Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Marcel Großkinsky

born in Buchen (Germany)

September 2016

Neural Sampling with Linear Feedback Shift Registers as a Source of Noise

This Bachelor Thesis has been carried out by
Marcel Großkinsky
at the
KIRCHHOFF-INSTITUTE FOR PHYSICS
HEIDELBERG UNIVERSITY
under the supervision of
Prof. Dr. Karlheinz Meier

Abstract

On state-of-the-art neuromorphic hardware small LIF networks are able to perform Neural Sampling. It is crucial that the single neurons are supplied with stochastic background noise. This is modeled as a set of spike trains originated from a Poisson process. In the current hardware generation, this Poisson noise has to be generated externally. Alternatively, one has to fall back on Linear Feedback Shift Registers (LFSRs), which are able to generate background noise for small network sizes and short times on chip. This thesis aims to investigate the influence sampling with LFSR generated background noise has. Second, several LFSR architectures are investigated, which are able to generate decorrelated spike trains under the constraint of using as few LFSRs as possible. One approach bases on a kind of Cox Processes: Poisson Processes with Bernoulli trials as stochastic rate. A second approach uses modified Gold-Code generators, which are able to generate background noise for a small LIF neuron network with only two registers and a minor loss in sampling quality.

Zusammenfassung

Auf aktueller neuromorpher Hardware sind kleine LIF-Neuronen-Netzwerke in der Lage, verschiedene Wahrscheinlichkeitsverteilungen zu samplen. Essentiell ist hierzu, dass die einzelnen Neuronen stochastischen Input als Hintergrundrauschen erhalten, das als Poisson Prozess modelliert wird. Momentan kann dieses Rauschen extern erzeugt und auf die Hardware gespielt werden, alternativ kann man auf Lineare Schieberegister zurückgreifen, deren Anzahl allerdings nur kurzzeitig für kleine Netzwerke ausreicht. In dieser Thesis wird der Einfluss untersucht, den das Samplen mit Schieberegister-Rauschen hat. Weiterhin werden verschiedene Möglichkeiten getestet, mehrere Schieberegister zu verschalten, um platzsparend dekorreliertes Rauschen zu erhalten. Ein Ansatz basiert auf einer Form von Cox-Prozessen: Poisson Prozessen mit Bernoulli Versuchen als stochastische Rate. Ein zweiter Ansatz verwendet veränderte Gold-Folgen Generatoren, die es mit geringen Einbußen in der Sampling-Qualität ermöglichen, mit nur zwei Generatoren Hintergrundrauschen für kleine Netzwerke von LIF-Neuronen zu generieren.

Contents

1 Introduction

The human brain consists of over 20 billion neurons, each of which has about 7000 synaptic connections to other neurons. Therefore, the brain is the most complex of the human organs and its mode of operation is still not completely understood. First theoretical models of single neurons go more than a hundred years back in time [Lapicque (1907)]. Since then, more natural and complex models of neural structures have been developed, which try to explain the inherently stochastic dynamics of neural networks. The *sampling hypothesis*, a relatively recent paradigm, states that the neural activity can be rather seen as drawing samples from probability distributions instead of computing these analytically. Buesing describes stochastic firing activity of neuron networks via Monte-Carlo Markov Chain sampling[Buesing et al. (2011)]. Based on his work, Petrovici et al. have developed a Neural Sampling model for leaky integrate-and-fire neurons [Petrovici et al. (2016)], which are able to sample from an underlying stationary probability distribution.

In recent years, scientific focus lies not only on the theoretical description of neural network dynamics but also on neuromorphic hardware. In this approach, circuits are built which correspond to analog neurons in silicon. Analog emulation of neurons and synapses has two major advantages: this method is less energy consuming and much faster than simulating neural structures on supercomputers. As part of the BrainScaleS project [BrainScaleS (2016a)], High Input Count Analog Neural Network (HICANN) chips are developed at the Kirchhoff-Institute for Physics. These are able to emulate neurons, which can be used for Neural Sampling as described in [Petrovici et al. (2016)].

On the current HICANN, the single neurons have to be externally supplied with a high-frequent background noise input for a Neural Sampling experiment. This background noise makes their membrane potential stochastic and is modeled as Poisson Process. The long-term goal is to generate this background noise on the HICANN chips themselves. Therefore, the use of linear feedback shift registers (LFSRs) as pseudorandom number generators (PRNGs) is unavoidable, as they have a simple structure but are able to produce random numbers efficiently in contrast to other methods. This thesis investigates the influence the of using deterministic PRNGs for Neural Sampling and evaluates possible LFSR architectures for creating efficient pseudorandom numbers using as little resources as possible.

The first part of this thesis deals with the theoretical fundamentals underlying the discussed concepts and ideas. Especially the generation of Gold

codes – a set of binary sequences with very low cross-correlations – is described.

The second part details the sampling results of the different experimental setups and begins with a short discussion of Cox process generated spike trains in section 3.1. Second, a detailed investigation of sampling using LFSR generated spike trains in general is given in section 3.2. Section 3.3 deals with the first of two presented LFSR architectures: registers run with increased rate and their spikes are split among different neurons. Finally, a setup based on Gold sequences is investigated in section 3.4.

2 Theory

2.1 Neural Sampling

One of the most important distributions is the Boltzmann distribution: Let $\mathbf{z} = (z_0, z_1, ..., z_{n-1})$ be one particular state of a system, where the probability of the state is given as:

$$p(\mathbf{z}) = \frac{1}{Z} \exp[-E(\mathbf{z})] \tag{1}$$

$Z = \sum_{\mathbf{z}} \exp[-E(\mathbf{z})]$ is the normalization constant and $E(\mathbf{z})$ is the energy associated with the state. For a system of binary units, such that $z_i = 0$ or 1, an energy can be defined:

$$E(\mathbf{z}) = \exp\left(-\frac{1}{2}\sum_{i,j} W_{ij} z_i z_j - \sum_i b_i z_i\right) \tag{2}$$

The energy of a state is basically a summation over all active ($z_i = 1$) weights W_{ij} and biases b_i, where the weight matrix W has two constraints: first, it is symmetric ($W_{ij} = W_{ji} \, \forall i, j$) and second, its diagonal is zero ($W_{ii} = 0 \, \forall i$). The system $\mathbf{z}$ can now be described as Boltzmann machine, a network, where the single z_i denotes the network nodes. The bias of z_i is denoted as b_i and the weight W_{ij} denotes the connection strength of z_i and z_j.

To enable a network of neurons to sample from a distribution $p(\mathbf{z}) = p(z_0, z_1, ..., z_{n-1})$ one has to specify an assignment of spiking activity to a set of random states $\{z\}$. A detailed derivation of the formalism can be found in Buesing et al. (2011) and Petrovici et al. (2016), the following section gives only a very rough idea of Neural Sampling[1].

The link between neural activity and sampling from arbitrary random distributions over binary variables is called *neural computability condition* (NCC):

$$u_k(t) = \log \frac{p(z_k = 1 | z_{\setminus k})}{p(z_k = 0 | z_{\setminus k})} \tag{3}$$

where $u_k(t)$ is the abstract membrane potential of neuron k and $z_{\setminus k}$ is the set of all z_i with $i \neq k$. Immediately after a spike, a neuron enters its

[1] Especially, the difference between biological weights used as neuron connection strength and theoretical weights used for the calculation of the DKL (see eq. (6))is not mentioned. Otherwise, a detailed discussion of calibration and weight translation would have had to follow, which is beyond the scope of this thesis.

refractory state for a time interval τ_{ref}, where it is unable to spike again. For a neuron in this state, $z_k = 1$, otherwise $z_k = 0$: We associate a binary variable with the state a neuron is in and sample using these random variables. Inserting the probability of the Boltzmann distribution and the energy defined above one finds:

$$u_k(t) = b_k + \sum_j W_{kj} z_k \tag{4}$$

This corresponds to the membrane potential of a leaky integrate-and-fire neuron [Lapicque (1907), Gerstner and Kistler (2002)], which is the differential equation given in eq. (5).

$$C_m \frac{du(t)}{dt} = g_L(E_l - u) + I \tag{5}$$

C_m denotes the membrane capacity and g_L the leak conductance of the LIF neuron. I is the input current for the specific neuron and can be split in inhibitory, excitatory, recurrent and external input currents. Besides, one can define the membrane time constant $\tau_m = \frac{C_m}{g_L}$, which quantifies the reaction speed of the membrane potential to stimuli. Whenever the membrane potential rises above the threshold potential v_{thr} a spike is emitted and the potential is set and kept at the reset potential v_{rest} for a time period τ_{ref}, in which it is unable to spike again.

The external input current relates to the bias term in eq. (4), while the inhibitory and excitatory inputs refer to the weight term and describe the input a neuron receives from the neurons its connected with.

In addition to the spikes the neurons receive due to their synaptic connections with other neurons, every neuron is fed with background noise. This is modeled as Poisson process generated spike times with a frequency in the order of a few hundred Hertz. This thesis discusses the replacement of such Poisson noise by LFSR generated spike times without a loss in sampling quality.

Sampling quality is measured using the Kullback-Leibler divergence. The DKL value of two probability distributions is defined as:

$$DKL(p, q) = \sum_{\mathbf{z}} p(\mathbf{z}) \ln \frac{p(\mathbf{z})}{q(\mathbf{z})} \tag{6}$$

It quantifies the dissimilarities between the two distributions p and q. In the case under consideration q is the theoretical distribution (calculated using eq. (1)) and p is the sampled distribution, which one gets by summing over all states the LIF network was in during sampling time.

2.2 Poisson and Cox Process

A Poisson process (PP) counts the number of events in a certain time interval t. The single events are independent and identically distributed. The number of events in a certain time interval follows a Poisson distribution.

We call the average number of events per time unit ν, while the probability of N events occurring in the time interval T is denoted as $P_N(T)$. We start with the probability of a single event in dt, which is $P_1(dt) = \nu dt$. Consequently, $P_0(dt) = (1 - \nu dt)$ is the probability that no event occurs.

Since the single events are independent, we can calculate the probability of no event in $T + dt$ by taking the product of the single probabilities, which leads to:

$$P_0(T + dt) = P_0(T)(1 - \nu dt) \tag{7}$$

After some more calculations, see for example [Kingman (1995), Cowan (2016)], it is possible to formulate a closed-form-solution for the probability of n events occurring in the time interval T:

$$P_n(T) = \frac{\exp(-\nu T)(\nu T)^n}{n!} \tag{8}$$

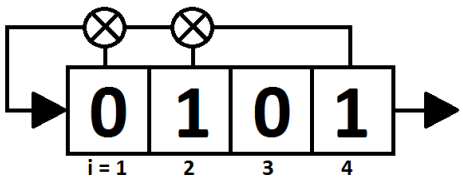
Both the expectation value and the variance of this distribution is νT.

In 1955 D. Cox introduced *doubly stochastic Poisson processes* [Cox (1955)], which are nowadays given the name of their discoverer. The term *doubly stochastic* stresses their characteristic feature: a Cox process (CP) is a nonhomogeneous Poisson process which rate is not only time-dependent but itself a stochastic process again. The expectation value at time t is now given by $\int_0^t \nu(\tau) d\tau$.

