Generating Signals for Simulation and Test of Complex DSP Systems

Laszlo Hars
Philips Research
Email: Laszlo.Hars@Philips.com

***Abstract*:** Simulation and prototype test signal generation is discussed for an example decimator-mixer DSP system, which exhibits most of the general test issues, like mirror image rejection and frequency response. Practical precision considerations and design algorithms are presented and a calibration procedure is described for high accuracy physical test-signal generation.

Introduction

When designing or building DSP systems, which are so complex that exact mathematical descriptions are impractical or impossible, one has to resort to simulation and testing of prototypes. This is also a good way to test theoretical models for correctness.

We chose, as an example, testing high decimation DSP systems with frequency shifting and quadrature conversion. This type of system presents most of the difficulties arising in simulation and test signal generation in general. The flatness of the frequency response in the useful frequency band and the amount of the alias rejection have to be checked. The alias rejection tells us how large a part of the disturbing signals will get to the useful band. The actual disturbing effects are very much dependent on the type of the disturbance. If it is noise-like, the powers of the different frequency disturbing signals aliased (mirrored) to the same frequency get added together. If the disturbance consists of phase related components, like cross talk or mixer feed-through, some signals may have aliases with equal frequency. Their amplitudes get added together. If the disturbance consists of a few signals of discrete frequencies, like for radio receivers when signals of other transmitters reach our input too, the smallest attenuation of the unwanted frequencies best characterizes the alias rejection.

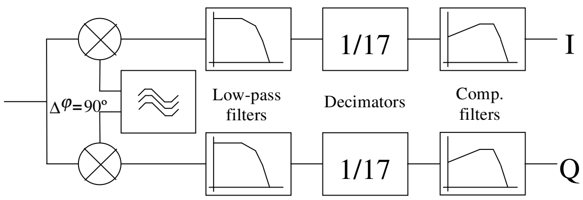
The simulation signals we describe provide enough information to estimate all of the above types of alias rejection. We provide signal generation and evaluation algorithms (in the form of MATLAB scripts) for each of these cases along with the test of the flatness of the frequency response in the pass-band and the slopes of the attenuation curve around the cut off frequency (at the edges of the pass-band).

The simulation signals we propose are sums of a large number of sine waves. If their phases are not chosen carefully the peak amplitude (crest factor) can become very large. We propose a simple and effective algorithm to keep the crest factor modest. Without this the simulation would require high precision arithmetic.

For testing prototype hardware the simulation signals can be turned to physical ones with the help of digital to analog converters. To smooth out the steps of the converter output precise interpolation filters are needed. Also, synchronization between signals and/or external events might be required. These functions are available in arbitrary waveform generators. However, close to their frequency limit the interpolation filters become less accurate, so calibration is necessary. We show how a spectrum analyzer can be used to collect calibration data. The digital signal samples get modified before feeding into the waveform generator to undo its filter roll-off and other inaccuracies. This way the waveform generators provide very high precision test signals.

An example DSP system

We consider an odd ratio decimation DSP system with frequency shifting and quadrature conversion designed for a GSM tester.



The sampling rate is 18.4167 MHz and the input frequency IF = 13.8125 MHz (ratio 4/3). The signal bandwidth is ±270.833 KHz with 20% transition band.

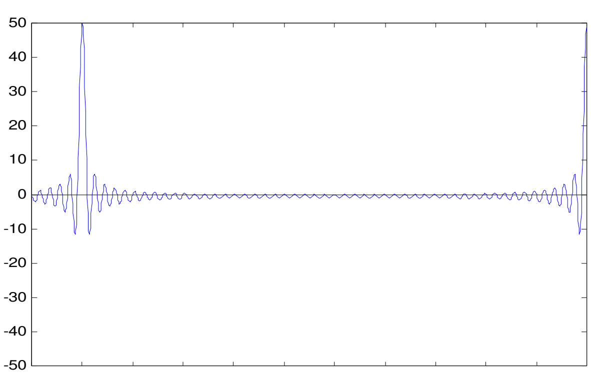
What we test

- Amplitude / frequency response
  - Pass-band ripple
  - Shape of transition band
  - Stop-band attenuation
- Alias superposition
- Alias rejection

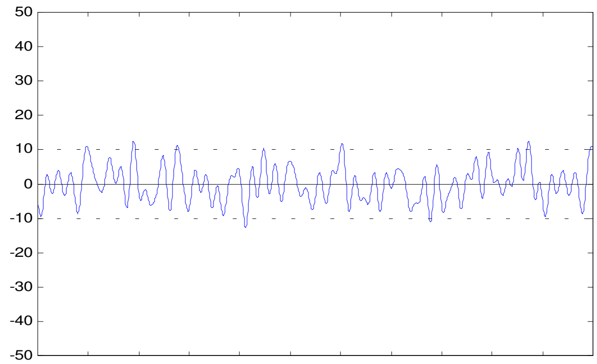
Multi-tone signals – Crest factor

We propose using the sum of a large number of sine waves for test signals. The resulting waveform, however, might have very large peak amplitude compared to the mean. One can scale the test signal to have unit amplitude, but at finite digital resolution the accuracy of the measurement will be reduced (large quantization noise).

