aliasing

April 1, 2015

1 Aliasing

Given a sampling rate Δt, we want to show that a harmonic function (here, a cosine) with a frequency higher than the the Nyquist frequency $\omega_{\mathrm{Ny}} = \frac{\pi}{\Delta t}$ cannot be distinguished by a lower frequency harmonic, sampled with the same time step.

1.1 Definitions

First, we import a Matlab-like set of commands,

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

To be concrete, we'll use $\Delta t = 0.4$ s and a fundamental period $T_n = 20$ s, hence a number of samples per period $N = 50$, or 2.5 samples per second.

```
In [2]: Tp   = 20.0
        N    = 50
        step = Tp/N
```

To the values above, we associate the fundamental frequency of the DFT and the corresponding Nyquist frequency.

```
In [3]: dw = 2*pi/Tp
        wny  = dw*N/2
        print "omega_1 =", dw
        print "Nyquist freq. =",wny,"rad/s =", wny/dw, '* omega_1'
```

```
omega_1 = 0.314159265359
Nyquist freq. = 7.85398163397 rad/s = 25.0 * omega_1
```

For comparison, we want to plot our functions also with a high sampling rate, so that we create the illusion of plotting a continuous function, so we say

```
In [4]: M = 1000
```

The function linspace generates a vector with a start and a stop value, with *that many* points in it (remember that the number of intervals is the number of points *minus* one),

```
In [5]: t_n=linspace(0.0,Tp,N+1)
        t_m=linspace(0.0,Tp,M+1)
```

The Nyquist circular frequency is $25\Delta\omega$.
The functions that we want to sample and plot are

$$\cos(h\Delta\omega t) \quad \text{and} \quad \cos((h-N)\Delta\omega t),$$

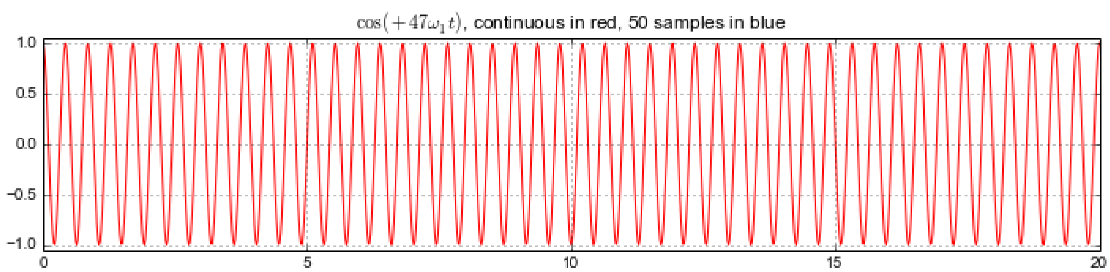
in this example it is $h = 47$ but it works with different values of h as well...
In the following, `hs` and `ls` mean high and low sampling frequency, while `hf` and `lf` mean high and low cosine frequency. Note that `t_m` and `t_n` are vectors, and also `c_hs_hf` etc are vectors too.

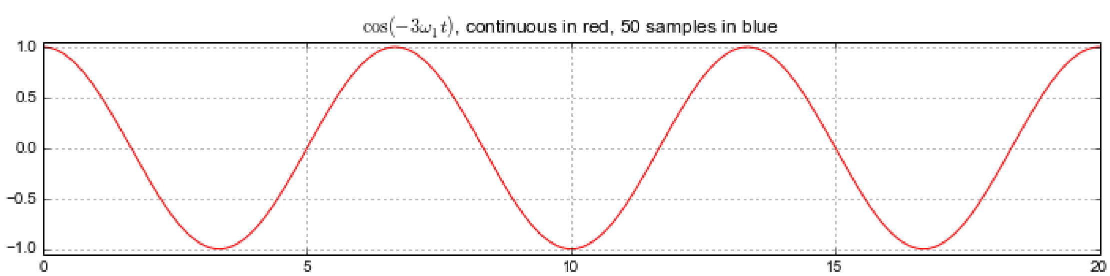
In [6]:
```
hf = 47
lf = hf - N

c_hs_hf = cos(hf*dw*t_m)
c_hs_lf = cos(lf*dw*t_m)

c_ls_hf = cos(hf*dw*t_n)
c_ls_lf = cos(lf*dw*t_n)
```

First, we plot the harmonics with a high frequency sampling (visually continuous, that is).

