ELECTRONOTES
1016 Hanshaw Rd
Ithaca, NY 14850

APPLICATION NOTE NO. 373

February 2009

INTERPRETATION OF DFT AS ONE FREQUENCY SINUSOIDAL WAVEFORMS

1 INTRODUCTION

A length-N time sequence x(n) has a length-N DFT (FFT) X(k) and nearly everyone knows that we are to interpret the DFT as a frequency-domain version of x(n). Exactly how? If x(n) is a sinusoidal waveform and if there are exactly k_0 full cycles in x(n), then X(k) is non-zero only for $k=k_0$ and for $k=N-k_0$ (Fig. 1). Otherwise, in the general case, all X(k) are non-zero – the phenomenon called "leakage" has occurred (Fig. 3).

{Note that this is k_0 cycles in N samples, generally from n=0 to n=N-1. A sample taken at n=N in this case would be the first sample of the k_0+1 cycle. Nor is it necessary that any or all of the k_0 cycles have the same actual samples as any other cycle. It's only the total number of cycles in the N samples – not the exact timing of the samples within any one cycle (Fig. 2).

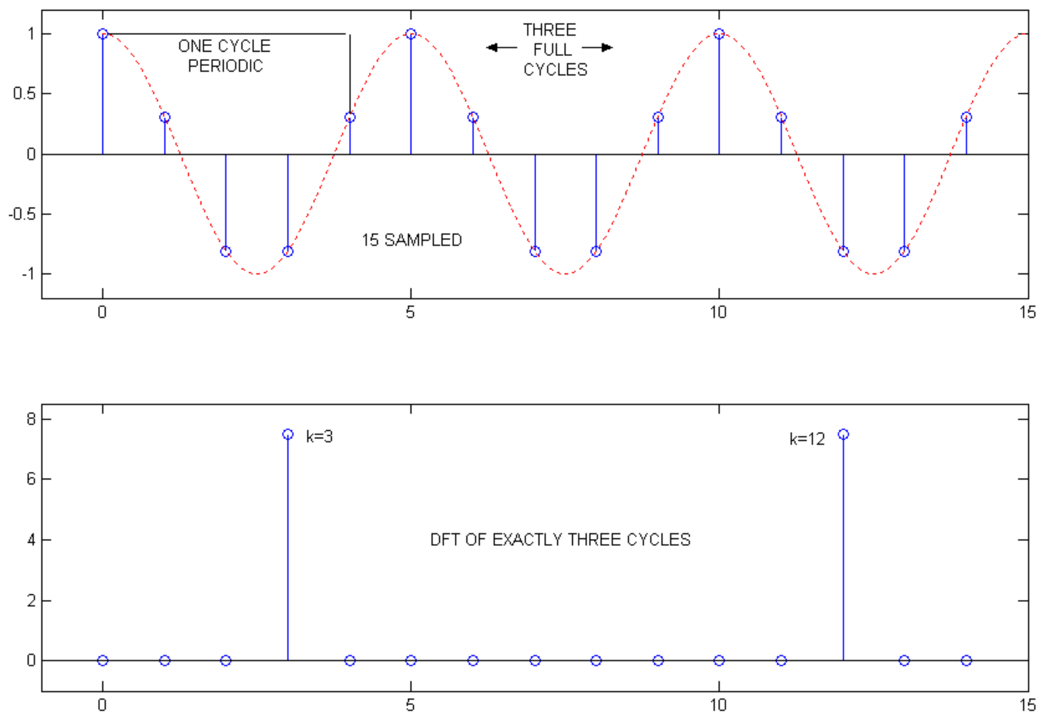


Fig. 1 Exactly three cycles – perfectly resolved

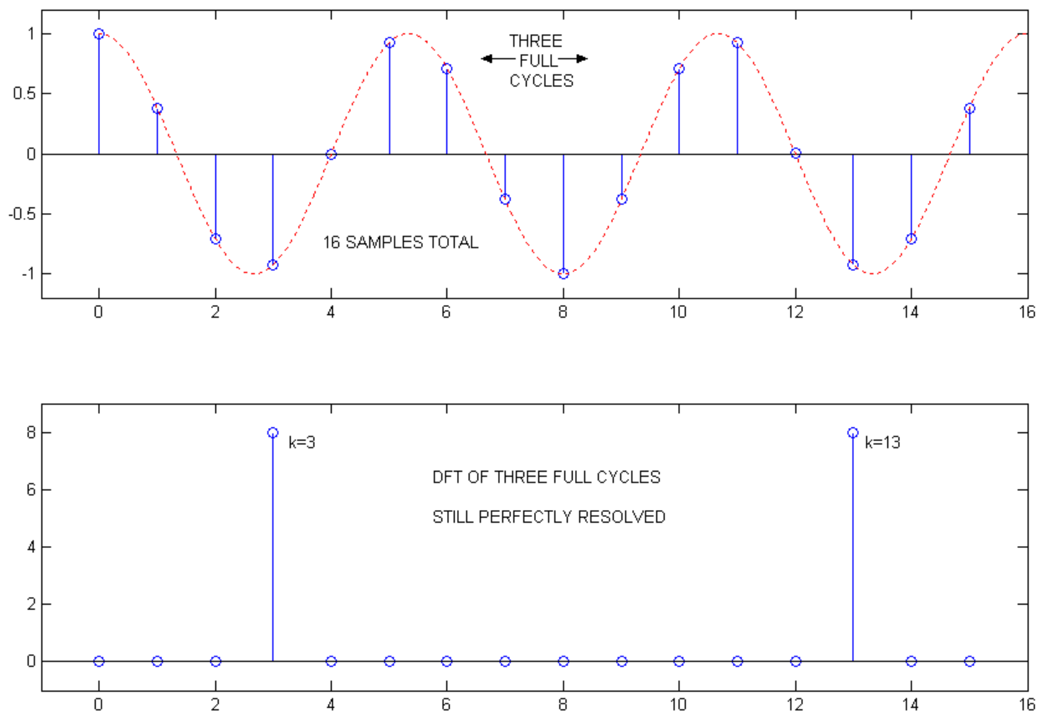


Fig. 2 The samples within the cycles need not be periodic within all the cycles – still have perfect resolution

2 SELF-WINDOWING

The key to understanding the "leakage" or lack of leakage is to recognize that the DFT is self-windowing, since it is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j(2\pi/N)nk} \qquad (1)$$

