

ANALYSIS OF LIMIT CYCLES BY MEANS OF AFFINE ARITHMETIC COMPUTER-AIDED TESTS

J.A. López, G. Caffarena, C. Carreras and O. Nieto-Taladriz

Departamento de Ingeniería Electrónica, Universidad Politécnica de Madrid
C/. Ciudad Univesitaria, s/n, 28040, Madrid, Spain (Europe)
phone: +34 915495700 X-547, fax: +34 913367323,
email: {juanant | gabriel | carreras | nieto}@die.upm.es

ABSTRACT

Modern analysis of the finite word-length properties of DSP algorithms is typically carried out using CAD tools. The objectives of this paper are threefold: (i) to show that affine arithmetic is specially well-suited for quantization analysis of digital filters; (ii) to present a new tool based on multiple-precision affine arithmetic elements that allows fast characterization of the signal ranges in linear systems; and (iii) to develop an exhaustive-search algorithm that detects limit cycles and requires less computation time than existing approaches. Final comparative results confirm the validity of the limit cycles detection algorithm on second-order shift and delta-operator realizations.

1. INTRODUCTION

Finite Word-Length (FWL) analysis of digital filter structures has been extensively studied in the past decades [1]-[9]. These studies are traditionally based on theoretical developments, and they typically consider three types of quantization effects [1],[2]: coefficient quantization noise (CQN), roundoff noise (RON) and limit cycles (LC). The first type is a deterministic effect, so all the parameters of the filter can be recalculated after quantization. The second type is a non-deterministic effect, but it is still linear, so the output of the filter vanishes whenever a zero input is applied. The third type is a non-linear effect, and self-sustained oscillations may appear depending on the selected filter structure [1]. Traditional studies of limit cycles provide guaranteed stability regions, i.e. regions where absence of limit cycles is guaranteed. These regions are generally obtained using the Lyapunov theory [1],[3], frequency domain criteria [4] and/or other analytical methods [3]. Obviously, reformulation of the underlying formulae is required even for small changes of the realizations. On the other hand, the exhaustive search algorithm proposed in [5] provides a useful method, directly applicable to any filter structure. This algorithm has been modified in [6] to improve its computation speed, and it has been tested using a delta-operator realization. Recent literature on modern CAD tools to study the FWL properties of DSP systems can be found in [7]-[9]. They typically compare the simulation of a high-precision version of a DSP algorithm to a quantized one, and provide the corresponding error statistics. Obviously, these approaches

can also be applied to non-linear DSP algorithms, but they are not prepared to detect non-linear quantization effects, such as limit cycles. Another disadvantage of these approaches is that simulations must be run on a sample-by-sample basis, so the computation time can be substantially reduced by means of interval [10] or affine arithmetic [11].

Affine arithmetic is an extension of interval arithmetic that incorporates the source and signed amplitude of all the perturbations, or noise terms [11], of all the intervals. The mathematical expression of a given affine form \hat{x} is

$$\hat{x} = x_0 + \sum_{k=1}^n x_k \cdot \varepsilon_k, \quad \text{with } -1 \leq \varepsilon_k \leq 1$$

where x_0 is the central value, and ε_k and x_k are the k -th noise term identifier and amplitude, respectively. Its main advantage is that noise terms sharing the same identifier can operate among themselves, thus alleviating the cancellation problem. Its main disadvantage is that they cannot fully characterize non-linear operations so, in general, new noise sources are successively required [11]. As linear operations are covered exactly, this drawback does not apply to filters and linear systems. Affine arithmetic simulations of linear DSP systems have been previously presented in [12], but they mainly focus on transforms and FIR filters. IIR filters are only slightly introduced.

Additional literature on the set of tools used here can be found in [13]-[16]. Data types and quantization characteristics are introduced in [13],[14]. Feedback loops are incorporated in [15], and comparative simulations between CQN and RON are carried out for different word-length sets. The capabilities of the interval simulations to detect balanced realizations are demonstrated in [16].

The novel features of this paper are (i) the integration of the affine arithmetic class into the computation model and (ii) the implementation of a new exhaustive-search algorithm based on affine arithmetic to efficiently detect limit cycles. Its structure is as follows: The capabilities of the set of tools and the simulation advantages on linear DSP systems are presented in section 2. Section 3 explains the affine-arithmetic-based limit cycles detection algorithm. In section 4, this algorithm is applied to a representative set of filter structures and the simulation results are given. Finally, section 5 presents the conclusions of this work.

