**30. Use of Oversampling in a D/A Converter Design**

**(a) A simple Digital-to-Analog Converter Design**

Consider the following finite length digital signal, which we assume has time spacing $T_1$,

$y_n$ (n=-5..5) = { 1,2,3,3,4,1,0.2,-1,-2,-2,-1}

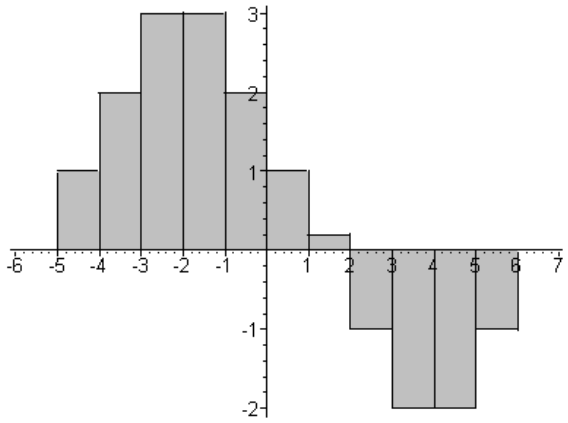



Fig 30.1

All samples other than those shown are 0. Using an arbitrary pulse shape $x_{pulse}(t)$, we can construct an amplitude-modulated pulse train w(t) whose spectrum is W(ω), as shown in summary box (25.4),

$$w(t) = \sum_{n=-\infty}^{\infty} y_n \, x_{pulse}(t - t_n) \tag{30.1}$$

$$W(\omega) = (1/T_1) X_{pulse}(\omega) \, Y'(\omega) , \tag{30.2}$$

where from summary box (23.5),

$$Y'(\omega) \equiv T_1 \sum_{n=-\infty}^{\infty} y_n \, e^{-i\omega n T_1} \quad . \tag{30.3}$$

If we were to select $x_{pulse}(t)$ to be a box of height 1 and width $T_1$, then we could regard the stair-step outline function shown in Fig 30.1 as a candidate *analog* signal y(t) whose samples are the $y_n$. In this special case, y(t) = w(t) so Y(ω) = W(ω). From (9.2) and (12.1) we have

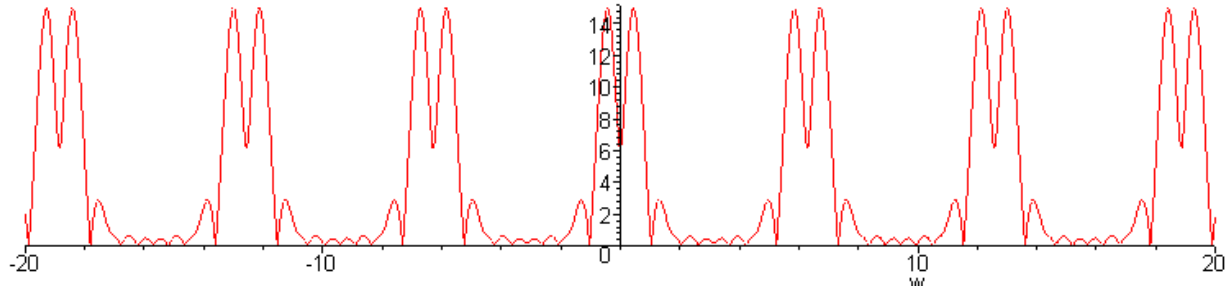
$$X_{box}(\omega, T_1) = e^{-i\omega T_1/2} T_1 \ \mathrm{sinc}(\omega T_1/2) \tag{30.4}$$

where the phase arises since the box $(0,T_1) = (0,1)$ is shifted $T_1/2$ to the right of the position of the symmetric box used in Section 9. Then using (30.2),

$$Y(\omega) = W(\omega) = e^{-i\omega T_1/2}\,\mathrm{sinc}(\omega T_1/2)\quad Y'(\omega) = e^{-i\omega T_1/2}\,\mathrm{sinc}(\omega T_1/2)\,T_1 \sum_{n=-\infty}^{\infty} y_n\, e^{-i\omega n T_1} \qquad (30.5)$$

