Analysis of the LNS Implementation of the Fast Affine Projection algorithms

Felix Albu[1], Jiri Kadlec[2], Antonin Hermanek[2], Anthony Fagan[1], Nick Coleman[3]

[1]Dept. of EE, UCD, Dublin, IRELAND
felix@ee.ucd.ie

[2]Dept. of Signal Proc., UTIA, PRAGUE, CZECH REPUBLIC
kadlec@utia.cas.cz

[3]Dept. of EE, Univ. of Newcastle, UNITED KINGDOM
j.n.coleman@ncl.ac.uk

Abstract -- **It has been shown that a 32-bit logarithmic arithmetic unit can operate faster than, and maintain the accuracy of, a 32-bit floating point unit. It uses the logarithmic number system (LNS), in which a real number is represented as a fixed-point logarithm. In this paper we investigate the implementation of three fast affine projection (FAP) algorithms using LNS. We propose a simplified Conjugate Gradient FAP (SCGFAP) algorithm. We show that the 32-bit or 20-bit LNS implementation of the CGFAP and the SCGFAP algorithm are superior to those of the classical FAP algorithm.**

Keywords – **Conjugate Gradient, Fast Affine Projection, Logarithmic Number system.**

I INTRODUCTION

The problem of unwanted echoes arises wherever the microphone picks up the unwanted signals such as the signals radiated by the loudspeakers or other reflections of the speech signal via different echo paths. The reverberation time is typically on the order of a few hundred milliseconds. Thus, the need for an adaptive echo-cancellation filter (ECF) with a long impulse response (hundreds of coefficients) is evident. It can be placed in parallel to the loudspeaker-enclosure-microphone system (LEM system) [2]. The well-known normalised LMS (NLMS) algorithm has been widely used but it has slow asymptotic convergence. The affine projection algorithm (APA) [3] can be considered as a generalisation of the NLMS algorithm. However, its fast version [4], when implemented with an embedded FRLS (Fast Recursive Least Squares) algorithm suffers from numerical instability [4-5]. Other difficulties associated are memory requirements and code overhead. Because of these disadvantages using an FRLS procedure instead of standard RLS procedure does not necessarily represent the most economical solution. An alternative approach for the FAP (Fast Affine Projection) algorithm has been proposed [6]. This proposed algorithm has fewer equations and is attractive when the projection order is small. It is a fast version of APA and reduces its complexity to $2L+5N^2+9N$. Another form of the standard FAP that uses a sliding-window RLS type approach has been proposed in [7-8]. The complexity of the real version is $2L+3N^2+8N$. This algorithm only uses 2 divisions per iteration instead of the 2N divisions per iteration proposed in [6]. These alternative FAP algorithms lead to a more accurate estimation of the auto-correlation matrix inverse because a conventional RLS algorithm is used. All methods have no feedback incorporated. If the estimations deviate from the accurate value, the errors propagate to the next iterations, causing the adaptive filter to fail sometimes. A frequently proposed remedy is to re-start periodically a new inversion process. Even so, sometimes the numerical errors accumulate so fast that the re-starting period would have to be made very small. It’s difficult to foresee the unhealthy signs of instability for the numerous internal variables (especially for the conventional FAP algorithm). Therefore the complexity associated with this procedure is high. Another improved FAP algorithm using the conjugate gradient (CG) method to do the matrix inversion was proposed in [9]. It was called CGFAP and it was proved that it is stable and easy to implement in comparison with other FAP algorithms. It uses a feedback scheme so that the numerical errors do not accumulate. CG is a non-linear programming method that iteratively seeks the minimum of a quadratic cost function [10]. It has been verified on floating point and fixed-point DSP platforms including 16 and 24 bits ones [9]. In this paper we study the behaviour of 20-bit or 32-bit LNS implementations of the classical FAP algorithm [4], the CGFAP algorithm [9] and our proposed SCGFAP algorithm. This conjugate gradient FAP algorithm (CGFAP) is presented in the following lines [9]:

Initialisation $$\underline{V}(-1)=\underline{0}, \underline{\eta}(-1)=0, \underline{s}(-1)=0, \quad \mathbf{R}(-1)=\delta\mathbf{I}, \alpha=1, \underline{P}(-1)=\underline{b}/\delta \qquad (1)$$

