Ch 8.5, 8.6: Trigonometric Interpolation and the FFT

Tuesday, October 28, 2025 1:19 PM (This is long – instructor may skip some at their discretion)

following the book at a high level but not at the details

(these notes are about 50% from E. Corona's typed notes, 50% mine. sources, eg. myself or web)

Consider approximating a function $f(x)$ on the domain $[0, 2\pi]$

We'll write it as a Fourier Series

$$f(x) = a_0 + \sum_{k=1}^{\infty} a_k \cdot \cos(kx) + b_k \sin(kx)$$

↳ ∞ sum, so this is really a limit. Does it converge? In what sense? Classic questions but beyond the scope of our class. Suffice it to say, f continuous and periodic isn't enough to guarantee pointwise convergence, but does guarantee convergence in a L^2 sense.

Our basis is $\{\cos(0 \cdot x)\} \cup \{\cos(kx), \sin(kx)\}_{k=1}^{\infty}$

which is orthogonal! (w.r.t. the L^2 inner product we discussed)

$$\int_0^{2\pi} \cos(kx)\cos(jx)\,dx = 0 \text{ if } k \neq j$$

$$\int_0^{2\pi} \cos(kx)\sin(jx)\,dx = 0, \text{ etc.}$$

So to find the Fourier Coefficients a_k, b_k we have a diagonal Gram matrix G, and

$$\int_0^{2\pi} \cos^2(kx)\,dx = \frac{\sin(2kx)}{4k} + \frac{x}{2}\Big|_0^{2\pi} = \pi$$

(or $= 2\pi$ if $k=0$)

likewise $\int_0^{2\pi} \sin^2(kx)\,dx = \pi \quad (k \neq 0)$

$$a_k = \frac{\langle f, \cos(kx)\rangle_{L^2}}{\|\cos(kx)\|_{L^2}^2} = \frac{1}{\pi}\int_0^{2\pi} f(x)\cdot\cos(kx)\,dx \qquad k = 1, 2, 3, \ldots$$

$$\left(a_0 = \frac{\langle f, 1\rangle}{\|1\|^2} = \frac{1}{2\pi}\int_0^{2\pi} f(x)\,dx\right)$$

$$b_k = \frac{\langle f, \sin(kx)\rangle}{\|\sin(kx)\|^2} = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)\,dx$$

For computational work, we'll want to truncate this infinite series

$$f(x) \approx p(x) = a_0 + \sum_{k=1}^{n} a_k \cos(kx) + b_k \cdot \sin(kx)$$

so $2n+1$ terms

a "trigonometric polynomial" i.e. a finite number of sines & cosines

Does f need to be periodic?

No, but it helps: the smoother f is, the faster its Fourier coefficients decay with k, so we can use a smaller n.

Tuesday, October 28, 2025 2:12 PM

ex: if $\int_0^{2\pi} f(x)^2 dx < \infty$ then $\sum a_k^2 + b_k^2 < \infty$

so $a_k, b_k \to 0$

but if also f is continuously differentiable then

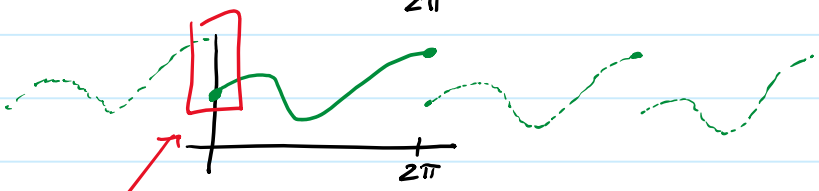
$\sum k^2(a_k^2 + b_k^2) \to 0$

so $a_k, b_k \to 0$ quickly to compensate for growth of k^2

How does this relate to periodicity? Because we consider the smoothness of the periodic extension of f

Ex:

$f(x)$ is periodic



f not periodic

so its periodic extension isn't continuous

so a_k, b_k decay slowly

at points of jump discontinuities the Fourier series shows large oscillations of about constant overshoot/undershoot but shrinking width known as the Gibbs phenomenon

