ELECTRONOTES
1016 Hanshaw Rd
Ithaca, NY 14850
607-266-8492

APPLICATION NOTE NO. 365

April 15, 2006

FREQUENCY ANALYSIS/RESOLUTION WITH FEW SAMPLES

INTRODUCTION

When we have a relatively short time sequence, the DFT does not always seem to give the "correct" frequencies based on what we know or suspect about the time waveform (in addition to the samples themselves).

Comment 1: N numbers in and N numbers out.

Of course, the input x(n) to a DFT is only N real numbers for a real length N sequence. We thus expect the DFT X(k) to contain only N real numbers. In general, even for a real input sequence, the DFT gives us N complex numbers. However, this does not mean that we have twice as many real numbers, since the X(k) are symmetric: X(N-k)=X(k)*. (Here * implies complex conjugation.) The DFT has nothing to work with other than the N numbers. Anything we "know or suspect" about the sequence x(n) is not available to the DFT.

Comment 2: DFT as sampling of DTFT.

In general, an arbitrary signal will not have frequencies that correspond to frequencies that the DFT "knows." We are accustomed to compute a DFT as:

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

The DFT is a sampling (in frequency, indexed by k) of the "self-windowed" DTFT:

$$X(f) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nfT} \qquad \text{all } f \qquad (2)$$

(or by equivalent equations) where T is the sampling time, the reciprocal of the sampling frequency f_s. From this we note that:

$$X(k) = X(f = f_k = k/NT = kf_s/N) \qquad (3)$$

This equation is extremely useful because it tells us how to "calibrate" a DFT as a spectrum analyzer. A particular k implies a certain frequency kf_s/N. The possible frequencies are thus integer multiples of f_s/N. Because both x(n) and X(k) are discrete, x(n) and X(k) are periodic with period N. Thus x(n) may be obtained as samples of a sequence that is not periodic in N, but when represented by X(k), it is forced to be a weighted sum of DFT harmonics $f_k=kf_s/N$. In general this is of course artificial.

At the same time, only input frequencies that are multiples of f_s/N can be perfectly resolved (non-zero values only for k and N-k). A frequency between integer multiples of f_s/N will be a combination of all DFT frequencies, strongly favoring close values of k, and this phenomenon is called "leakage."

To understand leakage, we need to recognize that the DFT is "self windowing" in that it uses only x(n) for n=0 to N-1. The DTFT of this "rectangular window" (a length N running sum) is a periodic sinc which has zeros spaced at mf_s/N for integer values of m except for m=0. Thus the spacing of the zeros of the window is exactly the spacing of the DFT frequencies, equation (3). Further, we understand the DTFT of a sampled but infinite duration sinusoidal signal of frequency f to consist of "spikes" at f, at -f, at f_s-f, at f_s+f, and in general in pairs spaced about all multiples of f_s. (This is just the spectral replications due to sampling in time.) This can be any frequency f, and is not in general a DFT harmonic (as we said above).

In order to understand what the DFT is supposed to give us, we first need to understand what the DTFT is, and then sample this DTFT. Because the signal x(n) is obtained by multiplying the infinite duration sinusoidal signal by a rectangular window, we need to convolve the DTFT of the sinusoidal signal (the spikes) with the DTFT of the discrete rectangular window (the periodic sinc). In a simplified view, we envision a single spike with the peak of a periodic since attached to its top. As the continuous frequency f moves up and down (as it would perhaps if we turned the frequency dial of a function generator), it drags the periodic sinc up and down with it.

Once we fix f to a specific value (stop turning the knob) we then obtain the DFT by sampling the now stationary DTFT. In the rare instance where f is an integer multiple of f_s/N, the DFT samples occur at the peak of the periodic sinc (for $k=Nf/f_s$, and at $k = N - Nf/f_s$) but is zero for all other values of k (Fig. 1). In the general case, sampling the DTFT gives non-zero values for all k, although we would expect spectral energy to be concentrated about $k=Nf/f_s$ (k not an integer in this case). If we saw the largest peak at, say, k=2 with a somewhat smaller peak at k=3 we might correctly surmise that some value of k between 2 and 3, closer to 2, would represent the truth (Fig. 2). Figures 1 and 2 were generated using dtftview.m from the appendix.