The focus is on using Bernoulli trials as stochastic process. If each event occurring with rate r is assigned to one of two classes with probability p and $1 - p$, respectively, the generation of these two classes by these Bernoulli trials is equivalent to their generation by a Poisson process with rate rp and $r(1 - p)$, respectively. This equivalence is exploited for later LFSR architectures. A more detailed proof following Siegrist (2016) is given in section 5. The implementation of the process follows Burnecki and Weron (2010) and is explained there in more detail.

2.3 Linear Feedback Shift Register (LFSR)

Linear feedback shift registers are commonly used pseudorandom number generators, which produce a strictly deterministic series of numbers by linear recoupling. For a very detailed discussion, see [Golomb (1981)].



timestep	state	feedback
0	0101	0
1	0010	0
2	0001	1
3	1000	1
4	1100	0
...		

Figure 1: Left: Linear Feedback Shift Register of length $n = 4$. Feedback taps are 1,2 and 4. Right: first five states of the generated sequence with initial seed $s = 0101$.

Starting with a seed, some specific bits of the register called (feedback) taps are XORed, basically a modulo-2 summation, which gives as result a 0 or a 1. The current number is shifted one bit to the right such that the last bit drops out and the calculated bit becomes the new first bit. A sketch and an exemplary sequence for a LFSR of length $n = 4$ is given in fig. 1.

Due to that construction, every LSFR has a finite number of different states which are run through. Since a zero-state would result in a the zero-state again because the result of the summation is always a 0, a LFSR of length n can at most generate a sequence of $2^n - 1$ different numbers. Not all possible tap configurations lead to sequences of the maximum length (maximum or m-sequences): the feedback taps can be interpreted as a polynomial, for example the LFSR in fig. 1 has the polynomial $x^4 + x^2 + x + 1$. If the polynomial is primitive, the generated sequence is a m-sequence.

As stated in Golomb (1981), m-sequences fulfill the three conditions in order to be a pseudo-noise sequence:

Balance Property

The number of ones in the complete sequence is exactly one greater or smaller than the number of zeros.

Run Property

A *run* is a subsequence of zeros or ones in the m-sequence. The following conditions have to hold:

- $\frac{1}{2}$ of the runs are of length 1
- $\frac{1}{4}$ of the runs are of length 2
- $\frac{1}{8}$ of the runs are of length 3
- ...

Moreover the total number of runs of zeros and ones equals each other.

Correlation Property

The linear autocorrelation function of the m-sequence approximates a Kronecker delta function and is two-valued.

2.4 Neuromorphic Hardware and Spike Generation on LFSRs

The HICANN chip is the primary building block for the wafer-scale system and consists of 128,000 synapses[BrainScaleS (2016b)]. 512 on-chip membrane circuits can emulate neurons directly, which limits their number of synapses to 256, or be grouped together to form neurons with up to 16,000 synapses .

On the current HICANN, eight 16-bit-LFSRs can be used. 16 bits result in a m-sequence of 65535 states, which are cycled through every 3.28 s (biological time) assuming a 5 ns system clock (hardware time) and a speedup factor of 10^4. One state of the LFSR corresponds therefore to a timebin tb of 0.05 ms biological time. Other clock rates are also possible [Gruebl (2016)].

The generated sequence of states can be used as spike generator by the following way: interpret the state s as binary number b and spike in the specific time-interval t if

1. $b_t > \theta$
2. $b_{t-1} < \theta$

where the threshold θ determines the firing rate ν and the second condition has to hold for technical reasons. One complete cycle of the LFSR produces at most $2^n - 1 - \theta$ spikes, if only the first spiking condition is taken into account. Therefore the spiking probability of one particular state is $p(s) = \frac{2^n-1-\theta}{2^n-1} = 1 - \frac{\theta}{2^n-1}$. During the total sampling time T, $N = T/tb$

states are run through. That means there are $N \cdot p(s)$ spikes generated and the generating rate gets $\nu = \frac{N \cdot p(s)}{T} = \frac{1}{tb}(1 - \frac{\theta}{2^n - 1})$. We can adjust an appropriate threshold by choosing $\theta = (2^n - 1) \cdot (1 - 2 \cdot tb \cdot \nu)$. Since due to the second condition spikes are discarded, if they immediately follow an other spike, one has to decrease the threshold. Then more spikes are generated and it does not matter if some of them are discarded. This decreasing is done by inserting a factor of two. This does not cancel the effect of the second condition exactly, but the absolute value of the frequency is of lower interest as long as it is in the right order of magnitude.

2.5 Gold Sequences

Gold sequences [Gold (1967)] are commonly used in Code Division Multiplex techniques, e.g., for the GPS system or for UMTS. The basic idea is to generate sets of sequences with low cross correlations by using two connected LFSRs. The concrete structure can be seen in fig. 2.

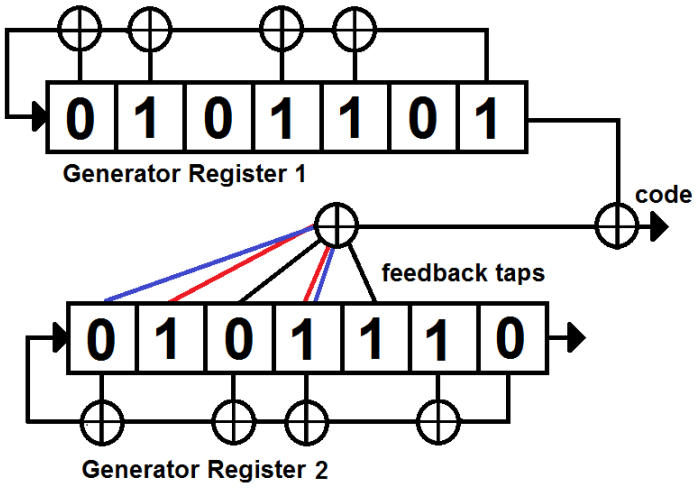


Figure 2: Gold sequence architecture: the last bits of two LFSRs with different tabs are XORed and result in one code bit. In this sketch the feedback tabs for different Gold sequences are colorcoded.

For the Gold code creation the last bits of two LFSRs of the same length n are XORed. The resulting sequence has a period length of $2^n - 1$, which is the same period lengths as the single m-sequences have. The use of a second LFSR, which does not extend the m-sequence is justified by the fact that one can generate up to $2^n - 1$ m-sequences almost in parallel by using

the last bit not directly but an offset bit determined by additional feedback taps. Due to a comparable small additional effort of some feedback taps, one gets a complete m-sequence. Before one had to use an additional LFSR. The sketch shows the architecture for Gold code generation, which has to be slightly modified if the LFSRs are used as spike generators: a spike is emitted for neuron i if the upper LFSR is above the threshold and the resulting bit equals 1.

2.6 Characteristic Values of Spike Trains

Since the background noise spike trains are the main object of investigation, in this section the focus is on several important characteristic values, which can be calculated from them. The first value is the correlation value of interspike intervals, which gives a measure of the regularity of the spike times. Second the auto- and crosscorrelation of spike trains is important since both are characteristically related to LFSR sequences. Third, the KS test measures the *poissonity* of a spike train.

Correlation Value of Interspike Intervals

For every given set of interspike intervals (ISIs) their correlation value is defined as

$$CV_{ISI} = \frac{\sigma(ISI)}{\overline{ISI}} \tag{9}$$

where $\sigma(ISI)$ is the standard deviation and $\overline{ISI}$ the mean of the ISIs, respectively. For a perfectly regular spike train CV_{ISI} is zero, since the standard deviation is zero, too. For a Poisson spike train of sufficient length the correlation value gets one, because the standard deviation equals the mean.

Auto- and Crosscovariance Functions (ACF and CCF)

Since high correlations will appear in LFSR generated spike trains due to their construction, both autocovariance and crosscovariance can be an important measure of their characteristics.

We use the definition of the autocovariance function given in Brockwell and Davis (2006): if $X_t, t \in T$ is a process such that $Var(X_t) < \infty \, \forall \, t \in T$, then the autocovariance function $\gamma_X(.,.)$ of X_t is defined by:

$$\gamma_X(r,s) = Cov(X_r, X_s) = E[(X_r - E[X_r])(X_s - E[X_s])] \, \forall r, s \in T \tag{10}$$

The same formula holds for the crosscovariance function, if one replaces the second X_s with Y_s, which denotes a second process. In this thesis, the

term *process* refers to series of interspike interval times, where the time index in the equation denotes the index of the ISI.

Note that the autocovariance function – in contrast to the autocorrelation function – is not normalized to the interval $[-1, 1]$. We do not use the latter one, since we are also interested in the absolute magnitude of the correlation and therefore a normalization would make no sense.

Kolmogorov-Smirnov Test

The two-sample KS test presented by Smirnov and Kolmogorov (Smirnov (1939)) is used to estimate the probability for two data samples to obey the same probability distribution. We want to use this test for a direct determination of the quality of different background spike trains without the need of performing Neural Sampling and using the DKL value as evaluating measure. Therefore we investigate, if the KS test result and the final DKL value correlate. Then the test can serve as a good measure. In our case we compare two spike trains generated by different sources. If the KS test provides the result that the two data samples follow the same distribution we expect a good sampling result. Otherwise, if the hypothesis has to be rejected, the final DKL-values should be higher.

The empirical distribution function (edf) F_n for n independently identically distributed random variables is defined as:

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \chi_x(X_i) \tag{11}$$

where χ_x denotes the indicator function: $\chi_x = 1$ if $X_i \leq x$, otherwise it's 0. For a value x the edf value $F_n(x)$ is the proportion of random variables lower or equal to x.

The KS statistic D for two empirical density functions $F_m(x)$ and $F_n(x)$ is

$$D = \sup_x |F_n(x) - F_m(x)| \tag{12}$$

Basically, this test measures the maximum distance between the empirical distribution function $F_n(x)$ calculated from the data and reference cumulative distribution function or from two edfs, respectively. An example is given in fig. 3.

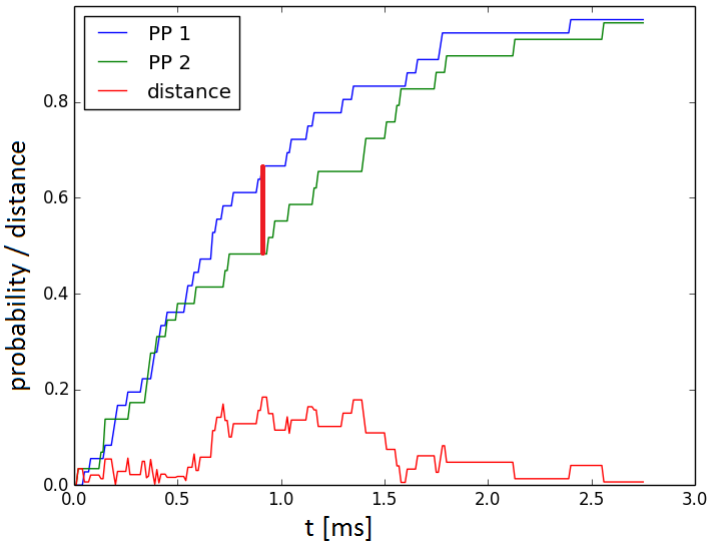


Figure 3: Example of the Kolmogorov-Smirnov Test. The blue and green lines are the edfs of two different Poisson processes. The red vertical line equals the test statistic D and determines the final p-value.