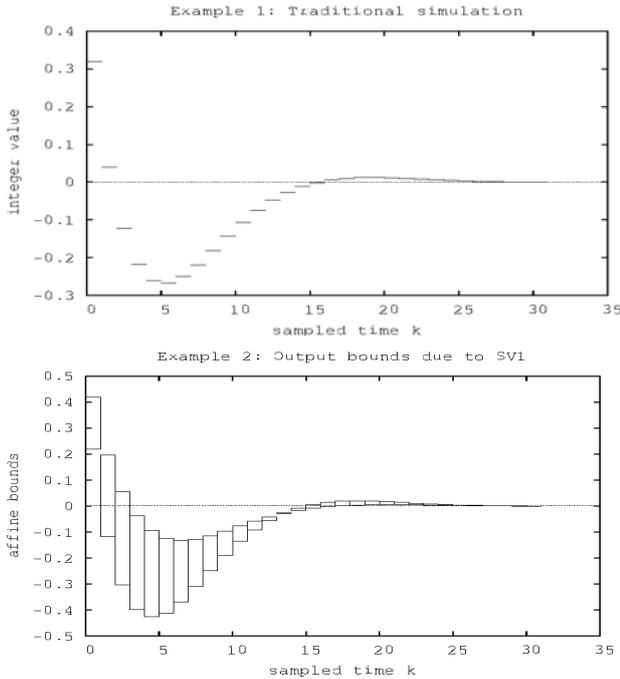


Figure 1: Simulation examples: (a) traditional simulation (b) output bounds due to a limited uncertainty in one of the state variables.

2. THE AFFINE-ARITHMETIC-BASED TOOL

The Abaco package [13] is a computer-aided set of tools based on multi-precision elements to efficiently estimate a given degree of reliability on a target system. Its utilities include: writing to an intermediate language, reading from it, specifying the signal word-lengths, applying a given quantization overflow (two's complement, saturation, wrap-around) or underflow characteristic (magnitude truncation or roundoff), executing the description of the algorithm, providing the execution statistics, plotting the results, etc.

The intermediate language supports a range of data types (numbers, intervals, domains and affines). Probability values can be associated to any of them. Moreover, as all the elements are based on the multi-precision library [17], different word-lengths and/or data types can be combined in the simulations. Therefore, unlike the approach proposed in [12], (i) simulations are not limited to any number of bits, and (ii) quantization is carried out as a separate task, independent from the structure of the data elements. In the case of affine arithmetic, this means that there is no need to increase the number of noise terms, so provided results are more accurate in this case.

A detailed description of the affine arithmetic procedures and their general advantages is given in [11], although filters -and linear systems in general- are described using only affine operations (i.e., addition, subtraction and constant multiplication). Therefore, these systems do not require any increment of the number of noise terms.

An example of the traditional simulation of a second-order realization when its state variables are initially non-zero is shown in Figure 1.a. Figure 1.b shows the affine simulation

Step 1: Create one affine element \hat{x}_i for each state variable SV_i . All noise terms are initialised to zero, except noise terms ε_{i_i} , whose amplitude must cover all possible values of SV_i (any initial orientation -or sign- of these terms are valid).

Step 2: Create a list L containing the combinations of affines \hat{x}_i to be tested for limit cycles. Initialise the affine counter c to zero.

Step 3: Simulate $k \cdot N$ time steps (parameter k is user-defined).

Step 4: If the sum of widths of all \hat{x}_i decreases in magnitude, update list L and go back to step 3.

Step 5: If \hat{x}_c contains more than one value, split it in a affines and insert the combinations in list L . Increase counter c modulo N , and go back to step 3.

Step 6: Increase counter c modulo N until \hat{x}_c contains more than one value, and go back to step 5. If all \hat{x}_i have only one value, test this combination for limit cycles and present the results.

Step 7: Take the next combination from L and go back to step 3. Repeat until the list is empty.

Figure 2: Limit Cycles detection algorithm

of the same realization when there is a bounded uncertainty in one of the state variables. It can be observed that the plot describes all the possible values that the output can take for the given uncertainty.