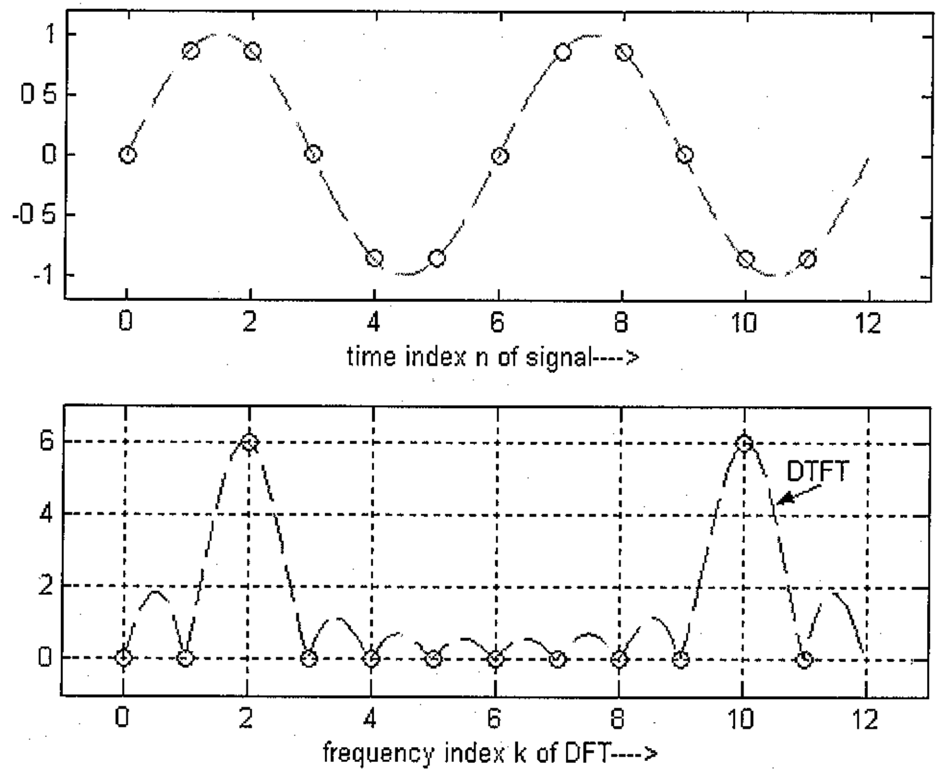

Fig. 1 Exactly two full cycles are resolved into just k=2 (and k=10)

Accordingly, the frequency resolution is expected to be poor if N is not large enough (no better than f_s/N). This might be why we would claim that the DFT did not give us a very good answer. But, this claim of poor performance is based on information the DFT does not have. The DFT has only got its N numbers - not the fact that we know or choose to believe that the samples x(n) are actually samples of a sinusoidal signal. In such a case, we interpret the fact that X(k) is non-zero everywhere as "leakage." But, we can get the exact same DFT by summing DFT harmonics. This is not difficult since the DFT itself gives us the required weights. In the absence of more information, we don't know that this direct summation ("DFT series," the usual inverse DFT) is not the truth.

Comment 3: Bandlimited in the sense of the DFT.

Just above we suggested that x(n), regardless of its actual origin, can be considered to be a weighted sum of DFT harmonics. This set of DFT harmonics is "forced" upon the samples because we are using the DFT. In general, even if the

