Calculation of the Line Spectrum Frequencies Using the Quotient-Difference Scheme

Davor Petrinović, *Member, IEEE*

Abstract -- A new, computationally efficient technique for calculation of the Line Spectrum Frequencies (LSF) that can be applied to any order of the LPC analysis is proposed in this paper. It is based on the Quotient-Difference (Q-D) root-finding algorithm that enables simultaneous solution for all the LSFs. It is an iterative procedure that offers the tradeoff between accuracy and complexity, what is especially important for the real-time applications. To improve the convergence, a nonlinear mapping of the LSFs is also proposed. For low accuracy applications, the method is even more effective then the fast converging Newton-Rapshon method, but is at the same time exceptionally simple, has a very regular structure and requires only basic mathematical operations.

Index Terms -- Line Spectrum Frequencies, Calculation, Root-finding, Quotient-Difference Algorithm

I. Introduction

Line Spectrum Frequencies (LSF) are a very popular parameter set for representation of the LPC speech model [1] and most of the speech coders used today are based on them. LSFs are usually determined from the direct form coefficients (a-coefficients) of the LPC filter and their calculation must be performed in real time with a typical rate of 30 to 100 times per second. Computation of the LSFs is only one part of the whole coder, but a great deal of the CPU time is usually spent on this task alone. Therefore, the techniques that reduce the complexity of this procedure are very interesting. The required accuracy of the LSF calculation depends on the application. For design and for the evaluation of the coder higher accuracy is required, but for the actual coder implementation lower accuracy is sufficient. Therefore the algorithms that offer the tradeoff between the accuracy and complexity are the best choice.

So far, there have been no closed form equations for transforming the a-coefficients into the LSF representation, but several numerical procedures were proposed in literature [2],[3]. This problem is equivalent to finding the zero-crossings of the sum of cosines with integer related frequencies [2]. Using the Chebyshev polynomials, the problem can be reduced to finding the simple real roots of an ordinary polynomial in the interval [–1, 1], [4]. Several techniques are known for solving the algebraic equation and a lot of them have already been employed and tested for the LSF calculation (e.g. evaluation on a fine grid with bisection, [4], ordinary [6],[7], modified [6] and accelerated [7] Newton-Raphson method, etc).

All these techniques can be classified into one of two basic groups. The first group is based on the evaluation of a certain function on a fine grid with N points and resolving the intervals that contain the solutions, e.g. [2],[3],[4]. If no further bisection is used, then the complexity of these methods is essentially proportional to pN, where p denotes the order of the LPC analysis. The accuracy depends on N as well, but to resolve all the roots the minimal value of N is bounded (i.e. N>100 for f_s=8kHz and p=10). The advantage of these techniques is the ability to compromise complexity and accuracy and possibility of implementation using the integer arithmetic. An improved method that enables fast identification of the intervals that contain the solution is proposed in [5], but only for the 10^{th} order LPC polynomial.

On the other hand, if the floating point arithmetic is available, what is quite common for the modern DSPs, then the techniques of the second group offer further possibilities for complexity reduction, e.g. [6],[7]. Due to very fast convergence of the Newton-Raphson algorithm, high accuracy can be achieved with 4 to 6 iterations for each root. Numerical complexity of these methods is proportional to $3/4p^2 \cdot it$, where it denotes the number of iterations. However, no tradeoff between complexity and accuracy is possible, since the convergence is guaranteed only if all the roots are found with full precision.

A new technique proposed in this paper combines the advantages of both groups and employs the Quotient-Difference (Q-D) algorithm [8] for determination of the real positive roots of a polynomial. The accuracy can be adjusted by varying the number of iterations it, while the complexity is proportional to $(2p\text{-}2)it$. Due to a very regular structure of the Q-D iteration, implementation is very simple and suitable for parallel or hardware realization.

II. Quotient-difference root-finding algorithm

Practically all conventional root finding algorithms determine the roots of a polynomial in one-by-one fashion. The Quotient-Difference scheme is an exception to this rule, since all the roots are determined simultaneously using a very simple recursive algorithm [8] explained next.