Q: what if f is periodic but on $[a,b]$ not $[0,2\pi]$

A: shift/scale to $[0,2\pi]$

Tuesday, October 28, 2025 2:21 PM

Define $\mathcal{T}_n$ to be the set of all trigonometric polynomials of the form

$$p(x) = a_0 + \sum_{k=1}^{n} a_k \cos(kx) + b_k \cdot \sin(kx)$$

We defined $\{a_k, b_k\}$ for L^2 approximation of f

but we'll also see we can do interpolation of $\{x_j, y_j\}_{j=0}^{2n}$, $x_j \in [0, 2\pi)$ and often $y_j = f(x_j)$

First let's get nicer notation using complex numbers

Euler's Formula: $e^{iy} = \cos(y) + i \cdot \sin(y)$, $i = \sqrt{-1}$

(good idea for a tattoo) so (set $y = k \cdot x$) $e^{ikx} = \cos(kx) + i \cdot \sin(kx)$

i.e. $\cos(kx) = \frac{1}{2}\left(e^{ikx} + e^{-ikx}\right)$, $\sin(kx) = \frac{1}{2i}\left(e^{ikx} - e^{-ikx}\right)$

$$p(x) = a_0 + \sum_{k=1}^{n} a_k \cdot \frac{1}{2}\left(e^{ikx} + e^{-ikx}\right) + b_k \cdot \frac{1}{2i}\left(e^{ikx} - e^{-ikx}\right) \qquad \text{note } \frac{1}{2i} = \frac{-i}{2}$$

$$= \underbrace{a_0}_{c_0} + \sum_{k=1}^{n} \underbrace{\tfrac{1}{2}(a_k - ib_k)}_{c_k} e^{ikx} + \underbrace{\tfrac{1}{2}(a_k + ib_k)}_{c_{-k}} e^{-ikx}$$

$$= \sum_{k=-n}^{n} c_k e^{ikx} \quad \text{but allow } c_k \in \mathbb{C} \text{ (and } c_k = \overline{c_{-k}} \text{ (complex conjugate) including } c_0 = \overline{c_0} \text{ i.e. } c_0 \in \mathbb{R})$$

Write $\boxed{z = e^{ix} \in \mathbb{C}}$

then $p(z) = \sum_{k=-n}^{n} c_k z^k$ (if we wanted, define $P(z) = z^n p(z)$ so $P(z) = \sum_{k=0}^{2n} \tilde{c}_k z^k$, $\tilde{c}_k = c_{k-n}$ is an actual polynomial)

Now let's interpolate

Data $\{x_0, x_1, \ldots, x_{2n}\} \subseteq [0, 2\pi)$ and $y_j = f(x_j)$. Define $z_j = e^{ix_j}$

So want

$$p(z_j) = y_j \quad \text{or} \quad \underbrace{z_j^n p(z_j)}_{P(z_j)} = z_j^n \cdot y_j, \quad P(z_j) = \sum_{k=0}^{2n} \tilde{c}_k z_j^k$$

i.e. solve the linear system

$$\underbrace{\begin{bmatrix} 1 & z_0 & z_0^2 & \cdots & z_0^{2n} \\ 1 & z_1 & z_1^2 & \cdots & z_1^{2n} \\ \vdots & & & & \\ 1 & z_{2n} & z_{2n}^2 & \cdots & z_{2n}^{2n} \end{bmatrix}}_{V} \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \vdots \\ \tilde{c}_{2n} \end{bmatrix} = \begin{bmatrix} z_0^n \cdot y_0 \\ z_1^n \cdot y_1 \\ \vdots \\ z_{2n}^n \cdot y_{2n} \end{bmatrix} \qquad V \cdot \vec{c} = \vec{f}$$

V, a Vandermonde matrix... but a very nice one!

