**Overview**

- Delta-Sigma Toolbox – some of the key functions
- 2nd-order DT modulator

---

**synthesizeNTF**

synthesizeNTF \( \rightarrow \) finds an NTF with specified order and out-of-band gain (\( H_{\text{inf}} \)), having: 1) optimized zeros (if desired), and 2) poles of a maximally-flat all-pole transfer function

\[
\text{order} = 5; \quad \text{OSR} = 64; \quad \text{opt} = 1; \quad H_{\text{inf}} = 1.5; \quad H = \text{synthesizeNTF(order, OSR, opt, H_{inf});}
\]

\[
f = \text{linspace}(0, 0.5, 1000); \quad z = \exp(2i\pi f); \quad \text{plot}(f, \text{dbv(evalTF(H,z))}); \quad \text{sigma}_H = \text{dbv(rmsGain(H, 0, 0.5/OSR))};
\]

Assuming \( \sigma^2_H = 1/3 = -4.8\text{dB} \), we obtain:

\[
\sigma^2_{\text{err, low}} = \frac{\sigma^2_{\text{err, low}}}{\sigma^2_H} = -118\text{dB}
\]
Another limitation is when $|H|$ is close to unity and zeros are optimized. With a low value of $|H|$, the poles of H converge to $z=1$. If all zeros of H are at $z=1$, $|H|$ approaches 1 and if the zeros are optimal, $|H|$ does not converge to 1 any more – this problems are due to the fact that poles and zeros are optimized separately (zeros taken from previous table), and not taking each other into account.

If $|H|$ is close to 1 or OSR is low → use `synthesizeChebyshevNTF` → still not optimal, but better than the standard `synthesizeNTF`.

Some theory

Take $P(z) = n^{th}$-order polynomial, with $|P(e^{j\omega})|$ maximally flat around $\omega=0$.

$P(z)$ will be the denominator of the NTF.

The coefficients of $P(z)$ are real.

Maximally flat around $z=1$ →

$$P(z)\left(\frac{1}{z}\right) = P(1) + a(z-1)^n \left(\frac{1}{z}\right)^n$$

If $a=0$ → $P$ is constant (very flat!); a larger $a$ yields a lowpass function with increasing sharpness; $a$ must be positive, otherwise $P(z)$ has a decreasing magnitude when moving away from $z=1$ (the opposite of what we want); taking $P(1)=1$ we obtain

$$P(z)\left(\frac{1}{z}\right) = 1 + a(z-1)^n$$

The product of the two roots of each equation is 1 → one root is inside the unit circle, the other outside → collecting all roots inside the circle yields the poles of the desired NTF.

Some theory – II

The roots of $a(z-1)^{2n} + (-z)^n = 0$ are the poles (and their inverse) of the desired NTF

$$a(z-1)^{2n} + (-z)^n = 0 \rightarrow e^{j2\pi k}a(z-1)^{2n} = e^{-j2\pi k}(-z)^n \rightarrow e^{j2\pi k}+\frac{1}{a^n}(z-1)^2 = -z$$

Thus, the roots are given by the $n$ complex quadratic equations

$$z^2 + h_k z + 1 = 0, \quad h_k = \frac{e^{-j(2\pi k)}}{a^n} - 2, \quad k = 0...n-1$$

The product of the two roots of each equation is 1 → one root is inside the unit circle, the other outside → collecting all roots inside the circle yields the poles of the desired NTF.

Example of DT ΔΣ modulator

`simulateDSM` → time-domain simulations of the modulator found with `synthesizeNTF`, assuming that STF is unity.

```matlab
OSR = 64;
nLev = 3; % number of levels in the quantizer
Nfft = 2^13; tone_bin = 57;
t = [0:Nfft-1];
u = 0.5*(nLev-1)*sin(2*pi*tone_bin/Nfft);
v = simulateDSM(u, H, nLev);
n = 1:350;
stairs(t(n), u(n), 'r'); hold on;
stairs(t(n), v(n), 'b');
```

Example of DT ΔΣ modulator
simulateSNR

**calculateSNR:**

**simulateSNR** → the amplitude of the input signal is swept in this call (however, this function does seem to have problems!)

\[
\text{OSR} = 64; \\
\text{nLev} = 3; \\
\text{amp} = [-130:5:-20, -17:2:-1]; \\
\text{snr} = \text{simulateSNR}(H, \text{OSR}, \text{amp}, [], \text{nLev}); \\
\text{plot}(\text{amp}, \text{snr}, '-b', \text{amp}, \text{snr}, 'db'); \\
[\text{pk}\_\text{snr}, \text{pk}\_\text{amp}] = \text{peakSNR}(\text{snr}, \text{amp})
\]

