How Network Topology Defines its Behavior

Serial Code Detection with Spiking Networks

Heinz, Gerd
Gesellschaft zur Förderung angewandter Informatik e.V. (GFaI)
Berlin, Germany
heinz@gfai.de

Abstract — **In relation to technical sensors, the sound- or code-analysis possibilities of nerve systems are fascinating. How can we understand the amazing information reduction possibilities of nerve nets? In the paper, operation of a nerve net is abstracted by linear, time-invariant systems theory. Abstracted nerve nets give the possibility to use convolution integrals (Faltung) for analysis. The structure of a network reappears, interpreting a specific burst. Nerve system looks like a gigantic convolution engine with billions of nodes and branches, transgressing every technical possibility of analysis. Basing on interference networks (IN) theory, the paper tries to connect the theoretical basis of neural network (NN) theory and linear time-invariant (LTI) systems. As a first step, finite impulse response (FIR) filter theory allows to use time-discrete convolution for switching between system response of a nerve net, its behavior and the net structure.**

Keywords — Faltung; convolution integral; interference network; artificial neural network; FIR; nerve network.

I. MOTIVATION

Each of us knows the powerful ability of our nerve system to analyze noises. We differ between the whisper of the wind or the branding of waves, we know the songs of birds, we hear dangerous noises of a defect car engine, we feel, if an airplane starts. We know, when the washing machine comes to the end, when the coffee maker is ready. We decode the melody of our mobile phone. We subdivide between different voices of our friends. Our brain system has a fantastic ability to differ noises in frequency, tremolo, length or modulation. Moreover we are able to communicate with spoken words and sentences. Last not least many of us speak and understand more than one language.

Medical research detects since more than 150 years relations between brain injuries and artifacts in behavior [28], [30], [31]. Modern MRT makes it possible to inspect the global activity of brain regions relating to acoustical inputs nearly in real time. But medical research knows bursts – random like periods of rapid spiking followed by silent periods [26] of nerve cell soma.

On the other hand, the theory of Neural Nets (NN) gave lots of ideas, how nerve cells can learn. But nearly nothing is known about the basics: How does nerve system work?

From standpoint of electrical filter theory nerve nets have no processor for the Fast Fourier Transform (FFT). Our sound and code analysis systems work in time domain. A basic approach for filter theory in time domain is "convolution" (Faltung). We will test convolution to switch between system responses of nerve nets, behavior and net structure.

Thinking about behavior of nerve systems the author proposed in 1993 the mathematics of "convolution" [16] to understand bursts. Although this is a theoretical abstraction, it could be helpful for a better understanding of code selection or sound detection in nerve nets. Let's analyze this 20 years old idea more in detail.

Simplified, any convolution process can be seen as unified multiplication of two series or functions. If we have two time-functions, we need to mirror one and to shift it contrarily over the other – folding a paper sheet, this can be done very easy. Thus the procedure is given the name "Faltung" or "convolution".

II. HISTORY OF "CONVOLUTION"

Implicitly, the knowledge about convolutive functions [14] seems to be very old. For example, the "Cauchy product" [8] showed the discrete convolution of two sequences. The English Wikipedia [27] tells about comparable formulas of S.F. Lacroix, P.S. Laplace, J.B.J. Fourier or S.D. Poisson. But the terms "convolution" or "Faltung" appeared years later.

For the first time, David Hilbert used the word "Faltung" in 1906 [10], but not in the sense we use nowadays.

Vito Volterra analyzed "fonctions permutables et leurs compositions" [13]. In 1913 he introduced the integral form and showed equality of a today unknown symmetrical form and the known unsymmetrical form of the convolution integral (which seems to suggest, one of the input functions is standing), but without using the terms "convolution" or "Faltung".

Following own investigations, the name "Faltung" and the known formula was used for the first time by Felix Bernstein [1] in 1920. Beginning 1922, Gustav Doetsch spread the terminus to the public with his works about – what he has called - the "Laplace-Transformation" [2,3,4].

Using discrete time series of data-streams (PC) a discrete form of Laplace-Transformation became interesting. Following the English Wikipedia, Ragazzini and Zadeh [12] introduced 1952 the discrete form. In the 1960's Dobesch wrote interesting introductions [5,6] about the discrete form, nowadays called "z-Transformation".

Workshop „Autonomous Systems",
Hotel Sabina Playa, Cala Millor, Mallorca, 13.-17. Oct.2013

After world war II the German word “Faltung” was more and more replaced by the English word “convolution”.

Interesting properties were found for systems characterization in circuit theory, suggesting the standing time function is called *transfer function* or *pulse response*. For example, if the system input is a spike, the system answers with the transfer function. If it is a ramp, the system integrates over the input, while the DC output gain is the sum of the coefficients and so on.

After 1957, convolution obtained a central rule with Mikusinski’s operator calculus [9] for time-domain calculation in linear systems.

III. Convolution Basics

Most importance has convolution up to now for digital filter theory in linear, time-invariant (LTI-) systems. In practice, a Finite Impulse Response Filter (FIR) is a direct technical implementation of convolution. Semiconductor integration technologies of modern receivers (WLAN) use FIR- and IIR- filters within the modulation and demodulation process of signals. Analog-Digital Converter circuits (ADC) use them to band-limit signals.

In our sense convolution can show the flow of one time function *x(t)* over a (standing) other *z(t)*, resulting in a third time function *y(t)*, whereby one time function appears time inverted (folded).

$$y(t) = x(t) * h(t) = \int_0^t x(t-\tau)\, h(\tau)\, d\tau \qquad (1)$$

In discrete time it is the series of convolution sums in form of the Cauchy product [8] – the universal multiplication of two vectors.

$$y_n = \sum_{k=0}^{n} h_k \; x_{n-k} \qquad (2)$$

Equation 1 shows the convolution in integral form, equation 2 is the discrete, computer-based style. Find convolution examples for Scilab in [24].

We suggest, incoming functions agree in domain of definition. Time functions should be nonzero only in the region of interest.

IV. Interference Network (IN)

Any information processing in nerve system needs hundreds of synchronous, parallel pathways to carry successfully a dominant signal [28]. But all the pathways (branches) carry information very slow. And any pathway has different thickness, length and delay. Nerves doesn’t go straight through the body. Thus synchrony of many of them is practically impossible. Synchrony has to appear in a new sense: information processing can only happen at locations of interference of the signal, that comes multiple from far away. This concept, developed in “Neuronale Interferenzen” 1993 [16] demanded a new, specific wave-like information theory, softly comparable to optics.

Different new, theoretical concepts appeared, like waves of information, interference integrals (used for the investigation of acoustic cameras) or cross- or self-interference to distinguish between “seeing” and “hearing”.

