The Fast Fourier Transform (FFT)

Digital Signal Processing

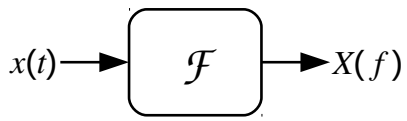
Notes

Contents

Notes

Fourier Transform: Calculate the frequencies in a signal

- Fourier transform



where $x(t)$ is the signal, t is time or sample index, $X(f)$ is the frequency representation and f is the frequency.

- Inverse Fourier transform

Notes

Discrete Fourier Transform (DFT)

- Discrete Fourier transform of a discrete periodic signal $x(t)$

$$X(f) = \mathcal{F}\left(X(f)\right) = \frac{1}{N}\sum_{t=0}^{N-1} x(t)\exp\left(-j\frac{2\pi}{N}ft\right)$$

- Inverse Discrete Fourier transform

$$x(t) = \mathcal{F}^{-1}\left(x(t)\right) = \sum_{f=0}^{N-1} X(f)\exp\left(j\frac{2\pi}{N}ft\right)$$

But how to calculate these on a computer?

Notes

Simple Algorithm to Compute DFT

Simple DFT algorithm can compute the real and imaginary parts independently then combine at the end.

Separate the real and imaginary parts of the complex exponential using Euler's identity:

$$\exp\left(-j\frac{2\pi}{N}ft\right) = \cos\left(\frac{2\pi}{N}ft\right) - j\sin\left(\frac{2\pi}{N}ft\right)$$

so that

$$X(f) = \frac{1}{N}\sum_{t=0}^{N-1} x(t)\left(\cos\left(\frac{2\pi}{N}ft\right) - j\sin\left(\frac{2\pi}{N}ft\right)\right).$$

Then complex multiplication:

$$X(f) = \frac{1}{N}\sum_{t=0}^{N-1}\left(\cos\left(\frac{2\pi}{N}ft\right)\mathrm{Re}(x(t)) + \sin\left(\frac{2\pi}{N}ft\right)\mathrm{Im}(x(t))\right.$$
$$\left. -j\left(\sin\left(\frac{2\pi}{N}ft\right)\mathrm{Re}(x(t)) + \cos\left(\frac{2\pi}{N}ft\right)\mathrm{Im}(x(t))\right)\right).$$

Notes

Simple Algorithm to Compute DFT:

```
simpleDFT(xRe, xIm, XRe, XIm, N)
    a = 2 × π/N
    for f = 0 to N − 1
        XRe[f] = 0, XIm[f] = 0, b = a × f
        for t = 0 to N − 1
            arg = b × t, cosarg = cos(arg), sinarg = sin(arg)
            XRe[f] = XRe[f] + cosarg × xRe[t] + sinarg × xIm[t]
            XIm[f] = XIm[f] − sinarg × xRe[t] − cosarg × xIm[t]
        end for
    end for
```

$\times 2$ for loops, one *nested* inside other, both length N.

$\therefore$ **running time** $\propto N^2$.

A bit slow, especially for large N.

Notes

How to Reduce Running Time?
Divide and Conquer

Consider the DFT equation:

$$X(f) = \frac{1}{N}\sum_{t=0}^{N-1} x(t)\exp(-j\frac{2\pi}{N}ft)$$

The sum can be split into even and odd components:

$$N \times X(f) = \sum_{\tau=0}^{N/2-1} x(2\tau)\exp\left(-j\frac{2\pi}{N}f2\tau\right) + \sum_{\tau=0}^{N/2-1} x(2\tau+1)\exp\left(-j\frac{2\pi}{N}f(2\tau+1)\right)$$

$$= \underbrace{\sum_{\tau=0}^{N/2-1} x(2\tau)\exp\left(-j\frac{2\pi}{N/2}f\tau\right)}_{\text{even}} + \underbrace{\exp\left(-j\frac{2\pi}{N}f\right)}_{\text{twiddle}} \underbrace{\sum_{\tau=0}^{N/2-1} x(2\tau+1)\exp\left(-j\frac{2\pi}{N/2}f\tau\right)}_{\text{odd}}$$

So that

$$N \times X(f) = X^{x_e}_{N/2}(f) + T(N,f) \times X^{x_o}_{N/2}(f)$$

where $x_e(\tau) = x(2\tau)$ and $x_o(\tau) = x(2\tau+1)$.

