

Neural Networks

PHYS 250 (Autumn 2018) – Lecture 15

David Miller

Department of Physics and the Enrico Fermi Institute
University of Chicago

November 27, 2018

Outline

1

Reminders

- Reminders from Lecture 13

2

DFT to FFT

- Reminders of the DFT
- Cooley-Tukey algorithm
- Butterfly calculations
- Danielson-Lanczos Lemma

Reminders from last time

We left off discussing details of our discrete Fourier transform and how we might speed it up.

PDEs and Fourier Series

- **Fourier Series → Fourier Transforms**

- We discussed how we can move to a continuous function definition of the expansion over a basis of functions
- We then broke this down into discrete steps and obtained the **Discrete Fourier Transform**

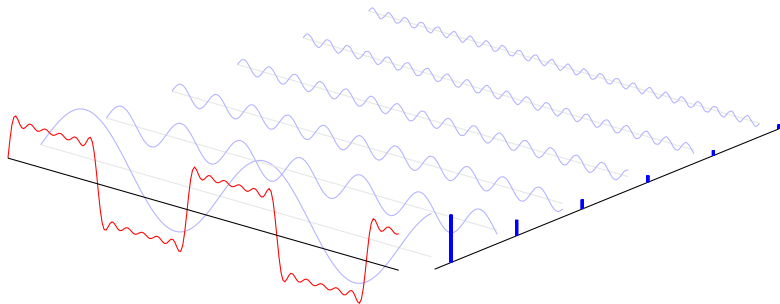
- **Issues encountered:**

- We realized that there is an issue related to the finite sampling of a function: **aliasing**
- Began to break down the Fourier transform even further for a **fast** implementation

Today we will discuss the evolution towards the **FFT**, some of the practical limitations, and specific real-world (scientific and otherwise!) examples of using FFT's!

Square wave Fourier series

We already saw how we can break down a “simple” function into its components:



So let's figure out how to use this to its full capacity!

Outline

1 *Reminders*

- Reminders from Lecture 13

2 *DFT to FFT*

- Reminders of the DFT
- Cooley-Tukey algorithm
- Butterfly calculations
- Danielson-Lanczos Lemma

Reminders of the Discrete Fourier Transform (DFT)

We determined the form of the **DFT** last time:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \quad \xrightarrow{\text{DFT}} \quad \hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k e^{-2\pi i k n / N} \quad (1)$$

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega t} d\omega \quad \xrightarrow{\text{DFT}} \quad f_k = \frac{\sqrt{2\pi}}{N} \sum_{n=1}^N \hat{f}_n e^{-i\omega_n t} \quad (2)$$

This has certain drawbacks which we will discuss shortly, but it also has huge advantages. Namely, we can re-write this to see some amazing computational properties.

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N Z_N^{nk} f_k \quad (Z_N = e^{-2\pi i / N}) \quad (3)$$

$$f_k = \frac{\sqrt{2\pi}}{N} \sum_{n=1}^N Z_N^{-nk} \hat{f}_n \quad (n = 0, 1, \dots, N) \quad (4)$$

*Moving to a **fast** DFT, aka Fast Fourier Transform*

As we mentioned last time, with this formulation, the computer needs to compute only powers of $Z \rightarrow Z_N^{nk}$.

What does this buy us, though???

*Moving to a **fast** DFT, aka Fast Fourier Transform*

As we mentioned last time, with this formulation, the computer needs to compute only powers of $Z \rightarrow Z_N^{nk}$.

What does this buy us, though???

Well, evaluating Eqs. 3–4 definition directly requires $\mathcal{O}(N^2)$ operations: there are N outputs f_k , and each output requires a sum of N terms.

*Moving to a **fast** DFT, aka Fast Fourier Transform*

As we mentioned last time, with this formulation, the computer needs to compute only powers of $Z \rightarrow Z_N^{nk}$.

What does this buy us, though???

Well, evaluating Eqs. 3–4 definition directly requires $\mathcal{O}(N^2)$ operations: there are N outputs f_k , and each output requires a sum of N terms.

What if we can make this scales as $N \ln N$???

*Moving to a **fast** DFT, aka Fast Fourier Transform*

As we mentioned last time, with this formulation, the computer needs to compute only powers of $Z \rightarrow Z_N^{nk}$.

What does this buy us, though???

Well, evaluating Eqs. 3–4 definition directly requires $\mathcal{O}(N^2)$ operations: there are N outputs f_k , and each output requires a sum of N terms.

What if we can make this scales as $N \ln N$???

This may not seem like much of a difference, for $N = 10^{2-3}$, the difference of 10^{3-5} is the difference between a minute and a week.

This is what the FFT buys us!