In the 1990th, my talks about pulse-propagating, artificial neural nets (ANN) or time-delaying neural nets (TDNN) caused permanent misunderstandings. Thinking about pulse interferences in large and slow networks I found, that any smallest delay mismatch changes the information processing of the net. State machine abstractions in ANN- or TDNN-theories destroy the function of the net. Therefore, between 1995 and 1997 I proposed the new term “Interference Network” (IN) for the precise modeling of spiking and delaying neural networks with nerve-like properties [17…23].

The basic idea is, that the reason for the processing capability of a nerve network lays in different delays on nerve fibers – not in the way, how to learn or how to construct the weights. Neural network theories until 1993 had artificial clocking structures, destroying every natural system response of the network.

A first step to apply the idea of the very special processing capabilities of IN was the manuscript “Neuronale Interferenzen” (1993) [16]. The second step was a simple application of an IN to produce acoustic photos and films between 1994 and 1996. The “Acoustic Camera” (AK) was born [25]. But the AK became a self-runner, the theory behind the central and successful IN-idea remain unknown. The third step is the analysis of serial code detection – this work.

V. Abstract Model of a Nerve Network

We assume a simplest, (probabilistic[1]) interference network (IN) to model a nerve net on a high abstraction level. The net consists of nodes and branches. The nodes combine information of connected branches and refreshes the outputs. Branches have only the task, to delay time functions relative to their width and -length. The amplitudes are unchanged. In nerve system we find behind analog signal levels spike-like pulses with binary level, we simplify “0” or “low” for -60…-70 mV and “1” or “high” for +30 mV (spike). Output functions of nodes regenerate the levels.

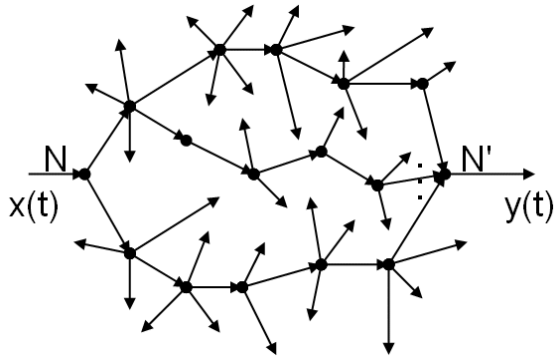


Figure 1. Abstract, diffuse nerve cell arrangement. Different neurons carry paths between neuron N and N’. Each path has a different delay.

Using a branch of delay τ, it delays the incoming time function *f(t)*, delivering an output $f(t-\tau)$. If the branch has to deliver a time function at $\tau=0$, we have to pre-delay the input (non-causal) with a *positive* τ,

$$f(t) \rightarrow f(t-\tau), \qquad f(t+\tau) \rightarrow f(t). \qquad (3)$$

[1] A pyramidal neuron has round 7400 synapses [28].

Following an old idea in Kap. 8b on page 181 of [16] we construct a simplest IN-model to understand convolution properties in a simplified, pulsing nerve system.

Two nerve cells *N, N'* shall have the ability to communicate over a number of *n* dendritic- and/or axonal branches together. Each branch has a different signal delay τ_i. The next circuit (Fig. 2) simplifies the net of Fig. 1 again.

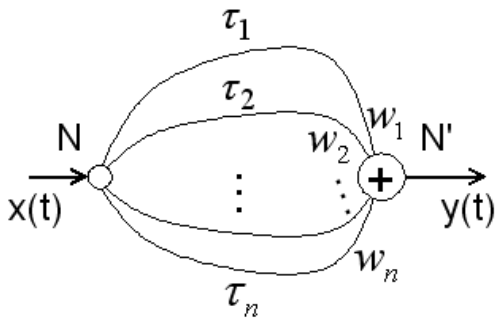


Figure 2. Simplified abstraction of the network of Fig.1. The two cells *N, N'* connect over branches with different delays τ_i and weigths w_i.

Any incoming time function *x(t)* is delayed on the different paths with different delays τ_i . So any single time function at neuron *N* runs independently delayed over the network, producing at *N'* an input after the delay. Neuron *N'* multiplies its inputs with the weights w_i and adds the incoming time functions. Last but not least, any level adaptor (with additional threshold function etc.) guarantees output *y(t)* levels in the same range {0…1} as input levels *x(t)*. To calculate the output *y(t)* we add all $x(t-\tau_i)$ multiplied by weight w_i. Without threshold, the net output can be for example

$$y(t) = \frac{1}{n}\sum_{i}^{n} w_i x(t-\tau_i)\,. \qquad (4)$$

In this simple, abstract form, the network can be represented by a *vector of delays T* and a *vector of weights W*, the index is the branch number

$$T = [\tau_1, \tau_2, ..., \tau_n], \qquad (5)$$

$$W = [w_1, w_2, ..., w_n]\,. \qquad (6)$$

The delay vector *T* has an interesting property, we need later: it resists the addition or subtraction of global delay constants τ. The network function stays unchanged by variations of τ

$$T = T \pm \tau\,. \qquad (7)$$

The function of the net is simple. Any input *x(t)* appears with a different delay at neuron *N'*. If the input is a single spike, the output of the net is a burst. Changing the weights, the function can be opposite: giving the specific burst that opens the net, the net answers with a spike. The net acts like a keyhole. Only the right pattern – the key – can open it. In this work we will consider both cases, the code-generation and the code-detection of the net.

The net has practical importance. Overall in nerve system we find bursts. But in the living animal not so much is known about nerve connections. We have no tools to see or to get the net structure. Moreover, using a microscope, without chemical staining we can nearly not find a single nerve fiber.

Is it possible to reconstruct partial nerve structures from burst records?

VI. Construction of Transfer Function

The idea behind the following algorithm is to transform the net structure into a filter structure. Interpreting the transfer function *H* of the filter, we can re-translate the delay structure. Inspecting the figures in Kap.8b, p.181 [16] we find analogies to finite impulse response (FIR-) filters. Sorting delays by a fixed delta-T, an association to convolution appears.

If we interpret a *transfer function H* of a linear system (in time domain), *H* can be seen as a discrete time function with sample rate f_s and with growing index *i*

$$i = [\ldots\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ \ldots] \qquad (8)$$

Each sample index (2…9) can carry a zero (for nothing) or a number for a weight. Relative to any fixed starting point, the index of the sample carries the relating delay. Interpreting a real transfer function, for example

$$H = [\ldots\ 0,\ 1,\ 0,\ 0,\ -1,\ 0,\ 1,\ 0,\ldots] \qquad (9)$$

we interpret the first *1* as the weight for the delay index 2. The next *-1* is the weight for delay index 6. The last *1* is the weight for delay index 8. And the delay of index 8 is eight times the sample interval. The distance between following values of transfer function *H* is the sample interval t_s (of the sample rate $f_s = 1/t_s$). So our transfer function *H* example represents three delays, $T = [3t_s, 6t_s, 8t_s]$ with the corresponding weights $W = [1,-1 \text{ and } 1]$.