In consequence, we heed to consider the rectangular window:

$$r(n) = 1 \quad n=0,1,2,\ldots.N-1 \qquad (0 \text{ else}) \qquad (2)$$

which has a DTFT

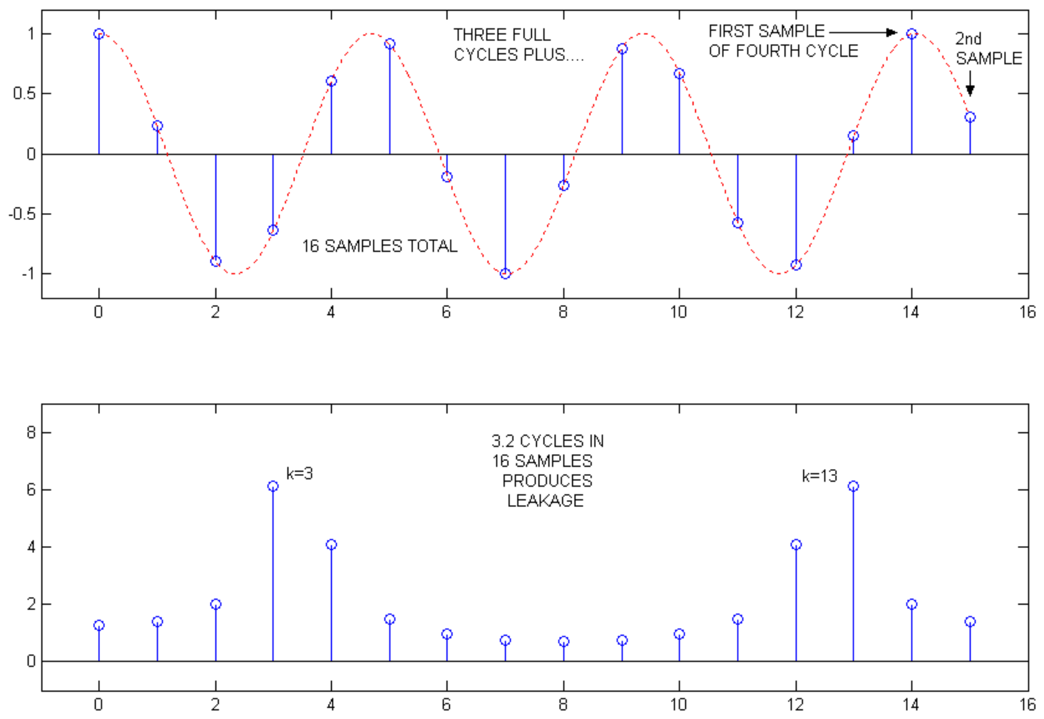


Fig. 3   Having an non-integer number of cycles results in leakage

$$R(e^{j\omega}) = \sum_{n=-\infty}^{\infty} r(n)e^{-jn\omega}$$

$$= \sum_{n=0}^{N-1} e^{-jn\omega}$$

$$= e^{-j[(N-1)/2]\omega} \; [\, \sin(N\omega/2) \,/\, \sin(\omega/2) \,] \qquad (2)$$

which is a “periodic sync” or “Dirichlet kernel,” periodic in 2π, with zeros at $\omega_m=(2\pi/N)m$ except at integer N, where it is of magnitude N (Fig. 4). Note that the spacing of the zeros is the same as the spacing of the DFT harmonics: $\omega_k = (2\pi/N)k$.

It is convenient here to use a frequency measured as k_c (continuous k) which is the same as the DFT index k, except k is limited to integers. It is normal to talk about DFT's in terms of frequencies k, where k is an integer. But something like k=3.2 does not make sense. Here k_c=3.2 is perfectly sensible. Note that:

$$k_c = (N/2\pi)\omega \tag{3}$$

so we can write:

$$R(k_c) = e^{-j\,[\,(N-1)/N\,]\,\pi k_c}\ [\,\sin(\pi k_c) / \sin(\pi k_c/N)\,] \tag{4}$$

Fig. 4 shows the magnitude of $R(k_c)$ calculated for k_c=0 to N with N=20 for this example. This is the familiar periodic sinc. Fig. 5 shows the real and imaginary parts of this same $R(k_c)$. Note that the "peaks" of this response are at 0 and at multiples of N.

In an actual spectral analysis application with the DFT, the self-windowing of the DFT is equivalent to multiplying the data by a rectangular window, and accordingly the Discrete-Time Fourier Transform (DTFT) of the full-length signal is convolved with the periodic sinc. Initially we can assume cosine phase for the full-length signal, so it is represented by two real non-zero lines, one at k_o, and the other at $N-k_o=-k_o$, where k_o

