September 11, 2007

THE BURG-WILSON ALGORITHM

In 1969.G.Wilson published an algorithm for factoring a finite length autocorrelation function into minimum and maximum phase factors. The algorithm applied the Newton-Raphson iteration procedure to the polynomials involved in the autocorrelation equation, resulting in quadratic convergence. Wilson derived the conditions for convergence and proved that under appropriate starting conditions, the algorithm would always converge to the desired solution. However the algorithm involved solving a messy matrix equation, which was really the only unbeautiful thing about his paper.

Later, John Parker Burg saw how to avoid the messy matrix equation based on Z-transform polynomial equations contained in his PH.D thesis. He expanded the problem to the complex case and rederived wilson's results in signal processing terms. This more elegant algorithm is known as the Wilson-Burg.

Burg has recently discovered an improvement on this algorithm which is sufficiently significant to change its name to that of Burg-Wilson. It is derived below.

BURG-WILSON ALGORITHM

We wish to solve the equation

$$R(Z) = SA^{*}(Z^{-1})A(Z) \qquad \text{Eq2.1}$$

where $R(Z)$ is a finite length, complex auto correlation function, S is positive real number, $A(Z)$ is a prediction error filter and $A^{*}(Z^{-1})$ is its complex conjugate time reverse.

For the Nth order case,

$$R(Z) = \sum_{n=-N}^{N} r_n Z^n \qquad \text{Eq 2.2}$$

with $r_{-n} = r_n^{*}$, and $A(Z) = \sum_{n=0}^{N} a_n Z^n$ Eq 2.3

with $a_0 = 1$.

A further discussion of equations 2.2 and 2.3 is required to complete the description of $R(Z)$ and $A(Z)$.

As Eq2.3 shows, $A(Z)$ is an N^{th} order polynomial. For $A(Z)$ to be a prediction error filter, its Z^0 coefficient must be unity, which we have indicated. Also, all roots of the polynomial must lie outside the unit circle. This second condition means that $A(Z)$ is minimum phase.

As Eq2.2 shows, $R(Z)$ is complex conjugate symmetric about its Z^0 term. However, for $R(Z)$ to be a finite length autocorrelation, there must be a solution to Eq2.1. Eq2.1 actually says that $R(Z)$ is the auto correlation of $A(Z)$, scaled S.

In Wilson's algorithm, the scalar S was included in the two factors by considering $\sqrt{S}A(Z)$ and $\sqrt{S}A^*(Z^{-1})$. In this case $\sqrt{S}A(Z)$ is minimum phase, but not necessarily a prediction error filter since S would not be required to be one. The Burg-Wilson algorithm separates out the scalar, so $A(Z)$ is a PEF.

The BW algorithm is iterative, so we shall describe one loop in the iteration. Let S_n and $A_n(Z)$ be our estimates of the solution at the beginning of the N^{th} iteration.

We easily calculate the N^{th} guess of $R(Z)$ as

$$R_n(Z) = S_n A_n^*(Z^{-1}) A_n(Z) \quad \text{Eq4.1}$$

Taking its variation, we have

$$\delta R_n(Z) = \delta S_n A_n^*(Z^{-1}) A_n(Z) + S_n \delta A_n^*(Z^{-1}) A_n(Z) + S_n A_n^*(Z^{-1}) \delta A_n(Z) \qquad \text{Eq4.2}$$

We want $\delta R_n(Z)$ to be such that

$$R(Z) = R_n(Z) + \delta R_n(Z) \qquad \text{Eq4.3}$$

Dividing Eq4.3 by $R_n(Z)$ and using Eq 4.1 and 4.2, we have

$$\frac{R(Z)}{S_n A_n^*(Z^{-1}) A_n(Z)} = 1 + \frac{\delta S_n}{S_n} + \frac{\delta A_n^*(Z^{-1})}{A_n^*(Z^{-1})} + \frac{\delta A_n(Z)}{A_n(Z)} \qquad \text{Eq4.4}$$

Now we can solve Eq4.4 for $\delta A_n(Z)$, and thus for $\delta A_n^*(Z^{-1})$, that make Eq4.3 true. But we then have for out next estimate of $A(Z)$ to be

$A_{n+1}(Z) = A_N(Z) + \delta A_n(Z)$, so we can rewrite Eq4.4 as

$$\frac{R(Z)}{S_n A_n^*(Z^{-1})A_n(Z)} + 1 = \frac{\delta S_n}{S_n} + \frac{A_{n+1}^*(Z^{-1})}{A_n^*(Z^{-1})} + \frac{A_{n+1}(Z)}{A_n(Z)} \qquad \text{Eq 5.1}$$

Now there are at least two different way to solve the equation

$$\frac{1}{A_n^*(Z^{-1})A_n(Z)} = Q_n(Z) = \ldots. + q_{-1}Z^{-1} + q_0 + q_1 Z^1 + \ldots \quad \text{Eq 5.2}$$

where $Q_n(Z)$ is an infinitely long, complex conjugate autocorrelation function. One is the Levinson-Burg algorithm (inverse of) and another is the algorithm shown in Burg's PH.D thesis. We want to convolve $Q_n(Z)$ by $R(Z)$ to get the part of the result from Z^0 to Z^N. This means that we need to have calculated $Q_n(Z)$ from $q_{-N}Z^{-N}$ to $q_{2N}Z^{2N}$.

