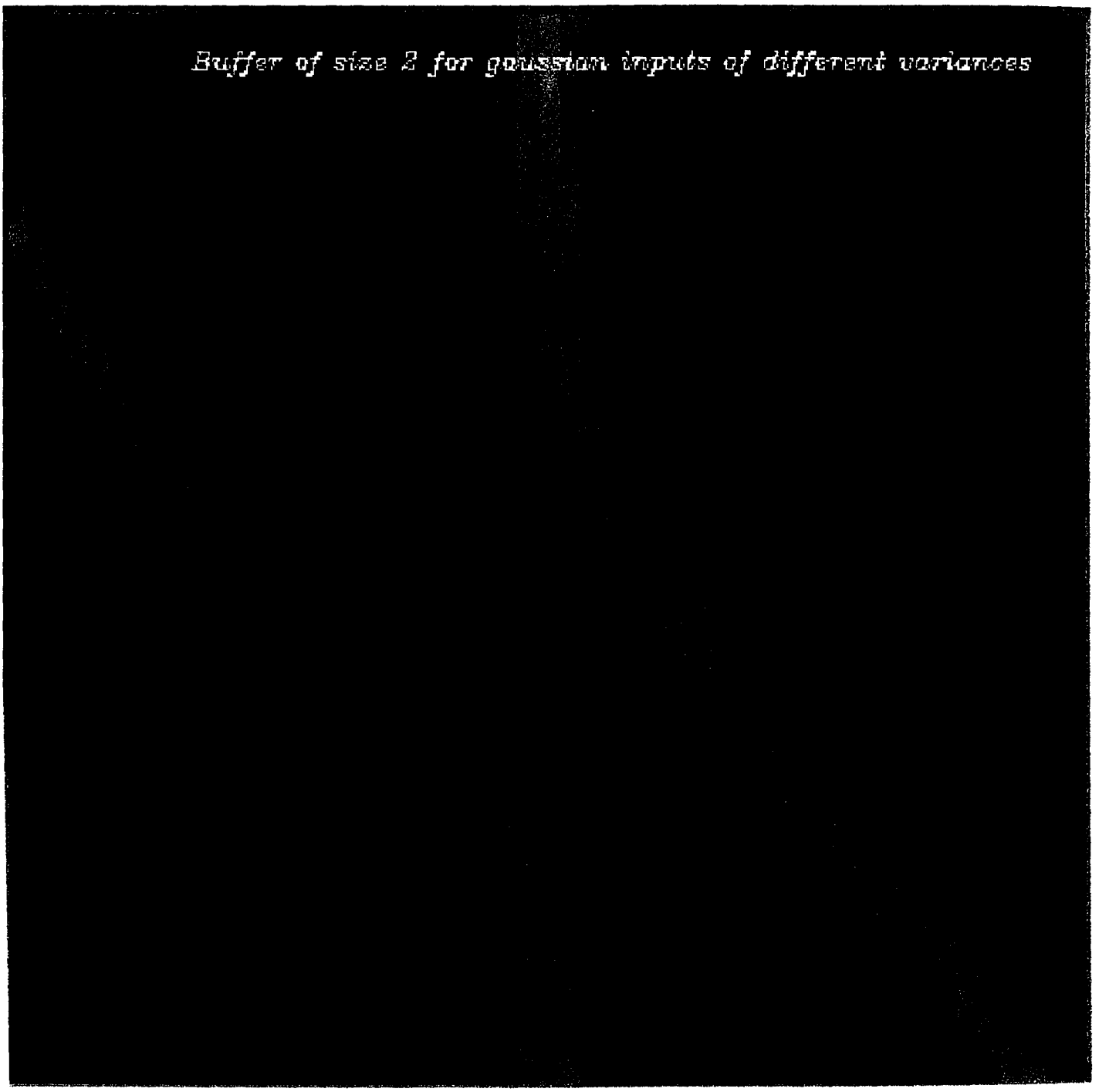

Buffer of size 2 for gaussian inputs with different means

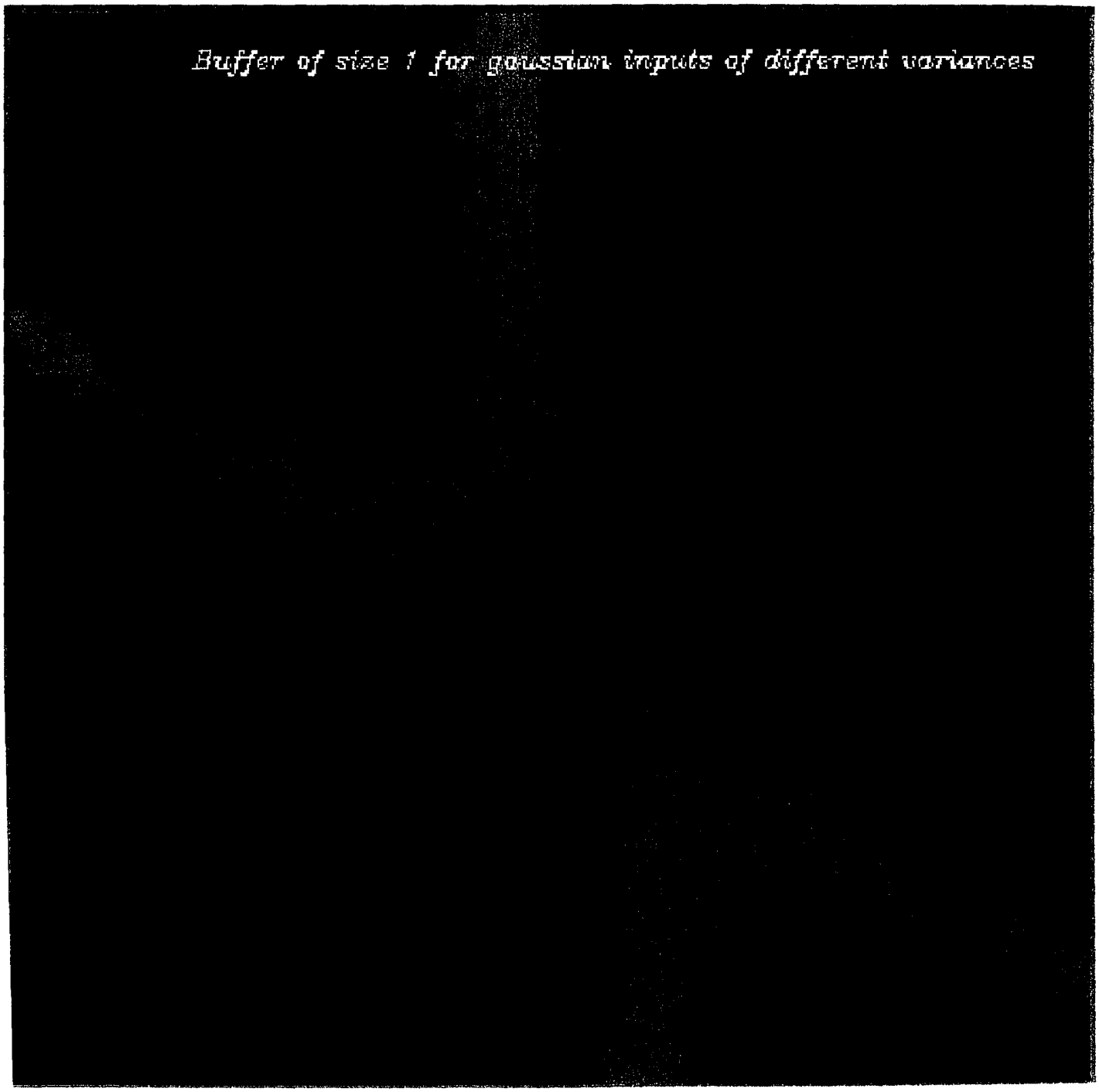

Buffer of size 1 for gaussian inputs with different means


