NTNU
Innovation and Creativity

Satellite Cluster Consepts

A system evaluation with emphasis on deinterleaving and emitter recognition

Bent Einar Stenersen Bildøy

Master of Science in Electronics
Submission date: June 2006
Supervisor: Jens F. Hjelmstad, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

The Euclid project is described as a passive ESM system implemented in a satellite cluster to achieve target detection and geolocation. This master thesis makes a detailed analysis of the emitter classification and deinterleaving capabilities of the cluster for actual scenarios.

Assignment given: 29. January 2006
Supervisor: Jens F. Hjelmstad, IET

Summary

In a dense and complex emitter environment, a high pulse arrival rate and a large number of interleaved radar pulse sequences is expected, from both agile and stable emitters. This thesis evaluates the combination of interval-only algorithms with different monopulse parameters, in comparison to a neural network to do accurate emitter classification.
This thesis has evaluated a selection of TOA deinterleaving algorithms with the intent to clearly discriminate between pulses emitted from agile emitters.
The first section presents the different techniques, with emphasis on pinpointing the different algorithmic structures.
The second section presents a neural network combinational recognition system, with a main focus on the fuzzy ARTMAP neural network, where also some practical implementations has been presented.
The final section gives a partial system evaluation based on some statistical means, seeking to get an estimate on the information flow from the ESM receiver as a function of both the density and the expected parametric values, i.e. PW since this is proportional to the amount of processed pulses.

Table 1: List of abbreviations

Abbreviations	Definition
FT	Fourier transform
PW	Pulse width
PRI	Pulse repetition interval
PRF	Pulse repetition frequency
DOA	Direction of arrival
TOA	Time of arrival
PST	Pulse sorting transform
SEI	Specific emitter identification
VLSI	Very-large-scale integration
FPGA	Field Programmable Gate Array
MOP	Modulation on pulse
MLP	Multilayer perceptron
MTI	Moving target indicator
KF	Kalman filter
PE	Processing element
ESM	Electronic support measure
PDW	Pulse description word
EDW	Emitter description word
AD/DA	Analog-to-digital/ digital-to analog
FFT	Fast Fourier Transform
DL	Downlink
PDF	Probability density function
TDOA	Time difference of arrival
SW	Software
IC	Integrated circuits
HW	Hardware

Introduction

This thesis is based on the Euclid project, described as a passive ESM system implemented in a satellite cluster, to achieve target detection and geolocation. The proposed satellite cluster consists of three satellites operating in a delta configuration, with a relative short inter-satellite geometry; a 2 km distance along track, and a third satellite at a 20 km distance. The configuration doesn't impose great demands on satellite control, but the relative position of the satellites needs to be established with high precision. The satellites operate in a nominally circular orbit at 600 km altitude, with an inclination of 90 degrees. Each satellite carries a microwave payload with a $(2*2)m^2$ flat lightweight antenna with a compact RF-circuit and data processing system, enabling bandwidth coverage from 1 GHz to 12 GHz.

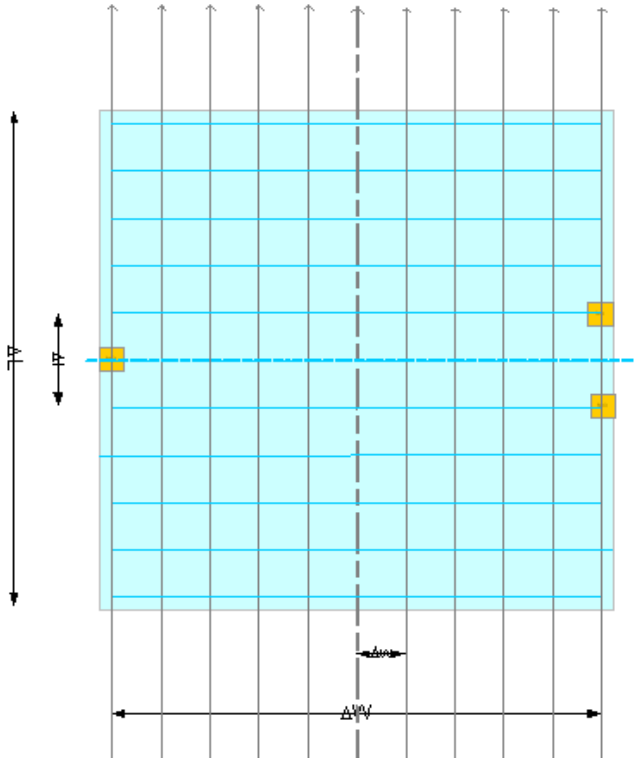


Figure 1: Proposed satellite configuration

Emitters

A brief summary from [1] of the targeted emitters:

- *Surveillance radars*. Surveillance radars detect and track military and civil aviation. Low frequencies are used, typically in the L- band, with large aperture to function optimally in all kinds of weather and over great distances. This means that they also use a low PRF.
- *Air defense radars*. Air defense radars that operate in 2 dimensions will represent most of the observed systems. They operate typically in the C-band.
- *Marine radar*. Typical marine radars operate in the X-band and/or the S-band. For large vessels it is a requirement to use both, as smaller boats generally uses the first because of the smaller wavelength which implies a smaller and lighter antenna.
- *Coastal surveillance radar.* Operates in the same band as the marine radars. Surveillance of coastal areas will generally use S-band, while surveillance of harbor areas operate in X-band for higher accuracy.
- *Mobile ground based surveillance radars.* Operates in X-band.

The mentioned radar bands are presented in Table 2.

Table 2: Radar band denomination

Band	Frequency
L	1 - 2 GHz
S	2 - 4 GHz
C	4 - 8 GHz
X	8 - 12 GHz

Pulsed radar can be characterized by a pattern of pulse intervals that repeats itself from a given start time, called the phase. The time between the pulses is referred to as the PRI, which is the inverse of the PRF. The simplest of emitters, called stable emitters, utilize only one PRI, and its range varies from a few hundred milliseconds down to several microseconds, where the latter is often referred to as High PRF.
A pulsed radar can be range ambiguous if a second pulse is transmitted prior to the return of the first pulse. In general, the radar PRF is chosen such that the unambiguous range is large enough to meet the radar's operational requirements. Therefore, long-range search (surveillance) radars would require relatively low PRFs.
In addition, to avoid jamming, lowering the probability of intercept etc, many emitters apply some kind of agility to the pulses. An often used scheme is to vary the PRI from one pulse-group to another, or even on a pulse-to –pulse basis. This is referred to as a staggered or a switched dwell emitter, where the two generally are separated with the assumption that a staggered emitter changes its PRI more often than a switched dwell emitter. Jittered pulses are also frequently used, where the exact pulse PRI will vary randomly around an exact value. This is useful in compensating for blind speeds in MTI radars.
On the receiver end, if the pulses received are flagged with the arrival time, these time differences will be equivalent to the emitters PRI, depending on the accuracy of the

receiver. This makes the TOA of a pulse such a crucial parameter in recognizing the emitters.
PRF is commonly divided into the following categories:

- Low PRF =0.25-4 kHz
- Medium PRF = 8- 40 kHz
- High PRF =50-300 kHz

Algorithmic description

The initial proposed solution from [2] suggested a neural network as part of the ESM system chain, which was chosen because of the network's ability to classify and recognize patterns. These patterns represent parameters measured from intercepted pulses in the ESM system; specifically the PRI, the RF and the PW. The original solution assumed that TOA deinterleaving could be reliably achieved only for constant PRIs. In so, it would be advantageous to get a thorough understanding of the capabilities and limitations of the different TOA deinterleaving algorithms as this is the basis for reliable deduction of the individual PRFs of an interleaved sequence. In addition, the neural network as a classifier is based on different principals than standard classification techniques, and a basic knowledge of the behavior of the proposed network for the given parameters, and its possible implementations, would make the evaluation of the total system more accurate.

Pulse sorting

The ultimate goal of the signal processing is to classify and recognize emitters based on the unique characteristics of their signals. What characteristics that are used are dependent on the ESM system. Some parameters can be measured based on one pulse, and are called monopulse parameters. These are listed in Table 3. Other parameters, such as PRI, can be derived by analyzing groups of pulses.

Table 3: Some monopulse parameters

Monopulse parameters
Frequency (RF)
Amplitude
Direction of arrival (DOA)
Time of arrival (TOA)
Pulse width (PW)

Real-time signal recognition is a complicated task, mainly because of the parametric agility some emitters have, but also if operating in dense environments. This makes the classification difficult, and most (standard) ESM systems are designed to primarily recognize emitters with more or less stable characteristics. However, as the technology evolves, more and more emitters utilize agility in parameters as RF and PRI.

Each radar emits a sequence of pulses whose times of arrival are regularly spaced. The interception of several such signals leads to an interleaved sequence of times of arrivals. From this interleaved mixture, the individual pulse trains need to be sorted, i.e. deinterleaved. This is referred to as the TOA deinterleaving.

The proposed overall method is based on a neural network, which operates on a sequence of presorted PDWs. This means that the pulses are preprocessed prior to the presenting to the network. The proposed network parameters are RF and PW, defined as the ‘what’-parameters, and with the PRI of the pulse train as a third and additional parameter. Even though the network has previously performed with very satisfactorily results, the used parameters impose some drawbacks, and this will be discussed later. Some evaluation must also be done in connection to the PRI values, because they are assumed to stem from stable emitters. Omitting the agile emitters will degrade the performance of the system accordingly, as the state-of-the-art emitters will be of this sort. The PRI for each PDW is presented to the network, where the PRI are derived earlier in the chain. The following section will give a deeper understanding of the limits and possibilities of this stage, evaluating simple and more complex algorithms.

Techniques

As an alternative to do a complete sampling of the pulse trains, it is evaluated to represent the pulses by pulse parameters, which will lead to a significant reduction in size of information required to be stored after A/D conversion.
The type of pulse representation chosen is somewhat arbitrary, as the form has little or no meaning for the wanted shape of the graphical display. Pulse shapes could be square, Gaussian or a delta-function. The pulses described by the PDWs are values for PA, PW and TOA, but could also include, or use other parameters. The main goal is to get as good a representation as possible for the given algorithm.

TOA deinterleaving is traditionally used in most ESM-systems, but is a very processor intensive task. The idea is to get an indication of probable PRIs, using as few computations as possible.

It is crucial to use the algorithms that are robust in terms of detecting pulses in high-pulse densities, when most of the pulses will be corrupted. TOA algorithms are used to extract all radar patterns that can be reliably recognized, leaving a residue of pulses that can be analyzed using different techniques.
TOA measurement is done at the leading edge of the pulse, and represented with a digital word. Arithmetic computations are performed on a sample of these words by the algorithm.

Sequence search

The sequence search is the simplest of the TOA algorithms, and it works by forming trial pulse trains from an initial pulse pair, or a triplet. The initial pulses are thereby hypothesized to be the PRI of the pulse train by a given window or tolerance level.
The PRI is called a match provided there are a sufficient number of pulses at the PRI within the predefined tolerance level. The events are then removed, to simplify further processing. If no matches are found, a different PRI is selected.
A typical tolerance level assumes a jitter no more than 10 % of the pulse train's mean PRI. This means that the window is initially set at ± 0.3 times the current best estimate, which represents ± 3 Gaussian standard deviations from the mean.

The *Sequence Search* provides identification of sequences, greater accuracy and reliability than the differential histogram presented in the following, but is slower because of a greater number of computations. The simplicity favors the technique as a secondary method, as presented later. To avoid extracting multiples of the correct PRI, it is important to extract the smallest intervals first. Some precaution must also be made to the case of trains with PRIs that differ from integer values of the sampling interval.

Differential Histogram

The TOA of each pulse is represented with a delta function at the appropriate sampling interval, and the deinterleaving algorithm will analyze the sample, and attempt to extract the individual sequences. The sample interval *k* is defined as the TOA measurement resolution, and the sample length as *N* sampling intervals (SI). Each TOA is thus represented as an integral multiple of the sampling interval.
A stable PRI sequence can thus be represented as

(0.1) $$\alpha_i = \sum_{r=0}^{N} f_i(rk)$$

where the sequence has a PRI of m_i SI, a start time of q_i SI, and n_i pulses in the sample, and

(0.2) $$f_i(rk) = \begin{cases} 1 & r = am_i + q_i, 0 \le a \le \mathrm{int}\left(\frac{N-q_i}{m_i}\right) = n_i \\ 0 & otherwise \end{cases}$$

The total sample consists of a sequence of events E.
In short, each TOA is subtracted from every subsequent TOA, and a count accumulated at each TOA difference. The total sample consists of a sequence of events E.
The number of computations required for a sample of E elements is of order.

(0.3) $$\sum_{i=0}^{E} i \approx \frac{E^2}{2} \text{ where } E \gg 1 \text{ and } E \ll N$$

When several signals are present, counts will occur at multiples, sums, and differences in addition to the correct PRIs, which gives an ambiguous result.

The difference histogram can be viewed as an autocorrelation of the sample, as seen by applying a delay *d* SI;

(0.4) $$y(d) = \sum_{r=0}^{N} p(rk)\, p((r-d)k)$$

For each delay *d*; i.e. each PRI entry in the histogram, the sample *y(d)* contains a count equal to the number of solutions to the equation

(0.5) $$q_i + am_i = q_j + bm_j + d \text{ where } i, j = 1 \to x$$

The full count of events in the *i*th stable sequence occurs at the appropriate PRI and multiples of the PRI when

(0.6) $$d = cm_i, \;\; y(d) > n_i - c \;\text{ for }\; c = 1, 2, ...$$

Each stable PRI sequence is identified by the correct count at each multiple of the PRI, and this is utilized to derive the threshold for which a sequence is said to be present. A threshold must be defined to which a sequence is present if the count goes above the threshold. It must be chosen carefully to allow for missing pulses and interfering pulses. This solution makes the threshold critical; as the non-detected and false identifications are directly dependent on it.

The differential histogram simply counts the number of event pairs separated by a certain PRI. It doesn't identify the sequences, but as it is based on subtractions, the processing is fast.

Example 1: The following figures indicate the difficulty in trying to deduce the number of pulse trains present in an interleaved sequence.[1] From Figure 2, the periodicity of the pulse train emerges as a period of 5, with the harmonics. Once an additional sequence with period 8 (Figure 3), and another with period 12 (Figure 4) is introduced, the individual periods becomes hard to discern.
The total number of pulses is seen from 'diff_hist.m'. Only a fraction of the histogram is displayed.

1 ***The code in the simulations differs from the mathematical theory presented earlier in that the interleaved pulses aren't represented as a binary sequence, but as arrival times. This is justified by the use of unit SI. For a binary sequence the row values represent the differences in arrival times, and in so does the simulation omit the subtraction procedure.***

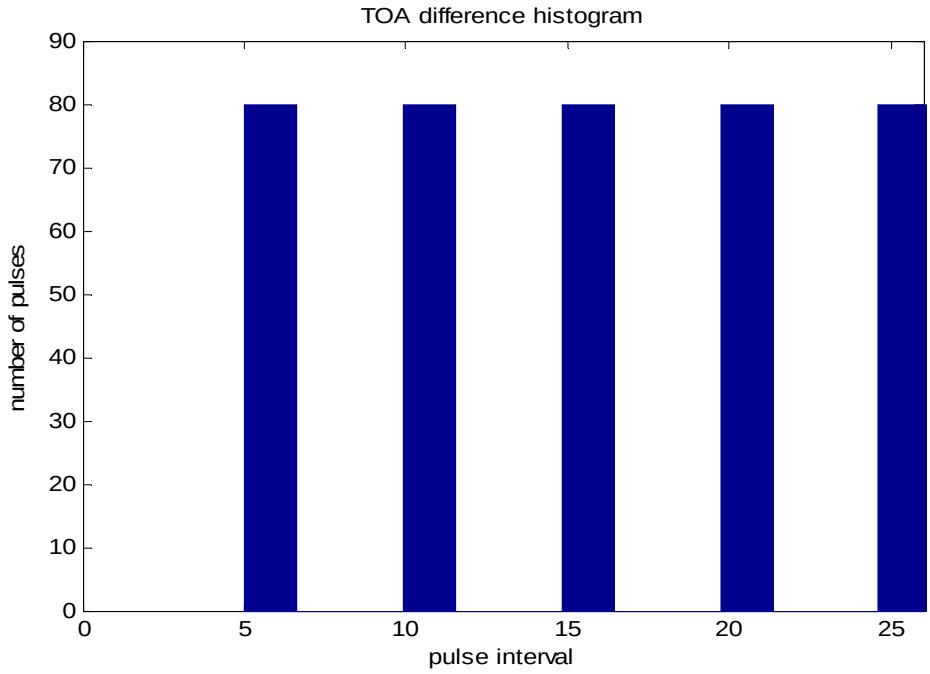


Figure 2

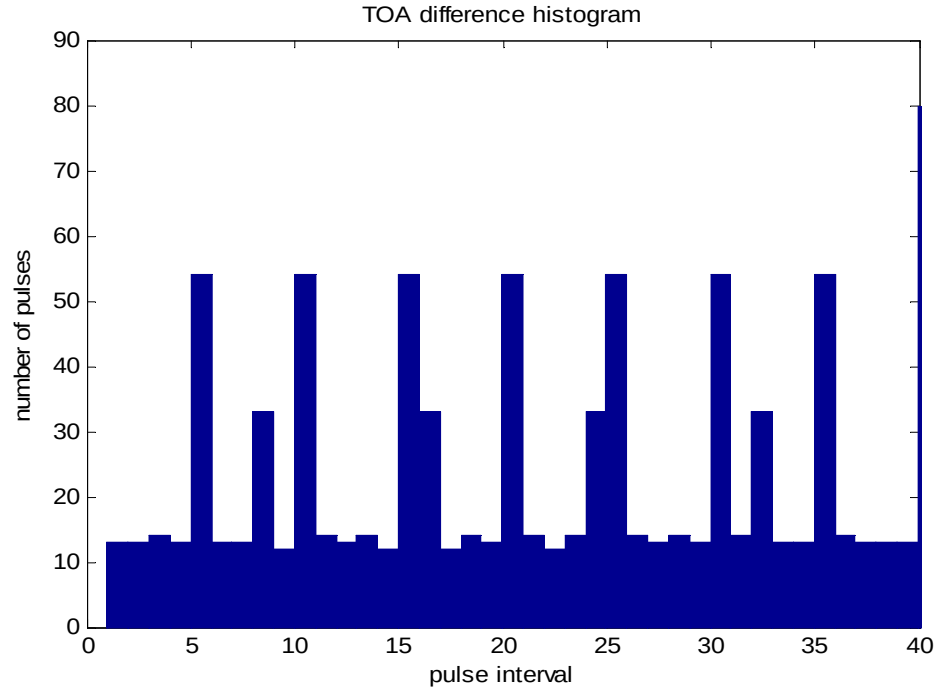


Figure 3

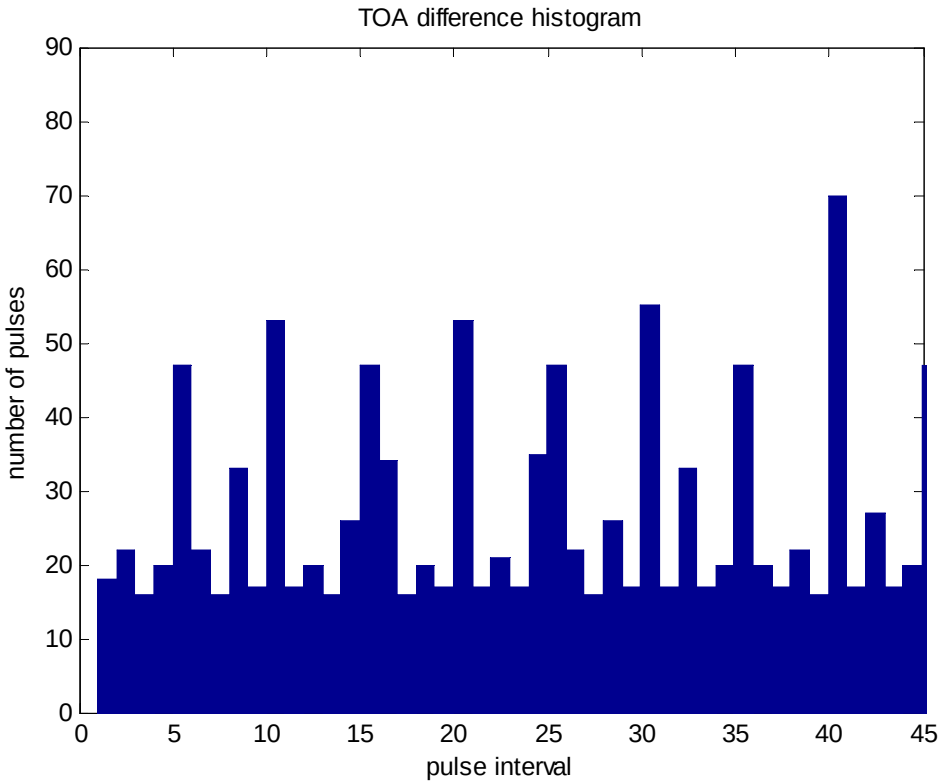

TOA difference histogram
number of pulses
pulse interval
0
10
20
30
40
50
60
70
80
90
0
5
10
15
20
25
30
35
40
45

Figure 4

Threshold

The optimal threshold function is crucial to every histogram technique. As the bins in the histogram correspond to the interval between the pulses, it follows that the larger the observed time interval between pulses, the smaller the number of appearances of that interval in a finite observation time, i.e. the threshold value is inversely proportional to the bin number.