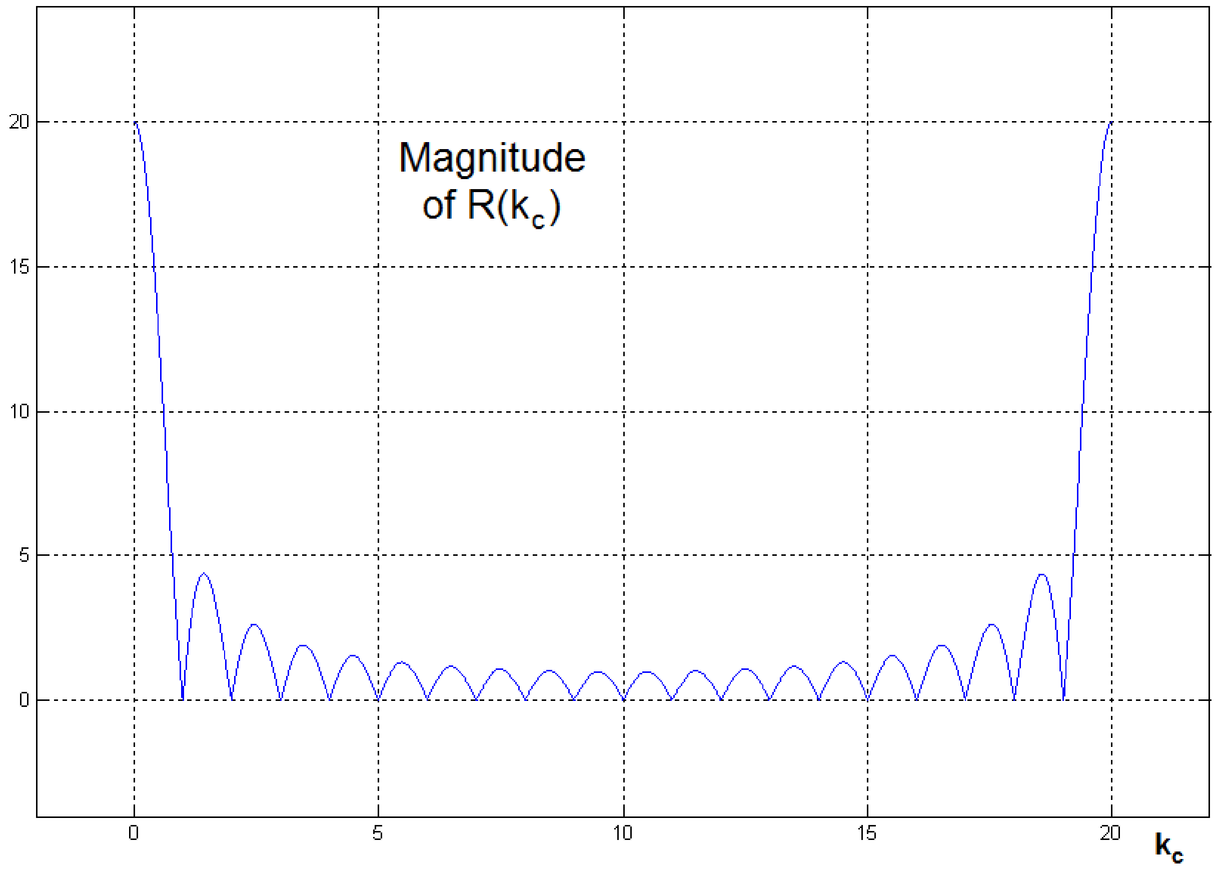


Fig. 4 The magnitude of $R(k_c)$. See also Fig. 5 for the real and imaginary components corresponding to this magnitude plot. (length 20)

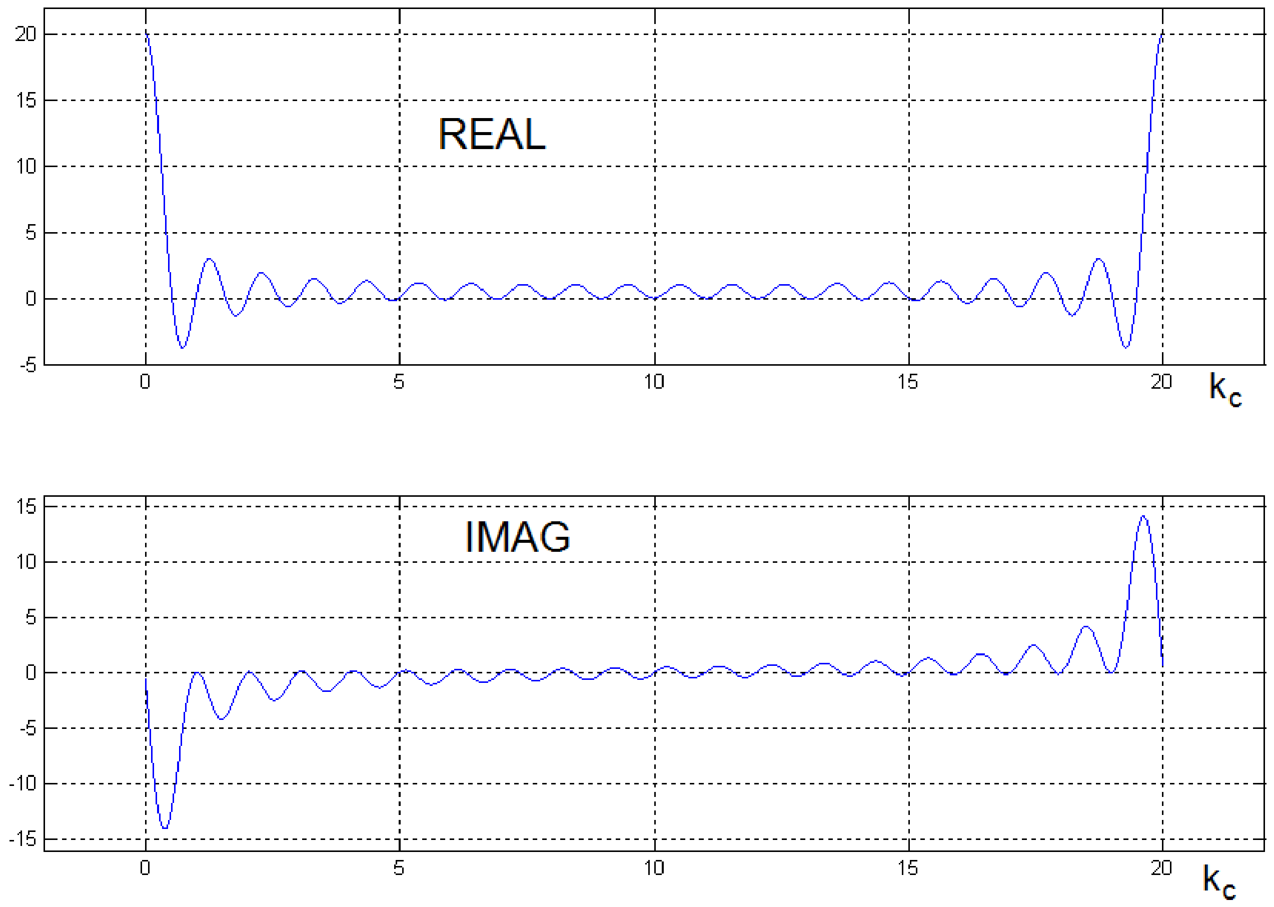


Fig. 5 The real and imaginary parts of $R(k_c)$ corresponding to the magnitude plot of Fig.4

is the value of k_c that corresponds to the frequency of the cosine. That is, k_o is the "answer" we are looking for. The general notion of what we are doing can be seen in Fig. 6 where we have taken Fig. 4, offset it by $+k_o$ and by $–k_o$ (k_o=3.3 for the example) and plotted the two magnitudes. This only illustrates the idea – we can not add the magnitudes to get the magnitude of the sum.

Fig. 7 shows the correct way to achieve the convolution result – the sum of the two displaced periodic sinc functions. That is, we take the complex form of $R(k_c)$, displace and add these, and then take the magnitude. Fig. 8 shows the corresponding real and imaginary parts.

