ICESTARS

[✓ **Public**]

Confidential

Project:	**ICESTARS**
Project Number:	FP7/2008/ICT/214911
Work Package:	WP2
Task:	T2.4
Deliverable:	D2.8 (V1.1)

Title:	Public report on numerical improvements of the VoHB algorithm – Description of a new Convolution algorithm
Author(s):	T.Rahkonen, J-P.Hamina, J. Aikio
Affiliation(s):	University of Oulu, Finland
Contact:	timo@icestars.eu, janne@icestars.eu
Date:	30-Sept-2010

Table of Contents

1 Introduction

This deliverable is related to a detailed distortion analysis algorithm called Volterra on Harmonic Balance (VoHB). It breaks the total nonlinear distortion given by harmonic balance to a vector sum of smaller components so that the dominant causes, mixing and cancelling mechanisms can be easily seen. This is done so that for each nonlinear element, a polynomial presentation is first fitted in the frequency domain. These polynomial expansions are then used to calculate different mixing mechanisms. Both for the fitting and the expansion, frequency domain convolution is needed.

VoHB calculates and plots the distortion coming from each nonlinear component separately. In addition, it is also capable of showing various mixing mechanisms between harmonic bands. For this purpose, the following procedure is followed: first, the polynomial model is fitted for each nonlinear circuit element. This is done to obtain the band-to-band frequency conversion gains. The fitting is performed in the frequency domain using convolution to obtain higher-order spectra that are needed for fitting. Then, the terms of the polynomial expansion are handled separately as phasors, so that different order of products can be recognized. While the convolution is a linear operation, superposition applies, and mixing results from different harmonic bands can be easily separated. For example, 2nd-degree mixing results at the fundamental band can be separated to products mixing from baseband and 2nd harmonic. This ability of steep band-selection, as well as easy handling of small delays and broadband 90 degree phase shifts occurring in capacitors is the main reason for doing this analysis in the frequency domain. [1,2]

Convolving discrete spectra has certain problems, however. For example, some implementations require all tones to be multiples of some minimum frequency spacing. Section 2 discusses the previous implementations and their limitations, while the new implementation is shown in Section 3.

The improvements possible with the presented convolution implementation are needed to extend the usability of the VoHB to more complex transmitter operations, like envelope tracking amplifiers, or envelope or 2nd harmonic injection techniques.

2 Earlier approaches and their limitations

Convolution algorithms [3] have been used in spectral balance simulators to calculate the spectral regrowth, starting from a non-distorted spectrum, and building up the distortion sidebands order by order [4]. In the VoHB algorithm, the situation is slightly different, as it uses the simulated large-signal voltage and current spectra of a harmonic balance simulation. Hence, the input voltage spectrum of a nonlinear element is usually already distorted, and further convolutions highly increase the order of the output results.

The simplest solution for spectral convolution is to use something similar to Matlab's [5] conv() function. Here, it is not necessary to fill the entire spectrum with evenly spaced tones, but it suffices to have only the relevant tones within the harmonic bands, as long as the harmonic relations of the tones are maintained in the vector created. This allows the presentation of the spectrum with a fixed number of tones that are clustered on harmonic bands.

Matlab's conv() has the following (efficiency) shortcomings, however:

- It does not utilise the inherent conjugate symmetry of the spectrum of a real signal
- Some guard-bands must be padded between the harmonic bands to avoid the overlapping of the bands due to spectral regrowth. These guard bands get filled with higher-order results and are later removed, so that calculating of these is a wasted effort
- The spacing between the tones is fixed. This limits either the generality or the efficiency of the analysis of more than 3-tone signals

A straightforward implementation of spectral convolution is to build a row-by-row shifted Toeplitz matrix and multiply it with the original spectrum [6]. The Toeplitz matrix has empty corners, the calculation of which can be easily avoided. In addition, the symmetry of the spectrum has been employed by the existing implementation described in [2]. Nevertheless, this version still needs the guard bands between harmonic zones to avoid overlapping due to spectral regrowth, and does not handle nicely unevenly spaced multitones.

