Small tutorial on CIC filters*

J. Arzi
E-mail : contact AT tsdconseil.fr

December 1, 2015

Foreword

CIC filters are much used in applications where one needs to change the sampling frequency in a big ratio, for instance for decimation (reduction of the sampling rate: oversampled acquisition systems, ADC $\Sigma\Delta$) or interpolation (DAC).

The advantage of these filters is that they are particularly efficient in terms of complexity (no mulmultiplication, only integrators and comb filters are necessary). In return for this simplicity, they have some restrictions (for instance in terms of minimal register widths) and drawbacks (frequency droop in the passband, and aliasing), which we will discuss here.

Note : All the figures and simulations in this tutorial were generated with SCILAB, and with the CIC filter design mini-toolbox. You can get these free softwares here:

- **SCILAB** (numerical computing software, like MATLAB)**:** http://www.scilab.org
- **CIC filter design mini-toolbox** (for SCILAB :) http://www.tsdconseil.fr/log/scriptscilab/cic/index-en.html

*Original french version of this tutorial is available at the following address: http://www.tsdconseil.fr/log/scriptscilab/cic/cic.pdf

Contents

1 Definition of the CIC filter

A CIC filter is nothing but a a succession of N (filter order) moving averages of width RM (R is the decimation or interpolation factor, and M is a free integer parameter usually fixed to 1 or 2). Each moving average is a very simple FIR defined as:

$$y_n = \frac{1}{RM} \cdot \sum_{i=0}^{RM-1} x_{n-i}$$

that is, every output sample is computed as the mean of the RM last input samples. Hence, the global transfert function (that is, of the N successive stages of moving average) of a CIC filter is:

$$h(z) = \left(\frac{1 + z^{-1} + \cdots + z^{-(RM-1)}}{RM} \right)^N$$

In terms of frequential response, as the Fourier transform of a door (moving average) is a cardinal sinus (sinc function), the global response is:

$$|H(\nu)| = \left| \frac{\sin(RM\pi\nu)}{RM \sin \pi\nu} \right|^N$$

where $\nu = f/f_s$ is the frequency normalized against the sampling frequency ($\nu = 0.5$ is the Nyquist frequency).

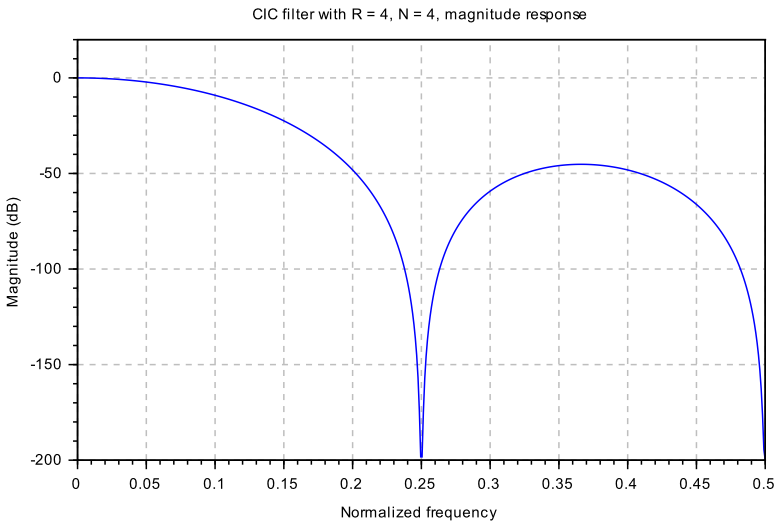


Figure 1: *CIC filter example, with* $R = 4$, $N = 4$ *et* $M = 1$

As we can see on the figure 1 above, the first zero is located at $\nu_0 = 2 \cdot \frac{1}{RM}$, that is at twice the output Nyquist frequency if $M = 1$ (for a decimator).

2 CIC filter for decimation

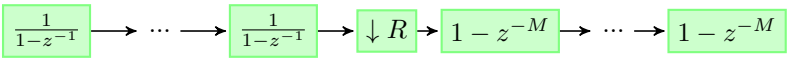
2.1 Principle

When we want to reduce the sampling frequency of a signal, we cannot just suppress some samples. Indeed, by doing that, the high frequency spectrum alias on the lower part of the spectrum, and so artefacts are introduced. Thus, before decimating (that is, suppressing unneeded samples), one has to apply a lowpass filter (here the CIC filter) so as to supress all high frequency components from the signal (that is, as much as possible, all frequencies above the output Nyquist frequency):