Mathematically this can be expressed as

(0.7) $$p(\tau) = \frac{x * E}{\tau}$$

where τ is the bin number, E is the number of observed pulses, and x is a constant less than 1.
From [3] it is found that the optimal function of threshold should take the form

(0.8) $$Thr(\tau) = x * (E - c) * e^{\frac{-\tau}{kN}}$$

where E is the total number of pulses, N is the total number of bins in the histogram, and c is the difference level. Difference levels are presented in the following.

CDIF

The CDIF algorithm is more robust than the previously mentioned algorithms, and it combines histogram techniques with sequential search techniques.

As opposed to histogram techniques that forms TOA differences between all differences, the CDIF TOA histogram forms TOA differences only between adjacent events initially. This is called the first difference. First; the count at each interval and the double interval are compared to a threshold. If both counts are above threshold, the sequential search is initiated. If no sequence is identified, the TOA difference between each event and the next (but) one event is calculated, and the count is accumulated.
The requirement of having the second harmonic present, limits the searches to cases where sequences of three events occur, rather than only pairs.
The attempt is to remove the smallest PRI radars quickly.
Three pulses are chosen as the starting point to reject random pulse pairs and sub intervals of a staggered PRI, which provide a more accurate PRI.

Weight schemes are used to enhance detection of uninterrupted sequences.

1) The simple weight scheme. Counts events fitting in the sequence, and adds the number of these events separated by the correct interval.
2) A complex weighting scheme; for each unbroken sequence of events found in the sample, the reciprocal of the probability is added to the count. This requires monitoring of average pulse density and pulse width, but will give an enhancement proportional to the pulse density.

Because of the difficulty of detecting and identifying agile radars, the signals that can be reliably extracted should be removed first.
After extraction of stabile PRI sequences, sequences with identical PRIs are identified as staggered. The final signals to be evaluated are those with potentially jittered signals. This is the most difficult task, and in short it is solved by adjusting to a greater PRI tolerance.

Example 2: The graphs display the four first difference levels of an interleaved sequence with individual periods of 5 and 8.
The black dotted line represents the threshold from equation(0.7), and the blue whole line represents the threshold from equation(0.8).
There are some disadvantages in using the CDIF; the most important is the high number of difference levels required to correctly identify the sequences. As in all the histogramming techniques, the choice of threshold is crucial.
Another drawback with the CDIF is that when many pulses are missing, the multiples of the true PRI will be detected in the CDIF.

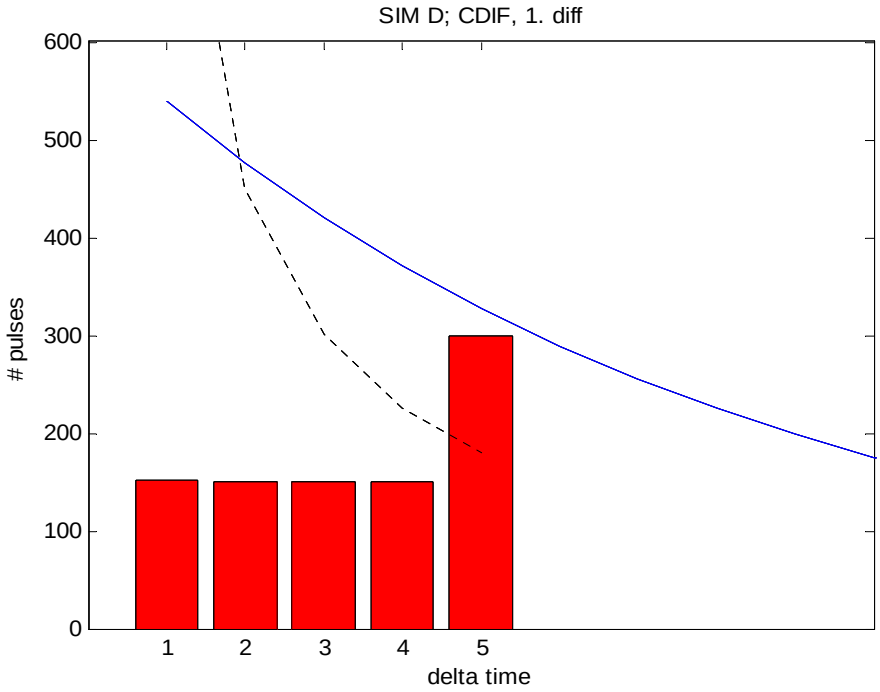


Figure 5

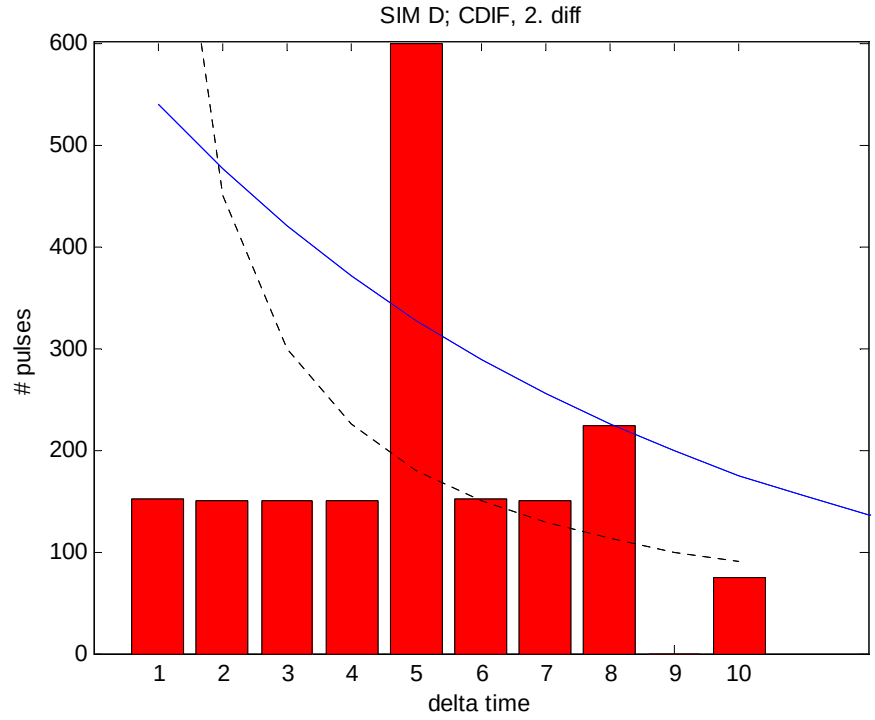


Figure 6

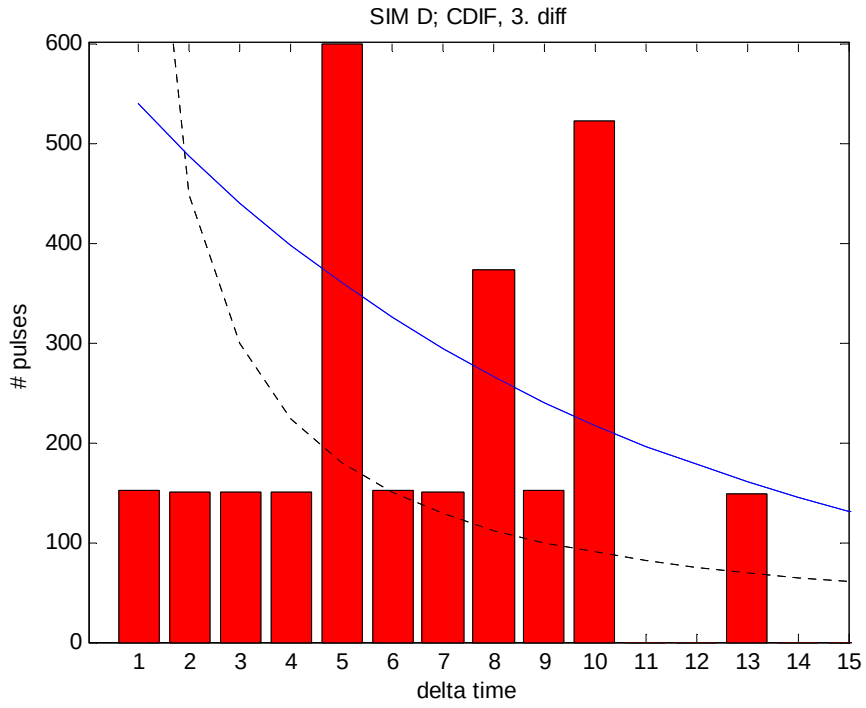


Figure 7

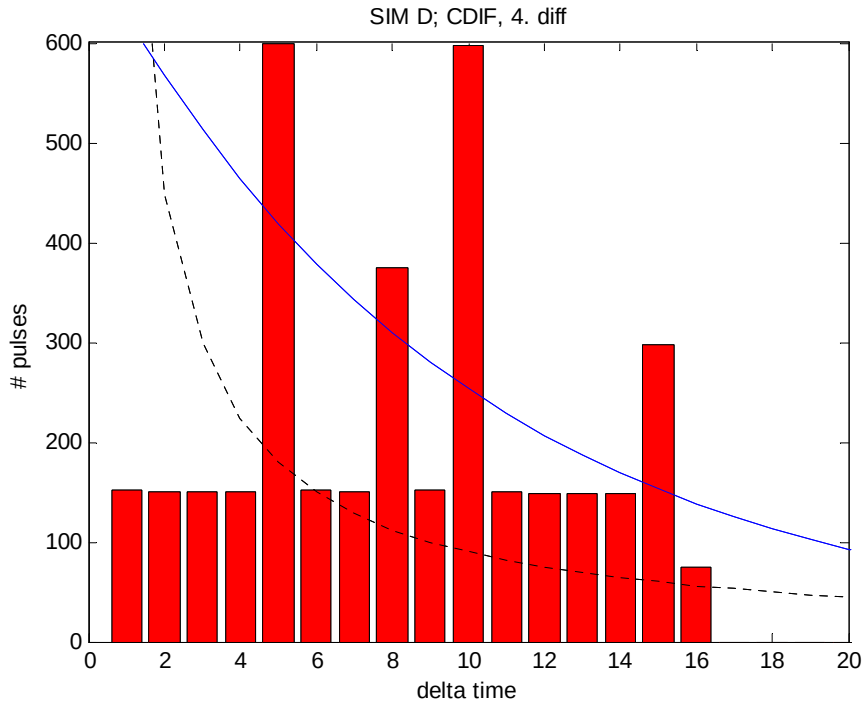


Figure 8

SDIF

This algorithm is a modified and improved version of the CDIF. Compared to the previously mentioned TOA difference algorithm, it is less sensitive to interference and missing pulses, and in so the SDIF has been successful in high pulse density radar environments, and for complex signal types such as frequency-agile and staggered PRI radar signals.

The SDIF algorithm, generally consists of two parts; the estimation of the PRI and the sequential search. To extract the true PRI, it is sufficient to compute the second difference and compare it to the threshold value. It is no need to also do a comparison of the second harmonic of the PRI to the threshold, which is the key limitation of the CDIF, in addition to the requirement of a high number of difference levels even in very simple cases. If one discards the required double PRI, one avoids being limited to cases where the sequences of three events occur, and in so; the accumulation in the difference histograms is no longer necessary. As mentioned in chapter Threshold, the proven optional threshold for CDIF is inversely proportional to the bin ordinal number. For the SDIF, the optimal threshold takes the form of the exponential function described in equation(0.8).

In the case of high pulse densities and a large number of interleaved signals, the PRI analysis by SDIF histogram, becomes complicated, and makes detection of emitters unreliable.

Example 3: The graphs display the first and second difference of an interleaved sequence with the individual periods 5, 8, and 12. The graphs display the first and second difference after the pulses with period 5 has been removed.
The simulation code is found in testtotal_SDIF.m.

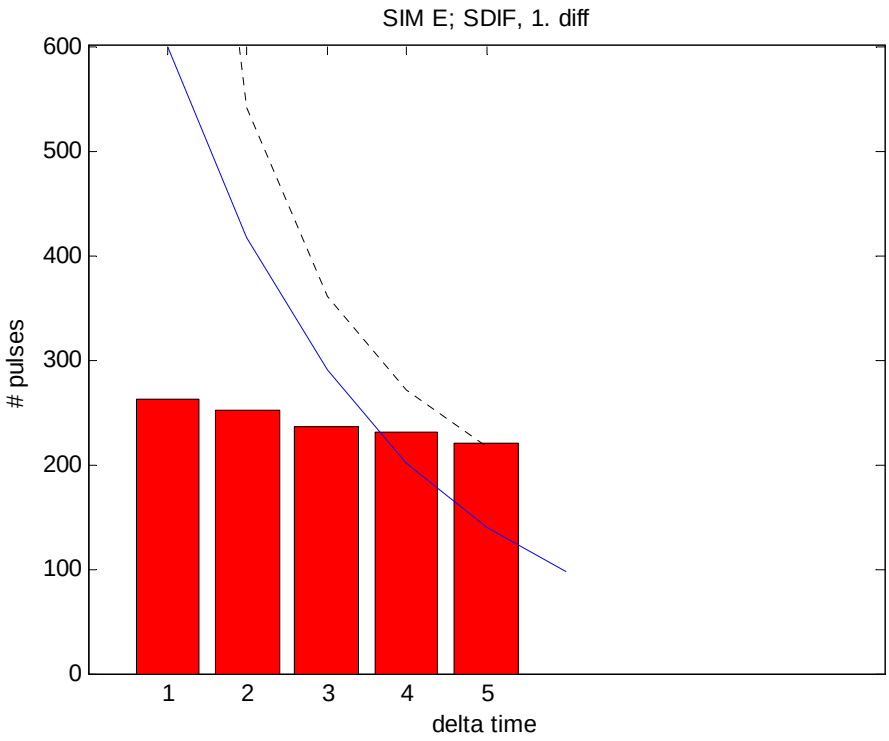


Figure 9

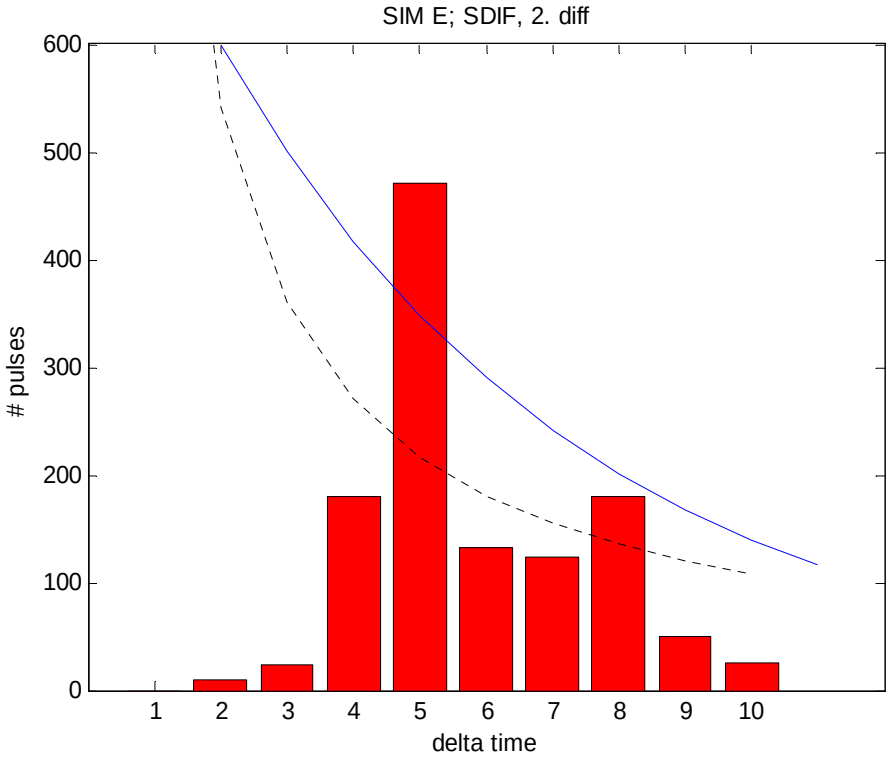


Figure 10

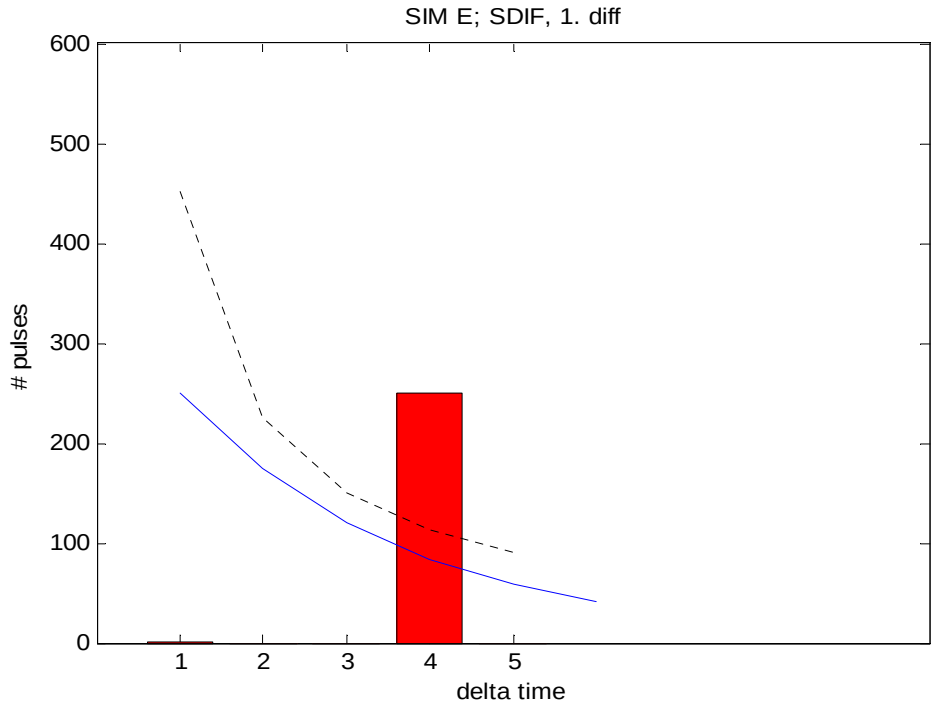


Figure 11

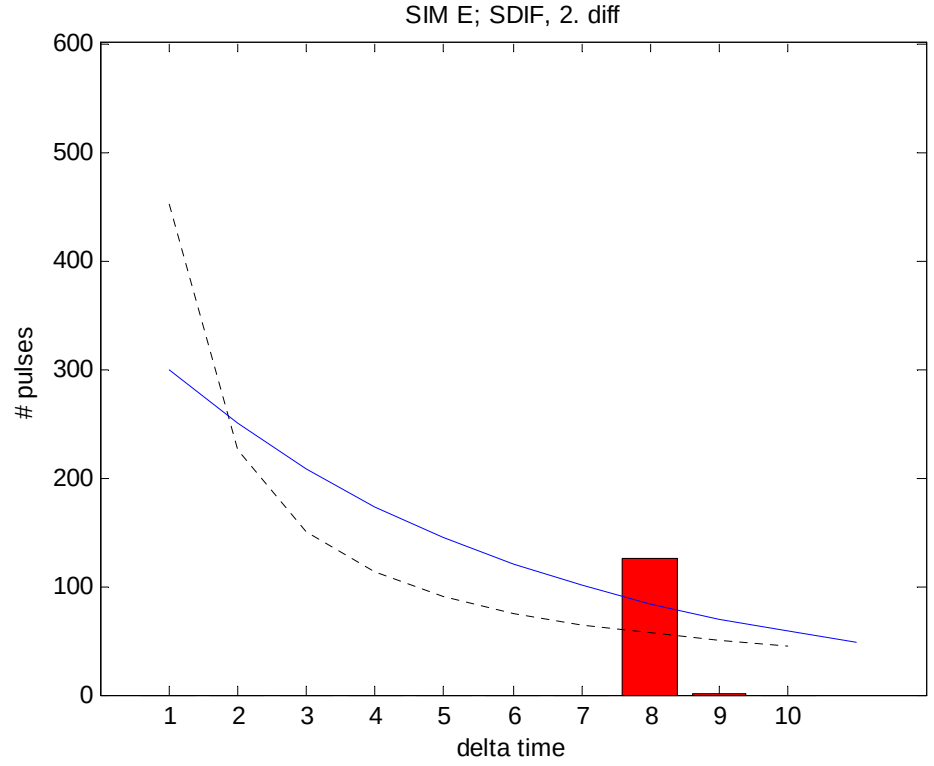


Figure 12

This simulation illustrate that is sufficient to compute the second difference and compare the histogram value to the threshold, as opposed to also compare the threshold to the double PRI and in so reduces the computation time with a factor of two.
The procedure is the same for dissolving the third period in the simulation above.