In addition to the plot of the convolution sum in Fig. 7, two additional results are shown. The open circles are the values of the sum at integer values of k_c. That is, using the traditional notion of k as the DFT index. These are of course on the line. Overplotted at stars are the magnitude values of the DFT of the sequence, obtained independently of the calculation of $R(k_c)$ as we would get using equation (1). These are, exactly in the center of the circles as we expect and hope.

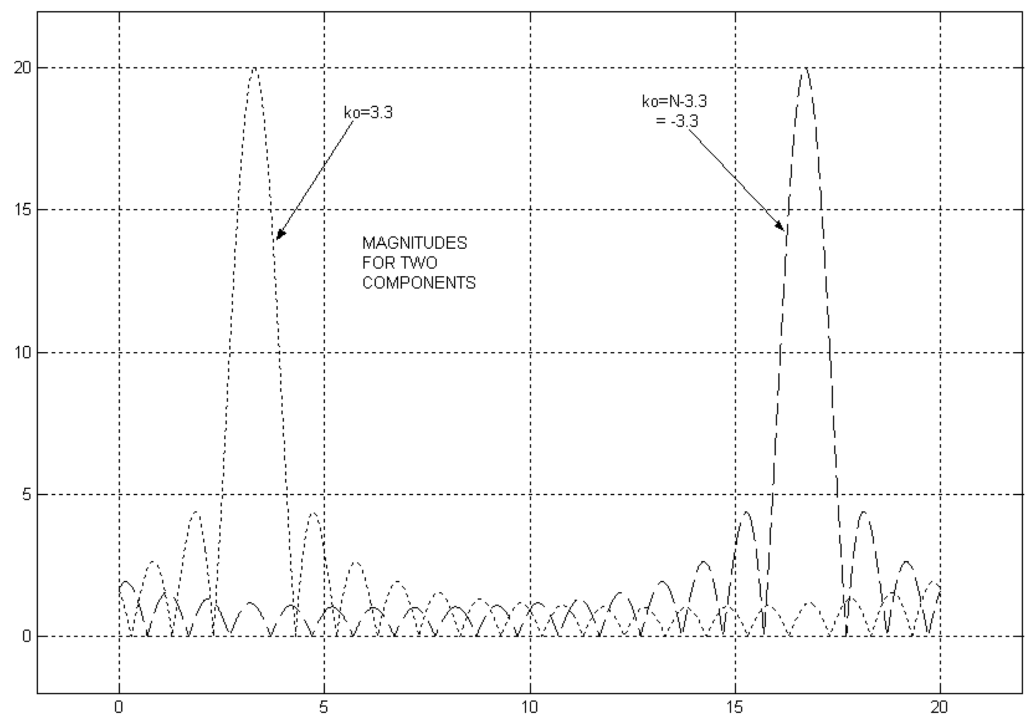


Fig. 6 The periodic sinc functions are offset (rotated actually) by the convolution process. We can not add these magnitude functions. (see Fig. 7 below)

Much as we used Fig. 4, involving just the magnitude, to suggest a procedure, Fig. 7 is suggestive of success, but not a fully-convincing demonstration. Fig. 8 shows the real and imaginary parts of Fig. 7, with the real and imaginary parts of the directly-calculated DFT, X(k), overplotted as stars. We see an exact match.

The graphs presented indicate that we understand the DFT "leakage" problem in terms of convolution of the DTFT with the DTFT of a rectangular window, followed by sampling. We want to follow up these graphical with equations and Matlab code to support the results and to suggest further investigations.

3 THE EQUATIONS

The summation leading to Fig. 7 and Fig. 8 is:

$$D(k_c) = R(k_c-k_o) + R(k_c+k_o) \tag{5}$$

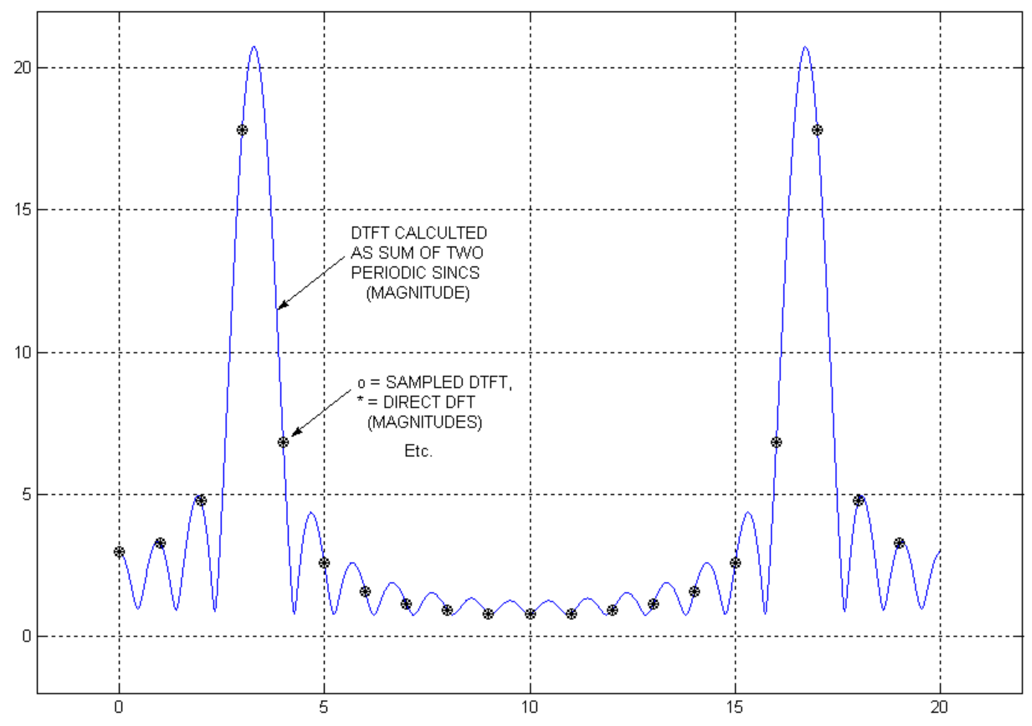


Fig. 7 Summation of periodic sincs at $k_c=k_o=3.3$ and $k_c=20-k_o=16.7$. Samples at the integer points are shown as open circles, but these open circles are then "filled" by the stars, which represent the direct calculation of the DFT. Here we are plotting magnitudes.

