1/19/2015

<u>Fast Fourier Transform</u>

DFT $\quad f(x) = \sum_{u=0}^{N-1} F(u) e^{j2\pi ux/N} = \sum_{u=0}^{N-1} F(u) W_N^{-ux}$

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi ux/N} = \frac{1}{N} \sum_{x=0}^{N-1} f(x) W_N^{ux}$$

Twiddle factor: $W_N = e^{-j2\pi/N}$

Computational cost: $O(N^2)$ $\qquad$ N mult. for N vars.

<u>Sidebar: W_N^{ux} rewrites</u>

$$W_N^2 = e^{-j2\pi 2/N} = e^{-j2\pi/(N/2)} = W_{N/2}^1$$

$$W_N^{u+N} = e^{-j2\pi(u+N)/N} = W_N^u e^{-j2\pi} = W_N^u$$

$$W_N^{u+N/2} = e^{-j2\pi(u+N/2)/N} = W_N^u e^{-j\pi} = -W_N^u$$

$$W_N^{u(2x+1)} = e^{-j2\pi u(2x+1)/N} = W_N^{2ux} W_N^u = W_{N/2}^{ux} W_N^u$$

$$W_N^{u(x+N/2)} = W_N^{ux} W_N^{uN/2} = W_N^{ux} e^{-j\pi u} = (-1)^u W_N^{ux}$$

<u>Assumption</u>

$N = 2^K$ (power of two) or $N = 2M$

Decimation-in-Time

$$F(u) = \frac{1}{N} \sum_{x\ \text{even}} f(x)\, W_N^{ux} + \frac{1}{N} \sum_{x\ \text{odd}} f(x)\, W_N^{ux}$$

$$= \frac{1}{2}\left[\frac{1}{M} \sum_{x=0}^{M-1} f(2x)\, W_N^{2ux} + \frac{1}{M} \sum_{x=0}^{M-1} f(2x+1)\, W_N^{u(2x+1)} \right]$$

$$= \frac{1}{2}\left[\underbrace{\frac{1}{M} \sum_{x=0}^{M-1} f(2x)\, W_M^{ux}}_{F_{even}(u)} + \underbrace{\frac{1}{M} \sum_{x=0}^{M-1} f(2x+1)\, W_M^{ux} W_N^{u}}_{F_{odd}(u)} \right]$$