Buffer of size 0.5 for gaussian inputs with different means


Buffer of size 6 for gaussian inputs with different variances


Buffer of size 4 for gaussian inputs with different variances


Buffer of size 2 for gaussian inputs of different variances

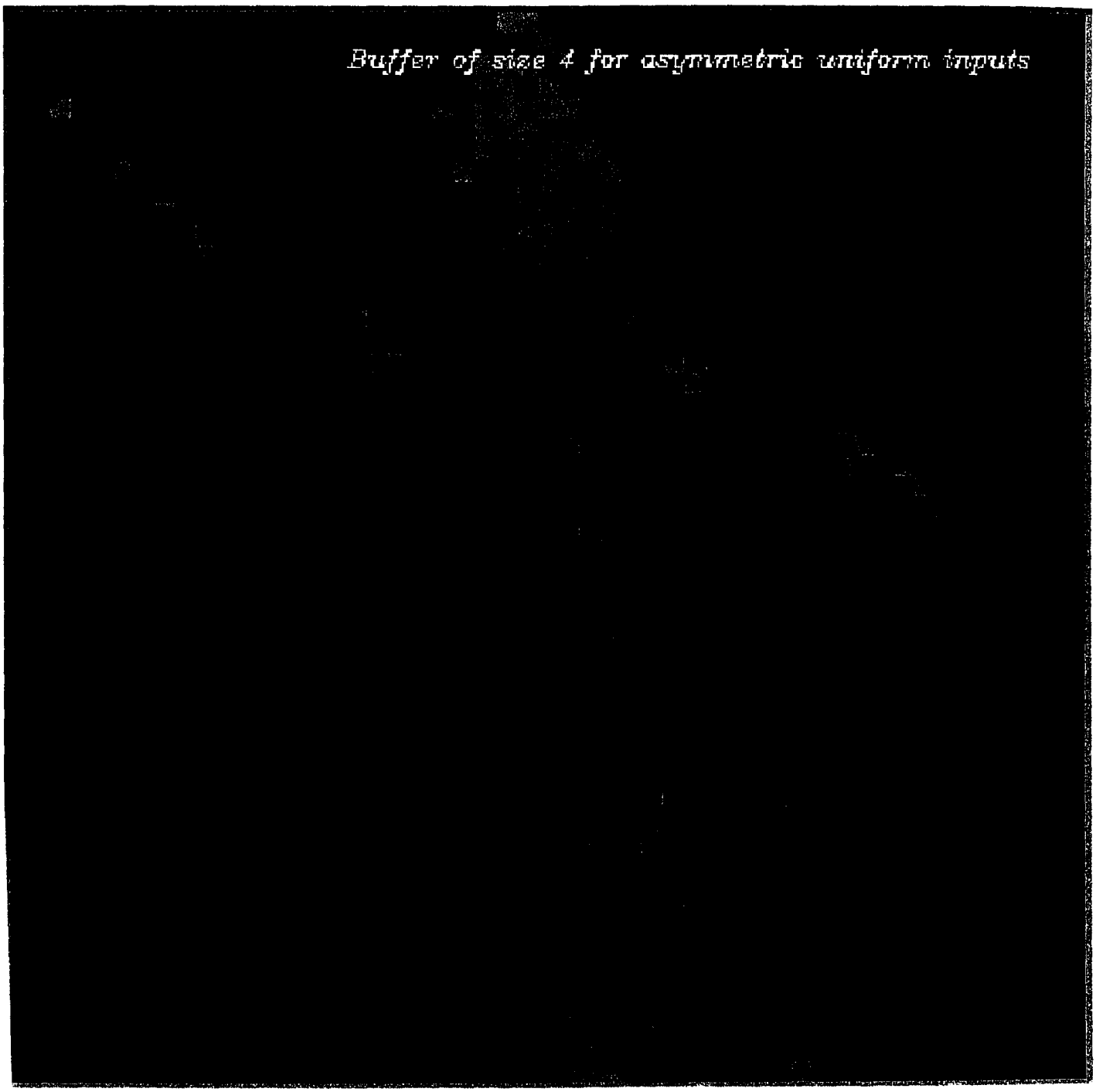

Buffer of size 1 for gaussian inputs of different variances


Buffer of size 4 for symmetric uniform inputs


Buffer of size 2 for symmetric uniform inputs

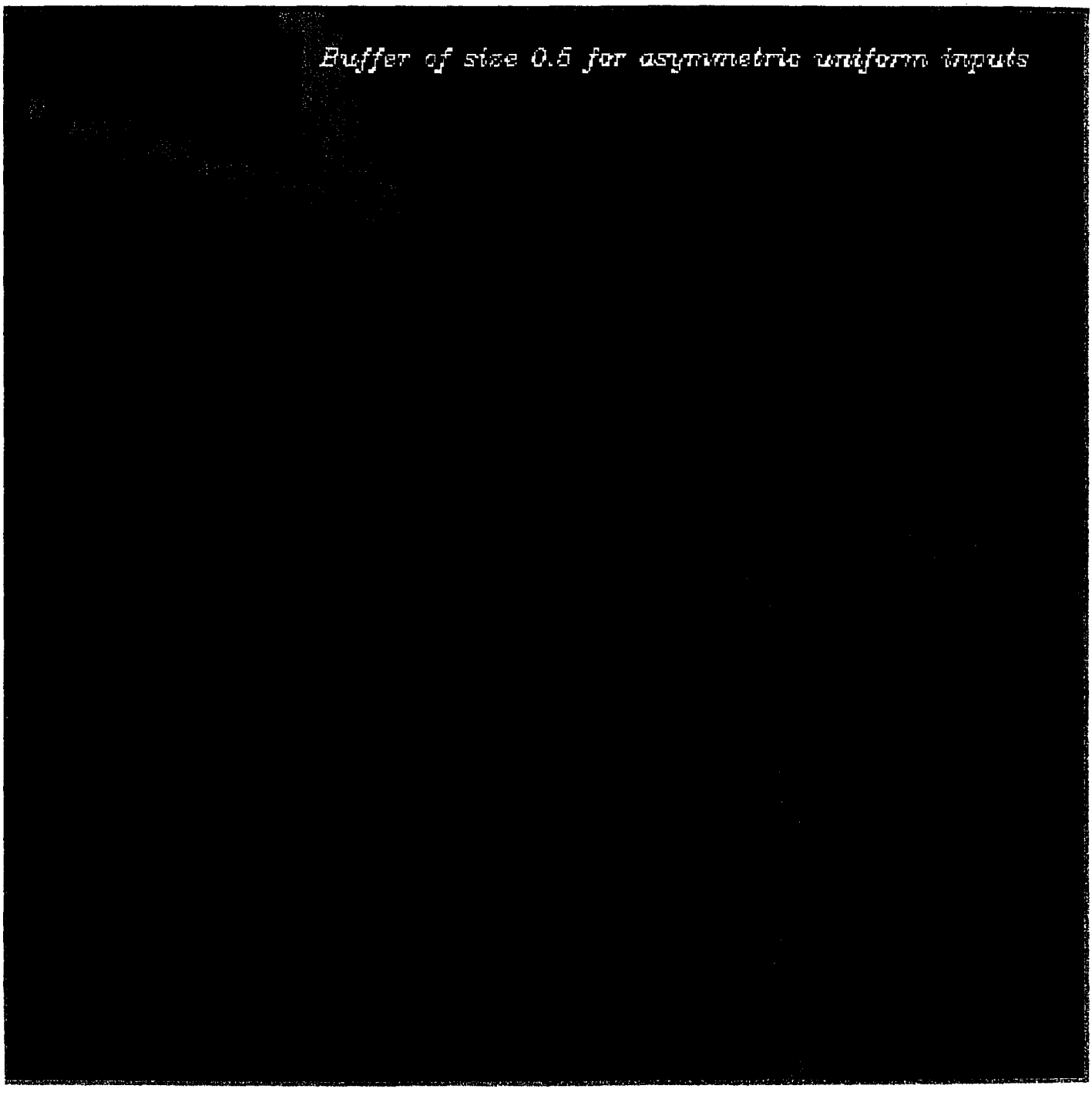

