COSC 462

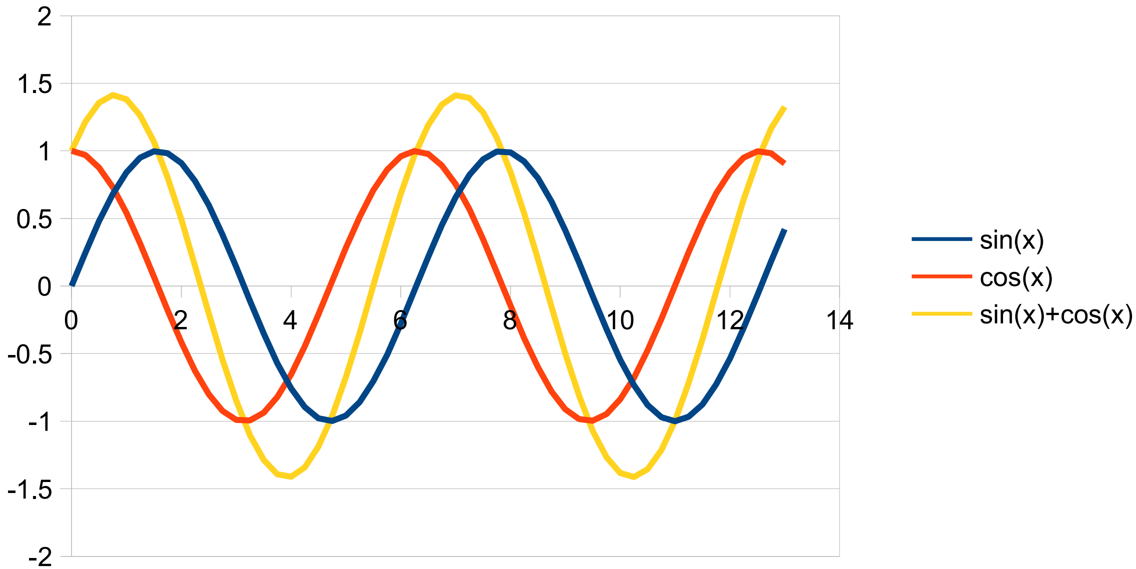
Fast Fourier Transform

Piotr Luszczek

FFT Applications

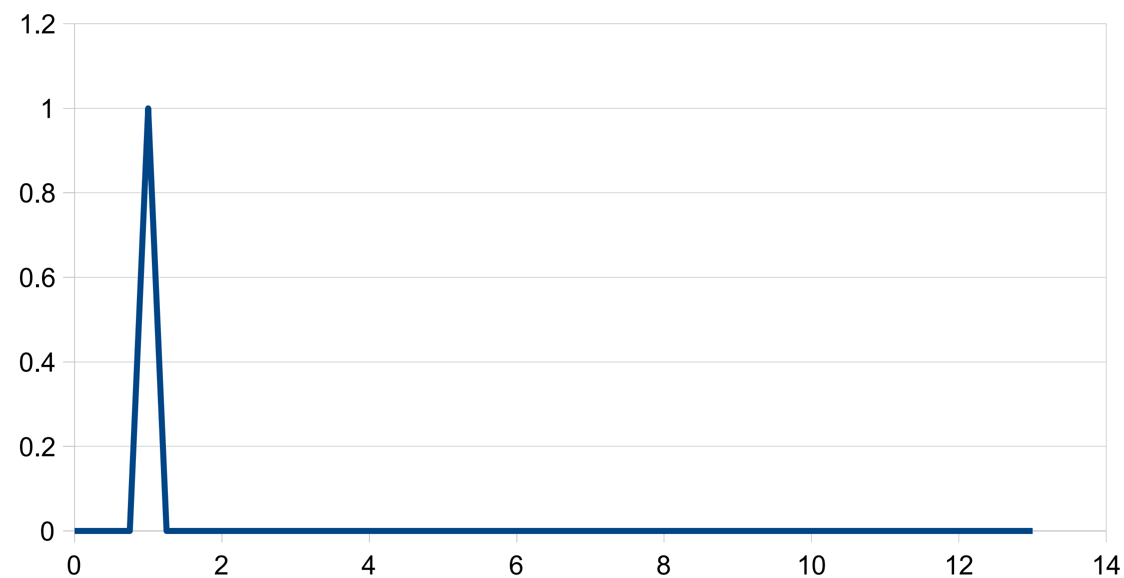
- Image processing
  - Compression
  - Filtering
- Signal analysis
  - Compression
  - Filtering
  - Transformation
- Electronic structure calculation
  - 3D FFT
- Deep learning
  - Convolutional Neural Networks
- Related problems
  - Polynomial multiplication
  - Convolutions