However, in this naive implementation, the CIC filter $h(z)$ must work at the highest sampling frequency, which can be very large (think about a $\Sigma\Delta$ ADC sampling at several MHz). In a more efficient implementation, we factorize the transfert function this way:

$$h(z) = \left(\frac{1 - z^{-RM}}{1 - z^{-1}}\right)^N = \left(\frac{1}{1 - z^{-1}}\right)^N \cdot \left(1 - z^{-RM}\right)^N$$

The first term is N stages of integrators, and the second term is N combs of width RM, which can thus be moved after the R-factor decimation step (and to become simple differentiators in the common case where $M = 1$):



Warning: This particular implementation of a sequence of moving averages works only with fixed point because overfows in integration stages must be accurately compensated by the stages of differentiation. In particular, it does not work in floating point.

The R parameter is set according to the desired decimation ratio, and the parameter N is free, and can be chosen according to the desired compromise between suppression of aliasing and attenuation in the passband (in practice, it is common to use in a final stage a FIR compensation filter for compensating the defects of the CIC filter, see section 3 hereafter).

2.2 Frequency response

Let us take a concrete example, where we wish to go from an input sampling frequency of 10 MHz to an output sampling frequency of 625 KHz, that is, a decimation factor of $R = 16$, and let us consider a 5-stages CIC filter ($N = 5$).

As we can see on the figure 2 on the next page, the filter attenuation is satisfying in the high frequencies ($>$ 60 dB from 500 KHz), but the signal in the passband (0 - 312 KHz) is sensibly distorted (see right figure), and the attenuation near the output Nyquist frequency is not very high (about 20 dB), which means that there will be some spectrum aliasing after the decimation (see bottom figure).

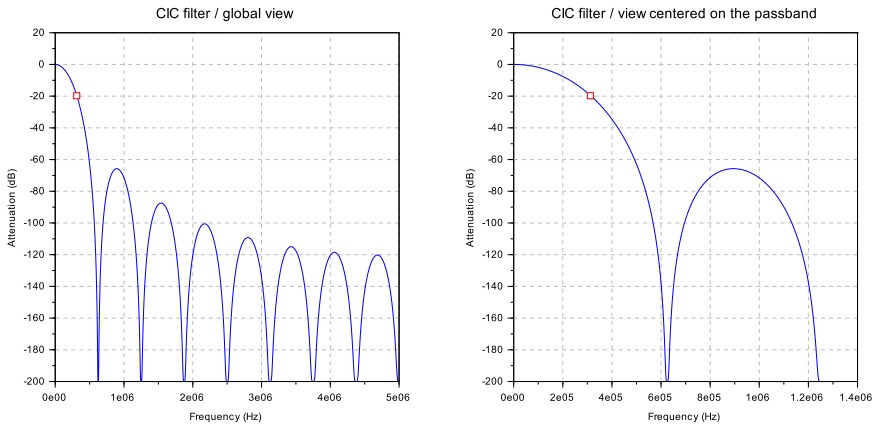


Figure 2: *Decimation with a CIC filter,* $R = 16$*,* $N = 5$ *and* $M = 1$ *- Frequency response before decimation. The red square indicate the output Nyquist frequency* $(= f_{Input\ Nyquist}/R)$*. This is at this point that spectrum aliasing begins after decimation (see next figure).*

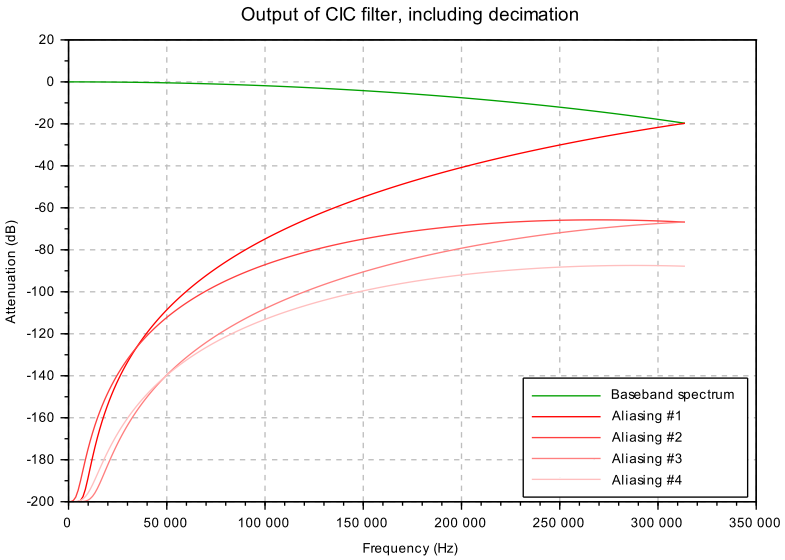


Figure 3: *Decimation with a CIC filter,* $R = 16$*,* $N = 5$ *and* $M = 1$ *- Frequency response after decimation (spectrum aliasing)*

2.3 Necessary computing resolution

Let k_1 be the resolution (in bits) of the input (signed) signal, and k_2 the width of the computing registers of the CIC filter (integer or fixed-point registers).

