Chương 3: Biến đổi Fourier nhanh (FFT)

Xử lý tín hiệu số

Tín Vũ

tinvu1309@gmail.com

Mục lục

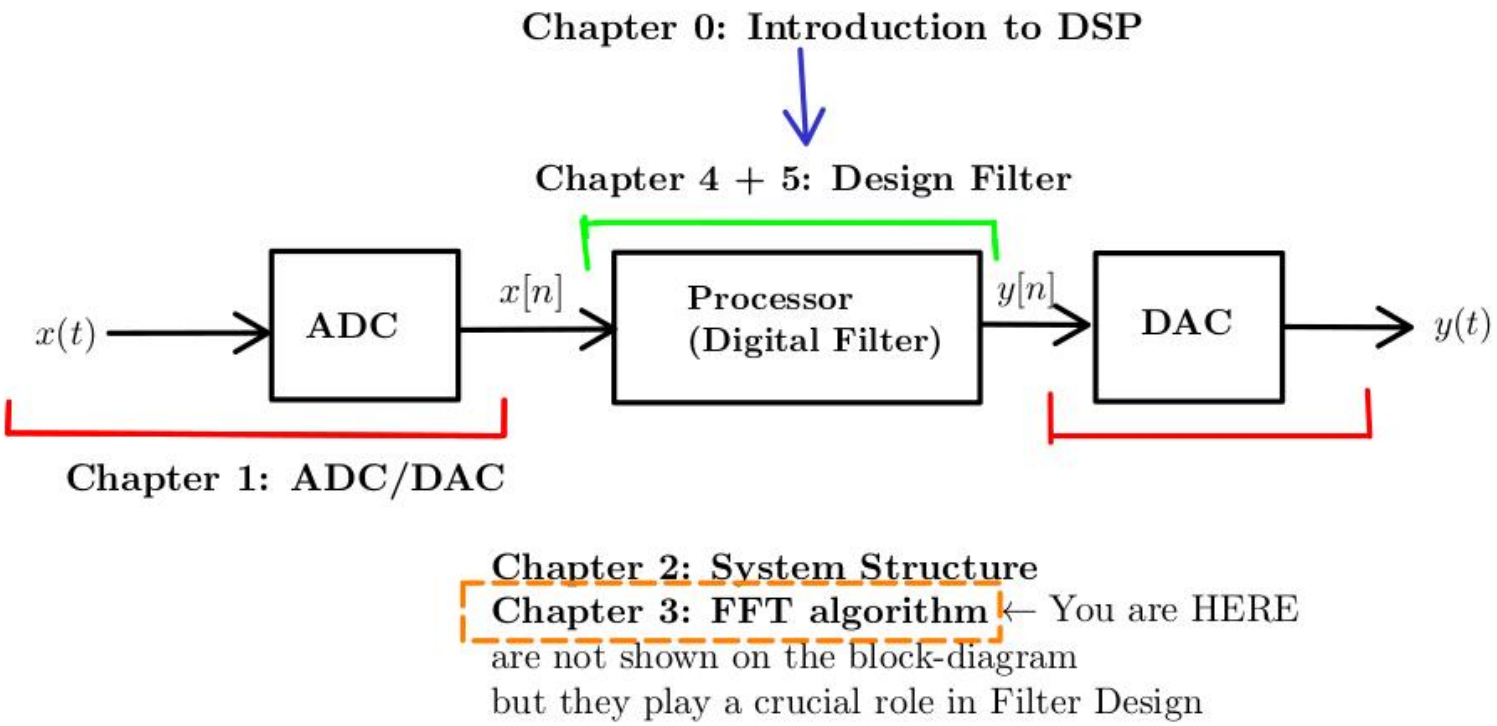
Giới thiệu playlist

- Mình là Tín Vũ, hiện đang là sinh viên học tại Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội. Mình tạo playlist video này để hỗ trợ các bạn học môn **Xử lý tín hiệu số**.
- Khác với môn học tiên quyết Tín hiệu hệ thống trước đó, bài giảng môn học này **hoàn toàn bám sát với đề cương và giáo trình nội bộ** của trường mình, nên các bạn trường khác cần phải lưu ý rất kĩ điều này.
- Không chỉ dừng lại ở lý thuyết, playlist này **có bổ sung hướng dẫn lập trình cơ bản bằng GNU Octave/Matlab** để vẽ phổ tín hiệu, đáp ứng tần số và thiết kế bộ lọc.
- Môn học này bao gồm **6 chương**, các chương đều liên quan rất chặt chẽ với nhau nên hãy học cẩn thận ngay từ Chương 0 để ôn thi cuối kì đỡ vất vả.