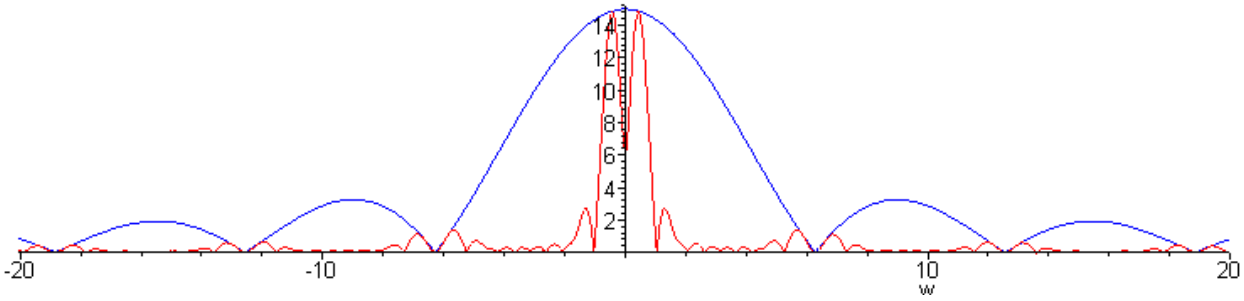
Y(ω) is the Fourier Integral Transform spectrum of the stair-step analog signal in Fig 30.1. Here are plots first of |Y'(ω)| from (30.3) using Fig 30.1 data, and second for |W(ω)| using (30.5).

```
y := array(-5..5,[1,2,3,3,2,1,.2,-1,-2,-2,-1]):
Yp := w -> sum(y[n]*exp(-I*n*w*T1),n=-5..5):
plot(abs(Yp(w)), w = -20..20, color = [red,blue],numpoints = 1200);
```




|Y'(ω)| Fig 30.2

```
sinc := x -> sin(x)/(x):
Y := w -> exp(-I*w*T1/2)* sinc(w*T1/2)*Yp(w):
plot([abs(Y(w)),abs(15*sinc(w*T1/2))], w = -20..20, color = [red,blue], numpoints =1200);
```




|Y(ω)| Fig 30.3

We see the expected image spectra in Y'(ω) in the first plot, but these spectra are quite suppressed in the second plot due to the taming effect of the sinc function in (30.5).

The above discussion describes the output of the following simple D/A converter design.

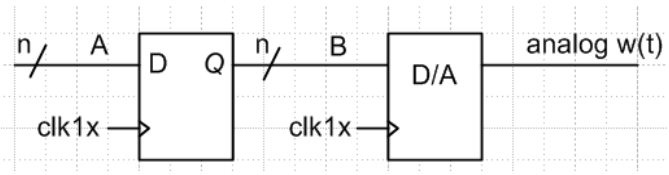



Fig 30.4

The purpose of the register on the left is to provide a stable signal on bus B to the D/A converter. We assume that the D/A converter is "glitch free" on its output, and just does what it should do.

**(b)What happens if the D/A converter is oversampled?**

We now trivially modify the above design by changing the D/A clock from clk1x to clk4x which runs 4X faster than clk1x.

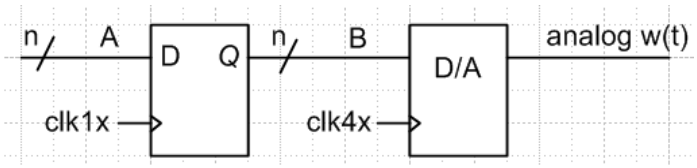



Fig 30.5

The D/A converter is now "4x oversampling" the data on the B bus. Besides making the D/A converter work harder, it seems clear that the output signal w(t) will be exactly the same as shown in Fig 30.1. Thus the plots of | Y'(ω) | and | W(ω) | shown above apply to this design as well as that of Fig 30.4.