The red line equals the test statistic and determines the pvalue of the test, which states the probability of the test result arising by chance from noise. In this case, the null hypothesis states that both spike trains come from the same probability distribution. If this value is large, one therefore has to reject the null hypothesis.

3 Results

In the following sections the results of the performed experiments are discussed. During all setups LIF neurons were used and their parameters like v_{rest} or v_{thr} were left unchanged. The network consists of five neurons, where not stated differently, though some experiments were repeated with ten LIF neurons, too. As the results of the two network sizes do not differ significantly, they are not shown in this thesis.

First, every experiment begins with a calibration of the neurons for 100 s, after that the neurons sample ten different distributions for 1000 s each. Both during calibration and sampling the neurons are supplied with the discussed spike trains. The weights and biases of each distribution are drawn randomly. Please find the detailed experimental setup in section 5.

Section 3.1 deals with Neural Sampling performed with CP spike times. Afterwards section 3.2 gives a very detailed discussion of sampling with LFSR spike times. The problems and their possible improvements or solutions are presented in section 3.3 and section 3.4, where two possible LFSR architectures are investigated.

3.1 Sampling Based on Cox Processes

Doubly stochastic Poisson processes are characterized by the random process which generated the single samples. We investigate the split property. Each of the PP spike times is therefore put into one of several subtrains. The PP runs with multiplied intensity, which ensures that every single spiketrain in the resulting set has the appropriate rate.

3.1.1 Splitting of Spike Trains by a Bernoulli Process

In this section, we perform the next step from theoretical Neural Sampling using split Cox Process (CP) spike trains towards pure LFSR spike trains.

A PP spike train S with doubled intensity is divided into two subtrains S_0 and S_1 such that both S_0 and S_1 have the same mean frequency of 1000 Hz. This means, that each spiketime in S is assigned randomly to S_0 or S_1. There are several possibilities how to perform the assignment. One could draw a uniformly distributed random number u and assign the spike s_i to S_0 if $u <= 0.5$. Since we want to find a method to perform this split using LFSRs, we use a different method. We create random numbers which have the values 0 and 1 with probability 0.5 each. These values correspond to class labels, and S_0 (S_1) is just the set of all spikes with class label 0 (1).

This method has another advantage: if we do not only use a spike train generated with doubled rate, but use an quadrupled (or octuplicated) rate, the approach changes hardly: we can draw one (or two) additional random numbers with the same possible values 0 and 1 and define two-bit (or three-bit) class labels: 00 refers to class 1, the other class labels are consequently 01, 10 and 11. The class labels for a split of S into eight spike trains are obviously the binary representations of the numbers 0 to 7.

Since every single bit is drawn with the same probability 0.5 each class label appears with the correct probability.

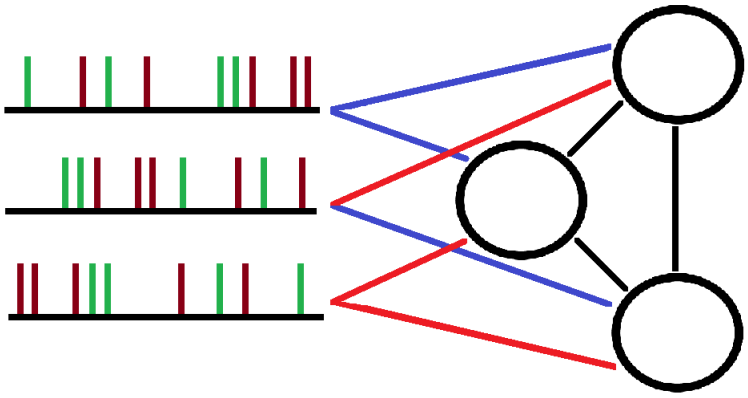


Figure 4: Setup for a split experiment. The sketch shows the random splits of spike trains into a green and brown subset. The blue and red lines denote excitatory and inhibitory synapses, respectively. In this sketch the split factor is 2 and the connections from spike trains to synapses are ordered.

split factor	DKL (order)	DKL (random)
2	0.0031 ± 10	0.0032 ± 9
4	0.0030 ± 8	0.0031 ± 8
8	0.0025 ± 9	0.0028 ± 11

Table 1: Final DKL values for different ordering schemes and split factors corresponding to the experiment in fig. 4

Figure 4 shows first an exemplary experimental setup of a 3-LIF-neuron network. Since the split-factor is 2, each of the original spike trains is split into one brown and one green class. These spike trains are sent to the neurons in order, which means, that the first spike trains connect to excitatory

synapses and the last spike trains are connected with inhibitory synapses. In contrast, there is also the possibility to assign spike trains to synapses randomly.

The experimental results are shown in table 1. All different values are in the order of 10^{-3}, which is the same as achieved by PP Neural Sampling. There is neither a significant difference in the choice of the split-factor not the randomness of the ordering of the synaptic connections.

3.2 Sampling with LFSRs

This subsection discusses basic experiments in which LFSRS are used the background noise generation. In section 3.2.1 the influence of register length as well as of tap configuration is evaluated. The results are discussed in detail since they are reference for the results of later experimental setups. Secondly, in section 3.2.2 a first - naive - way of reducing the area used by the LFSRs is presented, which failure is the reason why we examine more complex LFSR architectures later on.

3.2.1 Sampling with Standard LFSRs

To answer the question whether Neural Sampling using LFSRs for noise generation works we choose a very easy experimental setup. Each of a set of neurons is connected to two LFSRs. One register acts as acts as excitatory, the other one as inhibitory spike source. Additionally, each neuron is connected with all others by randomly drawn weights and has an individual bias input.

A sketch for four neurons can be seen in fig. 5, where the biases and feedback taps of the LFSRs have been ommited for clarity. One can think of different parameters which might have an influence on the DKL value. The most obvious parameter is the register length, but also the choice of the feedback taps could be of importance. Of course, the threshold affects sampling because it determines the final rate, but since the LFSRs are supposed to bring the neurons in their high conductance state, for all experiments the threshold was chosen such that the LFSRs emit spikes with a frequency of approximately 1000 Hz. The absolute value is of less importance as long as it is high enough for Neural Sampling to work at all.

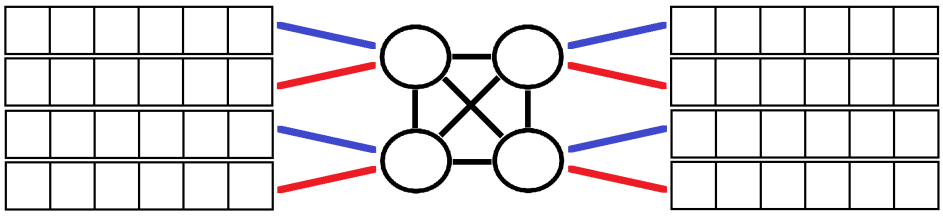


Figure 5: LFSR architecture for Neural Sampling. Four neurons are fed by two LFSRs each, one for inhibitory and one for excitatory spike input, respectively. For clarity, biases and feedback taps have been ommited.

In fig. 6 the DKL courses of a few LFSRs with different lengths are plotted. Two effects are important: first of all, the DKL courses follow the DKL trajectory achieved by Poissonian background noise, but start to flat out at some point in time such that the DKL remains at the same value for the rest of the sampling time, which is 1000 s. That means, that only during a small fraction in time Neural Sampling works. This turning point is clearly dependent on the register length: for example, the DKL course of the LFSR with length 12 begins its flattening at about 200 ms. 12 bit result in a period of $2^{12} - 1$ states, therefore the period length of the register is $0.05\,\text{ms} \cdot (2^{12} - 1) = 204.8\,\text{ms}$, which is exactly the time of the turning point.

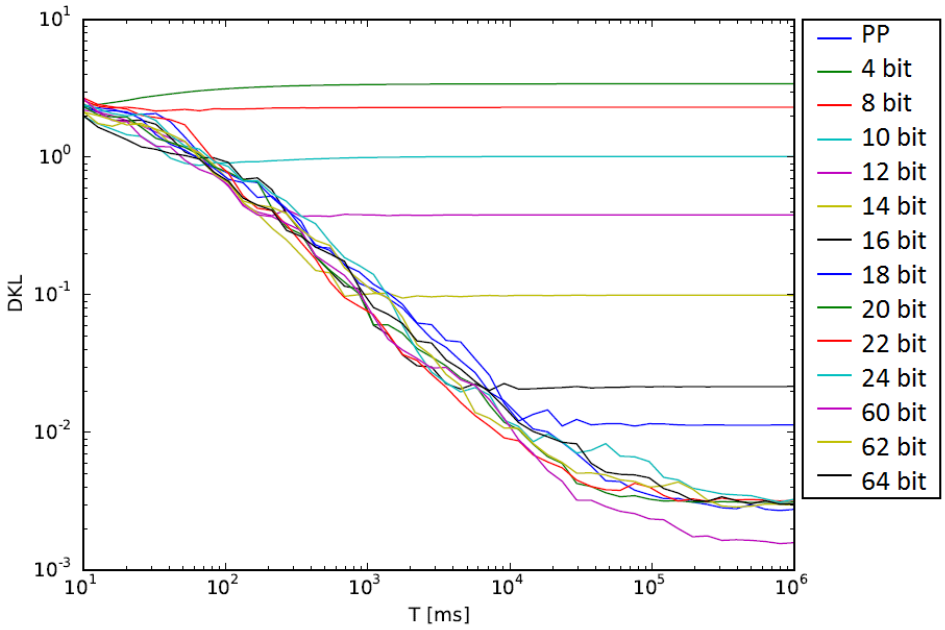


Figure 6: DKL course for LFSRs of lengths in the range from 4 to 64. All LFSRs of a certain length share the same feedback taps, but are initialized differently. The DKL does not change once the first period of the LSFRs was completed. The final DKL value is clearly dependent on the period length of the registers.

While for very short register lengths (and hence periods) the DKL not even starts to decrease, one observes for long register lengths ($n > 20$), that the DKL does not improve until the full period is reached, but plateaus at the same order as in the Poissonian case. Since the Neural Sampling theory is formulated in terms on Poissonian noise this sounds reasonable: Poissonian background noise works best, and differently generated spike trains can at most lead to equivalent, but not better results, which are achieved by it. Further extending of the register lengths the can not reduce the differences between Poissonian and LFSR generated spike times to zero: the first one is at least in theory a real random process (though its implementation is deterministic), while the LFSR is not only fully deterministic but not very complex at all. Especially the run property of a LFSR leads to interspike intervals which differ from the ISIs of a Poisson process (refer to fig. 7): In the Poissonian ISI histogram with its shape like an exponential decay, every ISI time appears with the correct frequency, while the LFSR ISIs are only an

approximation. The concrete structure of the LFSR ISI histogram is length-dependent, too. For longer registers the histogram will look more like the Poissonian (see section 5, fig. 29), but the run property prevents a perfect convergence of the two shapes.