In other words, if the *T*-vector contains a value of 8 ms and the sample rate is 10 kHz, we have to fill the index number 80 (8*10) of *H* with the corresponding weight. Thus, any resulting *H*-vector will be longer than the generating *T*-vector.

It is possible, that the delay-vector *T* contains some identical values at different places. In *H*, we have to add the corresponding weights.

To construct *H*, we follow the following steps:

a) Multiply the *T*-vector with the sample rate, that *H* will contain. Recall, all members of the *T*-vector must be integers, they are later index numbers in *H*. The length of *H* is larger than the maximum delay in *T*, we need *max*(*T*) to construct the whole, empty *H*-vector, we find

$$length(H) = max(T). \qquad (10)$$

b) Take a delay τ_i of the *T*-vector. Add the corresponding weight w_i (with the same index *i*) at the position in *H* that is defined by the index *i*, as in the following formula

$$H(T(i)) = H(T(i)) + W(i). \qquad (11)$$

c) Do task b) for all delays and weights.

Let's bring this ideas into an algorithm. For automatic conversion of the net structure including the vectors (*T,W*) into

the system transfer function *H* the Scilab-function *H = trans(T,W,fs)* is introduced. An additional parameter is the sample rate *fs*, it defines the delay time between subsequent values of the *H*-vector. For a better understanding of the code we use uppercase letters for vectors and lowercase letters for single numbers and values. Vectors start in Scilab with the first element indexed by number one (not with zero).

```
function [H] = trans(T,W,fs);
 if length(T) == length(W) then
  T = T * fs;       // apply sample rate of H
  T = round(T);     // T becomes index - integer
  H = 1:max(T); H = H * 0; // create an empty H
  for i = 1:length(T),     // for all T(i), W(i)
    j = T(i),       // delay becomes the H-index
    H(j) = H(j) + W(i),    // add the weight to H
  end // for
 else // if
  printf('\n\nerror: T and W have different size\n');
 end // if
endfunction;
```

To minimize the size of the resulting transfer vector *H*, it is possible to subtract the minimum delay of *T* from all delays of T (compare with Eqn.7)

$$T = T - (min(T)+1). \tag{12}$$

The addition of *+1* points to the index number one in *H*. This is comparable to removing leading zeros of *H*. At the end of *H*, zeros can be removed too. But going into the frequency domain, any changing at frontal zeros can cause problems, because it changes the transfer function.

VII. Applying a Convolution

Our time function *y(t)* can be characterized as convolution of the input time function *x(t)* and the transfer function *h(t)*.

$$y(t) = h(t) * x(t) \tag{13}$$

Using vector/matrix style syntax for discrete convolution we write[2] here

$$Y = H * X, \tag{14}$$

Y, *H* and *X* are vectors. For the computation of convolution within Scilab, we use the *convol*-function

$$Y = convol(H,X). \tag{15}$$

In mathematics and so in Scilab, the orientation of *H*- and *X*-vector is opposite. If we like to use unidirectional vectors, we should use *cross-correlation* instead of convolution.

Please test the Scilab-sources (freeware) in [24] to try some own experiments.

VIII. Spike Output

We know different philosophical speculations about the general function of nerve system. The favorite in this paper is the presumption, a nerve cell fires, if it has detected the code that was learned before. With this suggestion, the nerve cell has to fire, if the input code (vector *X*) is identical (using convolution: time-inverse) to the transfer function *H*. The transfer function acts as a keyhole, the input function as the key.

In other words, to implement such a nerve-like behavior, we are looking for an input function *X*, that gives a resulting *Y* in form of a Dirac-like spike of the level a^2

$$Y = X * H = [\,0, 0, \ldots a^2, \ldots 0, 0]. \tag{16}$$

As solution we only find a single trivial case. If the resulting *Y* becomes a Dirac-like pulse, *X* and *H* have also to be one-shots. For example, for *X = rev(H) = [0,a,0]* we get $Y = convol(X,H) = [0,0,a^2,0,0]$.

To solve our task better, we have to find powerful approximations.

If we remember, that the comparable transformation of convolution is the cross-correlation, we find, that we get highest cross correlation for identical *H* and *X*. Translating to convolution this means, *X* has to be exactly the time-inverse to *H*. This might be a first result. So I wrote a *rev()* function [24] for time-inversion.

From RADAR-technology we know useful approximations of comparable kind, like chirps or Barker-codes [29] to get sharp pulses at the end of a complex transmission chain. Using any Barker-code, we get best approximations for the behavior we need. For example, if we use the Barker-code of length 5 the convolution is

$$H = [1, 1, 1, -1, 1], \quad X = rev(H), \tag{17}$$

$$Y = convol(X,H) = [1, 0, 1, 0, 5, 0, 1, 0, 1]. \tag{18}$$

As longer is the Barker-code, as higher becomes the Dirac-pulse relative to the environment. No wonder, that modern wireless-, GPS- and RADAR -technologies use Barker-codes for example in form of Direct Sequence Spread Spectrum (DSSS) or Pulse-Compression technologies. However, there is a problem: nerve systems do not have negative pulses.

IX. Frequency Analysis

Frequency analysis implies, to analyze a network with a Fourier series of *sin*- and *cos*-functions, corresponding to the Euler formula as solution of Eqn.19. This appears to be an antagonism for spiking networks, spikes are not sinusoidal. However, if we try it, we find special properties of interference nets, that associate with the well known existence of synapses of the exciting and the inhibiting type.

To analyze frequency properties of *H*, we apply the z-Transformation, substituting *H(n) by F(z)* with $z = \exp(\sigma+j\omega)$ to calculate the complex frequency transfer function. Using $\sigma = 0$, we get the Time-Discrete Fourier Transformation (TDFT) in frequency domain. To get the *frequency transfer function* (Fourier spectrum) *F* we use the symbolical equation

$$F(e^{j\omega}) = \sum_{n=-\infty}^{\infty} H(n)e^{-j\omega n}\,. \tag{19}$$

[2] Notation: A star (*) denotes the convolution (in analog- or matrix form). For matrix multiplication we use an upper dot (·). For the "dot product" of vectors Scilab uses the notation dot-star (.*). If necessary, we will follow this style in our WWW-examples.

