BSA, a Fast and Accurate Spike Train Encoding Scheme

Benjamin Schrauwen
Electronics and Information Systems Department
Ghent University
Ghent, Belgium
Email: benjamin.schrauwen@elis.rug.ac.be

Jan Van Campenhout
Electronics and Information Systems Department
Ghent University
Ghent, Belgium
Email: jvc@elis.rug.ac.be

***Abstract*— In this paper we introduce a new algorithm for encoding analog information into spike trains, given that the reconstruction will take place using a FIR filter. An older technique called HSA is reviewed and an optimal threshold value is found. A new technique called BSA is introduced. These methods are then compared experimentally.**

I. Introduction

Spiking neural models already have a long history describing biological neurons (see [1] for an overview). Recently there has been an increasing interest in these neuron models, not for modeling real neurons, but as problem-solving networks ([2], [3]). This has become possible only recently because of the increased calculation power needed to simulate these more accurate neuron models.

Spiking neural networks consist of spiking neurons connected to each other, and stereotypical events, called spikes, are communicated via these connections. Due to the stereotypical nature, these spikes can be fully characterised by their firing time; the exact waveform of the spike is of no interest. Several neuron models are possible, though we will not overview them here ([1], [4]). A much used model is the leaky integrate-and-fire model (LIF). A neuron is modeled as a leaky integrator which fires when the accumulator crosses a given threshold. When the neuron fires, the integrator is reset.

Neuroscientists have a vast set of mathematical tools for analysing biological neurons and the spikes that they emit. But all these techniques are analysing in nature: they all hypothesize that the neurons that have to be investigated are given (by nature), and that we have to figure out how they work. When using spiking neural models as an engineering tool, we want to devise our own networks. We want them to solve some problem in an efficient way. Therefore we must be able to synthesise a neural network, not only analyse it. In order to construct a spiking neural network we generally need three things: a suitable neuron model, a way to train the network (both its topology and its parameters), and a method for communicating information to and from the network. The latter will be addressed in this paper.

Almost all real-world signals are analog in nature. When we want to use a spiking neural network to process those analog values, we need to convert them to spikes and vice versa. We thus want to be able to convert analog signals to spike trains and back to analog signals, and in such a way that the error introduced by encoding and decoding is minimal.

II. Reading the Spiking Language

Converting spike trains into analog values has already been studied thoroughly by theoretical neuroscience ([1], [5], [6]). There a two main (contrasting) classes of decoding methods and many intermediates ([7]).

A. Average rate

This idea is as old as neuroscience itself. Average rate coding assumes that all information carried by a spike train is encoded in the instantaneous average firing rate. This is classically measured by counting spikes in a certain time window (bin), usually lasting a fraction of a second or longer. This is classical rate coding. But if we shorten the size of the bins so that every bin has maximally one spike in it, then there is not much averaging going on. This is where average rates go over into correlation coding. This indicates that the boundary between rate coding and correlation coding is rather vague ([3]).

B. Correlation coding

Correlation coding assumes that the the exact pattern of spikes, both at the single cell level as well as between multiple cells, encode the information. Exact spike timing, which has been used often and has resulted in a multitude of theoretical proofs ([3], [8]), also falls in this category.

The encoding and decoding of analog values when using the exact spike timing scheme is trivial. The exact time a spike is fires encodes the analog value[1].

C. Stimulus estimation

Stimulus estimation is a often used technique in neuroscience and is located somewhere in between average rate coding and correlation coding. The idea is that the stimulus of a biological neuron can be estimated from the spike train by linearly filtering it. The estimate stimulus s_{est} can be written as

$$s_{est} = (h \star x)(t) = \int_{-\infty}^{+\infty} x(t-\tau)h(\tau)d\tau = \sum_{k=1}^{N} h(t-t_k) \quad (1)$$

with $x(t)$ the spike train of the neuron given by $x(t) = \sum_{k=1}^{N} \delta(t-t_k)$ with t_k denoting the set of firing times of the neuron, and $h(t)$ the impulse response of a linear filter.