The main requirements for the new implementation were:

- utilise the symmetry of the spectra
- avoid unnecessary multiply-by-0
- allow efficient convolution of only the given harmonic bands
- handle efficiently general multi-tone signals with both evenly and unevenly spaced tones
- it would cope with the output of harmonic balance analysis, where tones with different indices may fall to the same physical frequency
- facilitate a means of traversing the convolution backwards to see which all tones are mixing to the given frequency

3 Description of the new convolution algorithm

3.1 *Preprocessing*

When imported from Harmonic Balance, the vectors containing frequencies and the corresponding voltages and currents are all of same length, and in same order, but they still need to be sorted by ascending frequency. Frequencies are labeled as multiples of the input tones (call indices), and also this presentation is sorted to the same order as the frequency vector. Next the conjugate values are mirrored on the negative side of the spectrum. Then the frequency vector is scanned through to see if there are indices resulting in the same physical frequencies – if found, these will be summed together. During the search for identical frequencies the borders of harmonic bands are determined, and the indices of the band edges are then stored into a vector.

3.2 *Convolution algorithm*

Here, the frequency domain convolution is based on the formula of discrete convolution:

$$\mathrm{conv}(\mathrm{U},\mathrm{V})(\mathrm{n}) = \sum_{k=-\infty}^{\infty} \mathrm{U}(\mathrm{k})\mathrm{V}(\mathrm{n}-\mathrm{k}) \,.$$

As each voltage and current spectrum already contains all the necessary higher-order frequencies, we can use the same frequency vector throughout, and also force the output signals to the same frequency grid. For example if the vector described in Table 1 is convolved with itself, the resulting vector would have the same length and same values of frequencies. The calculation of additional higher-order frequencies that would be created during convolution is omitted.

Table 1. Example vector with values, frequencies and matrix indexes.

U(-f4)	U(-f3)	U(-f2)	U(-f1)	U(DC)	U(f1)	U(f2)	U(f3)	U(f4)
-f4	-f3	-f2	-f1	DC	f1	f2	f3	f4
1	2	3	4	5	6	7	8	9

The algorithm begins convolving from the lowest frequency point (index 1 in table 1) and the corresponding frequency (-f4) is stored as OutFreq which has the same meaning as n in the convolution definition. To calculate all products at OutFreq, two pointers are initialized: LoInd at –fmax (index 1) and HiInd at DC (index (N+1)/2). The function then seeks all (LoInd, HiInd) pairs for which the condition

Frequency(LoInd) = OutFreq - Frequency (HiInd) is fulfilled (FreqVect is a vector containing the frequencies of the tones). If the condition is true, a product Spectrum1(LoInd)* Spectrum2(HiInd) is calculated and summed to the output spectrum at frequency OutFreq. In the above example, for the first multiplication Frequency(LoInd) = -f4 and OutFreq – Frequency(HiInd) = -f4 –DC = -f4, so the amplitudes at these index points need to be multiplied with each other. Hence, voltage phasors U(-f4) and U(DC) in Table 1 are multiplied and the value is stored.

LoInd is swept upwards from minimum frequency and HiInd downwards from DC, so that next LoInd is increased by 1 and HiInd is decreased by 1. Again the condition

$$\text{Frequency(LoInd)} = \text{OutFreq} - \text{Frequency (HiInd)}$$

is checked. If the condition is not true, then one of the indices needs to be changed depending on which side of the equation is larger now. If

$$\text{Freqvect(LoInd)} > \text{OutFreq}-\text{Freqvect(HiInd)}$$

is true HiInd is decremented by 1, and if the condition is false, then LoInd is increased by 1.

Convolution for one frequency value in a vector is complete when HiInd reaches 1. After this, OutFreq is set to the next higher frequency (-f3 in the presented example). Since the spectrum is two-sided the complex conjugate of the calculated value can be placed to the corresponding position on the positive frequency side. Therefore it suffices to calculate only the negative half of the spectrum.

