EE513 Project 2: A Digital Oscillator

Implementor's Notes

Matt Ruffner
matthew.ruffner@uky.edu

ABSTRACT

In this assignment, a multi-frequency digital oscillator was created utilizing the un-damped exponential impulse response of a marginally stable IIR filter. Second Order Sections are generated for each frequency of oscillation requested. These are then parallelized and converted to Direct Form coefficients representing the entire Transfer Function (TF).

1. APPROACH

The discrete time representation of a marginally stable transfer function can be seen in Eq. 1. Taking the Z-Transform yields Eq. 2.

$$y[n] = 2cos(\alpha)y[n-1] - y[n-2] + kx[n] \tag{1}$$

$$\hat{H}(z) = \frac{\hat{Y}(z)}{\hat{X}(z)} = \frac{Kz^2}{z^2 - 2cos(\alpha) + 1} \tag{2}$$

Where K represents the relative sinusoid amplitude, and α controls the frequency of oscillation.

$$\alpha = \frac{2\pi f_o}{f_s} \tag{3}$$

It turns out that in order to normalize the sinusoid amplitudes realtive to each other, all that is required is a scaling by alpha. Letting K represent the final coefficient in the TF, and k be the function input amplitude factor, Eq. 4 follows.

$$K = \alpha k \tag{4}$$

From this, the function with prototype seen in in Listing 1 is derived. A test MATLAB script, `digosc_test.m` is also supplied, which verifies the correct operation via spectral analysis. Parameters to the function are shown in Listing 1.

Listing 1: DIGOSC function prototype.

```
function [a,b] = digosc(freqs, amps, fs)
```

ACM ISBN N.A..
DOI: N.A.

2. MATLAB SOURCE

Listing 2: Relevant code from the project source file `digosc.m`

```
% vector representing coeffecients for normalizing the calculation of frequencies and amplitudes
alphas=(2*pi.*freqs)/fs;

% where the TF of the digital oscillator is
% (k*z^2) / (z^2 - 2*cos(alpha)*z + 1)

bs=[]; % vector to hold numerators
counter=1;
for n=amps % iterate through each specified amplitude
    bs=[bs; alphas(counter)*n 0 0]; % multiply coefficient for z^2 term by desired amplitude
    counter=counter+1;
end

as=[]; % vector to hold denominators
for n=alphas
    % create denominators based on calculated alpha values
    as=[as; 1 -2*cos(n) 1];
end

dfilts = []; % vector to hold digital filter representations
for n=1:nfreqs % convert b/a num/denom coeffs to dfilt objects
   % build vector of direct form 1 representations
   dfilts=[dfilts, dfilt.df1(bs(n,1:end), as(n,1:end))];
end

% parallelize the individual transfer functions
fdf=dfilt.parallel(dfilts);

% extract the state space variables of the parallelized TFs
[A,B,C,D]=fdf.ss;

% use ss2tf to convert to one TF with unified numerators and denominators
[a,b]=ss2tf(A, B, C, D);
```