Monday 2012-12-17 Dan Ellis <dpwe@ee.columbia.edu>

---

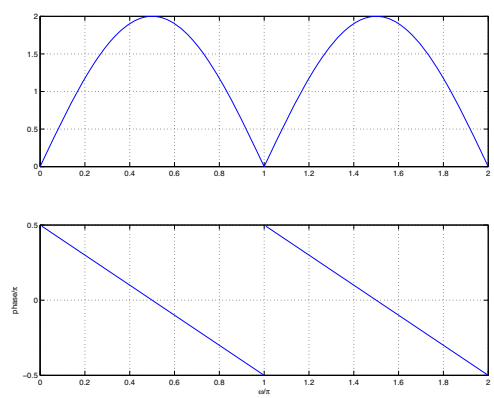
1. (a) This is a 3-point finite impulse response (FIR) filter. It is in fact antisymmetric ($h[n] = -h[N-n]$), so it is a type-III linear-phase FIR filter (odd-length, antisymmetric).

   (b) We have certain expectations for the magnitude and phase response based on the system being a type-III linear phase FIR, but let's work it out from scratch:

$$H_1(e^{j\omega}) = \sum_n h_1[n]e^{-j\omega n} = 1 - e^{-2j\omega} \tag{1}$$
$$= e^{-j\omega}(e^{j\omega} - e^{-j\omega}) \tag{2}$$
$$= e^{-j\omega}2j\sin\omega \tag{3}$$

   Thus its magnitude is given by $|\sin\omega|$ and its phase is the phase of $je^{-j\omega}$, where the j term means the phase starts at $\pi/2$ for $\omega = 0$. There is be a jump of π when the $|\sin|$ tern flips sign. Here it is in Matlab:

```
>> h1 = [1 0 -1];
>> [H1,W] = freqz(h1, 1, ...
      [0:pi/128:2*pi]);
>> subplot(211)
>> plot(W/pi, abs(H1));
>> grid
>> subplot(212)
>> plot(W/pi, angle(H1)/pi);
>> grid
>> xlabel('\omega/\pi')
>> ylabel('phase/\pi')
```

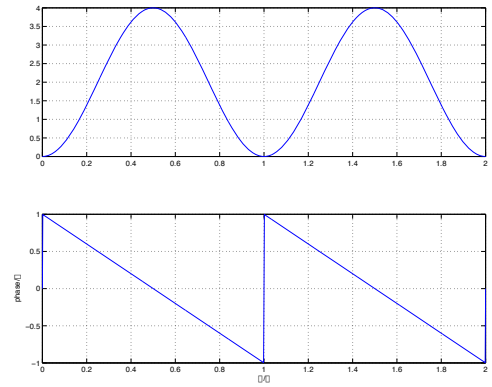


   (c) A system composed of two instances of H_1 in series has the effect of applying its frequency response twice, hence:

$$H_2(e^{jw}) = (H_1(e^{jw}))^2 \tag{4}$$
$$= e^{-2j\omega}(-4)\sin^2\omega \tag{5}$$
$$= -e^{-2j\omega}2(1-\cos 2\omega) \tag{6}$$

   The magnitude response is simply the square of the one from part (b), which appears as a sinusoid shifted to be non-negative. The phase response is simply doubled from the one of part (b), which appears as a linear phase term with twice the slope, starting at π because of the negative sign:

```
>> h2 = conv(h1,h1);
>> [H2,W] = freqz(h2,1,[0:pi/512:2*pi]);
>> subplot(211)
>> plot(W/pi, abs(H2));
>> grid
>> subplot(212)
>> plot(W/pi, angle(H2)/pi);
>> grid
>> xlabel('\omega/\pi')
>> ylabel('phase/\pi')
```



(d) We didn't actually calculate the impulse response $h_2[n]$ in part (b), but it is simply $h_1[n]$ convolved with itself, i.e.