Typically k_2 must be higher as k_1 for the CIC filter to work correctly, and we will compute now how much resolution margin $m = k_2 - k_1$ is necessary, depending on the filter parameters.

For each stage of the filter (a pair of integrator and differenciator), the constraint that must hold is that the definite integral on the duration of RM successive samples should not overflow the maximal dynamic of the computing registers (otherwise the differenciator $1-z^{-RM}$ cannot compute the right value), that is :

$$RM \cdot 2^{k_1-1} < 2^{k_2} \Leftrightarrow RM < 2^{k_2-k_1+1}$$

(because the worst case appears when the signal is constant et is equal to -2^{k_1-1}, that is, the largest magnitude value that can be represented as a signed integer).

In the general case of N stages CIC filter, this condition generalizes as such:

$$(RM)^N < 2^{k_2-k_1+1}$$

Thus :

$$\boxed{m = k_2 - k_1 > N \log_2 RM - 1} \quad (1)$$

For instance, if $R = 16$, $N = 4$ and $M = 1$, and if all input samples are 16 bits ($k_1 = 16$), then the condition 1 gives $m > 15$, thus $k_2 = m + k_1 > 31$, that is, one has to use 32 bits registers.

For even higher decimation ratios, we can see that, at least for 16 bits input samples, 32 bits registers are not sufficient anymore, and an alternative could be (at least for CPU or DSP implementations) to use a cascade of several CIC of lower order (for instance, for $R = 64$, two successive CIC filters of identical ratio $R_1 = R_2 = 8$).

2.4 Complexity

For every input sample, one has to do N additions / accumulations (one for each integrator stage), and then, at a rate R times lower, N differencies, hence a global complexity of:

$$C = N\left(1 + \frac{1}{R}\right) \sim N \quad \text{cycles per input sample}$$

For instance, if $R = 16, N = 4$, et $M = 1$, then $C \sim 4$ cycles / sample.

3 Compensation filter

In return for their simplicity, CIC filters have two main drawbacks:

- They have not a flat response in the passband,
- Thet introduce some aliasing in the passband due to the fact that their attenuation near the Nyquist frequency is not high enough.

So, we add commonly, as a last stage, a FIR filter, called compensation filter, which role is to:

1. Fix the frequency droop in the passband,
2. Suppress as much as possible the aliasing introduced at the output of the CIC filter, and filter as much as possible above the final Nyquist frequency before the last stage of decimation

3.1 Compensation filter design

A commonly used design method is the *frequency sampling technique*[1], which enable to realize a filter with an arbitrary response in the frequency domain (function `fir2` under MATLAB). This method consists in using the Inverse Discrete Fourier Transform (IDFT) on the specified spectrum, so as to get the filter impulse (time domain) response.

In practice, so as to compensate for the CIC filter imperfections, the spectrum is chosen as:

$$\begin{aligned} |H(j\nu)| &= \left| \frac{1}{H_{CIC}(j\nu)} \right| && \text{if } \nu < \nu_c \\ H(j\nu) &= 0 && \text{if } \nu > \nu_c \end{aligned}$$

where ν_c is the desired cut-off frequency. That is, we compensate the CIC filter response in the passband, and we attenuate as much as possible the remaining signal above the passband[2].

Typically, the cut-off frequency ν_c is significatively lower than the Nyquist frequency at the ouput of the CIC filter, because otherwise, it would not be possible to suppress the aliasing due to the CIC filter. Optionnaly, a last stage of decimation can be done at the ouput of this compensation filter.

3.2 Compensation filter example

For instance, we have taken the following example of oversampled acquisition system:

- Input signal sampled at 6.4 KHz
- CIC decimation ratio: $R = 16$ (400 Hz at the output of the CIC filter)
- Compensation filter decimation ratio: $R_2 = 2$ (hence a final sampling frequency of 200 Hz)
- Cut-off frequency of the compensation filter: 80 Hz

The compensation filter was designed using the **mini toolbox for CIC filter design**, and the results have reproduced on the figures 4 and 5 on next page.