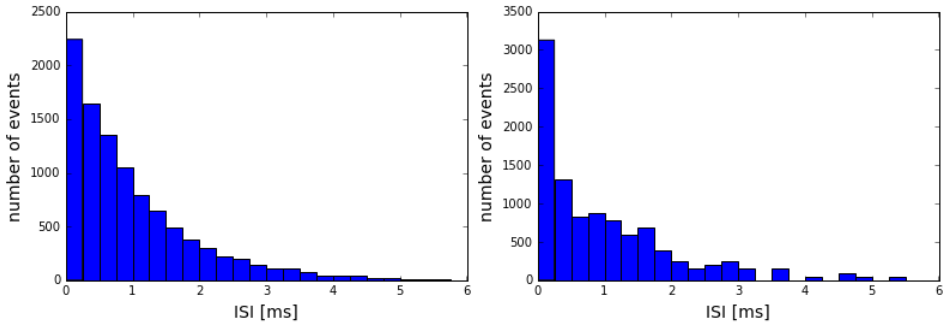


Figure 7: Histograms of the interspike intervals of spike trains generated by a Poisson Process (left) and a LFSR with length 12 (right). Sampling time is $10s$. The left histogram is shaped like an exponential decay while the right one is only an approximation. Longer register lengths lead to better, but not perfect, approximations.

The plateaus in the DKL trajectories arise from the periodicity of the LFSR spike trains. The Kullback-Leibler divergence gives a hint of the dissimilarity of two probability distributions, in this case between the theoretical and sampled distribution. An end of the DKL development results from the fact, that for the single neurons the probability of encountering the different states does not change anymore. This means that the spikes emitted by the neurons repeat themselves with the same period as the LFSRs have. This can be seen in fig. 8

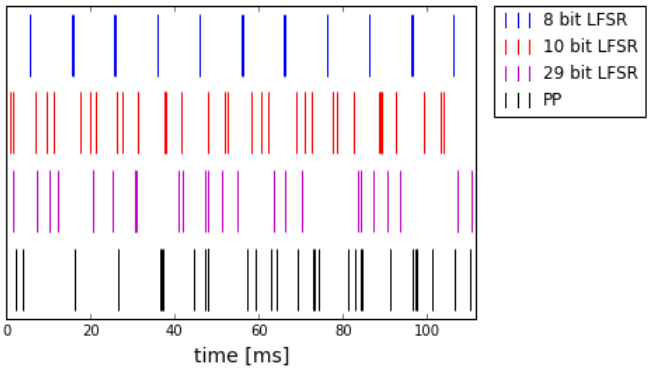


Figure 8: Raster plot of three spike trains generated by LFSRs of lengths 8,10 and 20 and one generated by a Poisson Process. For the 8 bit LFSR two spikes immediately after another follow a longer break of about 10ms. The second topmost spike train has a period of about 45 ms, while the third and fourth spike train show no periodicity in this time interval (the Poisson spike train has no periodicity at all).

Since we are interested in a measurement of the *poissonity* of a spike train, we calculate their characteristic values and apply the KS test. We start with a discussion of the auto- and cross-correlations (AC and CC) of the spike trains: fig. 9 to fig. 13 show the unnormalized AC and CC of Poisson and 12 bit LFSR generated spike trains. spike times are sampled up to $T = 1000\,\text{ms}$. Figure 9 shows the deltapeak approximation of the Poisson AC. Since the single spike times are drawn independently of each other, any shifted version of the spike train with itself results in low autocovariance values, with the exception of a shift by zero. In fig. 10 the CC of two spike trains generated by a Poisson process is shown. The same argument as for the AC holds here as well and explains the low crosscovariance values: single spike times are independent from each other, therefore the CC is low without any exception (note the scale of the y-axis).

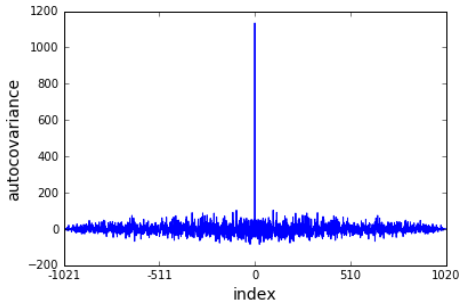


Figure 9: Deltapeak-like AC for a spike train originating from a Poisson process.

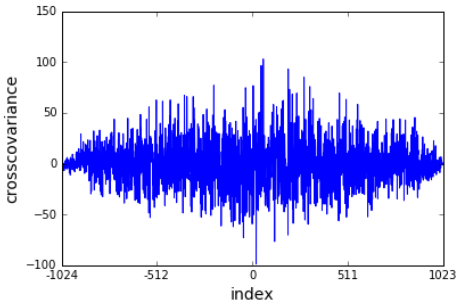


Figure 10: CC for two Poisson generated spike trains. Note the scale on the y-axis.

The AC of the 12 bit LFSR in fig. 11 approximates a sequence of deltapeaks with a distance of the LFSR period P: every $2^{12}-1$ states the sequence repeats itself and generates therefore the same spike times, which result in the plotted deltapeaks. Again the same argument holds for the crosscovariance and explains the peaks, but in addition there is an offset of the middle peak. This peak corresponds to the offset of the two initial seeds in the sequence, since for the two LFSRs the same feedback taps were used.

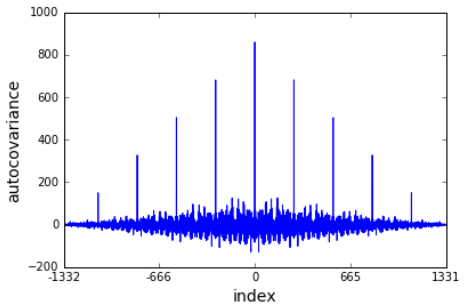


Figure 11: The AC of a spiketrain generated by a 12 bit LFSR is a sequence of deltapeaks for $T > P$.

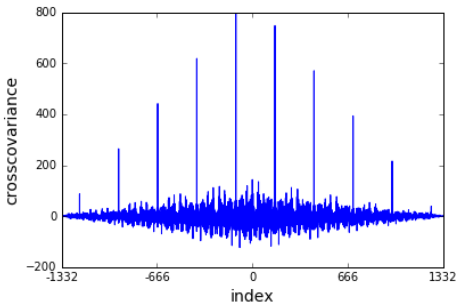


Figure 12: CC of two spike trains generated by two 12 bit LFSRs with the same feedback taps but different initial seeds: sequence of deltapeaks with offset.

These covariances diminish if different feedback taps are used, which is shown in fig. 13.

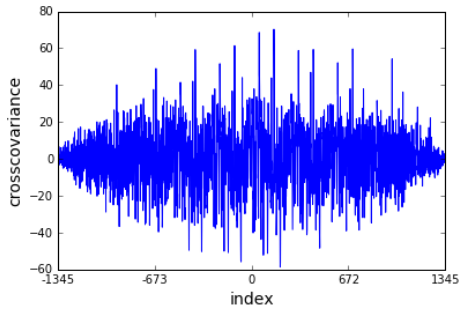


Figure 13: CC of two spike trains generated by two 12 bit LFSRs with different feedback taps. Note the similarity to the Poissonian case.

Figure 14 shows the CV_{ISI} for LFSRs which have lengths in the range from 4 to 64 bits. The registers are run for 1000 ms which means that a bit length of 16 is sufficient for unrepeated spikes. For every spike train generation the same feedback taps are used, but as experiments show a use of different sets of feedback bits does not affect the final correlation values. Due to the periodic repetition of spike times the standard deviation decreases for longer sampling times and short register lengths, while the absolute value stays the same. A register length of 10 or higher is sufficient for the CV_{ISI} to be around 1 like for Poisson generated spike trains, though for some register lengths depending on the initial seed the covariance value rises.

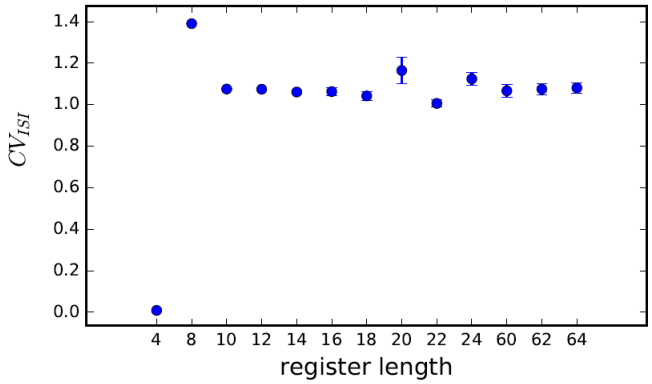


Figure 14: correlation values of interspike intervals for LFSRS of lengths in the range from 4 to 64. If the register length is larger than 10 bits, the CV_{ISI} gets approximately 1.

The corresponding plot of the KS values (fig. 15) looks completely the same, for the 4 and 8-bit register the fraction of pvalues which are above 0.05 is very low, but from 10 bit register length on 60% to 70% of the pvalues are higher than 0.05, so the similarity between a Poisson spike train and a LFSR one is big. This trend is also for longer times observable.

Due to the course of both the CV_{ISI} and the KS-plots these characteristic values are of less importance than the covariance of the spike trains, which explain the DKL trajectory very well. The first two do not distinguish between longer register lengths: these values divide the register lengths in two subsets: short – CV_{ISI} far from 1 and fraction of pvalues smaller 0.05 – and sufficiently long – CV_{ISI} around 1 and fraction of pvalues larger than 0.05.

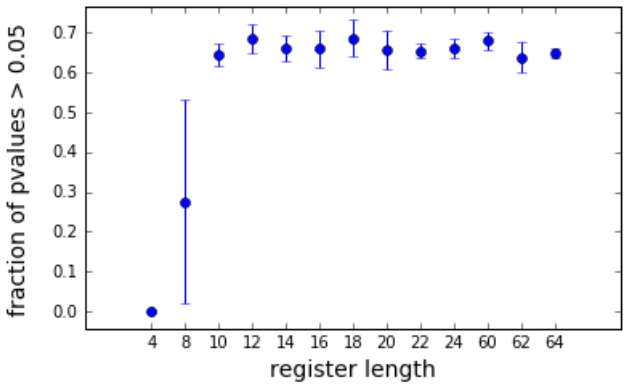


Figure 15: fraction of pvalues larger than 0.05 for LFSRS of lengths in the range from 4 to 64. If the register length is longer than 10 bits, the fraction of large p-values is about 60 to 70%.

There are two possible issues arising to the auto- and crosscorelations of LFSR m-sequences: Two spike trains generated by parallel LFSRs are up to an offset identical, additionally both of them repeat themselves, if the sampling time is larger than the period length. To determine, which of the effects - autocovariance or crosscovariance - is of more importance, the above sampling experiment is repeated, but this time different sets of feedback taps are used. This reduces the crosscovariance as shown in fig. 13. The result of the experiment can be seen in fig. 16.

