From DFT to Fast FFT: A Practical Derivation and Implementation Notes

LineInTuner Notes

September 13, 2025

1 The Discrete Fourier Transform (DFT)

Given a length-N sequence $x[n]$, the DFT is

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1. \tag{1}$$

This direct computation is $\mathcal{O}(N^2)$: N outputs, each summing N complex multiplications.

In many applications we only need the magnitude spectrum, defined as

$$|X[k]| = \sqrt{\mathrm{Re}\{X[k]\}^2 + \mathrm{Im}\{X[k]\}^2}. \tag{2}$$

2 Even/Odd Decomposition (Why FFT Works)

Assume N is even. Split the input into its even and odd-indexed subsequences:

$$x_e[r] = x[2r], \tag{3}$$

$$x_o[r] = x[2r+1], \qquad r = 0, \ldots, \frac{N}{2} - 1. \tag{4}$$

Then

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} && (5)\\
&= \sum_{r=0}^{\frac{N}{2}-1} x_e[r] e^{-j2\pi k(2r)/N} + \sum_{r=0}^{\frac{N}{2}-1} x_o[r] e^{-j2\pi k(2r+1)/N} && (6)\\
&= \sum_{r=0}^{\frac{N}{2}-1} x_e[r] e^{-j2\pi kr/(N/2)} + e^{-j2\pi k/N} \sum_{r=0}^{\frac{N}{2}-1} x_o[r] e^{-j2\pi kr/(N/2)} && (7)\\
&= E[k \bmod N/2] + W_N^k\, O[k \bmod N/2], && (8)
\end{aligned}$$

where $E[\cdot]$ and $O[\cdot]$ are the $N/2$-point DFTs of x_e and x_o, and $W_N^k = e^{-j2\pi k/N}$ are the *twiddle factors.*

This identity expresses an N-point DFT in terms of two $(N/2)$-point DFTs and $\mathcal{O}(N)$ additional work. Recursing yields $\mathcal{O}(N \log_2 N)$ complexity.

3 Radix-2 Decimation-in-Time (DIT) FFT

For $N = 2^m$, the DIT FFT proceeds in $m = \log_2 N$ stages. An efficient *iterative* in-place algorithm has two key components:

3.1 Bit-Reversal Permutation

The recursive even/odd splitting implies a particular data access order. The iterative algorithm first permutes the input to *bit-reversed* order: interpret the index i in binary with m bits and reverse those bits to obtain j; swap entries at i and j when $i < j$. This ensures subsequent stages operate on contiguous subproblems.

3.2 Butterfly Computation

Let m denote the butterfly size at a given stage, doubling each stage: $m = 2, 4, 8, \ldots, N$. Define the primitive twiddle for the stage as

$$w_m = e^{-j2\pi/m}. \tag{9}$$

Process the array in blocks of length m. Within each block, for $j = 0, 1, \ldots, m/2 - 1$, maintain a running twiddle $w = w_m^j$ and perform the butterfly:

$$\begin{aligned} t &= w \cdot a[i + j + m/2], && (10) \\ u &= a[i + j], && (11) \\ a[i + j] &\leftarrow u + t, && (12) \\ a[i + j + m/2] &\leftarrow u - t. && (13) \end{aligned}$$

Here $a[\cdot]$ is the in-place complex working array. Advancing w by a complex multiply $w \leftarrow w \cdot w_m$ avoids repeated trigonometric calls.

After the final stage, $a[k] = X[k]$. For magnitude output, return $|a[k]|$.

4 Why Bit-Reversal and Butterflies Yield the DFT

The even/odd decomposition proves that an N-point DFT equals a combination of two $(N/2)$-point DFTs with twiddles. Applying the same decomposition recursively yields a computation DAG (dependency graph). The bit-reversal permutation reorders the input so that the DAG can be evaluated *iteratively* with contiguous memory accesses: each stage merges pairs of sub-transforms already computed at the previous stage. The butterflies are exactly those merge operations. Thus, the iterative algorithm computes precisely the same $X[k]$ as the definition.

5 Complexity and Practical Notes

- **Complexity**: $\mathcal{O}(N \log_2 N)$ vs $\mathcal{O}(N^2)$ for direct DFT.
- **Twiddle reuse**: One $\sin / \cos$ per stage; use complex multiplies to step twiddles within the stage.
- **Sign convention**: The forward FFT uses the negative sign in the exponent, consistent with the DFT above.
- **Normalization**: Often omitted in the forward transform; apply $1/N$ or $2/N$ (single-sided) when converting to amplitude.
- **Real inputs**: A real FFT can halve computation/storage by exploiting conjugate symmetry, but a standard complex FFT is simpler to implement correctly first.
- **Windowing**: For non-coherent tones, apply a window (e.g., Hann) to reduce spectral leakage before the FFT.