Real polynomial $B(x)$ of degree n can be defined by real coefficients b_0 to b_n or by its roots x_1 to x_n as in:

$$B(x) = \sum_{k=0}^{n} b_k x^k = b_n \prod_{k=1}^{n} (x - x_k) \qquad (1)$$

The author is at the Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, CROATIA. The study was financed by the Ministry of Science and Technology of Croatia, under the project no. 036-024.

If all n roots are simple with distinct nonzero absolute values then they can be found using the quotient-difference algorithm. This is automatically satisfied for polynomials with real roots obeying the following inequality,

$$x_n > x_{n-1} > \ldots > x_2 > x_1 > 0 \tag{2}$$

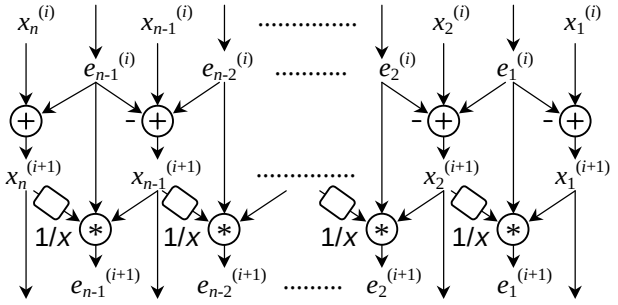

Figure 1. One iteration of quotient-difference scheme

Necessary operations required for one iteration of this procedure are graphically depicted in Fig. 1. Each iteration gives a new estimate of the root positions, based on the results of the previous iteration, and based on auxiliary variables e_1 to e_{n-1} as in (3), (iteration is denoted with i).

$$x_k^{(i+1)} = \begin{cases} x_1^{(i)} - e_1^{(i)} & k = 1 \\ x_k^{(i)} - e_k^{(i)} + e_{k-1}^{(i)} & 1 < k < n \\ x_n^{(i)} + e_{n-1}^{(i)} & k = n \end{cases} \tag{3}$$

This is the 'difference' step of the algorithm, followed by the 'quotient' step that is used for update of the auxiliary variables:

$$e_k^{(i+1)} = e_k^{(i)} \frac{x_k^{(i+1)}}{x_{k+1}^{(i+1)}} \qquad 1 \le k \le n-1 \tag{4}$$

The initial values of x_k and e_k are calculated from coefficients b_0 to b_n:

$$x_k^{(0)} = \begin{cases} -b_{n-1}/b_n & k = n \\ 0 & 1 \le k < n \end{cases} \tag{5}$$

$$e_k^{(0)} = \frac{b_{k-1}}{b_k} \qquad 1 \le k \le n-1 \tag{6}$$

If the condition (2) is satisfied, then the estimates $x_k^{(i)}$ converge to actual roots, while the auxiliary variables $e_k^{(i)}$ converge to zero, ie:

$$\lim_{i\to\infty} x_k^{(i)} = x_k \ , \qquad \lim_{i\to\infty} e_k^{(i)} = 0 \tag{7}$$

III. LSF CALCULATION USING THE Q-D SCHEME

The procedure for LSF determination starts with the inverse filter $A(z)$, that is equal to the denominator of the p^{th} order LPC model $H(z)=\sigma/A(z)$. This filter is defined by its coefficients a_1 to a_p, and can be expressed as a sum of two filters $R(z)$ and $Q(z)$ of order $p+1$, as in:

$$A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k} = \big(R(z) + Q(z)\big)/2 \tag{8}$$

where the coefficients of $Q(z)$ and $R(z)$ are symmetrical and anti-symmetrical respectively. These auxiliary polynomials are found using the following expressions:

$$\begin{aligned} R(z) &= A(z) - z^{-(p+1)} A(z^{-1}) \\ Q(z) &= A(z) + z^{-(p+1)} A(z^{-1}) \end{aligned} \tag{9}$$