Some modifications will remove the imperfections.

1) If many emitters are present, the first difference histogram will produce a few values exceeding the threshold value, none of which corresponds to the correct PRI value. This is why the next difference level has to be computed without the previous sequential search.
2) In case of missing pulses, higher harmonics will exceed the threshold. This is only a problem if the time PRI exceeds the threshold, as the sequence search starts from the lowest PRI. This implies that sub-harmonic checking is necessary, and if it corresponds to some harmonic, this becomes the potential PRI. If not, sequential search is performed for all PRIs that exceed the threshold.

Spectra

Sequential search and histograms are computational expensive, typically in the order N^2, where N is the number of pulses to be evaluated. As an alternative, it is suggested to try to determine the number of pulse trains present and the frequency of each pulse train in the frequency domain. This has a computational complexity in the order of N log N. This makes the standard deinterleaving method; the sequential search used as a secondary method, relatively easy.

There exists several techniques to reveal the spectral content of an interleaved sequence, and some of them will be discussed here.

One solution is to sample the received signal with a frequency at least twice the expected highest PRF, and perform a FFT on the digital signal. The PRFs will appear as tops in the power density spectrum, computed as the square root of the FFT. This method does however rely on high processor performance because it typically requires the processing of at least 1 Mega samples per second, depending on the expected highest PRF.
A Fourier analysis of PDW-data will be able to extract the same information compared to the sampling; but with a reduction of size, with a data-rate of typically 1-100 kHz (depending of the PRF of the pulse trains), which especially gives an improvement for high PRFs. This is equivalent to the procedure done in the histogram methods.

The received interleaved signal consist of the superposition of M pulse trains, where t_0, t_1,...,t_N denotes the arrival times of N+1 consecutive pulses, setting $t_0 = 0$. The procedure can be thought of as taking the interval $[t_0, t_{N-1}]$, containing the first N arrival times, normalizing its length to approximately 2π, and then 'wrapping' the interval around the unit circle.

$$x(n) = e^{j\frac{2\pi}{t_{\max}}t_n} \quad \text{for } n = 0,1,....,N-1 \tag{0.9}$$

The magnitude of the discrete Fourier transform of the signal contains the information necessary to deduce the number of pulse trains, and to make an estimate of the frequencies.
Example 4: The graphs display the spectra of an interleaved sequence with individual periods 5, 8, and 11 and 13. The figures clearly distinguishes the periods, where the periodicity has been found by representing the pulses with delta pulses, and introduce zeros in between, representing zero samples. The second graph is plotted with logarithmic magnitude.

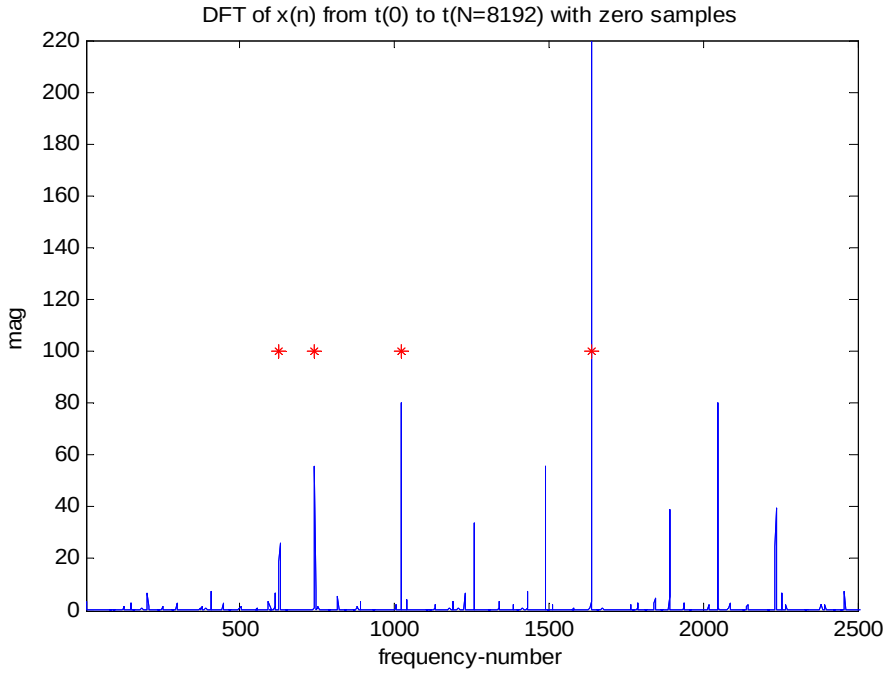


Figure 13

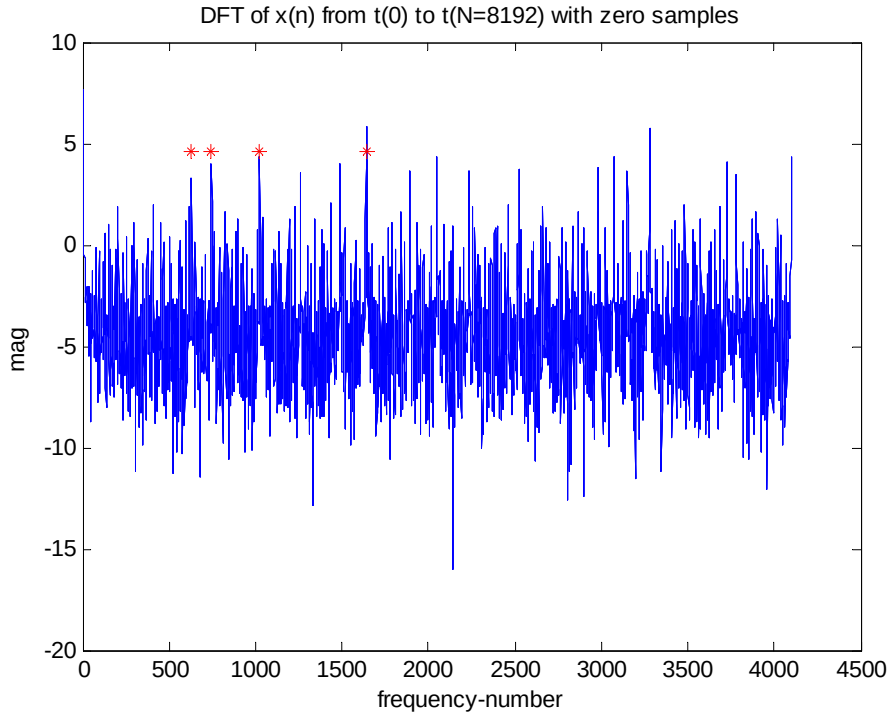
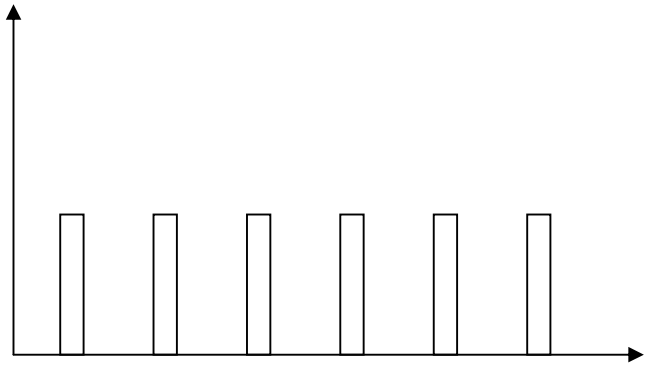


Figure 14

Pulse sorting transform

The Fourier transform represents a frequency domain representation for a time function, and as shown earlier it is very useful when trying to visualize the PRFs of the interleaved sequences. However, some problems may occur when using the Fourier transform to evaluate the spectra of interleaved pulse trains. If one considers a simple case where the observed pulse train consists of two pulse trains with the same period, but out of phase with half the period (anti-phase), the time sequence will appear to consist of a single pulse train with double the frequency. When evaluating the Fourier transform of the interleaved pulse train the individual frequencies are absent.



The PST is a modified version of the FT that allows one to identify both the single double frequency for the case over, and the individual frequencies. The main advantage when compared to other sorting techniques is that the PST presents the entire spectrum of the signal in a convenient form for pulse sorting.
The PST shares some properties with the FT, which is natural as it is a modification of the FT. The FT can be seen as a special case of the PST, but at the same time it is important to not view the PST as a generalization of the FT, since a generalized function should inherit most of the properties of its seed.
Linearity, scale change and time translation are common properties, but the PST lacks the convolution property. As the form of the FT varies according to if the applied signals are discrete or continuous in time, and finite or infinite in extent; so does the PST. A presentation with focus on the continuous time energy signals will be presented in the following to get a theoretic background, with the special case of the DFT version used in the simulations presented thereafter.
The FT for a continuous time energy signal v(t) is

(0.10) $$V(\Omega) = \int_{-\infty}^{\infty} v(t) * e^{-j\Omega t} \partial t$$

Where Ω is frequency measured in radians per second. The PST is a function of three variables;

(0.11) $$V(\Omega, \varphi, \beta) = \int_{\frac{\varphi}{\Omega} - \frac{\pi}{\Omega\beta}}^{\frac{\varphi}{\Omega} + \frac{\pi}{\Omega\beta}} \sum_{l=-\infty}^{\infty} v(t + l/F) * e^{-j\beta\Omega t} \partial t \text{ for } \beta > 1, F = \frac{\Omega}{2\pi} \text{ and } 0 \le \varphi \le 2\pi$$

β is the bin factor, φ is the phase, and the intervals of length $\frac{2\pi}{\Omega\beta}$ centered at $\frac{\varphi}{\Omega}$ are the phase bins.

Equation(0.10) can be written as

(0.12) $$V(\Omega) = \int_{0}^{1/F} \sum_{l=-\infty}^{\infty} v(t + l/F) * e^{-j\Omega t} \partial t$$

when choosing a frequency $F = \frac{\Omega}{2\pi}$ and noting that at this frequency the complex exponential is periodic with period $T = \frac{1}{F}$. The time function is thus segmented into increments of length *T* over its domain, and each segment is time shifted to a common interval from 0 to *T*, and added. The sum is then as shown in equation(0.12).
Because the exponential is periodic, the method of multiplying the time function v(t) with the exponential, and then integrating over infinite time, is equivalent with the method of forming the sum as shown in equation(0.12), multiplying by the exponential, and then integrate over only one period.
The PST is then formed from multiplying with β in the exponential, and then integrating over the interval of time centered at $\frac{\varphi}{\Omega}$ of length $\frac{2\pi}{\Omega\beta}$.

For β>1, the interval $\frac{2\pi}{\Omega\beta}$ is shorter than the interval $0 \to \frac{1}{F}$, which means that for a given φ and β, we are integrating over only parts of the function v(t). The parts are composed of the periodic segments of v(t), and it is this segmentation that gives the PST its sorting ability.

A second formulation of equation(0.11) is useful because it is easy to implement on computers for the discrete case, and its similarity to the discrete case presented later is easy to see.

The formulation is valid only for integer β.

(0.13) $$V(\Omega,\varphi,\beta)=\int_{-\infty}^{\infty}\hat{v}(t)*e^{-j\beta\Omega t}\partial t$$

$\hat{v}(t)$ is the segmented function $\hat{v}(t)=\begin{matrix}v(t)\rightarrow for\left|\varphi-\mathrm{mod}_{2\pi}(\Omega t)\right|<\frac{\pi}{\beta}\\0\rightarrow otherwise\end{matrix}$

The requirement of integer β is because of the complex exponential. In equation, the signal components are added first, and then multiplied by an exponential function. In an infinitely long complex exponential function is used to divide the signal into segments, which are equivalent only for integer β.
Illustrated by the figure, v(t) is the dotted line, and $\hat{v}(t)$ is the solid part. The solid parts occur at regular intervals spaced T seconds apart. The spiral depicts the exponential function, periodic with period $\frac{T}{\beta}$. As there are β cycles of the complex exponential in one period of length T; for integer β, the phase of the exponential is the same in each segment of $\hat{v}(t)$. This means that by multiplying the exponential by $\hat{v}(t)$ and integrating over all time, (0.13)is equivalent to(0.11) for integer β.

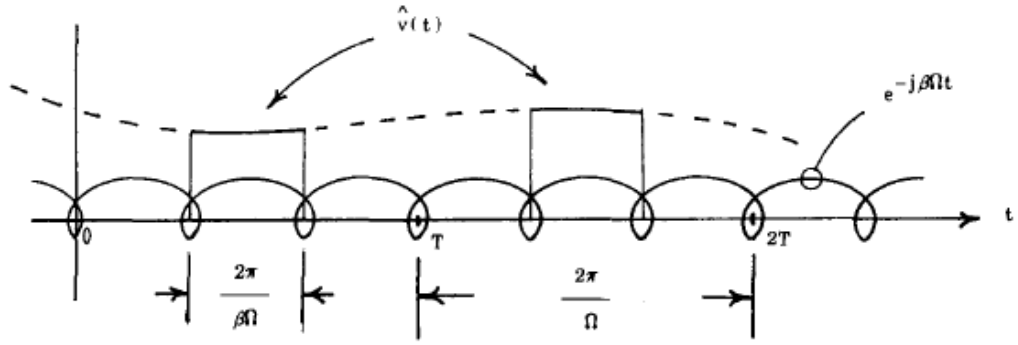


Figure 15

The FT provides an equivalent representation in the frequency domain because all the information contained in the time function is retained in the FT. This is clear because the original time function can be recovered from its transform.
If V (Ω,φ,β) is known for all Ω, φ and β, the inverse can be found by setting φ =π, and β =1. If φ is fixed or β is fixed but not an integer, the orthogonality is lost, and it becomes difficult to invert the function.
If V is known for all Ω and φ, and β is a fixed integer, the inverse can be calculated from

(0.14) $$v(t)=\frac{\beta}{2\pi}\int_{-\infty}^{\infty}\sum_{i=0}^{\beta-1}V_{i,\beta}(\Omega)*e^{j\beta\Omega t}\partial\Omega$$

Where $V_{i,beta}$ is the PST in the *i*th phase bin $V_{i,\beta}(\Omega) = V(\Omega, \varphi, \beta)|_{\varphi=(1+2i)*\frac{\pi}{\beta}}$

The FT of v(t) at frequency $\beta * \Omega$ is equal to the sum of the phase bins for the PST at frequency omega, that is; $V(\beta\Omega) = \sum_{i=0}^{\beta-1} V_{i,\beta}(\Omega)$

To evaluate the inverse transform for fixed integer beta; add the terms $V_{i,beta}$ from each phase bin, multiply by the exponential, and integrate over omega. Then multiply the result by $\frac{\beta}{2\pi}$

For discrete time signals the units for frequency change from radians per second to radians per sample.
For discrete time signals over a finite time interval (or periodic signals), the DFT is formulated as

(0.15) $$V(k) = \sum_{n=0}^{n<N/k} \sum_{l=0}^{k-1} v(n + lN/k) * e^{-j2\pi\beta kn/N}$$

corresponding to

When limiting the sum over *n* to the phase bin, and multiplying the frequency by β to obtain the discrete PST, V is

(0.16) $$V(k, \varphi, \beta) = \sum_{n=a}^{b} \sum_{l=0}^{k-1} v(n + lN/k) * e^{-j2\pi\beta kn/N}$$

with $$a = \frac{N}{2\pi k}(\varphi - \frac{\pi}{\beta}) \text{ and } b = \frac{N}{2\pi k}(\varphi + \frac{\pi}{\beta})$$

A second formulation that is valid only for integer β, and that is used in the simulations is

(0.17) $$V(k, \varphi, \beta) = \sum_{n=0}^{N-1} \hat{v}(n) * e^{-j2\pi\beta nk/N}$$

for $$\hat{v}(n) = \begin{matrix} v(n) \rightarrow |\varphi - \mathrm{mod}_{2\pi}(\frac{2\pi kn}{N})| < \frac{\pi}{\beta} \\ 0 \rightarrow otherwise \end{matrix}$$

Since the frequency isn't continuous, some modifications must be done when evaluating the inverse, compared to the previously mentioned method.

The following section illustrates the usefulness of the PST for various situations. The time and frequency axis consists of 120 values because 120 is a highly composite number, and at the same time a fairly low number will minimize the time needed to perform the simulations because of the complexity of the algorithm.
$\beta = 20$ because it is convenient for displaying the results, as mentioned in [4].