Processing in sampling interval *n*

$$\mathbf{R}(n)=\mathbf{R}(n-1)+\underline{\xi}(n)\underline{\xi}^T(n)-\underline{\xi}(n-L)\underline{\xi}^T(n-L) \qquad (2)$$

$$\underline{g}(n)=\mathbf{R}(n)\underline{P}(N-1)-\underline{b}\ ; \qquad (3)$$

$$\gamma(n)=\frac{\underline{g}^T(n)\mathbf{R}(n-1)\underline{s}(n-1)}{\underline{s}^T(n-1)\mathbf{R}(n-1)\underline{s}(n-1)}; \qquad (4)$$

$$\underline{s}(n)=\gamma(n)\underline{s}(n-1)-\underline{g}(n); \qquad (5)$$

$$\underline{P}(n)=\underline{P}(n-1)-\frac{\underline{g}^T(n)\underline{s}(n)}{\underline{s}^T(n)\mathbf{R}(n)\underline{s}(n)}\underline{s}(n); \qquad (6)$$

$$\underline{V}(n)=\underline{V}(n-1)+\alpha\eta_{N-1}(N-1)\underline{X}(n-N) \qquad (7)$$

$$y(n)=\underline{V}^T(n)\underline{X}(n)+\alpha\overline{\underline{\eta}}^T(n-1)\underline{\tilde{R}}(n); \qquad (8)$$

$$e(n)=d(n)-y(n); \qquad (9)$$

$$\underline{\varepsilon}=e(n)\underline{P}(n); \qquad (10)$$

$$\underline{\eta}(n)=\begin{bmatrix}0\\ \overline{\underline{\eta}}(n-1)\end{bmatrix}+\underline{\varepsilon}(n) \qquad (11)$$

Table 1. Conjugate gradient FAP algorithm

Where $\underline{b}$ is a N vector with only one non-zero element, which is unity at the top, $\mathbf{I}$ is an NxN identity matrix, δ is a regularization factor that prevents the input auto-correlation matrix $\mathbf{R}(n)$ from becoming ill-conditioned. $\underline{\tilde{R}}(n)$ is an N-1 vector that consist of the N-1 lower-most elements of the N vector $\underline{R}(n)$, which is the left column of $\mathbf{R}(n)$. $\overline{\underline{\eta}}(n)$ is an N-1 vector that consist of the N-1 upper-most elements of the N vector $\underline{\eta}(n)$, the scalar $\eta_{N-1}(n)$ is the lower-most element of $\underline{\eta}(n)$, $x(n)$ is the input signal and $d(n)$ is the desired output signal. There is a wide acceptable range for the regularised factor that prevents the input auto-correlation matrix from becoming ill-conditioned.

II. REAL TIME REQUIREMENTS

The total computational cost for the CGFAP algorithm is $2L+2N^2+9N+1$ MACs and 1 division [9]. As an iterative method, CG approaches $\underline{P}(n)$ with a delay and this tracking error isn't a problem since $\mathbf{R}(n)$ varies at a slower rate because $L >> N$ as explained in [9]. This delay allows us not to update $\underline{P}(n)$ every sample. The conjugate gradient method is summarised by equations 3-6. Our simulations will show that we can update $\underline{P}(n)$ every second up to forth sample without seriously affecting the output error. Therefore, the average number of MACs and divisions is *p* times smaller for CG section ($2L+2N^2/p+(4+5/p)N-1+2/p$ MACs and $1/p$ divisions, where *p* depends on *L* or *N* values, usually between 2 and 4). For example, with L=1000 and N=5, an NLMS needs 2025 MACs (25 for a division), the CGFAP needs 2121 MACs, while the SCGFAP with p=2 needs 2070 MACs and SCGFAP with p=4 needs about 2045 MACs. Therefore, in this case, the increase is about 1 % for SGCFAP with *p*=4 in comparison with NLMS. However, this percentage will increase if L is lower or N is higher. Assuming that a division is equivalent with 25 MACs SCGFAP with *p*=2 needs the least amount of computation of the three algorithms as long as $N \le 15$ and $N \le 26$ if *p*=4 (see Fig.1). The CGFAP is inferior to SCGFAP and exceeds the performance of the classical FAP algorithm for $N \le 8$.