If the order p is even, then one root in each of the auxiliary polynomials is known in advance and can be removed:

$$R(z) = (1 - z^{-1})R'(z), \qquad Q(z) = (1 + z^{-1})Q'(z) \tag{10}$$

Polynomials $R'(z)$ and $Q'(z)$ of the order p both have symmetrical coefficients with roots located exactly on the unit circle, i.e. :

$$\begin{aligned} R'(z) &= \prod_{k=1}^{p/2} (1 - e^{j\vartheta_{2k}} z^{-1})(1 - e^{-j\vartheta_{2k}} z^{-1}) \\ Q'(z) &= \prod_{k=1}^{p/2} (1 - e^{j\vartheta_{2k-1}} z^{-1})(1 - e^{-j\vartheta_{2k-1}} z^{-1}) \end{aligned} \tag{11}$$

LSFs to be determined are equal to the angles of these roots, ϑ_1 to ϑ_p, and for a stable LPC filter $H(z)$ satisfy the following inequality:

$$0 < \vartheta_1 < \vartheta_2 < \ldots < \vartheta_{p-1} < \vartheta_p < \pi \tag{12}$$

Before proceeding, one additional transformation of polynomials $R'(z)$ and $Q'(z)$ is necessary. The reason for this transformation is not clear at this point, but it will be explained in section IV. Since the same transformation is performed on both polynomials and since their structure is identical, the procedure will be explained for $R'(z)$ only.

Symmetrical polynomial $R'(z)$ can be defined by real coefficients r'_1 to $r'_{p/2}$ as in:

$$\begin{aligned} R'(z) = 1 + z^{-p} + r'_{p/2} z^{-p/2} + \ldots \\ + \sum_{k=1}^{p/2-1} r'_k \left(z^{-k} + z^{-(p-k)} \right) \end{aligned} \tag{13}$$

These coefficients are computed from a_1 to a_p using the following expression:

$$\left.\begin{aligned} r'_k &= a_k - a_{p+1-k} + r'_{k-1} \\ q'_k &= a_k + a_{p+1-k} - q'_{k-1} \end{aligned}\right\} \quad k = 1,2,\ldots p/2 \tag{14}$$

where $r'_0=q'_0=1$.

If a rational function of Z of the form $N(Z)/D(Z)$ is substituted for z^{-1} in the expression (13), than a new transfer function $R''(Z)$ is obtained in the new variable Z.

$$R''(Z) = R'(z)\big|_{z^{-1}=\frac{N(Z)}{D(Z)}} \tag{15}$$

where:

$$N(Z) = -\alpha + Z^{-1}, \qquad D(Z) = 1 - \alpha Z^{-1} \tag{16}$$

This is the well known nonlinear mapping of the frequency axis used in IIR filter design for changing the cutoff frequency of a prototype low-pass filter [9].

If we define a symmetric p^{th} order polynomial $W_k(Z)$ as:

$$W_k(Z) = N(Z)^k D(Z)^{p-k} + N(Z)^{p-k} D(Z)^k = w_{k,p/2} Z^{-p/2} + \sum_{l=0}^{p/2-1} w_{k,l} \left(Z^{-l} + Z^{-(p-l)} \right) \quad (17)$$

then it is easy to show that $R''(Z)$ can be expressed as:

$$R''(Z) = R''_n(Z) / D(Z)^p \quad (18)$$

where the nominator $R''_n(Z)$ is given by:

$$R''_n(Z) = W_0(Z) + \frac{r'_{p/2}}{2} W_{p/2}(Z) + \sum_{k=1}^{p/2-1} r'_k W_k(Z) \quad (19)$$

It is obvious that $R''(Z)$ becomes a transfer function of an IIR filter with p poles at $Z=\alpha$, but only the zeros defined by $R''_n(Z)$ are important, since they represent an image of the original roots of $R'(z)$. The nominator $R''_n(Z)$ is also symmetrical since it is composed of a sum of symmetrical polynomials $W_0(Z)$ to $W_{p/2}(Z)$, so can be written as:

$$R''_n(Z) = r''_{p/2} Z^{-p/2} + \sum_{k=0}^{p/2-1} r''_k \left(Z^{-k} + Z^{-(p-k)} \right) \quad (20)$$