Tuesday, October 28, 2025 5:45 PM

Supposing $\{x_j\}_{j=0}^{2n}$ are equispaced on $[0, 2\pi)$, $x_j = j \cdot \frac{2\pi}{2n+1}$

Ex: $n=2$, so we have $2n+1 = 5$ nodes

then V is very nice:

$$z_j = e^{i \cdot x_j} = \left(e^{i \cdot \frac{2\pi}{2n+1}}\right)^j$$

ω (Greek lowercase omega, not double-u)

So $z_j^{\,k} = (\omega^j)^k = \omega^{jk}$ and $V = \begin{bmatrix} 1 & \omega^0 & \omega^{2\cdot 0} & \cdots & \omega^{2n\cdot 0} \\ 1 & \omega^1 & \omega^2 & \cdots & \omega^{2n} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{4n} \\ \vdots & & & & \\ 1 & \omega^{2n} & \omega^{4n} & \cdots & \omega^{4n^2} \end{bmatrix}$ $\quad V_{k,j} = \omega^{k \cdot j}$

which has orthogonal columns:

if columns denoted $\vec{v}_k$, then

$$\langle \vec{v}_k, \vec{v}_l \rangle := \text{conj}(\vec{v}_k)^T \cdot \vec{v}_l = \sum_{j=0}^{2n} \overline{\omega^{kj}} \cdot \omega^{lj} = \sum_{j=0}^{2n} \omega^{j(l-k)}$$

for $\mathbb{C}$ numbers, we need our inner product to be conjugate linear in an entry. APPM convention is 1st entry.

$$= \sum_{j=0}^{2n} \left[\omega^{(l-k)}\right]^j$$

Use $1 + r + r^2 + r^3 + r^5 = \frac{1-r^6}{1-r}$ if $r \neq 1$

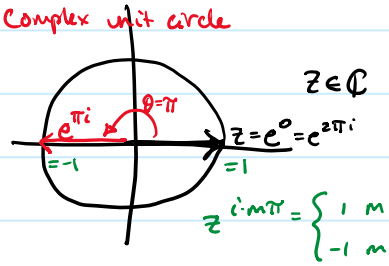
$$= \begin{cases} 2n+1 & \text{if } l-k = 0 \\ \dfrac{1-(\omega^{(l-k)})^{2n+1}}{1-\omega^{(l-k)}} & \text{if } l-k \neq 0 \end{cases}$$

$1 - \left[\omega^{(l-k)}\right]^{2n+1} = 1 - \left[\omega^{(2n+1)}\right]^{l-k}$

$\omega^{2n+1} = \left(e^{i\frac{2\pi}{2n+1}}\right)^{2n+1} = e^{2\pi i} = 1$

$$= \begin{cases} 2n+1 & l = k \\ 0 & l \neq k \end{cases}$$

Complex unit circle



$z^{i \cdot m\pi} = \begin{cases} 1 & m \text{ even} \\ -1 & m \text{ odd} \end{cases}$

So $\frac{1}{\sqrt{2n+1}} \cdot V =: W$ has "orthonormal columns"

i.e. it's a unitary matrix ($\Rightarrow$ orthonormal rows too)

"$:=$" is used in math for a definition.

$W^{-1} = W^* := \text{conj}(W^T)$ ← definition of "conjugate transpose", or "adjoint"

and $\boxed{V^{-1} = \frac{1}{2n+1} \cdot V^*}$ So solving $V\vec{c} = \vec{f}$ is as easy as $\vec{c} = V^* \vec{f}$, fast and well-conditioned.

that gives the formula

$$\tilde{c}_k = \frac{1}{2n+1} \sum_{j=0}^{2n} z_j^{-k} \cdot (z_j^{\,n} y_j) = \frac{1}{2n+1} \sum_{j=0}^{2n} e^{i\left(\frac{2\pi j}{2n+1}\right)^{n-k}} y_j$$

Warning: in physics, they write $W^\dagger$ not W^* "dagger" In applied math that would be confused with our + symbol for the pseudoinverse

Tuesday, October 28, 2025 7:41 PM

$$\tilde{c}_k = \frac{1}{2n+1}\sum_{j=0}^{2n} z_j^{-k}\cdot(z_j^n y_j) = \frac{1}{2n+1}\sum_{j=0}^{2n} e^{i\left(\frac{2\pi j}{2n+1}\right)^{n-k}} y_j$$

and $\tilde{c}_k = c_{k-n}$ so

$$c_k = \frac{1}{2n+1}\sum_{j=0}^{2n}\left(e^{i\frac{2\pi j}{2n+1}}\right)^{-k}\cdot y_j$$

or if we define the Discrete Fourier Transform as the linear operator $F:\mathbb{C}^N\to\mathbb{C}^N$ with matrix representation

$$F_{kj} = \left(e^{i\cdot\frac{2\pi j}{N}}\right)^{-k} \quad \text{then} \quad c_k = \frac{1}{N}\sum_{j=0}^{N} F_{k,j}\,y_j$$