It is useful, nevertheless, to think of the output of the oversampled design as follows :

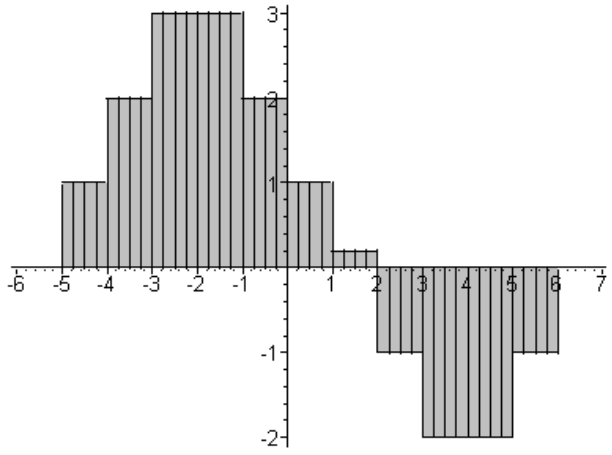



Fig 30.6

Now the output rectangles are 1/4 as wide because the D/A is clocking 4X faster. The analog outline is the same, but our analysis will be different and perhaps instructive. We shall now compute W(ω) in terms of the thin rectangles of Fig 30.6. Looking at the four $y_o = 1$ samples to the right of the vertical axis, those four boxes will make this contribution to the spectrum

$$W(\omega) = X_{box}(\omega,T_1/4)\,[\ldots + y_0 + y_0\, e^{-i\omega(T_1/4)} + y_0\, e^{-2i\omega(T_1/4)} + y_0\, e^{-3i\omega(T_1/4)} + \ldots.\,] \qquad (30.6)$$

Each thin box has a phase $e^{-i\omega(T_1/4)}$ relative to the box to its left due to (12.1). Since the pulse is 4x narrower than before, $X_{box}(\omega,T_1/4)$ is given by (30.4) with $T_1 \rightarrow T_1/4$.

We can write the square bracket in (30.6) as

$$[\,1 + e^{-i\omega(T_1/4)} + e^{-2i\omega(T_1/4)} + e^{-3i\omega(T_1/4)}\,]\,y_0 \equiv F(\omega)\,y_0 \qquad (30.7)$$

Of course every group of four terms will have this same common factor F(ω), so we can factor it out of the entire sum. The sum now looks like this:

$$W(\omega) = X_{\mathbf{box}}(\omega,T_1/4)\, F(\omega)\, \{ \ldots\ y_0 + y_1\, e^{-4i\omega(T_1/4)} + y_2\, e^{-8i\omega(T_1/4)} + \ldots \} \qquad (30.8)$$

But now the expression in curly brackets is exactly $Y'(\omega)/T_1$ of (30.3), our original Digital Fourier Transform spectrum of y(t). Thus we conclude that

$$W(\omega) = (1/T_1)\, X_{\mathbf{box}}(\omega,T_1/4)\, F(\omega)\, Y'(\omega)$$

$$= [\, e^{-i\omega T_1/8}\ (1/4)\ \mathrm{sinc}(\omega T_1/8)\,]\, F(\omega)\, Y'(\omega)\ . \qquad (30.9)$$

So this is W(ω) as computed in terms of the thin boxes of Fig 30.6. But we already argued that oversampling the D/A does not change the analog output signal w(t) or its spectrum W(ω), so somehow the expressions in (30.9) and (30.5) must be the same. This can only be true if

$$e^{-i\omega T_1/2}\, \mathrm{sinc}(\omega T_1/2) = [\, e^{-i\omega T_1/8}(1/4)\ \mathrm{sinc}(\omega T_1/8)\,]\, F(\omega) \quad ?$$

or, writing out the sinc functions,

$$e^{-i\omega T_1/2}\, \sin(\omega T_1/2)(2/\omega T_1) = [\, e^{-i\omega T_1/8}(1/4)\ \sin(\omega T_1/8)\ (8/\omega T_1)\,]\, F(\omega) \quad ?$$

or

$$e^{-i\omega T_1/2}\, \sin(\omega T_1/2) = [\, e^{-i\omega T_1/8}\ \sin(\omega T_1/8)\,]\, F(\omega) \quad ?$$