Notes

DFT Iterative Calculations

Some useful points to remember:

- The DFT is symmetric about the point $N/2$ which means

$$X(f) = X(f + N/2).$$

- Also noting

$$\begin{aligned}\exp(-j2\pi(f+N/2)/N) &= \exp(-j\pi)\exp(-j2\pi f/N)\\ &= -\exp(-j2\pi f/N).\end{aligned}$$

Therefore we now have

$$N \times X(f) = \begin{cases} X^{x_e}_{N/2}(f) + T(N,f) \times X^{x_o}_{N/2}(f) & \text{for} \quad 0 \le f < N/2, \\ X^{x_e}_{N/2}(f-\frac{N}{2}) - T(N,f-\frac{N}{2}) \times X^{x_o}_{N/2}(f-\frac{N}{2}) & \text{for} \quad N/2 \le f < N. \end{cases}$$

Notes

How to Reduce Running Time?

Splitting into even and odd components can be performed repeatedly (recursively) until $N/2 - 1 = 1$.

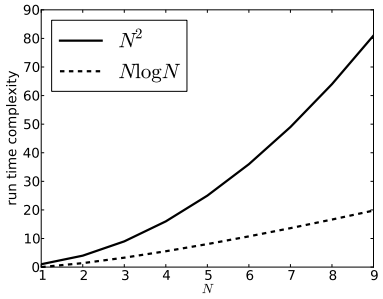
```
recursiveFFT(x, X, N, s)
  if N = 1
    XRe[0] = xRe[0], XIm[0] = xIm[0]
  else
    N2 = N/2
    recursiveFFT(x, X, N2, 2s) //even
    recursiveFFT(x + s, X + s, N2, 2s) //odd
    a = 2 × π/N
    for f = 0 to N2 − 1
      T = exp(−j × f × a) //twiddle
      even = X[f], odd = X[f + s]
      X[f] = even + T × odd
      X[f + N2] = even − T × odd
    end for
  end if
```

- Decimation in Time (DIT) principle:
  - Proposed by Cooley and Tukey in 1965.
- Single for loop.
- However this is a recursive function...

Notes

So what is the running time?

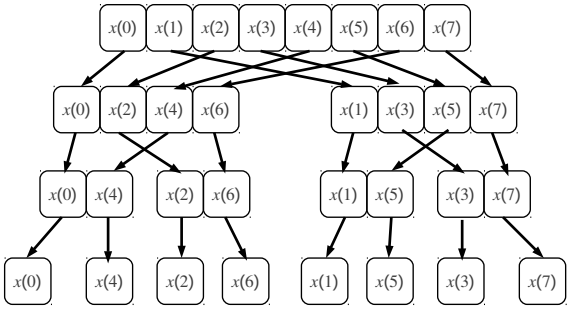
- Total running time is approximately $T(N) \propto NlogN$
- This is much faster than `simpleDFT` algorithm with N^2.



Notes

Any Further Improvements?
Remove Recursion

Recursion tree for input sequence $x(t)$ with 8 samples:



- Recursion not good for computer systems with small stacks.
- Stack overflow may occur on many levels (4 levels here).

Notes

Removing the Recursion

Observe the sequence in the bottom level of the recursion tree:

t (base 10)	0	4	2	6	1	5	3	7
t (base 2)	000	100	010	110	001	101	011	111

Reversing the bits produces numbers in a natural order:

t (base 2, bit reversed)	000	001	010	011	100	101	110	111
t (base 10, bit reversed)	0	1	2	3	4	5	6	7

This information can be used to *simulate* the recursion.

- The data can be sorted using indices arranged in bit reverse order.
- An algorithm can then:
  - Start from the bottom of the tree and iterate up
  - *But without having to perform recursion.*

Notes

Recursionless FFT

```
fft(x, X, N)
    Copy data from x to X
    SwapDataUsingBitReversedIndices(X)
    mmax = 2, n = N << 1
    //calculate transforms length 2,4,8,...,N
    while (n > mmax) //loops log2N times
        istep = mmax << 1
        initialise T //(twiddle factor)
        for m = 1 to mmax in steps 2
            for i = m to n in steps istep
                j = i + mmax
                X[i] = X[i] + T × X[j]
                X[j] = X[i] − T × X[j]
            end for
            update T //using trigonometric recurrences
        end for
        mmax = istep
    end while
```

This algorithm has the same run time complexity of $N \log N$.

Notes

Summary

- Fourier Analysis is made practical using computer algorithms, particularly efficient computer algorithms such as the Fast Fourier Transform (FFT).
- Better algorithms can be designed with insight of the underlying mathematics.

More information can be found in the following books:

- "Numerical Recipes", by Press, Teukolsky, Vetterling and Flannery, Cambridge University Press, 2007.
- "Introductory Digital Signal Processing with Computer Applications", 2nd Edition, by Lynn and Fuerst, Wiley 1998.

Notes

Notes