Example 5: The pulse train consists of two interleaved sequences, where both sequences has a period of 10, but the first sequence starts at n=3, and the other at n=6. Figure 16 displays the first 6 pulses of each sequence.
Figure 17 displays the FFT of the PT, which indicates two smaller components at k=12 and k=48, and two larger components at k=24 and k=36. This tells us nothing of the nature of the individual pulse trains other than each has a period of 10.
Figure 18 and 20 displays the magnitude of the PST, and the two pulse trains appear clearly as peaks at k=12, around phase bins 0.45π and 1.25π

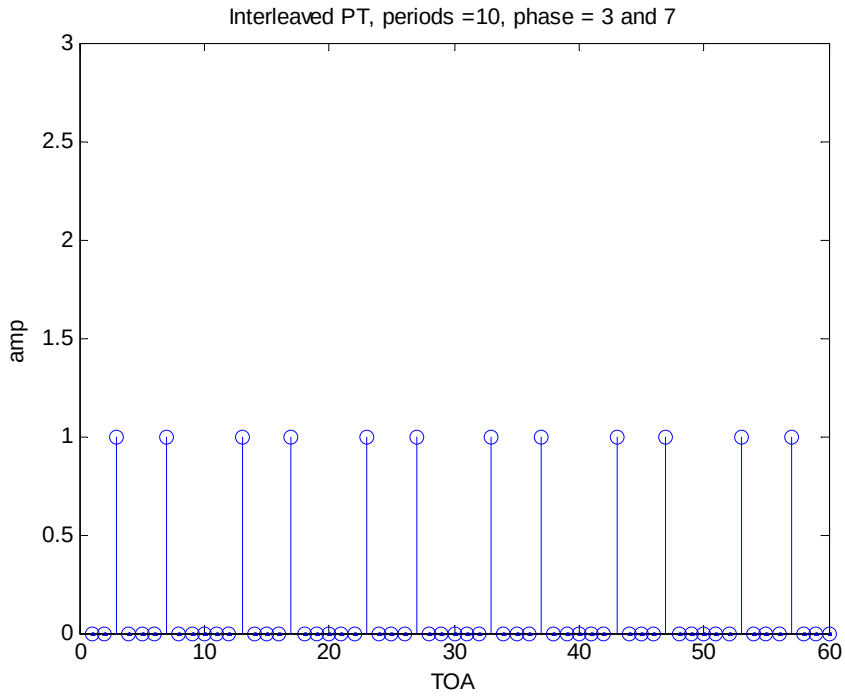


Figure 16

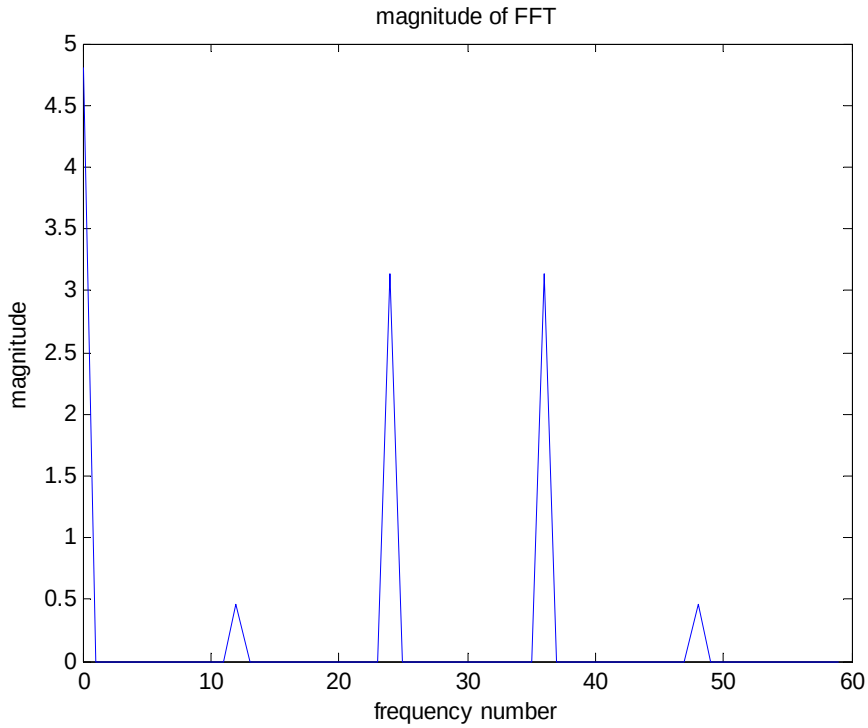


Figure 17

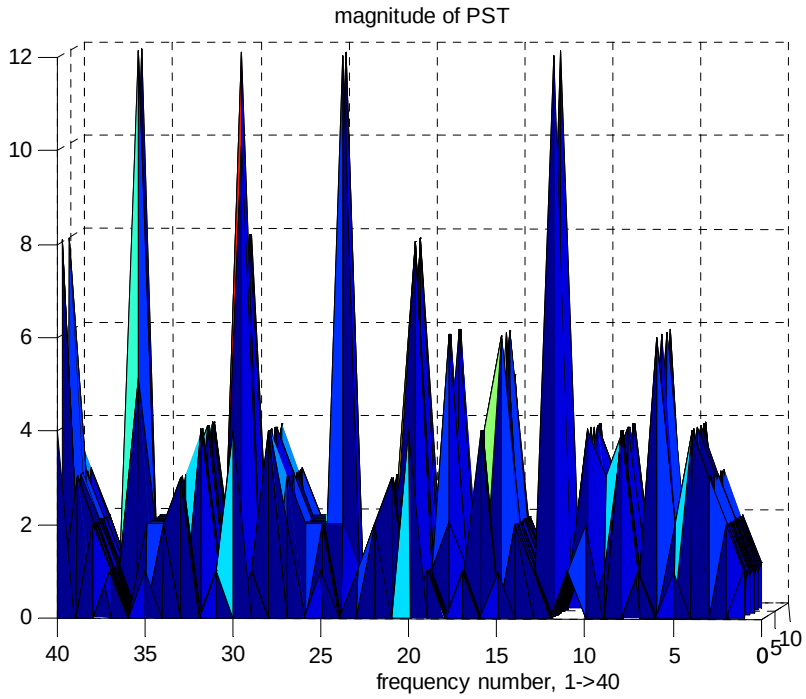


Figure 18

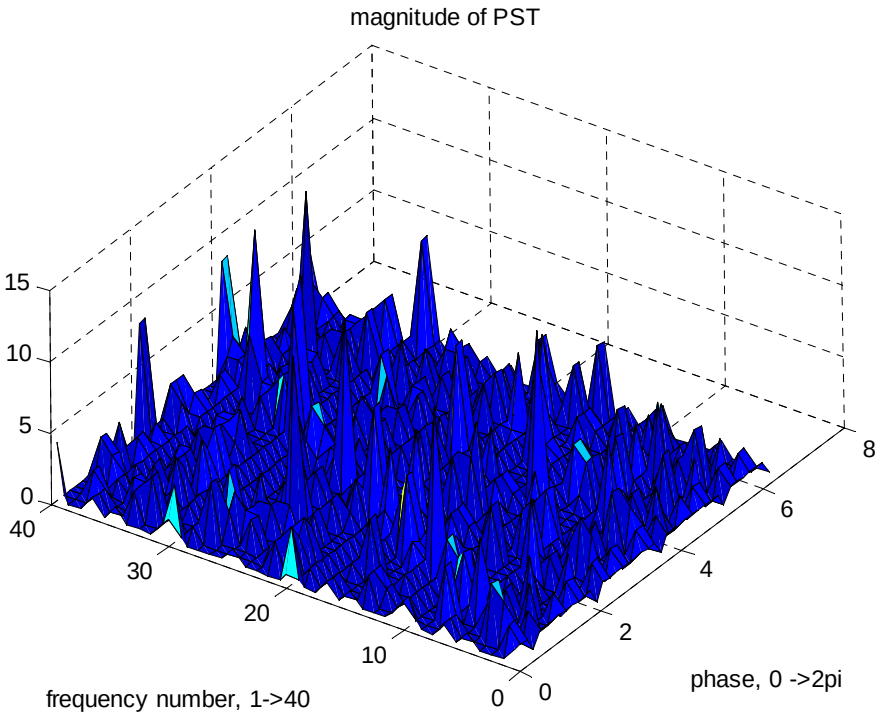


Figure 19

Example 6: The total pulse train consists of two interleaved sequences both with period 12. The phase is n=6 and n=12, which makes it difficult to decide the true nature of the sequence, i.e. it can either be viewed as a single sequence of period 6, or as two interleaved sequences of period 12 in anti-phase. Figure 20 displays the sequence in the time domain. Figure 21 displays the DFT, where only one train with k=20 is present, which indicates only a single pulse train with period 6. The discrete PST magnitude plot shows the mentioned component at k=20, but also indicates two smaller components at k=10, appearing in phase bin π and 2π.

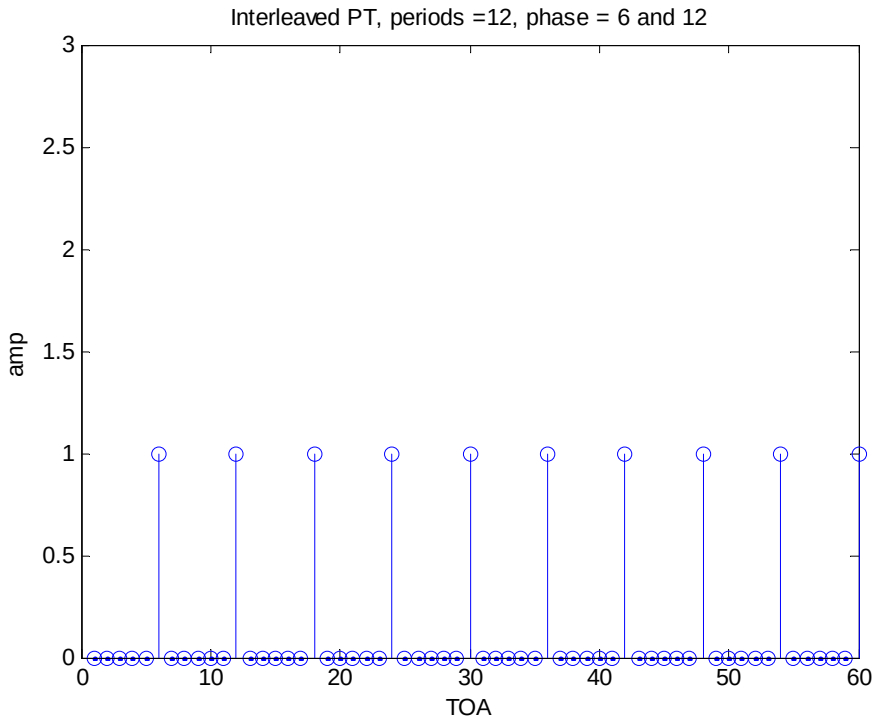


Figure 20

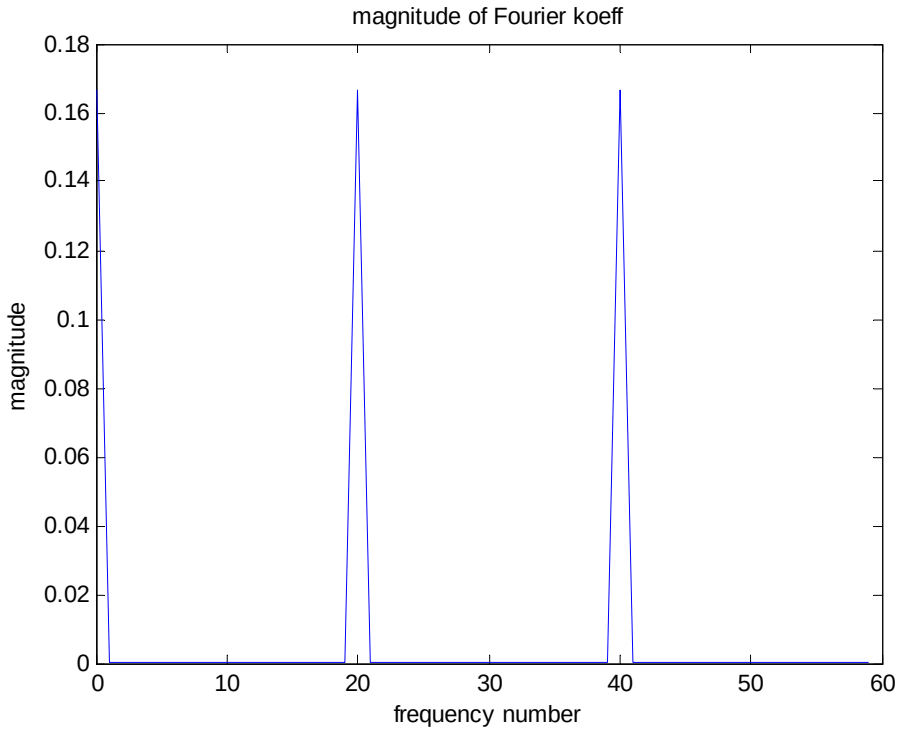


Figure 21

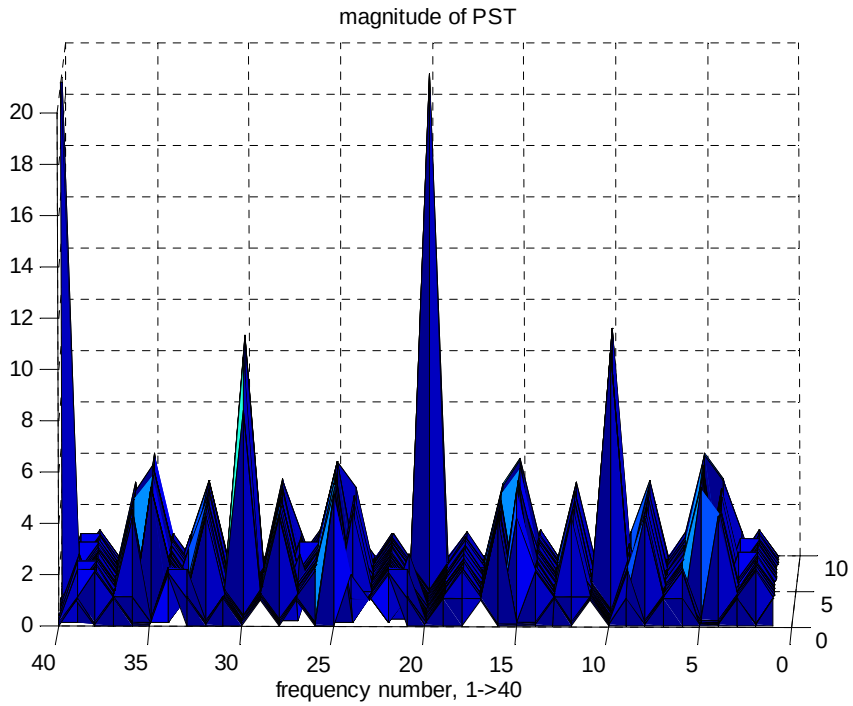


Figure 22

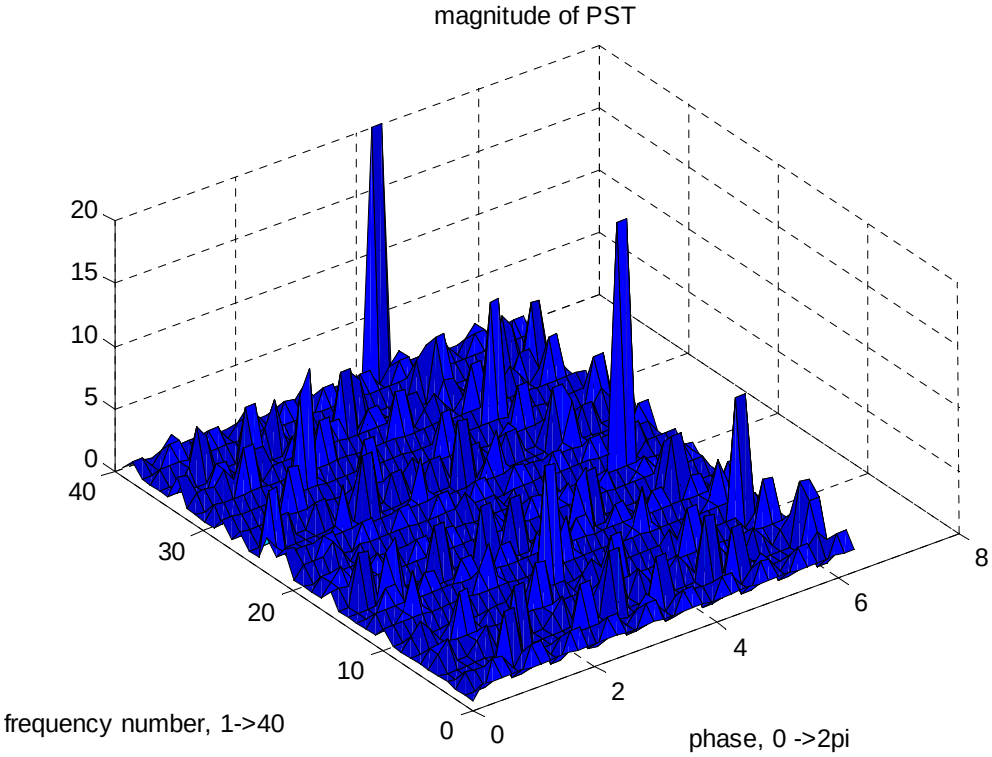


Figure 23

Example 7: The total pulse train consists of five interleaved sequences with periods 3, 4, 5, 8 and 10. The phases are x=6, 12, 4, 5 and 9. Figure 24 displays the 60 first pulses, while figure 25 displays the magnitude of the DFT, and Figure 26 plots the magnitude of the discrete PST.