Below is the pseudo code description of the convolution algorithm. V1 and V2 are complex, 2-sided voltage spectrums and FreqVect is the corresponding frequency vector.

```
function Convolved (V1, V2, FreqVect)
SET Flength to length of FreqVect
COMPUTE DCIndex is Flength divided by two rounded up
FOR FreqBeingCounted goes from minimum(FreqVect) TO DCIndex
   OutFreq = FreqVect(FreqBeingCounted)
   CellValue = 0
   HiInd = DC + FreqBeingCounted - 1
   LoInd = 1;
   WHILE LoInd < HiInd DO
      IF Freqvect(LoInd) = OutFreq-Freqvect(HiInd) THEN
         CellValue = CellValue+V1(LoInd)*V2(HiInd)+V1(HiInd)*V2(LoInd)
         LoInd = LoInd + 1
         HiInd = HiInd - 1
      ELSE IF Freqvect(LoInd) > OutFreq-Freqvect(HiInd) THEN
             HiInd = HiInd - 1
          ELSE
             LoInd = LoInd + 1
          ENDIF
      ENDIF
   ENDWHILE
   IF LoInd = HiInd and Freqvect(LoInd) = OutFreq -Freqvect(HiInd) THEN
      CellValue = CellValue+V1(LoInd)*V2(HiInd)
   ENDIF
Convolved(FreqBeingCounted) = CellValue
Convolved(Flength-FreqBeingCounted) = Complex conjugate of CellValue
ENDFOR
```

Figure 1. Pseudo code of the basic convolution algorithm

In practice, it is not wise to drive HiInd down to 1, as the indices of the multipliers begin to repeat same pairs again, pointing to the same frequencies from different voltage vectors. Instead, the loop is repeated only while Hind > LoInd, and half of the terms can found simply by swapping the indices to the input vectors. This was done on row

```
CellValue = CellValue+V1(LoInd)*V2(HiInd)+V1(HiInd)*V2(LoInd)
```

3.3 *Storing an index table of the multiplicands*

As the frequency vector is the same for any spectral convolution, also the index pairs (LoInd, HiInd) of the multiplicands are the same every time. Hence, processing time can be saved by storing the found frequency pairs, and using the vector of found pairs in successive convolutions – this bypasses the search performed at the first time. Building up the table is slightly slower, but all the following evaluations of convolution are speeded up by a factor of 3.5, roughly. The generation and utilization of the index table is not illustrated in the pseudo code, but in short, with the help of the index table we can now skip the search of correct frequencies and simply pick the multiplicands pointed by the index table. For each output frequency, the index table contains a list of found index pairs, the number of found pairs, and one flag signal. The flag is related to the fact, that the negative half of the spectrum may have an even or odd number of tones, and depending on this, the last tone pair found needs to be multiplied either with one or two.

3.4 *Convolution of selected frequency bands only*

When calculating band-to-band mixing, it is efficient to calculate convolution for only two bands that the user selects. The actual convolution algorithm is the same as described in the previous chapter. However, the lower and upper indexes are now set so that only certain bands are calculated. The function first calculates where the results of the two selected bands will be mixed to. There are three possibilities for this:

1. There will be two separate output bands on both the positive and negative side of the spectrum
2. There will be two separate bands on both the positive and negative side, but one of these bands will go outside the highest harmonic. Hence only one band will be stored.
3. Both results end up to the same absolute frequency value range on different sides of the frequency spectrum. In this case there will be only one result band. This case happens if one of the input bands is DC.

3.5 *Tracking backwards for contributions*

The idea of VoHB is to be able to show what tones and nonlinearities are causing the distortion in a given tone. In most cases it suffices to lump up all the mixing products coming from the 2nd harmonic, for example, and this is effectively performed using the band wise convolution function. However, we may face a need to recognize all mechanisms affecting tone compression in multi-tone signal, for example, and for this purpose, a general backtracking method has been proposed.