To verify this fact, we define $z \equiv e^{-i\omega T_1/4}$ ( variable for the Z transform). The above then reads

$$z^2\,(z^{-2}-z^2) = [z^{1/2}\ (z^{-1/2} - z^{1/2})\,]\,[\, 1 + z + z^2 + z^3\,] \quad ?$$

or

$$(1-z^4) = (1 - z)(\, 1 + z + z^2 + z^3) \quad ?$$

But this is a standard factorization so we have W(ω) is indeed the same either way we compute it. For some other oversampling factor like 6x or 8x, the verification is similar. We have just shown that

$$X_{\mathbf{box}}(\omega,T_1) = F(\omega)\, X_{\mathbf{box}}(\omega,T_1/4)$$

which we can think of as saying the product of two filters on the right gives the one on the left.

**(c) Add oversampling and zero-stuffing**

We now add a multiplexor to the D/A converter design which causes the first sample in each group to pass through, but "grounds" the last three samples in each group:

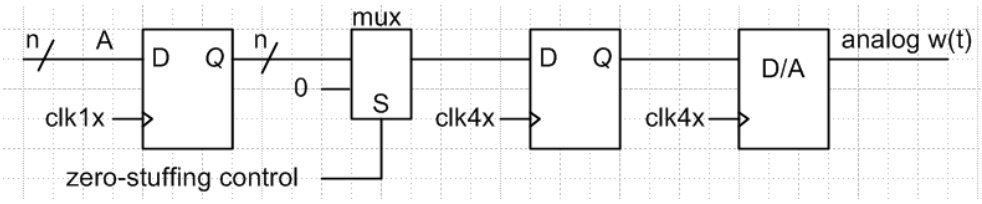



Fig 30.7

The output of this design is the following analog signal,

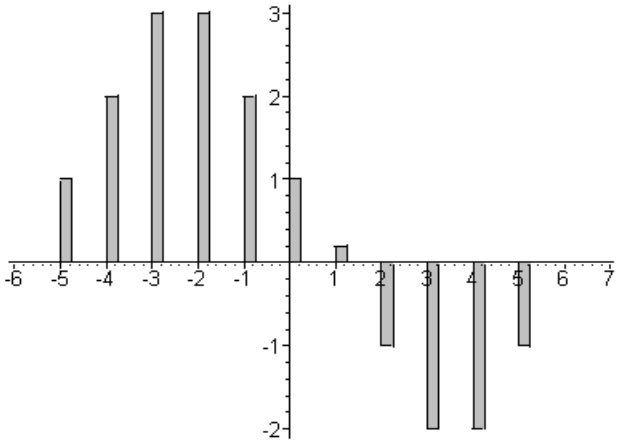



Fig 30.8

Due to the zero stuffing, we have in effect reduced the aperture of the signal from 100% to 25%. We saw in Section 26 how this broadens the sinc function envelope (narrower box $\Rightarrow$ broader sinc) which in turn reduces the sinc distortion of the main spectrum. That same effect appears below.