Plugging these offsets into equation (4) leads to figures (6) and (7). To be general, we need to assume an arbitrary phase, relative to cosine, so the summation becomes:

$$D(k_c) = e^{j\varphi}R(k_c-k_o) + e^{-j\varphi}R(k_c+k_o)$$

$$= e^{-j[(N-1)/N]\pi(k_c-k_o)}e^{j\varphi}\sin[\pi(k_c-k_o)]/\sin[\pi(k_c-k_o)/N]$$

$$+ e^{-j[(N-1)/N]\pi(k_c+k_o)}e^{-j\varphi}\sin[\pi(k_c+k_o)]/\sin[\pi(k_c+k_o)/N] \qquad (6)$$

And now,

$$X(k) = D(k_c=k) \qquad (7)$$

offers a complicated equation for the DFT.

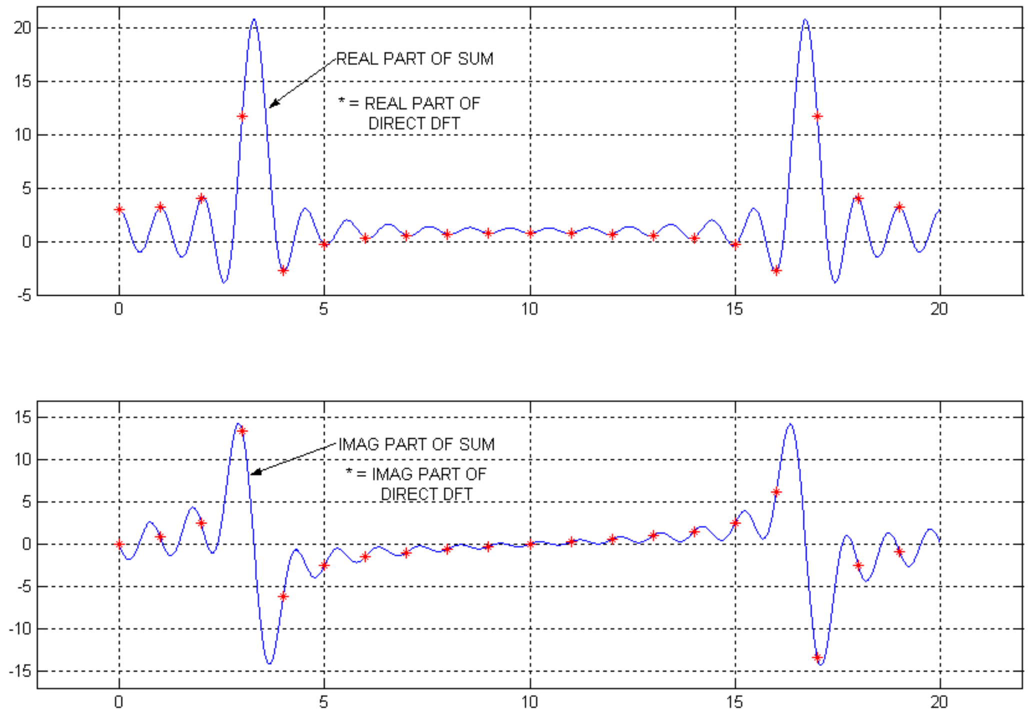


Fig. 8 To show that the direct DFT matches the sampled sum of the periodic syncs, we plot (with stars) the real and imaginary parts separately (not just the magnitude as in Fig. 7

In addition to verifying that the convolution followed by sampling procedure calculates the usual DFT, we have the ability to calculate the DFT of shifted versions of the original sequence by modifying the phase: $\varphi \rightarrow \varphi + (2\pi/N)k_o m$, where m is the number of samples to shift.

4 SEEING THROUGH THE LEAKAGE

In general, when all the DFT “bins” are non-zero, we can’t say much about the spectrum. However, if we know it corresponds to a single sinusoidal waveform, we can often guess more about it. For example, looking back at Fig. 3, we notice that the largest bins are between k=3 and k=4, suggesting that the actual frequency is between these two values, which is true. Further, the bin for k=3 is larger than the bin for k=4, suggesting that the actual frequency is closer to k=3 than it is to k=4, which is also true. In general, if we were to work teratively, we might have good luck in finding a good match to a given leakage pattern, and this might suffice if we had only a few cases to identify.

In the case of a single sinusoidal, two things are likely to be true. First, we do not know the ideal length of the sequence (for perfect resolution) to be analyzed, and second, we don't know the starting point (the phase). Thus we might have a sequence x known to represent a sinusoidal waveform. We might choose a length N, and take samples for N=0 to N-1. Or, we might choose some other length N sequence, for example, from N=1 to N. That is, the two sequences are shifted by 1. We know that the two sequences would have a phase difference of $2\pi k_o/N$, although we do not know k_o. It is k_o we are looking for. The two sequences will have DFT's given by:

$$D_0(k_c) = e^{-j[(N-1)/N]\pi(k_c-k_o)} e^{j\varphi} \sin[\pi(k_c-k_o)] / \sin[\pi(k_c-k_o)/N]$$

$$+ e^{-j[(N-1)/N]\pi(k_c+k_o)} e^{-j\varphi} \sin[\pi(k_c+k_o)] / \sin[\pi(k_c+k_o)/N] \quad (8a)$$

$$D_1(k_c) = e^{-j[(N-1)/N]\pi(k_c-k_o)} e^{j\varphi} e^{j2\pi k_o/N} \sin[\pi(k_c-k_o)] / \sin[\pi(k_c-k_o)/N]$$

$$+ e^{-j[(N-1)/N]\pi(k_c+k_o)} e^{-j\varphi} e^{-j2\pi k_o/N} \sin[\pi(k_c+k_o)] / \sin[\pi(k_c+k_o)/N] \quad (8b)$$

where D_0 corresponds to the original sequence, and D_1 represents the shifted sequence.