Here is the resulting waveform of 50 cosine waves with 0º initial phase.

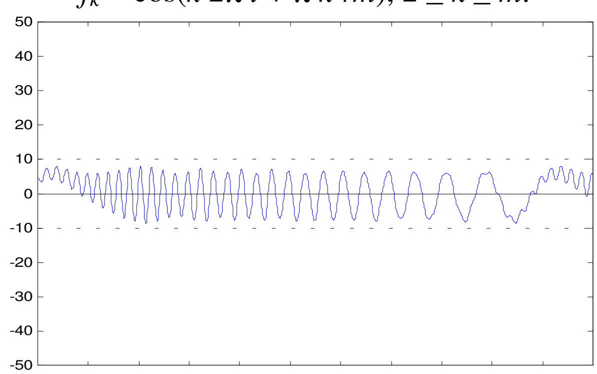


If the phase is chosen randomly in $[0,2\pi]$ for each of the compound signals, the peak amplitude is significantly reduced:



In the following case the squared-frequency-rule gives another factor of close to 2 improvement: *m* cosine waves added with initial phase proportional to the squared frequency

$$f_1 = \cos(2\pi \cdot t),\ t = 0 \ldots 1$$
$$f_k = \cos(k \cdot 2\pi \cdot t + \pi \cdot k^2/m),\ 2 \le k \le m.$$



If not all the integer frequencies are used in $1 \ldots m$, the above rule cannot be applied. A general-purpose optimization program could still search for a good phase set. On a fast PC (500 MHz PIII) 10-20 hours are enough to get a 10-20-fold decrease (4 bit resolution improvement).

Simulation

The length of the simulation sample set is chosen such that each signal component has an integer number of periods even after decimated by 17. This is important if we use FFT for the spectrum calculation: if a spectral component doesn't fall exactly on a basis frequency, the result of the FFT is difficult to interpret. There are problems with the spectral leakage, side lobes, peak drop, etc.

In order to make the frequency response of the decimation process apparent the first simulation signal is generated with frequency components around the IF, like a modulated carrier. Its spectrum is Hermitian-symmetric. We assign random phase to each spectral component, so that the crest factor won't be very large, and the simulation signal looks more realistic. The constants we use are:

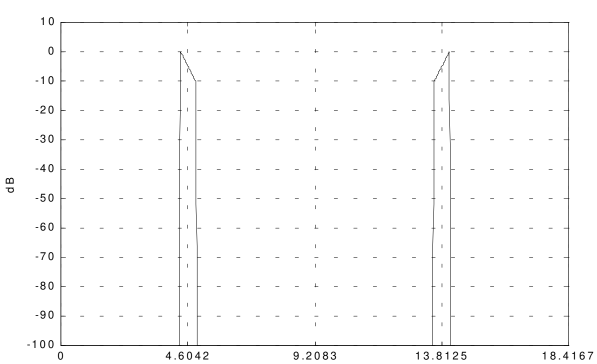
```
fb = 13e6/48;               % GSM bit frequency
fs = 68 * fb;                % sample frequency
fi = 51 * fb;                 % input frequency
fe =[-1.2 -1 1 1.2]*fb+fi;  % freq band edges
ge =[-80 -10 0 -80]/20;     %   signal levels
nf = 500*17;                % numb of samples
tt = (0:nf-1) / fs;         % sampling points
ff = (0:nf-1)/nf * fs;      % DFT frequencies
fd = ff - fs/2;             %for signed freqs
```

Generate the frequencies:

```
pf = eps + zeros(1,nf); % eps to avoid log 0
for i = 2 : length(fe)
    j = (ff > fe(i-1) & ff <= fe(i));
    pf(j) = pf(j)+ 10.^
      (ge(i)+(ge(i-1)-ge(i))*
      (ff(j)/fe(i)-1)/(fe(i-1)/fe(i)-1));
end
rand('seed',0);                  % Random phase
ph = rand(1,nf/2-1);
pf(2:nf)=(pf(2:nf)+pf(end:-1:2)).*
          exp(1i*2*pi*[ph 0 -ph(end:-1:1)]);
```