The spectrum of this output signal w(t) is given by (30.2) where we use the box spectrum of (30.4) for our thin box, but with $T_1 \to T_1/4$ ,

$$X_{box}(\omega, T_1/4) = e^{-i\omega T_1/8} T_1 \ \mathrm{sinc}(\omega T_1/8)$$

$$W(\omega) = (1/T_1)\{ e^{-i\omega T_1/8} T_1 \ \mathrm{sinc}(\omega T_1/8)\} Y'(\omega) = e^{-i\omega T_1/8} \mathrm{sinc}(\omega T_1/8)\ Y'(\omega)$$

$$= T_1 \ e^{-i\omega T_1/8} \mathrm{sinc}(\omega T_1/8) \sum_{n=-\infty}^{\infty} y_n e^{-i\omega n T_1}$$

The plot of $|Y'(\omega)|$ is already given in Fig 30.2. The new $W(\omega)$ plot is this:

```
W := w -> exp(-I*w*T1/8)*sinc(w*T1/8)*Yp(w):
plot([abs(W(w)),abs(15*sinc(w*T1/8))], w = -30..30, color = [red,blue],numpoints = 1200);
```

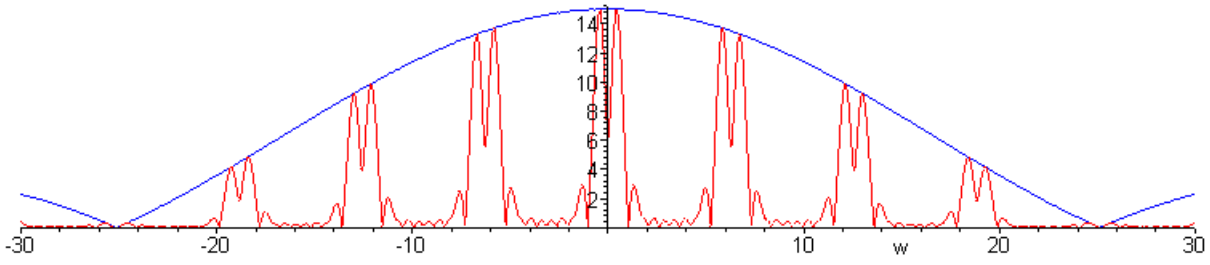



Fig 30.9

The good news is that the blue sinc distortion is smooth near the central main spectrum. The bad news is that there are lots of high-amplitude image spectra the must be dealt with.

**(d) Add a $\omega_1/2$ digital low-pass interpolation filter**

The new D/A design is this,

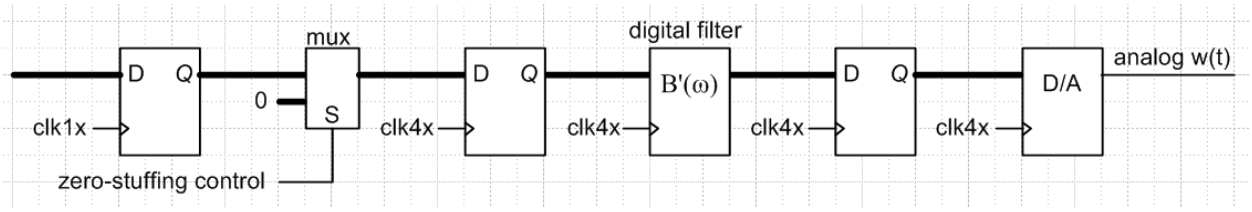



Fig 30.10

where B'(ω) is the transfer function of a low-pass filter which is clocked at the faster clk4x rate. We add low-cost registers at each stage in the pipeline to provide a stable input to the next stage.

In Section 29 we constructed an approximate brick-wall filter with this spectrum B'(ω),

```
wc := w1/2:  dt := T1/4:
for i from -10 to 10 do b[i] := evalf((Pi/wc)*sinc(wc*i*dt+.0000001)) od:
Bp := w -> (1/4)*sum(b[n]*exp(-I*n*w*T1/4),n=-10..10):
plot(abs(Bp(w)), w = -30..30);
```