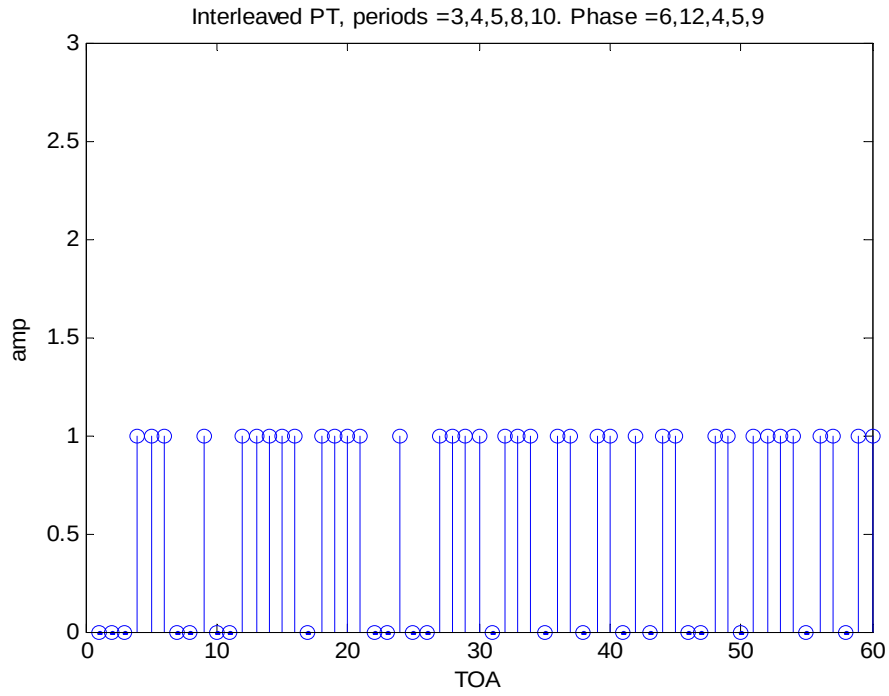


Figure 24

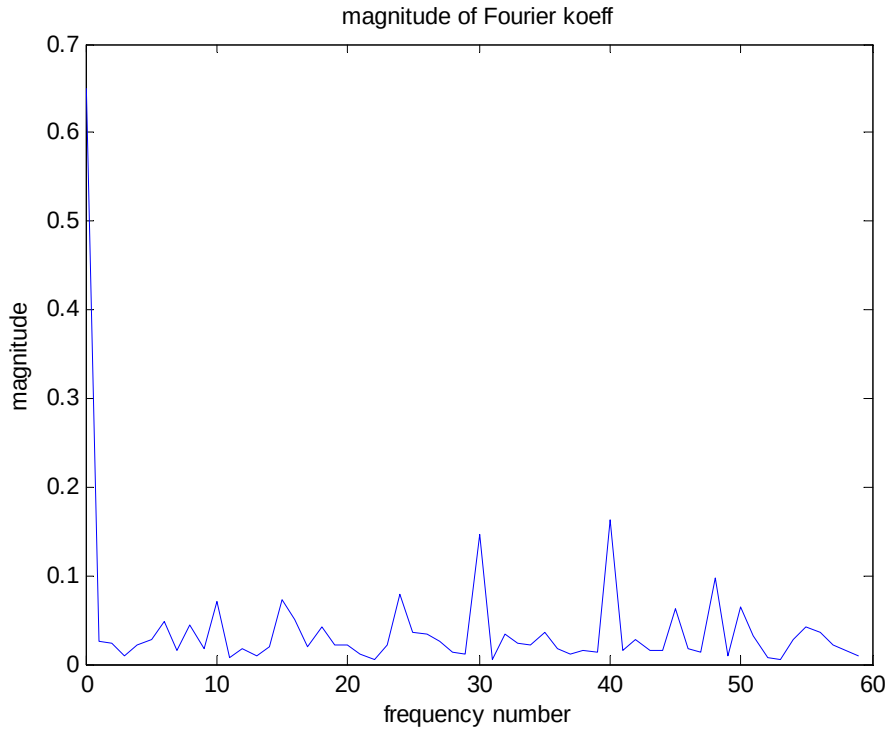


Figure 25

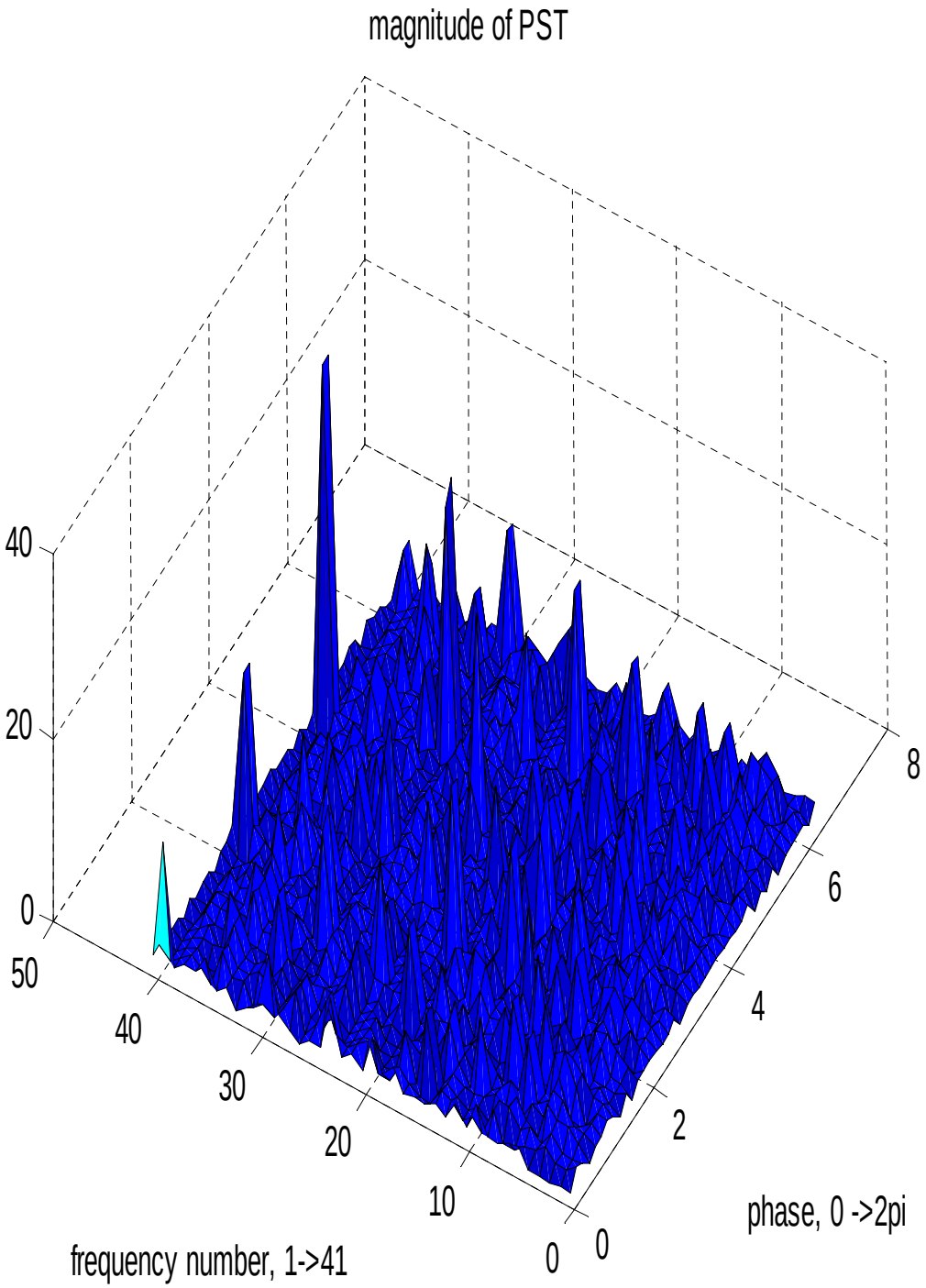

magnitude of PST
40
20
0
50
40
30
20
10
0
frequency number, 1->41
0
2
4
6
8
phase, 0 ->2pi

Figure 26

Neural network

The ESM system is made up of at TOA deinterleaver and a neural net recognition system, in addition to a receiver and a signal separator.

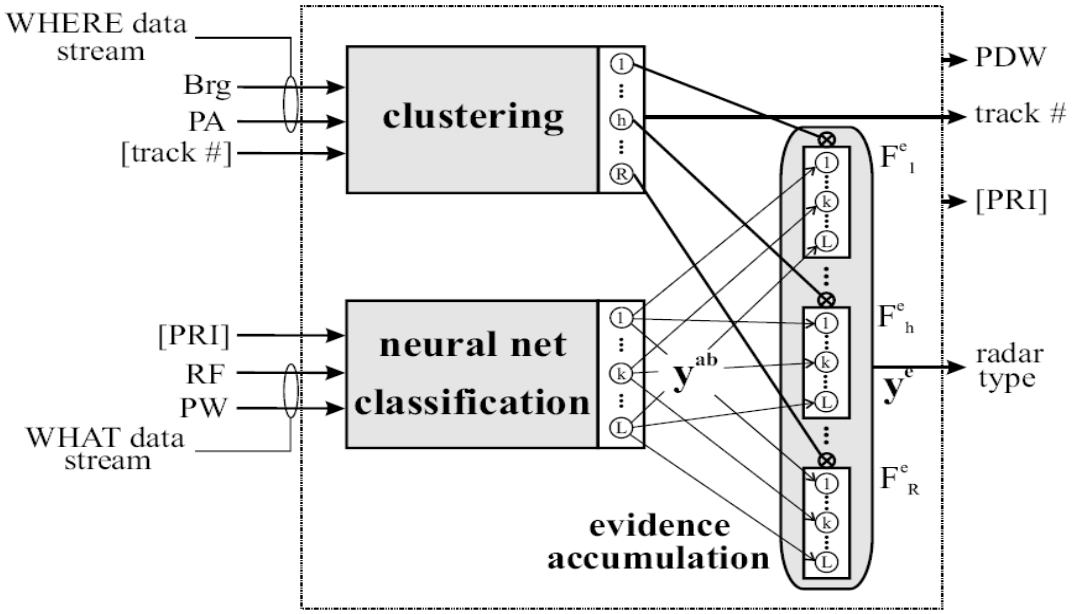
The intercepted radar pulses are presented to the TOA deinterleaver, which discover periodicities in the incoming Pulse Description Words, on the basis of some algorithm.

The recognition subsystem is in some ways inspired by the human brain, in that the processing is done by separating parameters that describe what you see, and where it is located, i.e. spatial information. The parameters describing a radar emitter typically consist of values like the pulse width and carrier frequency, and also the pulse repetition interval, even tough this parameter is not directly measurable, but has to be derived. The parameters that inhabit the spatial information will typically be derived from a directional receiver, and give values on such parameters as direction and pulse amplitude. They will not contribute in the same degree as the other parameters in defining the emitter type, but will be useful when grouping pulse trains, and comparing previously recorded tracks[2].

The outputs from the clustering device and neural net classification are presented to the evidence accumulation field, where responses associated with classifications are accumulated over time with spatial information.

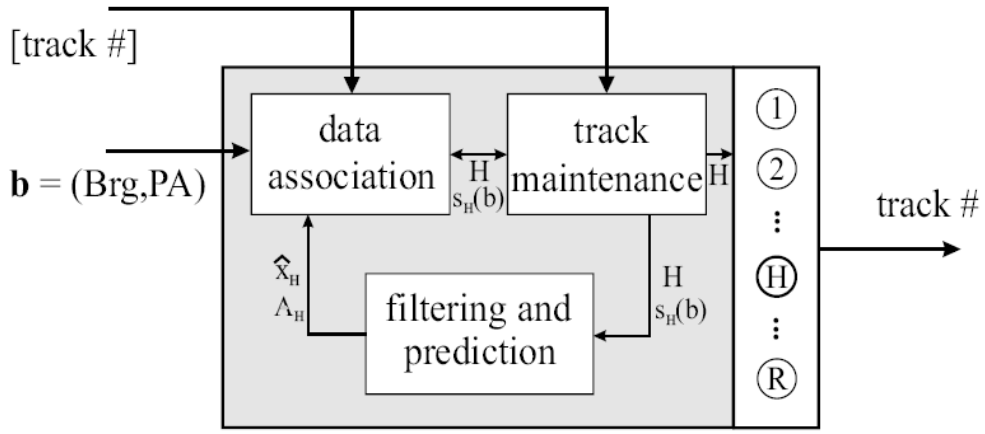
The neural network subsystem predicts a radar type for each pulse on the basis of pulse width and radio frequency, in the form of a response pattern. This is then associated with the output from the clustering device. This will allow for PDW information to be directly connected to the spatial information, resulting in enhanced classification accuracy.

2 ***A track is defined as the recognized emitter signature from one loading of the pulse buffer to the next.***



Clustering

The choice of algorithm for solving the clustering problem is based on the rate of change in values, and a less sophisticated solution may be used if parameters vary slowly. The spatial parameters are initially associated with already existing tracks, and if no matches are found, a new track is initiated. When there is a lack of responses associated with a track, it is deleted.

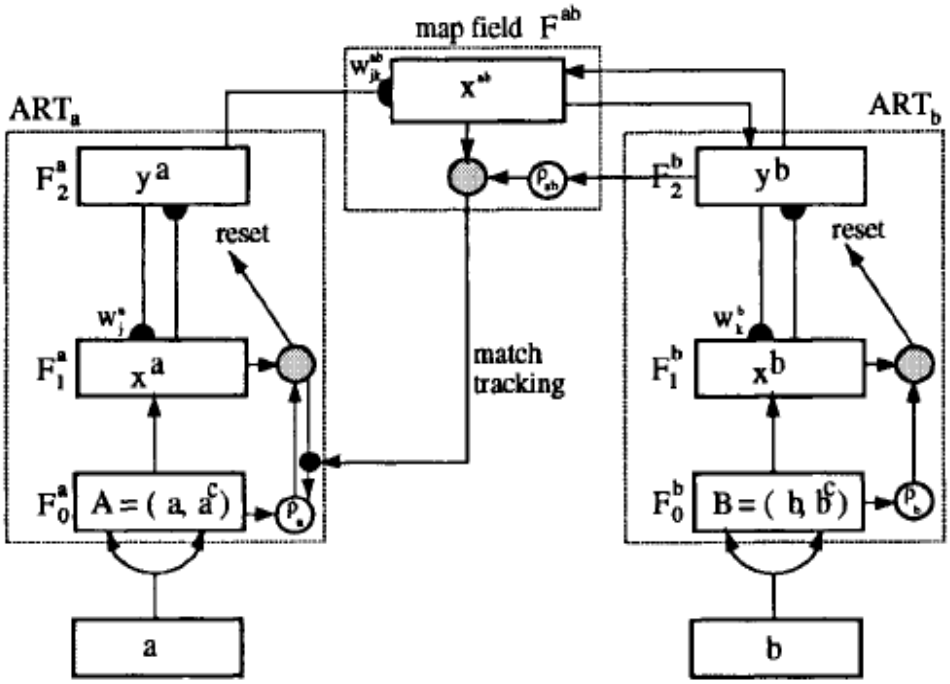


The Kalman filter is a very popular algorithm, and optimum for estimating stochastic system processes. But since it is a complex algorithm to implement, it has some drawbacks. Mainly, in the actual scenario where on-line processing is a prerequisite, the fact that the KF uses a huge number of operations becomes a strain on the system, and the KF can be considered a memory killer [5]. In so, efforts are made in considering other possible solutions, to avoid letting the KF induce a bottleneck.

Fuzzy ARTMAP

The proposed neural net is called fuzzy ARTMAP, and an overview of the architecture shows the basic principles, where an input vector is presented to ART_a, and the map field is used to associate a category number to each neuron of ART_a's F2 layer. ART_b is used to present target values, and is not used during testing. The vigilance parameter ρ is a measure on the degree of mismatch. The weights represent the 'memorized' patterns, and the two primary weight vectors are the top-down weights from F2 to F1 in ART_a that represent the group of input patterns that chose node j in the category layer as their node, and the weights from every node j in F2 to the output layer F2 in ART_b that correspond to the output pattern that node j is mapped to.

An input vector presented to the F1 layer in ARTMAP, represents a point in space. For each neuron in the F2 layer, a choice function is evaluated, with a purpose of choosing the smallest hyperbox where the point is represented. If the point is represented in more than one box, the smallest box is chosen. If the point is not represented in any boxes, the box that needs to be expanded the least, or a new one, is chosen.
After the neuron is chosen, the vigilance criterion is evaluated. If the box is already too large, the neuron with the second highest choice function decides the new box. The vigilance criterion will by this decide the maximum size of the hyperboxes.
When choosing fast learning, represented with β, the box is expanded just enough to include the point that the input vector represents. With β less than one makes the box move in the direction of the point.


map field F^{ab}
ART_a
ART_b
F_2^a
y^a
reset
w_j^a
F_1^a
x^a
F_0^a
$A = (a, a^c)$
a
match
tracking
F_2^b
y^b
reset
w_k^b
F_1^b
x^b
F_0^b
$B = (b, b^c)$
b

The parameters must be normalized accordingly prior to network presentation.
ART flow when presented with a normalized vector:

- complement code
- create network
- learn
    - 1) activate categories, bottom-up, then top-down
    - 2) add new category
    - 3) update weights
    - 4) calculate match
    - 5) (if match >vigilance; category codes input) update weights & induce resonance
    - 6) (else; sort next category in sorted list, check that the maximum number of categories isn't reached) create new category, update weights and induce resonance
    - 7) if no change in network in epoch, equilibrium reached, stop training
- categorize

ARTMAP flow:

- complement code
- create network
- learn, with supervised input, same procedure as last time
- categorize

The simulation gives the response of the network for a combination of stable and staggered emitters, with parameters given in the tables in the following. Results are given when trained on each stable emitter.

The idea is to get some measurement on how close in the Euclidian sense the test set can lie in relation to the training set, before the network is unable to categorize the test set for the given vigilance.
A trade off between the ability to recognize and the degree of separation must be made, because the network cannot simultaneously have absolute separation and at the same time recognize similarities.

- Emitter 1 has values with PRF 24 kHz, RF 1 GHz, and PW 800 ns.
- Emitter 2 has values with PRF 200 Hz, RF 5 GHz, and PW 3 µs.
- Emitter 3 has values with PRF 5 kHz, RF 11 GHz, and PW 10 ns.

The simulations where not as much intended to reflect a real emitter, but to get a clear understanding of the network with a good enough separation of the emitters in the 3 dimensions. The only certain parameter range is the RF which is limited by the antenna, and given as part of the system description from [1]. The PW and the PRF are parameters that vary from manufacturer to manufacturer and model to model, where accurate behavioral description is classified for obvious reasons.

One of the distinct features of a neural network is its opaqueness. This means that the actual response is hard to describe by mathematical means. The following two simulations will try to deduce what kind of response the network has when the PRI parameter varies. This will give some idea of the limitations of the network when the deinterleaving module labels a staggered emitter with its PRI, even though it isn't a stable emitter. The choice of using the Euclidian distance measurement is based on its simplicity, and that it is a basis in many metric systems. For example; a given pulse is described by three parameters, and in so it has a point in the 3-dimensional parametric space. When a system is presented with a new pulse with the intent to do a classification, the two points in the parametric space are compared accordingly to the Euclidian distance, described as

(0.18) $$dis\tan ce = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}$$

Since the network requires normalized values[3], the parameters have been divided by the maximum expected value.

- The RF maximum value = 12 GHz
- The PRI maximum value = 0.1s
- The PW maximum value = 10 µs

3 ***The network also requires a minimum value to be chosen. This is not necessary for the simulations in the following.***

Example 8: The network is trained on the parameters in table 4, and tested on the parameters in table 5. The parameters are plotted in the 3D space in figures 27 and 28. The results are displayed in table 6. Figure 29 displays the Euclidian distance as a function of the pulse. The network indicates a -1 when the response on the presented test pulse is deduced as a new class emitter.
The most interesting features of the simulation of the PRI variations are where the network decides where to put the boundaries for what is considered a known PRI. An important factor to keep in mind is that the smaller the PRI, i.e. the higher the PRF, the smaller does the Euclidian distance become. This is why the network somewhat surprisingly considers a PRF of ~1.8 kHz trained on a PRF of 24 kHz more similar than a PRF of ~ 133 Hz trained on a PRF of 200 Hz.