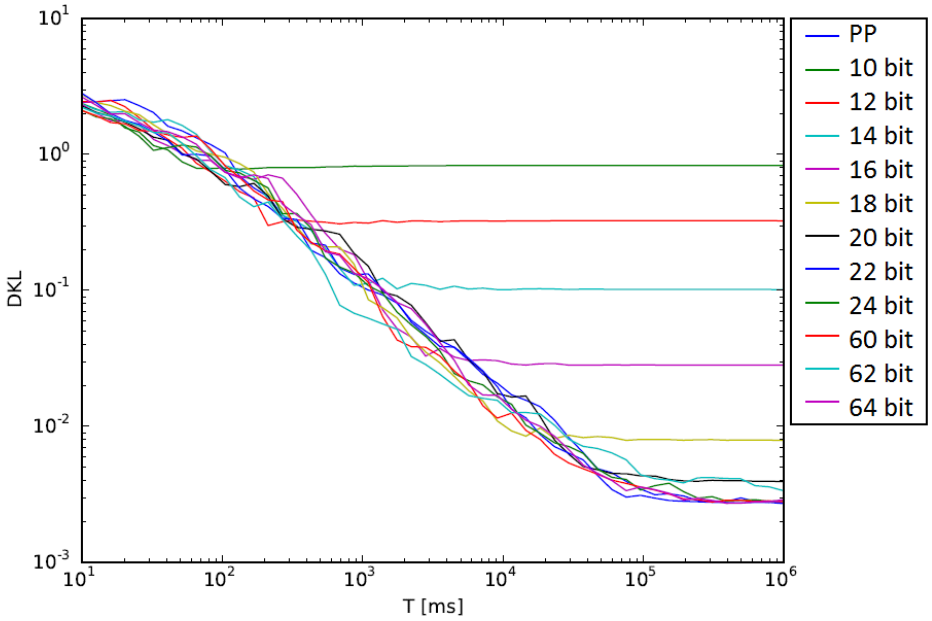


Figure 16: DKL course for LFSRS of lengths in the range from 10 to 64. All LFSRs of a certain length share different feedback taps and are initiliazed randomly. The DKL stops decreasing once the first cycle is completed. There is no significant difference to the same experiment with a same-taps configuration.

The plot looks similar to fig. 6. This indicates that it does not matter if some neurons get the same input with an offset or if the spike times are completely uncorrelated. However, the offset is a necessary condition as we will see. The periodic autocovariance remains, and therefore the main issue is creating long m-sequences, where the crosscovariance has second priority.

3.2.2 Sampling with a Reduced Set of LFSRs

The final DKL values in fig. 6 show that in principal Neural Sampling using LFSRs works, if the period length is at least in the same order as the sampling time and the used m-sequences are shifted to each other. This means that it is sufficient to build very long registers on chip, but since the space on the chip is limited and the priority lies on the realized number of neurons, one needs to find a trade-off between sampling quality and space used for LFSRs.

A network of five neurons needs in total ten synaptic inputs: five excita-

tory and five inhibitory. The very first idea one has is to simply reduce the total number of LFSRs and connect one LFSR directly to several neurons. In this experiment a set of two to ten LFSRs is connected to the neurons. Figure 17 shows the sampling results of this reduction. The connections are drawn randomly, the figure shows the mean DKL value of ten runs as blue line, where the filled area are the minimum and maximum values.

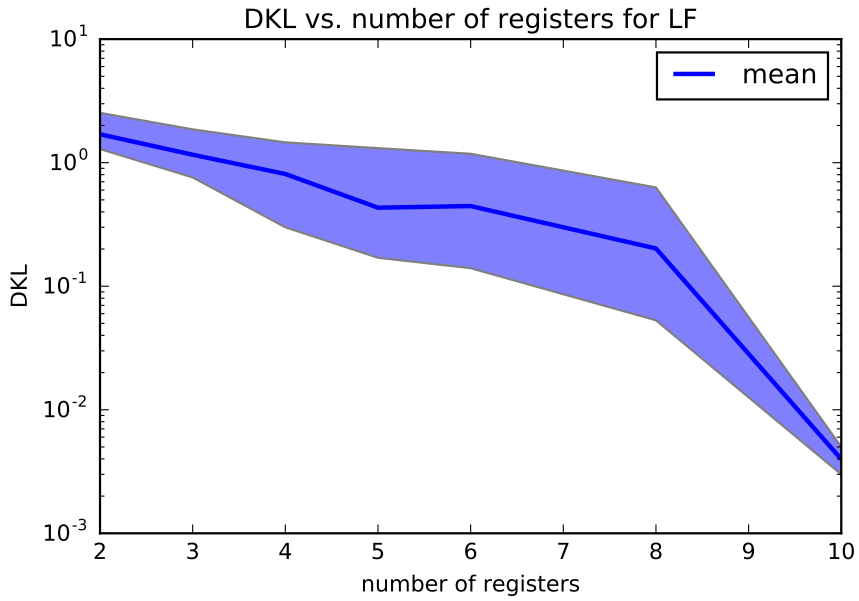


Figure 17: DKL vs. number of registers. The blue line gives the average DKL of ten runs, while the colored area ranges from minimum to maximum values. The more registers are used, the better the final DKL value is.

We observe that the more LSFRs there are, the better the sampling quality is. Even if a single LFSR is omitted and nine registers are used for the noise generation this results in a major decrease in sampling quality. In summary: This very basic kind of space saving does not seem to work, which means that the shift between spike trains is crucial, while a high and periodic crosscovariance does not seem to affect sampling that much.

3.3 LFSR Spike Splitting

In the following section several possible LFSR architectures are investigated, commonality of which is the use of two sets of PRNGs. One set generates

spike times while the other set is used for distributing these spike times among different neurons. In a first attempt, we model this splitting as a Cox process, which rate is driven by a series of Bernoulli trials. The trick is to generate the spike times with a doubled rate such that every synapse gets input after the split with the standard rate. There is one crucial effect splitting has: If a neuron gets a spike at a specific point in time, all other neurons connected with the same source can not receive a spike at the same point in time. This leads to correlations which will have an influence on sampling. Different parameters as split ordering and frequency of split number generation are discussed in section 3.3.1.

From the theoretical point of view the splits have to be performed randomly, but, again, the aim is to replace the Bernoulli trials with pseudorandom numbers generated by LSRs. This setup is discussed in section 3.3.2.

Though two LFSRs are replaced by one with doubled rate and one which takes care of the splits this approach is useful: On the one hand, the correlations for the two cases may differ and this may have an effect to the DKL value. On the other hand the split is not restricted to a factor of two, one could also think of an octuplicated rate and then three registers which encode a binary three-bit number. Three binary bits correspond to eight numbers, so in total four registers are used as noise source for four neurons. This results in a *win* of four registers instead of eight as would be the case in the setup discussed until now.

3.3.1 Splitting using numpy as PRNG

The following experiments will begin with a very complex model in the sense of LFSR architecture, before two possible simplifications and their combination is evaluated. On the first glance this approach sounds counterintuitive, but every simplification will lead to more and higher covariances, which will affect the sampling quality.

All splits of this section are done using random numbers generated by the *Numerical Python* module.

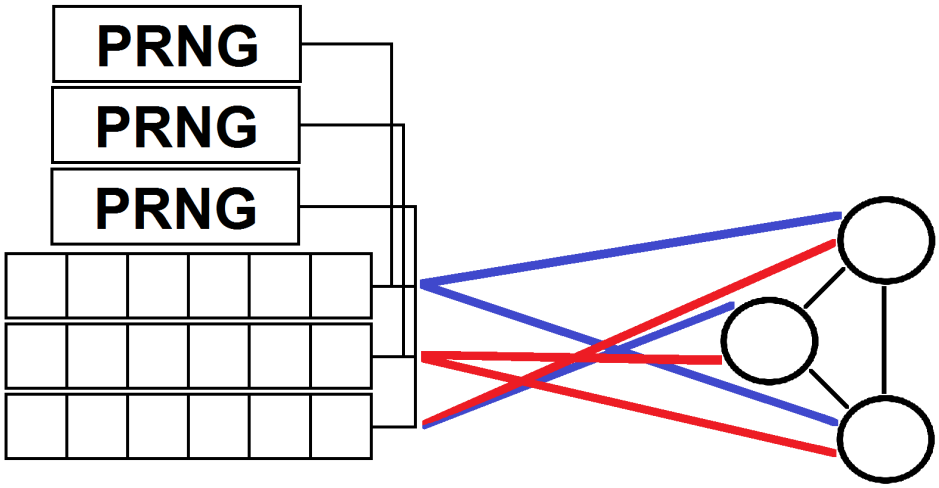


Figure 18: Most complex, *bestcase* architecture: one PRNG per split and random connections between spike trains and neurons.

The most complex architecture can be seen for a three LIF-neuron network in fig. 18. There are three LFSRs run with doubled rate and one PRNG for each register. The registers are connected to two neurons each and every connection is randomly drawn before starting sampling. The reason behind this way of connection is to prevent covariances which can arise for example if one single LFSR is connected to two excitatory synapses of two neurons: if there is an event at the excitatory synapse of neuron 1 there cannot be an event at the excitatory synapse of neuron 2 at the same time. Due to the random connection and multiple runs these covariances are reduced, but not disabled.

The experimental results for different register lengths (8 to 64 bit) are shown in fig. 19.

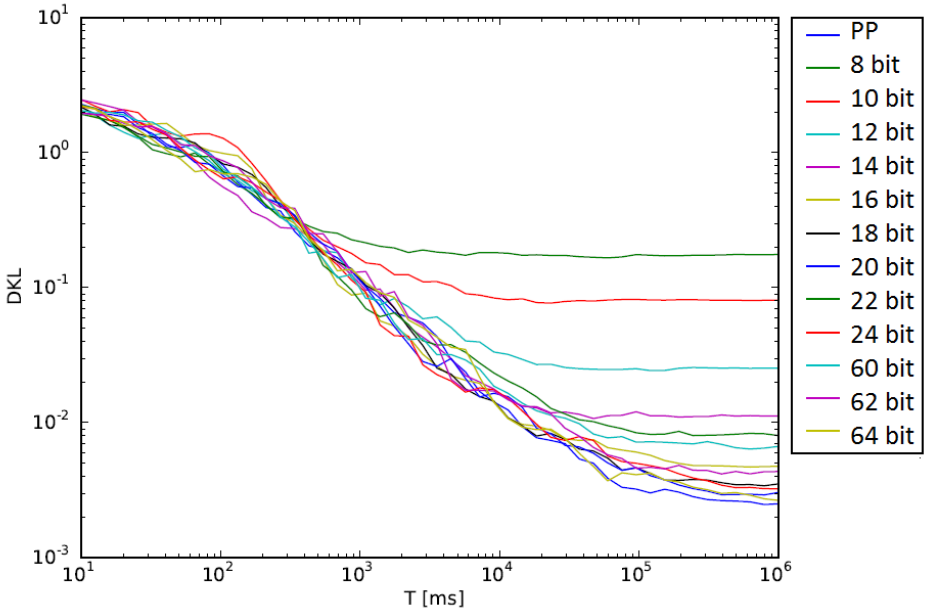


Figure 19: DKL plot of LFSR generated spikes. LFSRs run with doubled frequency, splits are performed with numpy. Decreasing DKL for short, but increases for long register lengths.