$\Downarrow$

$$F(u) = \frac{1}{2}\left[F_{even}(u) + F_{odd}(u)\, W_N^{u} \right]$$

$$F(u+M) = \frac{1}{2}\left[F_{even}(u) + F_{odd}(u)\, W_N^{u+M} \right]$$

$\Downarrow$

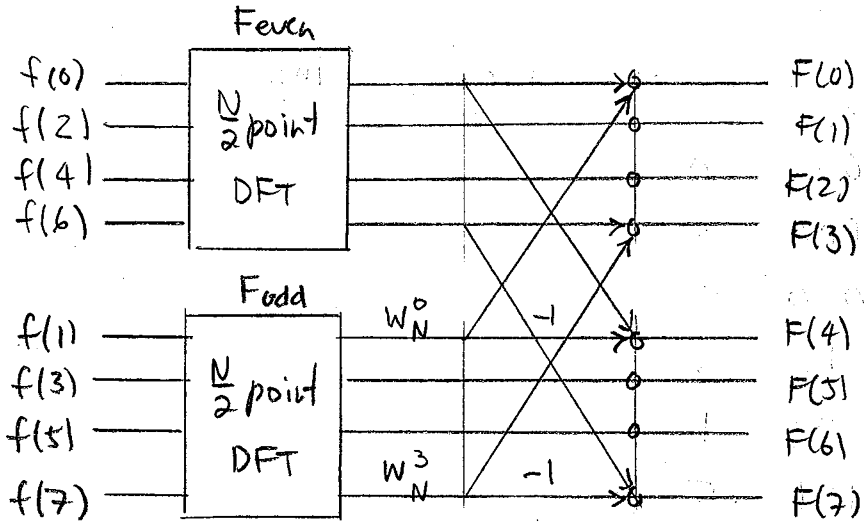
$u = 0, 1, \ldots, M-1$

$$F(u) = \frac{1}{2}\left[F_{even}(u) + F_{odd}(u)\, W_N^{u} \right]$$

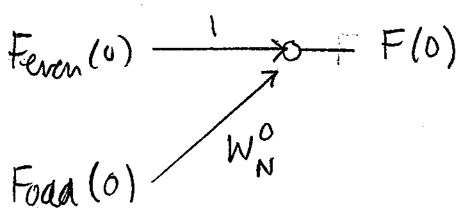
$$F(u+M) = \frac{1}{2}\left[F_{even}(u) - F_{odd}(u)\, W_N^{u} \right]$$

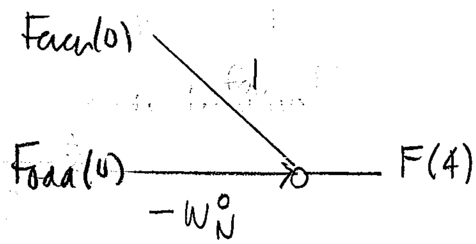
Above follows from $W_M^{(u+M)x} = W_M^{ux}\, W_M^{Mx} = W_M^{ux}$

$$W_N^{u+M} = -W_N^{u}$$



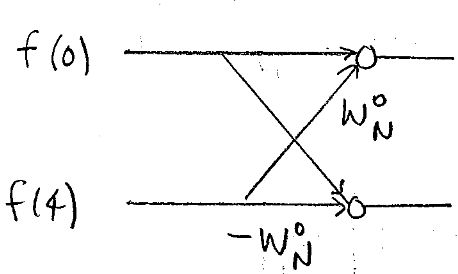
etc for all other output



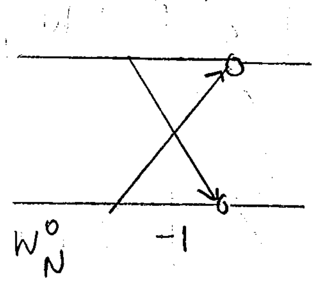


etc for F(1) - F(3), F(5) - F(7)

Apply same rewrite to obtain $\frac{N}{4}$ point impl. of $\frac{N}{2}$ point DFT
Continue until 2-point DFT reached



$\equiv$



Butterfly computation

Recursion depth: log N

Computational cost : O(N logN)

2 * num. butterflies * stages

Bit reversal

Even/odd splitting corr. to reordering of input data

0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7
input order		output order	

Decimation-in-frequency

$$F(u) = \frac{1}{N}\left[\sum_{x=0}^{M-1} f(x) W_N^{ux} + \sum_{x=M}^{N-1} f(x) W_N^{ux}\right]$$

$$= \frac{1}{2M}\left[\sum_{x=0}^{M-1} f(x) W_N^{ux} + \sum_{x=0}^{M-1} f(x+M) W_N^{u(x+M)}\right]$$

