Computational Physics – Lecture 16

(13 April 2020)

Recap

In the previous class, we understood the meaning of the Discrete Fourier Transform as a basis change in $\mathbb{C}^n$.

Recap

We ended by writing the DFT as

$$\tilde{f}(k_q) = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ik_q x_p\right), \tag{1}$$

and the inverse DFT as

$$f(x_p) = \frac{1}{\sqrt{n}} \sum_{q=0}^{n-1} \tilde{f}(k_q) \cdot \exp\left(ik_q x_p\right), \tag{2}$$

for $p = 0, 1, \ldots, n-1$ and $q = 0, 1, \ldots, n-1$, and

$$x_p = p\Delta \tag{3}$$

and

$$k_q = 2\pi q / n\Delta \tag{4}$$

Recap

That form was a rewording of the simpler form of the DFT

$$\tilde{w}_q = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} w_p \cdot \exp\left(\frac{-i2\pi qp}{n}\right), \tag{5}$$

and the inverse DFT

$$w_p = \frac{1}{\sqrt{n}} \sum_{q=0}^{n-1} \tilde{w}_q \cdot \exp\left(\frac{i2\pi qp}{n}\right), \tag{6}$$

for $p = 0, 1, \ldots, n-1$ and $q = 0, 1, \ldots, n-1$.

Today's plan

In today's class, we want to learn an algorithm called Fast Fourier Transform for computing the DFT and then we want to apply this algorithm to calculate the FT.

Complexity of the DFT algorithm

DFT is a $\mathcal{O}(n^2)$ operation.

Complexity of the DFT algorithm

DFT is a $\mathcal{O}(n^2)$ operation.

This is because calculating the matrix multiplication

$$\tilde{w}_q = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} w_p \cdot \exp\left(\frac{-i2\pi qp}{n}\right), \tag{7}$$

requires $\sim n$ multiplications plus n additions, n times.

Complexity of the DFT algorithm

DFT is a $\mathcal{O}(n^2)$ operation.

This is because calculating the matrix multiplication

$$\tilde{w}_q = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} w_p \cdot \exp\left(\frac{-i2\pi qp}{n}\right), \tag{7}$$

requires $\sim n$ multiplications plus n additions, n times.

The total number of arithmetic operations is then $\sim 2n^2$, which is $\mathcal{O}(n^2)$.

Complexity of the DFT algorithm

DFT is a $\mathcal{O}(n^2)$ operation.

This is because calculating the matrix multiplication

$$\tilde{w}_q = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} w_p \cdot \exp\left(\frac{-i2\pi qp}{n}\right), \tag{7}$$

requires $\sim n$ multiplications plus n additions, n times.

The total number of arithmetic operations is then $\sim 2n^2$, which is $\mathcal{O}(n^2)$.

It turns out that there is an $\mathcal{O}(n \log_2 n)$ method.

Complexity of the DFT algorithm

DFT is a $\mathcal{O}(n^2)$ operation.

This is because calculating the matrix multiplication

$$\tilde{w}_q = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} w_p \cdot \exp\left(\frac{-i2\pi qp}{n}\right), \tag{7}$$

requires $\sim n$ multiplications plus n additions, n times.

The total number of arithmetic operations is then $\sim 2n^2$, which is $\mathcal{O}(n^2)$.

It turns out that there is an $\mathcal{O}(n \log_2 n)$ method.

That is a big improvement: For a billion numbers, if an $\mathcal{O}(n \log_2 n)$ method takes a second, $\mathcal{O}(n^2)$ will need a year.

Fast Fourier Transform

Let us consider the special case in which $n = 2^m$. That is, for example, $n = 32$ or $n = 512$ or $n = 1024$.

Fast Fourier Transform

Let us consider the special case in which $n = 2^m$. That is, for example, $n = 32$ or $n = 512$ or $n = 1024$.

Now we can write

$$\tilde{w}_q = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} w_p \cdot \exp\left(\frac{-i2\pi qp}{n}\right) \tag{8}$$

$$\begin{aligned} &= \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p} \cdot \exp\left(\frac{-i2\pi q \cdot (2p)}{n}\right) \\ &+ \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p+1} \cdot \exp\left(\frac{-i2\pi q \cdot (2p+1)}{n}\right) \end{aligned} \tag{9}$$

