Two Interpretations of Frequency Warped Transfer Functions

Robin Schmidt (www.rs-met.com)

March 19, 2011

In the paper [1], Constantinides presents formulas, how to substitute z^{-1} in a digital prototype lowpass filter's z-domain transfer function in order to achieve various filter characteristics, namely lowpass (with different cutoff frequency), highpass, bandpass and bandreject. Here, we will use a somewhat simplified notation and discuss two different interpretations of these substitutions and their implementation.

The Substitutions for z^{-1}

First Order Transforms

First order transforms warp a lowpass prototype filter either into another lowpass filter (with different cutoff frequency) or into a highpass filter (which may also have a different cutoff frequency). In these transforms, we substitute each occurrence of z^{-1} in the transfer function with:

$$z^{-1} \leftarrow A_1(z) = g\frac{c + z^{-1}}{1 + cz^{-1}} \tag{1}$$

where g, c are parameters that can be computed from the prototype filter's normalized radian cutoff frequency ω_p and the target filter's normalized radian cutoff frequency ω_t, via:

$$\boxed{\begin{array}{lll} g = 1, & c = -\dfrac{\sin(\frac{\omega_p - \omega_t}{2})}{\sin(\frac{\omega_p + \omega_t}{2})} & \text{lowpass-to-lowpass} \\ g = -1, & c = -\dfrac{\cos(\frac{\omega_p + \omega_t}{2})}{\cos(\frac{\omega_p - \omega_t}{2})} & \text{lowpass-to-highpass} \end{array}} \tag{2}$$

Second Order Transforms

Second order transforms warp a lowpass prototype filter either into bandpass- or bandreject filter with adjustable normalized radian lower and upper bandedge frequencies ω_l, ω_u. In these transforms, we substitute each occurrence of z^{-1} in the transfer function with:

$$z^{-1} \leftarrow A_2(z) = g\frac{d + cz^{-1} + z^{-2}}{1 + cz^{-1} + dz^{-2}} \tag{3}$$

the coefficients g, c, d can be calculated via:

$$\begin{array}{|ll|}\hline a = -\dfrac{\cos(\frac{\omega_u+\omega_l}{2})}{\cos(\frac{\omega_u-\omega_l}{2})},\ t_1 = \tan(\dfrac{\omega_p}{2}),\ t_2 = \tan(\dfrac{\omega_u-\omega_l}{2}) & \text{intermediate values} \\ k = t_1/t_2,\ g = -1,\ c = \dfrac{2ak}{k+1},\ d = \dfrac{k-1}{k+1} & \text{lowpass-to-bandpass} \\ k = t_1 \cdot t_2,\ g = 1,\ c = \dfrac{2a}{k+1},\ d = \dfrac{1-k}{1+k} & \text{lowpass-to-bandreject} \\ \hline \end{array} \tag{4}$$

A Note on the g-Factor

When the mapping is lowpass-to-lowpass or lowpass-to-bandreject, then $g = 1$ so it actually could be dropped whenever it occurs as multiplier. In the lowpass-to-highpass and lowpass-to-bandpass mapping, $g = -1$, so a g^2-factor could be dropped and a simple g-factor could be replaced by a unary minus. However, in the following, g will be written out nonetheless for consistency and generality. At some later stage, one might consider complex valued filters in which $g = e^{j\theta}$, which means that g could be any unit-magnitude complex number. When the filter is to be real valued, the only possible choices for θ are $\theta = n\pi, n \in \mathbb{Z}$ such that $g = \pm 1$.

Interpretation 1 - Mapping of Poles and Zeros

We consider a general z-domain transfer function given in product form:

$$H(z) = k\frac{\prod_{m=1}^{M}(1 - q_m z^{-1})}{\prod_{n=1}^{N}(1 - p_n z^{-1})} \tag{5}$$

where k is an overall gain factor and q_m, p_n are the zeros and poles respectively. In this transfer function, we now have to substitute every occurrence of z^{-1} with the appropriate expression $A(z)$. When we do this, the poles and zeros of the resulting transfer function will have mapped to a new position. To find out the new positions of the poles and zeros, it will be sufficient to consider a general factor inside one of these two products, say $(1 - rz^{-1})$ where r is either one of the poles or one of the zeros such that we have $(1 - rz^{-1}) = 0$. We write this factor in the transfer function as:

$$G(z) = 1 - rz^{-1} \tag{6}$$

First Order Mapping

Now, let's substitute z^{-1} with the first order substitution $A_1(z)$ to obtain the warped transfer function $G'(z)$:

$$G'(z) = 1 - rA_1(z) = 1 - rg\frac{c + z^{-1}}{1 + cz^{-1}} \tag{7}$$

We want to find those values of z where the new factor of the transfer function becomes zero. So we require:

$$1 - rg\frac{c + z^{-1}}{1 + cz^{-1}} = 0 \tag{8}$$

solving this equation for z yields the value for z where $G'(z)$ becomes zero, so it is the root of the warped transfer function that corresponds to the root r of the original transfer function. We denote the warped root as r', and we obtain:

$$\boxed{r' = \frac{gr - c}{1 - gcr}} \tag{9}$$

This formula, to compute the warped root r' from the original root r, is recognized as a bilinear mapping from the z-plane to itself. Thus, we may interpret the substitution of z^{-1} with $A(z)$ as a mapping of the poles and zeros of the filter to new positions.

Second Order Mapping

For the second order transformations, we require:

$$1 - rA_2(z) = 1 - rg\frac{d + cz^{-1} + z^{-2}}{1 + cz^{-1} + dz^{-2}} = 0 \tag{10}$$

This leads to a quadratic equation for the new roots r', so we obtain two new roots r'_1, r'_2 for each original root r. These can be calculated via:

$$\boxed{\begin{array}{l} k_1 = 1/(2dgr - 2),\ k_2 = k_1 c(1 - gr),\ k_3 = c^2 - 4d,\ k_4 = k_1\sqrt{k_3(1 + g^2r^2) + (4d^2 - 2c^2 + 4)gr} \\ r'_{1,2} = k_2 \pm k_4 \end{array}} \tag{11}$$

Gain Adjustment

When we map the poles and zeros to their new positions, we typically want to match the DC-gain of the lowpass prototype at some critical frequency ω_c in the mapped filter. For example, a bandpass should have the same gain at its center frequency as the lowpass prototype at DC. To achieve that, we must also adjust the gain factor k in our general transfer function (5). To find the new value for k, which we shall denote as k', we first evaluate the DC magnitude k_0 of the unwarped prototype filter, excluding the gain factor k:

$$\boxed{k_0 = \left|\frac{\prod_{m=1}^{M}(1 - q_m)}{\prod_{n=1}^{N}(1 - p_n)}\right|} \tag{12}$$

Then, we evaluate the magnitude k_w of the warped filter at its critical frequency ω_c (again excluding any gain factor):

$$\boxed{k_w = \left|\frac{\prod_{m=1}^{M'}(1 - q'_m e^{-j\omega_c})}{\prod_{n=1}^{N'}(1 - p'_n e^{-j\omega_c})}\right|} \tag{13}$$

where q'_m, p'_m are the warped zeros and poles and M', N' are their numbers (which might be different from M, N because second order mappings double the number of poles and zeros). Then we set:

$$\boxed{k' = k\frac{k_0}{k_w}} \tag{14}$$

For lowpass filters, we choose $\omega_c = 0$, for highpass filters we choose $\omega_c = \pi$, for bandreject filters we may choose either of these two (these choices also simplify the computation of the factors in the products). For bandpasses, we choose the warped center frequency which is given by: $\omega_c = 2\arctan\left(\sqrt{\tan(\omega_l/2)\tan(\omega_u/2)}\right)$.

Interpretation 2 - Replacing Unit Delays with Allpass Filters

The expressions that replace z^{-1} can be regarded as z-domain transfer functions in their own right, namely as those of a first and second order allpass filter, respectively. On the other hand, the term z^{-1} as such corresponds to a unit delay, which, by the way, may be regarded as a special case of an allpass filter, too. When we now replace all unit delays in our implementation of the filter's difference equation with the new allpass filter, we also obtain the frequency warped filter. A complication arises, when the to-be-replaced unit delay is inside a feedback path, because the allpass filter which replaces the unit delay contains a delay free path. This, in turn, translates to a delay-free feedback path in the overall filter. The problem can be solved in two ways, either by rewriting the difference equation in a way to eliminate the delay-free feedback path or to use an algorithm that actually implements the delay-free feedback (as proposed by Härmä in [2]). In the following, we will consider the former of these approaches, applied to first and second order filters.