$$= \frac{1}{2M}\sum_{x=0}^{M-1}\left[f(x) + (-1)^u f(x+M)\right] W_N^{ux}$$

$\Downarrow$

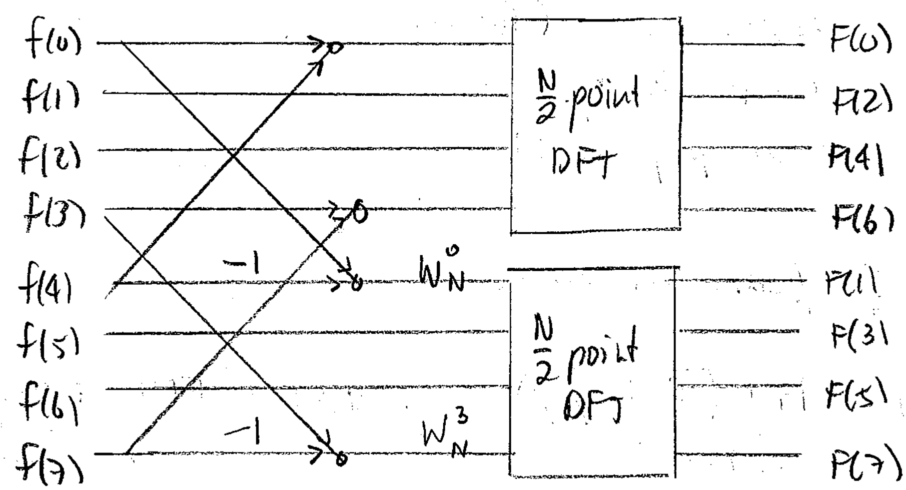
$$\underset{\text{even}}{F(2u)} = \frac{1}{2M} \sum_{x=0}^{M-1} \left[f(x) + (-1)^{2u} f(x+M) \right] W_N^{2ux}$$

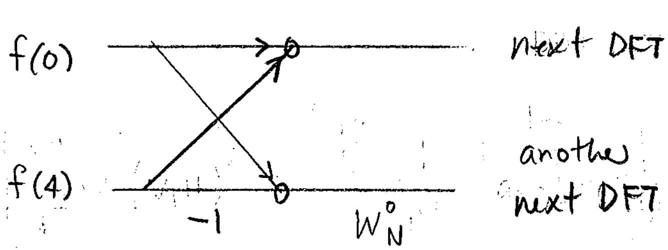
$$= \frac{1}{2M} \sum_{x=0}^{M-1} \left[f(x) + f(x+M) \right] W_M^{ux}$$

$$\underset{\text{odd}}{F(2u+1)} = \frac{1}{2M} \sum_{x=0}^{M-1} \left[f(x) + (-1)^{(2u+1)} f(x+M) \right] W_N^{(2u+1)x}$$

$$= \frac{1}{2M} \sum_{x=0}^{M-1} \left[f(x) - f(x+M) \right] W_M^{ux} \, W_N^{x}$$

etc for all other input





Inverse FFT

$$f(x) = \sum_{u=0}^{N-1} F(u)\, W_N^{-ux}$$

$\Downarrow$

$$\frac{1}{N} f^*(x) = \frac{1}{N}\left[\sum_{u=0}^{N-1} F(u)\, W_N^{-ux}\right]^*$$

$$= \frac{1}{N} \sum_{u=0}^{N-1} F^*(u)\, W_N^{ux}$$

That is, conjugate F( )
apply FFT (forward)
conjugate result
multiply by N

2D FFT (same as 2D DFT)

$F(u,y) = FFT(f(x,y))$ rows, $F(u,v) = FFT(F(u,x))$ columns

2D inverse FFT

$$\frac{1}{N_1 N_2} f^*(x,y) = \frac{1}{N_2} \sum_{v=0}^{N_2-1} \left[\frac{1}{N_1} \sum_{u=0}^{N_1-1} F^*(u,v)\, W_{N_1}^{ux}\right] W_{N_2}^{vy}$$

Note: Conjugate $F(u,v)$ but not $F(x,v)$

From Oppenheim & Schafer

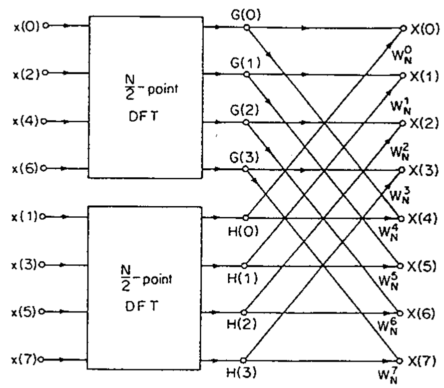


Fig. 6.3 Flow graph of the decimation-in-time decomposition of an N-point DFT computation into two $N/2$-point DFT computations ($N = 8$).

