Fast Computation of LSP Frequencies Using the Bairstow Method

Yuqun Xue 1,2, Zhijiu Zhu 1,2, Jianhua Jiang 1,2, Yi Zhan 1, Zenghui Yu 1, Xiaohua Fan 1 and Shushan Qiao 1,2,*

1 Smart sensing R&D center, Institute of microelectronics of Chinese academy of sciences, Beijing 100029, China, xueyuqun@ime.ac.cn (Y.X.); zhuzhijiu@ime.ac.cn (Z.Z.); jiangjianhua@ime.ac.cn (J.J.); yizhan@ime.ac.cn (Y.Z.); yuzenghui@ime.ac.cn (Z.Y.); fanxiaohua@ime.ac.cn (X.F.);
2 School of Electronic, Electrical and Communication Engineering, University of Chinese academy of sciences, Beijing 100049, China
* Correspondence: qiaoshushan@ime.ac.cn

Received: 20 January 2020; Accepted: 25 February 2020; Published: 26 February 2020

Abstract: Linear prediction is the kernel technology in speech processing. It has been widely applied in speech recognition, synthesis, and coding, and can efficiently and correctly represent the speech frequency spectrum with only a few parameters. Line Spectrum Pairs (LSPs) frequencies, as an alternative representation of Linear Predictive Coding (LPC), have the advantages of good quantization accuracy and low spectral sensitivity. However, computing the LSPs frequencies takes a long time. To address this issue, a fast computation algorithm, based on the Bairstow method for computing LSPs frequencies from linear prediction coefficients, is proposed in this paper. The algorithm process first transforms the symmetric and antisymmetric polynomial to general polynomial, then extracts the polynomial roots. Associated with the short-term stationary property of speech signal, an adaptive initial method is applied to reduce the average iteration numbers by 26%, as compared to the statics in the initial method, with a Perceptual Evaluation of Speech Quality (PESQ) score reaching 3.46. Experimental results show that the proposed method can extract the polynomial roots efficiently and accurately with significantly reduced computation complexity. Compared to previous works, the proposed method is 17 times faster than Tschirnhus Transform, and has a 22% PESQ improvement on the Birge-Vieta method with an almost comparable computation time.

Keywords: Linear Predictive Coding; Bairstow method; adaptive method; Line Spectrum Pairs; iterative method

1. Introduction

Linear predictive analysis of the speech signal is one of the most powerful speech analysis techniques, which can extract the short-time spectral envelope information of speech signals efficiently, and is widely used in the fields of speech representing for low bit rate transmission or storage, automatic speech and speaker recognition [1–5]. The predominant linear predictive analysis method is Linear Prediction Coding (LPC), featuring better fault tolerance during transmitting spectral envelope information. As an alternative representation of LPC, Line Spectrum Pairs (LSPs) were first introduced by Itakura [6], which have the advantages of good quantization accuracy and low spectral sensitivity.

During the computation of LSPs, the LPC polynomial \( A(z) \) is decomposed into a pair of symmetric and anti-symmetric polynomials. These two polynomials are called LPC polynomials. It has been proven that the roots of the LPC polynomials, namely the LSPs frequencies, are interleaved on the unit circle [6]. Many methods have been proposed to solve these LPC polynomials. Soong and
Juang [7,8] adopted the Discrete Cosine Transform (DCT) to evaluate the cosine functions, based on
the bisection method with a fine grid. They employed a closed formula to extract the roots of LPC
polynomial. Kang and Fransen [9] first proposed to transform the LPC polynomial into the sum of
cosine functions to avoid the complex computation in later processing. They employed the
autocorrelation method and the all-pass filter to extract the roots. Despite these methods solving the
LPC polynomials correctly, they have the disadvantages of high computational complexity from
trigonometric functions, and high memory usage from cosine tables. Kabal and Ramachandram [10]
employed the bisection method and the interpolation property to locate the position of
zero-crossings with a fine grid. While it requires no prior storages or the calculation of trigonometric
functions, the number of bisections cannot be decreased without compromising the search of
zero-crossings. Wu and Chen [11] first proposed to transform LPC polynomial into a pair of
general-form polynomials, then used the closed-form formulas and the modified Newton-Raphson
method to extract polynomial roots. It can avoid the computation of trigonometric functions with a
fine grid. However, the mathematical operations with complex numbers are still time-consuming.
Chen and Ruan [12] proposed the modified complex-free Ferrari formula to reduce the computation.
While this algorithm avoids complex number operations, it still needs modulus operations to get the
roots of the polynomials. Chen and Chang [13] employed the Tschebyscheff transform to reduce the
degree of the polynomials, then used Descartes–Euler to extract the roots. While it can accelerate the
computation processing, it still needs to compute trigonometric functions. Chung-Hsien Chang [14]
proposed a method to solve the LPC polynomials, which is suitable for hardware implementation.

Hence, an efficient method to solve LPC polynomials should have the following characteristics:
1. Avoiding evaluation of trigonometric functions,
2. Avoiding using complex operations and a fine grid,
3. Extracting the roots rapidly and accurately.