Fast Fourier Transform

$$
\begin{aligned}
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p} \cdot \exp\left(\frac{-i2\pi q \cdot (2p)}{n}\right) \\
&+ \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p+1} \cdot \exp\left(\frac{-i2\pi q \cdot (2p+1)}{n}\right) \qquad (10) \\
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p} \cdot \exp\left(\frac{-i2\pi q \cdot p}{n/2}\right) \\
&+ \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p+1} \cdot \exp\left(\frac{-i2\pi q \cdot (2p+1)}{n}\right) \qquad (11)
\end{aligned}
$$

Fast Fourier Transform

$$= \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p} \cdot \exp\left(\frac{-i2\pi q \cdot p}{n/2}\right)$$

$$+ \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p+1} \cdot \exp\left(\frac{-i2\pi q \cdot (2p+1)}{n}\right) \tag{12}$$

$$= \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p} \cdot \exp\left(\frac{-i2\pi q \cdot p}{n/2}\right)$$

$$+ \frac{1}{\sqrt{n}} \exp\left(\frac{-i2\pi q}{n}\right) \sum_{p=0}^{n/2-1} w_{2p+1} \cdot \exp\left(\frac{-i2\pi q \cdot p}{n/2}\right) \tag{13}$$

Fast Fourier Transform

So our DFT is

$$
\begin{aligned}
\tilde{w}_q &= \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} w_p \cdot \exp\left(\frac{-i2\pi qp}{n}\right) && (14) \\
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n/2-1} w_{2p} \cdot \exp\left(\frac{-i2\pi q \cdot p}{n/2}\right) \\
&+ \frac{1}{\sqrt{n}} \exp\left(\frac{-i2\pi q}{n}\right) \sum_{p=0}^{n/2-1} w_{2p+1} \cdot \exp\left(\frac{-i2\pi q \cdot p}{n/2}\right) && (15) \\
&= F_q^e + W^q \cdot F_q^o, && (16)
\end{aligned}
$$

where F_k^e and F_k^o are DFTs of $n/2$ numbers, and W is the number $\exp(-i2\pi/n)$.

Fast Fourier Transform

We can now break F_k^e, which is the DFT of $n/2$ numbers, into F_k^{ee} and F_k^{eo}, which are DFTs of $n/4$ numbers.

Fast Fourier Transform

We can now break F_k^e, which is the DFT of $n/2$ numbers, into F_k^{ee} and F_k^{eo}, which are DFTs of $n/4$ numbers.

Similarly, we can also break F_k^o into F_k^{oe} and F_k^{oo}.

Fast Fourier Transform

We can now break F_k^e, which is the DFT of $n/2$ numbers, into F_k^{ee} and F_k^{eo}, which are DFTs of $n/4$ numbers.

Similarly, we can also break F_k^o into F_k^{oe} and F_k^{oo}.

In other words, we can apply our scheme recursively.

Fast Fourier Transform

We can now break F_k^e, which is the DFT of $n/2$ numbers, into F_k^{ee} and F_k^{eo}, which are DFTs of $n/4$ numbers.

Similarly, we can also break F_k^o into F_k^{oe} and F_k^{oo}.

In other words, we can apply our scheme recursively.

Where will the recursion stop? It will stop at the step where we have to do DFTs of single numbers.

Fast Fourier Transform

We can now break F_k^e, which is the DFT of $n/2$ numbers, into F_k^{ee} and F_k^{eo}, which are DFTs of $n/4$ numbers.

Similarly, we can also break F_k^o into F_k^{oe} and F_k^{oo}.

In other words, we can apply our scheme recursively.

Where will the recursion stop? It will stop at the step where we have to do DFTs of single numbers.

But the DFT of a single number is the number itself, because

$$\tilde{w}_0 = \frac{1}{\sqrt{1}} \sum_{p=0}^{0} w_p \cdot \exp\left(\frac{-i2\pi \cdot 0 \cdot p}{1}\right) = w_0. \tag{17}$$

Fast Fourier Transform

Now to get the DFT of the original n numbers, we start with these single-number DFTs and work backwards.

Fast Fourier Transform

Now to get the DFT of the original n numbers, we start with these single-number DFTs and work backwards.

At each step, we have $\mathcal{O}(n)$ operations to do at each steps and there $\log_2 n$ steps. So the whole algorithm is $\mathcal{O}(n \log_2 n)$.

Fast Fourier Transform

Now to get the DFT of the original n numbers, we start with these single-number DFTs and work backwards.

At each step, we have $\mathcal{O}(n)$ operations to do at each steps and there $\log_2 n$ steps. So the whole algorithm is $\mathcal{O}(n \log_2 n)$.

The same algorithm can also be used to calculate the inverse DFT by simply changing the sign in the exponent.

Fast Fourier Transform

Now to get the DFT of the original n numbers, we start with these single-number DFTs and work backwards.