Cooley-Tukey algorithm

Let's start with the simplified form of the DFT:

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k e^{-2\pi i k n / N} = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k Z_N^{nk} \quad (5)$$

First, **STOP AND LOOK**. What are some features of this expression?
What operations are required?

- There are **imaginary components**

Even if the signal elements f_k to be transformed are real, Z_N is always imaginary, and therefore we must process both real and imaginary parts of the signal, even if the original signal is real.

- We have to add and/or multiply N^2 times unless we break this down further

Each add and multiply takes N register values, and N^2 multiplications requires N^2 multipliers, and N^2 adds or complete adders.

Cooley-Tukey algorithm

Let's start with the simplified form of the DFT:

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k e^{-2\pi i k n / N} = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k Z_N^{nk} \quad (5)$$

First, **STOP AND LOOK**. What are some features of this expression?
What operations are required?

- There are **imaginary components**

• Even if f_k is real, $e^{-2\pi i k n / N}$ is complex. For real f_k , the real and imaginary parts of $e^{-2\pi i k n / N}$ are $\cos(2\pi k n / N)$ and $-\sin(2\pi k n / N)$, and therefore we must process both real and imaginary parts of f_k and $e^{-2\pi i k n / N}$.

- We have to add and/or multiply N^2 times unless we break this down further

• The N^2 add and/or multiply is a major problem. We can reduce the number of operations by using the periodicity of \cos and \sin and the identity of complex numbers.

Cooley-Tukey algorithm

Let's start with the simplified form of the DFT:

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k e^{-2\pi i k n / N} = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k Z_N^{nk} \quad (5)$$

First, **STOP AND LOOK**. What are some features of this expression?
What operations are required?

- There are **imaginary components**
 - Even if the signal elements f_k to be transformed are real, Z_N is always complex, and therefore we must process both real and imaginary parts when computing transforms.
- We have to add and/or multiply N^2 times unless we break this down further
 - Both n and k range over N integer values, the $(Z_N^n)^k f_k$ multiplications require N^2 multiplications and additions of complex numbers.

Cooley-Tukey algorithm

Let's start with the simplified form of the DFT:

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k e^{-2\pi i k n / N} = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k Z_N^{nk} \quad (5)$$

First, **STOP AND LOOK**. What are some features of this expression?
What operations are required?

- There are **imaginary components**
 - Even if the signal elements f_k to be transformed are real, Z_N is always complex, and therefore we must process both real and imaginary parts when computing transforms.
- We have to add and/or multiply N^2 times unless we break this down further
 - Both n and k range over N integer values, the $(Z_N^n)^k f_k$ multiplications require N^2 multiplications and additions of complex numbers.

Cooley-Tukey algorithm

Let's start with the simplified form of the DFT:

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k e^{-2\pi i k n / N} = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k Z_N^{nk} \quad (5)$$

First, **STOP AND LOOK**. What are some features of this expression?
What operations are required?

- There are **imaginary components**
 - Even if the signal elements f_k to be transformed are real, Z_N is always complex, and therefore we must process both real and imaginary parts when computing transforms.
- We have to add and/or multiply N^2 times unless we break this down further
 - Both n and k range over N integer values, the $(Z_N^n)^k f_k$ multiplications require N^2 multiplications and additions of complex numbers.

Cooley-Tukey algorithm

Let's start with the simplified form of the DFT:

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k e^{-2\pi i k n / N} = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k Z_N^{nk} \quad (5)$$

First, **STOP AND LOOK**. What are some features of this expression?
What operations are required?

- There are **imaginary components**
 - Even if the signal elements f_k to be transformed are real, Z_N is always complex, and therefore we must process both real and imaginary parts when computing transforms.
- We have to add and/or multiply N^2 times unless we break this down further
 - Both n and k range over N integer values, the $(Z_N^n)^k f_k$ multiplications require N^2 multiplications and additions of complex numbers.

Example Cooley-Tukey algorithm (I)

The time savings comes from taking advantage of **periodicity**.

The values of the **computationally expensive complex factor** $Z^{nk} = ((Z)^n)^k$ **are repeated** as the integers n and k vary sequentially.

For instance, for $N = 8$ (leaving out subscript and $1/\sqrt{2\pi}$, but note this is Z_8^{nk}):

Example Cooley-Tukey algorithm (I)

The time savings comes from taking advantage of **periodicity**.

The values of the **computationally expensive complex factor** $Z^{nk} = ((Z)^n)^k$ **are repeated** as the integers n and k vary sequentially.