---

**realizeNTF** and associated functions

The synthesized NTF (and STF) are mapped here to a CRFB modulator with **realizeNTF** (synthesizeNTF returns STF=1 – setting all \(b_i\) except \(b_1\) to zero, we obtain a maximally-flat all-pole STF, see here below)

\[
H = \text{synthesizeNTF}(5, 64, 1); \\
f = \text{ linspace}(0, 0.5, 10000); \\
z = \exp(2i\pi f); \\
\text{magHa} = \text{dbv}(\text{evalTF}(H, z)); \\
\text{magGa} = \text{dbv}(\text{evalTF}(G, z)); \\
\text{plot}(f, \text{magHa}, 'b', f, \text{magGa}, 'm', 'Linewidth', 1)
\]

---

**Scaling of dynamic range**

**realizeNTF** returns unscaled coefficients → NTF and STF are ok, but there is no control of the internal states (i.e. integrator outputs) → dynamic range scaling is a must – this is an issue common to all active filter implementations!

Scaling is accomplished by dividing the admittance of all input branches of a given integrator by a factor \(k\), and multiplying with the same factor the admittance of all output branches of the same integrator – in this way, the rest of the circuit is unaffected, and so are the transfer functions

\[
\text{H} = \text{synthesizeNTF}(5, 64, 1); \\
\text{form} = '\text{CRFB}'; \\
[a, g, b, c] = \text{realizeNTF}(H, \text{form}); \\
b(2:end) = 0; \\
\text{Ha} = \text{NTF}, \text{Ga} = \text{STF}; \\
f = \text{linspace}(0, 0.5, 10000); \\
z = \exp(2i\pi f); \\
\text{magHa} = \text{dbv}(\text{evalTF}(\text{Ha}, z)); \\
\text{magGa} = \text{dbv}(\text{evalTF}(\text{Ga}, z)); \\
\text{plot}(f, \text{magHa}, 'b', f, \text{magGa}, 'm', 'Linewidth', 1)
\]

---

**scaleABCD**

scaleABCD → 1) determines maximum stable input amplitude (\(u_{max}\)) and maximum value for each modulator state for inputs up to \(u_{max}\); 2) dynamic range scaling is applied → maximum value of each state does not exceed the specified \(x\text{Lim}\) (remember: 0dBFS = nLev – 1)

mapABCD (inverse of stuffABCD) → maps the results in terms of coefficients for the desired topology

\[
\begin{array}{cccc}
\text{Scaled Coefficients:} \\
\hline
i & \alpha_i & \beta_i & b_i & c_i \\
\hline
1 & 0.0383 & 0.0004 & 0 & 0.1473 \\
2 & 0.0746 & 0.0045 & 0 & 0.2045 \\
3 & 0.0969 & 1 & 0 & 0.4097 \\
4 & 0.1758 & 1 & 0 & 0.4035 \\
5 & 0.1709 & 1 & 0 & 3.2333 \\
\end{array}
\]

\[
\text{nLev} = 3; \ x\text{Lim} = 0.9; \ f0 = 0; \\
[\text{ABCDs} \ u_{max}] = \text{scaleABCD}([\text{ABCDs}], \text{nLev}, f0, x\text{Lim}); \\
[a, g, b, c] = \text{mapABCD}([\text{ABCDs}], \text{form});
\]
Example of 2nd-order modulator

Low-speed modulator, e.g. for on-chip calibration engine; f_s=1kHz and f_s=1MHz → OSR=500, 1-bit DAC → SQNR ≈ 120dB → SNR in excess of 100dB possible
We assume that VDD is used as reference voltage (disregard supply noise)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>f_b</td>
<td>-1</td>
<td>kHz</td>
</tr>
<tr>
<td>Sampling Frequency</td>
<td>f_s</td>
<td>1</td>
<td>MHz</td>
</tr>
<tr>
<td>Signal-to-Noise Ratio</td>
<td>SNR</td>
<td>100</td>
<td>dB</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>VDD</td>
<td>3</td>
<td>V</td>
</tr>
</tbody>
</table>

We start selecting the standard CIFB topology

We assume that VDD is used as reference voltage (disregard supply noise)

Example of DT ΔΣ modulator

Code

```matlab
H = synthesizeNTF(2, 500, 0, 2); % out-of-band NTF peak gain = 2
form = "CIFB";
[a, g, b, c] = realizeNTF(H, form);
b(2:end)=0;
ABCD = stuffABCD(a, g, b, c, form);
[A B C D] = scaleABCD(ABCD);
% default: xLim=1, nLev=2
[a g b c] = mapABCD(ABCD, form);
```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2653</td>
<td>0.2212</td>
<td>0.3185</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5874</td>
</tr>
</tbody>
</table>