Tài liệu tham khảo

- Tài liệu tham khảo chính: Giáo trình Xử lý tín hiệu số (Nguyễn Linh Trung, Trần Đức Tân, Huỳnh Hữu Tuệ, ĐHCN, 2012).
- Tài liệu tham khảo phụ: Discrete-time Signal Processing (Alan V.Oppenheim, 2nd edition).

Quy trình xử lý tín hiệu số



Hình: DSP Learning Process

Biến đổi Fourier rời rạc (DFT)

Trong học phần Tín hiệu hệ thống trước đó, ta đã biết 2 cặp công thức gồm chuỗi Fourier thời gian rời rạc (DTFS) và biến đổi Fourier thời gian rời rạc (DTFT):

1 DTFS pairs:

$$x[n] = \sum_{k=<N_0>} c_k e^{jk\omega_0 n}$$

$$c_k = \frac{1}{N_0} \sum_{n=<N_0>} x[n] e^{-jk\omega_0 n}$$

2 DTFT pairs:

$$x[n] = \frac{1}{2\pi} \int_{2\pi} X(\omega) e^{j\omega n} d\omega$$

$$X(\omega) = \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n}$$

Ta thấy DTFS chỉ có thể được sử dụng để phân tích **tín hiệu rời rạc tuần hoàn**, còn DTFT chỉ được dùng khi phân tích **tín hiệu rời rạc không tuần hoàn vô hạn**. Không chỉ có vậy, để phân tích bằng DTFS hay DTFT, tín hiệu đầu vào **phải là một hàm số theo biến thời gian** n.

Hiển nhiên ta thấy rằng trong thực tiễn, không có dạng tín hiệu hoàn hảo nào thỏa mãn đủ điều kiện để phân tích DTFS hay DTFT. Các tín hiệu trong thực tiễn thường ở dạng ngẫu nhiên (không thể mô hình hóa bằng một hàm số), không tuần hoàn, độ dài hữu hạn (có khi độ dài cực kì ngắn).

Trong chương này, ta sẽ phát triển công thức và thuật toán để xác định phổ của tín hiệu trong thực tế.

Biến đổi Fourier rời rạc (DFT)

Dựa vào công thức DTFS và DTFT, các nhà toán học đã phát triển cặp công thức DFT (discrete Fourier transform pair) như sau:

$$x[n] = \frac{1}{N_0} \sum_{k=0}^{N_0-1} X[k] e^{jk\omega_0 n} = \frac{1}{N_0} \sum_{k=0}^{N_0-1} X[k] e^{jk\frac{2\pi}{N_0} n}$$

$$X[k] = \sum_{n=0}^{N_0-1} x[n] e^{-jk\omega_0 n} = \sum_{n=0}^{N_0-1} x[n] e^{-jk\frac{2\pi}{N_0} n}$$

Chúng ta phải phân biệt rất rạch ròi bản chất toán học của các công thức DTFS, DTFT và DFT, tuyệt đối không được nhầm lẫn chúng với nhau.
Nếu ta kí hiệu:

$$W_{N_0} = e^{-j\frac{2\pi}{N_0}}$$

Ta có thể viết lại các công thức DFT như sau:

$$x[n] = \frac{1}{N_0} \sum_{k=0}^{N_0-1} X[k] W_{N_0}^{-kn}$$

$$X[k] = \sum_{n=0}^{N_0-1} x[n] W_{N_0}^{kn}$$

Biến đổi Fourier rời rạc (DFT)

Để cho dễ nhìn hơn, ta viết lại công thức DFT dưới dạng tích ma trận:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \cdots \\ X[N_0-2] \\ X[N_0-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_{N_0}^{1} & W_{N_0}^{2} & \cdots & W_{N_0}^{N_0-1} \\ 1 & W_{N_0}^{2} & W_{N_0}^{4} & \cdots & W_{N_0}^{2(N_0-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & W_{N_0}^{N_0-2} & W_{N_0}^{2(N_0-2)} & \cdots & W_{N_0}^{(N_0-2)(N_0-1)} \\ 1 & W_{N_0}^{N_0-1} & W_{N_0}^{2(N_0-1)} & \cdots & W_{N_0}^{(N_0-1)^2} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \cdots \\ x[n-2] \\ x[n-1] \end{bmatrix}$$