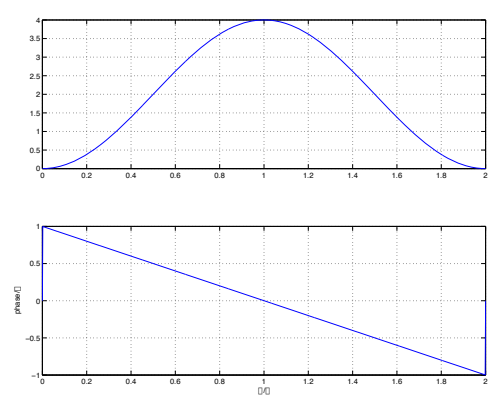
$$\begin{aligned} h_2[n] &= h_1[n] \circledast h_1[n] && (7)\\ &= h_1[n] \circledast (\delta[n] - \delta[n-2]) && (8)\\ &= h_1[n] - h_1[n-2] && (9)\\ &= \{\underline{1}, 0, -2, 0, 1\} && (10) \end{aligned}$$

Factor-of-two decimation means to discard every odd-indexed sample, and close up, so $h_3[n] = \{\underline{1}, -2, 1\}$. We notice that the discarded samples were zero anyway, which is to say that if we re-interpolated $h_3[n]$ by a factor of 2, we would get $h_2[n]$ back exactly. This is an instance of a comb filter: $h_2[n]$ is a comb filter, and $h_3[n]$ is the parent system, whose frequency response is replicated, in this case twice, in the comb filter response. Thus H_3's frequency response is simply the first half of H_2's, stretched out by a factor of two. But let's make sure:

$$\begin{aligned} H_3(e^{j\omega}) &= \sum_n h_3[n] e^{-j\omega n} && (11)\\ &= 1 - 2e^{-j\omega} + e^{-2j\omega} && (12)\\ &= e^{-j\omega}(e^{j\omega} - 2 + e^{-j\omega})) && (13)\\ &= -e^{-j\omega} 2(1 - \cos\omega) && (14) \end{aligned}$$

.. which is indeed simply $H_2(e^{j\omega/2})$. We can plot it:

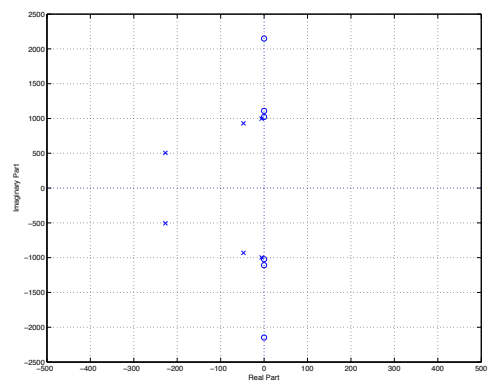
```
>> h3 = h2(1:2:end);
>> [H3,W] = freqz(h3,1,[0:pi/512:2*pi]);
>> subplot(211)
>> plot(W/pi, abs(H3));
>> grid
>> subplot(212)
>> plot(W/pi, angle(H3)/pi);
>> grid
>> xlabel('\omega/\pi')
>> ylabel('phase/\pi')
```



2. (a) We have been told that the four IIR analog filters we covered in class are the optimal minimax (minimizing the maximum error) filters of their type. This one has ripples in both the stop band and the pass band, so it must be an elliptical (Cauer) filter. An elliptical filter is parameterized by its order (giving the number of zeros and poles), its cutoff frequency, its passband ripple, and its minimum stopband attenuation. Here, we can see evidence of three zeros on the frequency axis (the notches in the stopband), and three poles (the maxima of the ripples in the passband). These will be reflected in negative frequency, so we would guess that the system is order 6. The passband edge is clearly very close to $\Omega = 1000$ rad/s, the passband ripples appear to be about 3 dB deep, and the greatest gain in the stoppband is about -30 dB. The actual form of these numbers used in design depends on the design technique used, but for Matlab we simply use these values. Thus, this filter was designed with the Matlab command `[b,a] = ellip(6,3,30,1000,'s')`.