Considering the above requirements, we proposed a fast LPC computation method, based on
the Bairstow method, which can extract the roots of LPC polynomials without using complex
operations, a fine grid and trigonometric functions. Luk [15] and Hsiao [16] used the Bairstow
method to solve the polynomial, but without specific algorithm implementation and performance
evaluation of the algorithm. Additionally, the initial method was stubborn, which limited the
convergence speed. Thus, an adaptive initial method, associated with the short-term property of the
speech signal, is proposed to guarantee its convergence in this paper. Then, we apply our method to
a speech compression codec system. Experiments show that, compared with other methods,
including those of Soong and Juang [8], Kabal and Ramachandram [10], the Chen and Wu method
[11], the modified Ferrari’s formula [12], Tschebyscheff transform [13], the Birge-Vieta method [14]
and the previous Bairstow-base LPC analysis methods [15,16], the proposed method can estimate
the LSPs frequencies accurately with the lowest computational complexity, together with high
speech quality.

The rest of this paper is organized as follows: A brief introduction of LSPs frequencies is
reviewed in Section 2. In Section 3, a detail description of the Bairstow method is given, then the
adaptive initial method is given to boost the estimation accuracy and the convergence speed. Section
4 provides a guideline to estimate the 10-order LSPs, and shows the experimental results of the
proposed method, as compared to previous methods. Finally, Section 5 provides the conclusion.

2. Line Spectral Pairs (LSPs) Frequencies

2.1. LPC Polynomial

This section introduces the background of the LSPs frequencies and some of the relevant
properties briefly. Given a p-order LPC, the minimum-phase LPC polynomial is expressed as
follows, according to Reference [14]:

\[ A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} = 1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_p z^{-p}, \]  (1)
where \( a_1, a_2, \ldots, a_p \) are the direct form of LPC coefficients. The LPC polynomial can be decomposed into a symmetric \( P(z) \) polynomial and an anti-symmetric \( Q(z) \) polynomial. The following formulas are their representations:

\[
P(z) = A(z) + z^{-p+1}A(z^{-1})
\]

(2)

\[
Q(z) = A(z) - z^{-p+1}A(z^{-1}).
\]

(3)

The roots of \( P(z) \) and \( Q(z) \) determine the value of LSPs frequencies. According to Reference [14], the above two auxiliary polynomials have the significant properties that their roots are on the unit circle and interleave with each other. Further, \( P(z) \) has a root \( z = -1 (w = \pi) \), and \( Q(z) \) has a root \( z = 1 (w = 0) \). These roots can be eliminated by polynomial division. Thus, the reduced polynomials can be obtained

\[
R^e(z) = \frac{P(z)}{z+1}, \quad R^o(z) = \frac{Q(z)}{z-1}, \quad P : \text{even}
\]

(4)

\[
R^e(z) = P(z), \quad R^o(z) = \frac{Q(z)}{z^2-1}, \quad P : \text{odd}.
\]

(5)

Furthermore, because the coefficients of both \( R^e(z) \) and \( R^o(z) \) are real, the location of the roots for each polynomial are conjugate on the unit circle and can be written as follows:

\[
R^e(z) = \prod_{i=1,3,\ldots,2M-1}(1-2q_iz^{-1}+z^{-2})
\]

(6)

\[
R^o(z) = \prod_{j=2,4,\ldots,2M_j}(1-2\theta_jz^{-1}+z^{-2})
\]

(7)

\[
M_i = \frac{p}{2}, \quad M_2 = \frac{p}{2}, \quad P : \text{even}
\]

(8)

\[
M_i = \frac{p+1}{2}, \quad M_2 = \frac{p-1}{2}, \quad P : \text{odd},
\]

(9)

where \( q_i = \cos(w_i) \), \( \theta_j = \cos(w_j) \), \( M_i + M_2 = P \). The coefficients \( w_i, w_j \) are the line spectral frequencies (LSF) [16] and they satisfy the ordering property as follows:

\[
0 < w_1 < \theta_1 < w_2 < \theta_2 < \ldots < w_{M_1} < \theta_{M_2} < \pi, \quad P : \text{even}
\]

(10)

\[
0 < w_1 < \theta_1 < w_2 < \theta_2 < \ldots < w_{M_2} = w_{M_1-1} < \theta_{M_2} < w_{M_1} < \pi, \quad P : \text{odd}.
\]

(11)

The ordering property of LSF can be illustrated in Figure 1.
Combing $z = e^{jw}$, Equations (6) and (7) can be described as follows:

$$R^e(e^{jw}) = 2e^{-jwM_1} (p(0)\cos(M_1w) + p(1)\cos(M_1-1)w + \ldots + p(M_1-1)\cos(w) + p(M_1)).$$ (12)

$$R^o(e^{jw}) = 2e^{-jwM_2} (q(0)\cos(M_2w) + q(1)\cos(M_2-1)w + \ldots + q(M_2-1)\cos(w) + q(M_2)).$$ (13)

### 2.2. General-Form Polynomial Transformation for the Computation of LSPs Frequencies

The zero crossings of $P(z)$ and $Q(z)$ are the values of LSPs frequencies. In order to find the LSPs frequencies, we need to solve polynomials $R^e(e^{jw}) = 0$ and $R^o(e^{jw}) = 0$.