Hiển nhiên ta thấy $W_{N_0}^{-kn} = W_{N_0}^{kn^*}$, vậy ta viết lại công thức IDFT dưới dạng tích ma trận:

$$\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \cdots \\ x[N_0-2] \\ x[N_0-1] \end{bmatrix} = \frac{1}{N_0} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_{N_0}^{1^*} & \cdots & W_{N_0}^{(N_0-1)^*} \\ 1 & W_{N_0}^{2^*} & \cdots & W_{N_0}^{2(N_0-1)^*} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & W_{N_0}^{(N_0-2)^*} & \cdots & W_{N_0}^{(N_0-2)(N_0-1)^*} \\ 1 & W_{N_0}^{(N_0-1)^*} & \cdots & W_{N_0}^{(N_0-1)^{2*}} \end{bmatrix} \begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \cdots \\ X[n-2] \\ X[n-1] \end{bmatrix}$$

Biến đổi Fourier rời rạc (DFT)

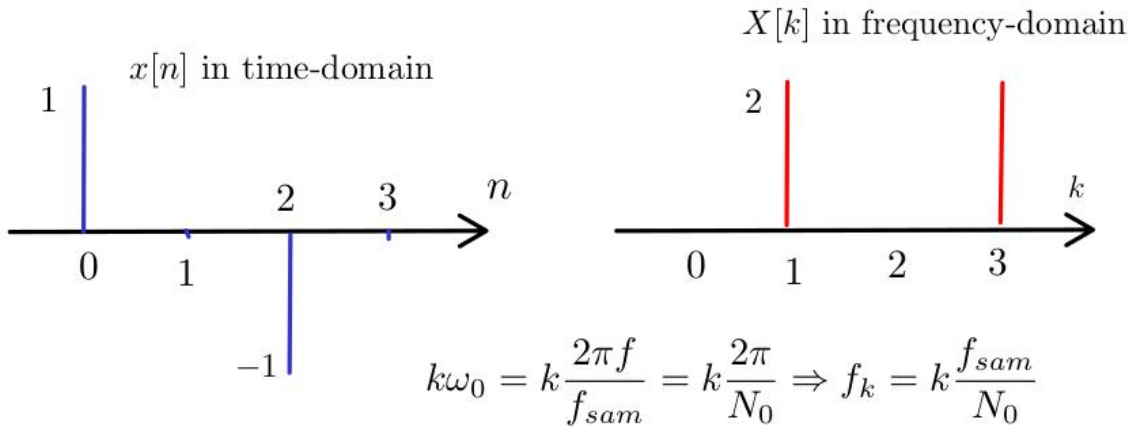
Example: plot the spectrum of $x[n] = \cos\left(\frac{\pi n}{2}\right)$ with $(0 \le n \le 3)$

We obtained data: $x[0] = 1, x[1] = 0, x[2] = -1, x[3] = 0$ and $N_0 = 4$

$\Rightarrow W_{N_0} = e^{-j\frac{2\pi}{N_0}} = e^{-j\frac{\pi}{2}}$ Use the DFT matrix, we have:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_{N_0}^1 & W_{N_0}^2 & W_{N_0}^3 \\ 1 & W_{N_0}^2 & W_{N_0}^4 & W_{N_0}^6 \\ 1 & W_{N_0}^3 & W_{N_0}^6 & W_{N_0}^9 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix}$$



$$k\omega_0 = k\frac{2\pi f}{f_{sam}} = k\frac{2\pi}{N_0} \Rightarrow f_k = k\frac{f_{sam}}{N_0}$$

Hình: DFT matrix

Biến đổi Fourier rời rạc (DFT)

Use the IDFT matrix to check the answer:

$$\begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 2 \\ 0 \end{bmatrix} \Rightarrow \textbf{Matched}$$

Hình: IDFT matrix

Đương nhiên, các bạn hoàn toàn có thể chọn các tín hiệu rời rạc với dữ liệu khác để thay vào công thức DFT và IDFT để kiểm tra kết quả tính toán; ở đây mình chỉ chọn ví dụ đơn giản nhất để minh họa.
Trong quá trình DFT của tín hiệu trong ví dụ trên, chúng ta đã phải thực hiện $4.4 = 16$ phép nhân và $3.4 = 12$ phép cộng, tổng là $17 + 12 = 28$ phép toán cho ma trận 4x4 (hay DFT 4 điểm). Kết quả này hoàn toàn tương tự với IDFT.
Tổng quát, nếu ma trận NxN (hay DFT/IDFT) N điểm, ta sẽ phải thực hiện N^2 phép nhân và $N(N-1)$ phép cộng, ta dễ thấy độ phức tạp của thuật toán này là $\mathcal{O}(N^2)$.
Năm 1965, hai nhà toán học người Mỹ J.W.Cooley và John Tukey đã phát triển thuật toán biến đổi Fourier nhanh (fast Fourier transform) giảm độ phức tạp của thuật toán còn $\mathcal{O}(N \log_2 N)$.

