Understanding the FFT. We'll do this in 3 steps a) motivation, b) write DFT as linear transform, c) discuss decimation-in-frequency

a) motivation: computation

DFT: $X[k] = \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{kn}{N}}$

each point takes $N$ complex multiplications, $(N-1)$ complex adds

thus for $N$ points, $O(N^2)$ computations.

FFT: if $N$ is factor of 2, can do in $O(N \log_2 N)$ computations

example: $N = 128$, so $\log_2 N = 8$
16x speedup!
$N = 1024 \rightarrow$ 102.4x speedup!!

question: is FFT always faster?

b) write DFT as a linear transformation

define $W_N = e^{-j2\pi/N}$

then DFT is:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad k=0,1,..N-1$$

$$x(n) = \sum_{k=0}^{N-1} X(k) W_N^{-kn} \qquad n=0,1,..N-1$$

If we define

$$x_N = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} \qquad X_N = \begin{bmatrix} X(0) \\ \vdots \\ X(N) \end{bmatrix}$$

$$\underline{W}_N = \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots W_N^{(N-1)} \\ \vdots & W_N^2 & W_N^4 & \cdots W_N^{2(N-1)} \\ \vdots & & & \\ W_N^0 & W_N^{N-1} & W_N^{(2)(N-1)} & \cdots W_N^{(N-1)(N-1)} \end{bmatrix}$$

$k$

← Note lots of symmetry! FFT will exploit

then

$$X_N = \underline{W}_N x_N$$

i.e. we can write DFT as a matrix-vector product.

and

$$x_N = \frac{1}{N} \underline{W}_N^* X_N$$

this means $x_N = \frac{1}{N} W_N^* W_N x_n \Rightarrow W_N^* W_N = N I$

or $W_N^* = N W_N^{-1}$

← useful property

example: $N=4$ DFT

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & W_4^4 & W_4^6 \\ 1 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

If we have some length-4 vector $x(n)$, we can find

$$X(\cancel{k}) = W_4 x$$

interpretation: consider $x_N = \frac{1}{N} W_N^* X_N$

and let $X_N$ have just a single freq: eg $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ for $N=4$

selects out 3rd column:

$x(n)$ is expanded in terms of basis function

$$x = X_N(0) W_N^*(:,0) + X_N(1) W_N^*(:,1) + \cdots$$

①

FFT

definitions:

1) DFT

$$\begin{cases} X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2\pi}{N}kn} \equiv \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \\ x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(k)\, e^{j\frac{2\pi}{N}kn} \equiv \frac{1}{N}\sum_{n=0}^{N-1} X(k)\, W_N^{-kn} \end{cases}$$

where $W_N = e^{-j2\pi/N}$

note: inverse DFT can use same method as fwd, just adjust sign of W exponent

2) periodicity N-periodic $W_N^{kn} = W_N^{k(N+n)} = W_N^{(k+N)n}$

---

Van Veen goes thru decimation in time, gets to eqn

$$X(k) = \underbrace{X_e(k)}_{N/2 \text{ DFT of even}} + W_N^k \underbrace{X_o(k)}_{N/2 \text{ DFT of odd}}, \quad k = 0, 1, \ldots N-1$$

---

let's look at this. does it really match DFT definition?

Say, $N=8$, then $X(k) = X_e(k) + W_8^k X_o(k)$

$k=0$: $X(0) = \sum_{r=0}^{3} x(2r)\, W_4^{0r} + W_8^0 \sum_{r=0}^{3} x(2r+1)\, W_4^{0r}$

$k=0$ means $\omega=0$ means DC!

but, $W_4^{0r} = \left(e^{-j\frac{2\pi}{4}r}\right)^0 = 1$

So, $X(0) = \sum_{r=0}^{3} x(2r) + (1)\sum_{r=0}^{3} x(2r+1)$

$= \text{sum of all } x(n)\text{'s}$

matches DFT definition //

harder: $k=7$

$$X(7) = X_e(7) + W_8^7 X_o(7)$$

?? $X_e, X_o$ 4 long, but repeating

$$= \sum_{r=0}^{3} x(2r)\, W_4^{r7} + W_8^7 \sum_{r=0}^{3} x(2r+1)\, W_4^{r7}$$

periodicity: $W_4^{7r} = W_4^{(4+3)r} = W_4^{(3)r}$

So, $X(7) = X_e(3) + W_8^7 X_o(3)$

---

just for fun and out of order: $k=1$

$$X(1) = X_e(1) + W_8^1 X_o(1)$$

$$= \sum_{r=0}^{3} x(2r)\, e^{-j\frac{2\pi}{4} r(1)} + W_8^1\, e^{-j\frac{2\pi}{8}(1)} \sum_{r=0}^{3} x(2r+1)\, e^{-j\left(\frac{2\pi}{4}\right)r}$$