[1]Some extra calculations may be needed, like subtracting a "background oscillation" ([3])

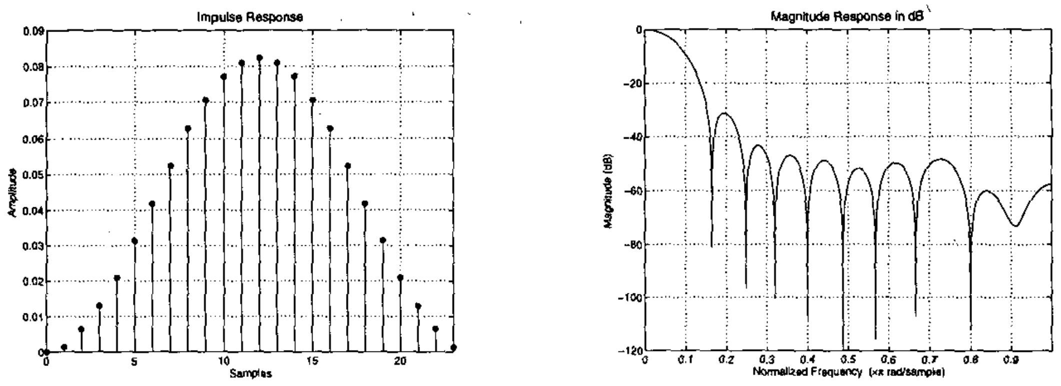


Fig. 1. Impulse response and frequency spectrum of the reconstruction filter

The filter $h(t)$ can be found by minimizing the mean square error between the stimulus and the estimated stimulus ([7]). Usually band-limited white noise stimuli are used to find the filter because this simplifies the mathematical analysis. We will not go much deeper into this subject, but using this technique, many interesting results in neuroscience have been achieved ([9]).

We don't want to analyse biological neurons. Our main goal is to encode and decode signals to spike trains. Therefore we don't want to find a filter modeling the neuron, but use a filter we constructed to convert spike trains into analog values using the aforementioned technique. The conversion from spike trains to analog values using linear filtering will be called decoding and is used in the rest of this paper.

For all our results we used a filter that was presented in [10]. It was found using a Genetic Algorithm (GA). The GA had to optimize the error between a set of sine waves and their processed versions[2]. This filter was pretty noisy and was cleaned up and normalised by the authors. In figure 1 we show the impulse response and the frequency spectrum of the filter. It is a 24-tap low-pass filter with quantized coëfficients (10 bit). Because of the quantisation noise we will never get a SNR better than 60 dB.

III. Speaking the spiking language

The second problem is converting analog values to spike trains. We will call this encoding. This has been much less researched by neuroscience because it was not needed for doing neuron analysis. We start with an overview of two existing techniques: HSA and modified HSA. The modified HSA will be fine-tuned. And finally we introduce a new algorithm called BSA for converting analog waveforms to spike trains. All technique will be tested using a set of test signals.

A. HSA

HSA (Hough Spiker Algorithm) was introduced in [11]. The basic idea behind this algorithm is to try to do a reverse convolution of the stimulus.

```
for i = 1 to size(input)
  count = 0;
  for j = 1 to size(filter)
    if i+j-1 <= size(input)
      if input(i+j-1) >= filter(j)
        count += 1;
      end if;
    end if;
  end for;

  if count == size(filter)
    output(i)=1;

    for j = 1 to size(filter)
      if i+j-1 <= size(input)
        input(i+j-1) -= filter(j);
      end if;
    end for;
  else
    output(i)=0;
  end if;
end for;
```

Fig. 2. Pseudo-code for HSA

When we convert spike trains to analog values we use a linear filter. All the algorithms in this paper require that the decoding filter has a finite impulse response (FIR filter)[3]. When filtering the spike trains with a discrete FIR filter, equation 1 becomes

$$o(t) = (x \star h)(t) = \sum_{k=0}^{M} x(t-k)h(k)$$

with M the number of filter taps. The encoding algorithm tries to reverse this. A pseudo-code implementation of this algorithm can be seen in figure 2.

At every instant of time τ, compare the shifted filter $h(t+\tau)$ with the matching values of $s(t)$. If the filter is everywhere smaller or equal, subtract $h(t+\tau)$ from $s(t)$ and emit a spike. Advance an infinitesimal amount of time and keep on repeating till the end of $s(t)$. The idea is that if the impulse response of

[2]Sine waves were first converted to spike trains using HSA, and then again converted to an analog value using linear filtering.