Biến đổi Fourier nhanh (FFT)

Thuật toán biến đổi Fourier nhanh (từ giờ ta sẽ gọi tắt là FFT) rất phức tạp để có thể giải thích toàn bộ (và cũng vượt quá yêu cầu môn học), nên ta chỉ nghiên cứu thuật toán FFT đơn giản nhất là **phân rã trong miền thời gian** (decimation in time - DIT, radix 2), vẽ lưu đồ dòng chảy cho tín hiệu 4, 8 và phát triển giải thuật cho N điểm. Cuối cùng, chương này kết thúc với sử dụng phần mềm mô phỏng (GNU Octave/Matlab) thực thi thuật toán FFT.
Ta xét công thức DFT:

$$X[k] = \sum_{n=0}^{N_0-1} x[n] W_{N_0}^{kn}$$

Ta liên tục tách nhỏ $X[k]$ nhằm tạo các "cặp" phân rã theo cơ số 2 như sau:

Biến đổi Fourier nhanh (FFT)

$$X[k] = \sum_{n=0}^{N_0-1} x[n] W_{N_0}^{kn} = \sum_{r=0}^{\frac{N_0}{2}-1} x[2r] W_{N_0}^{k2r} + \sum_{r=0}^{\frac{N_0}{2}-1} x[2r+1] W_{N_0}^{k(2r+1)}$$

$$= \sum_{r=0}^{\frac{N_0}{2}-1} x[2r] W_{\frac{N_0}{2}}^{kr} + W_{N_0}^{k} \sum_{r=0}^{\frac{N_0}{2}-1} x[2r+1] W_{\frac{N_0}{2}}^{kr} = X_{\text{n even}}[k] + W_{N_0}^{k} X_{\text{n odd}}[k]$$

$$= \left(\sum_{r=0}^{\frac{N_0}{4}-1} x[4r] W_{\frac{N_0}{2}}^{k2r} + \sum_{r=0}^{\frac{N_0}{4}-1} x[4r+2] W_{\frac{N_0}{2}}^{k(2r+1)} \right) +$$

$$W_{N_0}^{k} \left(\sum_{r=0}^{\frac{N_0}{4}-1} x[4r+1] W_{\frac{N_0}{2}}^{k2r} + \sum_{r=0}^{\frac{N_0}{4}-1} x[4r+3] W_{\frac{N_0}{2}}^{k(2r+1)} \right)$$

$$= \left(\sum_{r=0}^{\frac{N_0}{4}-1} x[4r] W_{\frac{N_0}{4}}^{kr} + W_{\frac{N_0}{2}}^{k} \sum_{r=0}^{\frac{N_0}{4}-1} x[4r+2] W_{\frac{N_0}{4}}^{kr} \right) +$$

$$W_{N_0}^{k} \left(\sum_{r=0}^{\frac{N_0}{4}-1} x[4r+1] W_{\frac{N_0}{4}}^{kr} + W_{\frac{N_0}{2}}^{k} \sum_{r=0}^{\frac{N_0}{4}-1} x[4r+3] W_{\frac{N_0}{4}}^{kr} \right)$$

Biến đổi Fourier nhanh (FFT)

Ta thấy do tính tuần hoàn:

$$X_{\text{n even}}[k] = X_{\text{n even}}\left[k + \frac{N_0}{2}\right]$$

$$X_{\text{n odd}}[k] = X_{\text{n odd}}\left[k + \frac{N_0}{2}\right]$$

Vậy từ tất cả các biến đổi cơ bản trên, ta có thể vẽ lại thành một sơ đồ phân rã hình cây nhị phân 3 tầng như sau:

Biến đổi Fourier nhanh (FFT)

$$X[k] = \sum_{n=0}^{N_0-1} x[n] W_{N_0}^{kn}$$

$$X_{\text{n even}}[k] = \sum_{r=0}^{\frac{N_0}{2}-1} x[2r] W_{\frac{N_0}{2}}^{kr} \qquad W_{N_0}^{k} \sum_{r=0}^{\frac{N_0}{2}-1} x[2r+1] W_{\frac{N_0}{2}}^{kr} = W_{N_0}^{k} X_{\text{n odd}}[k]$$

$$\sum_{r=0}^{\frac{N_0}{4}-1} x[4r] W_{\frac{N_0}{4}}^{kr} \qquad W_{\frac{N_0}{2}}^{k} \sum_{r=0}^{\frac{N_0}{4}-1} x[4r+2] W_{\frac{N_0}{4}}^{kr} \qquad \sum_{r=0}^{\frac{N_0}{4}-1} x[4r+1] W_{\frac{N_0}{4}}^{kr} \qquad W_{\frac{N_0}{2}}^{k} \sum_{r=0}^{\frac{N_0}{4}-1} x[4r+3] W_{\frac{N_0}{4}}^{kr}$$

$$X_{\text{n even}}[k] = X_{\text{n even}}\left[k + \frac{N_0}{2}\right]$$

$$X_{\text{n odd}}[k] = X_{\text{n odd}}\left[k + \frac{N_0}{2}\right]$$

Hình: DFT radix 2 binary tree

Biến đổi Fourier nhanh (FFT)