In [7]:
```
figsize(12,2.4)
figure(1);plot(t_m,c_hs_hf,'-r')
ylim((-1.05,+1.05))
grid()
title(r'$\cos(%+3d\omega_1t)$, continuous in red, 50 samples in blue'%(hf,))
figure(2);plot(t_m,c_hs_lf,'-r')
ylim((-1.05,+1.05))
grid()
title(r'$\cos(%+3d\omega_1 t)$, continuous in red, 50 samples in blue'%(lf,))
```

Out[7]:
```
<matplotlib.text.Text at 0x7f4b118dcdd0>
```

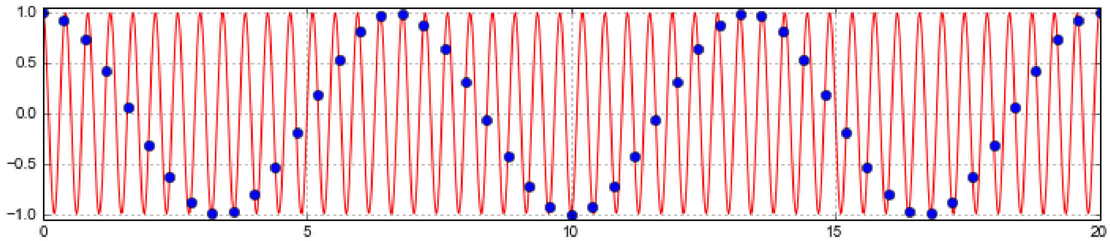


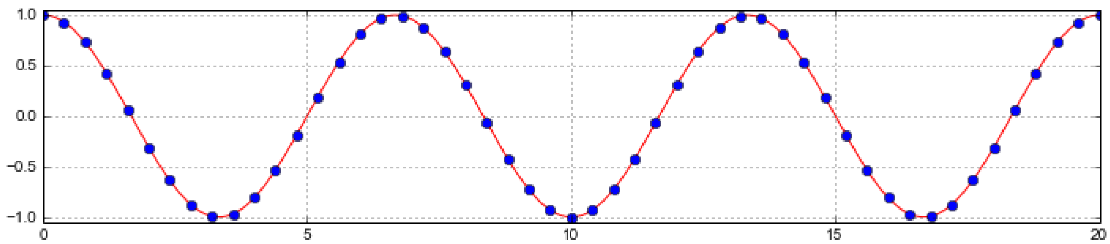


Not surprisingly, the two plots are really different.

In the next plots, we are going to plot the *continuous* functions in red, and to place a blue dot in every (t,f) point that was chosen for a low sampling rate.

In [8]:
```
figure(1) ; plot(t_m,c_hs_hf,'-r',t_n,c_ls_hf,'ob')
ylim((-1.05,+1.05));grid();
figure(2) ; plot(t_m,c_hs_lf,'-r',t_n,c_ls_lf,'ob')
ylim((-1.05,+1.05));grid();
```



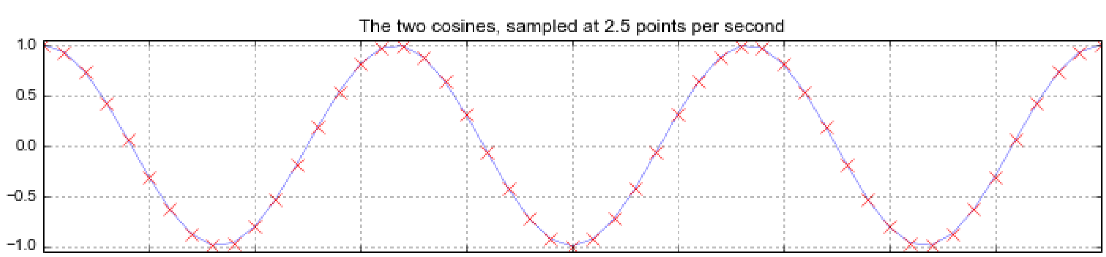


If you look at the patterns of the dots they seem, at least, very similar. *What happens is aliasing!*

It's time to plot *only* the functions samplead at a low rate:

- the high frequency cosine, sampled at 2.5 samples per second, blue line,
- the low frequency cosine, sampled at 2.5 samples per second, red crosses only.

In [9]:
```
figure(3) ; grid()
title('The two cosines, sampled at 2.5 points per second')
figure(3)
plot(t_n,c_ls_hf,'-b', linewidth=.33)
plot(t_n,c_ls_lf,'xr', markersize=8)
xticks((2,4,6,8,10,12,14,16,18,20))
ylim((-1.05,+1.05));
```



Let's try zooming into a detail, using blue crosses for the hf cosine and red crosses for the lf cosine:

```
In [10]: y = c_ls_lf[N/2-1]
         n0 = int(y*100)
         n1 = int(n0/5)*5
         n2 = n1 + 5
         print n1/100., y, n2/100.,
         axis([9.5, 10.5, n1/100., n2/100.,]); grid()
         plot(t_n,c_ls_hf,'+b',markersize=20)
         plot(t_n,c_ls_lf,'xr',markersize=20);
```

```
-0.95 -0.929776485888 -0.9
```