The DKL courses indicate that due to the random split the period length of the generated spike trains is artificially extended and - only for short register lengths recognizable - sampling works much better. Additionally the curves converge smoother, there is no sharp edge as before (e.g., see fig. 6) when the sampling time equals the period length. For example, in case of 8 bit LSFR length, where the period time is about 12 ms, the DKL decreases until the sampling time is 300 ms, and the final DKL value is about 0.2, which is one order of magnitude better. For large register lengths the split process is useless, since if the period length is larger than the sampling time, the additional gained decovariance has no effect.

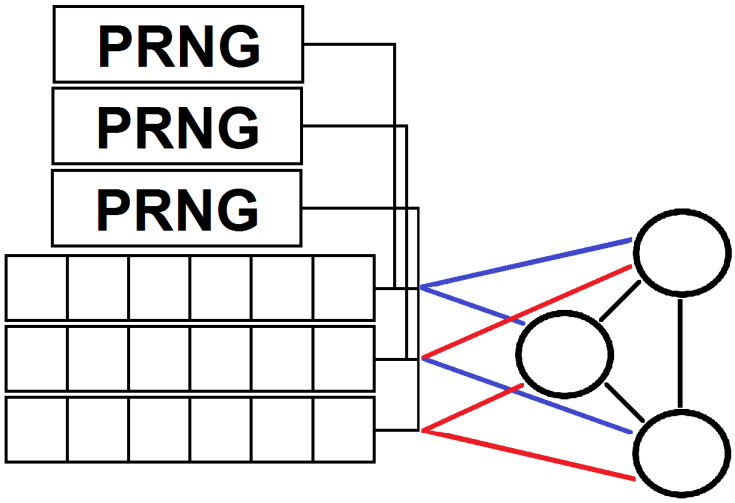


Figure 20: Second architecture: one PRNG per split and ordered connections between spike trains and neurons.

The first simplification is related to the order of the connections between LFSRs and neurons: if the ordering is of no importance this special kind of ordering in fig. 20 should not lead to a loss in sampling quality, too. In this scenario the spike times of the first LFSRs are used as excitatory input, while the latter LFSRs generate inhibitory input. The final DKL values given in fig. 23 indicate that both investigated LFSR architectures are equivalent. This is an interesting phenomenon, since one do not has to care about the ordering.

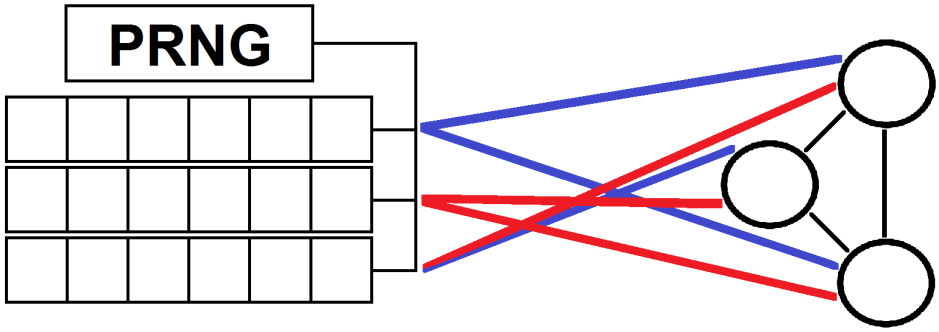


Figure 21: third architecture: one PRNG for all splits and random connections between spike trains and neurons.

The second simplification is the reduction of the number of pseudorandom generators: in the architecture of fig. 21 one PRNG takes care of all splits simultaneously. Of course, this simplification affects the covariances of the different spike trains as well. The results are listed in fig. 23.

In this case we observe major DKL increases for long register lengths. For short ones (8 to 12 bit) the influence of the short period length is stronger than the effect of the restricted split random numbers. For register lengths above 12 bit the DKL loss is about a factor of two. The reason for this deterioration is the fact, that the single PRNG leads to correlation between some of the neurons: For example, let spike register i generate the spike trains i_1 and i_2, and register j the spike trains j_1 and j_2. Now, every time a spike of i is distributed to i_1, a possible spike of j has to be part of j_1 and not j_2, since the corresponding bit of the split register is the same for all spike sources.

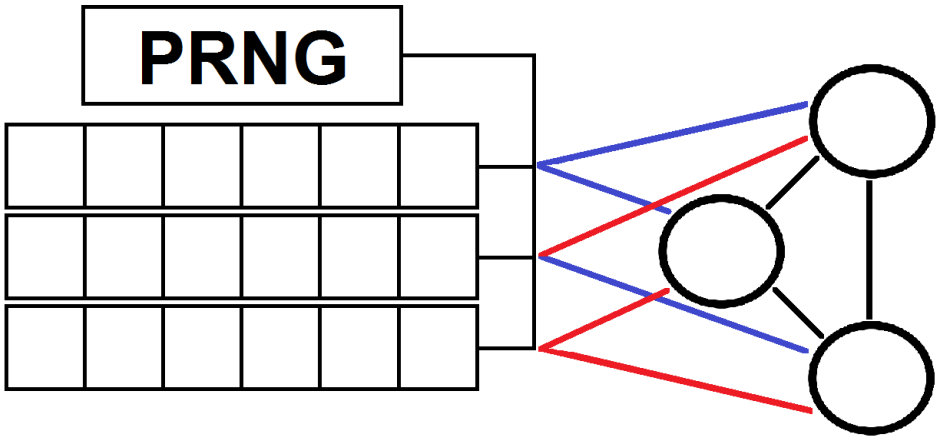


Figure 22: Easiest, *worstcase* architecture: one PRNG for all splits and ordered connections between spike trains and neurons.

Finally, before starting to use LFSRs for the split generation, both simplifications are combined, since though the second simplification had no influence if applied exclusively, this does not necessarily hold true if different neurons are linked as well. The resulting architecture is given for three neurons in fig. 22 and the experimental results are given in fig. 23.

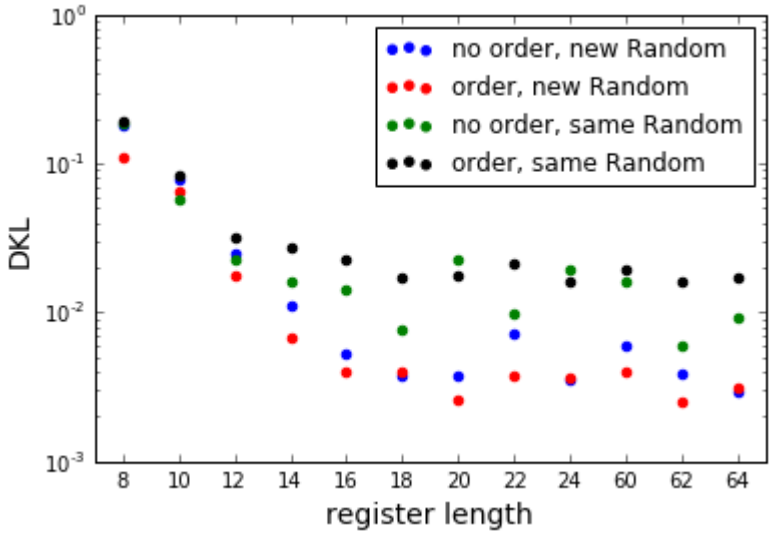


Figure 23: DKL values of four different architectures. Register lengths from 8 to 64 bit. The more complex the architecture is, the better is the final DKL value.

Again - and in contrast to the single application of input ordering - we can a DKL loss for large register lengths. The deteriorations sum up to almost one order of magnitude in sampling quality loss. This sounds extremely bad, but we used only a total number of $5 + 1$ pseudorandom number generators (5 LFSR and 1 numpy PRNG). If we compare the final DKL of $2 \cdot 10^{-2}$ to the DKL value we got for 6 registers in fig. 17, which is $2 \cdot 10^{-1}$, we achieved a total decrease in the DKL value by a factor of ten.

The next step consists of increasing the splitfactor: up to now we tried to save space on the chip by restricting the number of PRNGs which perform the splits, but – as mentioned in the introduction of this section – one can go one step further and increase the rate of the LFSRs. If these run with quadrupled (or octuplicated) rate and the spike times are divided into four (or eight) subsets, one would save additional space - always under the assumption that Neural Sampling still works, which is being investigated in this thesis.

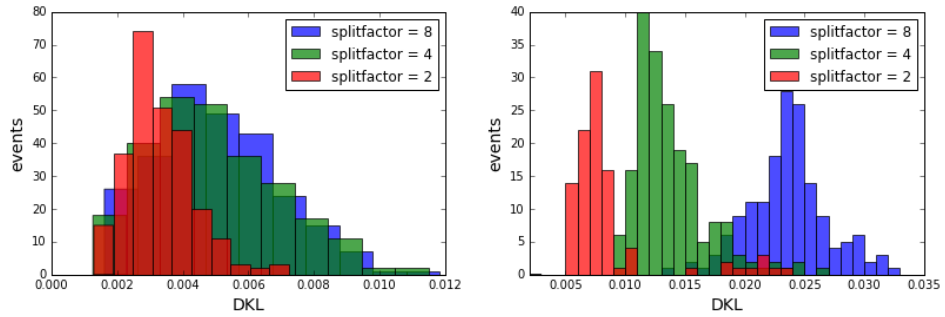


Figure 24: Histograms of DKLs. *bestcase*: random connections, new split numbers (left), *worstcase*: ordered connections, same split numbers (right)

Figure 24 shows the histograms of the final DKL values for twenty runs with different bias and weight drafts, in total 200 simulations.

The left histograms are the results for the *bestcase* scenario: the connections between LFSRs and synapses are randomly drawn and the split random numbers are only used once. The register length of the LFSRs is 20. To enable better comparisons and avoid splitting artifacts the number of neurons is set to eight - the lowest common multiple of two, four and eight.

The first observation is the spread of the histograms: for a splitfactor of 2 the DKL values are of smaller variation. The split factor of 4 results in the largest mean DKL, but all three histograms are in the order of 10^{-3}.

The right histogram shows the same for the *worstcase* architecture. All three distributions are shifted towards higher DKL values and they are much more separated from each other. For the octuplicated rate and split among eight spike trains, the DKL loss is about a factor of ten.

3.3.2 Splitting using LFSRs as Split Generator

The next change in the experimental setup is the swap of numpy PRNGs for LFSRs. Of course, depending on the period length of the latter one, correlations occur not only when generating spike times but also to the split generation. Figure 25 shows the DKL values depending on both spike and split register length, for two different types of architectures and one reference setup. The upper left plot uses fig. 18 as structure: in this *bestcase* setup the connection between LFSRs and neurons is randomly and all splits are performed with new noise. The upper right plot is the *worstcase* scenario, where there is only one LFSR for split performing and the connections are ordered (for the structure see fig. 22). In the bottom plot, the DKL values

are shown for standard LFSR sampling with an equivalent use of space. This means, if the split register length is 10 and the spike register length is 14, this is space-equivalent to standard LFSR sampling with all the registers being of length 12. Since this plot shall only give a rough overview over the improvement of the final DKL, the resulting equivalent register lengths are rounded such that former simulations could be reused. For example, every register length > 20 is set to 20, since this makes no significant difference.