This mess can be simplified by evaluating these two at k_c=0.

$$D_0(0) = e^{-j[(N-1)/N]\pi(-k_o)} e^{j\varphi} \sin[\pi(-k_o)] / \sin[\pi(-k_o)/N]$$

$$+ e^{-j[(N-1)/N]\pi(+k_o)} e^{-j\varphi} \sin[\pi(+k_o)] / \sin[\pi(+k_o)/N] \quad (9a)$$

$$D_1(0) = e^{-j[(N-1)/N]\pi(-k_o)} e^{j\varphi} e^{j2\pi k_o/N} \sin[\pi(-k_o)] / \sin[\pi(-k_o)/N]$$

$$+ e^{-j[(N-1)/N]\pi(+k_o)} e^{-j\varphi} e^{-j2\pi k_o/N} \sin[\pi(+k_o)] / \sin[\pi(+k_o)/N] \quad (9b)$$

The periodic sinc here is an even function:

$$\sin[\pi(-k_o)] / \sin[\pi(-k_o)/N] = \sin[\pi(+k_o)] / \sin[\pi(+k_o)/N] \quad (10)$$

and the exponentials can be combined to cosines using the Euler relationships:

$$D_0(0) = 2 \sin[\pi(-k_o)] / \sin[\pi(-k_o)/N] \cos[(N-1)\pi k_o/N + \varphi] \quad (11a)$$

$$D_1(0) = 2 \sin[\pi(-k_o)] / \sin[\pi(-k_o)/N] \cos[(N-1)\pi k_o/N + \varphi + 2\pi k_o/N] \quad (11b)$$

which has ratio

$$D_0(0)/ D_1(0) = \cos[(N-1)\pi k_o/N +\varphi] / \cos[(N-1)\pi k_o/N +\varphi + 2\pi k_o/N] \qquad (12a)$$

which can be simplified to:

$$D_0(0)/ D_1(0) = \cos(\pi k_o +\varphi - \pi k_o/N) / \cos(\pi k_o +\varphi + \pi k_o/N) \qquad (12b)$$

Here $D_0(0)$ and $D_1(0)$ are easily obtained as the k=0 values of the DFTs $X_0(0)$ and $X_1(0)$, or of the DC values of the sequences, so that is known. The values of k_o and of φ are unknown. One way to find a solution is to search a range of k_o and of φ until a good match to the original ratio is found. To get some idea of the correctness of the development and the nature of the result, consider that N can be any value, including N=1! This gives:

$$D_0(0)/ D_1(0) = \cos(\varphi) / \cos(2\pi k_o +\varphi) \qquad (13)$$

In the case of N=1, the DC values of the DFT's are just the length-1 sequences corresponding to the first samples. Thus if the sequence involves x(0) and x(1), the equation to be solved is:

$$x(0) / x(1) = \cos(\varphi) / \cos(\varphi + 2\pi k_o) \qquad (14)$$

which is obviously true.

So, the discussion comes down to determining the accuracy, the efficiency, and possible uniqueness of solutions to equation (14) or more generally, of equation (12b) – an equation in two variables that is also non-linear!

5 SEARCH PROGRAMS

Consider as test input a sequence x(n) that is a cosine of frequency k_o and phase φ but which has a length greater than the values of N we intend to use, so there is no problem in choosing two length N sequences offset by 1 sample. We easily compute the values of the DFTs for k=0, and thus know the ratio $D_0(0)/ D_1(0) = X_0(0)/X_1(0)$, where $x_0(n)$ is the first length N subsequence from x(n) and $x_1(n)$ is the offset subsequence. The task here, considering equation (12b) is to find k_o and φ. We search likely ranges of k_c and φ for a match to the ratio from the DC values of the DFTs.

Program 1 is a test Matlab program. We assume that x(n) was pre-computed (perhaps to length 1000) and we can thus examine a length N subsequence (and its offset). The basic search is comprised of the nested for loops near the bottom. Note that this nest has an increment of 0.01 for both scan variables (ph and kc) and a test error value of 0.0000001, both of which can be adjusted.

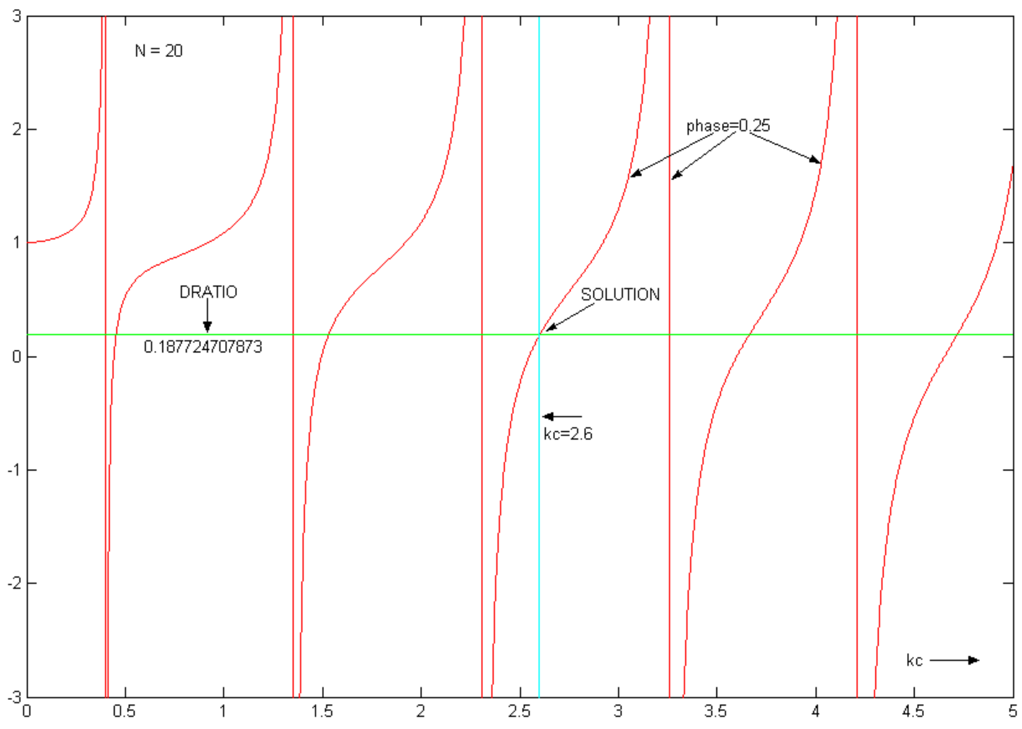
PROGRAM 1

```
% findk0.m

function [k0,freq,phase]=findk0(x,N)

x0=x(1:N);
x1=x(2:N+1);

X0=fft(x0);
X1=fft(x1);

X0half=X0(1:round(N/2));

[maxX,kmax]=max(abs(X0half));

shortout=0;
% First check for perfect resolution
% if perfect resolution, DRATIO will be of form 0/0 (very small/very small)
%       and search will not work
if abs(sum(X0half)-X0(kmax))<0.0001
    k0=kmax-1;
    freq=k0/N;
    phase=angle(X0(kmax));
    shortout=1;
end

if shortout==0;
% Now check likely direction of k0
sh=-1/2;
if abs(X0(kmax+1))>abs(X0(kmax-1))
    sh=1/2;
end

kmax=kmax-1
format long
DRATIO=X0(1)/X1(1)
format short

for ph=0:.01:6.28
   for kc= (kmax+sh-0.5):0.001:(kmax+sh+0.5)
       f=cos(pi*kc+ph-pi*kc/N)./cos(pi*kc+ph + pi*kc/N);

      if abs(f-DRATIO)<0.0000001
          k0=kc;
          freq=k0/N;
          phase=ph;
      end
   end
end


end
shortout
```

There are two things that are usefully done <u>before</u> the main search. The first is to assure that we do not already have perfect resolution. The reason this would be a problem is that with perfect resolution, the ratio $X_0(0)/X_1(0$ is of the form 0/0 which means in practical computations that it is the ratio of two very small numbers, which is unpredictable and meaningless for our purposes. Accordingly we look first for perfect resolution., and if found, use that for the direct answer.