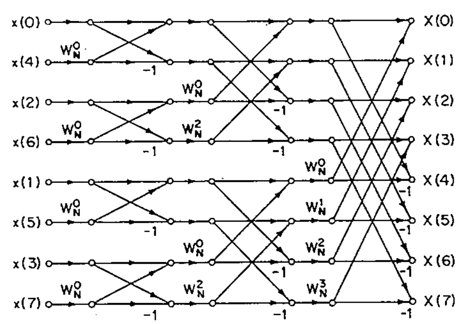


Fig. 6.10 Flow graph of eight-point DFT using the butterfly computation of Fig. 6.9.

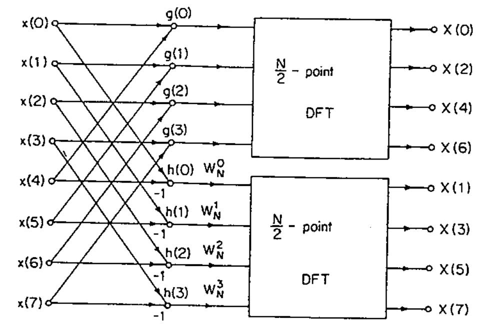


Fig. 6.15 Flow graph of the decimation-in-frequency decomposition of an N-point DFT computation into two $N/2$-point DFT computations ($N = 8$).

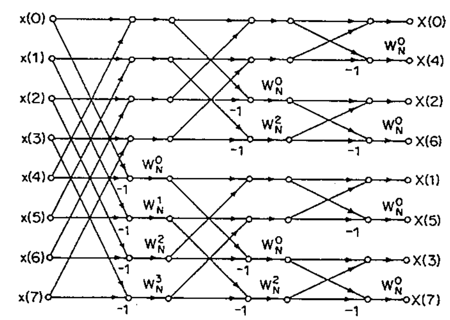


Fig. 6.18 Flow graph of complete decimation-in-frequency decomposition of an eight-point DFT computation.

```
#include ...
using namespace std;

const double pi = 4.0*atan(1.0);
typedef complex<double> comdbl;
enum direction { FORWARD=-1, INVERSE=+1 };

void fft(direction dirsign, vector<comdbl> &z)
{
  int i, j, m, N = (int)z.size();

  // bitreversal of input

  for(i=0, j=0; i<N-1; i++) {
    if (i < j)
      swap(z[i], z[j]);

    m = N;
    do {
      m /= 2;
      j ^= m;
    } while ((j & m) == 0);
  }

  comdbl w;    // twiddle factor
  double t;    // w angle
  comdbl ztmp; // z update

  // butterfly computations

  for (j=1; j<N; j*=2) {
    for (m=0; m<j; m++) {
      t = dirsign*(pi*m/j);
      w = comdbl(cos(t), sin(t));

      for(i=m; i<N; i+=2*j) {
        ztmp = w * z[i+j];
        z[i+j] = z[i] - ztmp;
        z[i] = z[i] + ztmp;
      }
    }
  }

  // forward fft scaling

  if (dirsign == FORWARD) {
    for (i=0; i<N; i++)
    z[i] /= (double)N;
  }
}
```

```
ostream & operator<<(ostream &out, const comdbl &z)
{
  out.setf(ios::fixed);
  out.precision(4);

  out << setw(8) << right << z.real()
      << " "
      << setw(8) << right << z.imag();

  return out;
}

void print(vector<comdbl> &z)
{
  int N = (int)z.size();
  for (int i=0; i<N; i++)
    cout << z[i] << "\n";
  cout << "\n";
}

void createsignal(vector<comdbl> &z)
{
  double t, u0=1.0;

  int N = (int)z.size();
  for (int i=0; i<N; i++) {
    t = (2.0*pi*u0)*((double)i/N);
    z[i] = comdbl(cos(t), 0.0);
  }
}

int main(int argc, char *argv[])
{
  int N=8;
  vector<comdbl> z(N);

  createsignal(z); print(z);
  fft(FORWARD, z); print(z);
  fft(INVERSE, z); print(z);
}
```

B Jähne (1994)
Digital Imag. Proc.