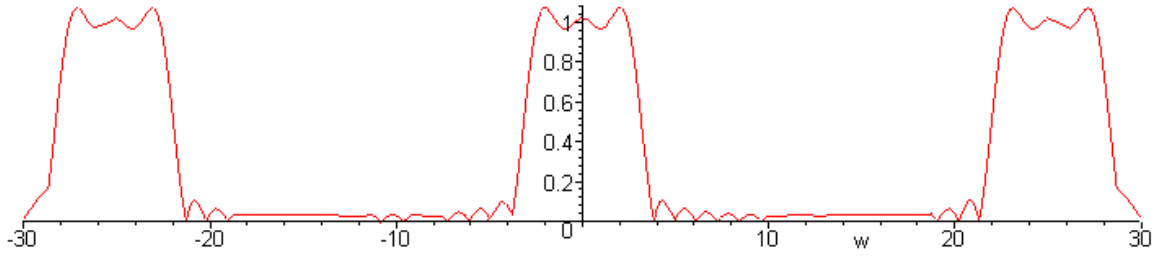



Fig 30.11

The output spectrum of the Fig 30.10 design which includes this filter is then

$$W_{new}(\omega) = B(\omega)\, W(\omega)$$

where W(ω) was plotted in Fig 30.9 :

```
plot([abs(Bp(w)*W(w)),abs(15*sinc(w*T1/8))], w = -30..30,color = [red,blue]);
```

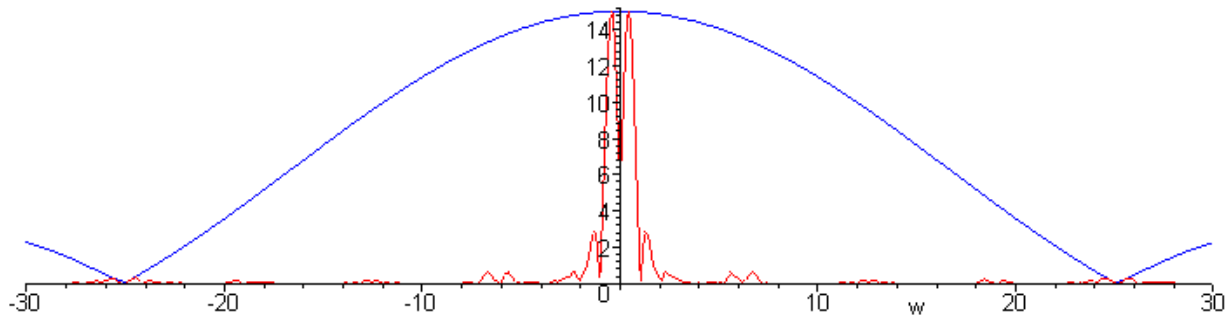



Fig 30.12

The effect of this digital filter is to remove the image spectra from Fig 30.9. Since this brick wall filter is not perfect, there is some small distortion of the central spectrum. On the other hand, the aperture reduction due to oversampling with zero-stuffing has broadened the sinc hump perhaps alleviating the need for a sin(x)/x post-filter (Section 26).

Since we never specified the original signal y(t) for which Fig 30.1 is the sampled version, it is difficult to compare the spectrum of that y(t) with the output of the design of Fig 30.2. Nevertheless, plotting the output of the Figure 30.2 design is quite interesting.

Here is Maple code which implements the time-domain convolution equation (29.5) of our brick wall filter with $\Delta t = 1/4$:

```
for i from -100 to 100 do b[i] := 0 od:    # prefill with all zeros
for i from -10 to 10 do b[i] := evalf((Pi/wc)*sinc(wc*i*dt+.0000001)) od:
for n from -40 to 40 do
  yout[n] := sum( yd[n-m]*b[m],m=-10..10):
od:
```

The resulting $y_{out}[n]$ sequence can be plotted using our ancient Maple V's primitive histogram routine which we have been using all along,

```
for i from -40 to 40 do bar[i] := Weight(i/4..(i+1)/4,yout[i]/4) od:
histogram([seq(bar[n],n=-30..30)],color=gray,xtickmarks=12);
```