At each step, we have $\mathcal{O}(n)$ operations to do at each steps and there $\log_2 n$ steps. So the whole algorithm is $\mathcal{O}(n \log_2 n)$.

The same algorithm can also be used to calculate the inverse DFT by simply changing the sign in the exponent.

Libraries like `numpy.fft`, GSL, and `FFTW` all use the FFT algorithm.

Fast Fourier Transform

What if $n \neq 2^m$?

Fast Fourier Transform

What if $n \neq 2^m$?

In that case we divide the problem by prime factors of n instead of factors of 2.

Fast Fourier Transform

What if $n \neq 2^m$?

In that case we divide the problem by prime factors of n instead of factors of 2.

If these prime factors are small, we get an almost $\mathcal{O}(n \log_2 n)$ computation. Otherwise, we come closer the $\mathcal{O}(n^2)$ computation.

Fast Fourier Transform

What if $n \neq 2^m$?

In that case we divide the problem by prime factors of n instead of factors of 2.

If these prime factors are small, we get an almost $\mathcal{O}(n \log_2 n)$ computation. Otherwise, we come closer the $\mathcal{O}(n^2)$ computation.

Whenever possible, it is advisable to construct your problem so that $n = 2^m$.

FT using DFT

We now understand the definition of the DFT of n numbers.

FT using DFT

We now understand the definition of the DFT of n numbers.

We also know how to calculate the DFT efficiently.

FT using DFT

We now understand the definition of the DFT of n numbers.

We also know how to calculate the DFT efficiently.

Now, it turns out that we can use the DFT to compute a numerical approximation to the FT of a function.

FT using DFT

Suppose I have a function $f(x)$.

FT using DFT

Suppose I have a function $f(x)$.

We know that the FT of this function is the infinite integral

$$\tilde{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp(-ikx). \tag{18}$$

FT using DFT

Suppose I have a function $f(x)$.

We know that the FT of this function is the infinite integral

$$\tilde{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp(-ikx). \tag{18}$$

Our task now is to evaluate this integral numerically.

FT using DFT

We begin by considering the n numbers

$$f(x_i) \quad \text{for } i = 0, \ldots, n-1. \tag{19}$$

where the x_i are evenly spaced *sample points*

$$x_i = x_{\min} + i\Delta, \tag{20}$$

for $i = 0, \ldots, n-1$, where the constant Δ is called the sampling rate.

FT using DFT

We begin by considering the n numbers

$$f(x_i) \quad \text{for } i = 0, \ldots, n-1. \tag{19}$$

where the x_i are evenly spaced *sample points*

$$x_i = x_{\text{min}} + i\Delta, \tag{20}$$

for $i = 0, \ldots, n-1$, where the constant Δ is called the sampling rate.

The maximum value of x_i is therefore $x_{\text{max}} = x_{\text{min}} + (n-1)\Delta$. (We will discuss later how to choose x_{min}, x_{max}, and Δ.)

FT using DFT

We begin by considering the n numbers

$$f(x_i) \quad \text{for } i = 0, \ldots, n-1. \tag{19}$$

where the x_i are evenly spaced *sample points*

$$x_i = x_{\min} + i\Delta, \tag{20}$$

for $i = 0, \ldots, n-1$, where the constant Δ is called the sampling rate.

The maximum value of x_i is therefore $x_{\max} = x_{\min} + (n-1)\Delta$. (We will discuss later how to choose $x_{\min}$, $x_{\max}$, and Δ.)

The process of obtaining $f(x_i)$ is called *sampling*. The set of $f(x_i)$ is called a *sample* of the function $f(x)$.

FT using DFT

Now we can write

$$\tilde{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp\left(-ikx\right) \tag{21}$$

$$= \frac{1}{\sqrt{2\pi}} \sum_{p=0}^{n-1} \Delta \cdot f(x_p) \cdot \exp\left(-ikx_p\right) \tag{22}$$

$$= \Delta \cdot \frac{1}{\sqrt{2\pi}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ikx_p\right), \tag{23}$$

where we have replaced the integral by an approximate Riemann sum in Equation (22), and taken the constant Δ out of the summation in Equation (23).

FT using DFT

Now, we know that given the n numbers $f(x_p)$, $p = 0, \ldots, n-1$, the Discrete Fourier transform of these n numbers is given by

$$\tilde{f}(k_q) = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ik_q x_p\right), \tag{24}$$

if $x_p = p\Delta$ and $k_q = 2\pi q/n\Delta$ for a constant Δ and $q = 0, 1, \ldots, n-1$.