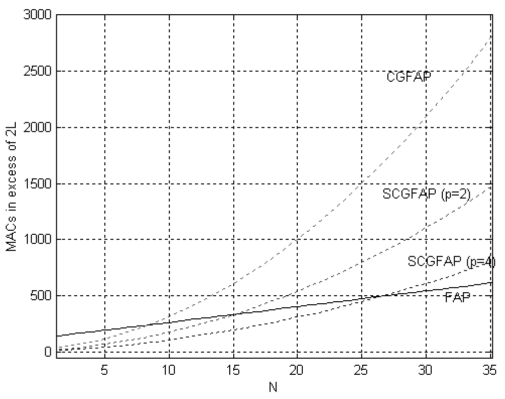

Fig. 1 Real time requirements of 3 FAPs

Although the computational effort comes in bursts, other subsequent computations can be executed in parallel with the CG iterations if an FPGA [14] or an advanced DSP is used. In this case the updating of the computation of the normalised residual echo vector may use previous values of $\underline{P}(n)$. The divisions are very time consuming for most commercial DSPs[9]. However this impediment is avoided by using an LNS processor or an FPGA based on LNS. Contemporary microprocessors perform real arithmetic using the floating-point system. Although this method has served well over the past decades, it suffers from a number of disadvantages which render it unsuitable for very high-speed computation and which inhibit its more widespread use, for example in application-specific integrated circuits or smaller microprocessor devices. Floating-point circuits are large, complex and much slower than fixed-point units; they require separate circuitry for the add/ subtract, multiply, divide, and square-root operations; and all floating-point operations are liable to a maximum half-bit rounding error. As an alternative to floating-point, the logarithmic number system offers the potential to perform real multiplication, division and square-root at fixed-point speed, and in the case of multiplication and division, with no rounding error at all. These advantages are, however, offset by the problem of performing logarithmic addition and subtraction. Hitherto, this has been slower or less accurate than floating-point, or has required very cumbersome hardware.

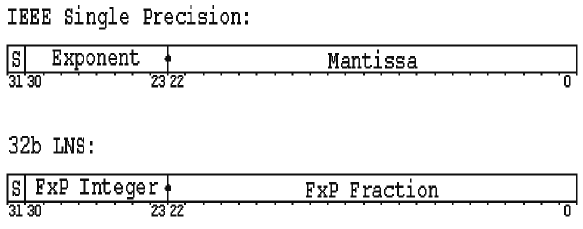

Fig.2 IEEE standard single precision floating point representation and the 32-bit LNS format

The 32-bit floating-point representation consists of a sign, 8-bit biassed exponent, and 23-bit mantissa. The LNS format is similar in structure (see Fig.2). The 'S' bit again indicates the sign of the real value represented, with the remaining bits forming a 31-bit fixed point word in which the size of the value is encoded as its base-2 logarithm in 2's complement format. Since it is not possible to represent the real value zero in the logarithmic domain, the 'spare' (most negative) code in the 2's complement fixed point part is used for this purpose, which is convenient since smaller real values are represented by more negative log-domain values. The LNS format can be thought of as a floating-point format with a fixed-point exponent and zero-length mantissa. The chosen format compares favourably against its floating-point counterpart, having greater range and slightly smaller representation error. A 20-bit LNS format is similar. It maintains the same range as the 32-bit, but has precision reduced to 11 fractional bits. This is comparable to the 16-bit formats used on commercial DSP devices. The 20-bit version requires just 10,920 bits of lookup tables. The 32-bit LNS implementation uses 321,536 bits of lookup tables. Table 1 presents the LNS arithmetic operations.

We employ a solution developed by the HSLA project team [1], which yields a drastic reduction in the size of the look-up tables required compared to those needed for conventional linear interpolation of both functions.

x + y	ADD	Lz = Lx + log(1+2^(Ly-Lx)), Sz depends on sizes of x,y
x - y	SUB	Lz = Lx + log(1-2^(Ly-Lx)), Sz depends on sizes of x,y
x * y	MUL	Lz = Lx + Ly, Sz = Sx OR Sy
x / y	DIV	Lz = Lx - Ly, Sz = Sx OR Sy
x^2	SQU	Lx << 1, Sz = Sx
x^0.5	SQRT	Lx >> 1, Sz = Sx
x^-1	RECIP	Lz = Lx, Sz = ¬Sx
x^-0.5	RSQRT	Lz = Lx >> 1, Sz = ¬Sx

Table 1. LNS Arithmetic Operations

This is achieved by the parallel evaluation of a linear approximant and an error correction term. More details about the logarithmic number system are available at http://napier.ncl.ac.uk/HSLA. We present results for an implementation based on the 32-bit and 20-bit logarithmic ALUs. Numerous simulations showed that the precision is equal to or better than the standard IEEE floating-point devices [11].