Buffer of size 1 for symmetric uniform inputs


Buffer of size 0.5 for symmetric uniform inputs


Buffer of size 4 for asymmetric uniform inputs

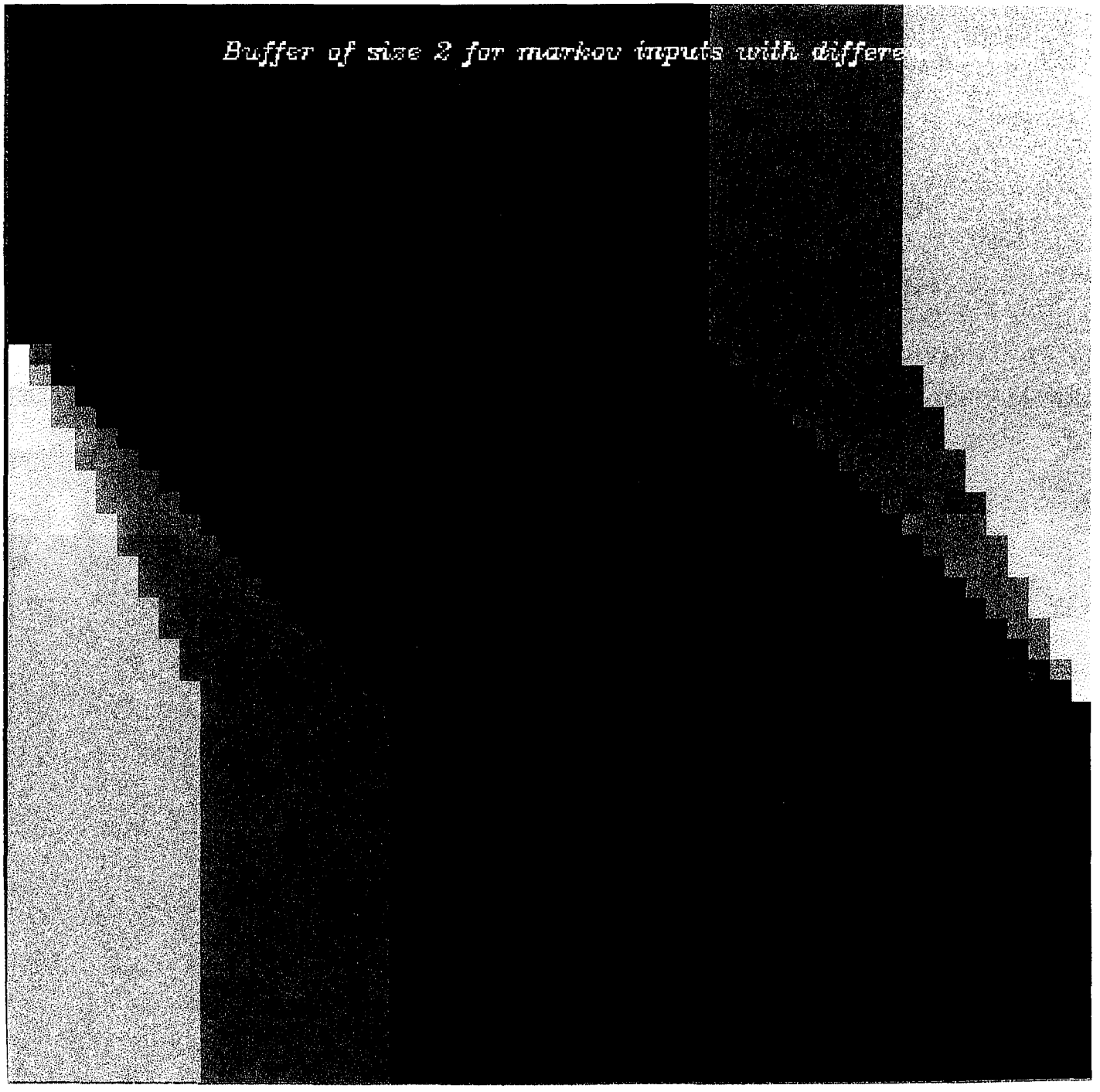

Buffer of size 3 for asymmetric uniform inputs