This paper discusses the commonly used value of $P = 10$, then Equations (12) and (13) can be expressed as a general polynomial:

$$R^e(e^{jw}) = f(0)\cos(Mw) + f(1)\cos(M-1)w + \ldots + f(M-1)\cos(w) + f(M).$$ (14)

The above formula can be deployed to a general-form polynomial as follows:

$$R(w) = 32\cos^5 w + 16f(1)\cos^4 w + 8(f(2) - 5)\cos^3 w$$
$$+ 4(f(3) - 4f(1))\cos^2 w + 2(f(4) - 3f(2) + 5)\cos w + (f(5) - 2f(3) + 2f(1))^2.$$ (15)

where $f(n)$ are the polynomial coefficients, $M = \frac{P}{2}$. We define $x = \cos(w)$. As $w$ is in the range $(0, \pi)$, then $x$ is in the range $(-1, 1)$. Therefore, the above two polynomials can be written as

$$R(w) = 32x^5 + 16f(1)x^4 + 8(f(2) - 5)x^3 + 4(f(3) - 4f(1))x^2$$
$$+ 2(f(4) - 3f(2) + 5)x + (f(5) - 2f(3) + 2f(1))^2.$$ (16)

According to Reference [17], the value of $f(n)$ can be found by the recursive relations:

$$f_p(i+1) = a_{i+1} + a_{i-1} - f_p(i), \quad i = 0, ..., 4$$ (17)

$$f_q(i+1) = a_{i+1} - a_{i-1} + f_q(i), \quad i = 0, ..., 4$$ (18)

where $f_p(n)$ and $f_q(n)$ represent the coefficient of $R^e(w)$ and $R^o(w)$, respectively. According to Reference [18], we compare Equation (16) with a general-form polynomial (19):

$$f(x) = x^5 + \alpha_1 x^4 + \alpha_2 x^3 + \alpha_3 x^2 + \alpha_4 x + \alpha_5,$$ (19)
where \( f(x) \) is a general-form polynomial with the same order of \( R(w) \). Comparing Equation (19) to (16), the coefficients relationship of \( R(w) \) and \( f(x) \) can be written as

\[
\begin{align*}
\alpha_1 &= f(1)/2, \\
\alpha_2 &= (f(2) - 5)/4, \\
\alpha_3 &= (f(3) - 4f(1))/8, \\
\alpha_4 &= (f(4) - 3f(2) + 5)/16, \\
\alpha_5 &= (f(5) - 2f(3) + 2f(1))/32.
\end{align*}
\]

(20)

In summary, after a series of steps, LPC polynomials are turned into a general-form polynomial and the relationship between LPC coefficients and general-form polynomial coefficients is found. Therefore, Section 3 introduces the method to solve the general-form polynomial.

3. Polynomial Solution Based on Bairstow Method

As mentioned in Section 2, the LPC polynomials can be transformed into a general-form polynomial. Furthermore, the relationship between the LPC coefficients and coefficients of the general-form polynomial is presented by Equation (20). To solve the general-form polynomial, this paper employs the Bairstow method.

The Bairstow method [18–20] is a rapid root-finding algorithm for general-form polynomial. Compared to other conventional methods, it can efficiently reduce the number of addition and multiplication operations, and avoids the evaluation of trigonometric and other complex computations. Thus, the proposed method can converge rapidly and reduce the computation time significantly.

The Bairstow method follows from the observation that the roots of a real quadratic polynomial:

\[
x^2 - rx - q,
\]

are the roots of a given \( N \)-order real polynomial

\[
f(x) = \alpha_0 x^N + \alpha_1 x^{N-1} + \ldots + \alpha_{N-1} x + \alpha_N, \alpha_0 \neq 0,
\]

only if \( f(x) \) can be divided by Equation (21) without the remainder. After deflating, the deflated polynomial can be represented as

\[
f(x) = f_i(x)(x^2 - rx - q) + Ax + B, \alpha_0 \neq 0,
\]

where the degree of \( f(x) \) is \( N-2 \). The remainder is expressed as \( Ax + B \). The coefficients of the remainder depend upon \( r \) and \( q \), that is

\[
A = A(r, q), B = B(r, q),
\]

and the remainder vanishes when \( r \) and \( q \) satisfy the condition

\[
A(r, q) = 0, B(r, q) = 0
\]

(25)

According to the scheme, if \( r_0 \) and \( q_0 \) are the initial approximations for \( r \) and \( q \), then the iterative process can be written as

\[
\begin{align*}
r_{i+1} &= r_i + \Delta r, \\
q_{i+1} &= q_i + \Delta q, \\
\Delta r &= -\left(AB_q - BA_q\right)/\left(A_i B_r - A_r B_q\right), i = 0, 1, 2, \ldots, \\
\Delta q &= -\left(AB_r - BA_r\right)/\left(A_i B_q - A_q B_r\right),
\end{align*}
\]

(26)

where \( A, A_q, B_r,B_q \) are the partial derivatives with respect to \( r \) and \( q \) evaluated at \( r_i \) and \( q_i \).
The values $A$ and $B$ can be found by means of a Horner-type [17] scheme. Comparing Equation (22) with (23), we can get the following recursive formula:

$$b_1 = 0$$
$$b_0 = \alpha_0$$
$$b_i = \alpha_i - r$$
$$b_2 = \alpha_2 - rb_1 - q, \quad i = 2, 3, \ldots, N - 2.$$  

$$b_i = \alpha_i - rb_i - qb_{i-2}$$
$$A = \alpha_{N-1} - rb_{N-2} - qb_{N-3}$$
$$B = \alpha_N - rb_{N-2}$$  