FFT: Continuous Case



Inverse FFT

Time Domain



FFT

Frequency Domain

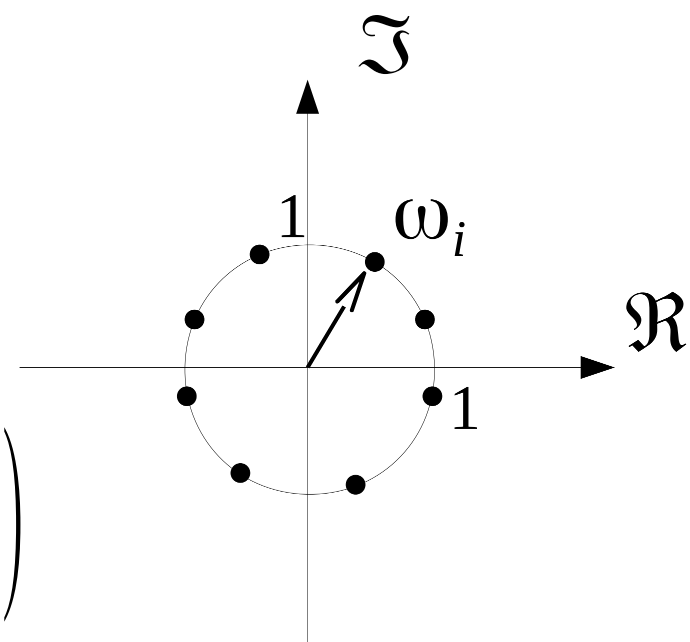
FFT: Continuous and Discrete Formulas

$$F(f) = \int_{-\infty}^{+\infty} f(t) \mathrm{e}^{-i 2 \pi t} \, dt$$

$$y_k = \sum_{n=0}^{N-1} x_n e^{-i 2 \pi k n / N}$$

$$y = W x = \begin{bmatrix} \omega_n^{0\times 0} & \omega_n^{0\times 1} & \omega_n^{0\times 2} & \cdots \\ \omega_n^{1\times 0} & \omega_n^{1\times 1} & \omega_n^{1\times 2} & \cdots \\ \omega_n^{2\times 0} & \omega_n^{2\times 1} & \omega_n^{2\times 2} & \cdots \end{bmatrix} x$$

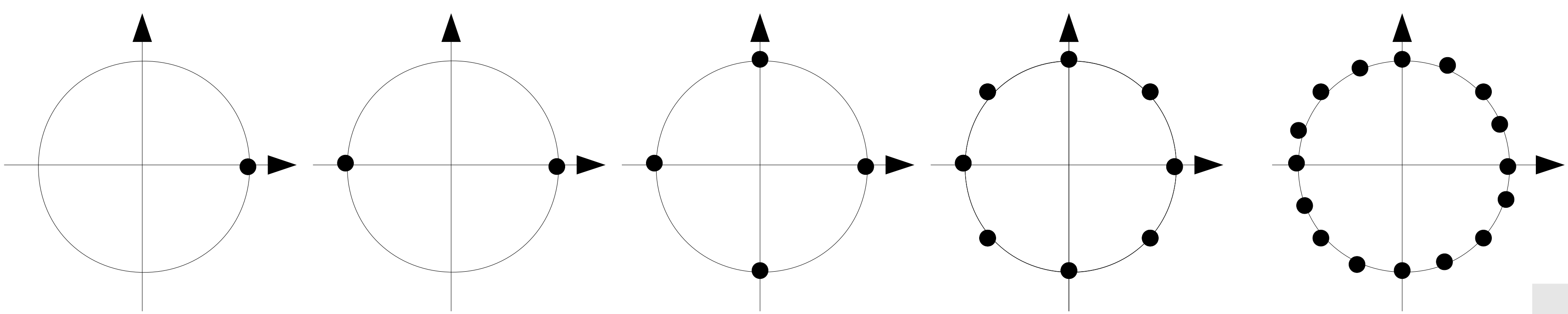
$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} y_k e^{i 2 \pi k n / N}$$