The term ωn relates to a "digital frequency" $\omega = 2\pi f/fs$.

Using Scilab [24], we write for the absolute vector of the Fourier spectrum $F = abs(fft(H))$. With some additional scaling we get a Fourier spectrum with identical number of coefficients between input vector H and output vector F.

For example, if H has $c = length(H) = 100$ coefficients, the resulting F will get 100 coefficients from $0…fs$. If H has a sample rate of 10 kHz, F will stop at 10 kHz. Each F-sample is $max(F)/c = 10$ kHz/100 = 100 Hz wide. To suppress Nyquist mirroring, we stop the plots at half the sampling frequency $fs/2$, using only the lower half of the coefficients of F, see example in Fig.5 and Scilab examples in [24].

Doing some exercises, we find a behavior that is well known in systems theory: Using an *unipolar* input X and an *unipolar* transfer function H (unipolar means with a level range between zero and one), *without exception* the FFT-output has its maximum at the lowest frequency. It is clear, that the DC-power of an unsymmetrical time function is high. What can this mean?

In our case it is higher than every convolution result at higher frequencies. In other words: it is generally impossible to use *unipolar* networks for series- or frequency analysis – anyway. We need a bipolar system with positive and negative weights to analyze serial codes (sounds etc.). We imagine the problems, that Fuzzy Sets or Artificial Neural Networks have, using unipolar functions only.

X. Unipolar or bipolar Signal Levels?

Thinking about exciting and inhibiting synapses in nerve system, a question is, how to model them simple and adequate on a highest abstraction level in systems theory. The existence of the exciting, postsynaptic potential (EPSP) and the inhibiting postsynaptic potential (IPSP) (see for a good introduction for example [28]) allows a bipolar transfer function H with exciting and inhibiting weights. But it makes no sense to introduce bipolar inputs X. We only know *unipolar, positive pulses in nerve systems*.

By contrast, Barker-codes [29] use the *bipolar* level interval in the range {-1…0…1}, nerve pulses are *unipolar* in the range {0…1}. Convolution of codes using unipolar signals is less efficient compared to bipolar codes, for example in the case

$$H = [1, 1, 1, 0, 1], \quad X = rev(H), \tag{20}$$

$$Y = convol(X,H) = [1, 1, 2, 2, 4, 2, 2, 1, 1]. \tag{21}$$

In comparison to equations 17 and 18 the resulting pulse is less sharp.

Considering all aspects, it seems necessary to introduce a *bipolar* transfer function H (for synaptic weights) but an *unipolar* transmission function X over branches to model nerve pulses. To test this, a random bipolar H can generate an unipolar X using the Scilab-function *clip()*,

$$X = rev(\ clip\ (H,\ minlevel,\ maxlevel)). \tag{22}$$

This function sets all coefficient values below *minlevel* to *minlevel* (here zero).

Trials with identical random noise show a nice surprise: Fig. 3 shows, that the resulting spike shape is not really significant influenced using bipolar or unipolar inputs for X. But using an unipolar value range for both, X and H, we find significant, worse results, especially the DC-power in the FFT, referring to the slowly growing convolution values near the peak.

For nerve nets this could mean, that evolutionary introduction of *bipolar transfer functions* with inhibition and excitation brings (beside the frequency selectivity) a high win in the ability, to built a robust and frequency selective information processing.

By contrast, the introduction of additional negative pulses would bring a further small win, but the absence of negative pulses in animal kingdom shows, evolution decided to see this as not really significant.

XI. Interpreting Bursts

Is it possible to reconstruct a net structure with vectors T and W from a given pulse response H? Trying it we get the net vectors back in sorted order. The Scilab-algorithm is the opposite to the construction of the transfer function. Each index in H corresponds via the sample rate with a delay in T. The H-value at that index is the corresponding weight in W.

```
function [T,W] = net(H,fs); // returns T and W
    j=1;
    for i=1:length(H) // H index i
        if H(i) == 0 then ; // do nothing
        else  // write the value to W, the index to T
            W(j) = H(i);
            T(j) = i;
            j = j+1;
        end; // if
    end; // for
    T = T ./ fs;          // remove sample rate
    T = T - min(T);       // scaling
endfunction;
```

Different publications on nerve system show uncountable measurements of bursts in nerve system [26,28], many papers on measurements report them. The question is: has bursting something to do with net structure and the vectors T and W?

Supposing, the reason of a burst is a network with *[T,W],* the delay and weight structure of that net creates a transfer function H directly as a burst. In return, the measurement of a burst can show the transfer function of a partial net! The function *[T,W] = net(H,fs)* gives the net structure back, if we give a burst measure into H, fs is again the sampling rate.

To summarize bursting and convolution:

Inspecting a net structure *[T,W]* comparable to Fig.2 it is possible to convert it into the transfer function H and vice versa.

We name a Dirac-like delta function (a spike or pulse of variable tallness) with D (it is a vector, thus we write it uppercase).

Open a door: If input X is the key for the keyhole H with $X = rev(H)$, then a spike D appears at the output Y,

$$X * H = Y: \qquad rev(H) * H = D. \tag{23}$$

Pulse response: Is X a single one-shot (P), the convolution with H gives the pulse response (this is H) at the output

$$X * H = Y: \qquad P * H = H \tag{24}$$

XII. EXAMPLE

Fig. 4 shows the Scilab-result of a convolution example, if the key is the right one. We find the input function *X* opposite to the transfer function *H*. In the last row, Scilab calculates our threshold function. Using a single spike as input, the example can show, why the Laplacian transfer function is called the "pulse response".

Using a sampling frequency of f = 1 kHz (1 ms) the delay mask *T* contains delays of 3 branches in milliseconds

$$T = [\tau_1, \tau_2, \tau_3] = [5,\ 3,\ 8]. \qquad (25)$$

The corresponding weight vector *W* of inputs has the values

$$W = [w_1, w_2, w_3] = [1,\ 0.5,\ 1]. \qquad (26)$$

A positive value stands for excitation, a negative for inhibition of the input.

The delay vector is robust against addition or subtraction of constants, the response code will not change.

FIR-form, reducing form: To get a minimum vector length for the transfer function *H*, delays reduces by the minimum delay $\tau_2 = 3$. After index sorting we get

$$T_R = [\tau_{R1}, \tau_{R2}, \tau_{R3}] = [0, 2, 5]. \qquad (27)$$

To get the transfer function *H*, the weights place at the corresponding delay number, that means a weight w_i is to set at the position of the index corresponding to delay τ_i

$$W_R = [w_{R1}, w_{R2}, w_{R3}] = [.5, 1, 1]. \qquad (28)$$

This is calling the reduced form or FIR-form of the net structure. Automatically, the reducing form results of any reconstruction with function *[T,W] = net(H,fs)*.