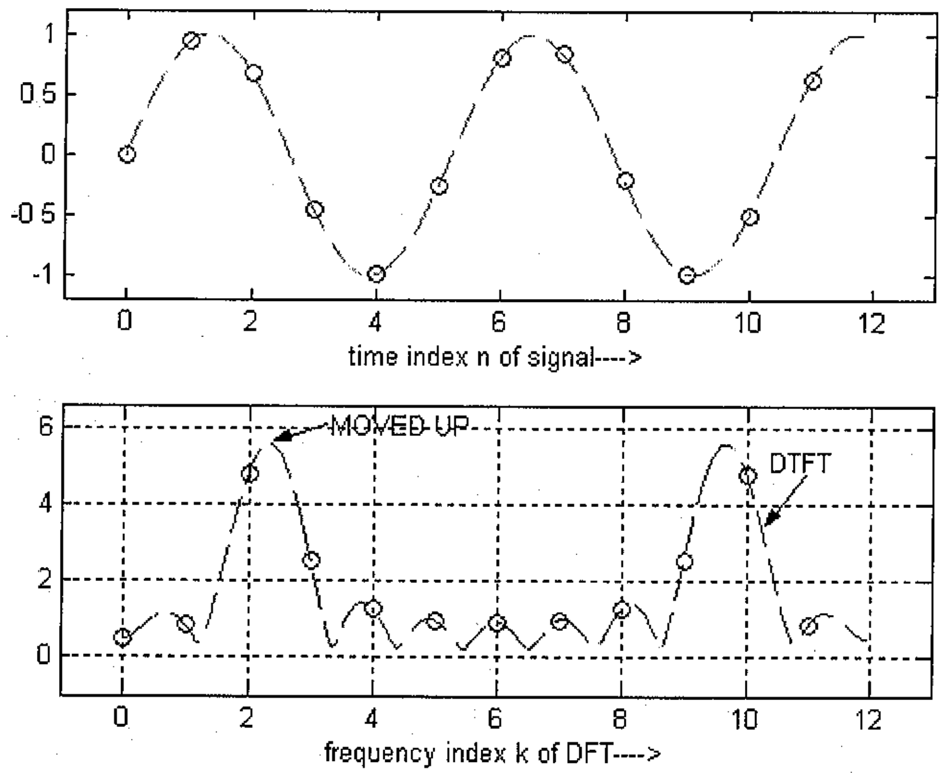

Fig. 2 Here the DTFT is dragged up from k=2 to k=2.3, while the sampling of the DFT, the DFT, remains at the integers, resulting in "leakage"

original signal is periodic, there is no reason to suppose we have chosen samples of exactly one full period. It would therefore seem informative to see just what this periodic waveform looks like. One way to do this would be to extract the amplitude, frequency, and phase from the DFT values, and reconstruct component by component. (Compare this to the well-known procedure of summing a Fourier series, which is similar - the difference is that the Fourier series uses frequencies that actually are the harmonics of the function being analyzed. See also AN-364 on computing Fourier series coefficients using the DFT.)

A far simpler way is to use DFT interpolation. To do this, we take our original time points x(n), take the DFT X(k), and then (properly) zero-pad the middle of the DFT to form a new DFT Y(k), and invert this to get the interpolated sequence y(n). This will return more details (an interpolation) of the function that it periodic and bandlimited to DFT harmonics. The program dftintrp.m from the toolbox allows us to examine these ideas.