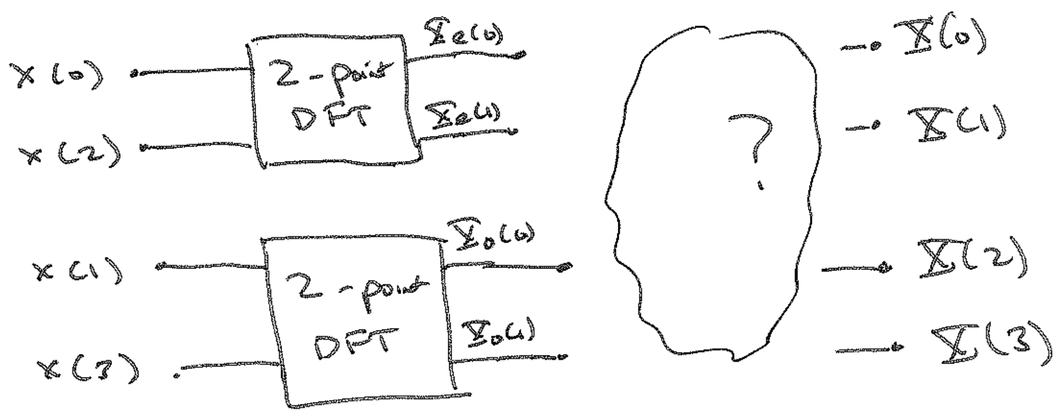
prove to yourself this $= \sum_{n=0}^{7} x(n)\, e^{-j\frac{2\pi}{8}n}$

---

<u>Apply the relation</u>: first split, $N=4$

$$X(k) = X_e(k) + W_4^k X_o(k)$$

fill out this ~~this~~ figure




↗ group even, odd

↑ note bit ~~reverse~~ reversed order

| decimal | 2 bit | bit reverse | decimal |
|---|---|---|---|
| 0 | 00 | 00 | 0 |
| 1 | 01 | 10 | 2 |
| 2 | 10 | 01 | 1 |
| 3 | 11 | 11 | 3 |

plug & chug

$$X(0) = X_e(0) + W_4^0 X_o(0)$$

$$X(1) = X_e(1) + W_4^1 X_o(1)$$

$$X(2) = X_e(2) + W_4^2 X_o(2) = X_e(0) + W_4^2 X_o(0)$$

$$X(3) = X_e(3) + W_4^3 X_o(3) = X_e(1) + W_4^3 X_o(1)$$

use periodicity

Q) what are these factors?

$$W_4^0 = \left(e^{-j\frac{2\pi}{4}}\right)^0 = 1$$

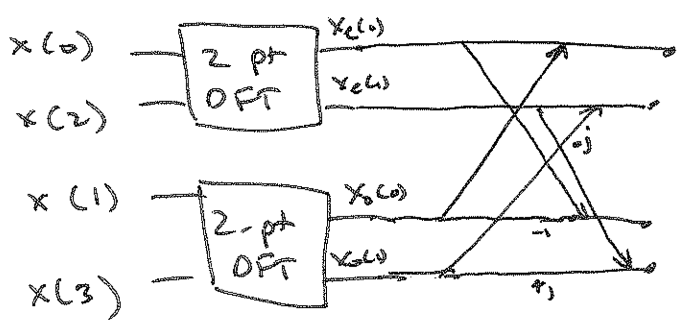
$$W_4^1 = \left(e^{-j2\pi/4}\right)^1 = e^{-j\pi/2} = -j$$

$$W_4^2 = \left(e^{-j2\pi/4}\right)^2 = e^{-j\pi} = -1$$

$$W_4^3 = \left(e^{-j2\pi/4}\right)^3 = e^{-j6\pi/4} = e^{j\pi/2} = j$$

simple!