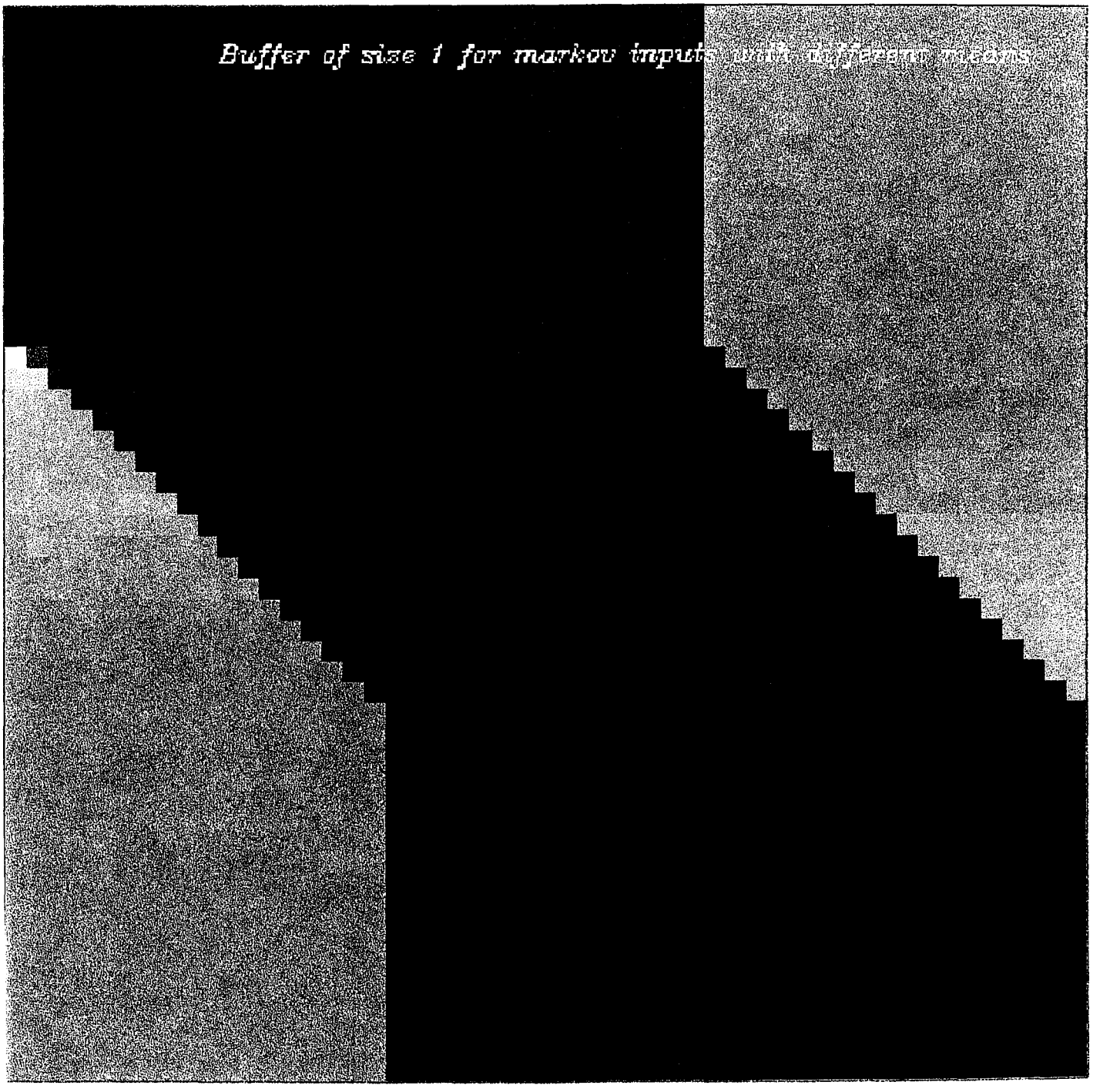

Buffer of size 1 for asymmetric uniform inputs


Buffer of size 0.5 for asymmetric uniform inputs

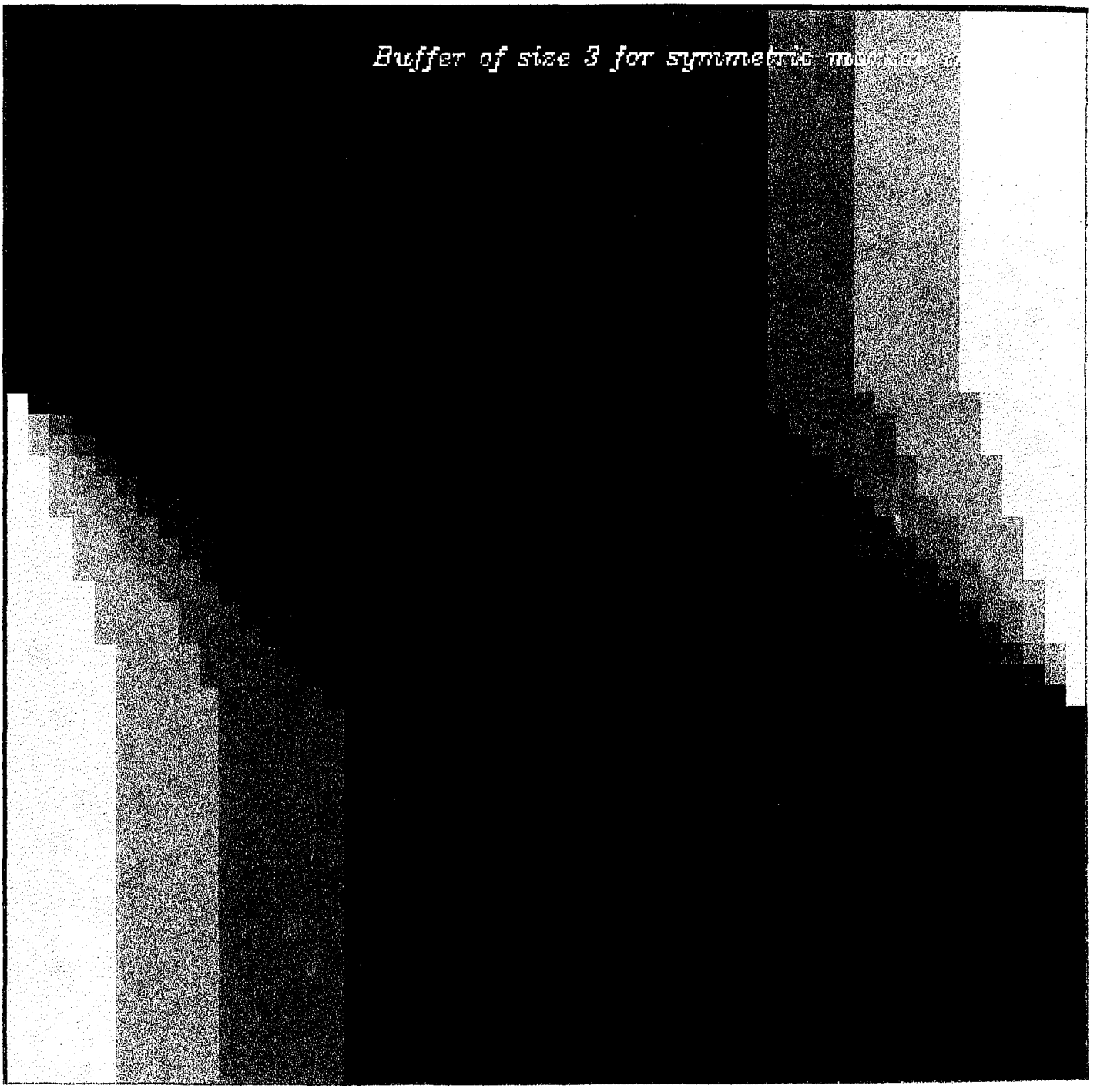

Buffer of size 4 for markov inputs with