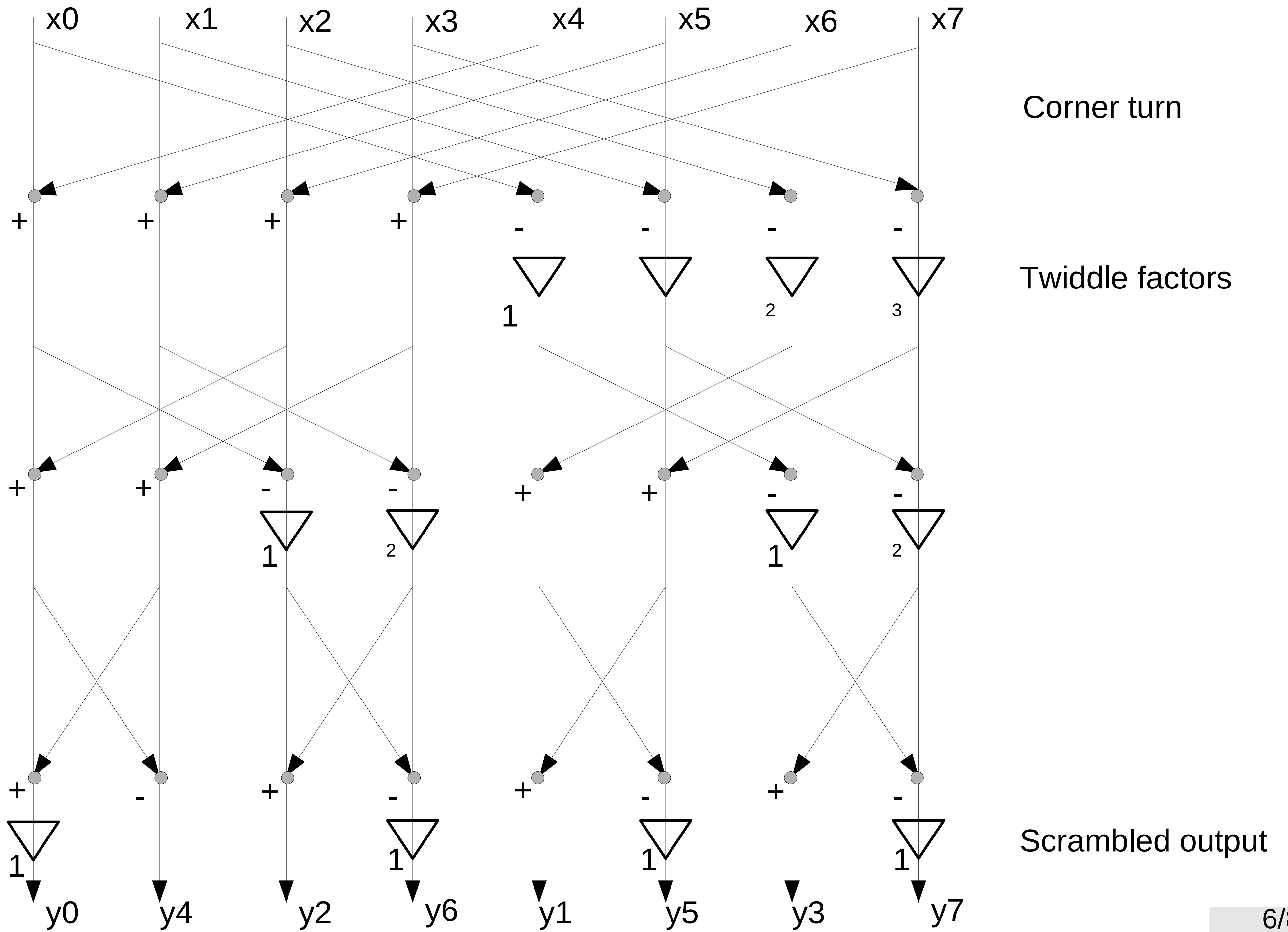
$$\omega_n = e^{i \frac{2\pi}{n}} = \cos\left(\frac{2\pi}{n}\right) + \sin\left(\frac{2\pi}{n}\right)$$