(27)

Define $c_{i-1} = -\frac{\partial b_i}{\partial r}, i = 1, 2, \ldots, N$. Combining Equation (27), we can get the following recursive formula:

$$c_0 = 1, c_1 = 0$$
$$c_i = b_i - rc_{i-1} - qc_{i-2}, \quad i = 1, 2, \ldots, N.$$  

(28)

and the value of $dr$ and $dq$:

$$\Delta r = -\frac{b_{N-3}c_{N-3} - b_{N-1}c_{N-2}}{c_{N-2}^2 - c_{N-3}(c_{N-1} - b_{N-1})}$$
$$\Delta q = -\frac{b_{N-1}(c_{N-1} - b_{N-1}) - b_{N}c_{N-2}}{c_{N-2}^2 - c_{N-3}(c_{N-1} - b_{N-1})}.$$  

(29)

The convergence criterion is defined to terminate the iterative procedure when the condition

$$|\Delta r| < \varepsilon \quad \text{and} \quad |\Delta q| < \varepsilon,$$  

(30)

is satisfied for a specified value $\varepsilon$. The predefined value $\varepsilon$ determines the accuracy of the root. Then, we can get the root of $f_i(x)$, assuming there are two distinct roots $x_0$ and $x_1$,

$$x_0 = \left(-r + \sqrt{r^2 - 4q}\right)/2$$
$$x_1 = \left(-r - \sqrt{r^2 - 4q}\right)/2.$$  

(31)

To obtain the root of $x_3$ and $x_4$, we employ a further division of $f_i(x)$ by $(x^2 - rx - q_i)$. Then, we obtain the deflated polynomial:

$$f_i(x) = f_i(x)(x^2 - r_i x - q_i) + A_i x + B_i,$$  

(32)

where the degree of $f_i(x)$ is $N - 4$. To find all the roots, we divide $f_i(x)$ by $(x^2 - r_i x - q_i)$, where $i = 1, 2, \ldots, \left\lfloor N/2 \right\rfloor$, until the deflation results in a quadratic polynomial.

Table 1 shows the pseudocodes of the proposed method according to Equations (26)-(31). $\hat{r}$ and $\hat{q}$ are the initial value of $r$ and $q$, respectively. According to References [17,21], this algorithm is sensitive to the initial value of $r$ and $q$. It will converge rapidly if the initial value of $r$ and $q$ are sufficiently close to their true values. In Section 4, we will discuss in detail how to choose the initial value of $r$ and $q$. 
Table 1. Pseudocode of the proposed method.

1. set \( \hat{r} \) and \( \hat{q} \) as the initial value of \( r \) and \( q \)
2. initialize \( r = \hat{r}, q = \hat{q}, \Delta r = \infty, \Delta q = \infty \)
3. while \( (|\Delta r| > \epsilon) \) or \( (|\Delta q| > \epsilon) \) do
4. \( b_0 = c_0 \)
5. \( b_1 = \alpha_1 - rb_0 \)
6. \( c_0 = b_0 \)
7. \( c_1 = b_1 - rc_0 \)
8. for \( (i = 2; i < N; i++) \) do
9. \( b_i = a_i - rb_{i-1} - qb_{i-2} \)
10. \( c_i = b_i - rc_{i-1} - qc_{i-2} \)
11. end
12. \( \Delta r = -(b_N c_{N-3} - b_{N-3} c_{N-2})/(c_{N-2}^2 - c_{N-3} (c_{N-1} - b_{N-1})) \)
13. \( \Delta q = -(b_{N-1} (c_{N-1} - b_{N-1}) - b_N c_{N-2})/(c_{N-2}^2 - c_{N-3} (c_{N-1} - b_{N-1})) \)
14. \( r = r + \Delta r \)
15. \( q = q + \Delta q \)
16. end
17. \( x_1 = \left(-r + \sqrt{r^2 - 4q}\right)/2 \)
18. \( x_2 = \left(-r - \sqrt{r^2 - 4q}\right)/2 \)
19. return \( x_1, x_2 \)

After calculating the LSPs with the proposed method, the LSPs should be sorted in descending order according to Equations (10) or (11) with the Insert-Sort [22] algorithm.

Denoting the roots of the general-form polynomial are \( x_i \), the LSPs frequencies are given by \( w_i = ar \cos(x_i), i = 1, 2, \ldots, P \). The values of \( w_i \) are ranked as follows:

\[
0 < w_1 < w_2 < \ldots < w_p, \quad P \text{ is even. (33)}
\]

4. Experiment and Results

4.1. Experimental Environment

As mentioned earlier in Section 2, the LSPs are the representative parameters of the LPC filter. To calculate the LSPs, we use the Aurora data [23] as the benchmark. The Aurora database is a professional speech corpus created by the Aurora project (www.eld.org). There are 1001 clean and 1001 noisy speech files from the Test-A set, spoken by 50 males and 50 females used in this paper. The test speech signals are sampled at 8 kHz with a resolution of 16 bits per sample. According to Reference [1], the speech signal has a short-term stationary property, so we select every 80 samples (10 ms) as a frame. In order to evaluate the performance of the proposed method, such as computing time, convergence speed, and its impact on speech quality, we choose ITU-T G.729, which is a data compression algorithm using conjugate-structure algebraic-code-excited linear prediction, as a verification platform. The proposed method is implemented using C language and is inserted into the speech encoding module of the ITU-T G.729 platform. Finally, 1,754,330 sets LSPs of 10-order LPC are evaluated on a personal computer (Intel i5-9300H).