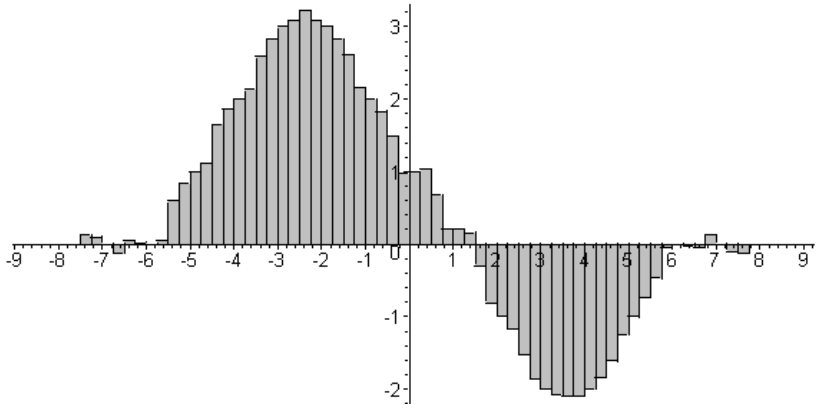



Fig 30.13

which we compare to our starting digital signal of Fig 30.1,

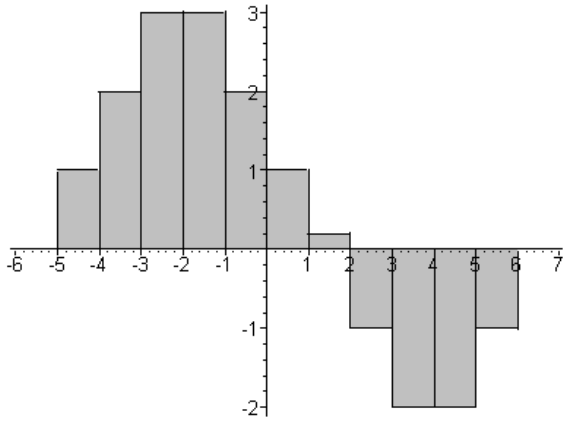



*Fig 30.1*

The output Fig 30.13 seems a little "ratty". If we increase the filter from 21 taps to 41 taps, things improve significantly, though there is still some ringing before and after the output pulse of interest,

```
for i from -100 to 100 do b[i] := 0 od:    # prefill with all zeros
for i from -20 to 20 do b[i] := evalf((Pi/wc)*sinc(wc*i*dt+.0000001)) od:
for n from -40 to 40 do
  yout[n] := sum( yd[n-m]*b[m],m=-20..20):
od:
```

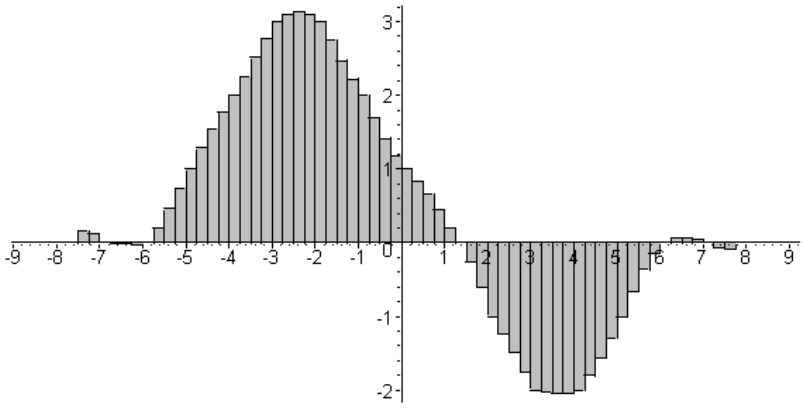



Fig 30.14

In the literature of oversampling, what we have been calling an oversampled digital low-pass filter is usually referred to as a digital interpolation filter, for obvious reasons comparing Fig 30.1 and Fig 30.14. In this application, since 3 out of every 4 incoming samples are zero from the zero-stuffing logic, it is possible to implement the filter more efficiently that we show in Fig 29.3 using polyphase techniques.