Vector-Matrix Interpretation

Vectors

$$f = \begin{bmatrix} f_0 \\ f_1 \\ \\ f_{N-1} \end{bmatrix} \qquad F = \begin{bmatrix} F_0 \\ F_1 \\ \\ F_{N-1} \end{bmatrix} \qquad b_u = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ W_N^{u} \\ \\ W_N^{(N-1)u} \end{bmatrix}$$

real   complex

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

Complex inner product yields

$$F_u = \langle f, b_u \rangle = \sum_{x=0}^{N-1} f_x \, b_u^{*} = \sum_{x=0}^{N-1} f_x \, e^{-j2\pi ux/N}$$

Orthogonality holds

$$\langle b_u, b_v \rangle = \frac{1}{N} \sum_{k=0}^{N-1} W_N^{ku} W_N^{-kv}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} e^{j2\pi ku/N} e^{-j2\pi kv/N}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} e^{j2\pi k(u-v)/N}$$

$$= \begin{cases} 1 & \text{if } u=v \\ 0 & \text{otherwise} \end{cases}$$

shown at beginning of class

Set $\{b_u\}$ forms orthogonal vector basis that allows other vectors to be expressed as linear comb. thereof

Matrix $B = [b_0 \; b_1 \; \dots \; b_{N-1}]$ leads to DFT :

$$F = Bf \,, \qquad f = B^{*T} F = B^H F$$

row inner prod. — complex conj Column inner prod. — Hermitian transpose (Matlab)

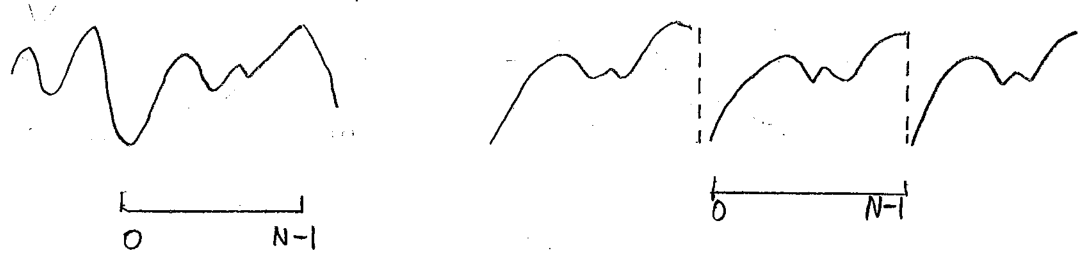
Note $B^{-1} = B^{*T}$ which implies DFT is unitary transf.

<u>WARNING</u> Aliasing / Moire effects

Must sample at 2× Nyquist (max signal freq) to avoid aliasing (folding of higher freq. into lower portion of spectrum)

Problems may arise when cropping out signal from larger data set due to implied periodicity for DFT processing

(Ex



Discontinuities introduces high freq.