III. SIMULATIONS

In these simulations the excitation signal is amplitude normalised speech, sampled at 8 kHz, the echo path has the length *L*, the projection number is N, and the additive noise is 30 dB below the echo. The convergence of the algorithm was compared by using the squared norm of the difference between the LEM model and the adaptive filter (in dB) [2]. The parameter μ for all the FAP and NLMS algorithms was set to 1. The CGFAP algorithm performs only one division per sample. This division is not performed and zero is assigned if the denominator is not positive or lower than a specified threshold. This threshold was fixed to 10^{-10} in our simulations. The echo path represents a room impulse response and is taken form [2]. The projection order is *N*=10. Like in other adaptive algorithm simulations [11], the 32-bit FLOAT or LNS CGFAP implementations have virtually identical performances (Figs. 3-4). Both figures show some losses in performances due to a lower finite precision of 20-bit versions in comparison with their 32-bit versions. However, the CGFAP initial convergence is better than that of the NLMS algorithm and it is stable (see Figs.3-4). They show that the LNS iterative FAP implementations have faster convergence rates than the NLMS algorithm. The classical FAP algorithm uses a sliding window fast RLS algorithm that it is difficult to implement, memory intensive and potentially numerically unstable. The regularization parameter used in the FAP algorithm uses the constant k that multiplies the average power of the input signal. This parameter plays an important role in the FAP's initial convergence and stability.

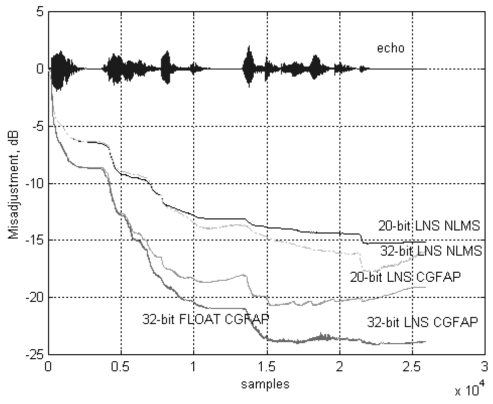

Fig. 3. The learning curves for 32-bit and 20-bit implementations of CGFAP and NLMS algorithms (32-bit FLOAT and 32-bit LNS curves almost co-incidental, L=1000, N=10)

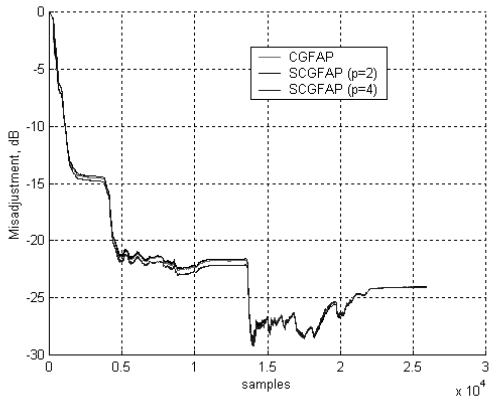

Fig. 4 Convergence of 20-bit LNS CGFAP implementation for different values of *p*

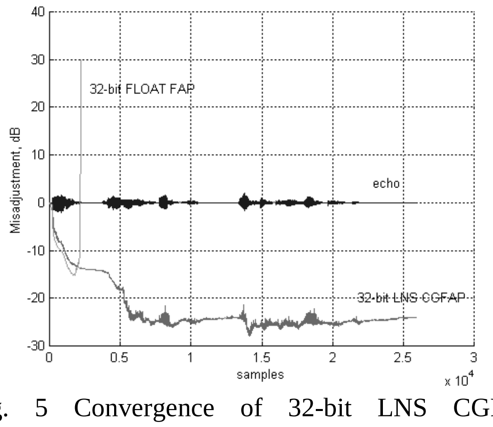

Fig. 5 Convergence of 32-bit LNS CGFAP implementation versus 32-bit FLOAT FAP implementation, Float is unstable after about 2200 iterations (L=256, N=10, k=5)

If k=5 the 32-bit FLOAT implementation of classical FAP algorithm (without the restarting procedure) is unstable after about 2200 iterations (Fig.5). Fig. 6 shows an example of unstable 20-bit LNS FAP implementation (k=100). Other examples, where the 32-bit or 20-bit FLOAT or LNS implementation of the Gay's basic FAP algorithm were unstable, always remained stable with the equivalent CGFAP or SGFAP algorithm implementations (p=2).

