Polyphase Representation of Downsampled Sequences

The standard expression for the Fourier transform of a sequence is

$$X(\omega) = \Sigma x(n)e^{-j\omega n}$$

Since n integrates out of the expression, this is a function of ω (omega), the frequency. When we downsample a sequence, we first set to 0 all of the odd numbered elements:

$X_{fe}(n)$ = …, 0, x(-2), 0, x(0), 0, x(2), 0, … (fe stands for "full even")

We can do this by adding 2 sequences together and dividing the sum by 2:

x (the original sequence) and x (with the odd element negated)

(…, x(-1), x(0), x(1), x(2),… + …, -x(-1), x(0), -x(1), x(2), …)/2

or

$[x(n) + x(n)*(-1)^n]/2 = X_{fe}(n)$

Clearly (hopefully), with the odd elements negated in one and not negated in the other, these elements become 0 in the sum. Note: the addition of 2 sequences is not usually defined in math and engineering books. It is assumed that everybody understands that x(n) + y(n) = z(n), e.g. x(1) + y(1) = z(1), with z(n) being the resultant "summed" sequence. It is an element by element sum or "inner sum".

Since $(-1)^n = e^{-j\pi n}$ we can incorporate the negation in the Fourier transform of the second sequence. Taking the Fourier transform of the sum of both sequences we have:

$$X_{fe}(\omega) = (1/2)[\Sigma x(n)e^{-j\omega n} + \Sigma x(n)e^{-j\omega n}e^{-j\pi n}]$$

$$X_{fe}(\omega) = (1/2)[\Sigma x(n)e^{-j\omega n} + \Sigma x(n)e^{-j(\omega+\pi)n}]$$

$$X_{fe}(\omega) = (1/2)[X(\omega) + X(\omega-\pi)]$$

In a similar way we can create the expression for $X_{fo}(n)$, the "full odd" sequence by subtracting the second sequence:

$$X_{fo}(\omega) = (1/2)[\Sigma x(n)e^{-j\omega n} - \Sigma x(n)e^{-j\omega n}e^{-j\pi n}]$$

$$X_{fo}(\omega) = (1/2)[\Sigma x(n)e^{-j\omega n} - \Sigma x(n)e^{-j(\omega+\pi)n}]$$

$X_{fo}(\omega) = (1/2)[X(\omega) - X(\omega-\pi)]$

Obviously (hopefully),

$X(\omega) = X_{fe}(\omega) + X_{fo}(\omega)$

Converting to the z domain in easy (hopefully), using the following formulas:

$e^{j\omega} \Leftrightarrow z$

$e^{j(\omega+\pi)} \Leftrightarrow -z$

Note: $e^{j(\omega+\pi)} = e^{j\omega}e^{j\pi} = z * (-1) = -z$

Therefore

$X_{fe}(z) = (1/2)[X(z) + X(-z)]$

$X_{fo}(z) = (1/2)[X(z) - X(-z)]$

Recall that these are the z transforms of the sequences

$X_{fe}(n) = \ldots, 0, x(-2), 0, x(0), 0, x(2), 0, \ldots$

$X_{fo}(n) = \ldots, x(-3), 0, x(-1), 0, x(1), 0, x(3), 0, \ldots$

Note: If we wish to shift the odd sequence to the left, we multiply the z transform by z.

Compressing the sequences means removing the 0s and filling in the positions held by the 0s with the remaining values (in order). The above sequences now become

$X_{ep}(n) = \ldots, x(-2), x(0), x(2), \ldots$

$X_{op}(n) = \ldots, x(-3), x(-1), x(1), x(3), \ldots$

Where ep and op mean "even phase" and "odd phase" respectively. $X_{ep}(n)$ and $X_{op}(n)$ are written as $X_0(n)$ and $X_1(n)$ respectively. When we compress the sequences with replace z with $z^{-1/2}$. So the new transforms are now

$X_0(z) = (1/2)[X(z^{-1/2}) + X(-z^{-1/2})]$

$X_1(z) = (1/2)[X(z^{-1/2}) - X(-z^{-1/2})]$

or

$X_0(z^2) = (1/2)[X(z) + X(-z)]$

$X_1(z^2) = (1/2)[X(z) - X(-z)]$

Approching even and odd phase the other way

$X_{ep}(n) = \ldots, x(-2), x(0), x(2), \ldots$

$X_{op}(n) = \ldots, x(-3), x(-1), x(1), x(3), \ldots$

$X_{ep}(\omega) = \Sigma x(2k)e^{-j\omega k} = \Sigma x(n)e^{-j(\omega/2)n}$, n=2k