(c2 is not important in a single-bit quantizer)

Rounding →

a1 = 1/4, a2 = 1/4, b1 = 1/4, c1 = 1/3

Effect of rounding: peak NTF=2.25, peak SQNR=115dB

From simulations, effective quantizer gain (including c2) is approx. 16/3

Code and signal denormalization

```matlab
umax = 0.966  (normalized to Vref=VDD)  →  almost rail-to-rail input dynamic range
```

The toolbox assumes that the input of a binary modulator is between -1 and +1; the same for the integrator states after dynamic range scaling – everything is of course unit-less

In this example, the full-scale input is 3V pp, while that in the toolbox is 2pp. Let us assume that the amplifier supports a differential swing with the same numerical range as the toolbox, i.e. 2V pp, and that the digital signal v_d is interpreted as either 0 or 1

The relationship between circuit variables (v_in, v_x1, v_x2, v_d) and state variables (u, x1, x2, v) becomes:

```matlab
\[
\begin{align*}
&u = \frac{v_n - v_{in}}{3V/2} \quad \Rightarrow \quad [3 \ 0] \rightarrow [1 \ -1] \\
&x_1 = \frac{v_{in}}{1V} \quad \Rightarrow \quad [1 \ -1] \rightarrow [1 \ -1] \\
&v = \frac{2v_n - 1}{1V} \quad \Rightarrow \quad [1 \ 0] \rightarrow [1 \ -1]
\end{align*}
\]
```

Therefore the 1st integrator equation becomes

```
\[
\begin{align*}
x_i(n+1) &= x_i(n) + h u(n) - a_i v(n) = x_i(n) + \frac{1}{4} u(n) - \frac{1}{4} v(n) \\
v_i(n+1) &= v_i(n) + h v_0(n) - a_i [2v_i(n) - 1] = v_i(n) + \frac{v_0(n)}{6} - \frac{1}{2} v_i(n)
\end{align*}
\]
```

On the other hand, the SC circuit above approx. yields (more on this soon)

```
\[
\begin{align*}
v_i(n+1) &= v_i(n) + 2c_1 v_0(n) - 2 \frac{C_0}{C_2} v_i(n)
\end{align*}
\]
```
**Capacitance ratios**

Since $V_{dd}=3V$, we obtain the capacitance ratio $\frac{C_1}{C_2} = \frac{1}{12}$.

Notice that input capacitor and feedback capacitor are shared, i.e. are the same component.

This is not true for the second integrator – the same procedure yields the ratios as in the figure below.

**Signal – first integrator**

During ph1, $V_{ip}$ is transferred, inverted and without delay, to the bottom output through the bottom $C_1$.

During ph2, $V_{ip}$ is loaded onto the top $C_1$, and then transferred, non-inverted, to the top output through the top $C_1$ during next ph1 (i.e., delayed by 1 clock cycle).

The transfer function from input to differential output becomes

$$v_{sl,ip}(n) = v_{sl,ip}(n-1) + \frac{C_1}{C_i} v_{ip}(n-1) \rightarrow V_{sl,ip}(1-z^{-1}) = \frac{C_1}{C_i} z^{-1} v_{ip}$$

$$v_{sl,lv}(n) = v_{sl,lv}(n-1) - \frac{C_1}{C_i} v_{ip}(n) \rightarrow V_{sl,lv}(1-z^{-1}) = -\frac{C_1}{C_i} v_{ip}$$

This is the bilinear (trapezoidal) integration method mapping CT to DT → half delaying (Euler forward), half non-delaying (Euler backward) → better than either Euler methods.

**Noise – I**

Assuming the opamp noise negligible, the input-referred noise of the first integrator is

$$v_{in}^2 = kT \frac{C_1}{C_i}$$

Why? During ph1, a noise voltage $v_{sl,ip}$ is loaded on top $C_1$, while a noise voltage of $v_{sl,lv}$ and the signal $v_{in}$ are loaded on bottom $C_1$; during ph2 a noise voltage $v_{sl,ip}$ and the signal $v_{in}$ are loaded on top $C_1$, and the noise voltage $v_{sl,lv}$ is loaded on bottom $C_1$. Therefore, since all noise voltages are uncorrelated, at the integrator output we obtain:

$$v_{in}^2 = (v_{in} + v_{sl,ip} + v_{sl,lv})^2 = 4(v_{in}^2 + v_{ip}^2)$$

Thus, both signal and $kT/C$ noise are multiplied by the same factor, and therefore the input referred noise is simply $v_{in}$.