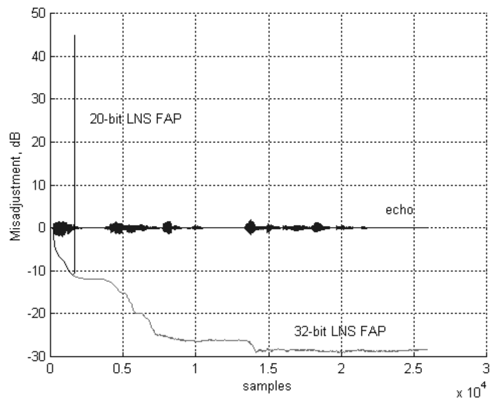

Fig. 6 Convergence of 32-bit LNS FAP implementation versus 20-bit FLOAT FAP implementation, Float is unstable after about 1600 iterations (L=256, N=10, k=100)

IV. CONCLUSIONS

These simulations show that the 32-bit or 20-bit LNS implementation of SCGFAP and CGFAP are stable, unlike the those of the classical FAP algorithm without the restarting procedure. CGFAP require fewer operations and could provide a low-cost, efficient solution for different applications. SCGFAP can be only marginally complex than NLMS and is a good candidate for implementation in voice applications. The CGFAP 20-bit LNS implementations appear to offer a very attractive alternative to conventional arithmetic if the precision its not a major issue. Our future work will be focused in implementing the SCGFAP on FPGA and examining the potential of other iterative method such as SOR or Gauss-Seidel [12,13].

V. ACKNOWLEDGEMENTS

This work has been performed under the EU ESPRIT 33544 HSLA Long-term research project (http://napier.ncl.ac.uk/HSLA), coordinated by the University of Newcastle, UK [1].

REFERENCES

[1] J.N. Coleman, E.I.Chester, 'A 32-bit Logarithmic Arithmetic Unit and Its Performance Compared to Floating-Point', *14th Symposium on Computer Arithmetic'*, Adelaide, April 1999

[2] C. Breining, P. Dreitseitel, E. Hansler, A. Mader, B. Nitsch, H. Pudeer, T. Scheirtler, G. Schmidt, and J.Tilp, ' Acoustic echo control- An application of very high order adaptive filters,' *IEEE Signal Processing Magazine*, pp. 42-69, July 1999

[3] K. Ozeki, T. Umeda, 'An adaptive Filtering Algorithm Using an Orthogonal Projection to an Affine Subspace and its Properties,' Electronics and Communications in Japan, Vol. 67-A, No.5, 1984
[4] S. Gay, S. Tavathia, 'The Fast Affine Projection Algorithm', pp. 3023–3026, *ICASSP'95 Proceedings*
[5] S. Gay, J. Benesty, editors, 'Acoustic Signal Processing for Telecommunication', Kluwer Academic Publishers, 2000
[6] Y. Kaneda, M. Tanaka, J. Kojima, 'An Adaptive Algorithm with Fast Convergence for Multi-input Sound Control', Active95, pp. 993-1004, Newport Beach, California, USA
[7] Q.G. Liu, B. Champagne, and K. C. Ho, " On the use of a modified FAP algorithm in subbands for acoustic echo cancellation," in Proc. 7th IEEE DSP Workshop, Loen, Norway, 1996, pp. 2570-2573
[8] M. Ghanassi, B. Champagne, "On the Fixed-Point Implementation of a Subband Acoustic Echo Canceler Based on a Modified FAP Algorithm", 1999 IEEE Workshop on Acoustic Echo and Noise Control, Pocono Manor, Pennsylvania, USA pp. 128-131
[9] Heping Ding, "A stable fast affine projection adaptation algorithm suitable for low-cost processors", ICAASP 2000, Turkey, pp. 360-363
[10] David Luenberger, "Linear and Non-linear Programming", 2nd Edition, Addison-Wesley, 1984
[11] J.N.Coleman, E.Chester, C.Softley and J.Kadlec, "Arithmetic on the European Logarithmic Microprocessor", IEEE Trans. Comput. Special Edition on Computer Arithmetic, July 2000, vol. 49, no. 7, pp. 702-715; and erratum October 2000, vol. 49, no. 10, p.1152.
[12] Erwin Kreyszig, 'Advanced Engineering mathematics', 7th edition, John Wiley & Sons, 1993
[13] R.Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, 'Templates for the solutions of linear systems: Building blocks for iterative methods', SIAM, 1994
[14] F. Albu, J. Kadlec, C. Softley, R. Matousek, A. Hermanek, N. Coleman, A. Fagan, "Implementation of (Normalised) RLS Lattice on Virtex", FPL2001, pp. 91-100, Belfast, UK.