We recall, *H* is a linear spaced time axis, subdivided by the sampling clock, the difference between subsequent samples equals the inverse sample rate *1/fs*, here 1 ms. The index of *H* is then

$$index(H) = [...1, 2, 3, 4, 5, 6, 7, 8, 9, ...]. \qquad (29)$$

The original, corresponding weights occupy the positions (5, 3, 8), ignoring leading and trailing zeros we get

$$H = (w_2, 0, w_1, 0, 0, w_3). \qquad (30)$$

The result is the transfer function *H* of the network

$$H = (0.5,\quad 0,\quad 1,\quad 0,\quad 0,\quad 1), \qquad (31)$$

see Fig.4. The result is the same, using reduced vectors T_r, W_r or original vectors *T,W*.

Doing the convolution we use an equal spaced input vector *X* with identical sample rate. If the vectors *X* and *H* differ in size, the shorter vector fills in the Scilab-example with zeros. The plot sizes between input and output differ, convolution of *length(X) = n*, *length(H) = m* gives *length(Y) = n + m – 1*, we fill the rest with a zero vector.

If the output is not normalized by the inverse number of inputs *1/n*, finally, we append any threshold function *U(k)* to reconstruct the output level. The sum of weights is $w = (.5 + 1 + 1) = 2.5$. Epsilon should be smaller than the smallest weight (0.5). Choosing $\varepsilon = 0.3$, the threshold function works correctly. If the output function comes in the region of the sum of weights (ω–ε), the output fires. Epsilon (ε) has to be smaller than the smallest weight.

$$w = \sum_{i=1}^{n} |\, w_i \,|, \quad 0 < \varepsilon < \min(|\, w_i \,|), \qquad (32)$$

$$U(k) = \begin{cases} 0, & if \quad Y(k) \le (w - \varepsilon) \\ 1, & if \quad Y(k) > (w - \varepsilon) \end{cases}. \qquad (33)$$

We find, that any *1/n* norm is not really a best solution. Using different variable types and vectors, the method *(ε · max(Y))* seems to be more robust for wide variations of tasks, see Scilab-code [24].

In the example Fig. 4, we use a nearly identical, but opposite vector *X* as a key, the output of the neuron gives a single spike: the neuron tells us: "I understood the serial code!".

Besides GIF- and PS-plot outputs the Scilab-procedure *transfunc.sce* (download [24]) gives an additional text-file output with some system values:

```
conv_3Spikes_noclip_1kHz_all.txt
sample rate      fs = 1 kHz
original delays   T = [5 3 8]
original weights  W = [1 0.5 1]
reduced delays   Tr = [0 2 5]
reduced weights  Wr = [0.5 1 1]
transfer fct H = [0 0 0 0.5 0 1 0 0 1 0]
key input    X = [0 1 0 0 1 0 0.5 0 0 0]
output Y = [0 0 0 0 0 1 0 1 1 0 2 0 0 1 0 1 0 0 0]
thresh U = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
abs(F) =
[2.5 1.4 0.5 1.1 1.8 1.9 0.5 1.9 1.8 1.1 0.5 1.4]
generator: transfuncnet.sce, 13-Aug-2013
www.gfai.de/~heinz/techdocs
```

Increasing the sample rate *fs*, the transfer function becomes proportional longer. Setting *fs* = 10 kHz we get a different *H* with weights at the index positions (50, 30, 80) or for the reduced and sorted vector at (0, 20, 50).

If the transfer function is periodic (see web-examples in [24]) a Fourier analysis can produce interesting results. The Scilab code generates in a last step a new plot window, showing the transfer function *H* and a rectangular, absolute Fourier window of *H*, see example in Fig.5.

XIII. CONCLUSION

To characterize abstract properties of nerve networks in time- and frequency domain, following [16], Kap.8b, p.181 an elementary net can be simplified using a *linear, time-invariant (LTI-) system description* containing a delay vector *T* and a weight vector *W*, Fig.1 and Fig.2.

A procedure *[H] = trans(T,W,fs)* calculates the (time-discrete) *transfer function H* (pulse response) of the net from *delay vector T* (delay mask) and *weight vector W*.

The inverse procedure *[T,W] = net(H,fs)* reconstructs the net structure *[T,W]* from transfer function *H*.

Interpreting burst measurements of nerve nets as transfer function *H* of a partial network, the reconstruction of the net structure *[T,W]* is possible.

An example shows, how the (delay) structure of the net codes its serial behavior. Weights are the exciting or inhibiting connections, Fig.4.

Inspecting frequency domain of unipolar and bipolar *H* and *X,* the maximum DC-level appears for nets with unipolar input vector *X* and unipolar transfer function *H*. In this sight, unipolar nets, common in ANN and Fuzzy theories, do not have the ability to detect serial codes or frequencies.

Analyzing convolutions with bipolar and unipolar signal levels of interfering time-functions the natural, mixed usage of *unipolar inputs* (positive spikes) and *bipolar transfer functions* (exciting and inhibiting) seems to be an interesting compromise between pure bipolar (+1…-1, best) and pure unipolar (0…1, worse) levels, Fig.3.

Find Scilab code for the given exercises and different convolution examples with interference nets on the website [24].

XIV. Thanks

Thanks to Prof. Dr. Wolfgang Halang and Prof. Dr. Herwig Unger (FernUni Hagen) for the given encouragement to proceed publications about properties of Interference Networks. Thanks to Dr. Emeterio Navarro (GFaI) and Jutta Dühring (Uni Hagen) for their important help in the review process. Special thanks to my wife for her patience over the last twenty years.

A last thank is given a supporter of the first hour of the IN-theory. We remember Prof. Dr. Hans-Heinrich Bothe (HTW Berlin / TU Berlin / Uni Kopenhagen-Lyngby), he died against a rock fall in the Südtiroler Alpen at July 30, 2013. With his warmness, persistence and power to organize large NAISO/ICSC Fuzzy-/Neuro-Conferences, Hans remains in our hearts.

XV. References

[1] Bernstein, F.: Die Integralgleichung der elliptischen Thetanullfunktion. Sitzungsberichte der preussischen Akad. der Wiss. XL, 21. 10. 1920, S. 735-747, www.dwc.knaw.nl/DL/publications/PU00014734.pdf

[2] Doetsch, Gustav: Die Integrodifferentialgleichungen vom Faltungstypus. Mathematische Annalen 89 (1923), Springer Verlag Berlin. Editor: Albert Einstein. http://gdz.sub.uni-goettingen.de

[3] Doetsch, Gustav: Theorie und Anwendung der Laplace-Transformation. J.Springer Berlin, 1937, 436 S.