The coefficients $r''_0, r''_1, \ldots r''_{p/2}$ can be determined directly from $r'_1, \ldots r'_{p/2}$ using the following matrix equation:

$$[\, r''_0, r''_1, \ldots r''_{p/2} \,] = [\, 1, r'_1, \ldots, r'_{p/2-1}, (r'_{p/2})/2 \,] \cdot \mathbf{W} \quad (21)$$

where $\mathbf{W}$ is the $(p/2+1)\times(p/2+1)$ matrix composed of the coefficients of $W_k(Z)$, as given in the following equation:

$$\mathbf{W} = \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,p/2} \\ \vdots & & & \vdots \\ w_{p/2,0} & w_{p/2,1} & \cdots & w_{p/2,p/2} \end{bmatrix} \quad (22)$$

Since matrix $\mathbf{W}$ depends only on the compression factor α, it can easily be pre-computed and then used for transformation with a simple matrix multiplication (21). The substitution given in (15)(16) maps the unit circle of the z-plane onto the unit circle of the Z-plane, so the roots of $R''_n(Z)$ are also located on the unit circle, but on different angular positions θ_2, θ_4, ..., θ_p. Therefore, $R''_n(Z)$ can be written as:

$$R''_n(Z) = r''_0 \prod_{k=1}^{p/2} (1 - e^{j\theta_{2k}} Z^{-1})(1 - e^{-j\theta_{2k}} Z^{-1}) \quad (23)$$

The new frequencies θ_{2k} are related to the original LSFs, ϑ_{2k}, by a simple nonlinear mapping:

$$\vartheta_{2k} = \arctan\left(\frac{(1-\alpha^2)\sin\theta_{2k}}{-2\alpha + (1+\alpha^2)\cos\theta_{2k}} \right) \quad (24)$$

In the following step of the procedure, the p^{th} order polynomial $R''_n(Z)$ with complex conjugate root pairs on the unit circle is transformed to a polynomial $C_R(y)$ of the order $p/2$ with real roots $y_1, y_2, \ldots y_{p/2}$ in the interval [-1,1]. This is the well known procedure [4] that is based on the evaluation of $R''_n(Z)$ on the unit circle ($Z=e^{j\omega}$),

$$R''_n(e^{j\omega}) = 2e^{-j\omega\frac{p}{2}} \left(\frac{r''_{p/2}}{2} + \sum_{k=0}^{p/2-1} r''_k \cos((p/2-k)\omega) \right) \quad (25)$$

and substitution $y=\cos(\omega)$, that results with $C_R(y)$:

$$C_R(y) = \sum_{k=0}^{p/2} c_{Rk} y^k \quad (26)$$

Computation of c_{R0} to $c_{Rp/2}$ from $r''_0, r''_1, \ldots r''_{p/2}$ can be done by direct expansion of Chebyshev polynomials for a chosen order p, or by an efficient iterative algorithm that works for any order (e.g. [6][7]). The final step before the application of the Q-D scheme is the expansion of $C_R(y)$ in Taylor series around $y=1$ with sign inversion, what is equivalent to substitution $x=1-y$. It results with polynomial $B_R(x)$ that has the same form as $C_R(y)$ but different coefficient values.

$$B_R(x) = \sum_{k=0}^{p/2} b_{Rk} x^k = b_{Rp/2} \prod_{k=1}^{p/2} (x - x_{2k}) \quad (27)$$

This final substitution ensures that all the roots of $B_R(x)$ are positive as required by the Q-D algorithm, satisfying the following inequality:

$$0 < x_2 < x_4 < \ldots < x_{p-2} < x_p < 2 \quad (28)$$

By combining all these transformations together, the final nonlinear mapping from x_{2k} to ϑ_{2k} is:

$$\vartheta_{2k} = \arctan_2\left(\frac{(1-\alpha^2)\sqrt{x_{2k}(2-x_{2k})}}{-2\alpha + (1+\alpha^2)(1-x_{2k})} \right) \quad (29)$$

for $k=1,2,...,p/2$, where "$\arctan_2$" denotes a four-quadrant arctangent function.