Example: determine the FFT of $x[n] = [1, 2, 3, 4]$ (starting point $n = 0$)

First, we determine $X[0]$ value:

$$X[0] = \sum_{n=0}^{3} x[n] = (x[0] + x[2]) + (x[1] + x[3])$$

$$X_{\text{n even}}[0] = \sum_{r=0}^{1} x[2r] = x[0] + x[2]$$

$$X_{\text{n odd}}[0] = \sum_{r=0}^{1} x[2r+1] = x[1] + x[3]$$

$$\sum_{r=0}^{0} x[4r] = x[0] \qquad \sum_{r=0}^{0} x[4r+2] = x[2]$$

$$\sum_{r=0}^{0} x[4r+1] = x[1] \qquad \sum_{r=0}^{0} x[4r+3] = x[3]$$

Hình: FFT example

Biến đổi Fourier nhanh (FFT)

And then, $X[1]$ value: $X[k] = X_{\text{n even}}[k] + W_{N_0}^k X_{\text{n odd}}[k]$

$$X[1] = \sum_{n=0}^{3} x[n] W_{N_0}^n = (x[0] - x[2]) - j(x[1] - x[3])$$

$$X_{\text{n even}}[1] = \sum_{r=0}^{1} x[2r] W_2^r = x[0] - x[2] \qquad W_4^1 X_{\text{n odd}}[1] = W_4^1 \sum_{r=0}^{1} x[2r+1] W_2^r = -j(x[1] - x[3])$$

$$\sum_{r=0}^{0} x[4r] W_1^r = x[0] \qquad W_2^1 \sum_{r=0}^{0} x[4r+2] W_1^r = -x[2]$$

$$\sum_{r=0}^{0} x[4r+1] W_1^r = x[1] \qquad W_2^1 \sum_{r=0}^{0} x[4r+3] W_1^r = -x[3]$$

Applying the symmetric properties above :

$$X_{\text{n even}}[0+2] = X_{\text{n even}}[0] = x[0] + x[2] \qquad X_{\text{n even}}[1+2] = X_{\text{n even}}[1] = x[0] - x[2]$$

$$X_{\text{n odd}}[0+2] = X_{\text{n odd}}[0] = x[1] + x[3] \qquad X_{\text{n odd}}[1+2] = X_{\text{n odd}}[1] = x[1] - x[3]$$

So, $X[2] = X_{\text{n even}}[2] + W_4^2 X_{\text{n odd}} X[2] = (x[0] + x[2]) - (x[1] + x[3])$

And, $X[3] = X_{\text{n even}}[3] + W_4^3 X_{\text{n odd}}[3] = (x[0] - x[2]) + j(x[1] - x[3])$