The simulation signal is calculated from its spectrum by inverse FFT. The finite precision arithmetic cause small imaginary components, which must be removed:

```
x = real(ifft(pf));
```



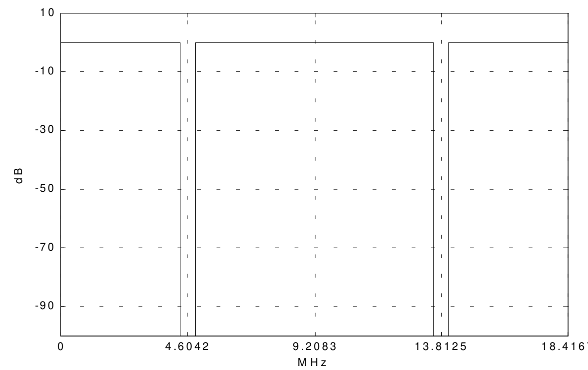
Spectrum of the 1st simulation signal

Note the slanted top of the spectral lobes: they help to check the phase state after the signal processing.

The second data set has the complementary spectrum: magnitude 1 everywhere outside of the useful band. Here the random phases also simulate noise-like disturbances.

```
pg = ones(1,nf);
j  = (ff>fe(2) & ff<=fe(end-1)) |
     (ff>fs-fe(end-1) & ff<=fs-fe(2));
pg(j) = eps;

ph = rand(1,nf/2-1);           % random phase
pg(2:nf)=pg(2:nf).*
         exp(1i*2*pi*[ph 0 -ph(end:-1:1)]);
y = real(ifft(pg));
```



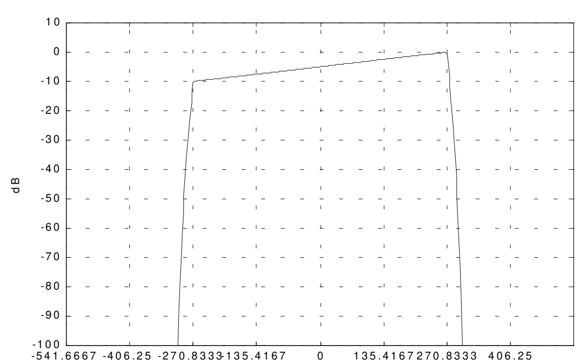
Spectrum of the 2nd simulation signal

I/Q split starts the decimation process:

```
I = zeros(1,nf);  Q = I;
I(1:4:end) = x(1:4:end);
I(3:4:end) =-x(3:4:end);
Q(2:4:end) = x(2:4:end);
Q(4:4:end) =-x(4:4:end);
```

The next step is low-pass filtering and decimation by 17. Since the test signal is periodic, a more accurate filter procedure is applied, **`circfilt`**: extend the signal before filtering and cut off the tails afterwards.

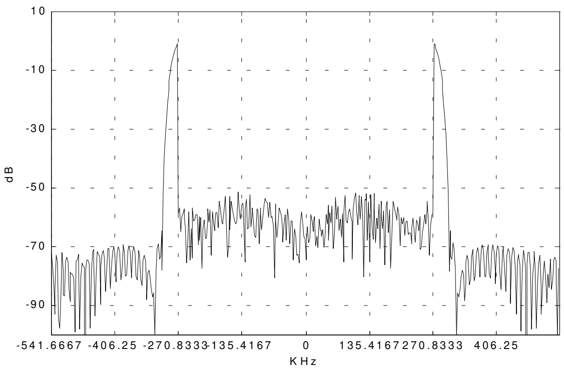
```
I1  = circfilt( low_pass, I);
Q1  = circfilt( low_pass, Q);
Sig = circfilt( comp_filt,
                I1(1:17:nf)+1i*Q1(1:17:nf));
```



Spectrum of the decimated base-band 1st signal

The spectrum of the 1st signal is preserved without any noticeable distortion.

Repeat the same signal processing with the second data set to see the alias rejection and the stop band-transition band response.



