Section III – Source Code

Filename sampled\_spectrum.m

function [discreteSpecFreq,discreteSpecAmp] = sampled\_spectrum(specFreq,specAmp,fs);

for k = 1:length(specAmp);

if (specFreq(k) > (fs/2))

while (specFreq(k) > fs/2)

specFreq(k) = specFreq(k) - fs;

end;

elseif (specFreq(k) < (-fs/2))

while (specFreq(k) < (-fs/2))

specFreq(k) = specFreq(k) + fs;

end;

end;

end;

discreteSpecFreq = (specFreq\*2\*pi/(fs));

discreteSpecAmp = specAmp;

Filename reconstructed\_spectrum.m

function [specFreq,specAmp] = reconstructed\_spectrum(discreteSpecFreq,discreteSpecAmp,fs);

specFreq = ((discreteSpecFreq\*fs)/(2\*pi));

specAmp = discreteSpecAmp;

Filename idealLowpass.m

function [x,y] = idealLowpass(specFreqIn,specAmpIn,fco);

% Ideal Lowpass

% Cuts any incoming signals above the specified frequency

%

% specFreqIn = An array of Omega Values

% specAmpIn = An array of Amplitude values of form A\*exp(xj)

% fco = Cutoff Frequency

%

% See also: idealHighPass, idealBandPass

%

% Ver. 1.1 by BR

for k = 1:length(specAmpIn);

if (abs(specFreqIn(k)) >= fco)

specAmpIn(k) = 0;

end;

end;

x = specFreqIn;

y = specAmpIn;