4.2. Selection and Update of Initial Values
As introduced in Section 3, the proposed method is sensitive to the selection of the initial values of \( r \) and \( q \), which greatly influences the convergence speed of polynomial optimization. The following experiment shows how to select appropriate initial values.

Three strategies are proposed to select the initial values of \( r \) and \( q \). The first method, called the fixed initial method, which has been employed by a previous works \([15]\), provides fixed initial values for the roots finding. The second method named statistic initial method calculates the statistical mean values of \( r \) and \( q \) respectively, then uses the statistical mean values as the initial values for the evaluation of the rest. The third method, called the adaptive initial method, which combines the short-term stationary property of the speech signal, then uses the final result of the current frame as the initial values of the next frame.

Aurora corpus is used as the benchmark set. For the fixed initial method, 1001 speech files from \textit{testa\_ns\_snr15} are evaluated with the initial values being 0, 4 or 8. For the statistic initial method, in order to guarantee the robustness, we use 200 speech files from \textit{testa\_clean1} as the analytical set, and 1001 speech files from \textit{testa\_n1\_snr15} as the evaluating set, to guarantee the robustness. After the analytical set is processed to satisfy the convergence criterion, the statistic initial method is employed to calculate their statistical mean values \( r_{\text{ave}} \) and \( q_{\text{ave}} \). Then, these values are used to initialize each frame for all speech samples from the evaluating set. For the adaptive initial method, without the necessity of precalculating initializations, 1001 speech files from \textit{testa\_n1\_snr15} are all adopted as the evaluating set. The initial values of the first frame are randomly generated, and the initial values of the subsequent frames are obtained directly from the previous frame.

Table 2 shows a sectional result of the experiment. \( r \) and \( q \) are the final converged values of each frame. \( \Delta r \) and \( \Delta q \) represent the differences between the initial values and the final values of \( r \) and \( q \) for each frame, respectively. Table 2 shows the \([\Delta r]\) and \([\Delta q]\), which are almost lower than 0.03, and Table 3 shows the statistical mean values of \([\Delta r]\) and \([\Delta q]\) of the 1001 test files. From the table, we know the average values of \([\Delta r]\) and \([\Delta q]\) are 0.0419 and 0.0305, respectively, and the standard deviation is small. Through the statistical analysis, we discover the value of the parameters between adjacent frames only fluctuates slightly.

\[
\begin{align*}
\Delta r &= r_{\text{initial}} - r_{\text{true}} = r_{\text{final}} - r_{\text{initial}} \\
\Delta q &= q_{\text{initial}} - q_{\text{true}} = q_{\text{final}} - q_{\text{initial}}.
\end{align*}
\]  

(34)

| frame | \( r_{\text{initial}} \) | \( r_{\text{final}} \) | \( |\Delta r| \) | \( q_{\text{initial}} \) | \( q_{\text{final}} \) | \( |\Delta q| \) |
|-------|----------------|-----------------|----------------|----------------|----------------|----------------|
| 1     | -0.244103     | -0.255179       | 0.011077       | -0.311074      | -0.301109      | 0.009965       |
| 2     | -0.255179     | -0.241979       | 0.013201       | -0.301109      | -0.279433      | 0.021676       |
| 3     | -0.241979     | -0.238239       | 0.003739       | -0.279433      | -0.276074      | 0.003359       |
| 4     | -0.238239     | -0.258420       | 0.020180       | -0.276074      | -0.299796      | 0.023722       |
| 5     | -0.258420     | -0.320360       | 0.061940       | -0.299796      | -0.267679      | 0.032117       |
| 6     | -0.320360     | -0.350183       | 0.029824       | -0.267679      | -0.246816      | 0.020863       |

Table 2. The adaptive initial method’s updating principle of \( r \) and \( q \).
Table 3. The mean and standard deviation of the $|\Delta r|$ and $|\Delta q|$.

| Mathematic analysis | $|\Delta r|$ | $|\Delta q|$ |
|----------------------|-------------|-------------|
| Mean                 | 0.0419      | 0.0305      |
| Standard Deviation   | 0.0078      | 0.0038      |

To investigate the convergence speed of the three initial selection strategies, we perform a statistical analysis of the experiment data. Table 4 and Figure 2 illustrate the average iteration numbers of the fixed, statics and adaptive initial method. Where $n_1$ and $n_2$ mean iteration numbers of the iterative loop for the first five-order polynomial, the iterative loop for the first three-order polynomial, respectively; $n_3$ and $n_4$ mean iteration numbers of the iterative loop for the second five-order polynomial, the iterative loop for the second third-order polynomial, respectively; Total means the iteration numbers for calculating all 10 roots for the two LPC polynomials. The second column of Table 4 means the different initial value for $r$ and $q$, where the first row applies $r_{initial} = 0$ and $q_{initial} = 0$, the second row applies $r_{initial} = 4$ and $q_{initial} = 4$, and the third row applies $r_{initial} = 8$ and $q_{initial} = 8$ for all test speech frames. In Figure 2, the two vertical lines in each subplot represent the average convergence iterations of the adaptive initial method with the dash, and the static initial method with the dot.