(Ex

Show dot patterns

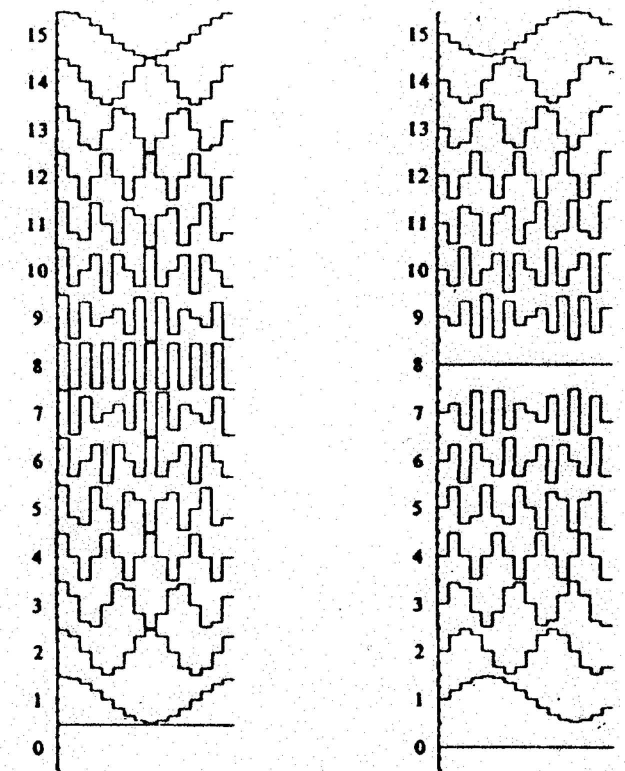


Figure 3.1: Basis functions of the DFT for $M = 16$; real part (cosine function) left, imaginary part (sine function) right.

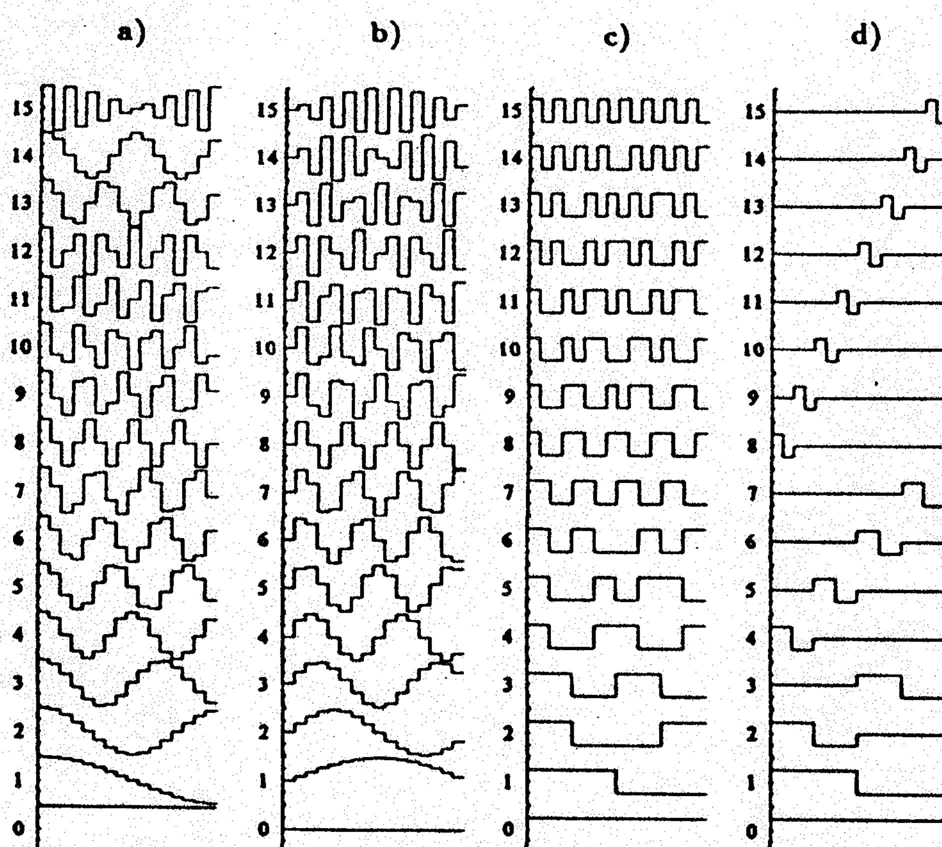


Figure 3.7: Basis functions of one-dimensional unitary transforms for $M = 16$-dimensional vectors: a) cosine transform; b) sine transform; c) Hadamard transform; d) Haar transform.