[3]Finite Impulse Response filter. The denominator of the transfer function of the filter is 1. These filters are non-recursive and have a finite impulse response.

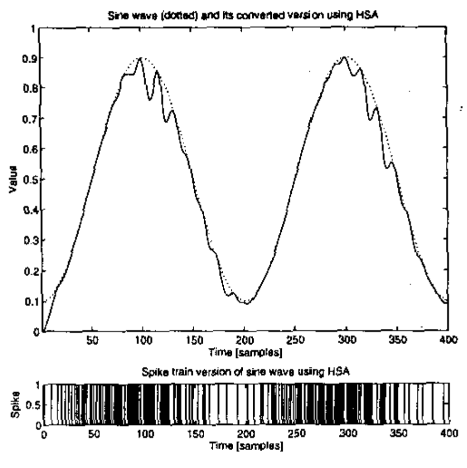


Fig. 3. Sine wave converted using HSA. In the upper plot we see the original sine wave in dotted line, and the converted version in solid line. The lower plot visualises the spike train.

the filter is smaller or equal than the input, then there has to be a spike in order to produce this.

This algorithm has no external parameters and only needs the reconstruction filter to convert analog values to spike trains. One of the main disadvantages is that $h(t)$ can only have positive values. If a filter has an impulse response with negative values, the algorithm will fail. But all FIR filters that have a steep filtering characteristic, have negative tap values. And because the filter bandwidth and filter steepness defines the bandwidth of the complete system, we are interested in filters that are as wide and steep as possible.

In figure 3 we see the HSA applied to a sine wave. Because we use a normalised filter with no negative values, the range a function can have is $[0,1]$. All the test signals we use are therefore transposed and scaled from $[-1,1]$ to $[0,1]$. We see that the converted signal is biased: it always stays below the original waveform. This is due to the explicit condition that the signal has to be greater than the filter before the algorithm will fire. The error tends to get bigger with higher values and also persists trough time.

Another (unpublished) version of the HSA (modified HSA) has been implemented in the operating system of the CAM-Brain Machine ([12], [13]) by Genobyte Inc. At our lab we have such a computer, and have access to the source code. Pseudo-code for the variant of HSA that is used there can be seen in figure 4. The impulse response of the filter is also compared to the input signal, but now the error between the signal and the impulse response is accumulated if and only if the signal is smaller or equal than the impulse response. A threshold needs to be provided with which the error is compared. If the error is smaller or equal than the threshold, then a spike is emitted and $h(t+\tau)$ is subtracted from $s(t)$. This is repeated until the end of the input signal. There was no way, other than trial-and-error, for knowing what the threshold needed to be.

```
for i = 1 to size(input)
  error = 0;
  for j = 1 to size(filter)
    if i+j-1 <= size(input)
      if input(i+j-1) < filter(j)
        error += filter(j) - input(i+j-1);
      end;
    end if;
  end for;

  if error <= threshold
    output(i)=1;

    for j = 1 to size(filter)
      if i+j-1 <= size(input)
        input(i+j-1) -= filter(j);
      end if;
    end for;
  else
    output(i)=0;
  end if;
end for;
```

Fig. 4. Pseudo-code for modified HSA

We first start by finding the optimal threshold for this modified HSA. This is delicate matter.

It is possible to search for an optimal HSA threshold every time a signal needs to be converted. But the computational overhead of this search is not acceptable during real-time conversion. We are much more interested in a fixed threshold that gives almost optimal behavior on a multitude of signals. Due to the fact that the optimal threshold of different signals do not differ very much, it is possible to use a fixed threshold without introducing large errors.

We need to compose a set of test signal that must be representable for all the signals we want to process. Because HSA is strongly non-linear, we cannot use linearity properties. Therefore we will need to provide signals with different amplitude, frequency and offset. This set of test signals then defines an optimization problem: find a threshold so that an error metric defined on the test set is minimal. This search space is not at all smooth in nature. Classical optimization techniques (i.e. steepest descent) can therefore not be used. We will have to fall back to partial enumeration of the threshold and find the optimum in this set.