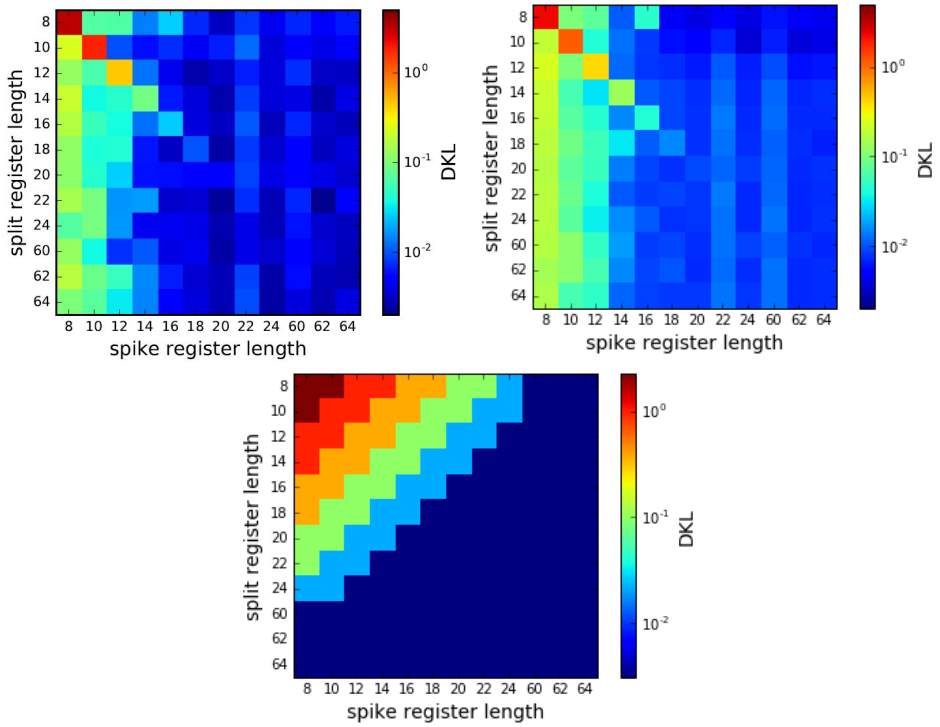


Figure 25: DKL values for LFSR generated spikes and splits (splitfactor is 2). *Bestcase* architecture(upper left), *worstcase* architecture (upper right), standard LFSR sampling (bottom, equivalent to *worstcase* architecture)

There are three important statements to make about the first plot of fig. 25. First of all, the diagonal elements of the plot are clearly worse than the off-diagonals. Especially the use of 12-bit registers both for split and spike generation is worse than the use of a 8- and a 10-bit register for the two tasks. Reason for this is that the two periods of the registers with different period lengths do not overlap but sum up to a longer period. Therefore, periodic spiking is to some extent prohibited.

Secondly, the choice, which register length is used for spike generation and which one for split performing is important. Independent of the split register length short spike registers will not lead to low DKL values. This is reasonable, if one thinks in terms of interspike intervals: 8 bit register length corresponds to 255 possible states and a period of 12.8 ms. Consequently, the spike times repeat every 12.8 ms. A simulation time of 10^6ms means that we use the same ISIs about 78000 times. Even if the splits are performed without any correlations (such that there are about $\binom{255}{127} = 5 \cdot 10^{73}$ possible resulting spike trains, large subsets will correlate due to the limited number of ISIs. Repeating spike input leads to DKL deterioration as seen above.

Third, for small LIF networks, a spike register length of 20 or more is sufficient for good spike generation, regardless of the split register length. This means, that five 8 bit and five 20 bit registers are sufficient for Neural Sampling in small networks – in contrast to the ten 20 bit registers which were necessary if standard LFSR sampling was performed.

The main results in the upper right plot are still the same: the diagonals cases result in bad DKL values, the same holds for short spike register lengths. The correlations which are typically for the diagonals appear for other lengths combinations, too. If the spike register length is a multiple of the split register length the arising correlations lead to increasing DKL values as well (as can for instance be seen for 16 bit spike and 8 bit split LFSR).

The bottom plot suggests, that some cases exist, in which a use of the splitting architecture is very efficient. The best example is the use of a spike register length of 20 bit and a split register length of 8 bit, which results in a DKL of about $3 \cdot 10^{-3}$. Space equivalent is the use of ten 14-bit LFSRs for standard sampling, which only results in DKL values of the order 10^{-1}, which is an immense improvement.

Figure 26 shows the result of the above experiments for a splitfactor of eight and in case of the *worstcase* scenario.

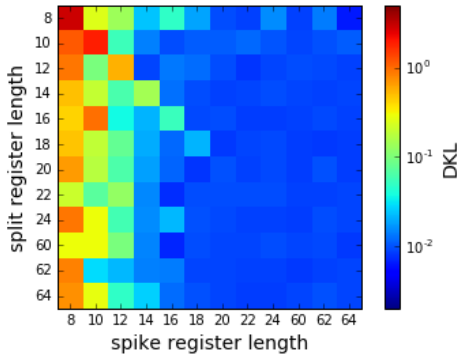


Figure 26: DKL values for LFSR generated spikes and splits. The splitfactor is 8, *worstcase* architecture

In total the three observations of the previous cases can be made here as well: The offset elements are worse than the architectures in their direct neighborhood. The symmetry breaking and good results for long spike register lengths are visible as well, though in general the final DKL values are worse than in the previous cases. The shift towards higher DKLs strengthens the observations made in fig. 24.

3.4 Sampling using Gold Codes

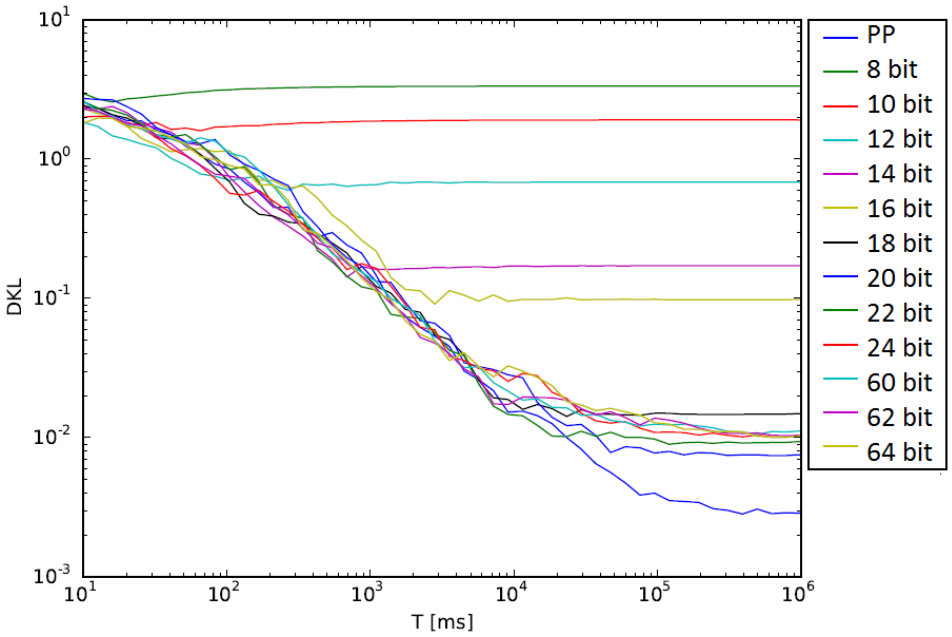


Figure 27: DKL courses for Neural Sampling using Gold code generators of different lengths (8 to 64 bit) as noise source. For long register lengths the final DKL values are low, but not as good as the values achieved with PP spike trains.

For the generation of a set of binary Gold sequences only two registers are necessary. A slightly modified structure allows a creation of several spike trains (see fig. 2). Figure 27 shows the DKL courses of a five neuron network if the spike times were generated using Gold codes. Since due to the construction of Gold codes (and their modification to sequences of spike times) the period length of the single sequences does not change, we observe the characteristic flattening of the DKL trajectory after the first period cycle. The second observation is the performance of longer registers: even the 60-bit and more registers do not achieve the same performance as Boltzmann machines supplied with Poission noise.

Reason for this is clearly not the periodicity of the spike times – as then other architectures, investigated previously, would have been able to achieve DKL values comparable to Poission processes. The loss in sampling quality can instead be explained by looking at the structure of the Gold spike

generator: all possiple spikes are timed with the upper LFSR: if this register value is larger than the threshold, a spike is possible. The feedback taps determine now, if the spike is sent to the neuron, but here the correlations enter: every neuron, whose feedback taps add up to one, gets a spike. This can be also seen if one looks at the input spike trains. The correlations due to the construction of the spike train generator are clearly observable. Figure 28 shows ten spike trains for a network of five neurons.

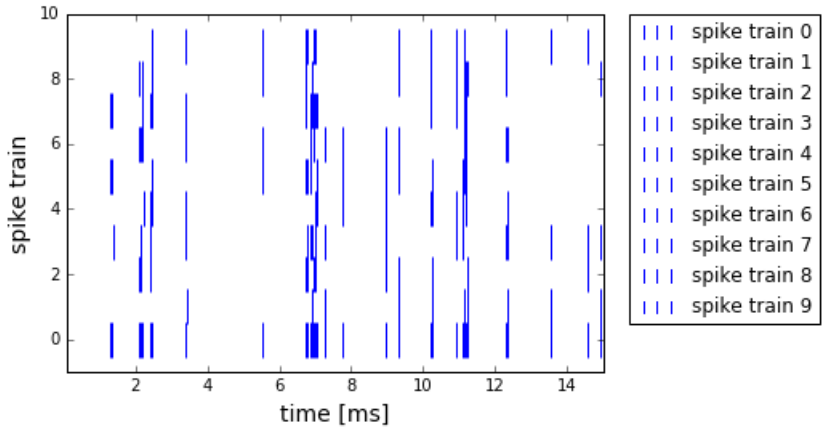


Figure 28: Ten spike trains created by a Gold code generator. Note the correlations between different spike trains.

Note that the correlations are not a result of the Gold code generation, but a result from the necessary modification of the Gold sequences to generate spike times.

Though there is a gap between Gold code sampling and Poisson sampling, an improvement in sampling with LFSRs has been achieved: the Gold code generator falls back on just two registers and is able to produce spike times for the whole LIF network.

4 Discussion and Outlook

This thesis has shown that background noise generated by Linear Feedback Shift Registers (LFSRs) is able to make the membrane potential of LIF neurons stochastic. Therefore, the LIF neurons are able to sample from arbitrary probability distributions, which is called Neural Sampling. A replacement of the original Poisson Process spike trains with LFSR generated background noise is possible. Furthermore, certain LSFR architectures were investigated with the aim of achieving good sampling performance with as little space as possible. The latter constraint is important for a possible use of LFSRs on neuromorphic hardware. There one wants at the one hand to generate the necessary background-noise on-chip, but on the other hand the area *wasted* for noise generation should be as small as possible. Architectures basing on Gold codes or Cox processes seem to be an efficient way of creating background noise.