Hình: FFT example

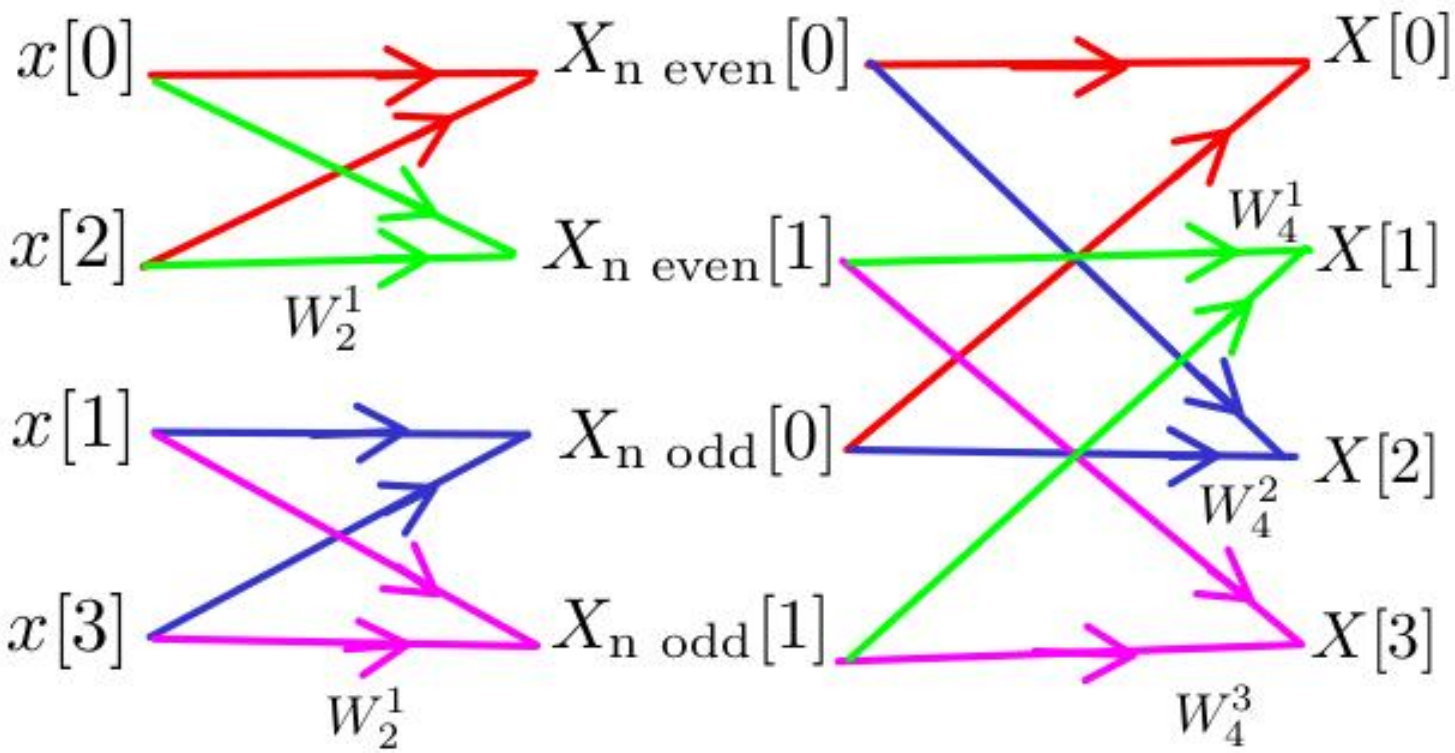
Biến đổi Fourier nhanh (FFT)

Từ các kết quả trên, ta tổng hợp lại được:

$$X[0] = (x[0] + x[2]) + (x[1] + x[3]) = X_{\text{n even}}[0] + X_{\text{n odd}}[0]$$
$$X[1] = (x[0] - x[2]) - j(x[1] - x[3]) = X_{\text{n even}}[1] + W_4^1 X_{\text{n odd}}[1]$$
$$X[2] = (x[0] + x[2]) - (x[1] + x[3]) = X_{\text{n even}}[0] + W_4^2 X_{\text{n odd}}[0]$$
$$X[3] = (x[0] - x[2]) + j(x[1] - x[3]) = X_{\text{n even}}[1] + W_4^3 X_{\text{n odd}}[1]$$

