Proving correctness of Fast Discrete Fourier transforms

Henning Thielemann

2025-03-13

Discrete Fourier Transform

Discrete Fourier Transform as Matrix $F \in R^{n \times n}$ for size n:

$$F_{j,k} = z^{j \cdot k} \quad \text{with } z^n = 1$$

Example $n = 6$:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} z^0 & z^0 & z^0 & z^0 & z^0 & z^0 \\ z^0 & z^1 & z^2 & z^3 & z^4 & z^5 \\ z^0 & z^2 & z^4 & z^0 & z^2 & z^4 \\ z^0 & z^3 & z^0 & z^3 & z^0 & z^3 \\ z^0 & z^4 & z^2 & z^0 & z^4 & z^2 \\ z^0 & z^5 & z^4 & z^3 & z^2 & z^1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

Inverse Discrete Fourier transforms

$$(F^{-1})_{j,k} = \frac{1}{n} \cdot z^{-j \cdot k} \quad \text{with } z - 1 \text{ no zero-divisor}$$

Applications

In $\mathbb{C}$: Swiss army knife of digital signal processing

- Spectrum analysis
- Frequency filtering
- Additive synthesis
- Resampling and interpolation
- Data compression
- Fast convolution

In finite fields:

- Big Integer multiplication

Runtime

- Direct implementation: Quadratic runtime
- Fast Fourier Transforms: Linear-logarithmic
- Actual runtime does not grow monotonic with data size,
- highly depends on composition of data size integer
- Problem: Cannot simply pad with zeros!

Algorithms

Fast Fourier Transform:

- Cooley-Tukey: composite data size
- Good-Thomas: coprime composite data size
- Rader: prime data size
- Bluestein: any size

Fast Convolution:

- Karatsuba
- Toom-Cook
- Fast Fourier Transform
- General idea: Polynomial interpolation

General ideas

- 2D-FFT: Cooley-Tukey, Good-Thomas
- 2D-FFT: two batched 1D-FFTs
- Convolution: Rader, Bluestein
- Convolution theorem: Convert Convolution to Multiplication via FFT
- Circular vs. Plain convolution

Trade-offs

- multiplications
- additions
- padding (more data to work on)
- shuffling

Minimal implementation

- Cooley-Tukey on 2^n data points
- Bluestein (Chirp) transform

Cooley-Tukey

- Composite data size
- Reorganize in 2D

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{pmatrix} \rightarrow \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \end{pmatrix}$$

Steps:

- Columnwise 1D-FFTs
- Elementwise Multiplication with Twiddle Factors $((k, j) \mapsto z^{k \cdot j})$
- Rowwise 1D-FFTs

Bluestein

- Any data size
- Steps:
  - Multiply with chirp ($j \mapsto z^{j^2/2}$)
  - Convolve with chirp
  - Multiply with chirp
- Convolution done via FFT
- Employ identity $\frac{(k+j)^2}{2} - \frac{k^2}{2} - \frac{j^2}{2} = k \cdot j$

Good-Thomas

- Composite data size, coprime factors
- Steps:
  - Conversion to 2D on a skew grid
  - 2D-FFT (= two batched 1D-FFTs)
  - Reordering to 1D with Chinese Remainder Theorem

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \end{pmatrix} \rightarrow \begin{pmatrix} x_0 & x_2 & x_4 \\ x_3 & x_5 & x_1 \end{pmatrix}$$

Rader

- Prime data size n
- Conversion to Cyclic Convolution on size $n-1$
- Convolution done via FFT on size $n-1$

Permutation:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \rightarrow \begin{pmatrix} x_1 & x_2 & x_4 & x_3 \end{pmatrix}$$

$$\begin{pmatrix} z^0 & z^0 & z^0 & z^0 & z^0 \\ z^0 & z^1 & z^2 & z^3 & z^4 \\ z^0 & z^2 & z^4 & z^1 & z^3 \\ z^0 & z^3 & z^1 & z^4 & z^2 \\ z^0 & z^4 & z^3 & z^2 & z^1 \end{pmatrix} \rightarrow \begin{pmatrix} z^1 & z^2 & z^4 & z^3 \\ z^2 & z^4 & z^3 & z^1 \\ z^4 & z^3 & z^1 & z^2 \\ z^3 & z^1 & z^2 & z^4 \end{pmatrix}$$

2 is primitive root of unity in $\mathbb{Z}_5$.

Conclusion

- Actual work: Lots of Lemmas on `Vector`
- Precise tracking of preconditions for properties
- `leansearch.net` is awesome

Questions:

- How to perform proofs about runtime? (respect sharing)
- Proofs about Floating Point computations? (`Add` vs. `AddMonoid`)
- nice to have:
  `instance Pow` for `Units ^ Fin` or `M ^ ZMod (Monoid.exponent M)`