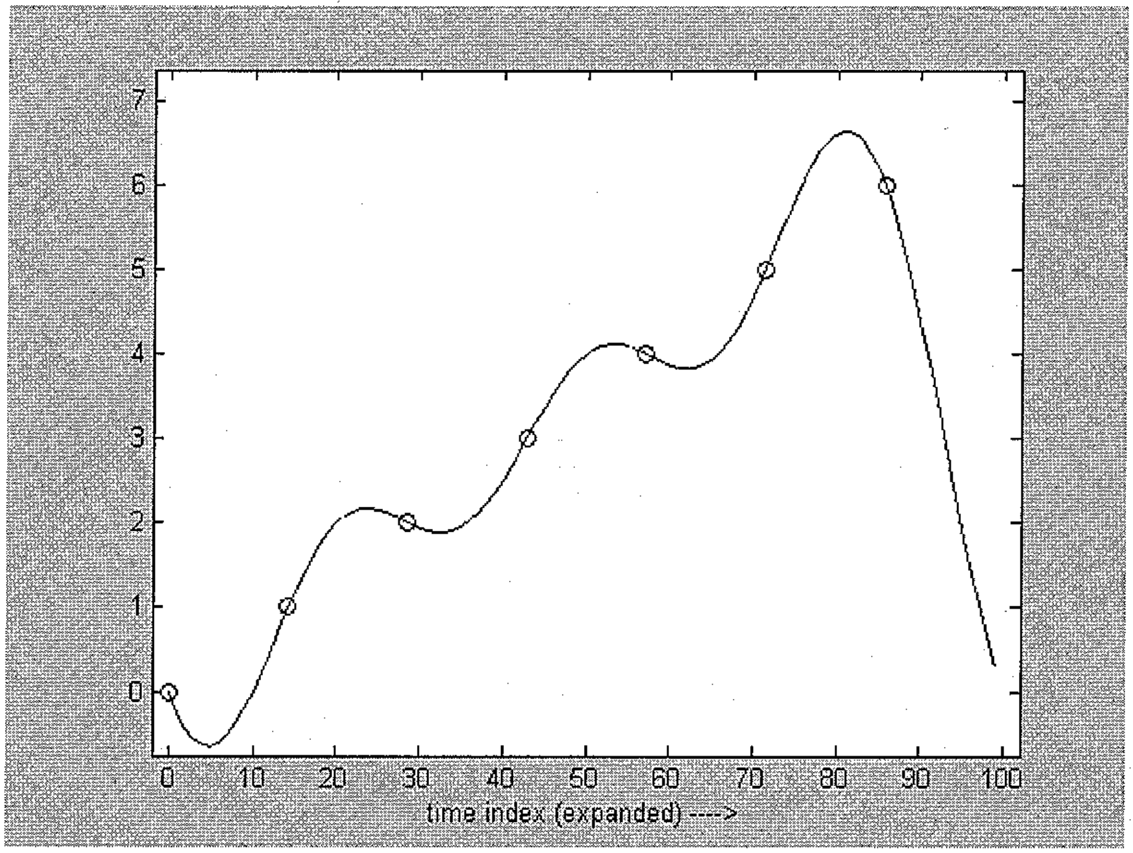

Fig. 3 Interpolating with the DFT – an expansion in DFT harmonics

Fig. 3, show an example where x(n) consists of six samples, 0, 1, 2, 3, 4, 5, and 6. Naturally we might think of these as samples of a ramp - but what ramp? Is it a ramp that keeps going up forever? Is it perhaps a portion of a periodic ramp that runs from 0 to 9 and then repeats? Or is it perhaps a periodic ramp that goes from 0 to 6 and then repeats? Clearly, whatever happens outside the limits of the six given samples will not affect the DFT results. What we are asking is what function, bandlimited in the sense of the DFT, goes through these six points. The waveform in Fig. 3 shows the six original points interpolated to 100 points total. If you will, this is the DFT's "best guess" as to what waveform the samples x(n) might have come from.

As we discuss in AN-364 on computing Fourier series coefficients using the DFT, the waveform of Fig. 3 is not the truncated Fourier series of a periodic ramp (a sawtooth waveform). It is a truncated sum of sinusoidal components, and there are four frequencies here (d.c., a "fundamental," and two harmonics). But they do not fall off as a Fourier series.

	Fourier Series	Length 7 DFT
fundamental	1	1
second harmonic	1/2	0.5550
third harmonic	1/3	0.4450

That is, while the length 7 DFT gives the correct Fourier series coefficients for the waveform of Fig. 3, these are not the truncated Fourier series coefficients of a sawtooth. There is no truncation here - the DFT is naturally bandlimited to its harmonics. One interpretation is that the DFT values contain all the harmonics through aliasing, and that is why they are larger than the Fourier series coefficients.

The use of the example with the ramp is interesting but here it serves mainly to illustrate the bandlimiting of the DFT, and we want to get back to a sinewave example.

Suppose we were give just three samples: 0, 0.5, and $\sqrt{3}/2$? Suppose we are also told that these are samples of a sinusoidal signal. Almost immediately we would get the correct answer by inspection as:

$$x(n) = \sin(2\pi n/12)$$

The problem where this is not obvious by inspection will be considered in Comment 4 below. But what does the DFT have to tell us about these three samples?

k	X(k)
0	1.3660
1	-0.6830 + 0.3170j
2	-0.6830 - 0.3170j

Here the DFT has analyzed the three given points into a dc term and just one frequency corresponding to $f_s/3$, or three points per cycle. By inspection, we saw a frequency of $f_s/12$, or 12 points per cycle. Of course, this is a horrendous frequency error.

Fig. 4 shows the interpolation of the three given points to a total of 100. True enough, we do find a frequency of 1/3, and we have three samples per cycle so there is no aliasing going on.