(b) We are being asked to sketch an s-domain pole-zero diagram, and we expect to include 6 poles and 6 zeros, in groups of 3 mirrored in the real axis (for positive and negative frequencies). The three zeros will be on the imaginary (frequency) axis at about $\Omega = 1005, 1090, 2150$ rad/s (just from reading the log-scaled horizontal axis of the plot), and the three poles will have their imaginary values given approximately by the locations of the peaks in the passband around $\Omega = 490, 910, 998$ rad/s, and their real parts reflecting the narrowness of those peaks, getting closer to the imaginary axis as frequency increases. From geometric considerations, a single pole would be -3 dB down at a distance from its center frequency equal to the negative of its real part, so we may estimate the real parts to be around $200, 50, 5$ rad/s. In fact, here is the true s-plane plot:

```
>> [B,A] = ellip(6,3,30,1000,'s');
>> zplane(B,A)
>> axis([-500 500 -2500 2500])
>> axis normal
>> grid
```



Notice that the real axis is zoomed-in relative to the imaginary axis, and that the poles and zeros around $\Omega = 1000$ are very close together.

(c) To derive a discrete-time filter, we will apply the bilinear transform, which will preserve the magnitude values from the analog prototype, but warp the frequency axis according to the relationship $\omega = 2\tan^{-1}\Omega$. In order to have the the DT filter's cutoff be at $\pi/4$ rad/samp, the original filter must have its frequency axis scaled by a factor Ω_0 so that:

$$\pi/4 = 2\tan^{-1}(1000/\Omega_0) \tag{15}$$

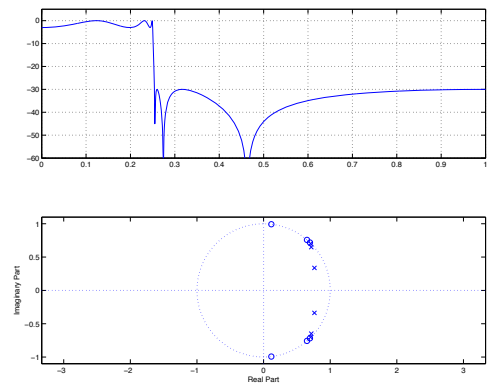
$$\rightarrow \tan(\pi/8) = 1000/\Omega_0 \tag{16}$$

$$\rightarrow \Omega_0 = 1000/\tan(\pi/8) \approx 2,414 \tag{17}$$

With this mapping, we expect passband ripple maxima at around $\omega = 2\tan^{-1}(\{490, 910, 998\}/2414) \approx \{0.13, 0.23, 0.25\}\pi$ and stopband notches at $\omega = 2\tan^{-1}(\{1005, 1090, 2150\}/2414) \approx \{0.25, 0.27, 0.46\}\pi$.

(You couldn't get full marks without some effort to map the precise frequency locations of these roots). The ripples will still be 3 dB in the passband and -30 dB in the stopband. Knowing Ω_0, we can use Matlab's `bilinear` function to do the actual mapping:

```
>> [a,b] = bilinear(A,B,0.5/Omega0);
>> [H,W] = freqz(b,a,[0:pi/1024:pi]);
>> subplot(211)
>> plot(W/pi, 20*log10(abs(H)));
>> grid
>> axis([0 1 -60 5])
>> subplot(212)
>> zplane(b,a)
```



The exact shape of the poles relative to the unit circle was not critical, but they needed to be shown getting close as frequency increased.

(d) The idea here is that the system is being fed a mixture of two (real) sinusoids, one at zero frequency, and one at $\omega = \pi$ rad/samp. We're only being asked for the DTFT magnitude, meaning that we don't have to worry about phase. And we don't know the precise details of the filter, so we're just being asked to estimate. We can write $x[n] = 1 + 0.5e^{j\pi n} + 0.5e^{-j\pi n}$, so its DTFT is simply $X(e^{j\omega}) = \sum_r \delta(\omega - 2\pi r) + 0.5\delta(\omega - \pi - 2\pi r) + 0.5\delta(\omega + \pi - 2\pi r)$ (where r cycles over the repeating periods of a discrete-time spectrum), and the output will simply have each of these line-spectral components scaled by the magnitude response at those frequencies. From our plot in part (c), we estimate the gain at $\omega = 0$ to be -3 dB, and at $\omega = \pi$ it actually looks to be around -43 dB, although your sketch could have put this at any value below -30 dB. Thus, the magnitude of the d.c. component is scaled by a gain of about $1/\sqrt(2) \approx 0.7$, and the the other component experiences a scaling by about $10^{-43/20} \approx 0.007$, and hence $|Y(e^{j\omega})| \approx \sum_r 0.7\delta(\omega - 2\pi r) + 0.0035\delta(\omega - \pi - 2\pi r) + 0.0035\delta(\omega + \pi - 2\pi r)$.

3. (a) For a 4-point DFT, the constant $W_4 = e^{-2j\pi/N}|_{N=4} = e^{-j\pi/2} = -j$. Thus, in matrix form, the entire DFT calculation is simply:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} \tag{18}$$

(b) Because "multiplying" by j simply involves interchanging the real and imaginary parts of a cartesian representation, and flipping the sign of the new real part, a 4-point DFT can be calcu-

lated without any actual real multiplies. Thus, re-expressing the above, we get:

$$\mathrm{Re}\{X[2]\} = \mathrm{Re}\{x[0]\} + \mathrm{Re}\{x[1]\} + \mathrm{Re}\{x[2]\} + \mathrm{Re}\{x[3]\} \tag{19}$$
$$\mathrm{Im}\{X[2]\} = \mathrm{Im}\{x[0]\} + \mathrm{Im}\{x[1]\} + \mathrm{Im}\{x[2]\} + \mathrm{Im}\{x[3]\} \tag{20}$$
$$\mathrm{Re}\{X[1]\} = \mathrm{Re}\{x[0]\} + \mathrm{Im}\{x[1]\} - \mathrm{Re}\{x[2]\} - \mathrm{Im}\{x[3]\} \tag{21}$$
$$\mathrm{Im}\{X[1]\} = \mathrm{Im}\{x[0]\} - \mathrm{Re}\{x[1]\} - \mathrm{Im}\{x[2]\} + \mathrm{Re}\{x[3]\} \tag{22}$$
$$\mathrm{Re}\{X[2]\} = \mathrm{Re}\{x[0]\} - \mathrm{Re}\{x[1]\} + \mathrm{Re}\{x[2]\} - \mathrm{Re}\{x[3]\} \tag{23}$$
$$\mathrm{Im}\{X[2]\} = \mathrm{Im}\{x[0]\} - \mathrm{Im}\{x[1]\} + \mathrm{Im}\{x[2]\} - \mathrm{Im}\{x[3]\} \tag{24}$$
$$\mathrm{Re}\{X[3]\} = \mathrm{Re}\{x[0]\} - \mathrm{Im}\{x[1]\} - \mathrm{Re}\{x[2]\} + \mathrm{Im}\{x[3]\} \tag{25}$$
$$\mathrm{Im}\{X[3]\} = \mathrm{Im}\{x[0]\} + \mathrm{Re}\{x[1]\} - \mathrm{Im}\{x[2]\} - \mathrm{Re}\{x[3]\} \tag{26}$$

(c) Slide 10 of slidepack L10 actually provides this figure, albeit before the optimizations of the final stage are applied. When we use the optimized final stage from slide 15, we are left with just three twiddle factors, W_8, W_8^2, and W_8^3, compared to the 5 multiplies in the full radix-2 implementation of slide 15. We might also note that W_8^2 is simply $-j$, which again can be accomplished by interchanging real and imaginary, so we only need 2 complex multiplies (four real multiplies) – although this optimization could also be applied to the radix-2 version, and similarly for the two W_4 extra twiddle factors that appear on that slide. In fact, the two implementations are largely equivalent, and of course they give the same result.

4. Unfortunately, there was a typo in the key formula in this question, which did not get discovered until near the end of the exam: the plus sign immediately before the summation in the DCT definition was missing, making it appear as if the entire summation was to be multiplied by the $\frac{1}{2}(x[0]+(-1)^k x[N])$ term. We will first interpret the question as written, i.e.

$$X[k] = \frac{1}{2}(x[0] + (-1)^k x[N]) \cdot \sum_{n=1}^{N-1} x[n] \cos \frac{\pi n k}{N} \quad \text{for } k = 0 \ldots N \tag{27}$$

(which is strange, but not entirely impossible), but please do not go away thinking this is the DCT!

(a) The unconventional indexing of $y[n]$ was supposed to make more obvious the relationship between the DCT and the DFT, but in the event it was just a meaningless confusion (red herring). We can approach this by writing out the full-form of the DFT of y, then pulling out and grouping some terms to make the summation range look more like the x expression:

$$Y[k] = \sum_{n=-(N-1)}^{N} y[n] e^{-j\frac{2\pi n k}{2N}} \tag{28}$$
$$= y[0] + y[N]e^{-j\pi k} + \sum_{n=1}^{N-1} (y[n]e^{-j\frac{2\pi n k}{2N}} + y[-n]e^{j\frac{2\pi n k}{2N}}) \tag{29}$$

If we assume that $y[N] = y[0] = 0$, and $y[n] = y[-n]$ for $0 < n < N$, we can further simplify the summation:

$$Y[k] = \sum_{n=1}^{N-1} y[n](e^{-j\frac{2\pi n k}{2N}} + e^{-j\frac{2\pi n k}{2N}}) \tag{30}$$
$$= \sum_{n=1}^{N-1} y[n](2\cos\frac{\pi n k}{N}) \tag{31}$$

Now, the question states that

$$Y[k] = X[k] = \frac{1}{2}(x[0] + (-1)^k x[N]) \sum_{n=1}^{N-1} x[n] \cos \frac{\pi n k}{N} \quad \text{for } k = 0 \dots N \tag{32}$$

To make this more tractable, let's break $Y[k]$ into two parts, $Y[k] = Y_e[k] + Y_o[k]$. Since the DFT is linear, these are the DFTs of two time-domain components, $y[n] = y_e[n] + y_o[n]$. Let us have:

$$Y_e[k] = \frac{1}{2} x[0] \sum_{n=1}^{N-1} x[n] \cos \frac{\pi n k}{N} \tag{33}$$

$$Y_o[k] = \frac{1}{2}(-1)^k x[N] \sum_{n=1}^{N-1} x[n] \cos \frac{\pi n k}{N} \tag{34}$$

If we can find both y_e and y_o in terms of x, we can sum them to get our overall solution. From eqn. (31), for Y_e, we get:

$$\sum_{n=1}^{N-1} 2y_e[n] \cos \frac{\pi n k}{N} = \frac{1}{2} x[0] \sum_{n=1}^{N-1} x[n] \cos \frac{\pi n k}{N} \tag{35}$$

$$\tag{36}$$

which will work provided we match every term in the summations, i.e.:

$$2y_e[n] = \frac{1}{2} x[0] x[n] \tag{37}$$

So the full expression for y_e becomes:

$$y_e[n] = \begin{cases} \frac{x[0]x[n]}{4} & 0 < n < N \\ \frac{x[0]x[-n]}{4} & -N < n < 0 \\ 0 & n = 0, n = N \end{cases} \tag{38}$$

i.e. it is the x sequence, made symmetric by reflecting in $n = 0$, with the $n = 0$ and $n = N$ points set to zero.

Turning now to y_o, note that:

$$Y_o[k] = (-1)^k \frac{x[N]}{x[0]} Y_e[k] \tag{39}$$

i.e., Y_o is just a scaled version of Y_e, but with every other frequency component having its sign flipped. Multiplying by $(-1)^k$ in the frequency domain is the same as convolving with $\text{IDFT}_{2N}\{(-1)^k\} = \delta(n - N)$ in the time domain, i.e. an N-point circular delay, so y_o is given by:

$$y_o[n] = \begin{cases} \frac{x[N]x[N-n]}{4} & 0 < n < N \\ \frac{x[N]x[N+n]}{4} & -N < n < 0 \\ 0 & n = 0, n = N \end{cases} \tag{40}$$

i.e., the sequence $x[1 \dots N-1]$ reversed, scaled, then made symmetric by reflecting at $n = 0$.

We can now sum these two parts to give the full answer:

$$y[n] = y_e[n] + y_o[n] = \begin{cases} \frac{1}{4}(x[0]x[n] + x[N]x[N-n]) & 0 < n < N \\ \frac{1}{4}(x[0]x[-n] + x[N]x[N+n]) & 0 < n < N \\ 0 & n = 0, n = N \end{cases} \tag{41}$$

I tested this for $N = 16$, for a random sequence, in Matlab:

```
>> N = 16;
>> x = randn(1,N+1);  % a random N+1 point sequence
>> n = 1:(N-1);  % the range of the summation
>> % The expression in the original question.  I'm using 1+ indexing to please Matlab
>> for k = 0:N; ...
        X(k+1) = 0.5*(x(1)+((-1)^k)*x(1+ N))*sum(cos(pi*k*[1:(N-1)]/N).*x(1+ n)); ...
        end
>> % Our solution for y.  We'll now assume we're looking at y[n], n=0..2N-1
>> y = zeros(1,32);
>> y(1+ n) = 0.25*(x(1)*x(1+ n) + x(1+ N)*x(1+ N-n));
>> % Make the second half be the symmetric version
>> y(18:32) = y(16:-1:2);
>> % Now take its DFT.  Discard tiny imaginary values
>> Y = real(fft(y));
>> % Compare X and Y
>> [X;Y(1:(N+1))]
ans =
  Columns 1 through 8
    0.0029   -0.3582   -0.0479    0.8533    0.2396   -1.0686   -0.1718   -0.4059
    0.0029   -0.3582   -0.0479    0.8533    0.2396   -1.0686   -0.1718   -0.4059
  Columns 9 through 16
   -0.1679   -0.6251   -0.0771    0.9404    0.0531    0.2138    0.0085    0.4503
   -0.1679   -0.6251   -0.0771    0.9404    0.0531    0.2138    0.0085    0.4503
  Column 17
    0.3240
    0.3240
>> % Yes, they match
```

(b) This part was also not quite the question I meant to ask, but actually it's easier as it is. $x[n] = \cos \pi r n$ for integer r is either $x[n] = 1$ (constant) for even r, or $x[n] = (-1)^n$ for odd r. With $x[n] = 1$, we get:

$$X[k] = \frac{1}{2}(x[0] + (-1)^k x[N]) \cdot \sum_{n=1}^{N-1} x[n] \cos \frac{\pi n k}{N} \tag{42}$$

$$= \frac{1}{2}(1 + (-1)^k) \cdot \sum_{n=1}^{N-1} \cos \frac{\pi n k}{N} \tag{43}$$

$$= \begin{cases} 0 & k \text{ odd} \\ \sum_{n=1}^{N-1} \cos \frac{\pi n k}{N} & k \text{ even} \end{cases} \tag{44}$$

To solve the summation, we can refer to the identity for the completed range:

$$\sum_{n=0}^{N-1} \cos \frac{\pi n k}{N} = \begin{cases} N & k = 0 \\ 1 & k \text{ odd} \\ 0 & k \text{ even}, \neq 0 \end{cases} \tag{45}$$

This comes from pairing the cosine terms: for even k, positive and negative values pair off exactly, but for odd k the $n = 0$ term is left unpaired.
Thus, using the actual limits in the expression,

$$\sum_{n=1}^{N-1} \cos \frac{\pi nk}{N} = \sum_{n=0}^{N-1} \cos \frac{\pi nk}{N} - \cos \frac{\pi 0k}{N} = \begin{cases} N-1 & k = 0 \\ 0 & k \text{ odd} \\ -1 & k \text{ even, } \neq 0 \end{cases} \tag{46}$$

so the answer for even r is:

$$X[k] = \begin{cases} N-1 & k = 0 \\ 0 & k \text{ odd} \\ -1 & k \text{ even, } \neq 0 \end{cases} \tag{47}$$

Because of the $(-1)^k x[N]$ factor, the results for odd r (i.e., when $x[n] = (-1)^n$) are more involved. If N is even, then $x[N] = 1$, but if N is odd then $x[N] = -1$, so we need to consider these cases separately. For even N, we have:

$$X[k] = \frac{1}{2}(x[0] + (-1)^k x[N]) \cdot \sum_{n=1}^{N-1} x[n] \cos \frac{\pi nk}{N} \tag{48}$$

$$= \frac{1}{2}(1 + (-1)^k) \cdot \sum_{n=1}^{N-1} (-1)^n \cos \frac{\pi nk}{N} \tag{49}$$

$$= \begin{cases} 0 & k \text{ odd} \\ \sum_{n=1}^{N-1} (-1)^n \cos \frac{\pi nk}{N} & k \text{ even} \end{cases} \tag{50}$$

Now, $(-1)^n \cos \frac{\pi nk}{N} = \cos \pi n \cos \frac{\pi nk}{N} = \cos \frac{\pi n(N+k)}{N}$ (from $\cos(A + B) = \cos A \cos B - \sin A \sin B$, but $\sin \pi n = 0$), which is also equal to $\cos \frac{\pi n(N-k)}{N}$, so from eq. (45), we get:

$$\sum_{n=0}^{N-1} (-1)^n \cos \frac{\pi nk}{N} = \begin{cases} N & (N-k) = 0 \\ 1 & (N-k) \text{ odd} \\ 0 & (N-k) \text{ even, } \neq 0 \end{cases} \tag{51}$$

Thus for odd r, even N we have:

$$X[k] = \begin{cases} -1 & k \text{ even}, \neq N \\ 0 & k \text{ odd} \\ N-1 & k = N \end{cases} \tag{52}$$

For odd N, we have:

$$X[k] = \frac{1}{2}(x[0] + (-1)^k x[N]) \cdot \sum_{n=1}^{N-1} x[n] \cos \frac{\pi nk}{N} \tag{53}$$

$$= \frac{1}{2}(1 - (-1)^k) \cdot \sum_{n=1}^{N-1} (-1)^n \cos \frac{\pi nk}{N} \tag{54}$$

$$= \begin{cases} 0 & k \text{ even} \\ \sum_{n=1}^{N-1} (-1)^n \cos \frac{\pi nk}{N} & k \text{ odd} \end{cases} \tag{55}$$

Combining this with eq. (51), we get the solution for odd r and odd N:

$$X[k] = \begin{cases} 0 & k \text{ even} \\ -1 & k \text{ odd}, \neq N \\ N-1 & k = N \end{cases} \tag{56}$$

This was both much more difficult and much less interesting than I intended, but since it can be solved, we decided to go ahead and grade it anyway.

(c) This part of the question only makes sense with the real DCT expression. What I was trying to show was that the DCT closely matches the DFT for cosine-phase signals, but does a much worse job expressing sine-phase signals. However, since the thing I called the DCT in the question was not the DCT at all, there isn't much comparison.