Computational and Complexity Considerations

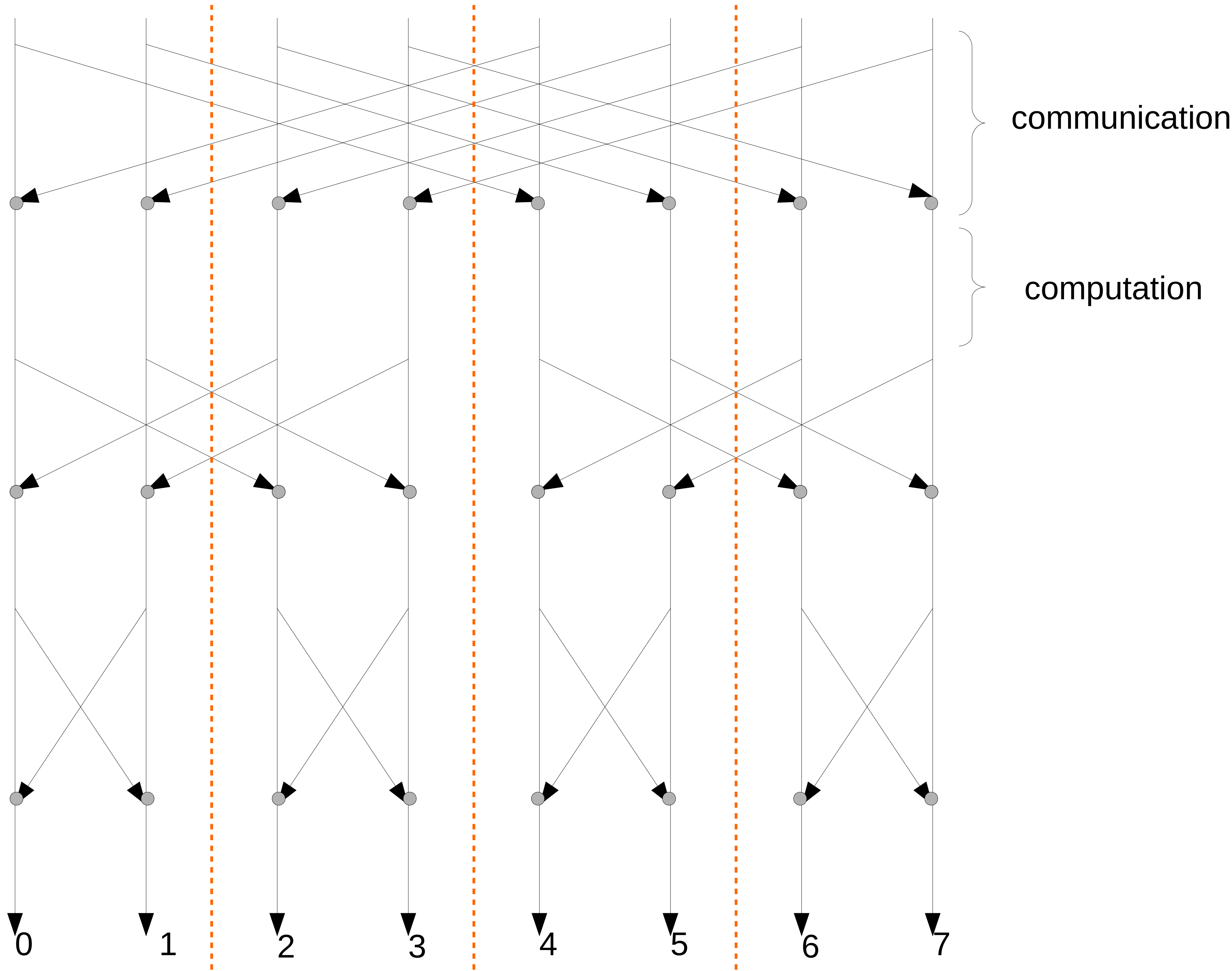
- Discrete Fourier Transform and inverse transform is a matrix-vector multiply (the matrix is symmetric)
  - Complexity: $\Theta(N^2)$
  - Matrix entries come from evaluation of transcendental functions
    - Very costly if implemented in software
    - Order of magnitude slower than add/multiply if done in hardware
- The transform matrix has a (recursive) structure
  - This observation leads to Fast Fourier transform
  - Complexity: $\Theta(N \log N)$
  - Values from transcendental functions can be build incrementally



Data Transfer Pattern: Butterfly



Partitioning, Agglomeration, and Mapping



Remaining Details: Divisibility, Padding, Caches

- Textbooks often deal with input/output vectors as powers of 2
  - $N = 2^m$
  - $P = 2^t$
- Modern memory hierarchy (caches, TLB) and structure (cache lines, pages, cache associativity) is constructed on powers of 2
  - Cache line = 32 or 64
  - TLB page = 2^{12} or 2^{20}
  - Accessing data in power-of-2 stride is sub-optimal
- Padding to power of 2 is trivial but wastes a lot memory
- Modern libraries include specialized code for other powers
  - 2^n, 3^m, 5^k, 7^i, 11^j, 13^x
- Processors count P has to divide N
- FFT algorithm for prime-number length exists...
  - But better performance can be achieved with padding