1. Given that

$$x(n) = \{4, -3, 2, -1, -5, 3, -1, 0\}, \quad n = 0 \to 7$$

   (a) Using only 2-point FFTs, calculate the FFT of $x(n)$

   Heres the MATLAB script. Implemented for any size sequence $x(n)$ where N is a power of 2

```
%% 1 part a

% sequence x(n)
x = [4, -3, 2, -1, -5, 3, -1, 0];

% get N, constant WN
N = length(x);
WN = exp(-1i * 2 * pi / N);

% result of FFT is result_x
result_x = x;
% temporary array holds values, go from one 2-point FFT to another
hold_x = bitrevorder(result_x);

% array to hold exponents for WN
exp_array = zeros(1, length(x) / 2);
% number of sections for corresponding FFTs
blocks = N / 2;

% loop through each layer of fast DFT algorithm
for n = 0:log2(N) - 1

    % populate array for exponents of WN
    for k = 0:2^n:(N / 2) - 1
        exp_array(k + 1: k + 2^n) = 0:(N / 2^(n + 1)):(N / 2) - 1;
    end

    % a = index of exponents for WN
    a = 1;
    % other algorithm variables, will explain in a figure later
    idx = 1;
    for m = 1:blocks
        for idx2 = 0:(2^n) - 1
            index = idx + idx2;

            % do the appropriate 2-point FFT
            radix_res = fft([hold_x(index), (WN^exp_array(a)) * ...
                hold_x(index + 2^n)], 2);

            % store results into result_x
            result_x(index) = radix_res(1);
            result_x(index + 2^n) = radix_res(2);

            a = a + 1;
        end
        idx = idx + 2^(n + 1);
    end

    blocks = blocks / 2;
    hold_x = result_x;
end
```

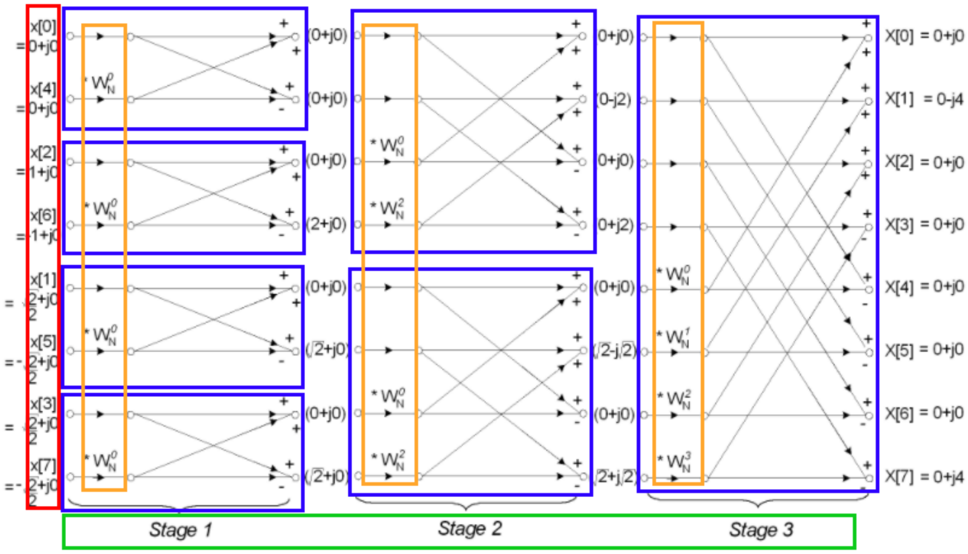


Figure 1: Annotated Radix-2 FFT diagram

Figure courtesy of
`https://riptutorial.com/algorithm/example/27088/radix-2-fft`.

In my algorithm shown above, the whole `for` loop is represented by the green sections, `blocks` is represented by the blue sections, the initial `bitrevorder(x)` is done on `x` to represent the red section, and the `exp_array` is represented by the orange sections. As for the indicies, the inner `for` loop loops through each block to appropriately compute the 2-point DFT.

Here is the resulting plot of `result_x` in MATLAB

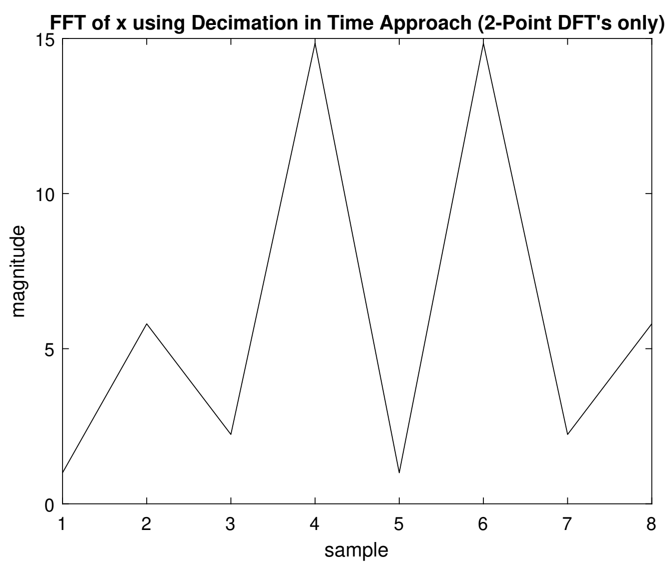


Figure 2: Radix-2 FFT magnitude result

(b) The standard 8-point DFT of $x(n)$ using the MATLAB routine `fft` looks as such

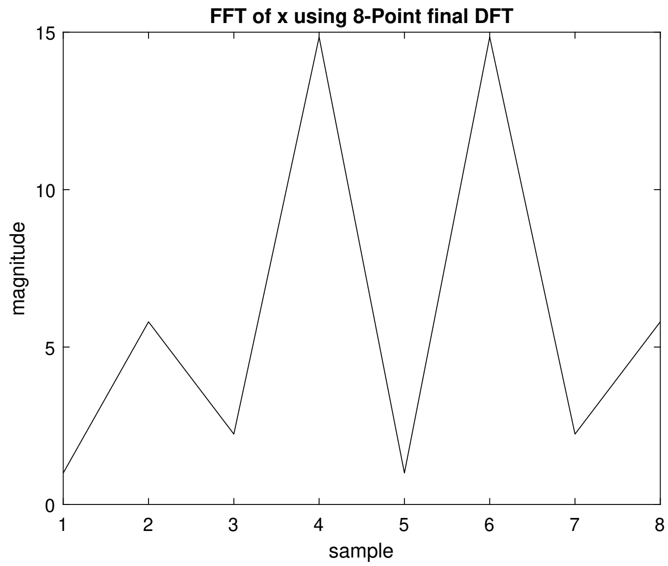


Figure 3: Standard 8-Point FFT magnitude result

(c) The results are identical, with an absolute difference on the magnitude of 10^{-15}

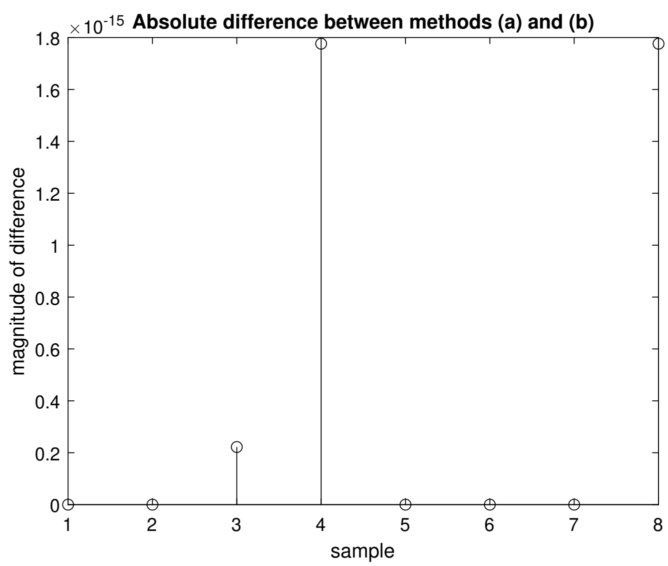


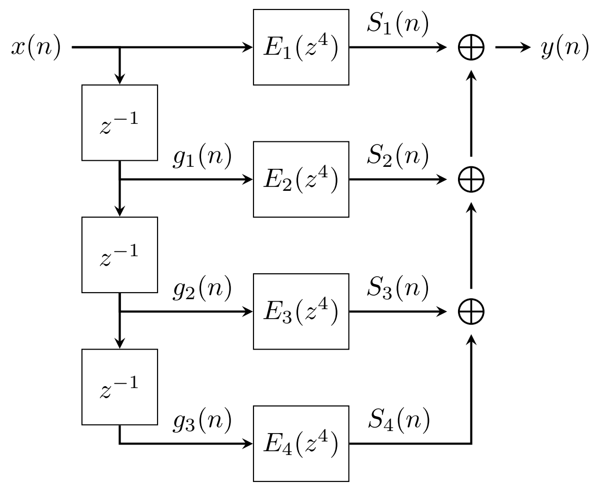
Figure 4: Absolute difference of results

2. (a) Get $H(z)$ in the form

$$H(z) = E_1(z^4) + z^{-1}E_2(z^4) + z^{-2}E_3(z^4) + z^{-3}E_4(z^4)$$

$$E_1(z) = 0.0168 + 0.0287z^{-1} + 0.2453z^{-2} - 0.0409z^{-3} \tag{1}$$
$$E_2(z) = 0.0264 - 0.1205z^{-1} - 0.1205z^{-2} + 0.0264z^{-3} \tag{2}$$
$$E_3(z) = -0.0409 + 0.2453z^{-1} + 0.0287z^{-2} + 0.0168z^{-3} \tag{3}$$
$$E_4(z) = 0.0334 + 0.6694z^{-1} + 0.0334z^{-2} \tag{4}$$

(b) Appropriate difference equation is given using the following diagram



$$g_1(n) = x(n-1) \tag{5}$$
$$g_2(n) = g_1(n-1) \tag{6}$$
$$g_3(n) = g_2(n-1) \tag{7}$$
$$S_1(n) = 0.0168x(n) + 0.0287x(n-4) + 0.2453x(n-8) - 0.0409x(n-12) \tag{8}$$
$$S_2(n) = 0.0264g_1(n) - 0.1205g_1(n-4) - 0.1205g_1(n-8) + 0.0264g_1(n-12) \tag{9}$$
$$S_3(n) = -0.0409g_2(n) + 0.2453g_2(n-4) + 0.0287g_2(n-8) + 0.0168g_2(n-12) \tag{10}$$
$$S_4(n) = 0.0334g_3(n) + 0.6694g_3(n-4) + 0.0334g_3(n-8) \tag{11}$$
$$y(n) = S_1(n) + S_2(n) + S_3(n) + S_4(n) \tag{12}$$

3. (a) The $2N$-point sequence $y(n)$ is defined as

$$y(n) = \begin{cases} x(\frac{n+1}{2}), & n \text{ odd} \\ 0, & n \text{ even} \end{cases}$$

where $X(k)$ is the N-point DFT of the sequence $x(n)$ for $n \in \{0, N-1\}$
The $2N$-point DFT of $y(n)$ is defined as

$$Y(k) = \sum_{n=0}^{2N-1} y(n) e^{\frac{-j2\pi nk}{2N}} \tag{13}$$

And since $\forall n_{\text{even}}$, $y(n) = 0$, the sequence $y(n)$ under the condition that n_{odd} summated equals the summated sequence $x(n)$. The only difference in the DFT would be the n in the exponential term, where it ranges from $n = 0 \to N-1$ $\forall n_{\text{odd}}$, meaning

$$\sum_{n=0}^{2N-1} y(n) e^{\frac{-j2\pi nk}{2N}} = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi(2n-1)k}{2N}} \tag{14}$$

where $2n-1$ represents the circular odd integers from $0 \to N-1$. Keep in mind, that $k \in \{0, 2N-1\}$, so the second equation will be circularly evaluated twice
So that

$$Y(k) = e^{\frac{j2\pi k}{2N}} \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}} = e^{\frac{j2\pi k}{2N}} X((k))_N, \quad k \in \{0, 2N-1\} \tag{15}$$

(b) Given

$$X_3[k] = \frac{1}{N} \sum_{l=0}^{N-1} X_1[l] X_2[((k-l))_N]$$

show that $x_3[n] = x_1[n]x_2[n]$
First, we know from sampling as well as the convolution theorem, that

$$N \sum_{r=-\infty}^{\infty} \delta[n - rN] \longleftrightarrow \sum_{k=-\infty}^{\infty} \delta(f - k/N) \tag{16}$$

so that

$$\frac{1}{N} \sum_{l=0}^{N-1} X_1[l] X_2[((k-l))_N] = \sum_{l=0}^{N-1} \sum_{n=0}^{N-1} \left(x_1[n] e^{\frac{-j2\pi nl}{N}} x_2[n] e^{\frac{-j2\pi n(k-l)}{N}} \right) \tag{17}$$

$$\sum_{n=0}^{N-1} x_3[n] e^{\frac{-j2\pi nk}{N}} = \sum_{l=0}^{N-1} \sum_{n=0}^{N-1} \left(x_1[n] x_2[n] e^{\frac{-j2\pi nl}{N}} e^{\frac{j2\pi nl}{N}} e^{\frac{-j2\pi nk}{N}} \right) \tag{18}$$

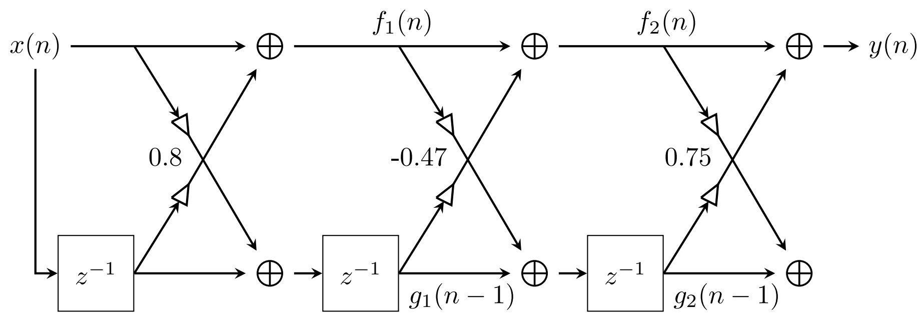
$$\sum_{n=0}^{N-1} x_3[n] e^{\frac{-j2\pi nk}{N}} = \sum_{n=0}^{N-1} x_1[n] x_2[n] e^{\frac{-j2\pi nk}{N}} \tag{19}$$

and then assuming element-wise equivalence, it is shown that

$$x_3[n] = x_1[n] x_2[n], \quad n = 0, 1, \ldots, N-1 \tag{20}$$

4. Consider an FIR lattice filter with coefficients $K_1 = 0.8$, $K_2 = -0.47$, $K_3 = 0.75$

(a) Draw the FIR lattice filter



(b) The difference equations are

$$y(n) = f_2(n) + 0.75g_2(n-1) \tag{21}$$

$$g_2(n) = g_1(n-1) - 0.47f_1(n) \tag{22}$$

$$f_2(n) = f_1(n) - 0.47g_1(n-1) \tag{23}$$

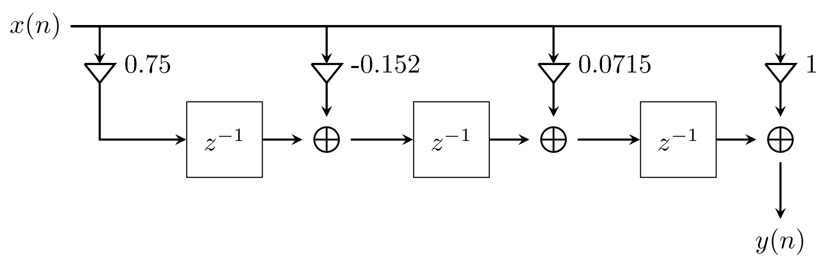
$$g_1(n) = x(n-1) + 0.8x(n) \tag{24}$$

$$f_1(n) = x(n) + 0.8x(n-1) \tag{25}$$

which gets reduced down to

$$y(n) = x(n) + 0.0715x(n-1) - 0.152x(n-2) + 0.75x(n-3) \tag{26}$$

(c) Draw the equivalent direct-form structure



5. Given

$$H(z) = 0.1325 - 0.0867z^{-1} + 0.4205z^{-2} + 1.3592z^{-3} + 0.4205z^{-4} - 0.0867z^{-5} + 0.1325z^{-6}$$

(a) We can split up the transfer function into multiple sections by knowing its roots. Since all roots are complex, the transfer function $H(z)$ can be split up into multiple 2nd-order sections

$$H(z) = \frac{1}{0.1325}S_1(z)S_2(z)S_3(z) \tag{27}$$

where

$$S_1(z) = z^2 - 1.0184z + 31.0874 \tag{28}$$

$$S_2(z) = z^2 + 0.3968z + 1 \tag{29}$$

$$S_3(z) = z^2 - 0.0328z + 0.0322 \tag{30}$$

Where we can take any sections $S_k(z)$ and $S_l(z)$ where $k \neq l$ to form either the first or second part of the cascaded filter. For this example, we will create the first filter F_1 out of $S_1(z)$ and $S_2(z)$ while the second filter F_2 will be $S_3(z)$. Using the algorithm in class, we find (by hand) that

$$F_1: \quad K_1 = 0.1984, \quad K_2 = 1, \quad K_3 = -0.0317, \quad K_4 = 31.0874 \tag{31}$$

$$F_2: \quad K_1 = -0.0317, \quad K_2 = 0.0322 \tag{32}$$

We can confirm this in MATLAB, where I wrote a function to do just this

```
function [Ks, gain] = fir_lattice(Hz)
% Author: Arpad Voros
% fir_lattice() routine takes in coefficients of a transfer function
% and determines the FIR Lattice coefficients
%   INPUT:      Hz - array of H(z) coefficients
%   OUTPUT:     Ks - coefficents of lattice, ordered from input to ...
    output cell array, if needs to be split up (special cases)
%               gain - the amount of gain for a given lattice

% initialize some variables
order = length(Hz) - 1;

% initialize outputs
Ks = {};
gain = {};

% check special case
if Hz(1) == Hz(order + 1)
    % include first gain
    if Hz(1) ≠ 1
        gain{end + 1} = Hz(1);
    end

    % split the roots
    rootsHz = roots(Hz);
    split = ceil(length(rootsHz) / 2);
    if rootsHz(split) == conj(rootsHz(split + 1))
        split = split + 1;
    end

    % recursive call
    [Ks{end + 1}, gain{end + 1}] = fir_lattice(poly(rootsHz(1:split)));
    [Ks{end + 1}, gain{end + 1}] = fir_lattice(poly(rootsHz(split + ...
        1:end)));
else
    % normalize, place into Az
    if Hz(1) ≠ 1
        gain{end + 1} = Hz(1);
        Az = Hz / Hz(1);
    else
        Az = Hz;
    end

    % temporary coefficients to be appended to Ks
    K = zeros(order, 1);

    % loop
    for n = order:-1:3
        % get last coefficient
```

```
        K(n) = Az(n + 1);

        % reverse coefficients
        Bz = flip(Az);

        % get new value of Az
        Az = (Az - K(n)*Bz)/(1 - K(n)^2);
        Az = Az(1:end - 1);
    end
    % final two coefficients
    K(2) = Az(3);
    K(1) = Az(2) / (1 + K(2));

    % append to rest of coefficients
    Ks{end + 1} = K;
end

% delete empty cells
gain = gain(¬cellfun('isempty', gain));
end
```

So by calling `fir_lattice` in the script below, we get

```
% coefficients of transfer function
h = [0.1325, -0.0867, 4.205, 1.3592, 4.205, -0.0867, 0.1325];

% display gain and K coefficients
[result, gain] = fir_lattice(h);
celldisp(gain);
celldisp(result);
```

```
gain{1} =       result{1}{1} =      result{2}{1} =

    0.1325          0.1984             -0.0317
                    1.0000              0.0322
                   -0.0317
                   31.0874
```

(b) Superscripts in this section do not indicate exponentiation, but rather help distinguish between F_1 and F_2, where f_k^1 and g_k^1 correspond to F_1 while f_k^2 and g_k^2 correspond to F_2. The appropriate difference equations for the cascaded filter above are

$$y(n) = f_1^2(n) + 0.0322 g_1^2(n-1) \tag{33}$$

$$g_1^2(n) = f_4^1(n-1) - 0.0317 f_4^1(n) \tag{34}$$
$$f_1^2(n) = f_4^1(n) - 0.0317 f_4^1(n-1) \tag{35}$$

$$f_4^1(n) = f_3^1(n) + 31.0874 g_3^1(n-1) \tag{36}$$
$$g_3^1(n) = g_2^1(n-1) - 0.0317 f_2^1(n) \tag{37}$$

$$f_3^1(n) = f_2^1(n) - 0.0317 g_2^1(n-1) \tag{38}$$

$$g_2^1(n) = g_1^1(n-1) + f_1^1(n) \tag{39}$$

$$f_2^1(n) = f_1^1(n) + g_1^1(n-1) \tag{40}$$

$$g_1^1(n) = x(n-1) + 0.1984x(n) \tag{41}$$

$$f_1^1(n) = x(n) + 0.1984x(n-1) \tag{42}$$