so picture is




$X(0) = X_e(0) + X_o(0)$

$X(1) = X_e(1) - j X_o(1)$

$X(2) = X_e(0) - X_o(0)$

$X(3) = X_e(1) + j X_o(1)$

now, look at the 2-point DFT

back to DFT definition

$$X(k) = \sum_{n=0}^{1} x(n) W_2^{nk} = \sum_{n=0}^{1} x(n) e^{-j\frac{2\pi}{2}nk}, \quad k=0,1$$

$$X(0) = x(0) + x(1)$$

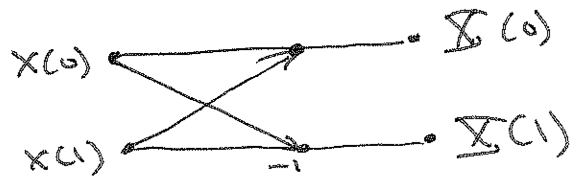
$$X(1) = x(0) e^{-j\pi(1)(0)} + e^{-j\pi(1)(1)} x(1) = x(0) e^{0} + x(1) e^{-j\pi}$$

$$= x(0) - x(1)$$

so 2-pt DFT is just sum, difference!

we could redraw the last figure (N=4) w/ 2 layers, but it's a lot of work

"Butterfly"



## Decimation in frequency algorithm

Say we have a 1024-point signal. → need 1024 pt DFT

→ what if we want just even DFT samples (in frequency)
- undersampling in freq ⇒ time aliasing
- we'll intentionally time-alias, do 512-pt transform to get even samples

→ what if we just want odd samples?
- multiply sequence by complex exponential, to shift in frequency domain
- time alias, then do 512 pt transform

this gives all points ~~for~~
→ we can keep subdividing by 2 until we get to a 2-point transform.

even samples (assuming $N$ = power of 2)

$$X(2r) = \sum_{n=0}^{N-1} x(n) W_N^{n(2r)} \qquad r = 0, \dots N/2 - 1$$

$$= \sum_{n=0}^{N/2-1} x(n) W_N^{2rn} + \sum_{n=N/2}^{N-1} x(n) W_N^{2rn}$$

(1) now, change variables: $m = n - N/2 \Rightarrow n = m + N/2$

2nd term goes to: $\sum_{m=0}^{N/2-1} x[m+N/2] W_N^{2rm} W_N^{2r(N/2)}$

$W^{rN} = e^{\frac{j2\pi rN}{N}} = 1$

so,

(3) $$X(2r) = \sum_{n=0}^{N/2-1} \left(x(n) + x(n+N/2)\right) W_{N/2}^{rn}$$

rename $m$ to $n$

(2) look at

$$W_N^{2rn} = e^{-j\frac{2\pi}{N}(2rn)}$$

$$= e^{-j\frac{2\pi}{N/2} rn} = W_{N/2}^{rn}$$

instead of doing $N$-point DFT & looking at alternate points do $N/2$ & look at all

## Odd frequency samples

$$\underline{X}(2r+1) = \sum_{n=0}^{N-1} x(n)\, W_N^{n(2r+1)} = \cancel{\sum_{n=0}^{N-1} \left(x(n) W_N^{n}\right) W_N^{n(2r+1)}}$$

again, split in two terms

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)\, \underbrace{W_N^{n2r}}_{\text{same as before}}\, \underbrace{W_N^{n}}_{\text{new}} + \sum_{n=N/2}^{N-1} x(n)\, W_N^{n2r}\, W_n^{n}$$

do change of variables again; for second term.
we get something like

$$\sum_{m=0}^{N/2-1} x(m+N/2) \underbrace{\left(W_N^{2rm}\, W_N^{m}\right)}_{\text{same as 1st term}} \overset{1}{\cancel{W_N^{\frac{N}{2}2r}}}\ \overset{-1}{\cancel{W_N^{N/2}}}$$

so finally,

$$\underline{X}(2r+1) = \sum_{n=0}^{N/2-1} \underbrace{\left(x(n) - x(n+N/2)\right)}_{\text{another form of time-alias}} \underbrace{W_N^{n}}_{\text{modulation "twiddle factor"}}\, W_{N/2}^{rn}$$

# Other transforms

## Discrete - Cosine transform - PorM 7.5

If ~~we~~ $x(n)$ is real & even, $X(k)$ is real & even and the exponential term in the DFT is replaced by a cosine.

→ steps: take a signal & make it symmetric 7.5.1

→ If go through the math, we find the transform pair:

$$C(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right)$$

$$x(n) = \sum_{k=0}^{N-1} \alpha(k)\, C(k) \cos\left(\frac{\pi(2n+1)k}{2N}\right)$$

The DCT is worse than the DFT at representing sinusoids (Fig 7.5.2) but better at representing other signals (7.5.3).

In general, DCT is good for image compression & is basis for JPEG

## Chirp-Z transform (PorM 8.3.2)

lets us compute the transform on places other than the unit circle. Also provides an alternate DFT calculation method that is well suited to special hardware

## Goertzel algorithm PorM 8.3.1

Re-write eqns so we have a parallel bank of filters, each of which gets one frequency in the DFT.