[4] Doetsch, Gustav: Anleitung zum praktischen Gebrauch der Laplace-Transformation. R.Oldenbourg, München, 1956

[5] Dobesch, Heinz: Laplace-Transformation von Abtastfunktionen. Verlag Technik Berlin, 1969

[6] Dobesch, H., Sulanke, H.: Zeitfunktionen. Verlag Technik Berlin, 1969

[7] Fettweis, Alfred: Beweisansatz der Identität zwischen Interferenz- und Faltungsintegral. Persönliche Kommunikation per Email, Bochum, 4.11.2011, www.gfai.de/~heinz/publications/animations/beweis.pdf

[8] Cauchy product see http://en.Wikipedia.org/wiki/Cauchy-product

[9] Mikusinski, Jan: Operatorenrechnung. VEB Verlag der Wissenschaften Berlin, 1957. Polish: Rachunek Operatorów. Warszawa 1957.

[10] Hilbert, David: Grundzüge einer allgemeinen Theorie der linearen Integralgleichungen. Vierte Mittelung. Nachrichten von der Königl. Ges. der Wissenschaften zu Göttingen, Math.-physik. Klasse 1906. Berlin, Weidmannsche Buchhandlung 1906, (157-227), S.159, http://gdz.sub.uni-goettingen.de/en/dms/load/img/?PPN=PPN252457811_1906&DMDID=dmdlog28

[11] Poisson, Siméon Denis: Mémoire sur la variation des constantes arbitraires, dans les questions de méchanique (Variation of incremental constants, pages 1-70). Mémoire sur la théorie des ondes (Theory of Waves, pages 71..186). In: Mémoires de l'académie royale des sciences de l´institut de France, depuis l´ordonnance du 21 mars 1816. download http://books.google.de/

[12] Ragazzini, J.R., Zadeh, L.A.: The analysis of sampled-data systems. Trans.Am.Inst.Elec.Eng. 71, Part II: 225-234, 1952. (en.Wikipedia.org)

[13] Volterra, Vito: Leçons sur les équations intégrales et les équations intégro-différentielles. Paris: Gauthier-Villars, 1913. (Google Books Reprint from University of Michigan Library, Lexington, KY, USA).

[14] Woltersdorf, Lothar von: Einige Klassen quadratischer Integralgleichungen. Sitzungsberichte der Sächsischen Akademie der Wissenschaften zu Leipzig, Band 128, Heft 2. Hirzel Verlag Stuttgart, 11/2000, ISBN 3-776-1091-7

[15] Wunsch, Gerhard: Geschichte der Systemtheorie. Oldenbourg Verlag 1985, Kap. 1.3.1.3.

[16] Heinz, G.: Neuronale Interferenzen. Manuscript and collection. 1993, 301 p., www.gfai.de/~heinz/publications/NI/index.htm

[17] Heinz, G.: Zur Mathematik des Nervensystems Raum-Zeit-Projektionen und Interferenzmuster zwischen verbundenen Wellenräumen.http://www.gfai.de/~heinz/historic/pressinf/bilder_d.htm

[18] G. Heinz, "Non-recursive interference calculi – a mathematical calculus Immanent in nervous activity", in Unger et al. (Eds.): Autonomous systems, development and trends. Springer SCI 391, 2011, pp.171-186

[19] G. Heinz, "Integrating System-Theory into Interference Networks - relation between convolution integral and interference integral", Lecture. Workshop "Autonome Systeme", Oct 30 - Nov 3, 2011, Gran Camp de Mar, 07160 Camp de Mar, Andratx (Mallorca, Spain). Homepage: www.gfai.de/~heinz/publications/animations/convolution.htm

[20] G. Heinz, "Interference Networks as Generalizing Signal Theory between Integral Transformations, Neural Nets, Optics, Electrics and Acoustics", Plenary invited lecture, 7th Int. Conference on Computing and Information Technology IC2IT, May 11-12, 2011, King Mongkut's University of Technology, Bangkok.

[21] G. Heinz, "Two Decades of Interference Network Research", Keynote speach, Heinz, G.: Two Decades of Interference Network Research. Keynote speach. 3rd International Workshop on nonlinear Dynamics and Synchronization (INDS'11) and 16th International Symposium on Theoretical Electrical Engineering (ISTET'11). University Klagenfurt, Austria, Jul 25-27 2011 (abstract and ppt on the web).

[22] G. Heinz, „Von der Erfindung bis zum weltweiten Vertrieb – Zur Entwicklung der akustischen Kamera". VDI-Bezirksverein Berlin-Brandenburg, Arbeitskreis Technikgeschichte Dr.-Ing. Karl-Eugen Kurrer und Dr. phil. Stefan Poser, Vorträge im Deutschen Technikmuseum, Reihe Geschichte Moderner Technik, 27. Januar 2011

[23] G. Heinz, "Introduction to Wave Interference Networks". Mallorca-Workshop 2010 'Autonomous Systems' 24.-29.10.2010 Camp de Mar, Mallorca, Spain, Shaker 2010, ISBN 978-3-8322-9514-1, pp.87-94, http://www.gfai.de/~heinz/publications/papers/2010_autosys.pdf

[24] G. Heinz, Codeselektion nervlicher Art, Scilab-Examples, http://www.gfai.de/~heinz/techdocs/index.htm#conv (7/2013)

[25] Heinz, G.: Zur Physik bildgebender Rekonstruktion akustischer Bilder und Filme im Zeitbereich. 33. Deutsche Jahrestagung für Akustik, DAGA 2007, Universität Stuttgart, Vortrag 243, 21.3.2007 http://www.gfai.de/~heinz/publications/papers/2007_DAGA_Reko.pdf

[26] Bursts in nerve system: http://en.wikipedia.org/wiki/Bursting (7/2013)