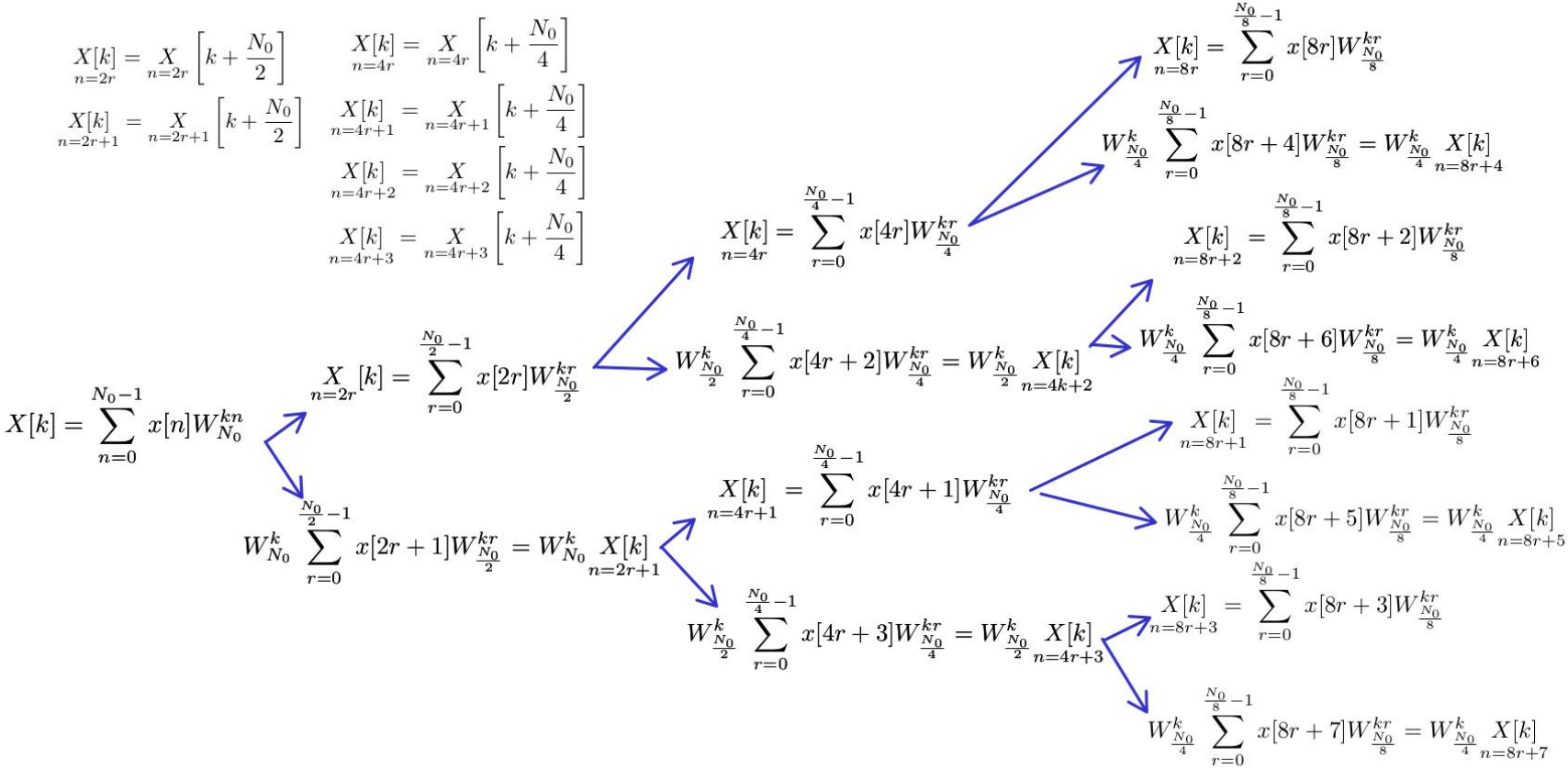
Hiển nhiên ta thấy rằng từ kết quả trên, số bước tính toán đã giảm đi đáng kể, thay vì tính theo cách nhân ma trận như trên, ta chỉ cần thực hiện **6 phép cộng** và **2 phép nhân** rất đơn giản (chỉ là nhân với đơn vị ảo thuần, thậm chí còn không phải nhóm lại phần ảo và thực như cách nhân ma trận).
Ta vẽ lưu đồ dòng chảy tín hiệu (còn gọi là Butterfly diagram) để tính FFT 4 điểm như sau:

Biến đổi Fourier nhanh (FFT)