3. LIMIT CYCLES DETECTION ALGORITHM

The limit cycles detection algorithm is based on the work by P. Bauer [5], where an exhaustive search of the state space is performed over a limited number of clock periods. However, affine elements can group several values in a single simulation. This fact provides two important advantages: first, it increases the simulation speed; and second, the algorithm is easily directed to detect the maximum amplitude limit cycles in the first place.

Given an N -order digital filter realization, the basic search algorithm is shown in figure 2. This algorithm is based in grouping as many samples as possible into a single affine, instead of an exhaustive search of all the possible combinations. This property significantly reduces the order of complexity of the algorithm and the execution time of the simulations.

A large value of parameter a provides smaller affines, but more simulations are required to verify absence of limit cycles. A small value requires fewer simulations, but affines are larger and there is a greater probability of finding limit cycles in the following simulation. If a is even, at least two affines always contain values around zero, and if a is odd, a single affine bounds the zero neighbourhood. Although we haven't studied the optimum value of parameter a yet, we have used values around $a = 3$ in the simulations of section 4.

The optimum value of parameter k is related to the speed of convergency [16] of the simulation. If k is large, the algorithm performs more simulations and fewer tests for limit cycles.

This option is preferable when non-linearities have little effect on the affine simulation response. On the contrary, a small value of parameter k is preferred whenever the affine simulation response is noticeably affected by them. In addition, if the poles of the filter are far from the unit circle (i.e. the filter response tends rapidly towards zero), it is preferable to assign k a small value and vice-versa. In all the simulations of section 4, this parameter has been set to 1.

The previous algorithm can be further refined according to the order in which affine combinations are successively split and inserted in the search list. For example, to detect maximum amplitude limit cycles, subdivisions containing the largest values are inserted first in the list; or to detect limit cycles with maximum SV_i amplitude, affine \hat{x}_i is always divided in first place.

One of the main aspects of the basic detection algorithm is affine subdivision. Although the algorithm is based on affines and affine simulations, state variables are initialised by assigning ‘interval + identifier’ pairs to each of them. Moreover, simulations provide interval-bounded responses containing all possible signal outputs. Therefore, input and output specifications are based on intervals, while computations are based on affines because of their linear memory of original intervals.

The two conditions to perform affine subdivision are: (i) both interval and affine descriptions must cover the same range of results, and (ii) the memory of the noise terms (i.e. its signed amplitude) must be preserved as much as possible. Given $\hat{x} = 0 + 7\epsilon_1 - 9\epsilon_2$, the following example shows the splitting procedure: (i) affine \hat{x} represents interval $I = [-16, 16]$, containing 33 values, and it can be represented using 3 affines of 5 noise terms (11 values) each; (ii) original noise terms are proportionally distributed to each affine, and remaining samples are individually tested for limit cycles; and (iii) central values are adjusted. Therefore, the final result of this example is: $\hat{x}_1 = 11 + 2\epsilon_1 - 3\epsilon_2$, $\hat{x}_2 = 0 + 2\epsilon_1 - 3\epsilon_2$, and $\hat{x}_3 = -11 + 2\epsilon_1 - 3\epsilon_2$.

4. RESULTS AND DISCUSSION

This section presents the comparative simulation results of our approach and the traditional exhaustive search tests for limit cycles over a set of realizations. The coefficients of the z -domain denominator polynomials, $D(z) = 1 + a_1z^{-1} + a_2z^{-2}$, correspond to the sections of the narrowband fourth-order elliptic filter of [6]. All the signals of the realizations are quantized as specified in the tables. The maximum amplitudes of the limit cycles at the output, the state variables that generate these oscillations and their corresponding oscillation period are also given. The final rows provide the number of individual simulations (N_{sim}) of the split and the exhaustive search parts of the algorithm, the computation times (*time*) and the number of exhaustive search tests (*tests*) of our approach and the exhaustive search algorithm of [5] and [6]. In all cases, simulations have been performed using the Abaco set of tools, and the computation

TABLE I: PERFORMANCE COMPARISON BETWEEN THE AFFINE-ARITHMETIC-BASED AND THE EXHAUSTIVE SEARCH ALGORITHMS. (FIRST SECTION, REALIZED IN TRANSPOSED DIRECT FORM)