Secondly, since the equations do not show the need for a particular N, it might be supposed that N should be chosen as small as possible (perhaps even just N=1). However, by using a larger N, we can get much more precision with the same search effort over k_c. It is easy to find the maximum magnitude of the DFT and the corresponding k. Then by looking at the magnitude one bin below and one bin above, we can identify a unit range of k_c to search. In Fig. 3, we would only search k_c=3 to k_c=4, for example.

While we should probably anticipate additional pitfalls (after all, we have two variables!), for the most part, the program works! If the search resolution is increased, by using increments of say 0.001, 0.0002, or even 0.0001, the run time of course increases, and there is a danger of hitting a few nearby answers and/or “spurious” answers. A further reduction of the error we allow for the correct ratio can correct this.



<u>Fig. 9</u> Right side of equation (12a) for a particular phase guess of 0.25

Some notion of the curious situation here is afforded by Fig. 9. Here we began with a cosinusoidal sequence of frequency 0.13 and phase 0.25, which generated a target value of the DC ratio of DRATIO= 0.187724707873 when we choose N=20. Further, using N=20 we find a peak in the magnitude of the DFT at k=3 and the second largest magnitude at k=2. Accordingly, we expect an answer between k=2 and k=3. But we have no idea at all about the phase. In Fig. 9, we have plotted the right side of equation (12a) by guessing a phase of 0.25 (the right answer) and scanning k_c between 0 and 5. This ratio of cosines predictably is periodic and blows up at certain points (the vertical lines which go to infinity and are off scale). We note that between k_c=2 and k_c=3, there are two crossings of the DRATIO horizontal line. One is vertical at about 2.3 while the other crosses with finite derivative at 2.6. The one at k_c=2.6 is the correct answer. But we cheated – we initially scanned k_c knowing the correct phase.

Fig. 10 shows a second scan where we also show (dotted) the case where phase is assumed to be 0.55. There is no reason why we should reject the intersection at about k_c=2.5 with this phase. In general, we would scan phases between 0 and 2π, each of which we would expect to give plausible crossings of the DRATIO line. Curiously, the program seems to ignore these possibilities. This has to mean that the error tolerance

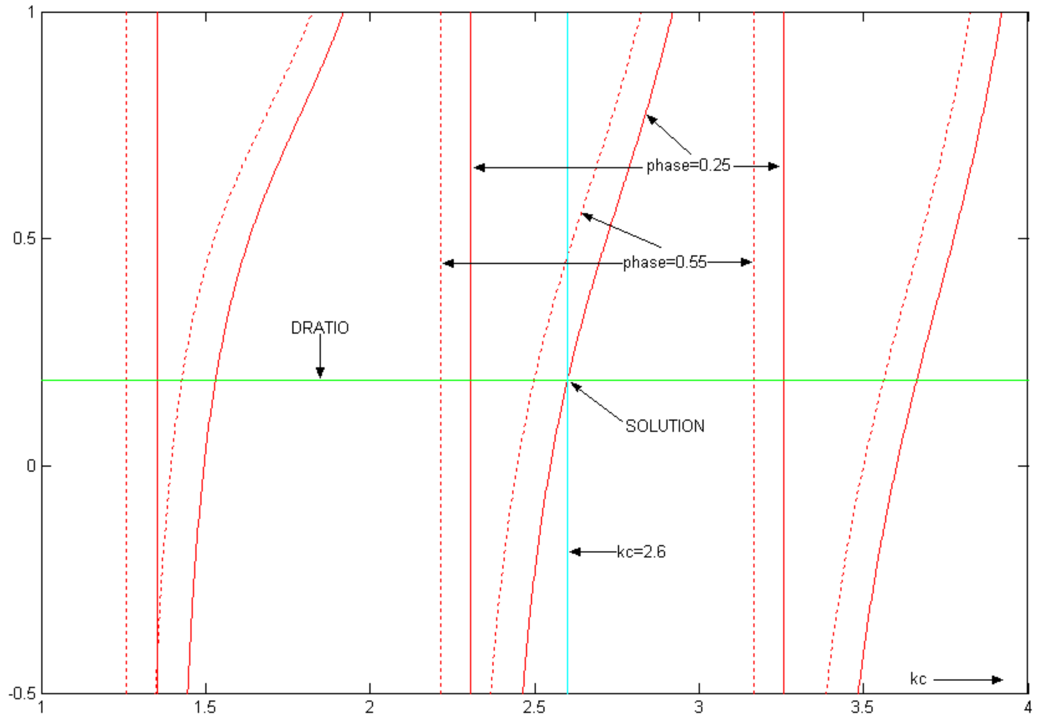


Fig. 10 In addition to the “correct” phase of 0.25 (solid), here we show a scan for a guess that the phase is 0.55 (dotted) which also gives what would appear to be a good answer at about kc=0.25.

in the program is so small that only the true solution is within the error. We note that increasing the tolerance and/or decreasing the scan intervals can lead to some spurious hits.

Fig. 11 gives the correct insight into what is going on. Here we have shown scans for the region around the correct answer. The phase guesses are shown on the plot (with the calculated points nearly upright marked with + signs) and k_c is scanned at intervals of 0.002. For the phase of 0.25, note that we have an exact (within tolerance) solution, while the other phases miss by an amount that exceeds the error tolerance (which is much too small to appear on the plot.