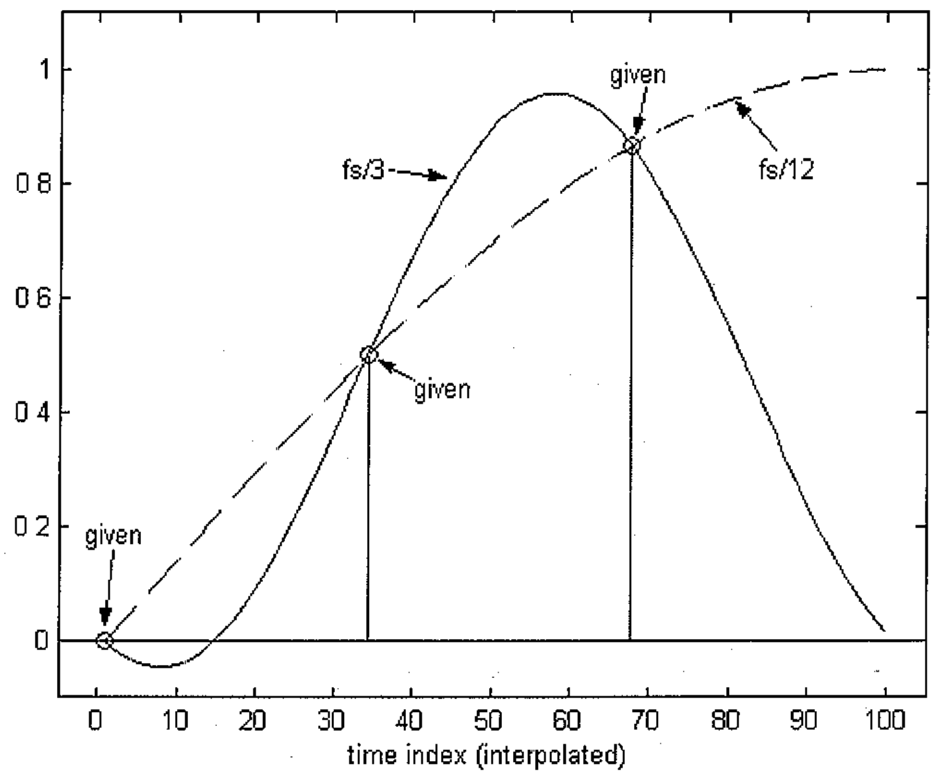

Fig. 4 DFT interpolation of three points fits the points with a frequency of $f_s/3$, the only DFT harmonic available. We would prefer a frequency of $f_s/12$.

Comment 4 Prony's Method

We have seen above that for a very small number of samples, the DFT can give a frequency that is not the "correct" one we know or suspect. We got a frequency that was $f_s/3$ with the DFT while we were looking for $f_s/12$. We can get exactly the right answer by using a procedure that uses an exponential model (the signal is assumed to be the sum of exponentially decaying sinusoidal signals). This sinusoidal assumption is information beyond what the DFT had as input. Here we use the prony.m program from the appendix. Fig. 5 shows a typical example of Prony's method (an example more general that a single sinusoidal waveform).

We can apply prony.m to our three samples 0, 0.5, and $\sqrt{3}/2$. Actually we will need a fourth sample for this case. The reason we need a fourth sample is because the program assumes decaying sinusoidals, and the fourth sample is needed to determine the decay parameter (determine that it is not decaying in this case). Fig. 6 shows the result of applying the second-order Prony model to the samples 0, 1/2, $\sqrt{3}/2$, and 1. This gives us exactly the right frequency of $f_s/12$.