In figure 5 we see a plot of the signal-to-noise ratio (SNR)[4] versus the threshold, and this averaged over a set of test signals[5]. The step size of the threshold was 0.0005. The dashed line indicated the maximum of the function. The optimal threshold for this filter is 0.0685. The maximum SNR is 25.2 dB.

When we use a certain filter, we can use the fixed threshold and get almost optimal results over a vast range of signals. But if we change the filter, the optimal threshold also changes. This is because different filters can have very different optimal thresholds. If we for example use a filter with a triangular

[4]Here the "signal" is the incoming signal, and the "noise" is the difference between the incoming signal and reconstructed signal. Reconstruction is done by linear filtering with the aforementioned filter.

[5]We used a set of 88 test signals consisting of 78 sine waves with different amplitudes, frequencies and offsets, and 10 constant valued signals with different values.

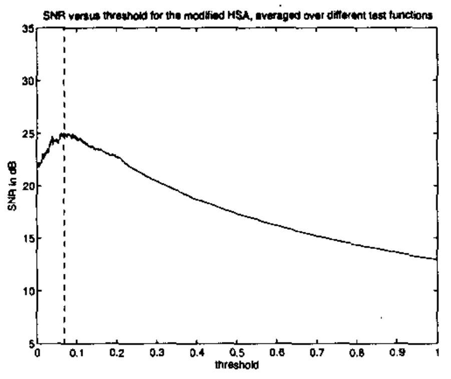


Fig. 5. Optimal threshold for the modified HSA

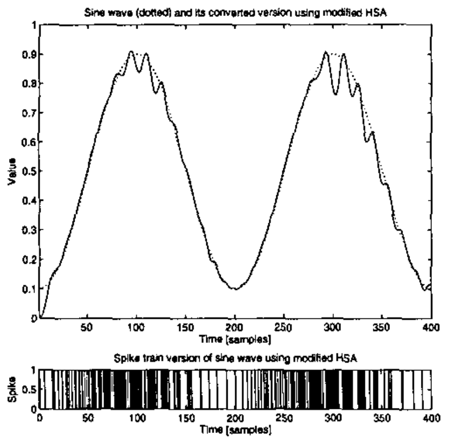


Fig. 6. Sine wave converted using modified HSA. In the upper plot we see the original sine wave in dotted line, and the converted version in solid line. The lower plot visualises the spike train.

impulse response, we get an optimal threshold of 0.0796.

In figure 6 we see the same converted sine wave, but now with the modified HSA. We use the fixed threshold we found before. The converted signal now exceeds the original. This is because we use a threshold. The remaining error is still pretty large.

B. BSA

We now introduce a new algorithm for converting analog values to spike trains called Bens Spiker Algorithm (BSA) ([14]). The pseudo-code can be seen in figure 7. This algorithm also assumes the use of a FIR reconstruction filter. It works as follows. Every instant in time τ calculate two error metrics: one is $\sum_{k=0}^{M} abs(s(k+\tau)-h(k))$, the other is $\sum_{k=0}^{M} abs(s(k+\tau))$. If the first error is smaller than the second minus a threshold, then fire and subtract the filter from the input, else do nothing.

BSA also needs a threshold. We use the same procedure as before to find it. The result is shown in figure 8. The optimal threshold for the given filter is 0.9550. The attained SNR is

```
for i = 1 to size(input)
  error1 = 0;
  error2 = 0;
  for j = 1 to size(filter)
    if i+j-1 <= size(input)
      error1 += abs(input(i+j-1) - filter(j));
      error2 += abs(input(i+j-1));
    end if;
  end for;

  if error1 <= (error2 - threshold)
    output(i)=1;

    for j = 1 to size(filter)
      if i+j-1 <= size(input)
        input(i+j-1) -= filter(j);
      end if;
    end for;
  else
    output(i)=0;
  end if;
end for;
```

Fig. 7. Pseudo-code for BSA

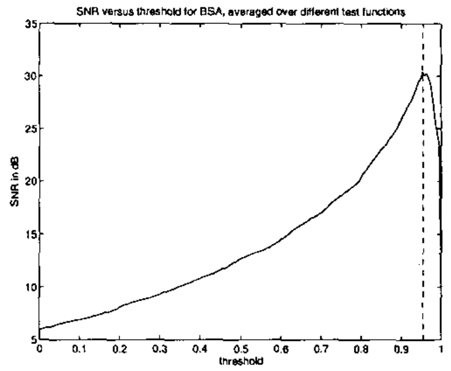


Fig. 8. Optimal threshold for BSA

now 30.4 dB. Notice that the function is much smoother than in the case of HSA. This means that the optimal threshold is much more constant over different test signals, and that the SNR does not change drastically when we alter the threshold a bit.