Advantage: we don't need to calculate every frequency.

## Performance

| | Mults | | | Adds | | |
|---|---|---|---|---|---|---|
| N | r-2 | r-4 | SR | r2 | r4 | sr |
| 16 | 24 | 20 | 20 | 152 | 148 | 148 |
| 64 | 264 | 208 | 196 | 1032 | 976 | 964 |
| 1024 | 10 248 | 7856 | 7172 | 30,728 | 28 336 | 27652 |

## Linear Filtering Approach

FFT: good if want all $X(k)$.

What if want $< \log N$ samples (applications such as detection)?

$\Rightarrow$ New algorithms

Goertzel: DFT as a linear filtering operation

$$X(k) = \sum_m x(m) W_N^{-km}$$

but $W_N^{-kN} = 1$

So

$$X(k) = W_N^{-kN} \sum_m x(m) W_N^{-km}$$

$$= \sum_m x(m) W_N^{-k(N-m)}$$

Looks like a convolution

$$y_k(n) = \sum_{m=0}^{N-1} x(m) W_N^{-k(n-m)}$$

$$y_k(n) = x(n) * h_k(n) \qquad h_k(n) = W_N^{-kn} u(n)$$

So

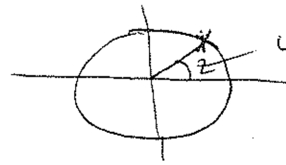
$$y_k(n)\Big|_{n=N} = y_k(N) = X(k) = \text{value of DFT @ } \omega = \frac{2\pi k}{N}$$

$$W_N^{-kn} = \left(e^{-j\frac{2\pi}{N}}\right)^{(-kn)} = \left(e^{j\omega_k}\right)^n$$

~~Idea~~ ~~each~~

So what is this system $\quad W_N^{-k} = \left(e^{-j\frac{2\pi}{N}}\right)^{-k} = e^{+j\frac{2\pi k}{N}}$ 10.16 $= e^{+j\omega_k}$

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}}$$



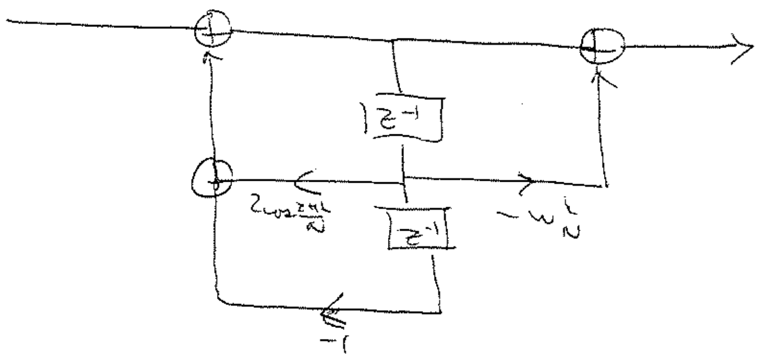

$\omega_k = \frac{2\pi k}{N}$

Resonant filter tuned to the correct frequency
Makes a lot of sense. Idea: 1 tuned circuit per desired frequency
⇒ bank of filters *
IIR system ⇒ difference equation implem

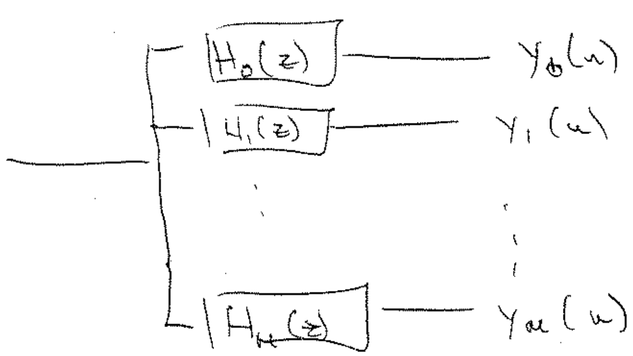
$$y_k(n) = W_N^{-k} y_k(n-1) + x(n) \qquad y(-1) = 0$$

Efficiency: merge complex poles $w_k$ and $w_k^*$
to get

$$H_k(z) = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}$$




Recurse for
$n = 0, 1, 2, \ldots, N$
at time $N$ get
$X(k)$ and $X(N-k)$ by
symmetry




$$y_k(N) = X(k)$$

Complexity: $N+1$ mults / value of $X(k)$ and $X(N-k)$
⇒ good for $M < \log_2 N$