[1] Note that other more sophisticated methods exist, see for example the reference [1].

[2] Note: in practice, one can use a smoother transition between passband and stopband (at $\nu = \nu_c$), so as to obtain less oscillations in the filter response.

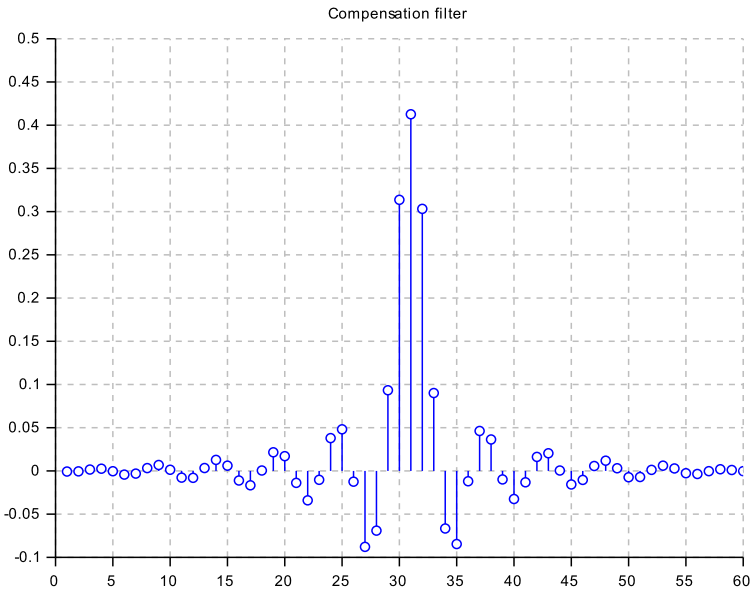


Figure 4: *Compensation filter example (impulse response of the FIR filter)*

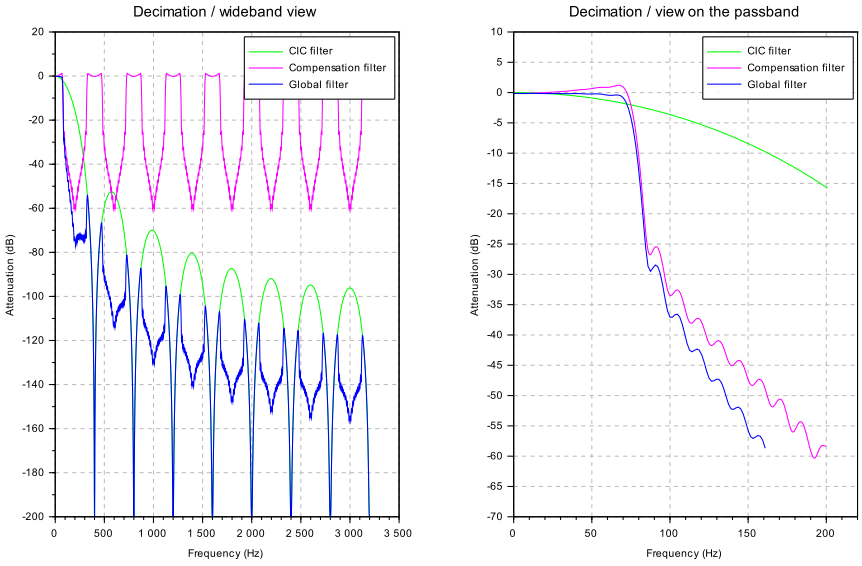
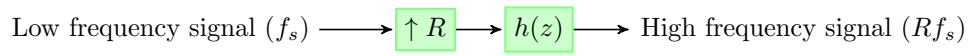


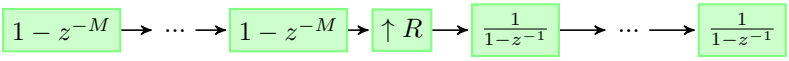
Figure 5: *Compensation filter example (global response)*

4 CIC filter for interpolation

For the interpolation case, the goal is to lowpass a signal which has been previously oversampled (by insertion of zeros), so as to suppress the aliasing created by the oversampling process:



In a way similar to the decimation case, for an efficient implementation, the CIC filter $h(z)$ is factorized as N stages of integrators and differenciators, but, this time, the laters are placed at first position in the chain, which enable to move the upsampling operation just before the integrators:



5 References

[1] G. DOLECEK, J. CARMONA, *On design of CIC decimators*, 2012

[2] A. LOLOI, *Exploring decimation filters*, 2013

[3] ALTERA, *AN 455 - Understanding CIC compensation filters*, 2007