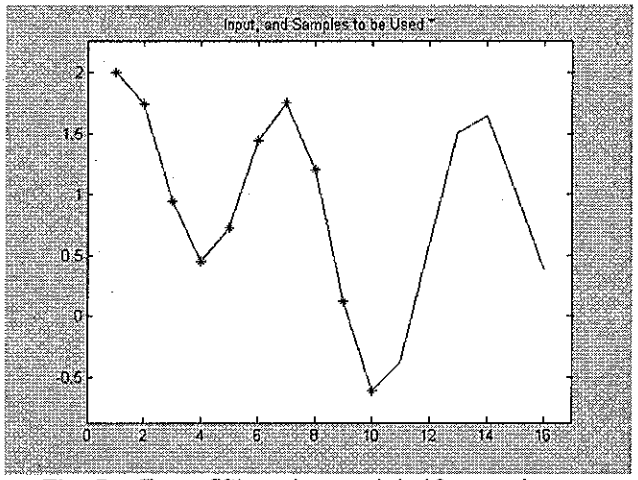

Fig. 5a For a fifth-order model, 10 samples are selected

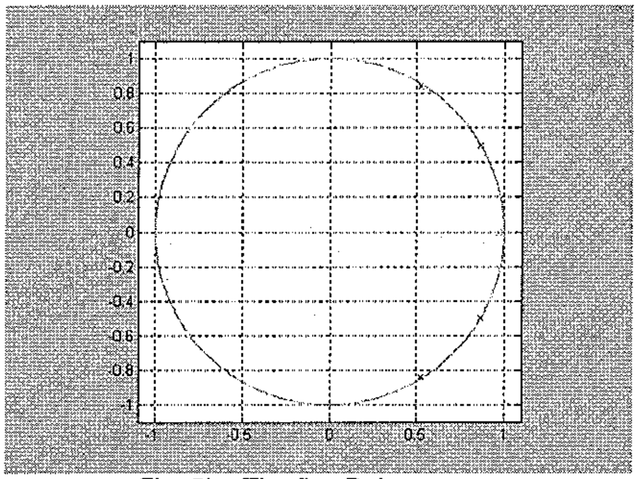

Fig. 5b The five Poles

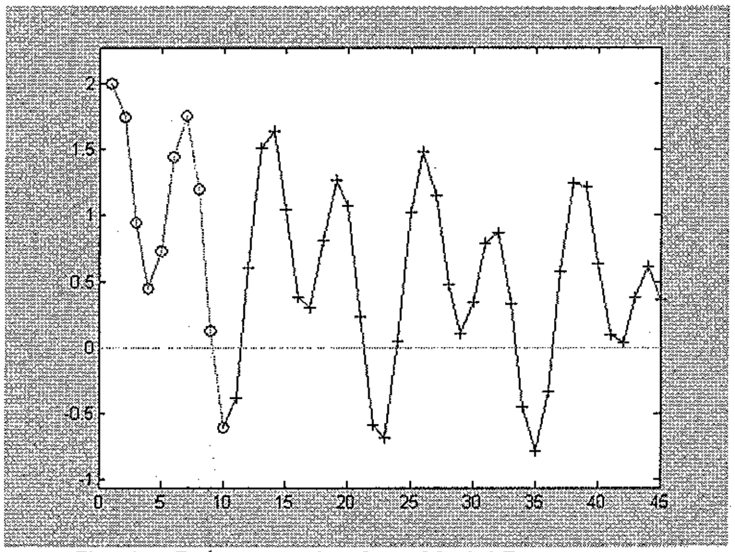

Fig. 5c Reconstruction from Model Parameters

Fig. 5 Typical Prony Example

This figure shows the results obtained with the example suggested in the help file for prony.m in the appendix:

```
n=0:15
x=0.5*sin(2*pi*n/12)
      +(0.99.^n).*cos(2*pi*n/6.2)
            +0.97.^n
[poles,coeff,ampl,radii,freq,phase]=prony(x,5)
```

The program chooses 10 samples of x (Fig. 5a) and solves for the poles (Fig. 5b). Using initial conditions, the sequence can then be reconstructed to the original 16, and indefinitely beyond (to 45 samples in Fig. 5c)

Fig. 6 Prony for 4 Samples

Prony's method as in Fig. 5 here applied to the four samples of the sinusoidal sequence. The result is 2nd-Order (two poles).

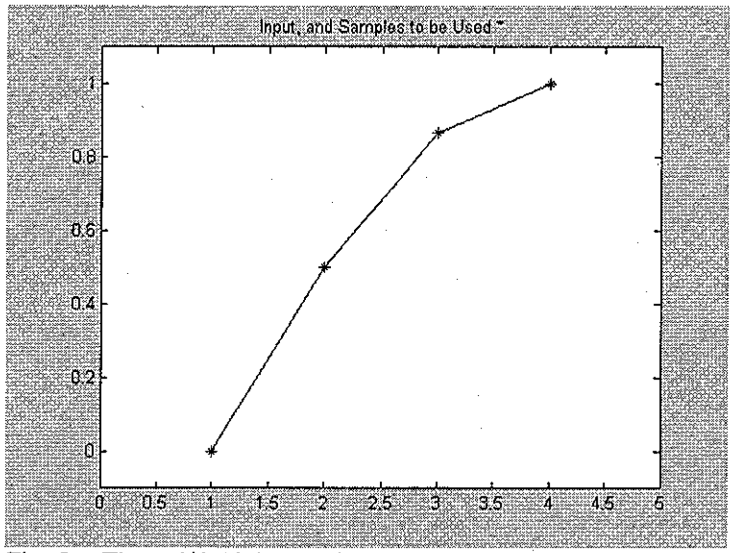

Fig. 6a The original 4 samples

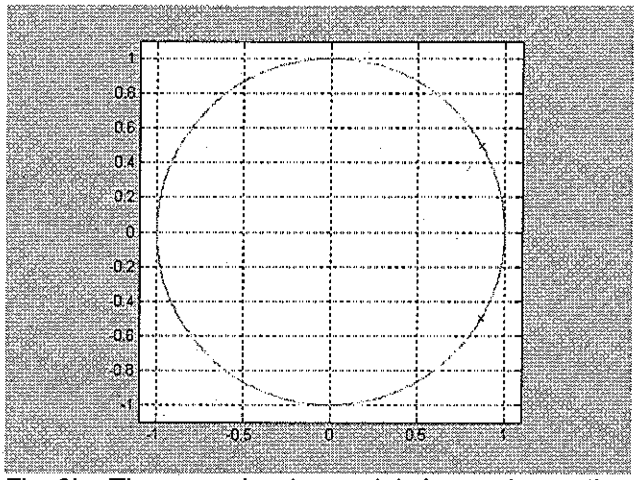

Fig. 6b The second-order model gives poles on the unit circle at a frequency $f_s/12$

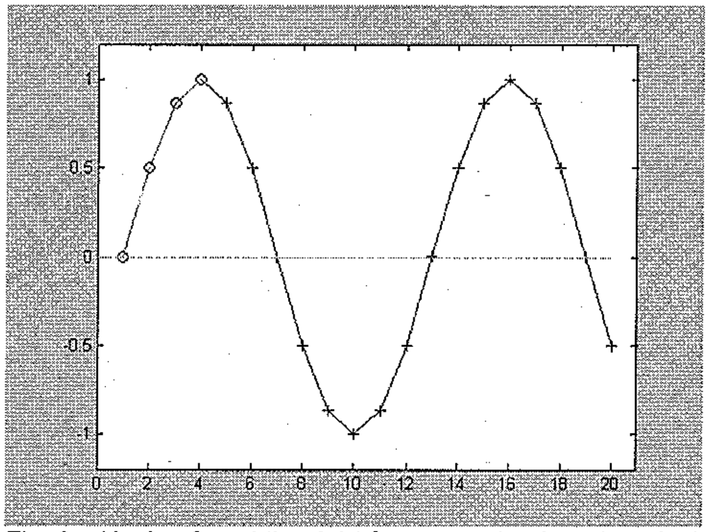

Fig. 6c Having found the right frequency (and no damping), initial conditions lead to construction of additional terms of the sequence.