In the case of BSA, the threshold also has a sensitivity to the filter. Every filter has his own optimal threshold. But now we notice that the sensitivity is much smaller. When we for example use the triangular filter, we get a optimal threshold of 0.9952 which is almost identical to that found earlier.

Figure 9 shows the conversion of the test sine wave. We notice a remarkable error at the first peak. This is a transient from switching on the conversion. The error disappears at later peaks. The residual error is now small.

IV. Results

We will present two types of graphs that illustrate the characteristics of HSA and BSA conversions. First we plot the SNR (in dB) versus the amplitude of the sine wave we use as input. Period of the sine wave is 300 samples (normalised frequency is 1/300) and its mean value is 0.5 (call this the bipolar case) or its minimum value is 0 (this is the unipolar

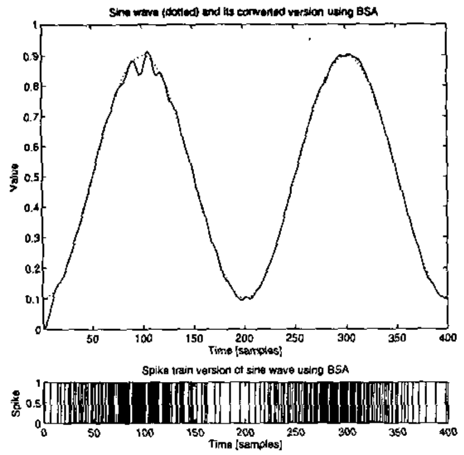


Fig. 9. Sine wave converted using BSA. In the upper plot we see the original sine wave in dotted line, and the converted version in solid line. The lower plot visualises the spike train.

case). In figure 10 we see this plot for the original HSA, the modified HSA with optimal threshold per signal (we will name this optimal HSA), and the modified HSA with fixed threshold (will be named fixed HSA). Figure 11 shows the same for BSA with optimal threshold for every signal (call it optimal BSA) and BSA with fixed threshold (name it fixed BSA).

In case of HSA we see, like expected, that the curve for optimal HSA is best, but fixed HSA perform almost as good. The original HSA performs the worst. HSA performs better in the unipolar case. Define the dynamic range (DR) as the maximal reachable SNR. For optimal, fixed and original HSA this DR is respectively 34.5, 30.2 and 29.2 dB in the unipolar case and 26.4, 23 and 19 dB in the bipolar case. But the bipolar case is for the user the most interesting, unfortunately HSA performs here worst.

Notice that the curves tend to decline with high amplitudes. This is called overloading. The input signal is to high to be correctly converted. The overloading level lies around -1 dB. Note that overloading in the bipolar case occurs at both boundaries of the signal amplitude interval (around 0 and 1). In the unipolar case increased error at high amplitudes occurs near the upper boundary (near 1). At low signal amplitudes we get an increased error because signals are again encoded near a boundary of the interval (now near 0). This can also be seen on the graphs. This again is a point for the use of bipolar coding.

The unipolar plots for BSA show very similar behavior, except that the overloading is reduced. The biggest change is with the bipolar case. Here the optimal and fixed BSA have a DR of 35.5, 33.1 dB respectively. This is about 10 dB better than in case of HSA. On the other hand the effect of overloading increases a bit. Note that due to the increased DR, BSA can encode smaller input signals.

Next we plot the SNR (in dB) versus the normalised frequency[6] of the input sine wave. The amplitude is 0.25 and the mean value is 0.5 (bipolar). HSA has a rippled frequency curve while BSA is almost flat. Note that the encoding/decoding system has a well defined bandwidth. Signals with a normalised frequency higher than 0.055 can not be represented. BSA is again almost 10 dB better than HSA.