Table 4

PRI	RF	PW	Name
4,17E-05	1,00E+09	8,00E-07	**Em1**
5,00E-03	5,00E+09	3,00E-06	**Em2**
2,00E-04	1,10E+10	1,00E-08	**Em3**

Table 5

PRI	RF	PW	Name
4,17E-05	1,00E+09	8,00E-07	**Em1**
5,42E-04	1,00E+09	8,00E-07	**Em1_stagger1**
1,04E-03	1,00E+09	8,00E-07	**Em1_stagge2**
2,04E-03	1,00E+09	8,00E-07	**Em1_stagger3**
5,00E-03	5,00E+09	3,00E-06	**Em2**
5,30E-03	5,00E+09	3,00E-06	**Em2_stagger1**
7,50E-03	5,00E+09	3,00E-06	**Em2_stagger2**
8,50E-03	5,00E+09	3,00E-06	**Em2_stagger3**
2,00E-04	1,10E+10	1,00E-08	**Em3**
4,70E-03	1,10E+10	1,00E-08	**Em3_stagger1**
5,70E-03	1,10E+10	1,00E-08	**Em3_stagger2**
6,70E-03	1,10E+10	1,00E-08	**Em3_stagger3**

Table 6

Name	Euclidian distance	0,9	0,99	0,999	0,99999
Em1	0	1	1	1	1
Em1_stagger1	0,005	1	1	-1	-1
Em1_stagger2	0,009983	1	1	-1	-1
Em1_stagger3	0,019983	1	1	-1	-1
Em2	0	2	2	2	2
Em2_stagger1	0,003	2	2	2	-1
Em2_stagger2	0,025	2	2	-1	-1
Em2_stagger3	0,035	2	-1	-1	-1
Em3	0	3	3	3	3
Em3_stagger1	0,045	3	-1	-1	-1
Em3_stagger2	0,055	3	-1	-1	-1
Em3_stagger3	0,065	3	-1	-1	-1

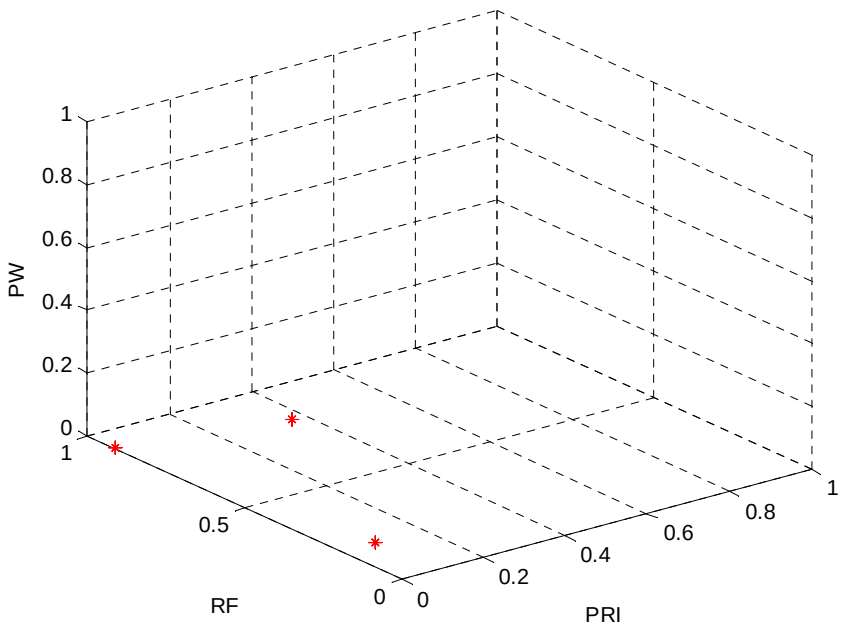


Figure 27

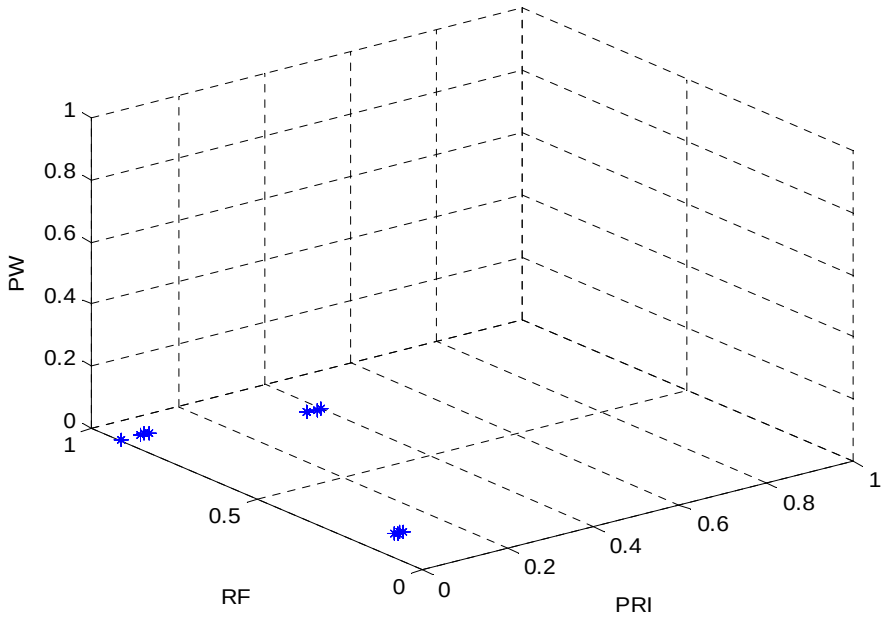


Figure 28

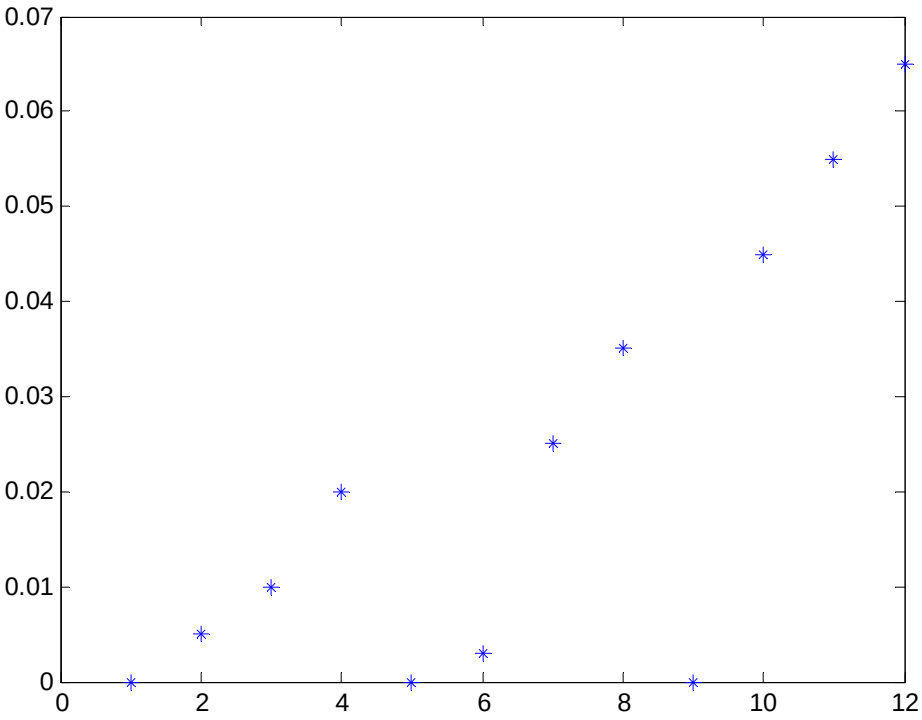


Figure 29

Example 9: The simulation is similar to the previous, only with a larger Euclidian distance of the parameters. The network is trained on the parameters in table 4, and tested on the parameters in table 7. The parameters are plotted in the normalized parametric space in figures 30. The results are displayed in table 8.
From the Euclidian distance it is clear that a vigilance value below 0.99 will make the network too modest regarding the separation of pulses.

Table 7

PRI	RF	PW	Name
4,17E-05	1,00E+09	8,00E-07	**Em1**
1,00E-02	1,00E+09	8,00E-07	**Em1_stagger1**
2,00E-02	1,00E+09	8,00E-07	**Em1_stagger2**
3,00E-02	1,00E+09	8,00E-07	**Em1_stagger3**
5,00E-03	5,00E+09	3,00E-06	**Em2**
4,50E-02	5,00E+09	3,00E-06	**Em2_stagger1**
5,50E-02	5,00E+09	3,00E-06	**Em2_stagger2**
6,50E-02	5,00E+09	3,00E-06	**Em2_stagger3**
2,00E-04	1,10E+10	1,00E-08	**Em3**
7,00E-02	1,10E+10	1,00E-08	**Em3_stagger1**
8,00E-02	1,10E+10	1,00E-08	**Em3_stagger2**
9,00E-02	1,10E+10	1,00E-08	**Em3_stagger3**

Table 8

Name	Euclidian distance	0,8	0,9	0,99
Em1	0	1	1	1
Em1_stagger1	0,099983	1	1	-1
Em1_ stagger2	0,199583	1	1	-1
Em1_ stagger3	0,299583	1	1	-1
Em2	0	2	2	2
Em2_ stagger1	0,4	2	-1	-1
Em2_ stagger2	0,5	2	-1	-1
Em2_ stagger3	0,6	-1	-1	-1
Em3	0	3	3	3
Em3_ stagger1	0,698	-1	-1	-1
Em3_ stagger2	0,798	-1	-1	-1
Em3_ stagger3	0,898	-1	-1	-1

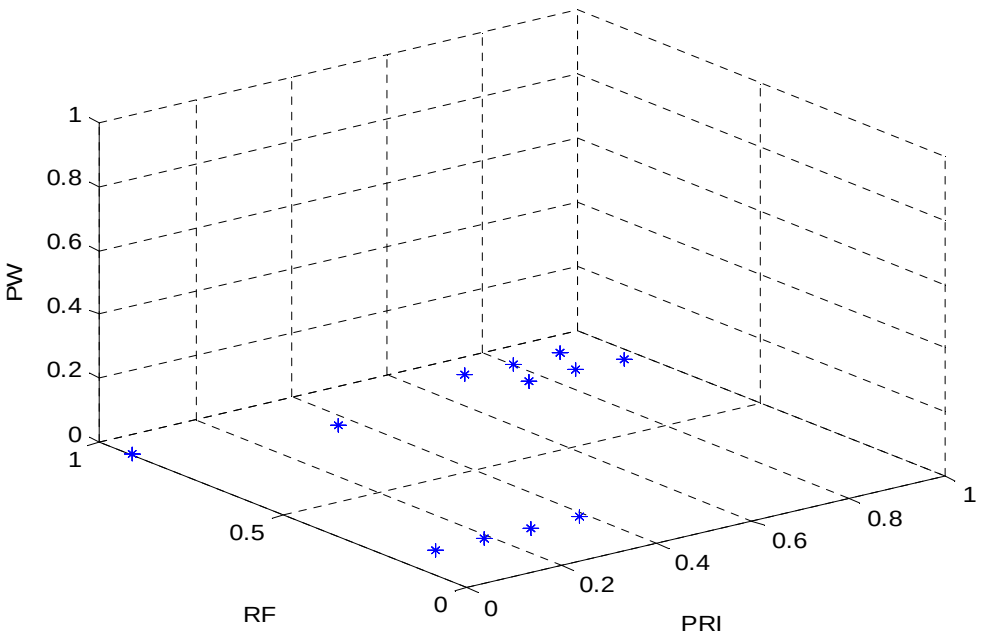


Figure 30

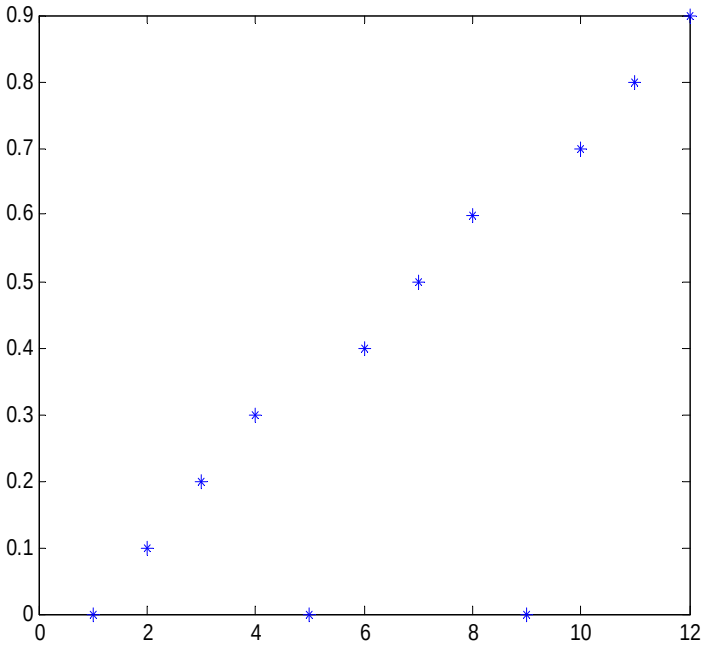


Figure 31

The following two simulations will try to deduce what kind of response the network has when the RF parameter varies.

Example 10: The network is trained on the parameters in table 4, and tested on the parameters in table 9. The parameters are plotted in figure 32. The results are displayed in table 10.
From table 10 it is clear that the network respond in the same way independent of the parameter, but dependent on the similarity, expressed here as the Euclidian distance from the trained parameter, which was as expected. In addition a Euclidian distance between 0.25 and 0.035 is an ideal value for maintaining the generalization of the network.

Table 9

PRI	RF	PW	Name
4,170E-05	1,000E+09	8,000E-07	**Em1**
4,170E-05	1,060E+09	8,000E-07	**Em1_rf_agile1**
4,170E-05	1,120E+09	8,000E-07	**Em1_rf_agile2**
4,170E-05	1,240E+09	8,000E-07	**Em1_rf_agile3**
5,000E-03	5,000E+09	3,000E-06	**Em2**
5,000E-03	5,036E+09	3,000E-06	**Em2_rf_agile1**
5,000E-03	5,300E+09	3,000E-06	**Em2_rf_agile2**
5,000E-03	5,420E+09	3,000E-06	**Em2_rf_agile3**
2,000E-04	1,100E+10	1,000E-08	**Em3**
2,000E-04	1,154E+10	1,000E-08	**Em3_rf_agile1**
2,000E-04	1,166E+10	1,000E-08	**Em3_rf_agile2**
2,000E-04	1,178E+10	1,000E-08	**Em3_rf_agile3**

Table 10

Name	Euclidian distance	0,9	0,99	0,999
Em1	0	1	1	1
Em1_rf_agile1	0,005	1	1	-1
Em1_rf_agile2	0,01	1	1	-1
Em1_rf_agile3	0,02	1	1	-1
Em2	0	2	2	2
Em2_rf_agile1	0,003	2	2	2
Em2_rf_agile2	0,025	2	2	-1
Em2_rf_agile3	0,035	2	-1	-1
Em3	0	3	3	3
Em3_rf_agile1	0,045	3	-1	-1
Em3_rf_agile2	0,055	3	-1	-1
Em3_rf_agile3	0,065	3	-1	-1

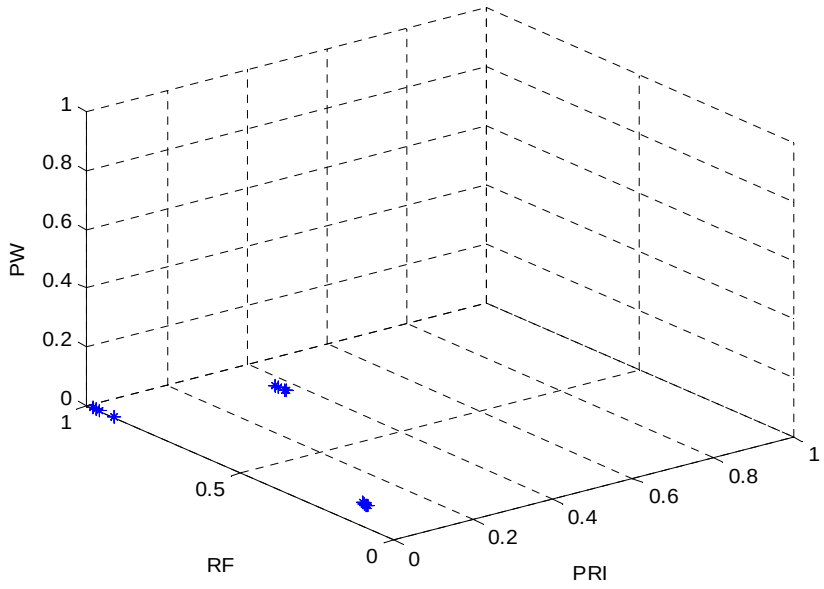


Figure 32

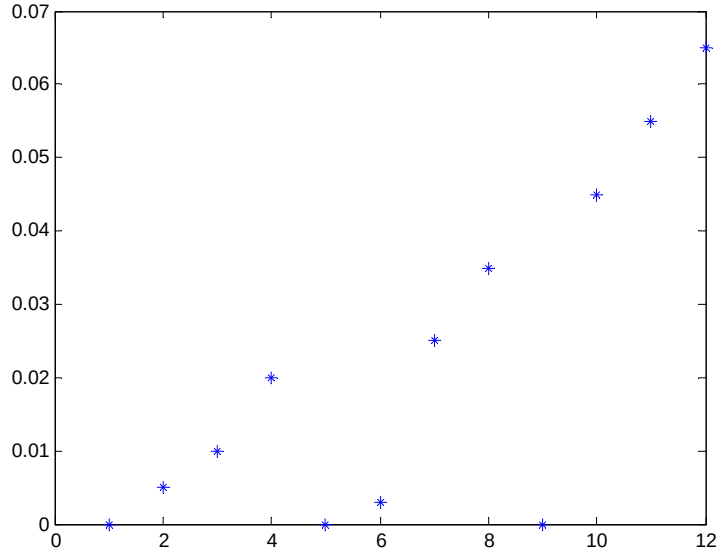


Figure 33

Example 11: The network is trained on the values in table 4, and tested on the values of table 11. The simulation seeks to confirm the already acquired information.

Table 11

PRI	RF	PW	Name
4,170E-05	1,000E+09	8,000E-07	**Em1**
4,170E-05	2,200E+09	8,000E-07	**Em1_rf_agile1**
4,170E-05	3,400E+09	8,000E-07	**Em1_rf_agile2**
4,170E-05	4,600E+09	8,000E-07	**Em1_rf_agile3**
5,000E-03	5,000E+09	3,000E-06	**Em2**
5,000E-03	9,800E+09	3,000E-06	**Em2_rf_agile1**
5,000E-03	1,100E+10	3,000E-06	**Em2_rf_agile2**
5,000E-03	1,180E+10	3,000E-06	**Em2_rf_agile3**
2,000E-04	1,100E+10	1,000E-08	**Em3**
2,000E-04	2,600E+09	1,000E-08	**Em3_rf_agile1**
2,000E-04	1,400E+09	1,000E-08	**Em3_rf_agile2**
2,000E-04	1,000E+09	1,000E-08	**Em3_rf_agile3**

Table 12

Name		0,8	0,9	0,99
Em1	0	1	1	1
Em1_rf_agile1	0,1	1	1	-1
Em1_rf_agile2	0,2	1	1	-1
Em1_rf_agilee3	0,3	1	1	-1
Em2	0	2	2	2
Em2_rf_agile1	0,4	2	-1	-1
Em2_rf_agile2	0,5	3	-1	-1
Em2_rf_agile3	0,566667	3	-1	-1
Em3	0	3	3	3
Em3_rf_agile1	0,7	1	1	-1
Em3_rf_agile2	0,8	1	1	-1
Em3_rf_agile3	0,833333	1	1	-1

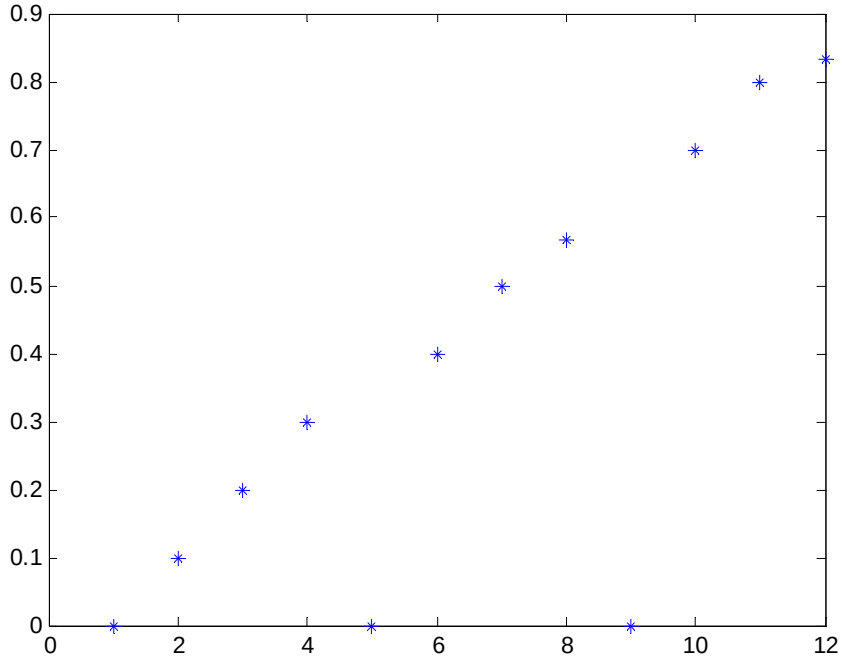


Figure 34

Table 12 confirms the choice of a vigilance of 0.99 as an ideal value. Another interesting feature of the simulation is that even though Em2_rf_agile seems closer to the third training parameters, it is classified as an Em2, while Em2_rf_agile2 Em2_rf_agile3 are classified as an Em3, for a vigilance of 0.8. The same is seen for the three last test patterns, which are classified as a class 1. This means that we get some measurement on the classification response when a test pulse lies in the vicinity of a training-pulse other than the intended.
There are several cases one must consider when dealing with the limitations of the network. When training the network, one can choose to train it on real data collected in the field, or on generated data. In this phase it is insignificant regarding the classification-accuracy of the network, but is necessary keep in mind if implementing.