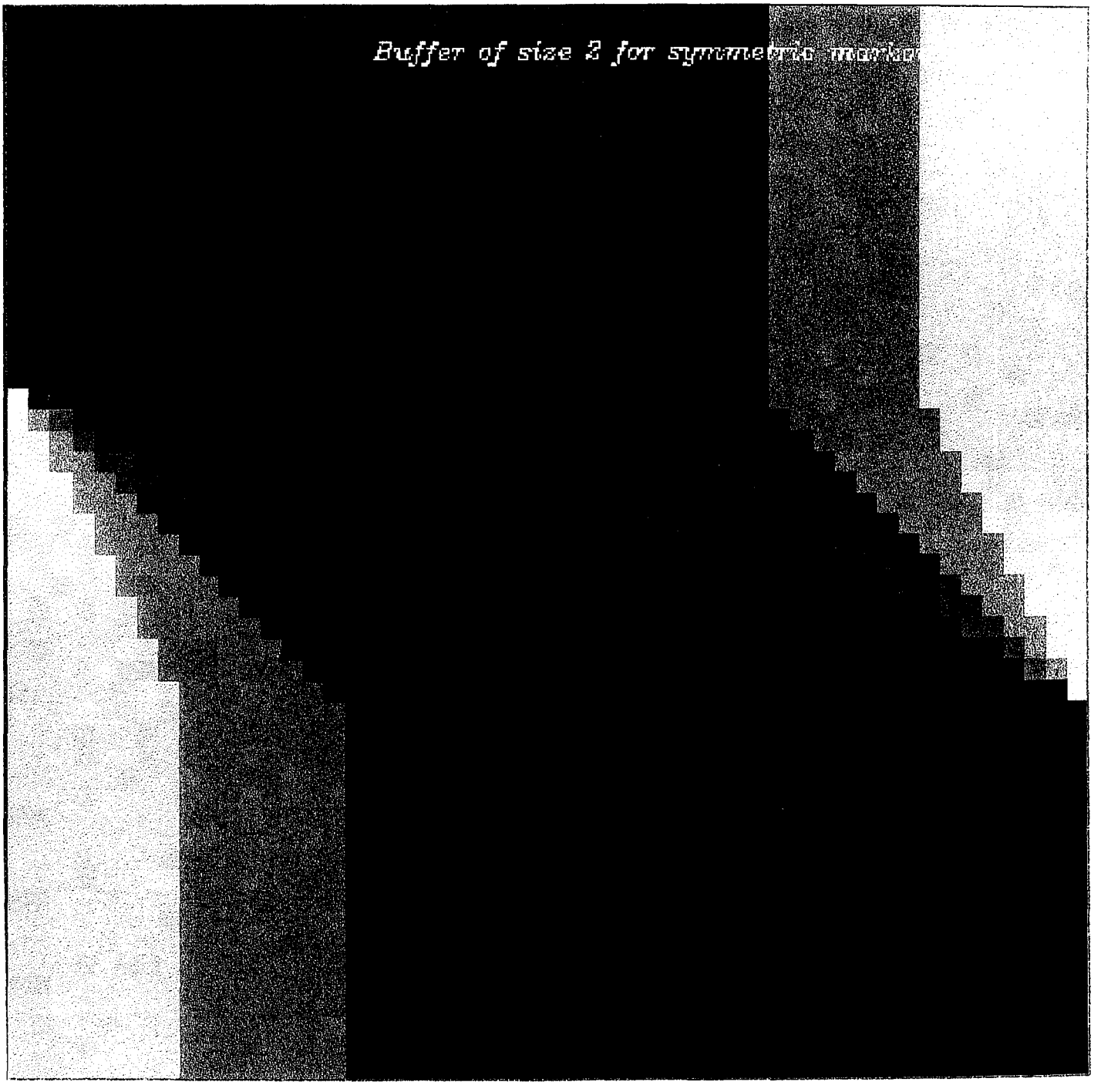

Buffer of size 3 for markov inputs with


Buffer of size 2 for markov inputs with differe

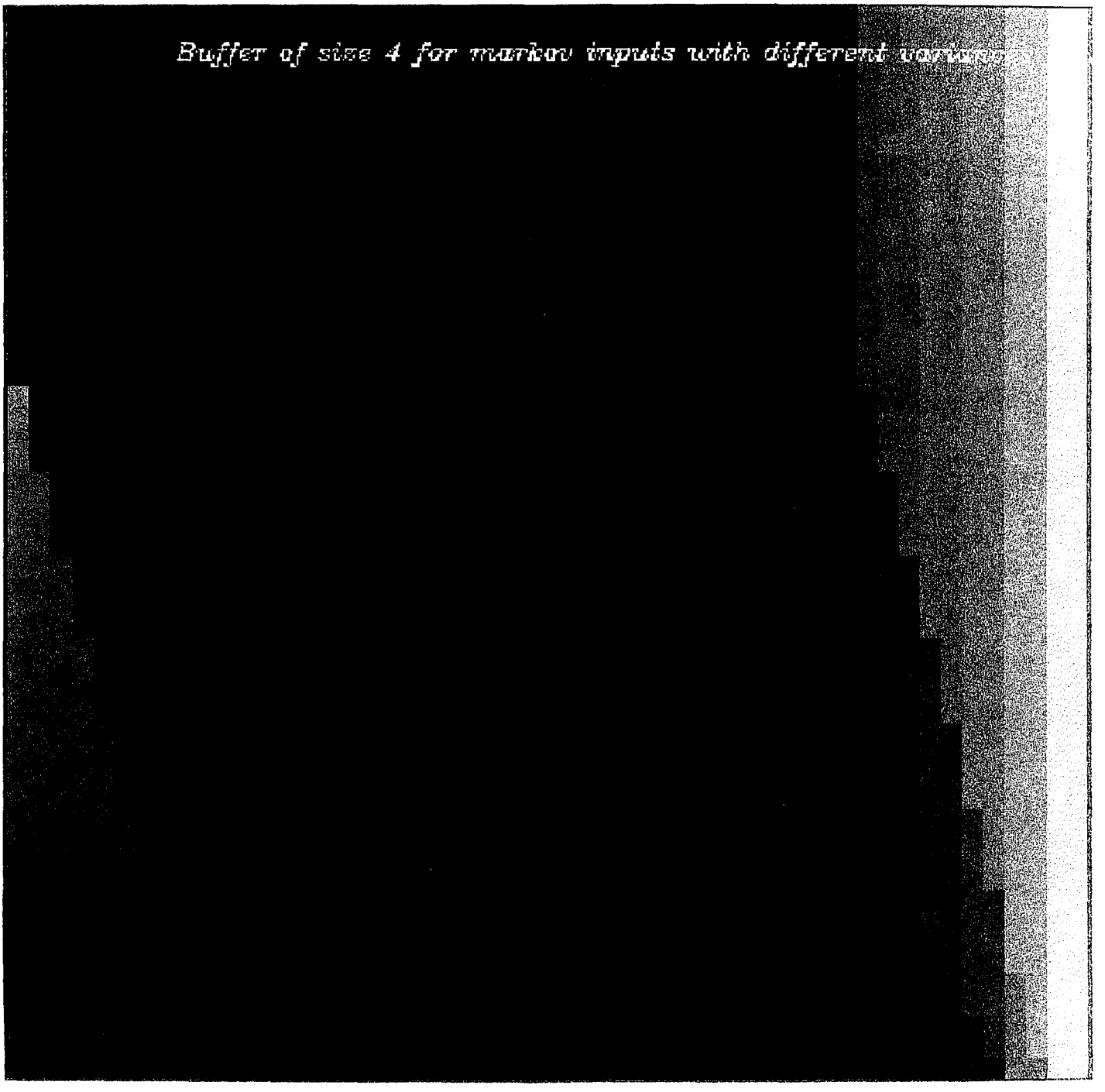

Buffer of size 1 for markov inputs with different means