Eliminating Delay-Free Feedback Paths

1st Order Filter With 1st Order Map

We consider the first order filter that implements the difference equation:

$$y[n] = b_0 x[n] + b_1 u[n] - a_1 p[n] \tag{15}$$

where $u[n], p[n]$ are intermediate signals. In the case of an unwarped filter, we would just have:

$$u[n] = x[n-1],\ p[n] = y[n-1] \qquad \text{unwarped case} \tag{16}$$

and we could implement the difference equation directly without any problems. Replacing z^{-1} with the first order allpass transfer function amounts to replace the unit delays with first order allpass sections. Our intermediate signals would now have to be calculated by first order difference equations as well:

$$\begin{aligned} u[n] =& g(cx[n] + x[n-1]) - cu[n-1] \qquad \text{warped} \\ p[n] =& g(cy[n] + y[n-1]) - cp[n-1] \qquad \text{case} \end{aligned} \tag{17}$$

which reduces to the unwarped case for $g = 1, c = 0$. For $c \neq 0$ we see, that these allpass sections contain a delay-free path. Specifically, we note that the computation of $p[n]$ requires knowledge of $y[n]$. Substituting $u[n], p[n]$ back into (15) yields:

$$y[n] = b_0 x[n] + b_1(g(cx[n] + x[n-1]) - cu[n-1]) - a_1(g(cy[n] + y[n-1]) - cp[n-1]) \tag{18}$$

This is a difference equation where $y[n]$ appears on left hand side as well as on the right hand side. We may, however, isolate $y[n]$ on the left hand side by simple algebraic manipulations. To simplify the notation, we define $x = x[n], x_1 = x[n-1]$, with analogous definitions for all other signals y, u, p. So, we may restate the equation above as:

$$y = b_0 x + b_1(g(cx + x_1) - cu_1) - a_1(g(cy + y_1) - cp_1) \tag{19}$$

Now, isolating y on the left hand side gives:

$$y = \frac{(b_0 + b_1 cg)x + b_1(gx_1 - cu_1) - a_1(gy_1 - cp_1)}{1 + a_1 cg} \tag{20}$$

For implementation purposes, we may streamline this difference equation to:

$$y = b'_0 x + b'_1(gx_1 - cu_1) - a'_1(gy_1 - cp_1) \tag{21}$$

when we define:

$$k = 1/(1 + a_1 cg),\ b'_0 = k(b_0 + b_1 cg),\ b'_1 = kb_1,\ a'_1 = ka_1 \tag{22}$$

The overall algorithm for implementing the first order filter that is warped by a first order allpass mapping may now be stated as:

For each incoming sample x, do:

1. compute output signal y via (21):
   $y \leftarrow b'_0 x + b'_1(gx_1 - cu_1) - a'_1(gy_1 - cp_1)$
2. update internal state variables u_1, p_1 via (17):
   $u_1 \leftarrow g(cx + x_1) - cu_1$
   $p_1 \leftarrow g(cy + y_1) - cp_1$
3. update internal state variables x_1, y_1 via:
   $x_1 \leftarrow x,\ y_1 \leftarrow y$

2nd Order Filter With 2nd Order Map

We consider the second order filter that implements the difference equation:

$$y[n] = b_0 x[n] + b_1 u[n] + b_2 v[n] - a_1 p[n] - a_2 q[n] \tag{23}$$

where $u[n], v[n], p[n], q[n]$ are intermediate signals. As in the first order case above, we will use the simplified notation that replaces the $[n - i]$ discrete time indices with a subscript i which may be dropped when it is equal to zero, so rewrite the difference equation as:

$$y = b_0 x + b_1 u + b_2 v - a_1 p - a_2 q \tag{24}$$

In the case of an unwarped filter, we would just have:

$$u = x_1,\ v = x_2,\ p = y_1,\ q = y_2 \qquad \text{unwarped case} \tag{25}$$

To this second order filter, we now apply a second order allpass mapping. Note that the order of the filter and the order of the mapping are actually independent of each other. It's only by accident that they are equal in the both cases that we consider here. The mapping results in the intermediate signals:

$$\begin{aligned} u &= g(dx + cx_1 + x_2) - cu_1 - du_2 \\ v &= g(du + cu_1 + u_2) - cv_1 - dv_2 \qquad \text{warped} \\ p &= g(dy + cy_1 + y_2) - cp_1 - dp_2 \qquad \text{case} \\ q &= g(dp + cp_1 + p_2) - cq_1 - dq_2 \end{aligned} \tag{26}$$

Among these intermediate signals, only u and v can be calculated directly when a new input value x arrives. The signal p depends on y and q depends on p and hence indirectly also on y. To find an explicit expression for y, we substitute the expressions for p, q back into the difference equation (24):

$$\begin{aligned} y = b_0 x + b_1 u + b_2 v \\ - a_1(g(dy + cy_1 + y_2) - cp_1 - dp_2) \\ - a_2(g(dp + cp_1 + p_2) - cq_1 - dq_2) \end{aligned} \tag{27}$$