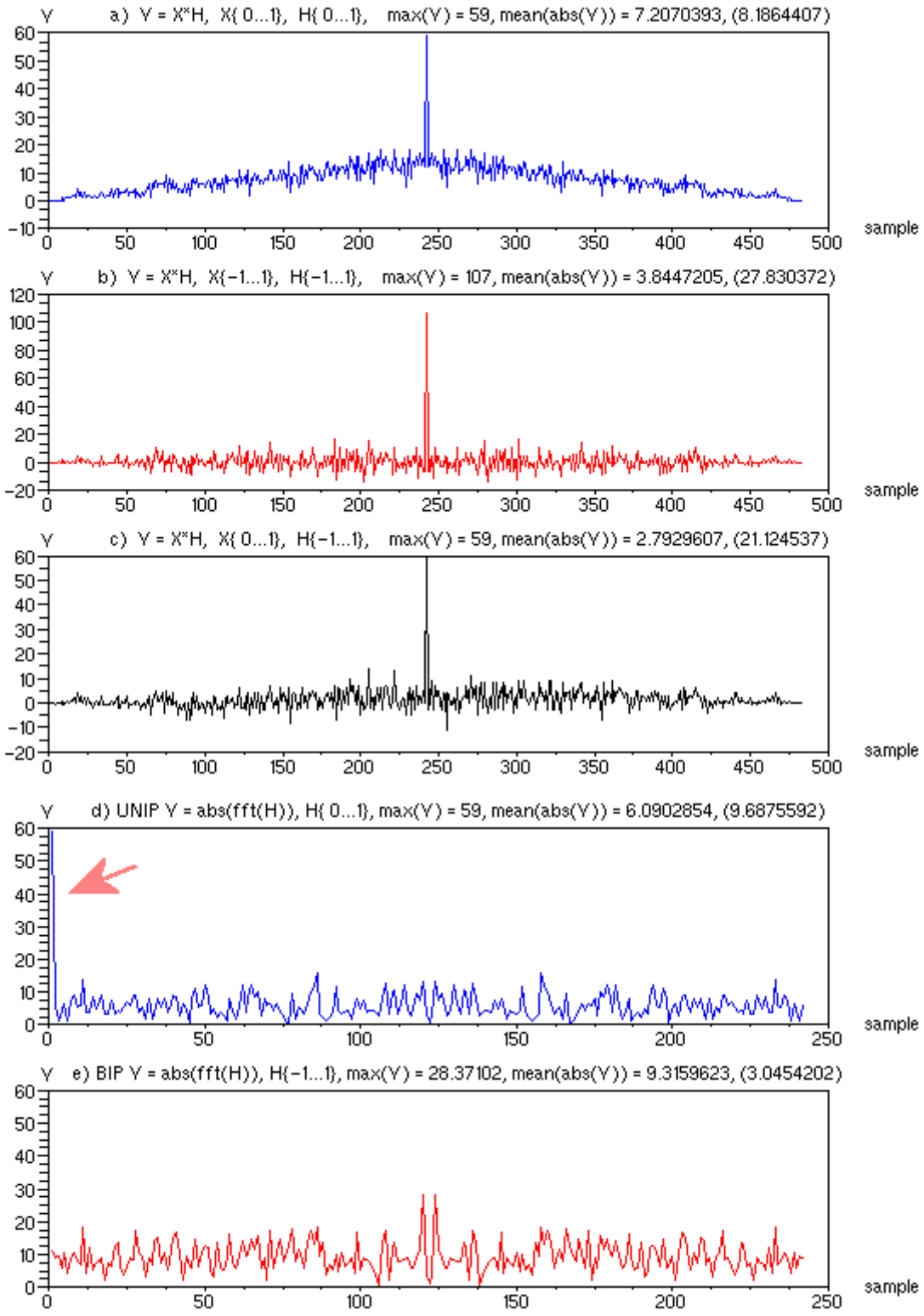
[27] Intro to convolution: http://en.wikipedia.org/wiki/Convolution (7/2013)

[28] Eccles, J.C.: Das Gehirn des Menschen. Seehamer Verlag, 2000.

[29] Barker codes http://en.wikipedia.org/Barker_code (7/2013)

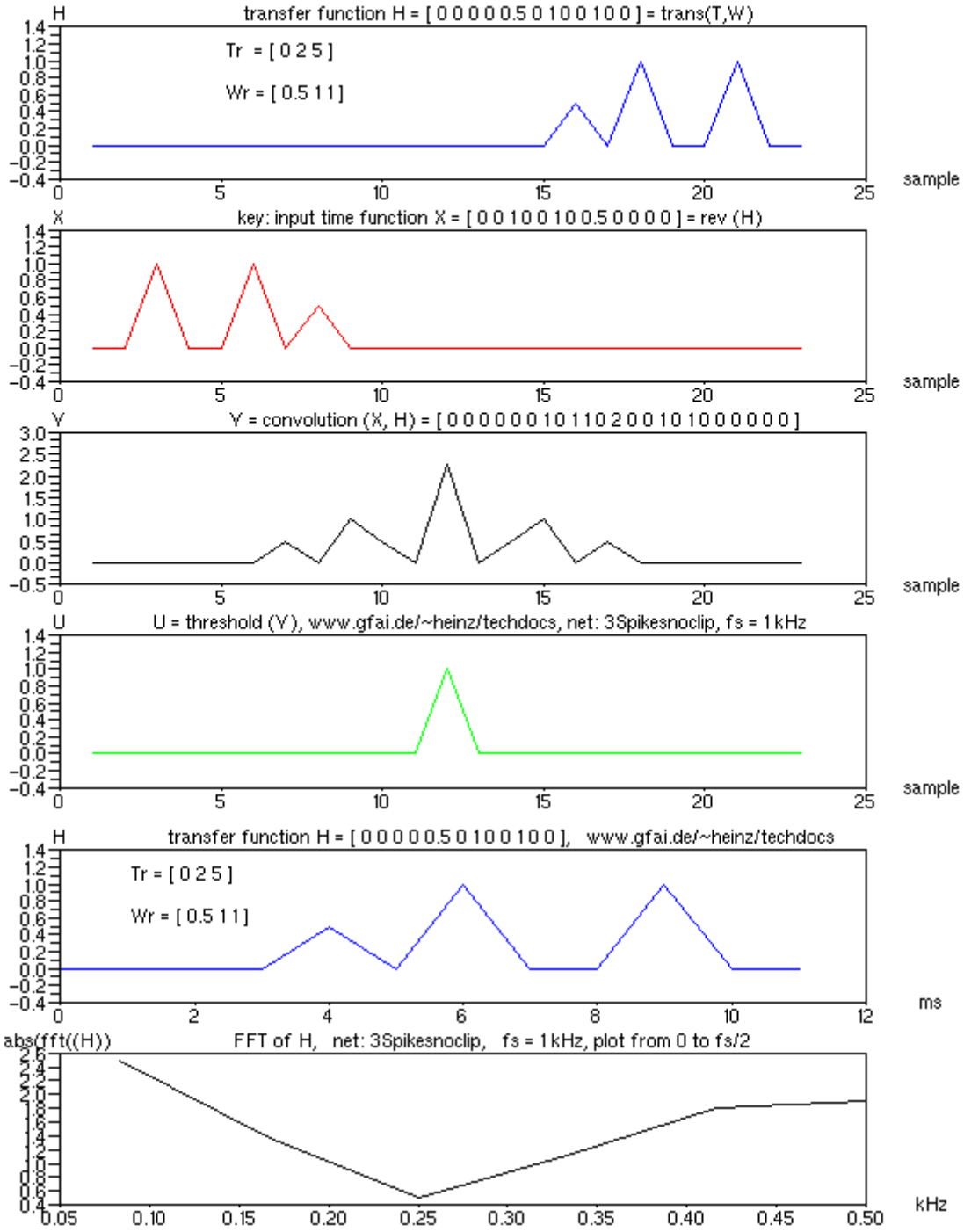
[30] Rumelhart, D.E., McClelland, J.L. et al.: Parallel Distributed Processing. Vol.1&2 MIT Press Cambridge, MA/ London, England, 1986.