Several possible cases can be mentioned in connection with missing data. A limited number of training cases will generally reduce the classification capability of a neural network; however the fuzzy ARTMAP has shown great potential in accuracy when trained only on a few pulses from each emitter, this is also the case from initial simulations.
Due to processing delays and other hardware limitations, some input patterns could be presented to the classifier with missing components. In operational mode, the classifier will be presented to new patterns, whenever a currently available emitter changes mode, or when encountering new emitters all together. Since it’s impossible to train the network on the total emitter environment, the network will employ methods to discriminate the previously unseen patterns from the already known patterns; this is invoked with a technique that gives a degree of familiarity between the patterns. This has not yet been tested, but initial simulations give results that tend toward a good generalization capability.

HW

In a modern ESM system, it is necessary to be able to handle high-pulse arrival rates and deinterleave pulse trains in a dense environment, where the data often are corrupted, several pulses are missing and it must operate on unknown PRI modulations. Normally, and as described earlier, pulse parameters are compared against previously established pulse groups (emitter bins), where they are checked serially to see if they fall within some defined tolerance level of the existing groups. If they are, they are assigned to the group; otherwise a new group is made. The next step is then to update the tolerance levels, based on current statistics.
Depending on the algorithm, some general drawback are the inherent slowness of performing ruled-based operations sequentially, the lack of being able to provide a quality measure on the degree of match between the parameters and the groups, and they will respond with less reliability when the data are noisy, and parameters are missing etc. In addition, a small failure in general HW can cause the system to fail.

Hardware implementations

This section seeks to give a varied presentation of possible solutions of implementing a NN in HW, and will include both realized and non-realized solutions. A detailed evaluation of all solution will not be given because of the difficulty of comparing HW from published performance figures.

There are many ways to implement the NN, and they can be categorized accordingly to whether they use analog or digital computation, how the synaptic weights are stored, if there is some kind of on-chip learning, and if the chips are meant to be standalone chips, or has the ability to make larger networks by connecting to other chips; multi-chip systems.

Important considerations to be made when dealing with VLSI are chip area, performance and power consumptions, and trade-offs must often be made between the performance and the flexibility when implementing NN in VLSI. It is also important to estimate their values as early as possible in the design process, as this will facilitate the architectural exploration.

Analogue integrated neural networks have an advantage when it is possible to use parallel data processing elements, which speeds up data processing. Analogue data PE are slower than their digital equivalents, but the analogue versions of the neural networks computing primitives; i.e. multiplication and addition, can be much smaller than their digital equivalents.
A massively parallel NN should be application specific, as the exact mapping on parallel HW depends on network topology. For high precision computations digital circuits are required, but the precision offered by analogue components are generally believed to be sufficient [6].

In addition, there are some additional features that make an analogue implementation worth considering. Many NN are asynchronous, i.e. self-timed, and does not need to be governed by a clock, and can thereby run at the maximum speed of the HW. In synchronous systems all components change states simultaneously at the clock edges, and thus draw current from power supply simultaneously with the demands this puts on the power supply peak currents. Asynchronous systems are power averaging.
Also; increasing clock frequency is a problem when distributing the clock over a large area without skew, when communicating between components

Analogue integrated NN are often taught using chip-in-the-loop training; instead of downloading a set of predetermined weights, each chip is trained by applying a input pattern and computing network error on the basis of the target values and the actual chip outputs, and adjusting the weights on the chip according to the learning algorithm in such a way that the network error decreases.
Another solution is to implement learning algorithms in HW. This has some advantages;

- parallelism; learning is a computationally heavy task, typically $\mathbf{O}(N^4)$ or $\mathbf{O}(N^6)$, where N is number of neurons, and it has even greater importance to utilize inherent parallelism in the learning algorithm than the NN
- Adaptivity. The system is taught while it's being used
- Asynchronous
- Fault tolerant
- Data conversion; the learning algorithm needs access to inputs/outputs/intermediate variables of the NN. If NN is analogue, the use of analogue HW for the learning algorithm eliminates AD/DA.

Analog solutions usually compute the inner product as a current sum, but they will differ in how the weights are stored, with suggested solutions as resistors, CCDs, capacitors, and (EEP) ROMs (floating gate) [7].

The analogue integrated neural networks are considered to be interesting, but are still in a more experimental state. Implementations has been limiting to small-scale networks, and it is still unproven that they will scale well to larger networks.

The field programmable learning array (FPLA) is a mixed signal counterpart to the all digital FPGA in that it enables rapid prototyping of algorithms in HW. Unlike the FPGA, the FPLA is targeted directly for machine learning by providing local, parallel, online analog learning using floating gate MOS synapse transistors.
HW implementations of learning algorithms can realize significant increase in performance; both in terms of speed and power consumption compared to standard computers, but it is naturally dependent of design technique. Many machine learning architectures and NN map easily to VLSI because of the use of many simple parallel elements, and the computing is done using only local information; this justifies the use of a silicon die with millions of transistors inherently in parallel.

Analog and mixed signals VLSI are often plagued by mismatch (and offsets) in devices, and increased accuracy is often synonymous with increased power consumption and die

area. However floating gate transistors can overcome these intrinsic accuracy limitations. An FPLA user interface consists of a design compilation and configuration tool, and a digital and analog I/O chip interface. The first enables one to extract and compile an algorithm (or parts of it) to an FPLA configuration, the second connects with surrounding digital circuitry and analog devices, for example sensors.
The prototype chip is a 2x2 array of programmable learning blocks (PLBs), consisting of two pFETS and two nFETs. The system comprises digital inputs for programming and bi-directional I/O for system operation, and has a size of 2 mm x 0.7 mm, which later has been reduced with 50%.

Digital chips generally lower computational speeds than the analogue, values varying from 3 MCPS to 1.28 GCPS [7]. Solutions regarding processors vary from one processor for each synapse, to one per neuron, or even one processor for many neurons. The most common solution is to use one processor per neuron. Weights are stored in shift registers, latches or memory; typically 1, 2, or 3 transistor DRAM or 4 or 6 transistor SRAM. Some hybrids exist, where weights are stored digitally, and the inner product is computes as a current sum.

Some existing digital implementations will be presented in the following.

A fully digital general-purpose digital neurochip uses 16 bit weights stored in an on-chip RAM, and supports on-chip learning, but exports the sigmoid function. The inner product is performed using a bit serial technique. Weights are fetched in parallel out of RAM, so RAM access time can be 1/8 of processor cycle time. Activations are 8 bits. The technology is 1.6 micron CMOS. One new activation is completed every 2 microseconds, i.e. 25 MHz clock cycle, and 32 16 bit weights in parallel on every cycle, equals ca 800 MCPS.

A SIMD presented in [7] general purpose multiprocessor architecture called CNAPS id implemented in 0.8 micron CMOS. High performance, with 1.6 GCPS inner product, 1.28 GCPS and 300 MCUPS (cell updates per second), and 12.8 GCPS using one bit weights. The synaptic capacity is 2M one bits, 256 K 8 bits, 128 K 16 bits, distributed among 64 processors. The chip consist of 64 processors, 32 bit instruction bus, 8 bit global output bus, 8 bit global input bus, 4 bit inter-processor bus. The processors include 4 k bytes weight memory implemented using 4-transistor SRAM, a memory base address unit, 32 16 bit registers, I/O, 8 x 16 bit multiplier, 32 bit saturating adder, and a logic- and shift unit. The die size is $\sim 5\,cm^2$, power < 4 W per chip, scalable, it has learning-on-chip, built-in redundancy (64/80 processors need to work), runs at 25 MHz. The chip requires an external instruction sequencer.

Several properties of the fuzzy ART facilitate the implementation in hardware; among them the lack of need to do multiplication at each synapse, the algorithm performs well with as few as 4 bits of weight precision, and also very little circuit area is needed at each synapse. However, since the synapses are bi-directional, and also the flow of weight values; some effort may be put into deciding the sequencing of the operations of the algorithm.

Implementations tested on datasets of 20000 patterns involving a SIMD massively parallel machine, demonstrates that the fuzzy ARTMAP can take advantage of parallel processing. The performance of the classifier in simulations can be expressed in terms of compression, memory and convergence, in addition to the accuracy.
Compression is the averaged ratio of training patterns to committed F2 layer nodes. Memory is described as the number of normalized registers used to store the set of learned prototype vectors, where the size is decided by what is sufficient to store real values of weights, vigilance parameter, etc. Convergence time is the time it takes for the classifier to converge, i.e. recognize presented input patterns. Some simulations have shown that memory use for the proposed system is typically less than 1000 registers; 10 % of memory used by similar classifiers, in an environment consisting of radar pulses from 15 emitters, with 17 different modes.

Some preprocessing are necessary before presenting the pattern vectors to the ART's; the first of these to stages are a normalization stage where an M-dimensional input pattern is transformed to a vector $\mathbf{a}=(a_1,a_2,\ldots a_{Ma})$, where every component lies in the interval [0,1]. The second preprocessing stage produces an output vector **I**, that includes the complement vector of **a**; $\mathbf{I} = (\mathbf{a},\mathbf{a}^c)$. The complement coding allows the recognition system to encode in its memory representation features that are consistently absent, as well as those that are consistently present.

Many high level neural systems have complex internal structures, but often based on a small number of ART-like building blocks. These solutions that realize the neural processing system are a necessity when used in real-time applications, since the dependency of software in real world applications such as robots, satellites, and portables is unreliable to some extent.
A suggested solution [8] for a functional real-time clustering microchip neural engine is based on ART 1 architecture, but with a more VLSI friendly algorithmic structure. The initial unimproved architecture clustered 100 pixels into 18 categories, and did the classification and updating of its weights in less than 1.8 microseconds. It also allowed for a modular expansion, where an M x N array clustered N x 100 pixels into M x 18 categories. But implementations had high area consumptions with high costs; with an area of $1\,cm^2$. In addition its yield was about 6 %. The dysfunctional chips performed satisfactorily mostly because of the fault-tolerant nature of the algorithm.
A common solution to the low yield is to build in redundancy and self testing subsystems that seek out the faulty subcells and disconnect them (used in commercial DRAMs), but in this case this solution increases silicon area, cost, and processing circuitry.
The most area consuming elements of the initial prototype was an array of several thousand current sources, which had to match within precision of 1 %.

An ART1 chip was designed and fabricated which clustered 50 binary input patterns in up to 10 categories, with a yield of ca 98 %, and an area less than 15 times the initial prototype. This chip was then used to implement some multi-chip systems; a two-chip ART 1, and a three-chip ARTMAP.

ART 1 is a self- organizing neural associative memory capable of generating (unsupervised) stable recognition codes in response to a series of arbitrarily- many,- ordered and –complex binary input patterns.
The chip is analog in nature, but its inputs and outputs are digital.

There are some specific structures that are useful when designing digital NN arithmetic elements, memory, and noise sources.

- Logic design styles; often the highest speed logic is the one that consumes the most power, so trade-offs between must be made between speed, power, and area. CPL (complementary pass transistor) is suggested.
- Latches/clocking; static or pseudo-static latches, non-overlapping two-phase or single-phase clocking
- The 4:2 adders implements an efficient, compact accumulator; it interfaces cleanly to standard 2's complement
- Memory; DRAM; highest density, but consumes more power; the refreshing needed in DRAMs could be substantial in very large networks. SRAM consumes less power. Shift registers can often be implemented with D or S-RAM, saving area and power.
- Noise sources are important in a wide variety of NN (especially when stochastic)

A SP (one processor per synapse) solution has the lowest synaptic storage density, but the highest computational throughput.
A NP (one processor per neuron) solution has weights stored in memory local to each processor, and has characteristics similar to the X1/CNAPS chip.
A FP (fixed number of processors on chip, and off-chip weights) solution has the lowest throughput, but the highest synaptic storage density and virtually unlimited capacity. The number of processors on a FP is I/O limited. Assuming 4 bit weights, 128 pins would be required to support 32 processors.

Discussion

In a dense environment, the interarrival time between successive pulses at the ESM receiver is a random variable distributed according to a negative exponential distribution with parameter L. This is equal to the sum of the pulse repetition frequencies of all active emitters in the instantaneous view of the receiver. The mean value of the interarrival time for a negative exponential distribution is $\frac{1}{L}$ and L is therefore also the average arrival rate of radar pulses. For a given time interval, the probability that *n* pulses arrive at the ESM receiver during *t* is from [9]

(0.19) $$\frac{P(t) = e^{-Lt} * (L * t)^n}{n!}$$

This means that the arrival rate of pulses is directly proportional to the sum of the PRFs of the active emitters illuminating the system.
It is clear that there will be a difference between the theoretical distribution and actual measured distribution, and that it varies as a function of the pulse arrival rate.

The time inside the receiver is defined as the service time *Ts*. The time needed to assign an incoming PDW to a pulse train is defined as *Ta*, and the pulse rate is *PR*.

For the Euclid project, an estimated 5 million pulses are visible every second, and with a pulse buffer of 10 ms, a single pulse buffer will contain 50 000 pulses ideally, i.e. if every pulses was recorded.

In a dense environment, the TDOA between successive pulses at the ESM receiver input is a random variable distributed according to a negative exponential with parameter *PR*. The PDF of the TDOA is

(0.20) $$f_T(t) = PR * e^{-PR*t}$$

The service time is a fixed value, chosen to be the maximum expected PW among the pulses. Maximum PW is chosen to be 2 µs. The fraction of pulses that the receiver is able to process or the fraction of pulse completion is then the fraction of the arriving pulses that are separated in time by *Ts* or more. This value is denoted by *FP*, and is given by

(0.21) $$FP = \int_{Ts}^{\infty} f_T(t) \partial t = e^{-PR*TS}$$

This can be thought of as for a given time interval, the number of PDWs emerging from the receiver is divided by the number of pulses arriving at the receiver input. The

analytical values given by equation (0.21) are shown in figures 35, 36 and 37 for varying expected PWs.
It is clear that *FP* decreases exponentially with the product *PR*Ts* according to equation(0.21). For the given arrival rate; as *Ts* increases, the fraction of pulse completion *FP* decreases.

The service time inside the ESM receiver and its distribution is generally dependent on what pulse parameters are measured and the service discipline of the receiver. The service discipline in short describes the amount of time before the processing of a new pulse starts. A paralyzable service discipline begins processing a new pulse after a fixed time τ. This could be the mean or the maximum PW of the arriving pulses, and for the simulations an expected maximum is utilized. A non-paralyzable service discipline is ready to process a new pulse as soon as the previous pulse is expired, and the service time is equal to the pulse width. For overlapping pulses, the service time is the minimum of the resultant width and certain maximum permissible value.
The service time also varies as a function of the diversity of the received pulses.

Table 13

Pulse arrival rate	Processed pulses
1000000	135335,2832
2000000	36631,27778
3000000	7436,25653
4000000	1341,850512
5000000	226,9996488
6000000	36,86527412

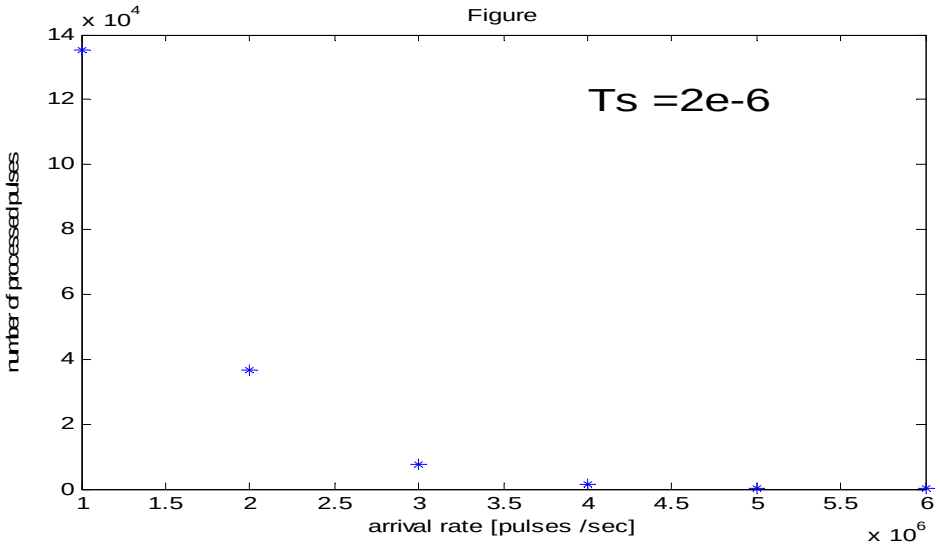


Figure 35

Table 14

Pulse arrival rate	Processed pulses
1000000	367879,4
2000000	270670,6
3000000	149361,2
4000000	73262,56
5000000	33689,73

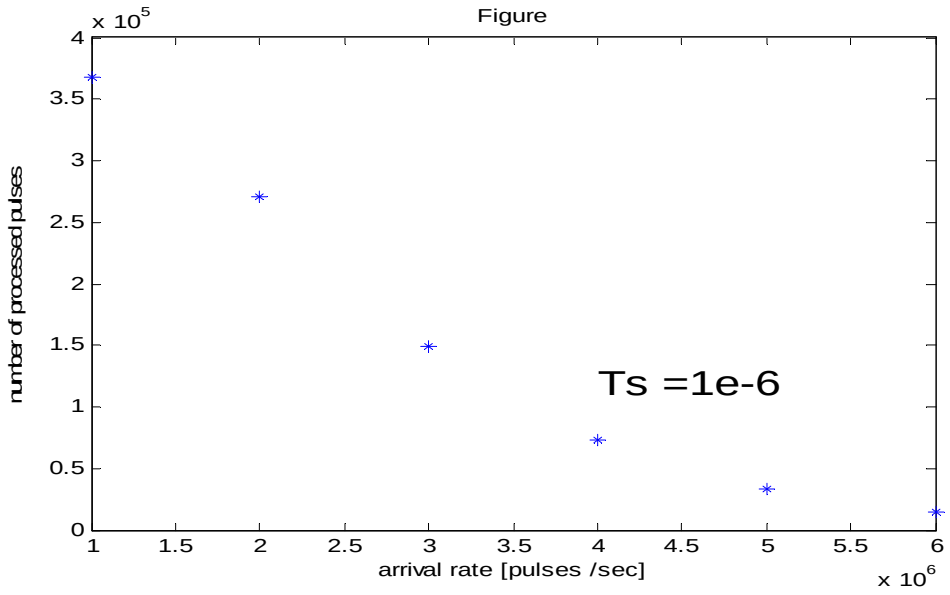


Figure 36

Table 15

Pulse arrival rate	Processed pulses
1000000	406569,7
2000000	330597,8
3000000	201616,5
4000000	109294,9
5000000	55544,98
6000000	27099,49

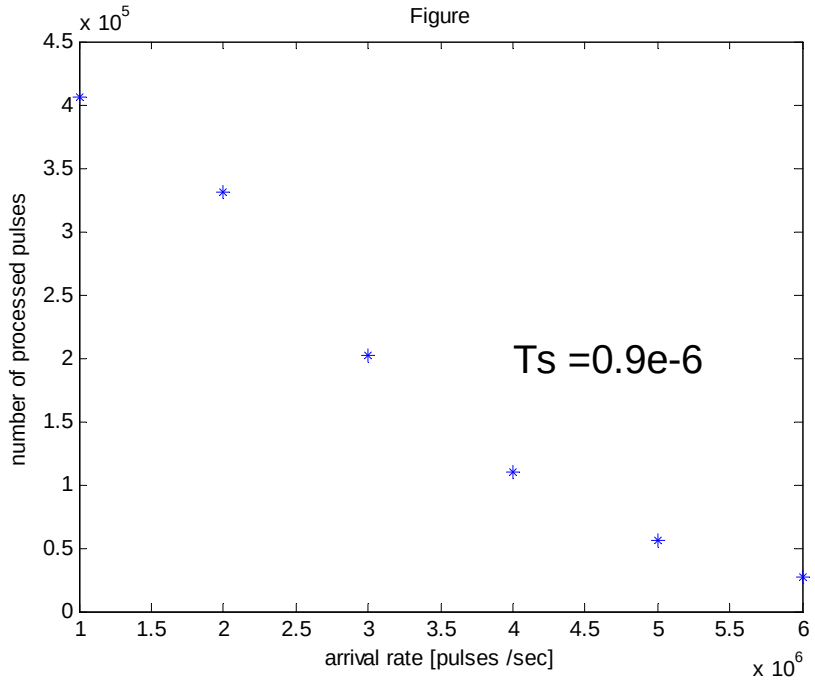
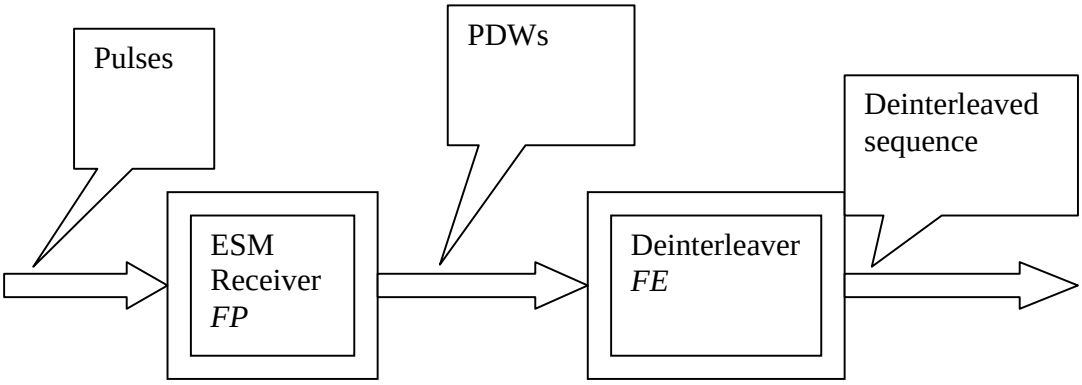


Figure 37

The figures illustrate the number of processed pulses based on equation(0.21), with the exact values presented in tables 13, 14 and 15. The number of processed pulses varies significantly as a function of the expected maximum PW.
Because the service time isn't a constant, but is distributed as an Erlang function, the actual values will vary from those shown in the figures. Typically this would reduce the amount of successfully processed pulses even more.