APPENDIX

DTFTVIEW View the DTFT and DFT of a Sinusoidal Waveform

```
function dtftview(freq,N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   function dtftview(freq, N)
%            freq=frequency (normally 0 to 1/2)
%            N = number of DFT points to show
%
%   This program computes N points of a sinewave of
%   frequency f, and then it plots dense samples (100:1)
%   of the same sinewave xt.  xn (stemmed) and xt (dashed)
%   are plotted in the top subplot.
%
%   A third time-domain waveform is computed, xz is a
%   zero-padded (100:1) version of xn, the zeros being
%   added to the end of xn.
%
%   The FFT's (DFT's) of xn and of xz are computed.
%                    xn  <----->  XK
%                    xz  <----->  XZ
%   and are displayed, XK (stemmed) and XZ (dashed) in the
%   botton subplot.  We understand XZ to represssent the DTFT
%   of xn while XK is the DFT (sampled DTFT).
%
%   Note that xt is in some sense an "interpolation" of xn
%   and that XZ is in some sense an "interpolation" of XK,
%   but XZ is not the DFT of xt.
%
%   Examples:
%
%      dtftview(2/12,12) gives the DFT of 12 samples
%          representing exactly two full sinewave cycles,
%          and k=2 and k=10 are the only non-zero DFT values.
%          The DFT samples the DTFT at all k=0.....11
%
%      dtftview(2.3/12,12 gives the DFT of 12 samples
%          representing 2.3 cycles.  The DTFT peaks at about
%          k=2.3, but the DFT samples in at the integers.  The
%          DFT is non-zero for all integers k.  This is what
%          is called "leakage."
%
%      dtftview(2/12,13) also shows leakage
%
%   B. Hutchins                                 Spring 2006
% Compute xn, and xt
n=0:N-1;
t=0:.01:N-0.01;
xn=sin(2*pi*freq*n);
xt=sin(2*pi*freq*t);
figure(1)
subplot(211)
plot(t,xt,'g--')
hold on
plot(n,xn,'o')
axis([-1 N+1 -1.2 1.2]);
hold off
xlabel('time index n of signal---->')
figure(1)

% Zero-pad xn 100:1 to get xz and compute FFT's (DFT's)
xz=[xn zeros(1,99*N)];
XK=fft(xn);
XZ=fft(xz);
k=0:N-1;
fa=0:.01:N-0.01;
subplot(212)
plot(k,abs(XK),'o')
hold on
plot(fa,abs(XZ),'r--')
hold off
xlabel('frequency index k of DFT---->')
grid on
mx=max([abs(XK) abs(XZ)]);
axis([-1 N+1 -1 mx+1]);
figure(1)
```

DFTINTRP Interpolate with DFT

```
function y=dftintrp(x,N)
%  function y=dftintrp(x,N)
%  x is input x(n)
%  N = number of points to interpolate to
%
%  This program interpolates x(n) with length as given
%  to y(n) of length N.  The interpolation is through
%  the use of the DFT, by inserting zeros in the center
%  of the DFT.  In the case of even length for x, there is
%  a center term that must be split in two and separated.
%
%  The resulting interpolation is viewed as a dense set
%  of samples corresponding to a continuous time signal
%  that is composed of the DFT harmonics, and which goes
%  through the given points.
%
```

```
%  examples:  y = dftintrp([0 1 2 3 4 5 6],100)
%                   DFT bandlimited interpolation of a ramp
%             y = dftintrp(sin(2*pi*[0:2]/12),100)
%                   Interpolation of sinewave segment
%             y = dftintrp(sin(2*pi*[0:11]/12),100)
%                   Interpolation of full cycle

%
%  B. Hutchins                                Spring 2006


L=length(x);
X=fft(x);

% Odd length - just insert zeros in middle off FFT
if mod(L,2)==1
%
XR=[X(1:(L+1)/2) zeros(1,(N-L)) X((L+3)/2:L)];
xr=(N/L)*real(ifft(XR))
plot([0:N-1],xr)

hold on
pv=(N/L)*[0:L-1];
plot(pv,x,'o')
hold off
axis([-round(0.02*N) round(1.02*N) 1.1*min(xr)-0.05 1.1*max(xr)+0.05])
%
end

% Even length - split middle term and insert one fewer zero
if mod(L,2)==0
%
XR=[X(1:(L/2)) X((L+2)/2)/2 zeros(1,(N-L-1)) X((L+2)/2)/2 X((L+4)/2:L)];
xr=(N/L)*real(ifft(XR));
plot([0:N-1],xr)
hold on
pv=(N/L)*[0:L-1];
plot(pv,x,'o')
hold off
axis([-round(0.02*N) round(1.02*N) 1.1*min(xr)-0.05 1.1*max(xr)+0.05])
%
end

xlabel('time index (expanded) ---->')
figure(1)

y=xr;
```