Spectrum of the decimated 2nd signal

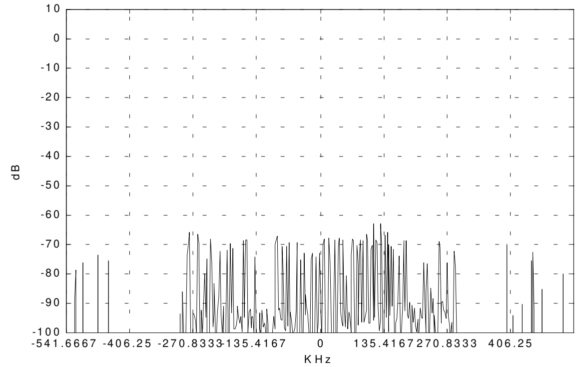
The figure shows that all disturbances together from the whole frequency band outside of the signal band don't cause more disturbances than -53dB below their original power. In the figure each spectral component is the sum of 16 aliased disturbing components of unit power (decimation by 17). Each disturbing signal component at a certain frequency is better attenuated, but they add up with random phase. When a large number of aliased components are added together, the resulting power is the square root of the amplitude sum.

In the transition band there isn't much attenuation. If it is necessary the low-pass filter should be designed to be sharper.

For a final test a third data set is generated. In the stop band it has many (99) random spectral lines of magnitude 1 with random phase and some (-100dB) white noise.

```
ix = find(ff > fs/2 & (ff < fe(1) |
         ff >= fe(end)));
pe = zeros(1,nf);
pe(ix(1+round(rand(1,99)*(length(ix)-1))))=1;
pe = pe + 1e-5*abs(randn(1,nf));
ph = rand(1,nf/2-1);
pe(2:nf)=(pe(2:nf)+pe(end:-1:2)).*
         exp(1i*2*pi*[ph 0 -ph(end:-1:1)]);
z = real(ifft(pe));
```

After the decimation



Decimation effects on the discrete spectral line simulation

Again, where different spectral components alias onto the same decimated spectral frequency (in the figure it occurs at a few points), the alias rejection seems to be less.

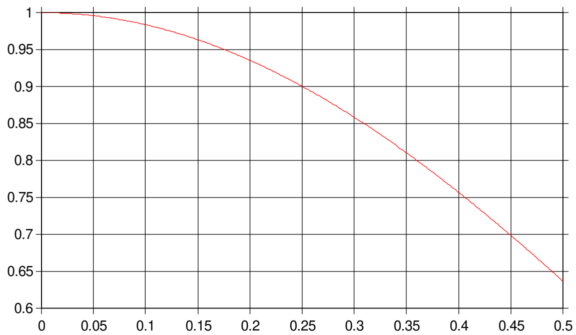
Physical test signals

For testing prototype hardware the simulation signals are turned to physical ones with the help of digital to analog converters usually inside of arbitrary waveform generators; the latter also provide synchronization between signals and/or external events. To smooth out the steps of the converter output precise interpolation filters are needed. Close to the cut off frequency these interpolation filters become less accurate, so usually calibration is necessary. We show how a spectrum analyzer is used to collect calibration data.

We modify the test signal samples before feeding them to the waveform generator undoing the filter roll-off and other inaccuracies to get very high precision test signals.

Amplitude / Frequency response

In the case of ideal staircase D/A conversion, the signal amplitude has a roll-off of the main lobe of the sinc function:



By appropriately filtering the digital data we compensate it. Higher frequency components get emphasized. A/D converters have finite slew rate, so the amplitude / frequency response is somewhat different. Only measurement data is available.

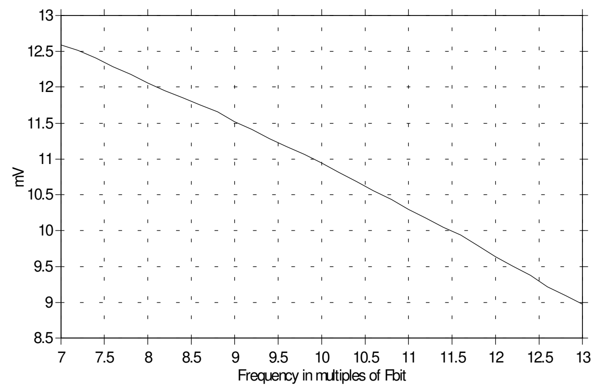
If we must measure the amplitude roll-off anyway, we can include the analog interpolation (linear phase, Bessel) filter smoothing the A/D steps in the calibration for free. We measure the overall amplitude / frequency response by generating a comb-like signal, the sum of several equal amplitude sine waves in the desired band and measure the peaks of the frequency components using a spectrum analyzer. (Filters and power meters are much more expensive, cumbersome and usually less exact.) The absolute accuracy of spectrum analyzers is not very high, but the displayed values of same level, different frequency signals in one sweep do not differ significantly.