**Noise – II**

The $kT/C$ noise is white between DC and Nyquist → its in-band power is therefore

$$v_{in}^2 = \frac{v_{in}^2}{OSR}$$

In order to achieve an SNR of 100dB with a full-scale sine input, we must have

$$10^{10} = \left(\frac{1.5\,V}{2}\right)^2 \frac{2}{v_{in}^2} \rightarrow v_{in}^2 = (10\mu V)^2$$

If we allocate all noise to the first integrator, and assuming $T=300K$, we obtain

$$v_{in}^2 = \frac{kT}{C_i \cdot OSR} \rightarrow C_i = \frac{1.38 \cdot 10^{-23} \cdot 300}{(10\mu V)^2 \cdot 500} = 74 \text{ fF}$$

from which $C_2=0.88\text{pF}$ (in reality, we need some margin on these values!)
Noise – III

Assuming again the opamp noise negligible, the input-referred noise of the second integrator is

\[ \nu_n^2 = 2 \left( \frac{2 k_B T}{4C} + \frac{k_B T}{C} \right) = \frac{5 k_B T}{C} \]

This noise is shaped by the inverse of the transfer function of the first integrator \( H_1(z) = \frac{C_2}{C_1} \frac{1+z^{-1}}{1-z^{-1}} \)

Thus, the in-band noise becomes

\[ \nu_n^2 = \frac{\pi^2}{30 \text{OSR}^2} \left( \frac{C_2}{C_1} \right)^2 \nu_n^2 = 10^4 \nu_n^2 \]

Thus, totally negligible even if we choose a very small C capacitor, e.g. C = 20fF \( \rightarrow \) in this case, the minimum value for C is dictated by process limitation rather than noise considerations

Operational amplifier

Classical folded-cascode topology; with 2I_b current in the input pair, the slew current available at each output is \( I_b \), the largest quantity that needs to be transferred from \( C_1 \) to \( C_2 \) is \( C_1 V_{DD} \) \( \rightarrow \) allocating \( \frac{1}{4} \) of \( \frac{1}{2} \) clock period for slewing, we obtain

\[ I_b = \frac{C_1 V_{DD}}{0.25 \cdot 0.5 \cdot T_{clk}} = 8 f_{oa} C_1 V_{DD} = 8 \cdot 1 \text{M} \cdot 83 \cdot 3 = 2 \mu A \]

Thus, 8\( \mu A \) are enough for the whole amplifier

DC gain

We know that the NTF begins to degrade if its zeros are moved inside the unit circle by approx. \( \pi/\text{OSR} \); the finite gain of the opamp shifts the pole of the first integrator by \( C_1/(AC_2) \) \( \rightarrow \) it would seem that the following DC gain \( A \) would be adequate:

\[ A = \frac{\text{OSR} C_1}{C_2} = 13 \]

This, however, comes from a linear analysis, and neglects non-linear errors coming from slewing and non-linear DC gain (if \( A \) is linear, a finite \( A \) does not cause distortion; if there is no slewing, settling errors due to finite bandwidth do not cause distortion) Upper bound for required DC gain: the settled voltage at the opamp input (i.e. in series with the input signal) is \( V_{in}/A \) \( \rightarrow \) \( 1/2A \) in the double-sampling of the first integrator: this voltage has a signal component (which is ok), broadband noise, and distortion: if we assume that it consists of only distortion (which is highly pessimistic), the requirement that distortion stay below -100dBFS yields

\[ \frac{V_{in}}{2A} < 10^{-5} V_{in} \rightarrow \frac{1}{2A} < 10^{-5} \cdot 1.5 \rightarrow A > 90dB \]
In practice, A=60dB should be adequate, but this has to be checked with extensive simulations using the real non-linear model of the opamp.

Second integrator: we have seen that its noise requirements are much relaxed \( \rightarrow \) the same is true with respect to DC gain and slew-rate \( \rightarrow \) this amplifier can be implemented as a scaled-down version of the first amplifier, e.g. by a factor 4 (or higher, but the area and power savings decrease rapidly).

Important to find a good balance between all noise source \( \rightarrow \) the various noise sources are scaled in a way to give the most economical implementation \( \rightarrow \) e.g. if q-noise is allocated 90% of the noise budget, then the capacitor sizes that should satisfy the remaining 10% \( kT/C \) noise may be quite large \( \rightarrow \) excessive area and power consumption.

Furthermore, a large q-noise is not desirable \( \rightarrow \) q-noise is not really random \( \rightarrow \) may compromise performance in e.g. hi-fi audio systems, since the human ear can detect tones that are 20dB below the total noise level!

A reasonable noise budget is the one on the right.

When OSR is very large, the first integrator dominates the thermal noise budget – this is not true for low OSR (wide bandwidth) because the integrator gain is low at the high end of the signal band, which means that noise is not much attenuated.