In the last summand we still have an implicit (undelayed) y, namely inside the p. So we must substitute this p with the right hand side of the 3rd line of (26) to arrive at:

$$\begin{aligned} y = b_0 x + b_1 u + b_2 v \\ - a_1(g(dy + cy_1 + y_2) - cp_1 - dp_2) \\ - a_2(g(d(g(dy + cy_1 + y_2) - cp_1 - dp_2) + cp_1 + p_2) - cq_1 - dq_2) \end{aligned} \tag{28}$$

Having resolved all implicit occurrences of y on the right hand side into explicit ones, we are now once again going to isolate y on the left hand side. After some algebraic simplifications, this results in:

$$y = k(b_0 x + b_1 u + b_2 v - a_1 p' - a_2 q') \tag{29}$$

where:

$$\begin{aligned} k &= 1/(1 + a_1 gd + a_2 g^2 d^2),\ k_1 = dc - c,\ k_2 = d^2 - 1 \\ r &= g(cy_1 + y_2) \\ p' &= r - cp_1 - dp_2 \\ q' &= g(dr - k_1 p_1 - k_2 p_2) - cq_1 - dq_2 \end{aligned} \tag{30}$$

In these formulas, k, k_1, k_2 are some constant coefficients that may be precomputed when the filter is set up and r, p', q' are some further intermediate signals that we have defined for convenience. We may now state the overall algorithm for the second order filter with a second order warping map as:

For each incoming sample x, do:

1. compute signals u, v via lines 1,2 of (26):
   $u \leftarrow g(dx + cx_1 + x_2) - cu_1 - du_2$
   $v \leftarrow g(du + cu_1 + u_2) - cv_1 - dv_2$

2. compute intermediate signals p', q' via (30):
   $r \leftarrow g(cy_1 + y_2)$
   $p' \leftarrow r - cp_1 - dp_2$
   $q' \leftarrow g(dr - k_1 p_1 - k_2 p_2) - cq_1 - dq_2$

3. compute output signal y via (29):
   $y \leftarrow k(b_0 x + b_1 u + b_2 v - a_1 p' - a_2 q')$

4. compute signals p, q via lines 3,4 of (26):
   $p \leftarrow g(dy + cy_1 + y_2) - cp_1 - dp_2$
   $q \leftarrow g(dp + cp_1 + p_2) - cq_1 - dq_2$

5. update internal states:
   $x_2 \leftarrow x_1, x_1 \leftarrow x, y_2 \leftarrow y_1, y_1 \leftarrow y$
   $u_2 \leftarrow u_1, u_1 \leftarrow u, v_2 \leftarrow v_1, v_1 \leftarrow v$
   $p_2 \leftarrow p_1, p_1 \leftarrow p, q_2 \leftarrow q_1, q_1 \leftarrow q$

2nd Order Filter With 1st Order Map

Without derivation, we give the algorithm for a second order filter with a first order map. Let

$$k = 1/(1 + a_1 gc + a_2 g^2 c^2) \tag{31}$$

For each incoming sample x, do:

1. compute signals u, v:
   $u \leftarrow g(cx + x_1) - cu_1$
   $v \leftarrow g(cu + u_1) - cv_1$

2. compute intermediate signals p', q':
   $p' \leftarrow gy_1 - cp_1$
   $q' \leftarrow g(cp' + p_1) - cq_1$

3. compute output signal y:
   $y \leftarrow k(b_0 x + b_1 u + b_2 v - a_1 p' - a_2 q')$

4. compute signals p, q:
   $p \leftarrow g(cy + y_1) - cp_1$
   $q \leftarrow g(cp + p_1) - cq_1$

5. update internal states:
   $x_2 \leftarrow x_1, x_1 \leftarrow x, y_2 \leftarrow y_1, y_1 \leftarrow y$
   $u_1 \leftarrow u, v_1 \leftarrow v, p_1 \leftarrow p, q_1 \leftarrow q$

We don't give an algorithm for the 1st order filter with 2nd order map, because the 2nd order filter with 2nd order map already includes this as special case (for $a_2 = b_2 = 0$). Algorithms how to compute outputs of arbitrary order filters can be found in [2]. We note, however, that higher order filters are often implemented as cascades of 2nd order sections anyway. If such a biquad-cascade implementation is used, the algorithms given here should be all that is needed.

[todo: perhaps more efficient algorithms can be obtained when starting from direct form 2 (instead of direct form 1 as was done here), because the DF2 has only 2 delay-elements]

References

[1] A.G. Constantinides. Spectral transformations for digital filters. Proc. Inst. Elec. Eng., vol. 117, pp. 1585-1590, Aug. 1970

[2] Aki Härmä. Implementation of frequency-warped recursive filters. Signal Processing 80 (2000) 543-548