Coefficients b_{Rk} can be determined from c_{Rk} using a simple recursive algorithm that requires only summations (i denotes the iteration number, $i=1,2,...p/2$).

$$c_{Rk}^{(i)} = \begin{cases} c_{Rp/2}^{(i-1)} & k = \dfrac{p}{2} \\ c_{Rk}^{(i-1)} + c_{Rk+1}^{(i)} & k = \dfrac{p}{2}-1, \dfrac{p}{2}-2, \ldots, i, i-1 \\ c_{Rk}^{(i-1)} & k = i-2, i-3, \ldots 0 \end{cases} \quad (30)$$

with the initial iteration given by:

$$c_{Rk}^{(0)} = c_{Rk}, \qquad k = 0,1,\ldots p/2 \quad (31)$$

After the final iteration ($i=p/2$), the signs of the odd power coefficients are inverted, resulting with desired b_{Rk} :

$$b_{Rk} = (-1)^k c_{Rk}^{(p/2)}, \qquad k = 0,1,\ldots p/2 \quad (32)$$

Q-D root-finding algorithm can now be applied on $B_R(x)$ as described in section II, resulting with simultaneous solution for all roots: $x_2, x_4, .., x_p$. The whole procedure is then repeated for $Q'(z)$ and its roots: $x_1, x_3, .., x_{p-1}$.

IV. Convergence and Accuracy

A new estimates of all polynomial roots are found in each iteration of the Q-D algorithm. Since the roots are

determined simultaneously, it would be advantageous to have the same or at least similar convergence speed for all of them. Otherwise the number of iterations would be imposed by the root with the slowest convergence. In our preliminary experiments Q-D algorithm was applied without any frequency scale compression (α=0) and it was observed that the roots that are closer to 0 converge much faster then the outer ones. It was also observed that the convergence speed can be equalized if the roots are geometrically related, such that the quotient of any two successive roots is approximately constant. This has led to an idea of applying a nonlinear frequency transformation that would map the roots to new positions that are better suited for the Q-D algorithm. The compression parameter α was varied and the convergence speed was evaluated on the actual speech data. It was found that the best equalization of the convergence speed for the 10th order LPC model is obtained with α=0.6. The maximum and the mean absolute values of the LSF errors determined on a database with 120000 LSF vectors are shown in Figure 2. as a function of the number of Q-D iterations, it_1. The upper 10 curves correspond to the max. errors while the lower 10 are for the mean errors. It can be seen that for α=0.6, all the LSFs are almost grouped together, except for the first two and the last one that converge a little bit faster. Convergence is faster at the beginning of the procedure (only two iterations are needed to reduce the mean error 10 times), but it slows down for higher iteration numbers (e.g. at it_1=20, five iterations are necessary for the same reduction).

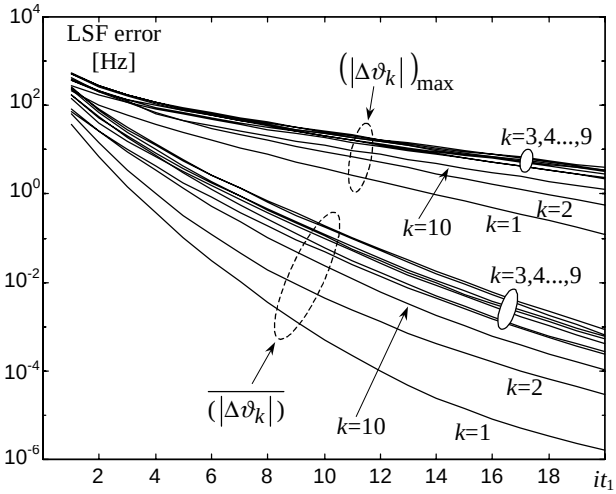

Figure 2. Maximum and mean absolute LSF errors vs. number of Q-D iterations for α=0.6

The quality of the LSF estimation was also evaluated by calculating the spectral distortion induced by inaccurate computation, as it is usually done in the evaluation of the LSF quantization. It was determined that for it_1=4 and α=0.6, the average and maximum spectral distortions are SD_{avg}=0.39dB and SD_{max}=4.01dB respectively, with only 1.4% of frames with distortion above 1dB. The accuracy of LSF estimation used in the actual coders doesn't have to be much higher then this.