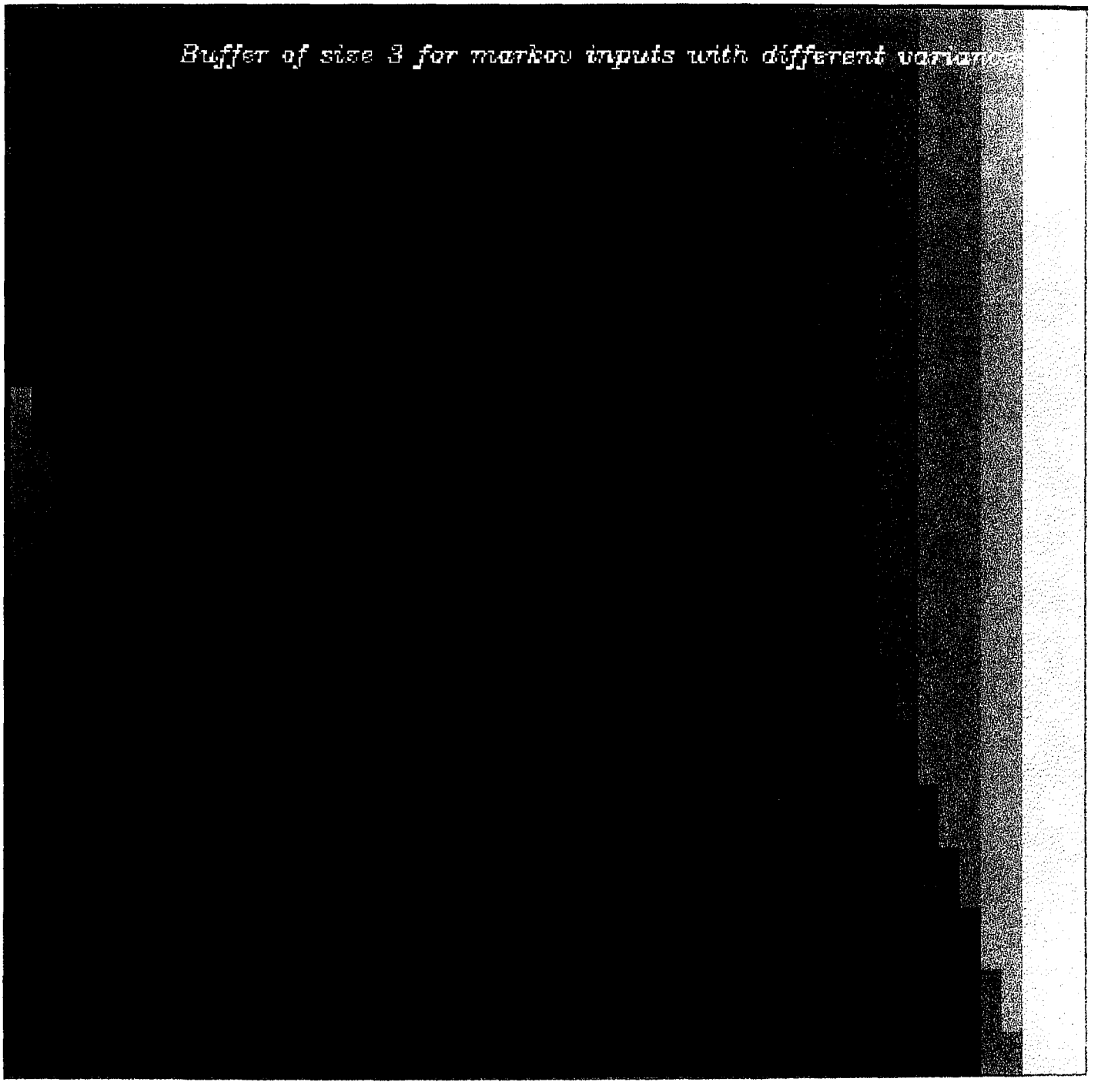

Buffer of size 4 for symmetric

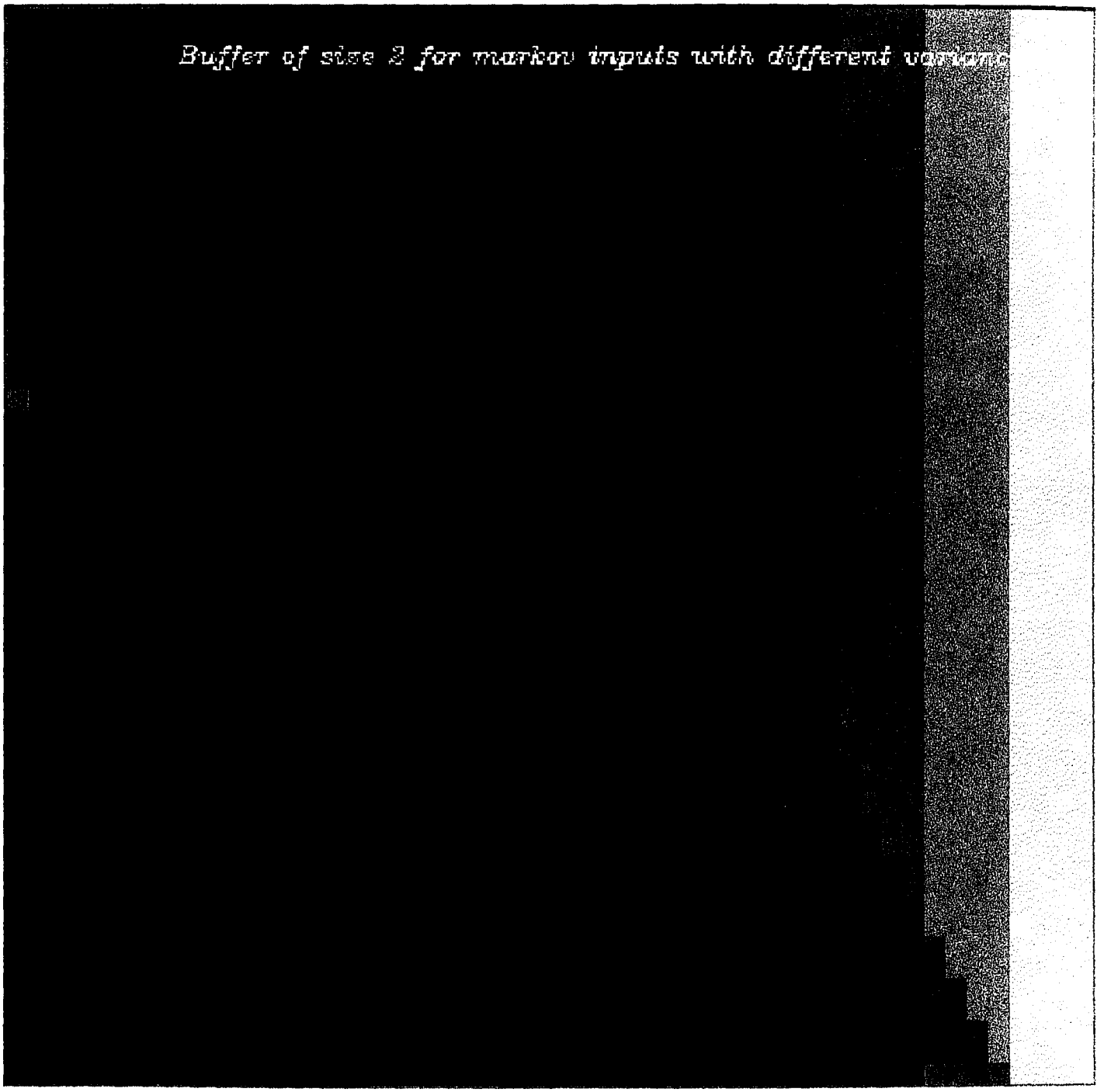

Buffer of size 3 for symmetric


Buffer of size 2 for symmetric


Buffer of size 1 for symme


Buffer of size 4 for markov inputs with different variance


Buffer of size 3 for markov inputs with different variance


Buffer of size 2 for markov inputs with different variance

Buffer of size 1 for markov inputs with different v