One can verify this with a sine wave oscillator by setting up max hold on the spectrum analyzer and manually tuning the center frequency of the spectrum analyzer by moving the displayed peak from one edge of the screen to the other. One should see a horizontal straight line representing the held peaks. A typical spectrum analyzer produces a ripple of this line less than ±0.02 mV at an input signal of 12 mV. We choose the reference level such that the line appears on the top of the screen.

With the comb signal, we set the spectrum analyzer as follows:

- sweep = 9.2 MHz (Nyquist frequency)
- resolution bandwidth = 100 KHz (a very large value giving accurate amplitude peaks, such that the maximum can be read reliably)
- video bandwidth = 300 Hz (a very small value to slow down the sweep, giving an averaging-like effect, which makes the displayed curve more accurate)

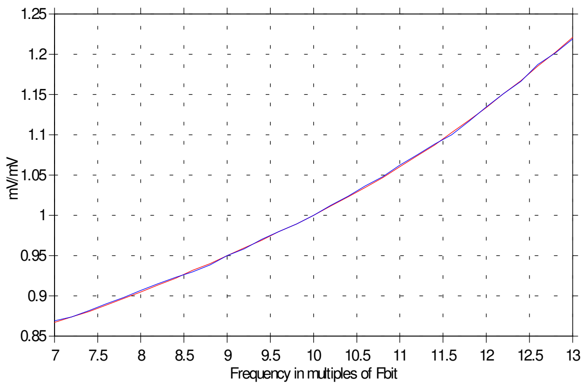
One can read the peak levels by positioning the marker on them.



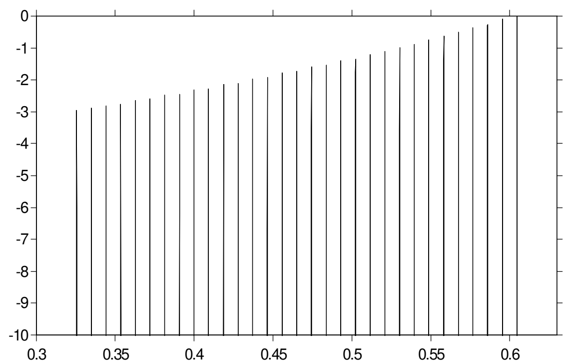
We design the corresponding pre-compensation filter with the Remez exchange algorithm. It requires alternating transition bands (with no restriction on the gain) and specified gain bands (with linear amplitude change). Approximating a general curve requires duplicating the frequency points with a little shift (imitating very short transition bands) and a piece-wise linear interpolation function for the gain / frequency curve elsewhere. The resulting filter is usually short and accurate:

```
n  = 12;                          % filter length
k  = 31;                          % # data points
ff = zeros(1,2*k-2);
ff(1:2:2*k-3) = frq(1:k-1);
ff(2:2:2*k-2) = frq(2:k)-1e-4;
ll = zeros(1,2*k-2);
ll(1:2:2*k-3)=lev(1:k-1);
ll(2:2:2*k-2)=lev(2:k);
b = remez(n,[0  0.01    ff      0.99  1],
            [1   1  ll.\lev(k/2)  1    1]);
f = abs(freqz(b,1,frq*pi));
```

```
plot(fc,abs(f),'r', fc,lev.\lev(k/2),'b')
ytick(0.85:0.05:1.25,':')
xtick(7:0.5:13,':')
ylabel('mV/mV')
xlabel('Frequency in multiples of Fbit')
```



The pre-compensation changes the spectrum as required:



After loading the samples generated in this way into the waveform generator, the resulting amplitude peaks are measured again with the spectrum analyzer. In our cases they were between 12.35 and 12.55 mV, that is the amplitude response was within ±0.8% flat (±0.07 dB AM ripple). For higher accuracy one can adjust the calibration data and rerun the filter design.

Bibliography

[1] A. D. Poularikas, *Fomulas and Tables for Signal Processing*, CRC Press – IEEE Press, 1999.

[2] N. Kalouptsidis, *Signal Processing Systems: Theory and Design*, John Wiley & Sons, 1997.

[3] J. G. Proakis, D. G. Manolakis: *Digital Signal Processing: Principles, Algorithms and Applications*, 3rd Edition, 1996 Prentice Hall.

[4] N. Fliege, Multirate Digital Signal Processing: Multirate Systems, filter Banks, Wavelets, John Wiley & Sons, 1995.

[5] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: *Numerical Recipes in C,* Cambridge University Press, 1992.

[6] L. R. Rabiner, B. Gold: *Theory and Application of the Digital Signal Processing*, 1975 Prentice Hall.

[7] A. V. Oppenheim, R. W. Schafer: *Digital Signal Processing*, 1975 Prentice Hall.