V. COMPUTATIONAL COMPLEXITY

All steps involved in the proposed LSF computations are listed in Table 1, displaying the number of basic mathematical operations required for each step, where n is equal to $p/2$. Computational complexity for the Newton-Raphson method [6] is also given in the last three rows for comparison. It should be noted that the complexity of the proposed method is only linearly proportional to p, what is especially important for high order LPC analysis.

Table 1. Computational complexity of the algorithm

Equ.	Operation	Add.	Mult.	Div.
(14)	$a_k \rightarrow r'_k, q'_k$	$4n$	0	0
(22)	$\rightarrow r''_k, q''_k$	$2(n^2+n)$	$2(n^2+n)$	2
[7]	$\rightarrow c_{Rk}, c_{Qk}$	n^2-n	$2(n-1)$	2
(30)	$\rightarrow b_{Rk}, b_{Qk}$	n^2+n	0	0
(5)(6)	$x_k^{(0)}, e_k^{(0)}$	0	0	$2n$
(3)(4)	Q-D	$4(n-1)*it_1$	$2(n-1)*it_1$	$2(n-1)*it_1$
	Total	$4(n-1)*it_1$ $+(4n^2+6n)$	$2(n-1)*it_1$ $+(2n^2+4n-2)$	$2(n-1)*it_1$ $+(2n+4)$
	(n=5, it_1=4)	194	100	46
	(n=10, it_1=20)	1180	598	384
[6]	Newton-Raps	$(3n^2+n-4)*it_2$ $+(2n^2+4n-4)$	$(3n^2+n-6)*it_2$ $+(n^2-5n+4)$	$4(n-1)*it_2$ $+(2n)$
	(n=5, it_2=6)	522	448	106
	(n=10, it_2=6)	2072	1878	236

VI. CONCLUSION

A new computationally efficient algorithm for calculation of the LSFs was proposed in the paper. It was shown that with properly chosen nonlinear frequency compression, a sufficiently accurate estimation of LSFs can be obtained in only 4 to 5 iterations of the Q-D algorithm. The computational complexity of the algorithm compares favorably to other known techniques.

VII. REFERENCES

[1] Itakura, F., "Line spectrum representation of linear predictive coefficients of speech signal", *J. Acoust. Soc. Amer.*, Vol. 57, 1975.

[2] Soong, F.K., Juang, B.H., "Line spectrum pair (LSP) and speech data compression", *Proc. of IEEE ICASSP84*, San Diego, May 1984, Vol. 1, pp. 1.10.1-1.10.4

[3] Kang, G.S., Fransen, L.J., "Application of Line-Spectrum Pairs to low-bit-rate speech encoders", *Proc. of IEEE ICASSP85*, Tampa, 1985, pp. 244-247

[4] Kabal, P., Ramachandran, R.P., "Computation of line spectral frequencies using Chebyshev polynomials", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-34, pp. 1419-1426, Dec. 1986.

[5] Grassi, S., Dufaux, A., Ansorge, M., Pellandini, F., "Efficient algorithm to compute LSP parameters from 10th order LPC coefficients", *Proc. of IEEE ICASSP97*, Munich, 1997, pp. 1707-1710

[6] Wu, C.-H., Chen, J.,-H., "A novel two-level method for the computation of the LSP frequencies using a decimation-in-degree algorithm", *IEEE Trans. Speech Audio Processing*, Vol. 5, pp. 106-115, Mar. 1997.

[7] Rothweiler, J., "A rootfinding algorithm for Line Spectral Frequencies," *Proc. of IEEE ICASSP99*, Phoenix, Mar 1999, Vol. 2, pp. 661-664

[8] Korn, G.A., Korn, T.M., *Mathematical Handbook for Scientists and Engineers*, New York, McGraw-Hill, 1968

[9] Oppenheim, A.V., Schafer, R.W., *Discrete-Time Signal Processing*, Englewood Clifs, NJ, Prentice Hall, 1989