Quantization	Roundoff		
	6	7	8
N. bits	6	7	8
a_1	- 1.914		
a_2	0.95		
$\max(y_{LC})$	- 2	- 0.625	- 0.3125
sv_1	- 2	- 0.625	- 0.3125
sv_2	2	0.625	0.3125
<i>period</i>	1	1	1
N_{sim} (<i>split</i>)	76	520	2041
N_{sim} (<i>search</i>)	1387	7179	25353
<i>time</i> (<i>split</i>)	0.22 s.	1.44 s.	5.72 s.
<i>time</i> (<i>search</i>)	1.23 s.	7.05 s.	35.28 s.
<i>tests</i>	288	1088	3148
N_{sim} [4],[5]	34350	170000	---
<i>time</i> [4],[5]	48.85 s.	639.69 s.	---
<i>tests</i> [4],[5]	4096	16384	65536

times evaluated on a SUN Workstation with a 233 MHz processor and 256 MB of memory.

Table I presents the results when the previously explained methods are applied to the first section of the elliptic filter realized on transposed direct form. Word-lengths are successively increased from 6 to 8 bits, and roundoff is always selected as the underflow strategy. It is straightforward to notice that the split procedure requires a small percentage of the number of simulations and achieves a significant reduction in the number of tests and the corresponding computation time, of almost two orders of magnitude. In addition, the differences between the two approaches increase when more complex structures are compared.

Table II presents the results of the two algorithms when they are applied to the second section of the elliptic filter of [6]. It can be seen that there is a significant improvement in the performance of the proposed algorithm, even with respect to the previous case. The reason for this is that the split procedure achieves the bounds of the limit cycles in only a few simulations.

Figure 3.a shows the simulation response of the quantized filter when the input is set to [-1,1] at $t = 0$, and 0 otherwise. Using this selection, the designer is able to model the effect of any possible input appearing at $t = 0$ in only one simulation. The figure also shows that the quantized response of the filter does not vanish -as it does in the ideal case-, and self-sustained bounds appear, due to the non-linear effects of the quantizers. Figure 3.b shows the evolution of the maximum amplitude limit cycles of the second section of the elliptic filter when all the signals are implemented using 8 bits. It must be said that the correspondence between the maximum sample value and the maximum amplitude of the limit cycle is not bi-univocal, but we have observed that this type of simulation provides a very fast and accurate approximation of these bounds.

TABLE II: PERFORMANCE COMPARISON BETWEEN THE AFFINE-ARITHMETIC-BASED AND THE EXHAUSTIVE SEARCH ALGORITHMS. (SECOND SECTION, REALIZED IN TRANPOSED DIRECT FORM)

Quantization	Roundoff		
	N. bits	6	7
a_1	-1.828		
a_2	0.843		
$max(y_{LC})$	2	-0.5	0.25
sv_1	2	-0.4375	0.25
sv_2	-2	0.4375	-0.21875
<i>period</i>	1	8	7
$N_{sim} (split)$	5	5	5
$N_{sim} (search)$	12	72	66
<i>time (split)</i>	0 s.	0 s.	0 s.
<i>time (search)</i>	0.01 s.	0.05 s.	0.04 s.
<i>tests</i>	1	1	1
$N_{sim} [4],[5]$	3142	1469573	---
<i>time [4],[5]</i>	2.17 s.	1105.14 s.	---
<i>tests [4],[5]</i>	4096	16384	65536

5. CONCLUSION

In this paper we have presented three main contributions related to the application of affine arithmetic to linear DSP systems. First, we have introduced the Abaco set of tools and, particularly, the affine arithmetic class. Second, we have demonstrated that affine arithmetic is especially well suited to model the behaviour of linear DSP realizations, such as IIR filters. Finally, we have proposed a new algorithm based on affine computations to efficiently detect and bound limit cycles. The final results clearly indicate that the application of this method significantly reduces the number of simulations and the computation time with respect to traditional approaches.

REFERENCES

[1] R. A. Roberts and C. T. Mullis, *Digital signal processing*. Reading, Mass.: Addison-Wesley, 1987.

[2] K. K. Parhi, *VLSI digital signal processing systems: design and implementation*. New York: Wiley, 1999.

[3] K. Erickson and A. Michel, "Stability analysis of fixed-point digital filters using computer generated Lyapunov functions- Part I: Direct form and coupled form filters," *Circuits and Systems, IEEE Trans. on*, vol. 32, pp. 113-132, 1985.

[4] T. Laakso, B. Zeng, I. Hartimo, and Y. Neuvo, "Elimination of limit cycles in floating-point implementations of direct-form recursive digital filters," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Trans. on*, vol. 41, pp. 308-313, 1994.