For instance, for $N = 8$ (leaving out subscript and $1/\sqrt{2\pi}$, but note this is Z_8^{nk}):

$$\hat{f}^0 = Z^0 f_0 + Z^0 f_1 + Z^0 f_2 + Z^0 f_3 + Z^0 f_4 + Z^0 f_5 + Z^0 f_6 + Z^0 f_7$$

Example Cooley-Tukey algorithm (I)

The time savings comes from taking advantage of **periodicity**.

The values of the **computationally expensive complex factor** $Z^{nk} = ((Z)^n)^k$ **are repeated** as the integers n and k vary sequentially.

For instance, for $N = 8$ (leaving out subscript and $1/\sqrt{2\pi}$, but note this is Z_8^{nk}):

$$\hat{f}^0 = Z^0 f_0 + Z^0 f_1 + Z^0 f_2 + Z^0 f_3 + Z^0 f_4 + Z^0 f_5 + Z^0 f_6 + Z^0 f_7$$

$$\hat{f}^1 = Z^0 f_0 + Z^1 f_1 + Z^2 f_2 + Z^3 f_3 + Z^4 f_4 + Z^5 f_5 + Z^6 f_6 + Z^7 f_7$$

Example Cooley-Tukey algorithm (I)

The time savings comes from taking advantage of **periodicity**.

The values of the **computationally expensive complex factor** $Z^{nk} = ((Z)^n)^k$ **are repeated** as the integers n and k vary sequentially.

For instance, for $N = 8$ (leaving out subscript and $1/\sqrt{2\pi}$, but note this is Z_8^{nk}):

$$\begin{aligned}\hat{f}^0 &= Z^0 f_0 + Z^0 f_1 + Z^0 f_2 + Z^0 f_3 + Z^0 f_4 + Z^0 f_5 + Z^0 f_6 + Z^0 f_7 \\ \hat{f}^1 &= Z^0 f_0 + Z^1 f_1 + Z^2 f_2 + Z^3 f_3 + Z^4 f_4 + Z^5 f_5 + Z^6 f_6 + Z^7 f_7 \\ \hat{f}^2 &= Z^0 f_0 + Z^2 f_1 + Z^4 f_2 + Z^6 f_3 + Z^8 f_4 + Z^{10} f_5 + Z^{12} f_6 + Z^{14} f_7\end{aligned}$$

Example Cooley-Tukey algorithm (I)

The time savings comes from taking advantage of **periodicity**.

The values of the **computationally expensive complex factor** $Z^{nk} = ((Z)^n)^k$ **are repeated** as the integers n and k vary sequentially.

For instance, for $N = 8$ (leaving out subscript and $1/\sqrt{2\pi}$, but note this is Z_8^{nk}):

$$\begin{aligned}\hat{f}^0 &= Z^0 f_0 + Z^0 f_1 + Z^0 f_2 + Z^0 f_3 + Z^0 f_4 + Z^0 f_5 + Z^0 f_6 + Z^0 f_7 \\ \hat{f}^1 &= Z^0 f_0 + Z^1 f_1 + Z^2 f_2 + Z^3 f_3 + Z^4 f_4 + Z^5 f_5 + Z^6 f_6 + Z^7 f_7 \\ \hat{f}^2 &= Z^0 f_0 + Z^2 f_1 + Z^4 f_2 + Z^6 f_3 + Z^8 f_4 + Z^{10} f_5 + Z^{12} f_6 + Z^{14} f_7 \\ \hat{f}^3 &= Z^0 f_0 + Z^3 f_1 + Z^6 f_2 + Z^9 f_3 + Z^{12} f_4 + Z^{15} f_5 + Z^{18} f_6 + Z^{21} f_7 \\ \hat{f}^4 &= Z^0 f_0 + Z^4 f_1 + Z^8 f_2 + Z^{12} f_3 + Z^{16} f_4 + Z^{20} f_5 + Z^{24} f_6 + Z^{28} f_7 \\ \hat{f}^5 &= Z^0 f_0 + Z^5 f_1 + Z^{10} f_2 + Z^{15} f_3 + Z^{20} f_4 + Z^{25} f_5 + Z^{30} f_6 + Z^{35} f_7 \\ \hat{f}^6 &= Z^0 f_0 + Z^6 f_1 + Z^{12} f_2 + Z^{18} f_3 + Z^{24} f_4 + Z^{30} f_5 + Z^{36} f_6 + Z^{42} f_7 \\ \hat{f}^7 &= Z^0 f_0 + Z^7 f_1 + Z^{14} f_2 + Z^{21} f_3 + Z^{28} f_4 + Z^{35} f_5 + Z^{42} f_6 + Z^{49} f_7\end{aligned}$$

Example Cooley-Tukey algorithm (II)