Table 4. The average iteration numbers of the fixed, statistic and adaptive method for defining the initial values with the same computation accuracy.

| Initial method | Average iteration numbers |
|---------------|--------------------------|
|               | $n_1$ | $n_2$ | $n_3$ | $n_4$ | Total |
| 0             | 4.9   | 5.6   | 10.2  | 6.9   | 27.6  |
| Fixed         | 4     | 11.5  | 10.6  | 11.3  | 9.3   | 42.7  |
| 8             | 14.1  | 11.5  | 14.0  | 10.3  | 49.9  |
| Statistic     | 4.7   | 5.4   | 5.2   | 6.4   | 21.7  |
| Adaptive      | 3.7   | 3.4   | 5.2   | 3.7   | 16    |
Figure 2. Histograms of the iteration numbers required to converge, with the adaptive initial method and the static initial method, respectively: (a) n1, (b) n2, (c) n3, and (d) n4.

As Table 4 and Figure 2 show, the setting of the initial values directly affect the convergence iterations. The fixed initial method performs worse than the statistic initial method, and the convergence speed of the adaptive initial method is the fastest, with the average iterations of n1, n2 and n4 at about 3.7, the average iterations of n3 at about 5.2. Further, compared with the fixed initial \( r_{\text{initial}} = 8 \) and \( q_{\text{initial}} = 8 \) method and statics initial method, the total average iteration numbers are reduced by 68% and 26%, respectively, with the same computation accuracy. This also verifies the conclusion in Table 2; the value of \( r \) and \( q \) fluctuate slightly in the adjacent frame due to the short term stationary property of speech signal, thus selecting the final result of \( r \) and \( q \) for the current frame as the initial value for the next frame can reduce the average iteration obviously. Therefore, the adaptive initial method is effective, and we apply the adaptive initial method as the selected initial strategy.

4.3. Solve the 10-order LSP Using the Proposed Method

We randomly select a speech frame to evaluate the computation accuracy of the proposed method. The selection of the initial method follows the strategy mentioned above. We solve the following \( F(x) \) polynomial and present the experimental results in Table 5, Table 6 and Table 7, where \( x_j \) means the jth root of the polynomial, where \( |d| \) means the threshold value of iteration termination.

Table 5 shows the detail steps to solve the \( F(x) \) polynomial using the proposed method, where \( k \) means the kth iterative loop. This process includes two stages, of which a 5-order polynomial and a deflated three-order polynomial are both solved in four iterations sequentially. The solving steps of each stage are elaborately illustrated. In the first stage, a five-order polynomial
is deflated, where $r_i$ and $q_i$ are the initial value of $r$ and $q$ in each iteration, respectively, $r_{i+1}$ and $q_{i+1}$ are calculated by the proposed method after obtaining new values. In the second stage, a deflated three-order polynomial is solved, where $r_{2i}$ and $q_{2i}$ are the initial values in each iteration of the deflated polynomial, and $r_{2i+1}$ and $q_{2i+1}$ are the results calculated by the proposed method using $r_{2i}$ and $q_{2i}$.

Table 5. The detail steps to extract the roots of $F(x)$ using the proposed method.

| $F(x) = x^5 - 0.539786x^4 - 0.760529x^3 + 0.293614x^2 + 0.088310x - 0.014837$ |
|---|---|---|---|---|---|---|
| First stage | $k$ | $r_i$ | $r_{i+1}$ | $q_i$ | $q_{i+1}$ | $|\Delta r|$ | $|\Delta q|$ |
| 1 | -0.702538 | -0.666299 | 0.088680 | 0.071163 | 0.036240 | 0.017517 |
| 2 | -0.666299 | -0.664132 | 0.071163 | 0.070034 | 0.002167 | 0.001129 |
| 3 | -0.664132 | -0.664117 | 0.070034 | 0.070034 | 0.000015 | 0.000000 |
| 4 | -0.664117 | -0.664117 | 0.070034 | 0.070034 | 0.000000 | 0.000000 |

| Second stage | $k$ | $r_{2i}$ | $r_{2i+1}$ | $q_{2i}$ | $q_{2i+1}$ | $|\Delta r|$ | $|\Delta q|$ |
| 1 | -0.652314 | -0.623734 | -0.256786 | -0.282222 | 0.028580 | 0.025436 |
| 2 | -0.623734 | -0.621934 | -0.282222 | -0.283870 | 0.001801 | 0.001648 |
| 3 | -0.621934 | -0.621919 | -0.283870 | -0.283885 | 0.000015 | 0.000015 |
| 4 | -0.621919 | -0.621919 | -0.283885 | -0.283885 | 0.000000 | 0.000000 |

Table 6 presents the difference between approximate and accurate values. The third column is the absolute difference between the accurate roots and the approximate roots, where $|\Delta| = |\text{accurate} - \text{approximate}|$. The experimental results show that the computation accuracy is $0.00001$, which means that the approximate values are very close to the accurate ones.