Let $X_n(Z) = R(Z)Q_n(Z)$ .We note that $X_n(Z)$ is an infinitely long, complex conjugate symmetric autocorrelation . Eq5.1 then becomes

$$\frac{X_n(Z)}{S_n} + 1 = \frac{\delta S_n}{S_n} + \frac{A_{n+1}^*(Z^{-1})}{A_n(Z^{-1})} + \frac{A_{n+1}(Z)}{A_n(Z)} \qquad \text{Eq5.3}$$

Being PEF's, both $A_n(Z)$ and $A_{n+1}(Z)$ have ones for their leading coefficients. Also, since $A_n(Z)$ is stable. We have

$$\frac{A_{n+1}(Z)}{A_n(Z)} = 1 + b_1 Z \ + b_2 Z^2 + \ldots \quad \text{Eq6.1}$$

Also

$$\frac{A_{n+1}^*(Z)}{A_n^*(Z)} = \ldots + b_2^* Z^{-2} + b_1^* Z^{-1} + 1 \quad \text{Eq6.2}$$

Since $\frac{\delta S_n}{S_n}$ only affects the Z^0 term, we have the equation

$$\frac{X_0}{S_n} + 1 = \frac{\delta S_n}{S_n} + 1 + 1 \qquad \text{Eq6.3}$$

or $X_0 = \delta S_n + S_n = S_{n+1}$ Eq6.4

Continuing with this "pure" Newton-Raphson approach, we get the equation

$$\frac{X(Z) - X_0}{S_n} + 2 = \frac{A^*_{n+1}(Z^{-1})}{A^*_n(Z^{-1})} + \frac{A_{n+1}(Z)}{A_n(Z)} \qquad \text{Eq6.5}$$

Lot

$$X_t(Z) = X_1 Z + X_2 Z^2 + \ldots \qquad \text{Eq6.6}$$

Then we have

$$1 + \frac{X_t(Z)}{S_n} = \frac{A_{n+1}(Z)}{A_n(Z)} \qquad \text{Eq7.1}$$

and thus we solve for $A_{n+1}(Z)$ as

$$A_{n+1}(Z) = [1 + \frac{X_t(Z)}{S_n}]A_n(Z) \qquad \text{Eq7.2}$$

This can be considered as the "pure" Newton-Raphson process in that we start with S_n, $A_n(Z)$ to find S_{n+1} and $A_{n+1}(Z)$.

But since Eq6.4 finds S_{n+1}, which is supposed to be better than S_n, why not use it in Eq7.1 instead of S_n? Since $S_{n+1} = X_0$, Eq6.5 is simplified to

$$\frac{X(Z)}{X_0} + 1 = \frac{A^*_{n+1}(Z^{-1})}{A^*_n(Z^{-1})} + \frac{A_{n+1}(Z)}{A_n(Z)} \qquad \text{Eq7.3}$$

We note that if $R(Z)$ is a finite length autocorrelation function, then $X(Z)$ is also an autocorrelation function (infinite length), and thus $X(Z)/X_0 + 1$ is also, With Eq7.1 changed to

$$1 + \frac{X_t(Z)}{X_0} = \frac{A_{n+1}(Z)}{A_n(Z)} \qquad \text{Eq7.4}$$

We are looking at the "positive half" of Eq7.3, which is a positive real function, and is thus also minimum phase. Thus

$$A_{n+1}(Z) = [1 + \frac{X_t(Z)}{X_0}]A_n(Z) \qquad \text{Eq8.1}$$

is also minimum phase.