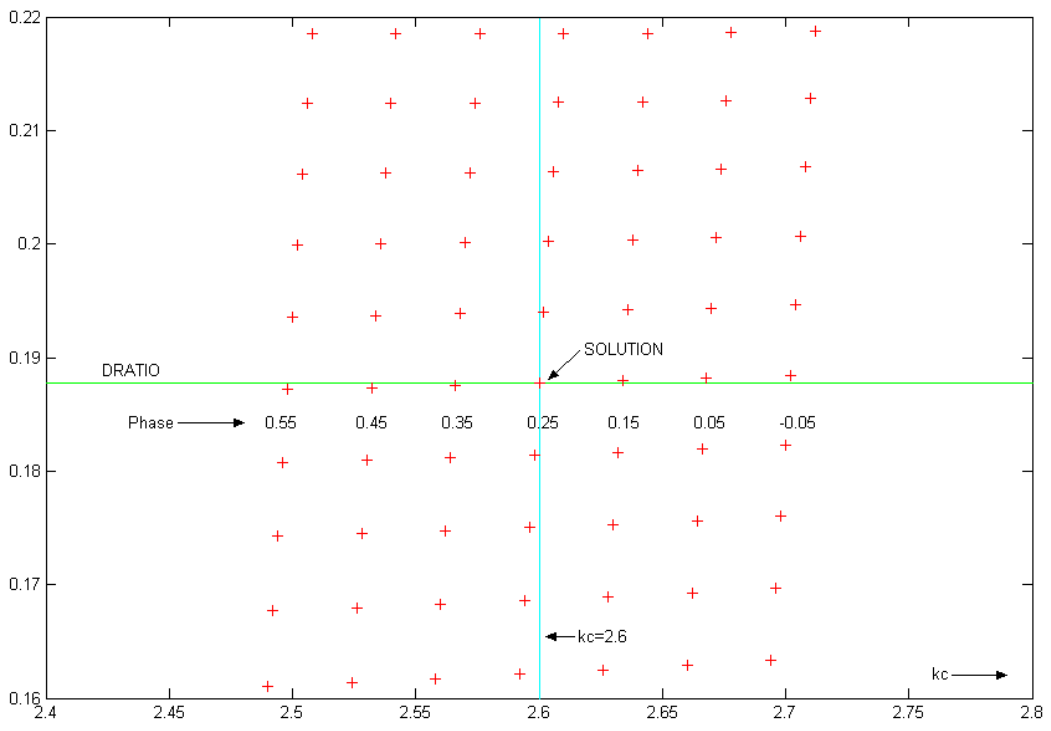


Fig. 11 In the details, we see that the scan for different phases misses DRATIO by amounts exceeding the specified error tolerance.

We now have more insight into why the program behaves the way it does. What we still have to discuss is why the exact answer is the one that comes out. It is one thing to say why most scans miss, but why does the exact scan hit?

Well, we have not taken the necessary precautions to “stir” the data. That is, we chose a frequency of 0.13 and phase of 0.25 for x(n) and then scanned a range, and in increments that assured hitting these values, so a very small error tolerance blocked all but the correct answer.

We can choose these parameters at random. At the same time, in order to get hits, we need to increase the error tolerance. At some times, this will mean that there are more than one answer during a scan. At others, there may be no answer. But we also know that we have the option of changing N, so we can calculate for a range of N (perhaps N=100 to N=200) and average the results. Preliminary results indicate that we can get the frequency to about 1% using this approach, even though the phase is not well-determined at all. Possibly the phase is less easily obtained from the longer DFT lengths used here.

In the end, this has been an interesting exercise, and our simple scan for a solution of one non-linear equation in two unknowns probably worked better than we had the right to expect. Almost certainly it could be improved – if it were worth our while.

The “ghost” hanging about here is equation (14). Note that this is time-domain and tells us only that two samples of a sinusoidal are separated by a phase $2\pi k_o$, and this is not new of course. We can easily write down a similar equation for x(2). We would then have two non-linear equations in two unknowns – enough equations, but still non-linear. The solution would be the same as the starting point for the well-studied “Prony’s Method” approach (see for example, EN#179 discussion). The lessons here are that when we know there is only a single sinewave. the DFT profile is revealing, particularly as it greatly reduces a search range for the correct frequency.

APPENDIX - THE PROGRAM THAT MADE SOME OF THE FIGURES

```
%  rx.m

N=20
k0=3.2

ph=0
ph=ph+(2*pi/N)*k0

kc=[0:.01:N-0.01];
Rkc=exp(-j*((N-1)/N)*pi*kc).*(sin(pi*kc)./sin(pi*kc/N));

figure(1)
subplot(211)
plot(kc,real(Rkc))
axis([-2 22 -5 22])
grid
subplot(212)
plot(kc,imag(Rkc))
axis([-2 22 -16 16])
grid
figure(1)


figure(2)
plot(kc,abs(Rkc))
axis([-2 22 -4 24])
grid
figure(2)


x=cos(2*pi*k0*[0:N-1]/N + ph)
X=fft(x);

kcc1=kc-k0;
Rkcc1=exp(-j*((N-1)/N)*pi*kcc1).*(sin(pi*kcc1)./sin(pi*kcc1/N));
Rkcc1=Rkcc1*exp(j*ph);
kcc2=kc+k0;
Rkcc2=exp(-j*((N-1)/N)*pi*kcc2).*(sin(pi*kcc2)./sin(pi*kcc2/N));
Rkcc2=Rkcc2*exp(-j*ph);
```

```
figure(3)
plot(kc,abs(Rkcc1),'r:')
hold on
plot(kc,abs(Rkcc2),'g--')
hold off
axis([-2 22 -2 22])
grid
figure(3)

DR=Rkcc1+Rkcc2;
figure(4)
subplot(211)
plot(kc,real(DR))
hold on
plot([0:N-1],2*real(X),'r*')
hold off
grid
axis([-2 22 -22 22])
subplot(212)
plot(kc,imag(DR))
hold on
plot([0:N-1],2*imag(X),'r*')
hold off
grid
axis([-2 22 -22 22])
figure(4)

figure(5)
plot(kc,abs(DR))
hold on
plot([0:N-1],abs(DR(1:100:100*N)),'yo')
plot([0:N-1],2*abs(X),'r*')
hold off
grid
axis([-2 22 -2 22])
figure(5)

[DR(1:100:N*100);2*X]'

D00=(sin(pi*k0)/sin(pi*k0/N))*2*cos(((N-1)/N)*pi*k0+ph)
D10=(sin(pi*k0)/sin(pi*k0/N))*2*cos(((N-1)/N)*pi*k0+ph+2*pi*k0/N)
% ? line above



figure(6)
stem([0:N-1],x)
axis([-1 N+1 -1.2 1.2])
figure(6)
sumofx=sum(x)
```