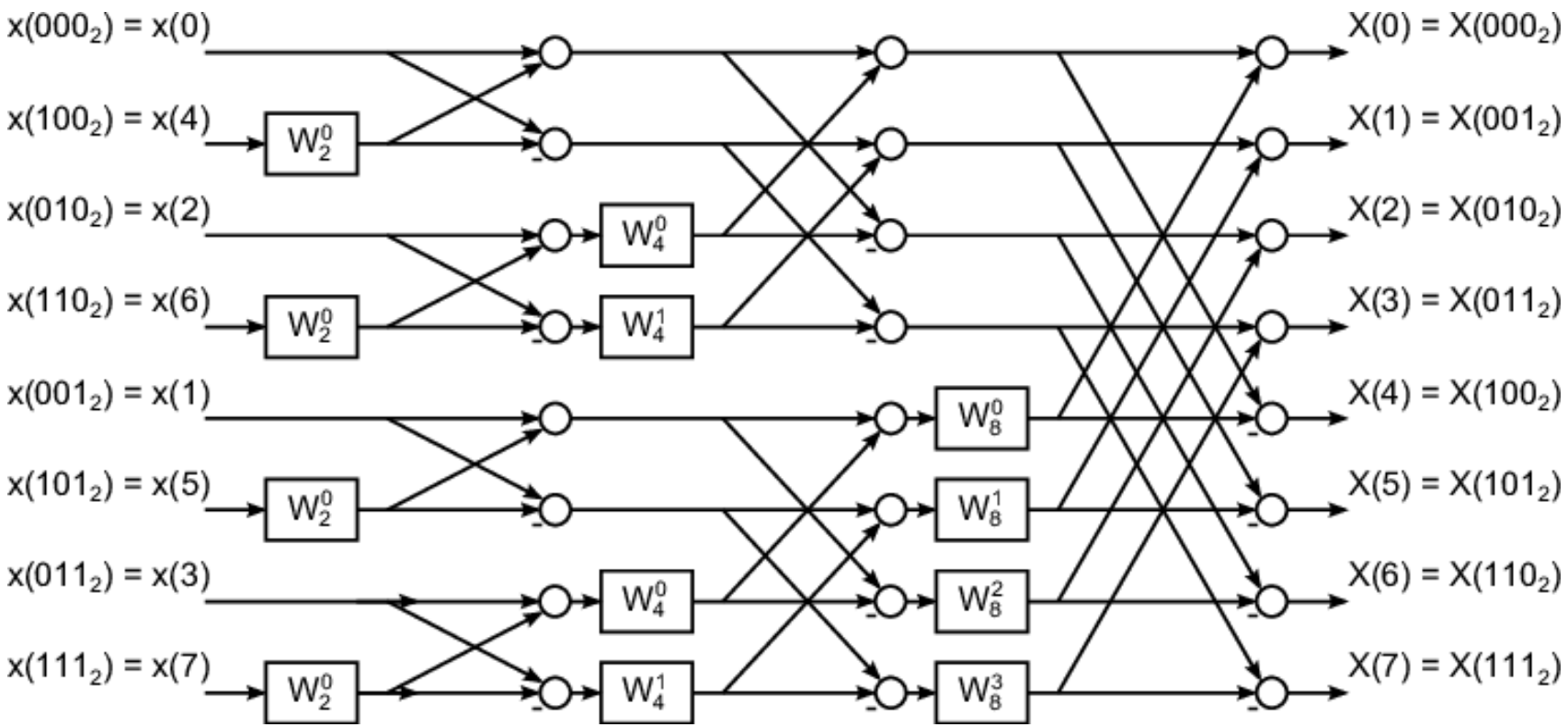
Hình: FFT 4 points Butterfly diagram

Biến đổi Fourier nhanh (FFT)



Hình: FFT 8 points

Biến đổi Fourier nhanh (FFT)



Hình: FFT 8 points Butterfly diagram

Biến đổi Fourier nhanh (FFT)

Nếu ta thay $W_{N_0}^{rk}$ thành liên hợp của nó $W_{N_0}^{-rk} = (W_{N_0}^{rk})^*$ trong các công thức và lưu đồ tín hiệu, sau đó chia toàn bộ kết quả cho N_0, ta thu được thuật toán IFFT hoàn toàn tương tự với FFT.

Thực hành

Ta khởi tạo một chuỗi tín hiệu rời rạc 8 điểm bằng mảng trong GNU Octave/Matlab và tính hệ số phổ của nó như sau:

```
x=[1 2 3 4 5 6 7 8];
N=8; % 8 points
X=fft(x,N);
```

Hiển nhiên, vì phần mềm sử dụng thuật toán DIT radix 2, nên N phải là lũy thừa của 2.
Nếu ta thay N khác, phần mềm sẽ "zero padding", bù các phần tử 0 vào mảng cho đến khi N đủ lũy thừa 2.
Ở chế độ mặc định, ta bỏ tham số N ra khỏi hàm `fft()`, phần mềm sẽ giải thuật cho L điểm (với L là chiều dài mảng). Từ giờ ta cũng sẽ không sử dụng tham số N nữa.

```
x=[1 2 5 6 8 9]; %L=6
X=fft(x)
```

Với hàm `ifft()` ta cũng tiến hành hoàn toàn tương tự, nhưng lưu ý nhân thêm hệ số $\frac{1}{L}$.
Ta muốn vẽ phổ biên độ và phổ pha của tín hiệu $x[n]$. Quy trình vẽ được tiến hành như sau:

Thực hành

```
x=[1 2 3 4];
L=length(x);
X=fft(x);
Mag_X=abs(X)/L;
Phase_X=angle(X);
k=linspace(0,L-1,L);
figure(1);
stem(k,Mag_X);
figure(2);
stem(k,Phase_X);
pause;
```

Tất nhiên đây chỉ là hướng dẫn **cực kì cơ bản** giúp cho các bạn làm quen với hàm `fft()` trong phần mềm mô phỏng. Bằng hàm `fft()` các bạn không chỉ vẽ được phổ của chuỗi DFT; mà còn có thể vẽ phổ tín hiệu liên tục, rời rạc và hơn thế nữa. Các bạn có thể nghiên cứu bài toán vẽ phổ để nghiệm thu lại hiện tượng chồng phổ (aliasing), tốc độ và tần số Nyquist (Nyquist rate and Nyquist frequency) ta đã tìm hiểu ở Chương 2.