[31] Anderson, J.A., Rosenfeld, E. et al.: Neurocomputing, Foundations of Research. Vol.1&2 MIT Press Cambridge, MA/ London, England, 1988.



. . . .

Figure 3. Spike detection by convolution of *X* and *H* with different signal levels, *X* is a time-reversal of *H*, *X* = *rev*(*H*). *H* has been produced by a random noise generator, it is 240 samples long. The level limitation stands in curly brackets { }. Case **a)** shows *unipolar* convolution, case **b)** shows *bipolar* convolution. Case **c)** shows mixed, nerve like vectors, *X* is unipolar, while *H* is bipolar. Case **d)** shows the FFT of unipolar *H*. The high DC-Potential (see the arrow) produces in the FFT a maximum at zero Hertz, prohibiting any serial code detection.. Case **e)** is the FFT of a bipolar *H*. Generation with clipping_conv.sce, download [24]



. . . .

Figure 4. Scilab-plots of an unipolar network. If the key X matches the keyhole code, a single spike appears at the output. From top to bottom: 1) automatically generated transfer function *H* with the reduced vectors *Tr* and *Wr*; 2) a best key *X* for the hole *H*; 3) interference of *X* and *H* as convolution; 4) resulting output of the network after the threshold function; 5) transfer function again; 6) Fourier-Transformation of the transfer function *H*. The mean of *H* has a significant DC-potential, so the FFT has its maximum at zero Hertz. Frequency selection is practically impossible using such unipolar nets with a dominant low-pass. Generation with transfuncnet.sce, download [24].

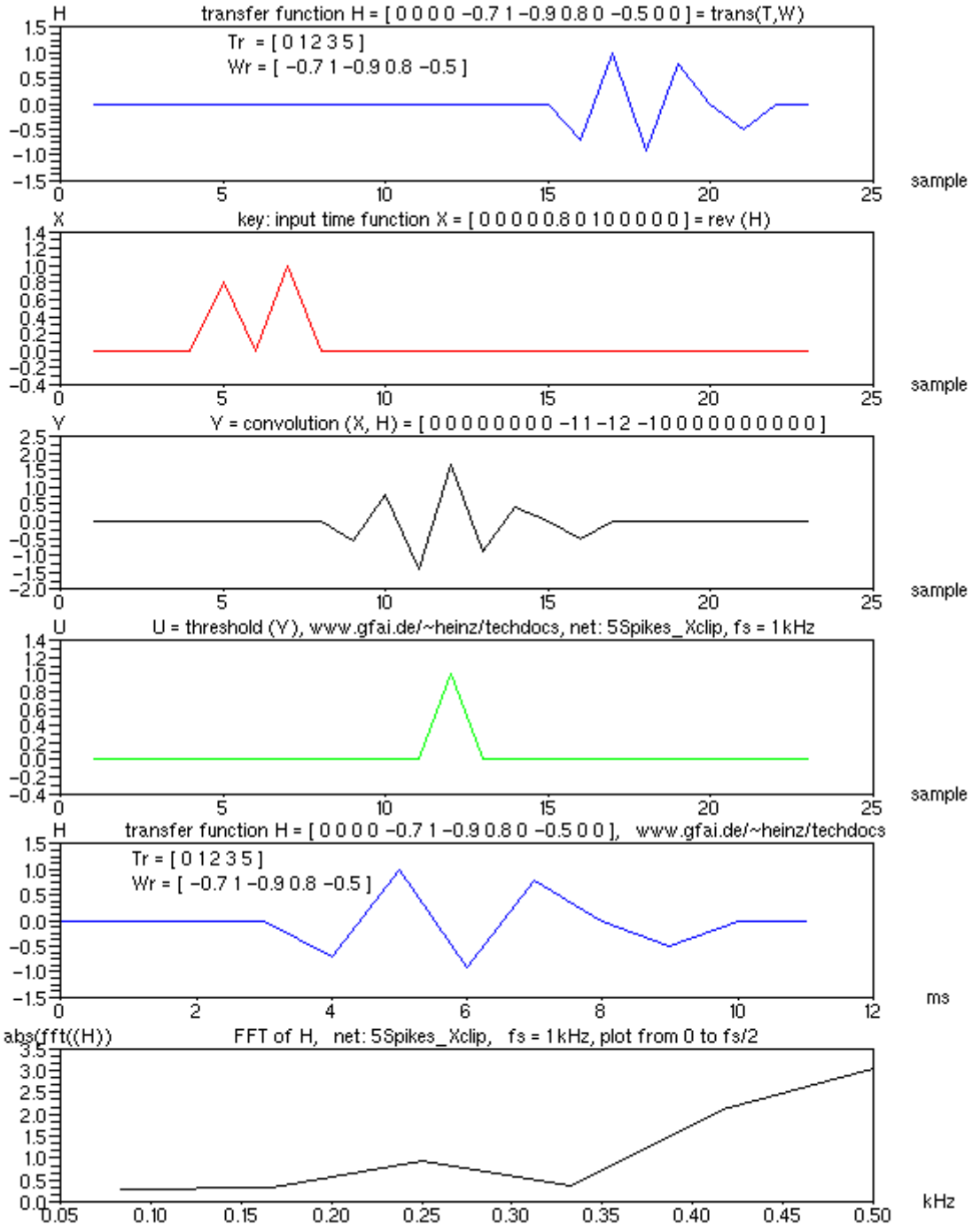


Figure 5. Scilab-plots of a mixed, nerve-like network with bipolar transfer function *H* and unipolar input *X*. From top to bottom: 1) automatically generated transfer function *H* with the reduced vectors *Tr* and *Wr*; 2) a best unipolar key *X* for the bipolar hole *H*; 3) interference of *X* and *H* as convolution; 4) resulting output of the network after the threshold function; 5) transfer function again; 6) Fourier-Transformation of the bipolar transfer function *H* showing a high-pass response. Generation with transfuncnet.sce, download [24].