$X_{op}(\omega) = \Sigma x(2k+1)e^{-j\omega k} = \Sigma x(n)e^{-j\omega(n-1)/2}e^{j\omega/2} = \Sigma x(n)e^{-j(\omega/2)n}e^{j\omega/2}$

$= e^{j\omega/2}\Sigma x(n)e^{-j(\omega/2)n}$, n=2k-1

And the corresponding z transform

$X_{ep}(z) = \Sigma x(2k)z^{-k} = \Sigma x(n)z^{-(1/2)n}$, n=2k

$X_{op}(z) = \Sigma x(2k+1)z^{-k} = z^{1/2}\Sigma x(n)z^{-(1/2)n}$, n=2k-1

Expanding (upsampling) the compressed (downsampled) sequence inserts 0s at every other location. This is the inverse of compressing mentioned above. Expanding simply substitutes z^2 for z in the z transform. *But there is a slight twist*. The odd sequence has its elements lined up exactly with the even sequence. If we want to reconstruct the original sequence we have to shift the odd sequence with respect to the even sequence and then add. This also in evident in the $z^{1/2}$ term in the odd phase z transform above:

$X(z) = X_{ep}(z^2) + z^{-1}X_{op}(z^2)$

The shift is represented by the z^{-1} term. Now we have an expression for X(z) in terms of its even and odd phases.

<u>Polyphase representation of downsampled sequences</u>

All the results can be combined into a concise set of expressions. From this point on

$X_{ep} = X_0$

$X_{op} = X_1$

We know from the previous expressions

$X_{fe}(z) = X_0(z^2)$

$X_{fo}(z) = z^{-1}X_1(z^2)$

The convolution of a sequence with a transfer function can be expressed in terms of the even and odd parts of the two:

$X(z)C(z) = \{(1/2)[X(z) + X(-z)] + (1/2)[X(z) - X(-z)]\} * \{(1/2)[C(z) + C(-z)] + (1/2)[C(z) - C(-z)]\}$

or more succinctly

$X(z)C(z) = [X_{fe}(z) + X_{fo}(z)] * [C_{fe}(z) + C_{fo}(z)]$

And we can substitute the even and odd phases in this expression:

$X(z)C(z) = [X_0(z^2) + z^{-1}X_1(z^2)] * [C_0(z^2) + z^{-1}C_1(z^2)]$

Expanding this expression we get

$X(z)C(z) = X_0(z^2)C_0(z^2) + z^{-1}X_0(z^2)C_1(z^2) + z^{-1}X_1(z^2)C_0(z^2) + z^{-2}X_1(z^2)C_1(z^2)$

And grouping the terms

$X(z)C(z) = X_0(z^2)C_0(z^2) + z^{-2}X_1(z^2)C_1(z^2) + z^{-1}X_0(z^2)C_1(z^2) + z^{-1}X_1(z^2)C_0(z^2)$

$X_0(z^2)C_0(z^2) + z^{-2}X_1(z^2)C_1(z^2)$ = even phase

$z^{-1}X_0(z^2)C_1(z^2) + z^{-1}X_1(z^2)C_0(z^2)$ = odd phase

To downsample the output of the filter, $X(z)C(z)$, the odd phase is removed and the elements are compressed ($z \Leftrightarrow z^{1/2}$):

$X(z^{(1/2)})C(z^{(1/2)})_{downsampled} = X_0(z)C_0(z) + z^{-1}X_1(z)C_1(z)$

The nice thing about this expression is that the even and odd phases of the sequence are grouped with the even and the odd phases of the filter. This is the polyphase expression. It can be further expressed as the product of the polyphase matrix and the input:

$[C_0(z)\ C_1(z)]\ [X_0(z)\ z^{-1}X_1(z)]^T$

$[C_0(z)\ C_1(z)]$ = the polyphase matrix

Thus the sampled filter output can be implemented as 2 parallel operations: