Cooley-Tukey FFT Algorithms

- Consider a length-N sequence $x[n]$ with an N-point DFT $X[k]$ where $N = N_1 N_2$
- Represent the indices n and k as

$$n = N_2 n_1 + n_2, \quad \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

$$k = k_1 + N_1 k_2, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

Cooley-Tukey FFT Algorithms

- Using these index mappings we can write

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

as

$$X[k] = X[k_1 + N_1 k_2]$$

$$= \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_N^{(N_2 n_1 + n_2)(k_1 + N_1 k_2)}$$

$$= \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_N^{N_2 n_1 k_1} W_N^{n_2 k_1} W_N^{N_1 n_2 k_2} W_N^{N_1 N_2 n_1 k_2}$$

Cooley-Tukey FFT Algorithms

- Since $W_N^{N_2 n_1 k_1} = W_{N_1}^{n_1 k_1}$, $W_N^{N_1 n_2 k_2} = W_{N_2}^{n_2 k_2}$, and $W_N^{N_1 N_2 n_1 k_2} = 1$, we have

$$X[k_1 + N_1 k_2]$$

$$= \sum_{n_2=0}^{N_2-1} \left[\left(\sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_{N_1}^{n_1 k_1} \right) W_N^{n_2 k_1} \right] W_{N_2}^{n_2 k_2}$$

where $0 \le k_1 \le N_1 - 1$ and $0 \le k_2 \le N_2 - 1$

Cooley-Tukey FFT Algorithms

- The effect of the index mapping is to map the 1-D sequence $x[n]$ into a 2-D sequence that can be represented as a 2-D array with n_1 specifying the rows and n_2 specifying the columns of the array
- Inner parentheses of the last equation is seen to be the set of N_1-point DFTs of the N_2-columns:

$$G[k_1, n_2] = \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_{N_1}^{n_1 k_1}, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

Cooley-Tukey FFT Algorithms

- Note: The column DFTs can be done in place
- Next, these row DFTs are multiplied in place by the twiddle factors $W_N^{n_2 k_1}$ yielding

$$\tilde{G}[k_1, n_2] = W_N^{n_2 k_1} G[k_1, n_2], \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

- Finally, the outer sum is the set of N_2-point DFTs of the columns of the array:

$$X[k_1 + N_1 k_2] = \sum_{n_2 = 0}^{N_2 - 1} \tilde{G}[k_1, n_2] W_{N_2}^{n_2 k_2}, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

Cooley-Tukey FFT Algorithms

- The row DFTs, $X[k_1 + N_1 k_2]$, can again be computed in place
- The input $x[n]$ is entered into an array according to the index map:

$$n = N_2 n_1 + n_2, \quad \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

- Likewise, the output DFT samples $X[k]$ need to extracted from the array according to the index map:

$$k = k_1 + N_1 k_2, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

Cooley-Tukey FFT Algorithms

- Example - Let $N = 8$. Choose $N_1 = 2$ and $N_2 = 4$
- Then

$$X[k_1 + 2k_2] = \sum_{n_2=0}^{3} \left[\left(\sum_{n_1=0}^{1} x[4n_1 + n_2] W_2^{k_1 n_1} \right) W_8^{k_1 n_2} \right] W_4^{k_2 n_2}$$

for $0 \le k_1 \le 1$ and $0 \le k_2 \le 3$

Cooley-Tukey FFT Algorithms

- 2-D array representation of the input is

$n_1 \backslash n_2$	0	1	2	3
0	x[0]	x[1]	x[2]	x[3]
1	x[4]	x[5]	x[6]	x[7]

- The column DFTs are 2-point DFTs given by

$$G[k_1,n_2] = x[n_2] + (-1)^{k_1} x[4+n_2], \quad \begin{cases} 0 \le k_1 \le 1 \\ 0 \le n_2 \le 3 \end{cases}$$

- These DFTs require no multiplications

Cooley-Tukey FFT Algorithms

- 2-D array of row transforms is

$k_1 \backslash n_2$	0	1	2	3
0	$G[0,0]$	$G[0,1]$	$G[0,2]$	$G[0,3]$
1	$G[1,0]$	$G[1,1]$	$G[1,2]$	$G[1,3]$

- After multiplying by the twiddle factors $W_8^{n_2 k_1}$ array becomes

$k_1 \backslash n_2$	0	1	2	3
0	$\tilde{G}[0,0]$	$\tilde{G}[0,1]$	$\tilde{G}[0,2]$	$\tilde{G}[0,3]$
1	$\tilde{G}[1,0]$	$\tilde{G}[1,1]$	$\tilde{G}[1,2]$	$\tilde{G}[1,3]$

Cooley-Tukey FFT Algorithms

- Note: $\tilde{G}[k_1, n_2] = W_8^{n_2 k_1} G[k_1, n_2]$
- Finally, the 4-point DFTs of the rows are computed:

$$X[k_1 + 2k_2] = \sum_{n_2=0}^{3} \tilde{G}[k_1, n_2] W_4^{n_2 k_2}, \quad \begin{cases} 0 \le k_1 \le 1 \\ 0 \le k_2 \le 3 \end{cases}$$

- Output 2-D array is given by

$k_1 \backslash k_2$	0	1	2	3
0	X[0]	X[2]	X[4]	X[6]
1	X[1]	X[3]	X[5]	X[7]

Cooley-Tukey FFT Algorithms

- The process illustrated is precisely the first stage of the DIF FFT algorithm
- By choosing $N_1 = 4$ and $N_2 = 2$, we get the first stage of the DIT FFT algorithm
- Alternate index mappings are given by

$$n = n_1 + N_1 n_2, \quad \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

$$k = N_2 k_1 + k_2, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

Prime Factor Algorithms

- Twiddle factors can be eliminated by defining the index mappings as

$$n = \langle An_1 + Bn_2 \rangle_N, \quad \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

$$k = \langle Ck_1 + Dk_2 \rangle_N, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

- To eliminate the twiddle factors we need to express

$$W_N^{(An_1+Bn_2)(Ck_1+Dk_2)} = W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2}$$

Prime Factor Algorithms

- Now $W_N^{(An_1+Bn_2)(Ck_1+Dk_2)}$

$$= W_N^{ACn_1k_1} W_N^{ADn_1k_2} W_N^{BCn_2k_1} W_N^{BDn_2k_2}$$

- It follows from above that if

$$\langle AC \rangle_N = N_2,\ \langle BD \rangle_N = N_1,$$

$$\langle AD \rangle_N = \langle BC \rangle_N = 0$$

then

$$W_N^{(An_1+Bn_2)(Ck_1+Dk_2)} = W_{N_1}^{n_1k_1} W_{N_2}^{n_2k_2}$$

Prime Factor Algorithms

- One set of coefficients that eliminates the twiddle factors is given by

$$A = N_2,\ B = N_1,$$
$$C = N_2\langle N_2^{-1}\rangle_{N_1},\quad D = N_1\langle N_1^{-1}\rangle_{N_2}$$

- Here $\langle N_1^{-1}\rangle_{N_2}$ denotes the **multiplicative inverse** of N_1 reduced modulo N_2
- $\Rightarrow$ If $\langle N_1^{-1}\rangle_{N_2} = \alpha$ then $\langle N_1\alpha\rangle_{N_2} = 1$ or, in other words $N_1\alpha = N_2\beta + 1$ where β is any integer

Prime Factor Algorithms

- For example, if $N_1 = 4$ and $N_2 = 3$, then
  $$\langle 3^{-1} \rangle_4 = 3 \quad \text{since} \quad \langle 3 \cdot 3 \rangle_4 = 1$$
- Likewise, if $\langle N_2^{-1} \rangle_{N_1} = \gamma$, then $N_2\gamma = N_1\delta + 1$ where δ is any integer
- Now, $\langle AC \rangle_N = \langle N_2 \cdot (N_2 \langle N_2^{-1} \rangle_{N_1}) \rangle_N$
  $$= \langle N_2(N_1\delta + 1) \rangle_N = \langle N_2 N_1 \delta + N_2 \rangle_N = N_2$$
- Similarly, $\langle BD \rangle_N = \langle N_1 \cdot (N_1 \langle N_1^{-1} \rangle_{N_2}) \rangle_N$
  $$= \langle N_1(N_2\beta + 1) \rangle_N = \langle N_1 N_2 \beta + N_1 \rangle_N = N_1$$

Prime Factor Algorithms

- Next,

$$\langle AD\rangle_N = \langle N_2 \cdot (N_1 \langle N_1^{-1}\rangle_{N_2})\rangle_N = \langle N\alpha\rangle_N = 0$$

- Likewise,

$$\langle BC\rangle_N = \langle N_1 \cdot (N_2 \langle N_2^{-1}\rangle_{N_1})\rangle_N = \langle N\gamma\rangle_N = 0$$

- Hence,

$$X[k] = X[\langle C_1 k + D k_2\rangle_N]$$

$$= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[\langle A n_1 + B n_2\rangle_N] W_N^{N_2 n_1 k_1} W_N^{N_1 n_2 k_2}$$

Prime Factor Algorithms

- Thus, $X[\langle Ck_1 + Dk_2 \rangle_N]$

$$= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[\langle An_1 + Bn_2 \rangle_N] W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2}$$

$$= \sum_{n_2=0}^{N_2-1} G[n_2, k_1] W_{N_2}^{n_2 k_2}$$

where

$$G[n_2, k_1] = \sum_{n_1=0}^{N_1-1} x[\langle An_1 + Bn_2 \rangle_N] W_{N_1}^{n_1 k_1}$$

and $0 \le k_1 \le N_1 - 1, 0 \le k_2 \le N_2 - 1$

Prime Factor Algorithms

- Example - Let $N = 12$. Choose $N_1 = 4$ and $N_2 = 3$
- Then, $A = 3$, $B = 4$, $C = 3\langle 3^{-1}\rangle_4 = 9$ and $D = 4\langle 4^{-1}\rangle_3 = 4$
- The index mappings are

$$n = \langle 3n_1 + 4n_2\rangle_{12}, \begin{cases} 0 \le n_1 \le 3 \\ 0 \le n_2 \le 2 \end{cases}$$

$$k = \langle 9k_1 + 4k_2\rangle_{12}, \begin{cases} 0 \le k_1 \le 3 \\ 0 \le k_2 \le 2 \end{cases}$$

Prime Factor Algorithms

- 2-D array representation of input is

$n_1 \backslash n_2$	0	1	2
0	$x[0]$	$x[4]$	$x[8]$
1	$x[3]$	$x[7]$	$x[11]$
2	$x[6]$	$x[10]$	$x[2]$
3	$x[9]$	$x[1]$	$x[5]$

- 4-point transforms of the columns lead to

$k_1 \backslash n_2$	0	1	2
0	$G[0,0]$	$G[0,1]$	$G[0,2]$
1	$G[1,0]$	$G[1,1]$	$G[1,2]$
2	$G[2,0]$	$G[2,1]$	$G[2,2]$
3	$G[3,0]$	$G[3,1]$	$G[3,2]$

Prime Factor Algorithms

- Final DFT array is

$k_1 \backslash k_2$	0	1	2
0	$X[0]$	$X[4]$	$X[8]$
1	$X[9]$	$X[1]$	$X[5]$
2	$X[6]$	$X[10]$	$X[2]$
3	$X[3]$	$X[7]$	$X[11]$

- 4-point DFTs require no multiplications, whereas the 3-point DFTs require 4 complex multiplications
- Thus, the algorithm requires 16 complex multiplications

Chirp z-Transform Algorithm

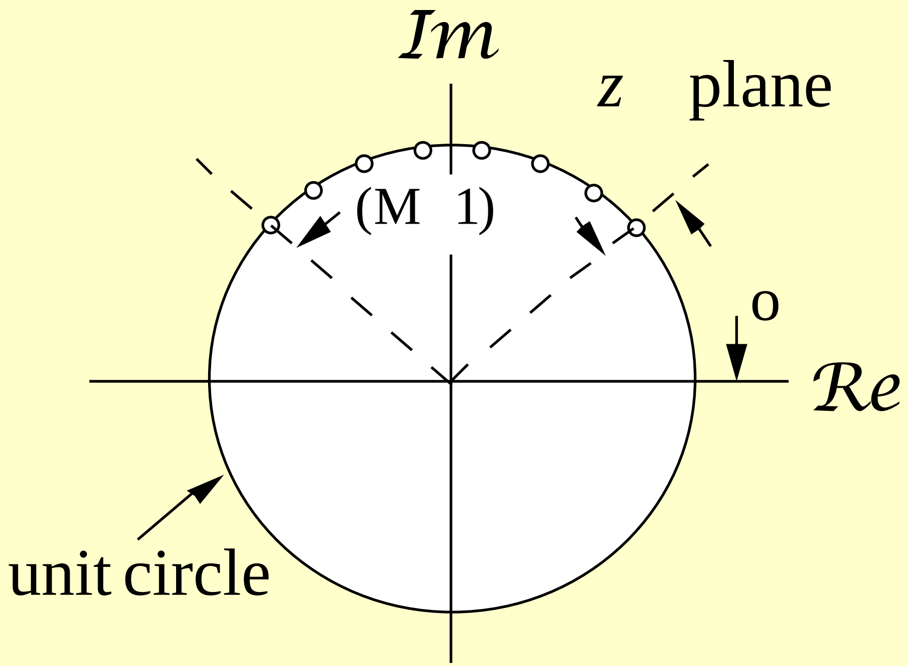
- Let $x[n]$ be a length-N sequence with a Fourier transform
- We consider evaluation of M samples of that are equally spaced in angle on the unit circle at frequencies

$$\omega_k = \omega_o + k\Delta\omega, \;\; 0 \le k \le M-1$$

where the starting frequency ω_o and the frequency increment $\Delta\omega$ can be chosen arbitrarily

Chirp z-Transform Algorithm

- Figure below illustrates the problem



Chirp z-Transform Algorithm

- The problem is thus to evaluate

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_k n}, \quad 0 \le k \le M-1$$

or, with W defined as

$$W = e^{-j\Delta\omega}$$

to evaluate

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_o n}W^{nk}$$

Chirp z-Transform Algorithm

- Using the identity $nk = \frac{1}{2}[n^2 + k^2 - (k-n)^2]$

  we can write

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n] e^{j\omega_o n} W^{n^2/2} W^{k^2/2} W^{-(k-n)^2/2}$$

- Letting $g[n] = x[n] e^{-j\omega_o n} W^{n^2/2}$

  we arrive at

$$X(e^{j\omega_k}) = W^{k^2/2} \left(\sum_{n=0}^{N-1} g[n] W^{-(k-n)^2/2} \right),$$

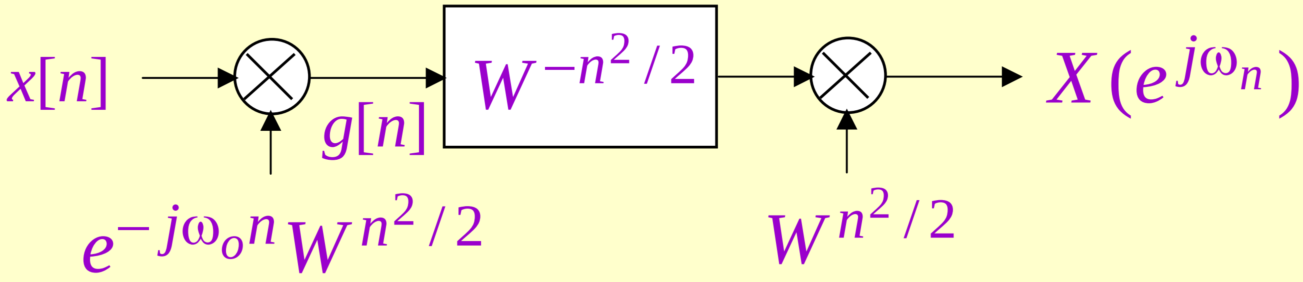
$$0 \le k \le M-1$$

Chirp z-Transform Algorithm

- Interchanging k and n we get

$$X(e^{j\omega_n}) = W^{n^2/2}\left(\sum_{k=0}^{N-1} g[k] W^{-(n-k)^2/2}\right),$$

- Thus, $X(e^{j\omega_n})$ corresponds to the convolution of the sequence $g[n]$ with the sequence $W^{-n^2/2}$ followed by multiplication by the sequence $W^{n^2/2}$ as indicated below



Chirp *z*-Transform Algorithm

- The sequence $W^{-n^2/2}$ can be thought of as a complex exponential sequence with linearly increasing frequency
- Such signals, in radar systems, are called chirp signals
- Hence, the name **chirp transform**

Chirp z-Transform Algorithm

- For the evaluation of

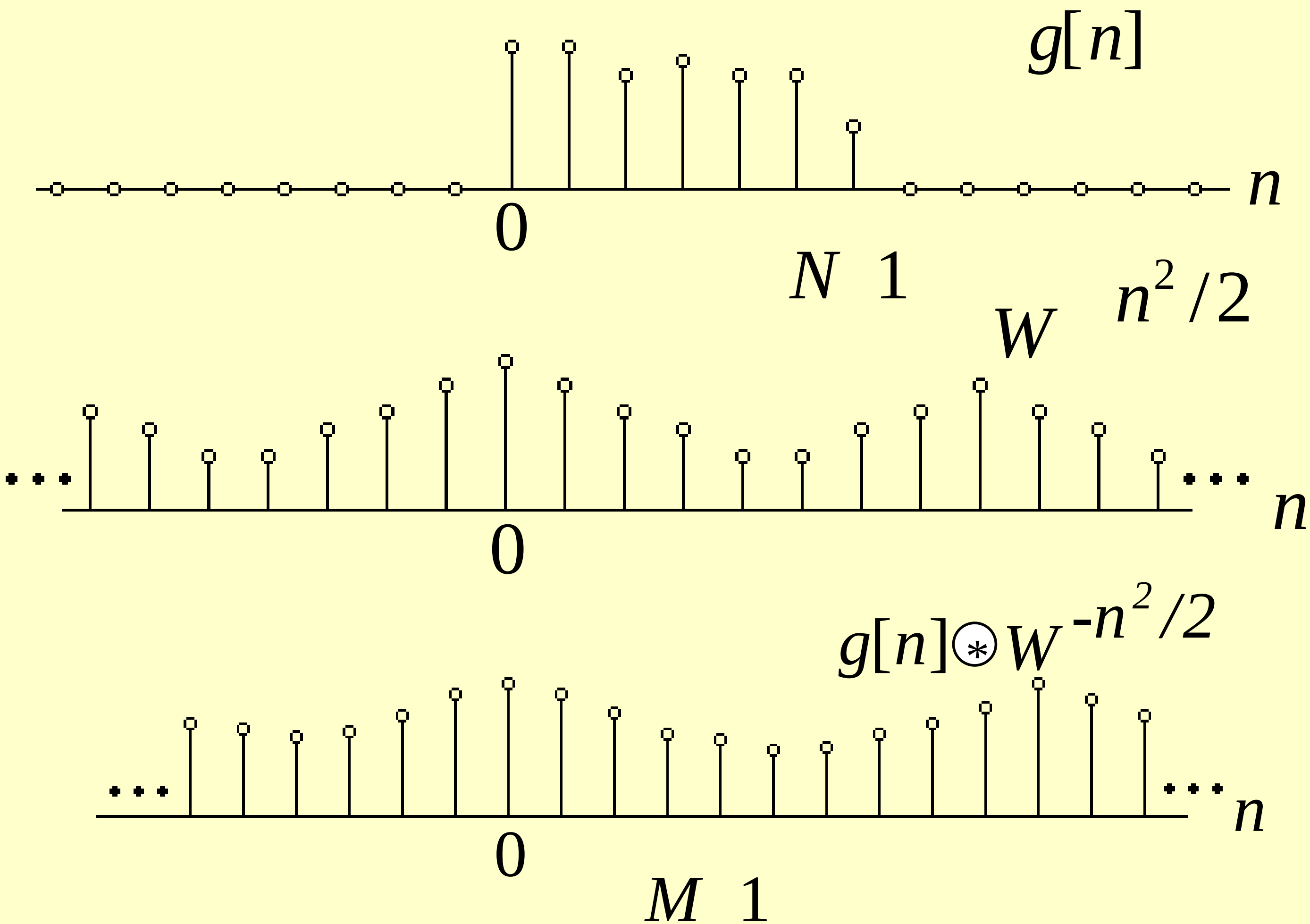
$$X(e^{j\omega_n}) = W^{n^2/2}\left(\sum_{k=0}^{N-1} g[k] W^{-(n-k)^2/2}\right),$$

  the output of the system depicted earlier need to be computed over a finite interval

- Since $g[n]$ is a length-N sequence, only a finite portion of the infinite length sequence $W^{-n^2/2}$ is used in obtaining the convolution sum over the interval $0 \le n \le M-1$

Chirp z-Transform Algorithm

- Typical signals

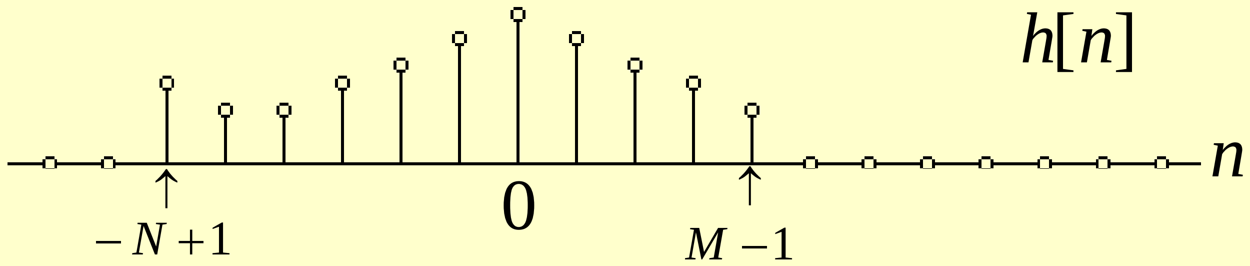


Chirp z-Transform Algorithm

- The portion of the sequence $W^{-n^2/2}$ used in obtaining the convolution sum is from the interval $-N+1 \le n \le M-1$

- Let $h[n] = \begin{cases} W^{-n^2/2}, & -(N-1) \le n \le (M-1) \\ 0, & \text{otherwise} \end{cases}$

  as shown below

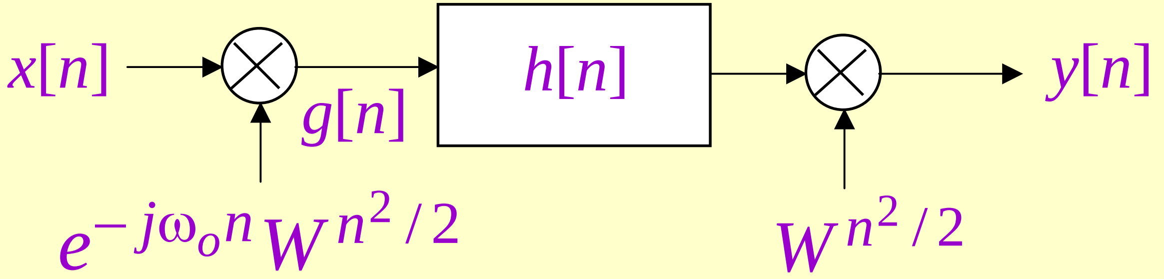


Chirp z-Transform Algorithm

- It can be seen that

$$g[n] \circledast W^{-n^2/2} = g[n] \circledast h[n],\ 0 \le n \le M-1$$

- Hence, the computation of the frequency samples $X(e^{j\omega_n})$ can be carried out using an FIR filter as indicated below



where $y[n] = X(e^{j\omega_n}),\ 0 \le n \le M-1$

Chirp *z*-Transform Algorithm

- Advantages -
- (1) $N = M$ is not required as in FFT algorithms
- (2) Neither N nor M do not have to be composite numbers
- (3) Parameters ω_o and $\Delta\omega$ are arbitrary
- (4) Convolution with $h[n]$ can be implemented using FFT techniques

Chirp z-Transform Algorithm