When sorting pulses on TOA, the common way to solve this is to allow groups of pulses into a buffer. The size of the buffer could typically be 128 pulses, or the number of recorded pulses in the last 10 ms.
This number is dependent on several factors within the ESM system.
The PDW flow from the receiver to the deinterleaver is most efficient when the deinterleaver module is preceded by a buffer. This is suggested to be at 256 pulses in [10]. This varies as the maximum PW and PR, and it seems low, but the pulse sorting is generally easier the smaller the buffer. Also; depending on the search algorithm, the size of the buffer is essential both in terms of speed, and if it is desirable to find emitters with complex PRF patterns. If the size is too small, it is not possible to fit all the pulses from emitters with high level stagger into it.

For simplicity it is assumed that the instant the deinterleaver completes servicing the n^{th} PDW, is the instant when the deinterleaver is ready to service the $(n+1)^{th}$ PDWs.
This may not be the case in real-time. When the buffer is filled or a given time threshold is reached, the deinterleaver ought to be done with the processing of the previous loaded pulse buffer. If this isn't the case, congestion occurs, inducing a bottleneck. For a real-time implementation it could be advantageous to utilize two or even several buffers. In this way the process of transferring the detected pulses to the deinterleaving module doesn't need to be perfectly in sync with the deinterleaver.

A sequential search algorithm, which is considered to be one of the simplest solutions, is basically to find three pulses with the same PRI, and then to find and remove all the pulses with that PRI. However this algorithm is not very flexible when it comes to jitter or high level stagger. Problems would also arise if pulse trains with high PRFs are mixed with low PRFs, which is often the case for ESM systems.
The same problem occurs for the histogramming techniques when applied directly on a pulse train, and in the case of a dense environment, some kind of preprocessing would be an advantage. This could be doing a rough sorting based on direction and frequency, where the dominating DOA and RF are classified with mean and standard deviation; grouping them into classes, and then extract the pulses in the different classes.

Simulations performed in [11], where both the CDIF and TOA histogram were implemented in a high level language, show successful performance on several test sets. The number of instructions (the low level microprocessor instructions required to analyze the samples), was between 14 000 and 22 000. From the simulations it was clear that the TOA difference histogram needs fewer instructions, but has a significantly poorer performance.
Because of few false PRI values for which sequential search should be performed, the SDIF have more correct detected emitters in a reduced time of analysis. In addition; only low level differences has to be calculated, and this reduces the number of histograms. With directional analysis performed first, SDIF is done on a smaller group of pulses from the input- buffer. This means that SDIF histogramming and the following sequential search is much faster. Simulation shows that the improvement of pre-clustering on DOA (and on RF after TOA), compared to just TOA increases with more complex environment.

The main advantage of using PRF spectra for detection of pulse trains, in addition to the relative low number of computations, is that even if several pulses are missing, the spectra do still occur. Threshold detection is being used, where a dominant/maximum PRF is located, and the pulses with this PRI are extracted. Several pulse trains will naturally give several tops in the spectra. Another advantage of the algorithm is that the spectra doesn't change significantly when jitter occurs.

None of the time domain algorithms performs well when implemented as the first part in an ESM chain, but should rather operate on pulses that are presorted, on DOA, RF or other parameters. This implies that the system divides the received pulses in directional bins, and defines clusters as a function of the internal differences in frequency. The optimal choice of inter-frequency division threshold must be made with that in mind that pulses from frequency agile radars might end up in to different RF bins if the threshold is too small. At the same time; if the threshold is too high, the clusters may end up excessively large.
The RF is discarded as a second sorting parameter because of RF agile radars. These would form large pulse clusters in the DOA-RF domain, and since the correlation between pulses are lost; PRI analysis could be almost impossible.
Some precaution must be taken when using TOA algorithms, in particular when using sequential search, in a high density environment, when the algorithm can extract pulses belonging to different emitters, if they have close TOAs. Using two-dimensional sequential search, with an additional parameter as PW, would improve the results [3].

Regardless of which domain is used to analyze the pulse trains, after a PRF-detection, the corresponding pulses are removed from the buffer, allowing other low-PRF pulse trains to be more easily recognized, both because the high PRFs mix in with the lower PRFs creating subharmonics, and the lower PRFs more often utilize jitter.
When a PRI is detected, the pulses in the buffer are removed. Since the techniques operate under non-ideal cases, the pulses might not appear at the projected intervals. This could be solved by predefining a standard deviation for which pulses are said to belong to the estimated PRF. Another possible solution is to find a triplet, and then utilize a Kalman

filter to track the emission. This way the standard deviation between measured and estimated pulse train parameters are tracked, which could describe on the jitter, if any. The search for the next pulse is done in a two step process, where the first step is to find the best pulse which has the best match with the predicted one, inside the ± 3 standard deviation interval. This corresponds to a 95 % chance of finding the next pulse, where 50 % is considered to be a minimum for a robust algorithm. After the next pulse is found, the Kalman filter is updated with new mean values of PRI, amplitude and PW, their covariance matrixes, and time-stepped one forward.

The time needed to assign a PDW to a pulse train is as mentioned dependent on the algorithm. If the PRF is known, the search could be reduced to N operations, where N is the number of pulses.
In general, the sequential triplet search is of order N^3; and if it is possible to detect PRI/PRF before every pulse in a pulse train is identified and removed from the buffer, the number of computations can be reduced to N. Instead of detecting the PRIs with a histogram method, the techniques realized in the frequency domain is of order N****log***(N), compared with N^2 for the histograms.

Identification/recognition can be done in many different ways, and at many different levels. A simple solution could be to associate similar emissions with the intention of reducing data, and then do a data correlation with a database. This has a disadvantage as emitters that aren't present in the database will not give positive recognition results. The next step would then be to merge different emissions to see if patterns are periodic; merge multiple PRFs to a MPRF mode. A PRI from one emitter to another can be very similar, but PRFs will vary significantly.
An alternative to correlation of PRIs is to compute the PRFs very accurately, and then utilize that the PRF of an emitter is always derived from the crystal-oscillator frequency, which makes it possible to do SEI. A radar's *N* PRFs will all be derived from

$$PRF_n = \frac{f_c}{k_n} \text{ where } k_n \in \mathbb{N} \tag{0.22}$$

This technique requires the PRF to be computed very accurately, and the $f_{crystal}$ is not deductible from a singular PRF as seen from(0.22), because the factor k_n is unknown.

The method will give recognition with a high confidence level, but the technique could be unsuitable for real-time satellite implementations because the complexity associated with the pattern recognition increases drastically in multiple emitter scenarios. But when the PRF is found, a new EDW from the pulse sorting module could easily be correlated with a track of the $f_{crystal}$, with the prerequisite that the measured PRF is a subharmonic of $f_{crystal}$.

Considering that the measured values for PW and RF in the PDWs will be more or less the same in each satellite, they need not necessarily be derived in every satellite. However, since the overall system concept calls for all satellites to have full functionality

and to be interchangeable, it is understood that all satellites implement full pulse sorting and identification capability.
Even tough the TAO for each pulse will be different for each satellite; the resulting PRI values associated with each emitter will also be more or less the same.
The values for parameters DOA and PA will not to the same degree have similar values for each satellite, and the expected values will depend of satellite configuration.

Table 16 is a suggested PDW from [10]

Table 16

PARAMETERS	BITS
TOA	32
Estimated background	14
Channel number	8
Phase angel, start	12
Start frequency	12
Pulse length	16
# phase shifts	8
# pulse drops	8
Max amplitude	14
Average/mean amplitude	14
Average/mean frequency	12
TOTAL	150

The ideal number of bits representing a PDW is difficult to derive. But generally will the number of bits used to generate a PDW determine the required comparison time inside the deinterleaver. This means that the longer the length of the PDW, the higher the probability that the received pulses are blocked by the deinterleaver. At the same time, if the number of bits is reduced, the resolution and accuracy of the measurements degrades. This means that to maintain a certain performance for a given pulse rate, PDW size and deinterleaving algorithm, the blocking probability will be reduced if the processor performance is increased or a bigger buffer is utilized.

Figure 38 displays the relative difference in bit flow from receiver as the number of bits in the PDW varies. The expected values are taken from tables 13, 14 and 15 with the estimated 5 million pulses arriving each second.

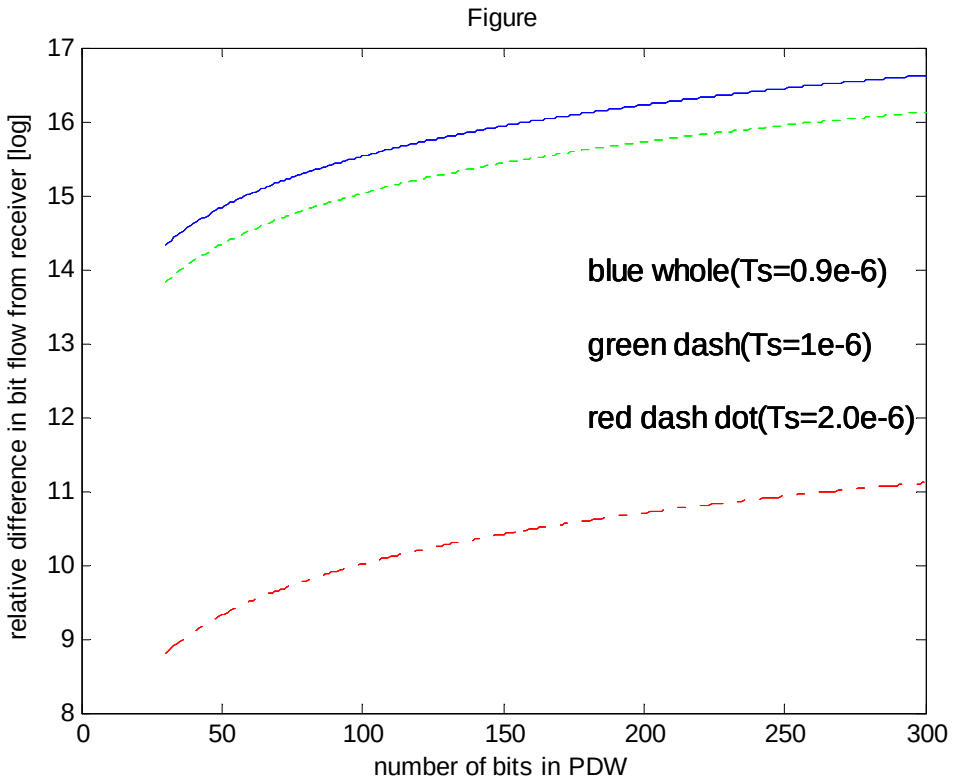


Figure 38

There are many neural network algorithms that have been studied, most of which have been implemented in software. When choosing for a HW or SW solution, one must be aware of that for many applications where real-time processing is necessary, and the size of the computational system have limitations on size, some type of special-purpose HW will be advantageous. Specifically will the analog circuits' high-speed computations with relatively low area and power consumption be an alternative to the digital implementations. However digital VLSI implementations have some advantages; high precision, ease of information storage, ease of sequencing and multiplexing, ease of interfacing and design, and high reliability, which are some of the reasons of the success this field have achieved during the last decades in building all types of computing machines. Together with the excessive area consumption, the digital implementations also have some processing delays associated with complex operations like multiplication, which are extensively used in neural networks.

There are some problems in connections with the implementation of NN to HW.

- the systems may not be scalable due to the size of many primary components, such as multiplexers and D/A's
- non general problem solving, inflexible
- often not comprise complete solutions, i.e. external weight updates
- it is a difficult, time-consuming and error-prone design process

A digital VLSI can be used to implement both very high- and very large scale networks efficiently, but there exists a trade-off between performance and flexibility. The limit of the implementable network depends on the energy per computation and synaptic storage density, and also the complexity of the synaptic interaction and the neuron model.
1 micrometer technology is considered an old technology, where 0.5, .35, .18 and .13 and further are the newest technologies. Newer technologies often lead to problems related to the design, where older circuit designs have to be redesigned to utilize the technology. The advantages include more electronics fit on less area, i.e. more IC per wafer, smaller transistors induce smaller parasitic capacitances giving a faster circuit, and less charge needed. Some disadvantages are the cost, bigger leak currents, and more static power consumption. It could impose problems when the system runs on battery supply.

The choice of technology must be considered in relation to the aforementioned, and is somewhat reliant on the field of use. For example in image sensors in APS, where capacitors is charged to a certain level, and then discharged when illuminated, it is important to avoid large leak currents since long exposure times the capacitance is discharged with the leak current and not as a function of illumination. This is why for high end image circuits used in medicine or space related applications still use 'old' technology.

Conclusions

This thesis has evaluated a selection of TOA deinterleaving algorithms with the intent to clearly discriminate between pulses emitted from agile emitters.
The first section presents the different techniques, with emphasis on pinpointing the different algorithmic structures.
The second section presents a neural network combinational recognition system, with a main focus on the fuzzy ARTMAP neural network, where also some practical implementations has been presented.
The final section gives a partial system evaluation based on some statistical means, seeking to get an estimate on the information flow from the ESM receiver as a function of both the density and the expected parametric values, i.e. PW since this is proportional to the amount of processed pulses.

The algorithms differ in performance when operating directly on dense and complex pulse environments. This is especially true for the histogram techniques, the frequency domain techniques would be a more efficient solution to the kind of scenario as in Euclid, especially as a first or second line of processing. Sequential search algorithms are efficient only for a small number of PRIs, with high-quality data when implemented as a primary deinterleaver. In addition, it will not function well on complex PRIs. A reliance on the instantaneous parameters such as DOA and possibly RF must be made prior to the TOA deinterleaving because of the difficulty involved with sorting the agile pulses. As the RF is found to be an unstable parameter also in terms of agility, opens for a sole DOA sorting prior to deinterleaving. The sequential search is however found to be adequate as a secondary algorithm, when PRIs are detected in the frequency domain.

The expected intercepted pulses are assumed to be in the order of 5 million. Only a fraction of these will be detected in each satellite, which means that the PDWs must be compressed in some way. If the recognitory task is implemented in satellites, the PDWs are reduced to EDWs, which typically reduces the amount of data 100-10000 times. This recognitory task of producing EDWs involves complex signal processing, and an alternative to traditional accumulation of responses from EDWs could be to produce directional description words. This could consist of a network pulse sorter prior to DL, to separate the different emitters from one another. Also a count or a sequential evidence accumulator could be implemented at the end of the chain.

Any HW implementation has to optimize three constraints; accuracy, processing speed and space. An analogue solution is efficient in chip area and speed, but this comes at a cost of a limited accuracy of the network components due to noise. Also; the field is considered to be in experimental state. For a digital implementation the quantization of the network parameters and the weight storage is the main limitations.

The demonstrated network performs as expected, with successful classification when presented with pulse parameters in the vicinity of the training pattern. But as the network is evaluated on ideal noise-free parameters, the limitations lie within these chosen parameters, as they will vary both as a function of the complexity of the emitter, and as a

function of the accuracy of the receiver. This is especially true for the PW, which is considered to be the most unreliable parameter because of time overlapping of radar pulses in very dense environments, and also because of multipath effects that distorts the pulse shape.

The proposed algorithm omits the traditionally multiple-parameter deinterleaving techniques in a way that merges the measurements with the aid of a NN, and not by other metric techniques. However, as the deduction of the individual PRIs of the interleaved sequence is the final process in when trying to deduce an emitter by its EDW makes it difficult in seeing the gain for complex emitter environment. This however implies a presorting, possibly presorted on a rough and fine RF, histogramming in frequency bins, further sorting on DOA, and then sent PRI analysis.
The advantage of the NN is that the PDW is automatically assigned to its corresponding emitter without any computational delay other than the delay associated with the propagation thru the network, which is considered to be fractional. Also; since the directional parameters are presented accordingly as the PDWs are presented to the network, a sequential accumulation of responses occur.

The successfully received pulses are assumed to be the only pulses that are completely received. At very high arrival rates, the ratio of successfully processed pulses will be low. Possibly solutions to this could be to implement several receiver-encoding channels, or to limit the measurements of the monopulse parameters to those that can be obtained from the leading edge of the pulse, and imposing a paralyzable discipline. To get a complete description of the environment, it could be advantageous to utilize some kind of time-slotting for when the satellites receives pulses, and the comparing results. This ensures that as much of the total scenario is recorded, even though the total information is distributed amongst several ESM receivers, i.e. in the different satellites.

References

[1] Phase center Localization, Preliminary Technical report 1, Hjelmstad (2005)
[2] Prosjektoppgave, Satellite Cluster Consepts, Bildøy, B-E (2005)
[3] Improved algorithm for the deinterleaving of radar pulses, Milojevic,D.J (1992)
[4] The pulse sorting transform, Overman, Mix, Lookadoo, Proc.IEEE (1990)
[5] Digital Elevation Mapping using Interferometric SAR, Aanvik,F., UNIK, I-1997.001-H (1997)
[6] Hardware Learning in Analogue VLSI Neural Networks, Lehman,T., Technological University of Denmark (1994)
[7] Digital Neural Network Implementations, Burr,J.B, Department of Electrical Engineering, Stanford University (1995)
[8] An ART1 Microchip and Its Use in Multi-ART1 Systems, Serrano-Gotarredona, Linares-Barranco, IEEE Trans. (1997)
[9] Analysis of Queuing Behavior of Automatic ESM systems, El-Ayadi, El-Barbary, Abou-Bakr, IEEE Trans. (2001)
[10] Pulssortering og emitteridentifikasjon, Malnes, FFI/RAPPORT, (2000)
[11] New techniques for the deinterleaving of repetitive sequences, Mardia, IEEE Proc. (1989)
[12] Radar Systems and Analysis and Design Using Matlab, Mahafza, Chapman & Hall/CSC (2005)
[13] Neural Networks, Haykin, Prentice Hall (1999)