Table 6. Accurate and approximate values determined by the proposed method ($\alpha = 0.00001$).

| $x_j$ | Accurate | Approximate | $|\Delta|$ |
|---|---|---|---|
| 1 | -0.792042 | -0.792041 | 0.000001 |
| 2 | 0.928270 | 0.928265 | 0.000005 |
| 3 | 0.641682 | 0.641686 | 0.000004 |
| 4 | -0.395398 | -0.395388 | 0.000001 |
| 5 | 0.175723 | 0.175723 | 0.000000 |

After the first and the second stages, we obtain the resulting value of $x_j$, then transform the $x_j$ into the LSF $w_j$ by using the formula $w_j = \arccos(x_j)$. Combing Table 5, Table 6 and Table 7, the results show that the optimization converges extremely fast, and the value of LSF is accurate.
Table 7. The results of LSP frequencies of \( F(x) = 0 \).

\[
F(x) = x^5 - 0.539786x^4 - 0.760529x^3 + 0.293614x^2 + 0.088310x - 0.014837
\]

| \( i \) | \( r_i \) | \( q_i \) | \( r_{2i} \) | \( q_{2i} \) | \( x_i \) | \( w_i \) |
|-------|--------|--------|--------|--------|-------|-------|
| 1     | -0.702538 | 0.088680 | \( N/A \) | \( N/A \) | 0.928265 | 0.381076 |
| 2     |         |        |        |        | 0.641686 | 0.874102 |
| 3     |         |        |        |        | 0.175723 | 1.394156 |
| 4     |         |        |        |        | -0.395388 | 1.977287 |
| 5     |         |        |        |        | -0.792041 | 2.484941 |

4.4. Performance of the Proposed Method

This section evaluates the performance of the proposed method, including the computation time and the speech quality, by making a comparison between the proposed method and other methods, including the Birge-Vieta method [14], Tschirnhaus Transform [13], original/Modified Ferrari’s method [12], Chen and Wu method [11], Soong and Juang [8], and the Full search.

Due to the lack of the algorithm performance parameters Reference [14], we established the Birge-Vieta method model strictly according to the reference paper. Table 8 shows the Perceptual Evaluation of Speech Quality (PESQ) [24] comparison between the proposed method and the Birge-Vieta method. Table 9 shows the comparison of the average computation time between the proposed method and the previous works, under the same computation accuracy (\( \alpha = |x_{i+1} - x_i| = 0.00001 \)).

Table 8. PESQ comparison of the proposed method and the previous methods.

| Methods                  | Test speech | PESQ |
|--------------------------|-------------|------|
| The proposed method      | \( testa\_clean1 \) | 3.35 |
|                          | \( testa\_n1\_snr15 \) | 3.46 |
| Birge-Vieta method       | \( testa\_clean1 \) | 2.74 |
|                          | \( testa\_n1\_snr15 \) | 3.36 |

Table 9. Comparison of the average computation time with the previous works (\( \alpha = 0.00001 \)).

| Methods                  | Time (ms) | Normalized Clock numbers | Environment |
|--------------------------|-----------|--------------------------|-------------|
| The proposed method      | 0.0016    | 5920                     | 3.7 GHz     |
| Birge-Vieta method       | 0.0018    | 6660                     | 3.7 GHz     |
| Tschirnhaus Transform    | 0.0614    | 104380                   | 1.7 GHz     |
| Modified Ferrari’s       | 0.0802    | 136340                   | 1.7 GHz     |
| Original Ferrari’s       | 0.1640    | 278800                   | 1.7 GHz     |
| Chen and Wu              | 0.1651    | 280670                   | 1.7 GHz     |
| Soong and Juang          | 0.5316    | 903720                   | 1.7 GHz     |
| Full Search              | 700.32    | 1190540                  | 1.7 GHz     |

The second column of Table 9 is the average computation time of converting 1,754,330 sets of LPC coefficients to LSF frequencies. Associated Table 7 and Table 8, the results show that the PESQ of the proposed method has improved by 22%, in comparison to the Birge-Vieta method under the \( testa\_clean1 \), while its computation time is comparable. After normalizing the CPU frequency of system environment to 3.7 GHz and with the same computation accuracy, the proposed method is 17 times faster than the Tschirnhaus transform method, 23 times faster than the Modified Ferrari’s
method, 47 times faster than the Original Ferrari’s method, 47.4 times faster than the Chen and Wu method, 152 times faster than the Soong and Juang method, and is 201 times faster than the Full Search. It is obvious that the proposed method is the fastest, with the same computation accuracy.

5. Conclusions

This paper proposed an efficient method to calculate the roots of LPC polynomial, which can transform LPC coefficients to LSPs accurately and rapidly. Associated with the short-term stationary property of speech signal, an adaptive initial method is applied to reduce the average iteration numbers by 26%, as compared with the statics initial method, with the PESQ score reaching 3.35. Experiment results show that the proposed method is 17 times faster than the Tschirnhus Algorithm, 152 times faster than the Soong and Juang method, and is 201 times faster than the Full method, 47 times faster than the Original Ferrari’s method, 47.4 times faster than the Chen and Wu method, 152 times faster than the Soong and Juang method, and is 201 times faster than the Full Search. It is obvious that the proposed method is the fastest, with the same computation accuracy.