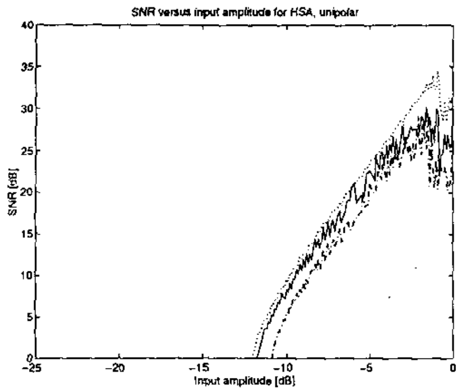
V. Conclusion

We showed that the modified HSA with well chosen threshold performs better than the original HSA (about 5dB). We introduced a new technique called BSA. This technique perform again better than the modified HSA, mainly in the bipolar case. The main advantages of BSA are that frequency and amplitude characteristics are smoother, that there is an improvement of 10 dB over modified HSA (bipolar case) and even 15 dB over the original HSA. In the unipolar case BSA performs as good as modified HSA but has less effect of overloading. BSA is also less sensitive to changes in the filter and the threshold (due to smoother threshold optimization curve).

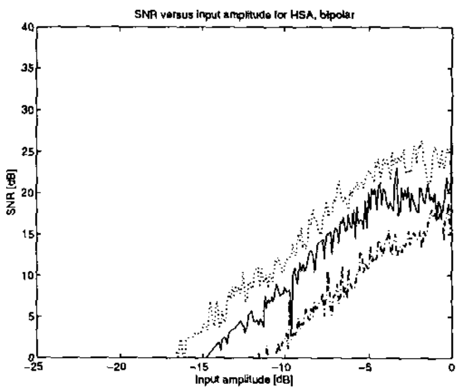
References

[1] P. Dayan and L. F. Abbott, *Theoretical Neuroscience*. Cambridge, MA: MIT Press, 2001.
[2] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.
[3] W. Maass and C. M. Bishop, *Pulsed Neural Networks*. Cambridge, MA: Bradford Books/MIT Press, 2001.
[4] W. Gerstner and W. Kistler, *Spiking Neuron Models*. Cambridge, England: Cambridge University Press, 2002.
[5] F. Gabbiani and W. Metzner, "Encoding and processing of sensory information in neural spike trains," *The journal of experimental biology*, vol. 202, pp. 1267–1279, april 1999.
[6] F. Gabbiani, "Coding of time-varying signals in spike trains of linear and half-wave rectifying neurons," *Network: Computation in Neural Systems*, vol. 7, pp. 61–85, 1996.
[7] C. Koch, "Linear stimulus encoding and decoding," november 1999, lecture notes. [Online]. Available: http://www.klab.caltech.edu/cns185/lectures/encoding_decoding_99.ps
[8] S. M. Bohte, J. N. Kok, and H. A. L. Poutr'e, "SpikeProp: Backpropagation for networks of spiking neurons," in *Proceedings of the European Symposium on Artificial Neural Networks*. Bruges: D-Facto publications, april 2000, pp. 419–424.
[9] F. Rieke, D. Warland, R. van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*. Cambridge, MA: Bradford Books/MIT Press, 1997.
[10] H. de Garis, N. Nawa, M. Hough, and M. Korkin, "Evolving an optimal de/convolution function for the neural net," in *Proceedings of International Conference on Neural Networks. IEEE*, vol. 1, july 1999, pp. 875–882.
[11] M. Hough, "SPIKER: Analog waveform to digital spiketrain conversion in ATR's artificial brain (cam-brain) project," in *International Conference on Robotics and Artificial Life*, january 1999.
[12] M. Korkin, H. de Garis, F. Gers, and H. Hemmi, ""CBM (CAM-brain machine)": A hardware tool which evolves a neural net module in a fraction of a second and runs a million neuron artificial brain in real time," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Stanford University, CA, USA: Morgan Kaufmann, 13-16 July 1997, pp. 498–503.
[13] M. Korkin, G. Fehr, and G. Jeffery, "Evolving hardware on a large scale," in *Proceedings. The Second NASA/DoD Workshop on Evolvable Hardware*. IEEE Comput. Soc., July 2000, Conference-Paper, pp. 173–81.
[14] B. Schrauwen, "Pulstreincodering en training met meerdere datasets op de CBM," Master's thesis, Universiteit Gent (ELIS), Gent, 2002.