There are actually only 4 independent values! Z^0, Z^1, Z^2, Z^3

$$\begin{aligned} Z^0 &= \exp(0) = +1, & Z^1 &= \exp\left(-\frac{2\pi}{8}i\right) = +\frac{\sqrt{2}}{2} - i\frac{\sqrt{2}}{2} \\ Z^2 &= \exp\left(-\frac{2\pi}{8}2i\right) = -i, & Z^3 &= \exp\left(-\frac{2\pi}{8}3i\right) = -\frac{\sqrt{2}}{2} - i\frac{\sqrt{2}}{2} \\ Z^4 &= \exp\left(-\frac{2\pi}{8}4i\right) = -Z^0, & Z^5 &= \exp\left(-\frac{2\pi}{8}5i\right) = -Z^1 \\ Z^6 &= \exp\left(-\frac{2\pi}{8}6i\right) = -Z^2, & Z^7 &= \exp\left(-\frac{2\pi}{8}7i\right) = -Z^3 \\ Z^8 &= \exp\left(-\frac{2\pi}{8}8i\right) = +Z^0, & Z^9 &= \exp\left(-\frac{2\pi}{8}9i\right) = +Z^1 \\ Z^{10} &= \exp\left(-\frac{2\pi}{8}10i\right) = +Z^2, & Z^{11} &= \exp\left(-\frac{2\pi}{8}11i\right) = +Z^3 \\ Z^{12} &= \exp\left(-\frac{2\pi}{8}11i\right) = -Z^0, & \dots & \end{aligned}$$

Example Cooley-Tukey algorithm (III)

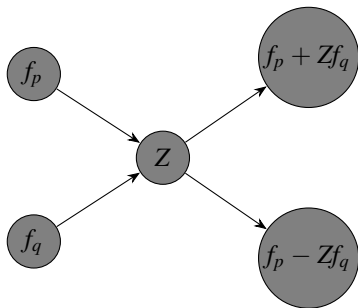
We can now put these equations in an appropriate form for computing by regrouping the terms into sums and differences of the f 's:

$$\begin{aligned}\hat{f}^0 &= Z^0(f_0 + f_4) + Z^0(f_1 + f_5) + Z^0(f_2 + f_6) + Z^0(f_3 + f_7) \\ \hat{f}^1 &= Z^0(f_0 - f_4) + Z^1(f_1 - f_5) + Z^2(f_2 - f_6) + Z^3(f_3 - f_7) \\ \hat{f}^2 &= Z^0(f_0 + f_4) + Z^2(f_1 + f_5) - Z^0(f_2 + f_6) - Z^2(f_3 + f_7) \\ \hat{f}^3 &= Z^0(f_0 - f_4) + Z^3(f_1 - f_5) - Z^2(f_2 - f_6) + Z^1(f_3 - f_7) \\ \hat{f}^4 &= Z^0(f_0 + f_4) - Z^0(f_1 + f_5) + Z^0(f_2 + f_6) - Z^0(f_3 + f_7) \\ \hat{f}^5 &= Z^0(f_0 - f_4) - Z^1(f_1 - f_5) + Z^2(f_2 - f_6) - Z^3(f_3 - f_7) \\ \hat{f}^6 &= Z^0(f_0 + f_4) - Z^2(f_1 + f_5) - Z^0(f_2 + f_6) + Z^2(f_3 + f_7) \\ \hat{f}^7 &= Z^0(f_0 - f_4) - Z^3(f_1 - f_5) - Z^2(f_2 - f_6) - Z^1(f_3 - f_7) \\ \hat{f}^8 &= \hat{f}^0.\end{aligned}\tag{6}$$

Butterfly calculations (I)

Now comes the real magic, and something that is used all over the place in fast, hardware-based calculations:

- notice the **repeating factors inside the parentheses**, they have the form $f_p \pm f_q$. These symmetries are systematized by introducing the **butterfly operation**.



Danielson-Lanczos Lemma

The discrete Fourier transform of length N (where N is even) can be rewritten as the **sum of two discrete Fourier transforms**, each of length $N/2$, one for **even-numbered** points and the other for **odd-numbered** points.

$$\hat{f}_n = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^N f_k Z_N^{nk} \quad (7)$$

$$= \frac{1}{\sqrt{2\pi}} \sum_{k=1}^{N/2} f_{2k} Z_{N/2}^{nk} + Z_N^n \sum_{k=1}^{N/2} Z_{N/2}^{nk} f_{2k+1} \quad (8)$$

$$= \hat{f}_n^{\text{even}} + Z_N^n \hat{f}_n^{\text{odd}}, \quad (9)$$

In fact, this procedure can be **applied recursively** to break up the $N/2$ even and odd points to their $N/4$ even and odd points.

If N is a power of 2, this procedure breaks up the original transform into $\ln N$ transforms of length 1.