[5] P. H. Bauer and L.-J. Leclerc, "A computer-aided test for the absence of limit cycles in fixed-point digital filters," *Signal Processing, IEEE Trans. on*, vol. 39, pp. 2400-2410, 1991.

[6] J. Kauraniemi, "Analysis of limit cycles in the direct form delta operator structure by computer-aided test," presented at Acoustics, Speech, and Signal Processing, 1997. ICASSP-97, 1997 IEEE Int. Conf. on, 1997.

[7] S. Kim, K.-I. Kum, and W. Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Trans. on*, vol. 45, pp. 1455-1464, 1998.

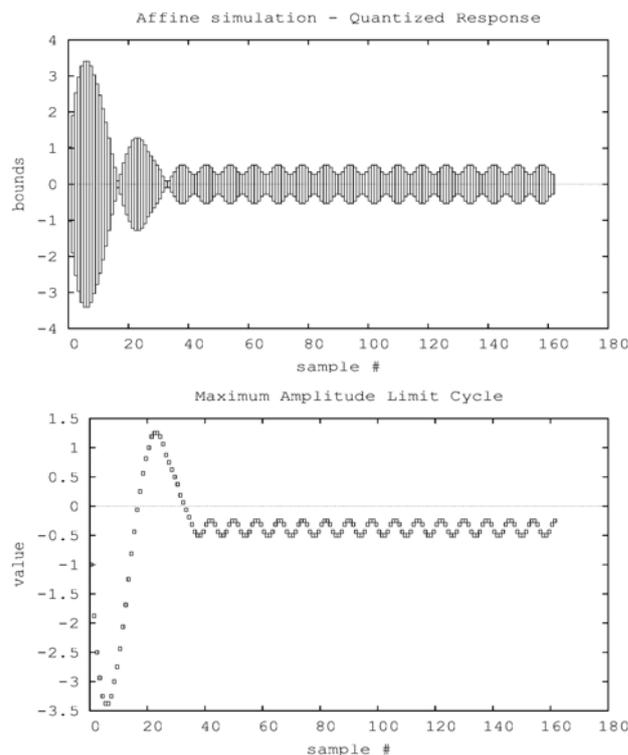


Figure 3: Detection of limit cycles using affine arithmetic simulations: (a) complete search space (b) simulation of the sample that generates maximum amplitude limit cycles.

[8] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens, "A methodology and design environment for DSP ASIC fixed point refinement," presented at Design, Automation and Test in Europe Conference and Exhibition 1999. Proc., 1999.

[9] H. Keding, M. Coors, O. Luthje, and H. Meyr, "Fast bit-true simulation," presented at Design Autom. Conf., 2001. Proc., 2001.

[10] R. E. Moore, *Interval analysis*. Englewood Cliffs, N.J.: Prentice-Hall, 1966.

[11] J. Stolfi and L. H. d. Figueiredo, "Self-validated numerical methods and applications," presented at 21st Brazilian Math. Colloquium, IMPA, Rio de Janeiro, Brazil, July, 1997.

[12] C. F. Fang, T. Chen, and R. A. Rutenbar, "Floating-point error analysis based on affine arithmetic," presented at Acoustics, Speech, and Signal Processing, 2003. Proc. (ICASSP '03). 2003 IEEE Int. Conf. on, 2003.

[13] C. Carreras, J. A. Lopez, and O. Nieto-Taladriz, "Bit-width selection for data-path implementations," presented at System Synthesis, 1999. Proc. 12th Int. Symp. on, 1999.

[14] C. Carreras and I. D. Walker, "Interval methods for fault-tree analysis in robotics," *Reliability, IEEE Trans. on*, vol. 50, pp. 3-11, 2001.

[15] J. A. Lopez, C. Carreras, G. Caffarena, and O. Nieto-Taladriz, "Fast characterization of the noise bounds derived from coefficient and signal quantization," presented at Circuits and Systems, 2003. ISCAS '03. Proc. of the 2003 Int. Symp. on, Bangkok, Thailand, 2003.

[16] J. A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Characterization of the quantization properties of similarity-related DSP structures by means of interval simulations," presented at Asilomar Conf. on Signals, Systems and Computers, Pacific Grove, CA, 2003.

[17] The GNU Project, "The GNU Multiprecision Arithmetic Library," <http://www.swox.com/gmp/>.