6 Rust-Oriented Implementation Sketch

Assume input x of length $N = 2^m$ and separate real/imag arrays.

```
// 1) Initialize
for i in 0..N { re[i] = x[i]; im[i] = 0.0; }

// 2) Bit-reversal permutation (m = log2(N))
for i in 0..N {
  j = reverse_bits(i, m);
  if i < j { swap(re[i], re[j]); swap(im[i], im[j]); }
}

// 3) Butterfly stages
for msize in [2, 4, 8, ..., N] {
  theta = -2*pi / msize;
  (wm_sin, wm_cos) = sin_cos(theta);
  for block in (0..N step msize) {
    w_re = 1.0; w_im = 0.0; // w = 1
    for j in 0..msize/2 {
      i1 = block + j;
      i2 = i1 + msize/2;
      // t = w * a[i2]
      t_re = w_re * re[i2] - w_im * im[i2];
      t_im = w_re * im[i2] + w_im * re[i2];
      // u = a[i1]
      u_re = re[i1]; u_im = im[i1];
      // a[i1] = u + t
      re[i1] = u_re + t_re; im[i1] = u_im + t_im;
      // a[i2] = u - t
      re[i2] = u_re - t_re; im[i2] = u_im - t_im;
      // w *= wm
      tmp_re = w_re * wm_cos - w_im * wm_sin;
      tmp_im = w_re * wm_sin + w_im * wm_cos;
      w_re = tmp_re; w_im = tmp_im;
    }
  }
}

// 4) Magnitude
for k in 0..N { out[k] = sqrt(re[k]^2 + im[k]^2); }
```

7 Worked Examples

7.1 Slow DFT by Hand (N=4)

Let $N = 4$ and define $W_4 = e^{-j2\pi/4} = e^{-j\pi/2} = -j$. Its powers are

$$W_4^0 = 1, \quad W_4^1 = -j, \quad W_4^2 = -1, \quad W_4^3 = j. \tag{14}$$

Choose a simple signal $x = [1,\ 0,\ -1,\ 0]$. Then

$$X[0] = x_0 W_4^{0\cdot 0} + x_1 W_4^{0\cdot 1} + x_2 W_4^{0\cdot 2} + x_3 W_4^{0\cdot 3} = 1 + 0 + (-1) + 0 = 0, \tag{15}$$
$$X[1] = 1 \cdot W_4^0 + 0 \cdot W_4^1 + (-1) \cdot W_4^2 + 0 \cdot W_4^3 = 1 + 0 + (+1) + 0 = 2, \tag{16}$$
$$X[2] = 1 \cdot W_4^0 + 0 \cdot W_4^2 + (-1) \cdot W_4^4 + 0 \cdot W_4^6 = 1 + 0 + (-1) + 0 = 0, \tag{17}$$
$$X[3] = 1 \cdot W_4^0 + 0 \cdot W_4^3 + (-1) \cdot W_4^6 + 0 \cdot W_4^9 = 1 + 0 + (+1) + 0 = 2. \tag{18}$$

Thus the magnitude spectrum is $|X| = [0,\ 2,\ 0,\ 2]$ (unnormalized). This matches the intuition: x is a cosine at bin $k = 1$ (and its conjugate at $k = 3$).

7.2 Fast Radix-2 FFT by Stages (N=8)

Let $N = 8$ and consider $x = [1,\ 0,\ -1,\ 0,\ 1,\ 0,\ -1,\ 0]$, i.e. $x[n] = \cos\big(2\pi \cdot 2\,n/8\big)$. We know in advance that the DFT has peaks at $k = 2$ and $k = 6$ with magnitude $N/2 = 4$ (unnormalized). We now walk through the radix-2 DIT FFT.

Bit-Reversal Permutation. With $m = \log_2 8 = 3$ bits, reversing the bits of index i gives

$$0 \to 0,\ 1 \to 4,\ 2 \to 2,\ 3 \to 6,\ 4 \to 1,\ 5 \to 5,\ 6 \to 3,\ 7 \to 7. \tag{19}$$

Reordering x accordingly (imaginary parts initially zero) yields the in-place array a ready for butterflies.

Stage $m = 2$. Block size $m = 2$; there is only $j = 0$ with twiddle $w = 1$. Each butterfly just adds/subtracts adjacent pairs. After this stage, pairs $(x[0], x[1])$, $(x[2], x[3])$, etc. are combined; values remain real.

Stage $m = 4$. Now $w_m = e^{-j2\pi/4} = -j$. For each block of length 4, $j = 0$ uses $w = 1$ (real add/sub), and $j = 1$ uses $w = -j$, introducing purely imaginary contributions that form the $k = 1$ and $k = 3$ partials within each 4-point subtransform.

Stage $m = 8$. Finally, $w_m = e^{-j2\pi/8} = e^{-j\pi/4} = \frac{1}{\sqrt{2}}(1-j)$. This mixes the two 4-point results to produce the full 8-point spectrum. Evaluating the butterflies yields (up to tiny roundoff)

$$|X| \approx [0,\ 0,\ 4,\ 0,\ 0,\ 0,\ 4,\ 0], \tag{20}$$

with peaks at $k = 2$ and $k = 6$, as predicted from $x[n] = \cos(2\pi \cdot 2\,n/8)$.

These stages illustrate how the FFT builds larger DFTs out of smaller ones: stage $m = 2$ forms 2-point DFTs, stage $m = 4$ combines them into 4-point DFTs, and stage $m = 8$ completes the 8-point DFT.

8 Validation

- **Delta input**: $x[n] = \delta[n]$ yields $X[k] = 1$ for all k (unnormalized).
- **Coherent cosine**: $x[n] = \cos(2\pi k_0 n/N)$ produces peaks at k_0 and $N - k_0$ with magnitude $\approx N/2$.
- Compare the fast FFT output against a slow DFT on small N for parity.