FFT: for $N = p \cdot q$

Ex: $p = 3$ (Decimation-in-time approach)

$$X[k] = \sum_{n=0,3,6\cdots}^{\frac{N}{3}-1} x[n]\, e^{-j\frac{2\pi kn}{N}} + \sum_{n=1,4,7,\cdots}^{\frac{N}{3}-1} x[n]\, e^{-j\frac{2\pi kn}{N}} + \sum_{n=2,5,8,\cdots}^{\frac{N}{3}-1} x[n] e^{-j\frac{2\pi kn}{N}}$$

$n = 3l$ $\quad$ $n = 3l+1$ $\quad$ $n = 3l+2$

$$X[k] = \sum_{l=0}^{\frac{N}{3}-1} x[3l]\, e^{-j\frac{2\pi k 3l}{N}} + \sum_{l=0}^{\frac{N}{3}-1} x[3l+1]\, e^{-j\frac{2\pi k(3l+1)}{N}} + \sum_{l=0}^{\frac{N}{3}-1} x[3l+2] e^{\cdots}$$

$$X[k] = \sum_{l=0}^{\frac{N}{3}-1} x[3l]\, e^{-j\frac{2\pi kl}{N/3}} + e^{-j\frac{2\pi k}{N}} \sum_{l=0}^{N/3-1} x[3l+1]\, e^{-j\frac{2\pi kl}{N/3}} + e^{-j\frac{2\pi k 2}{N}} \sum_{l=0}^{N/3-1} x[3l+2] e^{-j\frac{2\pi k}{N/}}$$

$$X[k] = G[k] + e^{-j\frac{2\pi k}{N}} H[k] + e^{-j\frac{2\pi k 2}{N}} V[k], \quad k = 0, 1, \ldots, N-1$$

where $G[k]$ is the $\frac{N}{3}$-point DFT of $\{x[0], x[3], \ldots x[N-$

$H[k]$ " " " " " " $\{x[1], x[4], \ldots x[N-$

$V[k]$ " " " " " " $\{x[2], x[5], \ldots x[N-1]$

After this single stage decomposition compute the N-point DFT of $x[n]$ using 3 $\frac{N}{3}$-point DFT's.

Total cost: $3\left(\frac{N}{3}\right)^2 + 2N < N^2$ for large N.

---

* In general factor N into its primes

i.e. $N = \underbrace{p \cdot q \cdots r}_{l \text{ primes}}$ and perform the DFT in l stages..

FFT 9

Radix-4 DFT is the most efficient DFT:

Reason: In 4-point DFT you don't perform any actual multiplication:

$$X[k] = \sum_{n=0}^{3} x[n]\, e^{-j\frac{2\pi kn}{N}}, \quad k = 0, 1, 2, 3$$

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix}$$

Comp. cost (# multiplications)

It is also an Order $(N \log N)$ algorithm.

Convolution using DFT. (FFT):

$$y[n] = h[n] * x[n] \quad \xleftrightarrow{DFT_N} \quad Y[k] = H[k] \cdot X[k]$$

$h[n]$: length L_1; $x[n]$: length L_2; $y[n]$: length $L_1 + L_2 - 1$

$N > L_1 + L_2 - 1$

DFT^{-1} ↕ $y[n]$

Implementation:

			Comp. cost
1st)	Compute	$H[k]$	$\frac{N}{2}\log_2 N$
2nd)	"	$X[k]$	"
3rd)	Compute	$H[k]\,X[k]$, $k = 0, 1, \ldots, N-1$	N
4th)	"	$DFT^{-1}[H[k] \cdot X[k]]$	$\frac{N}{2}\log_2 N$

Total $= N + 3\frac{N}{2}\log_2 N$ complex ⊗ $= 4N + 6N\log_2 N$ real ⊗

- For long (or large order filters) perform the convolution using FFT.

+ For each case carry out a computational cost analysis and decide!

Ex// Filter order:

$L_1 = 11$, $L_2 = 900$

$$y[n] = \sum_{k=0}^{L_1-1} h[k]\, x[n-k]$$

For a single $y[n]$ we need to perform 11 ⊗.

Total cost of time domain convolution

$\leq L_1 \cdot (L_1 + L_2 - 1) \cong 11\,(910) \cong 10000$ ⊗.

Freq. Domain convolution requires ($N = 1024$)

$4N + 6N\log_2 N \cong 4000 + 60000$ ⊗.

Time domain conv. is better in this example!

Ex// $L_1 = 101$, $L_2 = 900$

Time domain convolution $\cong 100 \cdot 910 \cong 90000$ ⊗

Freq. " " $= 4N + 6N\log_2 N = 64000$ ⊗

Freq. domain convolution is computationally better.