The different experimental setups share some general characteristics. As shown in section 3.2.1, the most crucial LSFR property is its length, since small periods result in repeating spike times. These in turn lead to a repetition of the network dynamics such that the single neurons do sample the right probability distribution. The tap configuration has a minor influence: If the registers are initialized differently, there is no observable effect on the DKL value between spike time sequences with no correlation and highly correlated ones. Since naive reduction of LFSRs to save area on the hardware chips does not seem to work (see section 3.2.2), different architectures were investigated.

In section 3.3 an approach based on Cox Processes was taken. Spike trains run with a multiplied rate, while their spikes are assigned to several synapses (and neurons). For this case there are quite a few parameters such as split factor and number of split generators. The first one is the number of synapses, every spike train is split to. The second one states, if one or several random number generators are used to perform the assignments of the single spike times to synapses. Some combinations of split and spike register length allow rather good sampling results since the splits partly destroy the correlations between single background spike trains.

A second approach based on Gold codes (used for example in satellite communication) was evaluated in section 3.4. In contrast to the splitting approach, there is a higher loss in sampling quality. This loss is caused by very strong correlations of the single spike trains. These again stem from the fact, that in the end a single LFSR produces the spikes, while the second LFSR just provides for the connections between the LFSR and the synapses.

However, this approach immense ly saves chip area at the same time.

Furthermore, two methods of determining the quality of spike trains for Neural Sampling were investigated. Both the KS test result and the Correlation Value of Interspike Intervals (CV_{ISI}) do not correlate with the DKL values of the corresponding Neural Sampling experiments. Consequently they can not be used for determining sampling quality.

This thesis investigates just the very fundamentals of background noise generation with LFSRs. There are still some open questions which have to be investigated. First, there might exist more powerful LFSR architectures in the sense of a more efficient use of the registers. If one accepts the minor loss in sampling quality, a reduction of up to 90% of the necessary chip area was achieved. The architecture based on Gold codes is therefore promising.

Possible improvements can be done by decreasing the strong correlations between the single spike trains. Therefore a replacement or modification of the single spike generating register is necessary. One can either use a set of LFSRs for the spike generation or find another structure which uncouples the different neurons more. For example, one could use standard LFSRs with a set of feedback taps for each synaptic input. A threshold depending on the number of feedback taps combined with the specific values of the XORed bits could lead to a modified architecture, which allows an individual generation of spike trains. Of course, the number of possible architectures is immense.

The first approach – increasing the number of registers again – can be especially important for larger networks. All statements in this thesis are verified for small network sizes as five or ten neurons. There is still the question of scalability: More neurons originally meant more LFSRs, as each additional neurons had to be supplied by two additional LFSRs. How far can one just use an another set of feedback taps and extend the architecture based on Gold codes to 20, 100 or more neurons? Where are the limitations of the splitting architecture – what is the limit the rate of the spike generating register can be set to? Does the network size in general encourage or weaken the influence of stronger correlated spike trains? The latter question is not easy to answer, since the neurons are connected with each other such that the influence of the external input could be reduced.

Second, in this thesis a very theoretical approach was taken. The focus was on the spike trains, only. No hardware LFSRs were constructed. That means, all architectures have been assumed to be possible later on the real chip. Especially, the same holds for the connections between the single LFSRs and between neurons and LFSRs. For example, the Gold code architecture uses two feedback taps for each synaptic input, before the feedback output is XORed with the output from the spike generating register. This

means there have to be many connections between these two registers. Again, space on chip is limited. It could turn out, that for large network sizes there are just too many of these connections. This could result in the necessity to find either modified or completely different architectures – though in theory Neural Sampling basing on this architecture works well. To sum up: Due to hardware limitations, some restrictions to the presented architectures could arise, which makes more effort in investigating hardware-orientated setups necessary.

All in all the foundation has been laid and – despite of the open questions – the step from externally generated Poisson noise to on-chip LFSR noise seem to be a step, which can be done in near future.

5 Appendix

Histogram for 24bit LFSR

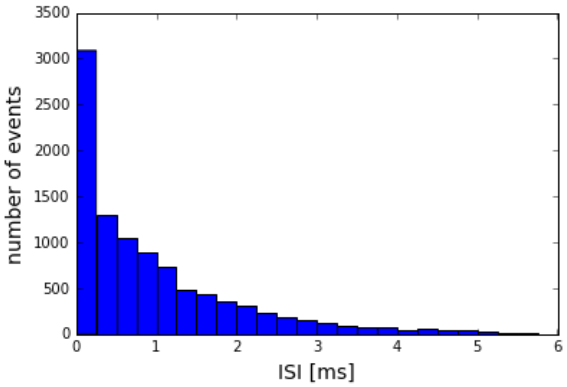


Figure 29: Histogram of the ISIs of spike trains generated by a LFSR with length 24. Sampling time is $10s$ with a rate of $1000Hz$. The histogram is more similar to one consisting of ISIs generated by a Poisson Process.

Neuron Parameters

For all experiments the neuron parameters were hold constant:

Membrane Capacity	C_m	.2
Membrane Time constant	τ_m	0.1
Threshold Potential	u_{thr}	-50.
Rest Potential	u_{rest}	-50.
Reset Potential	u_{reset}	-50.001
Offset Current	i_{offset}	0.
Rise Time (Excitatory)	τ_E	10.
Rise Time (Inhibitory)	τ_I	10.
Refractory Time	τ_{refrac}	10.

The background noise has an average rate of 1000 Hz.

Parameters

Each line in a DKL plot is the average of ten random drafts of bias and weight sets, where the spike times stayed the same in every draft.

The weights are drawn from a scaled beta distribution: w = 1.2∗Beta(0.5, 0.5)−

0.5, where Beta(a, b) has the pdf

$$p(x, a, b) = \frac{1}{B(a, b)} x^{a-1}(1-x)^{b-1} \tag{13}$$

and the Beta function

$$B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt \tag{14}$$

The biases are drawn from the same distribution.

Proof: a Cox Process with Bernoulli Trials as Rate Generates Poisson Processes

The following proof follows Siegrist (2016).

Let's consider a Poisson Process P with rate r. We can describe P by two different sequences. Firstly we can have a look at the inter-arrival times $\mathbf{X} = (X_1, X_2, ...)$ and secondly at the arrival times $\mathbf{T} = (T_0, T_1, ...)$. These sequences fulfill

$$\begin{aligned} T_n &= \sum_i^n X_i \\ X_n &= T_n - T_{n-1} \end{aligned} \tag{15}$$

For P every arrival is independent of all others. If we assume, that each arrival belongs to one of two classes, we can interpret this as a sequence of Bernoulli trials. Each arrival is with probability p of class 1 and therefore with probability $1 - p$ of class 2.

We show, that this splitting gives us two new, independent Poisson Processes, each consisting of arrivals of the same class with new rates rp and $r(1-p)$, respectively.

For the Bernoulli sequence U_k denotes the number of trials from the $(k-1)$st to the kth success, if we define belonging to class 1 as a success. $V_k = \sum_i^k U_k$ denotes the trial numbers of the kth success.

We define Y_k as the arrival time between the $(k-1)$st and kth class 1 arrival:

$$Y_k = \sum_{i=V_{k-1}+1}^{V_k} X_i \tag{16}$$

which consists of U_k terms.

The Y_i are now a random, geometrically distributed sum of variables, because the arrival times are independent of each other and Bernoulli experiments obey the geometrically distribution. Moreover, the variables are exponential distributed random variables, because of the underlying Poisson Process and due to that $\mathbf{Y} = (Y_1, Y_2, ...)$ is itself exponential distributed and a Poisson Process.

This can be shown by using the moment and probability generating functions:

For calculating the moment generating function of random sums the chain rules holds: if $(X_i)_{i\epsilon}$ is a set of independent random variables, T a random variable, then $Z = \sum_{i=0}^{T} X_i$ has the moment generating function

$$M_Z(t) = W_T(M_{X_i}(t)) \tag{17}$$

where M denotes the moment and W the probability creating function.

Since for an exponential distribution A with rate r $M_A^{exp}(t) = \frac{r}{r-t}$ for $t < r$ and for a geometric distribution B with parameter P $W_B^{geo}(t) = \frac{pt}{1-(1-p)t}$ we achieve as the moment generating function for Y:

$$M_Y(t) = W^{geo}(M^{exp}(t)) = \frac{\frac{pt}{r-t}}{1-(1-p)\frac{r}{r-t}} = \frac{pr}{pr-t} \tag{18}$$

For that reason, Y is exponential distributed, and since the arrival times are independent of each other, Y is a Poisson Process with rate pr.

References

BrainScaleS (2016a). http://brainscales.kip.uni-heidelberg.de/public/index.html, visited: 2016-09-12.

BrainScaleS (2016b). The hicann chip. http://www.kip.uni-heidelberg.de/vision/previous-projects/facets/neuromorphic-hardware/waferscale-integration-system/hicann/, visited: 2016-09-12.

Brockwell, P. J. and Davis, R. A. (2006). *Time series*. Springer series in statistics. Springer, New York , Heidelberg, 2. ed., [repr.] edition. Literaturverz. S. [561] - 566.

Buesing, L., Bill, J., Nessler, B., and Maass, W. (2011). Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol*, 7:1–22.

Burnecki, K. and Weron, R. (2010). Simulation of risk processes. Mpra paper, University Library of Munich, Germany.

Cowan, G. (2016). Derivation of the poisson distribution. https://www.pp.rhul.ac.uk/~cowan/stat/notes/PoissonNote.pdf, visited: 2016-09-12.

Cox, D. R. (1955). Some statistical methods connected with series of events. *Journal of the Royal Statistical Society. Series B (Methodological)*, 17(2):129–164.

Gerstner, W. and Kistler, W. (2002). *Spiking Neuron Models: An Introduction*. Cambridge University Press, New York, NY, USA.

Gold, R. (1967). Optimal binary sequences for spread spectrum multiplexing (corresp.). *IEEE Transactions on Information Theory*, 13(4):619–621.

Golomb, S. W. (1981). *Shift Register Sequences*. Aegean Park Press, Laguna Hills, CA, USA.

Gruebl, A. (2016). personal communication. ,.

Kingman, J. F. C. (1995). *Poisson processes*. Oxford studies in probability ; 3. Clarendon [u.a.], Oxford, repr. edition.

Lapicque, L. (1907). Recherches quantitatives sur l excitation electrique des nerfs traite comme une polarisation. *J. Physiol. Pathol. Gen*, 9(1):620–635.

Petrovici, M. A., Bytschok, I., Bill, J., Schemmel, J., and Meier, K. (2016). The high-conductance state enables neural sampling in networks of LIF neurons. *ArXiv e-prints*.

Siegrist, K. (2016). Poisson process: Splitting. `http://www.math.uah.edu/stat/poisson/Splitting.html`, visited: 2016-09-12.

Smirnov, N. V. (1939). On the Estimation of the Discrepancy Between Empirical Curves of Distribution for Two Independent Samples. *Bul. Math. de l'Univ. de Moscou*, 2:3–14.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 22.September 2016

.....
Marcel Großkinsky