FT using DFT

Now, we know that given the n numbers $f(x_p)$, $p = 0, \ldots, n-1$, the Discrete Fourier transform of these n numbers is given by

$$\tilde{f}(k_q) = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ik_q x_p\right), \tag{24}$$

if $x_p = p\Delta$ and $k_q = 2\pi q/n\Delta$ for a constant Δ and $q = 0, 1, \ldots, n-1$.

What if $x_p = x_{\min} + p\Delta$ as we have in our case?

FT using DFT

Well, in this case, we notice that

$$
\begin{aligned}
&\frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ik_q x_p\right) \\
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_{\min} + p\Delta) \cdot \exp\left(-ik_q \cdot [x_{\min} + p\Delta]\right) \qquad (25) \\
&= \exp\left(-ik_q \cdot x_{\min}\right) \\
&\qquad \cdot \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_{\min} + p\Delta) \cdot \exp\left(-ik_q \cdot p\Delta\right). \qquad (26)
\end{aligned}
$$

FT using DFT

Well, in this case, we notice that

$$
\begin{aligned}
&\frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-i k_q x_p\right) \\
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_{\min} + p\Delta) \cdot \exp\left(-i k_q \cdot [x_{\min} + p\Delta]\right) && (25) \\
&= \exp\left(-i k_q \cdot x_{\min}\right) \\
&\qquad \cdot \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_{\min} + p\Delta) \cdot \exp\left(-i k_q \cdot p\Delta\right). && (26)
\end{aligned}
$$

So if the sample points are of the form $x_{\min} + p\Delta$, just use the old DFT formula, which we had derived for sample points of the form $p\Delta$, but multiply the result with the phase factor $\exp\left(-i k_q \cdot x_{\min}\right)$

FT using DFT

So now we can go back and write the Fourier transform as

$$\tilde{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp(-ikx) \tag{27}$$

$$= \Delta \cdot \frac{1}{\sqrt{2\pi}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp(-ikx_p) \tag{28}$$

$$= \Delta \cdot \sqrt{\frac{n}{2\pi}} \cdot \exp(-ik_q \cdot x_{\min}) \cdot \mathrm{DFT}[f(x_p)], \tag{29}$$

where $\mathrm{DFT}[f(x_p)]$ is the result of using the standard DFT formula from with the n numbers $f(x_p)$.

FT using DFT

So now we can go back and write the Fourier transform as

$$\tilde{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp\left(-ikx\right) \tag{27}$$

$$= \Delta \cdot \frac{1}{\sqrt{2\pi}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ikx_p\right) \tag{28}$$

$$= \Delta \cdot \sqrt{\frac{n}{2\pi}} \cdot \exp\left(-ik_q \cdot x_{\min}\right) \cdot \mathrm{DFT}\left[f(x_p)\right], \tag{29}$$

where $\mathrm{DFT}[f(x_p)]$ is the result of using the standard DFT formula from with the n numbers $f(x_p)$.

This now enables you to numerically calculate the FT of an arbitrary function.

FT using DFT

Note that we have ended up computing a sample of the FT.

FT using DFT

Note that we have ended up computing a sample of the FT.

The frequencies are

$$k_0 = 0, k_1 = \frac{2\pi}{n\Delta}, k_2 = \frac{4\pi}{n\Delta}, \ldots, k_{n-1} = \frac{2\pi(n-1)}{n\Delta}. \tag{30}$$

FT using DFT

Note that we have ended up computing a sample of the FT.

The frequencies are

$$k_0 = 0, k_1 = \frac{2\pi}{n\Delta}, k_2 = \frac{4\pi}{n\Delta}, \ldots, k_{n-1} = \frac{2\pi(n-1)}{n\Delta}. \tag{30}$$

The quantity

$$f_n = \frac{1}{2\Delta} \tag{31}$$

turns out to be important. It is called the *Nyquist frequency*.

FT using DFT

Note that we have ended up computing a sample of the FT.

The frequencies are

$$k_0 = 0, k_1 = \frac{2\pi}{n\Delta}, k_2 = \frac{4\pi}{n\Delta}, \ldots, k_{n-1} = \frac{2\pi(n-1)}{n\Delta}. \tag{30}$$

The quantity

$$f_n = \frac{1}{2\Delta} \tag{31}$$

turns out to be important. It is called the *Nyquist frequency*.

So we can write

$$k_0 = 0, k_1 = \frac{4\pi f_n}{n}, k_2 = \frac{8\pi f_n}{n}, \ldots, k_{n-1} = \frac{4\pi(n-1) f_n}{n}. \tag{32}$$