PRONY

```
function [poles,coeff,ampl,radii,freq,phase]=prony(x,N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [poles,coeff,ampl,radii,freq,phase]=prony(x,N)
%
%    PRONY MODELING
%
%       OUTPUT
%              f     = frequencies of poles
%              r     = radii of poles
%              c     = coefficients
%              p     = phases
%
%       INPUT
%              x     = original signal
%              N     = number of poles to try
%
% Example:
%  n=0:15
%  x=0.5*sin(2*pi*n/12)+(0.99.^n).*cos(2*pi*n/6.2)+0.97.^n
%  [poles,coeff,ampl,radii,freq,phase]=prony(x,5)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  B. Hutchins     1994 2001 2006        Cornell Univ.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clf
%   Compute Length 2N Signal Vector for Model
%   Extract x1 from full x
idx=1:2*N;
x1=x(1:2*N);

%  Look at input
% Plot Input x and selected samples x1 as *
figure(1)
plot(x)
hold on
plot(idx,x1,'*')
xrng=(max(x)-min(x))/10;
axis([0 length(x)+1 min(x)-xrng max(x)+xrng]);
hold off
title('Input, and Samples to be Used *')


% PRONY METHOD PART 1,  Find the Poles
%  giving frequency and decay constants
%
%  Solve for Difference Equation Coefficients
%       Set Up Matrix b and Vector d1
```

```
  for K=1:N
       d1(K)=x1(N+K);
       for L = 1:N
          b(K,L)=x1(K+L-1);
       end
  end
% Solve for Coefficients a
a = inv(b)*d1';

% Find Poles
d=[-a',1];
d=fliplr(d);     % denominator of network = char. poly.
poles=roots(d);  % poles
r=abs(poles);    % pole radii
an=angle(poles);  % pole angles
figure(2)

% Plot poles
p1=roots(d);
pmax=max([abs(real(p1)); abs(imag(p1))]);
sc=ceil(pmax)+0.1;
n=0:500;
r1=exp(j*2*pi*n/500);
plot(r1,'g')
grid
hold on
plot(real(p1),imag(p1),'x')
hold off
axis('equal')
axis([-sc sc -sc sc])
% end plotting of poles

%
f=(an/(2*pi));  % actual pole frequencies
%  END PART 1

%
%

%  PRONY METHOD PART 2, Apply Initial Conditions
%            to find amplitude and phase
for K=1:N

   for L = 1:N
      cc(K,L)=poles(L)^(K-1);
   end
end

for K=1:N
   g(K)=x1(K);
end
c=inv(cc)*g';
magc=abs(c);
%                  END PART 2
%   Test Reconstruction to length 10N
for K=1:10*N
  xr(K)=0;
  for L=1:N
     xr(K)=xr(K)+c(L)*poles(L)^(K-1);
  end
end
xr=real(xr);


%  PLOTS
figure(3)
plot(x,'g')
hold on
pv1=[1:2*N];

xr1=xr(1:2*N);
plot(pv1,xr1,'o')
pv2=[(1+2*N):(1+10*N-1)];
xr2=xr((2*N+1):length(xr));
plot(pv2,xr2,'+')
lpv1=length(pv1);
pv2p=[pv1(lpv1), pv2];
xr2p=[xr1(lpv1), xr2];
plot(pv2p,xr2p,'r');
plot([0 10*N],[0 0],'c')
mx=max([xr2 x]);
mn=min([xr2 x]);
rg=0.1*(mx-mn);
axis([0 length(xr2)+5 mn-rg mx+rg]);
hold off


% phase relative to a cosine input
p=360*angle(c)/(2*pi);


radii=r;
coeff=c;
ampl=2*abs(coeff);
% correct amplitude for real poles
for k=1:N
   if abs(imag(poles(k)))<0.000001
   ampl(k)=ampl(k)/2;
   end
end
freq=f;
phase=p;
```