Chapter 8 Notes

Thursday, December 1, 2022 11:36 AM

- efficient computation of the DFT → fast fourier transform
- 2 primary approaches to efficiently computing the DFT
  - Divide-and-conquer: when computing a N-point DFT (where N is a composite number), the DFT is reduced to smaller DFTs that combine to the larger DFT (lends itself to a recursive implementation)
    - very useful when N is a power of 2 (binary tree)?
  - based on the formulation of the DFT as a filtering problem
    → Goertzel algorithm and chirp-z transform algorithm

Section 8.1

- FFT Algorithms
- Fundamental problem:
  - need to compute a N point array, X(k), based on N-point x(n) according to the following formula:

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

where $W_N^{kn} = e^{-2\pi kn/N}$

  - for each k value, this is N multiplications and N-1 additions
  - meaning, overall there are $N^2 - N$ operations
- Divide and Conquer overview:
  - N-point DFT is divided into successively smaller DFTs
  - assumption: N is not prime since we can just pad with 0s
  - decompose N: $N = L \cdot M$
    - so x(n) can either be expressed as a 1D N-point array or a 2D LxM matrix
    indices: $0 \le l \le L-1$ and $0 \le m \le M-1$
    - mapping (l, m) to n

$$n = Ml + m \quad \text{or} \quad n = l + mL$$

  - the computed DFT can be expressed in a similar manner:
    - mapping k to (p, q)

$$k = Mp + q \quad \text{or} \quad k = qL + p$$

  - adopting a column wise mapping for x(n) and row-wise mapping for X(k), the DFT becomes:

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(m,l) W_N^{(Mp+q)(mL+l)}$$

$$(Mp+q)(mL+l) = MpmL + Mpl + qmL + ql$$

$$W_N^{(Mp+q)(mL+l)} = W_N^{MpmL} W_N^{Mpl} W_N^{qmL} W_N^{ql}$$

$$W_N^{MpmL} = W_N^{Npm} = e^{-2\pi Npm/N} = e^{-2\pi pm} = 1$$

$$W_N^{mqL} = W_{N/L}^{mq} = W_M^{mq} \quad \text{and} \quad W_N^{Mpl} = W_{N/M}^{pl} = W_L^{pl}$$

thus:

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(m,l) W_M^{mq} W_L^{pl} W_N^{ql}$$

$$= \sum_{l=0}^{L-1} \left(W_N^{ql} \cdot \sum_{m=0}^{M-1} \left(x(m,l) W_M^{mq} \right) W_L^{pl} \right)$$

Question for John:

- is column vs. row adoption of DFT arbitrary?

Answer:

- is column vs. row adoption of DFT arbitrary?

Answer:

- signal is column-wise since transform is applied on the left
- computation steps:
  - 1. compute the M point DFTs
    → $F(l,q) = \sum_{m=0}^{M-1} x(l,m) W_M^{mq}$ $0 \le q \le M-1$ for every row (l)
  - 2. apply W_N^{lq} to $F(l,q)$
    → $G(l,q) = W_N^{lq} F(l,q)$ $0 \le l \le L-1$
  - 3. compute the L point DFTs $0 \le q \le M-1$
    → $X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp}$
- Complexity reduction:
  - step 1:
    - multiplication: NM
    - addition: $N(M-1)$
  - step 2:
    - multiplication: N
    - addition: 0
  - step 3:
    - multiplication: NL
    - addition: $N(L-1)$
  - total:
    - multiplication: $N(M+L+1)$
    - addition: $N(M+L-2)$
  - as compared to N^2 multiplications and N^2-N additions
- this process can be repeated over and over (assuming N is not prime) to further reduce complexity
- Summary of process:
  1. store the signal column-wise (L x M)
  2. compute M point DFT of each row
  3. Multiply resulting array by phase factors W_N^{lq} (rotation)
  4. Compute L point DFT of each column
  5. read resulting array row-wise
  → $X(p,q) = \sum_{m=0}^{M-1} W_M^{mq} \left[\sum_{l=0}^{L-1} x(l,m) W_L^{lp} \right] W_N^{mp}$
- Radix-2 FFT Algorithms
  - radix are how you can decompose N into $r_1 \cdot r_2 \cdot r_3 \cdots r_v = N$
  - interesting when $r_1 = r_2 = \cdots = r_v \equiv r$
    thus $N = r^v$
  - → $M = \frac{N}{2}$, $L = 2$, splitting x into two N/2 point data sequences $f_1(n)$ and $f_2(n)$ corresponding to even-numbered and odd-numbered samples of x
    → $f_1(n) = x(2n)$ $n = 0, 1, \cdots, \frac{N}{2} - 1$
    $f_2(n) = x(2n+1)$

and odd-numbered samples

$\rightarrow f_1(n) = x(2n) \quad n = 0, 1, \ldots, \frac{N}{2} - 1$

$f_2(n) = x(2n+1)$

- this makes the DFT:

$$X(k) = \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn} = \sum_{m=0}^{\frac{N}{2}-1} x(2m) W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} x(2m+1) W_N^{k(2m+1)}$$

$$= \sum_{m=0}^{\frac{N}{2}-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_2(m) W_{N/2}^{km}$$

$$= F_1(k) + W_N^k F_2(k) \quad k = 0, 1, \ldots, N-1$$

- the decomposition can be further applied to even further reduce the computational complexity
  - each $\frac{N}{2}$ point DFT split into 2 $\frac{N}{4}$-point DFT
  - this can be done $\log_2 N$ times until you are computing 1-pt DFTs
  - important note: input data shuffling
    - for 8-point sequence:
      - first decimation: $x(0), x(2), x(4), x(6), x(1), x(3), x(5), x(7)$
      - second decimation: $x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)$
- another radix-2 FFT Algorithm → decimation in frequency
  - instead of decimating (changing order) of the time domain, the frequency domain is decimated