[6]Frequency divided by the sampling frequency.

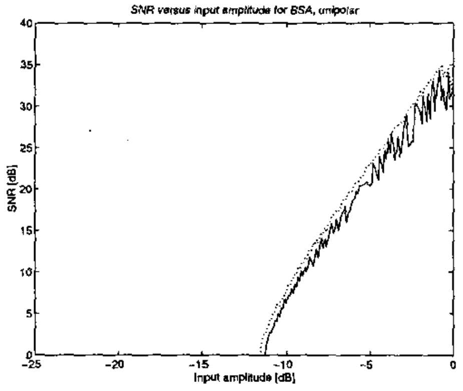


(a) The unipolar case (sine waves with minimum 0)

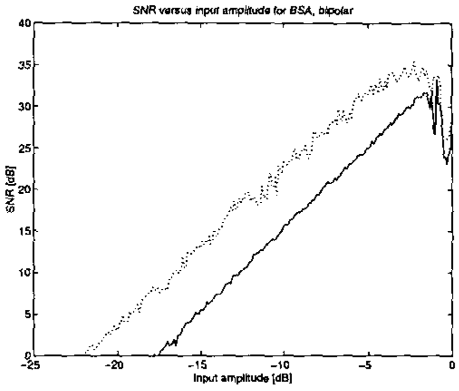


(b) The bipolar case (sine waves with mean 0.5)

Fig. 10. Plots of the SNR versus the amplitude for HSA. In figure 10(a) we see the plots for sine waves with minimum always 0 (unipolar), and in figure 10(b) plots for sine waves with mean 0.5 (bipolar). The sine waves have a period of 300 samples. The top plot (dotted) in both figures is HSA with optimal threshold calculated for every signal, the middle plot (solid) is HSA with fixed threshold, and the bottom plot (dash-dot) is the original HSA.

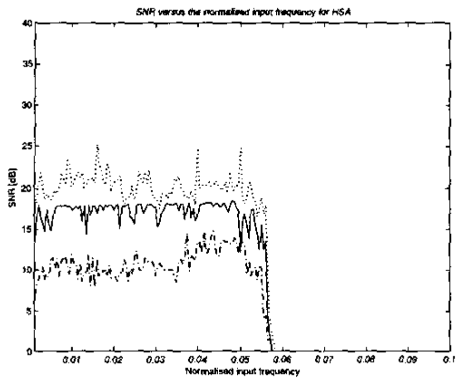


(a) The unipolar case (sine waves with minimum 0)

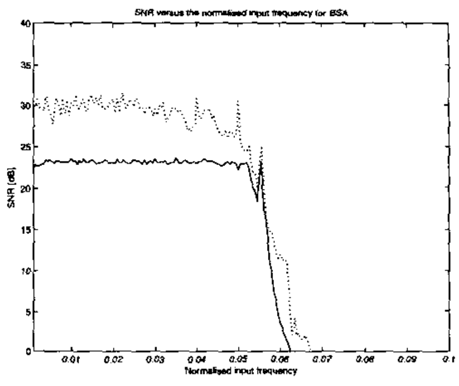


(b) The bipolar case (sine waves with mean 0.5)

Fig. 11. Plots of the SNR versus the amplitude for BSA. Here the same signals where used as for HSA. The top plot (dotted) in both figures is BSA with optimal threshold calculated for every signal, and the bottom plot (solid) is BSA with fixed threshold.



(a) HSA



(b) BSA

Fig. 12. These plots show the SNR versus the frequency of the input sine wave. The amplitude is 0.25 and the mean value is 0.5. In figure 12(a) we see the plot for HSA where the top plot (dotted) denotes optimal threshold calculated for every signal, the middle plot (solid) fixed threshold, and the bottom plot (dash-dot) the original HSA. The plots for BSA can be seen in figure 12(b) where the top plot (dotted) denotes optimal threshold calculated for every signal, and the bottom plot (solid) BSA with fixed threshold.