LOOKING AT SCALING IN THE WILSON ALGORITHM

Let us write the N^{th} guess of the minimum phase filter in the Wilson algorithm as $\Delta_n A_n(Z)$ where $A_n(Z)$ is a PEF. The algorithm becomes

$$\frac{S(Z)}{\sigma_n^2 A_n^*(Z^{-1})A_n(Z)} + 1 = \frac{\sigma_{n+1}A_{n+1}^*(Z^{-1})}{\sigma_n A_n^*(Z^{-1})} + \frac{\sigma_{n+1}A_{n+1}(Z\ )}{\sigma_n A_n(Z\ )}$$

Let $X_n(Z) = \frac{S(Z)}{A_n^*(Z^{-1})A_n(Z)} = \ldots + X_{n,-1}Z^{-1} + X_{n,0} + X_{n,1}Z$?(It seems one or more terms is missing caused by photocopy)

So we have

$$\frac{X_n(Z)}{\sigma_n^2} + 1 = \frac{\sigma_{n+1}}{\sigma_n}\left[\frac{A_{n+1}^*(Z^{-1})}{A_n^*(Z^{-1})} + \frac{A_{n+1}(Z\ )}{A_n(Z\ )}\right]$$

To make the Z^0 sides equal, we have

$$\frac{X_{n,0}}{\sigma_n^2} + 1 = 2\frac{\sigma_{n+1}}{\sigma_n} \qquad \text{Eq9.1}$$

or

$$\sigma_{n+1} = \frac{1}{2}\left[\frac{X_{n,0}}{\sigma_n} + \sigma_n\right] \qquad \text{Eq9.2}$$

Which is Newton's square root algorithm for finding the square root of $X_{n,0}$. Thus, the Wilson algorithm, in effect, does one iteration loop with σ_n and $A_n(Z)$ by calculating $X_{n,0}$ from $S(Z)$ and $A_n(Z)$ and then doing one loop of the the Newton algorithm.

Let $X_{\infty,0}$ be the limit of $X_{n,0}$. Then we have

$$X_{n,0} > X_{n+1,0} > X_{\infty,0} \qquad \text{Eq9.3}$$

and $\sigma_n > \sigma_{n+1} > \sqrt{X_{\infty,0}}$ Eq9.4

and $\sigma_n > \sqrt{X_{n,0}}$ Eq9.5

and $\sigma_{n+1} > \sqrt{X_{n,0}}$ Eq9.6

An interesting way to look at the new algorithm is that offer $X_{n,0}$ is found, then the Newten algorithm is run to convergence, instead of just one loop as in the Wilson algorithm, But actually, since we want σ_{n+1}^2, that is then just $X_{n,0}$.

In running the two algorithms, the new one converges faster to start with, as one might expect. The Wilson-Burg seems to catch up however on the $S(Z)$ that were looked at with roots close to the unit circle.

The "pure" Newton- Raphson algorithm as given by Eq7.2 is actually bad in that if S_n is too small, then $1+\frac{X_t(Z)}{S_n}$ may not be minimum phase and so $A_{n+1}(Z)$ is not a PEF!

It is a little strange, that we are just inputting $A_n(Z)$ to output $A_{n+1}(Z)$ and $scalar(n+1)$, where the Wilson algorithm is in effect input both $A_n(Z)$ and $\left[scalar(n)\right]^{\frac{1}{2}}$. The difference may be that we do the square root algorithm to infinity and don't need a starting value.

OCT 8, 2007

COMPARING THE WB AND BW ALGORITHMS USING THE SECOND EXAMPLE TEST CASE IN WILSON'S PAPER.

WILSON'S second example is meant to be severe in that it has multiple roots close to the unit circle. The solution polynomial was

$$P(Z) = (1+1.72Z+0.99Z^2)(1+1.6Z+0.98Z^2)(1-0.3Z+0.95Z^2)(1-0.96Z)(1+0.97Z)$$

Eq11.1

This example showed much faster convergence than then method of Baner, which is linear. Comparing WB and BW in this case, we shall see that BW is some better than WB. We start the WB with the filter scaled according to Wilson's suggestion. The BW algorithm does not have an input for scalar. To make a comparison, we square Wilson's scaling with the BW scalar. Since the leading term of $P(Z)$ is one, but should convergent to unity. We compare results after several loops of the algorithms.

LOOP	$WB \quad Scaling^2$	BW
1	237.744219022915	50.016781113136
2	62.32712331007424	10.44495913562912
3	18.15332379441431	4.349470367103882
4	6.82772146915455	2.49763526522948
5	3.56743028842835	1.71467495696778
6	2.30907361310224	1.37295871930091
7	1.72381234361224	1.20883205095333
8	1.43299574758966	1.10868548220676
9	1.26684388153405	1.04836861484199
10	1.15813770448167	1.01523617519669
11	1.08374092395799	1.00282110930902
12	1.03524336000258	1.00020274541703
13	1.01010788335388	1.00000183198106
14	1.00158475345168	1.00000000009879
15	1.00008017431161	0.99999999994077
16	1.00000030901668	0.99999999992519
17	0.99999999995837	
18	0.99999999993416	