The proposed method utilizes the index table generated by the initial convolution function. The contributions for any given tone can be obtained so that from the index table we search all index pairs resulting to the analysed tone. If the convolved spectra are original voltage spectra, this already shows what tones pairs will end up to the analysed frequency. If the convolved spectra are higher order spectra (i.e. convolution products themselves) we continue the process recursively.

With a multi-tone signal, this tear-down analysis generates a huge number of mixing products. Similar pairs can be combined (giving them an integer multiplier), and typically, very small terms are not drawn. In most cases this technique may give an overkill amount of information, but this backtracking mechanism is systematic: one can employ it for any output tone, and for any type of input spectrum.

4 Performance metrics

The previous convolution implementation could not cope with unevenly spaced tones, so comparing the results of the new and previous implementation one was only part of the work. Verfication was mostly done by comparing the convolution result to output spectrum of an ideal polynomial nonlinearity simulated by Harmonic Balance.

After verifying the correctness of the results, also the performance of the convolution algorithm was tested using both Matlab and C implementations.. Table 2 shows the performances of different functions, using 281 tone spectra and matlab implementation of the functions.

Table 2. Performance analysis results

Convolve & generate index LUT	0.0783s
Convolve using index LUT	0.0134s
Bandwise convolution	0.0094s
Combine tones at same phys. frequency	0.0031s
Convolution without index matrix creation	0.0466s

Building up the index matrix takes roughly twice the time of stand-alone convolution, but then the succeeding convolutions are faster by a factor of 3.48. Hence, the use of the index matrix is already beneficial if we need to calculate three or more convolutions – that is, in any practical analysis.

The following tables give an impression of the relative speed of convolution, when implemented in C and run within APLAC simulator. Tables 3 and show the time spent in convolution of evenly and unevenly spaced 4-tone spectra, and clearly point out the benefit of combining all the tones landing on the same physical spectrum. Table 5 shows kind of minimum situation with 2 tones and only 3rd-order HB.

Table 3. Convolution performance for evenly spaced 4-tone (nharm = 4, nTones = 4).

2000 HB analyses	17,25 s
20000 HB analyses + convolutions	20,89 s
Estimated time for one convolution	(20,89 s-17,25 s)/20000 = 182 µs

Table 4. Convolution performance for unevenly spaced 4-tone (nharm = 4, nTones = 4)

2000 HB	16,59 s
20000 HB + convolutions	37,97 s
Estimated time for one convolution	(37,97 s-16,59 s)/20000 = 1069 µs

Table 5. Convolution performance for 2-tone (nharm = 3, nTones = 2)

2000 HB analyses	6,55 s
20000 HB + convolutions	6,85 s
Estimated time for one convolution	(6,85 s-6,55 s)/20000 = 15 µs

The processing time requirements for other functions are minor compared to the convolution function, so their performance analysis is not presented.

5 Summary

A new convolution algorithm was implemented. It is faster than the previous one, and – as a major improvement – it does not require that the tones are placed on an even grid (i.e., all fi = ki*df) , which makes it possible to use all kind of multitone excitations. However, analysis of evenly spaced input signals is still much more efficient, as many distortion tones emerge on the same physical frequencies, and can be combined before calculation of the convolutions.

6 References

[1] Janne P. Aikio Frequency domain model fitting and Volterra analysis implemented on tops of harmonic balance simulation. PhD dissertation in University of Oulu, Oulu, Finland. Available at http://herkules.oulu.fi/isbn9789514284205/
[2] J.P. Aikio, T. Rahkonen *Detailed distortion analysis technique based on simulated large-signal voltage and current spectra*. IEEE Trans. Microwave Theory and Tech., vol 53, no.10, October 2005, pp. 3057-3066.
[3] http://en.wikipedia.org/wiki/Convolution
[4] C-R. Chang, M.B. Steer, G.W. Rhyne Frequency-domain spectral balance using the arithmetic operator method. IEEE Trans. Microwave Theory and Tech., vol 38, no.8, August 1990, pp. 1139-1143.
[5] http://mathworks.com
[6] http://en.wikipedia.org/wiki/Toeplitz_matrix