5. Conclusions

This paper proposed an efficient method to calculate the roots of LPC polynomial, which can transform LPC coefficients to LSPs accurately and rapidly. Associated with the short-term stationary property of speech signal, an adaptive initial method is applied to reduce the average iteration numbers by 26%, as compared with the statics initial method, with the PESQ score reaching 3.35. Experiment results show that the proposed method is 17 times faster than the Tschirnhus Transform, and has a 22% PESQ improvement in comparison to the Birge-Vieta method, with almost the same computation time. Compared to other root-finding methods, the proposed method is the fastest one with comparable speech quality.

Author contributions: Theory and conceptualization, X.Y.; methodology, X.Y.; software design and development, X.Y.; validation, X.Y.; formal analysis, X.Y. and Z.Z.; investigation, X.Y.; writing–original draft preparation, X.Y.; writing–review and editing, X.F., Z.Z., J.J, Y.Z, Z.Y. and S.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: I would like to thank Smart sensing R&D center, Institute of microelectronics of Chinese academy of sciences that supported us in this work and Zhejiang University for assistance of the Aurora corpus evaluation.

Conflicts of Interests: The authors declare no conflict of interest.

Reference

1. Rabiner, L.R.; Schafer, R.W. Theory and Applications of Digital Speech Processing; Pearson Education: London, UK, 2011; p.473.
2. Makhoui, J. Linear Prediction: A Tutorial Review. Proc. IEEE 1975, 63, 561–580.
3. Chowdhury, A.; Ross, A. Fusing MFCC and LPC Features Using 1D Triplet CNN for Speaker Recognition in Severely Degraded Audio Signals. IEEE Trans. Inf. Forensics Secur. 2019, 15, 1616–1629.
4. Alku, P.; Saeidi, R. The Linear Predictive Modeling of Speech From Higher-Lag Autocorrelation Coefficients Applied to Noise-Robust Speaker Recognition. IEEE/ACM Trans. Audio Speech Lang. Process. 2017, 25, 1606–1617.
5. Ramalho, L.; Fonseca, M.N.; Klautau, A.; Lu, C.; Berg, M.; Trojer, E.; Höst, S. An LPC-Based Fronthaul Compression Scheme. IEEE Commun. Lett. 2017, 21, 318–321.
6. Itakura, F. Line Spectrum Representation of Linear Predictive Coefficients of Speech Signals. J. Acoust. Soc. Am. 1975, 57, 535.
7. Soong, F.K.; Juang, B.H. Optimal quantization of LSP parameters. IEEE Trans. Speech Audio Process. 1993, 1, 15–24.
8. Soong, F.K.; Juang, B.H. Line Spectrum Pair (LSP) and Speech Data Compression. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), San Diego, CA, USA, 19–21 March 1984.
9. Kang, G.; Fransen, L. Application of line-spectrum pairs to low-bit-rate speech encoders. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Tampa, FL, USA, 26–29 April 1985; p. 8857.
10. Kabal, P.; Ramachandran, R.P. The Computation of Line Spectral Frequencies Using Chebyshev Polynomials. IEEE Trans. Acoust. Speech Signal Process. 1986, 34, 1419–1426.
11. 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 Process. 1997, 5, 106–115.
12. Chen, S.H.; Chang, Y.; Ruan, J.C. An Efficient Computation of LSP Frequencies Using Modified Complex-Free Ferrari Formula. Signal Process. Syst. 2008, 52, 153–163.
13. Chen, S.H.; Chang, Y.; Syuan, C.J.Y. The Computation of Line Spectrum Pair Frequencies Using Tschirnhaus Transform. In Proceedings of the International Symposium on Circuits and Systems, Taipei, Taiwan, 24–27 May 2009; pp. 288–291.
14. Chang, C.H.; Chen, B.W.; Chen, S.H.; Wang, J.F.; Chiu, Y.H. Low-Complexity Hardware Design for Fast Solving LSP’s With Coordinated Polynomial Solution. IEEE Trans. Large Scale Integr. Syst. 2015, 23, 230–243.
15. Luk, W.S. Finding roots of real polynomial simultaneously by means of Bairstow’s method. Bit Numer. Math. 1996, 36, 302–308.
16. Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP); International Telecommunication Union, Geneva, Switzerland, 1996.
17. Hildebrand, F.B. Introduction to Numerical Analysis, 2nd ed.; McGraw-Hill: New York, NY, USA, 1974; p. 613.
18. O’Donnell, J. A System for very low data rate speech communication. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Atlanta, GA, USA, 30 March–1 April 1981; pp. 8–11.
19. Rothweiler, J. On polynomial reduction in the computation of LSP frequencies. IEEE Trans. Speech Audio Process. 1999, 7, 592–594.
20. Hsiao, C.C.; Brodersen, R. A multi-rate root LPC speech synthesizer. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), San Diego, CA, USA, 19–21 March 1984; pp. 19–21.
21. Stoer, J.; Bulirsch, R. Introduction to Numerical Analysis, 2nd ed; Springer Science & Business Media: Berlin, Germany, 2013; p. 333.
22. Wikipedia. Available online: https://en.wikipedia.org/wiki/Insertion_sort (accessed on 15 January 2020).
23. Aurora Corpus. Available online: http://portal.elda.org/en/catalogues/free-resources/free-lrs-set-1/ (accessed on 15 January 2020).
24. Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs; International Telecommunication Union, Geneva, Switzerland, 2001.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).