$(F^{-1})_{kj} = \frac{1}{N}\cdot\overline{F}_{j,k}$ inverse DFT

In Fourier analysis, sometimes $-n, -n+1, \dots, -1, 0, 1, \dots, n$ is nicer (good symmetry...)

but othertimes $0, 1, \dots, 2n$ is nicer (on a computer, this is a standard "for" loop)

So Matlab and Python have utilities to permute: fftshift and ifftshift (undoes fftshift)

as well as 2D versions

In our notation, $V = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{2n} \\ \vdots & \omega^2 & \omega^4 & & \\ 1 & \omega^{2n} & & \cdots & \omega^{4n^2} \end{pmatrix}$ $\omega = e^{i\frac{2\pi}{2n+1}}$ or $e^{i\frac{2\pi}{N}}$ a "primitive" N^{th} root of unity since $\omega^N = 1$

and $V^* = \text{const}\cdot F$

However it's more canonical to define $\omega = e^{-i\frac{2\pi}{N}}$ so that $V^*\left(=\text{const}\cdot F\right) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & & \\ \vdots & & \ddots & & \\ 1 & & & & \end{pmatrix}$

⚠ Normalization conventions for the DFT vary:

Sometimes $F_{k,j} = \frac{1}{\sqrt{N}}\cdot\left(e^{i\frac{2\pi j}{N}}\right)^{-k}$ so then $F^{-1} = F^*$ i.e. unitary

so $(F^{-1})_{k,j} = \overline{F}_{j,k}$ inverse DFT

2D DFT (supplementary... not on exams)

If $X\in\mathbb{R}^{N\times M}$ is a 2D signal, we can take it's 2D DFT by

first taking the 1D DFT of its columns, then

take the 1D DFT of the resulting rows.

If $F_1\in\mathbb{C}^{N\times N}$, $F_2\in\mathbb{C}^{M\times M}$ are 1D DFT matrices, then

$$\hat{X} = \left(F_2\left((F_1\cdot X)^T\right)\right)^T = \left(F_2\cdot X^T F_1^T\right)^T = F_1 X F_2^T$$

(If we vectorize $X\mapsto \text{col}(x)\in\mathbb{C}^{N+M}$ by stacking columns, then $\text{col}(\hat{X}) = \left(F_2\otimes F_1\right)\cdot\text{col}(X)$) ← not covered on exam

Kronecker product

Tuesday, October 28, 2025 8:04 PM

The fast algo. to compute the DCT

Fast Fourier Transform

back to Gauss, modern version to Cooley + Tukey 1960's

To compute $F \cdot \vec{v}$ for $v \in \mathbb{C}^N$, you'd expect $O(N^2)$ flops but due to its special structure, the FFT does it in $O(N \cdot \log N)$ flops.

Q: which base of the log?
A: irrelevant to big-O notation ☺

Fact: it's $O(N \cdot \log N)$ for any $N \in \mathbb{N}$ but we (like all intro courses) will assume N is a power of 2, $N = 2^L$, and do the "radix-2" FFT.

If N isn't a power of 2, it's still possible but more complicated

Software like FFTW ("Fastest Fourier Transform in the West") is good at the non power of 2 case — it chooses the best algo. for your computer.

Key recursive idea:

The N-point DFT can be done via two $N/2$ point DFTs, then combining them] cost is $2N$ (cost is $W(N/2)$ each)

Let $W(N)$ be the computational cost / flops of a N-point DFT

$$W(N) = 2 \cdot W(N/2) + 2N \quad \} \text{ recursion relationship}$$

$$= 2 \cdot (2 \cdot W(N/4) + 2 \cdot N/2) + 2N = 4\,W(N/4) + 4N$$

$$= 4 \cdot (2 \cdot W(N/8) + 2 \cdot N/4) + 4N = 8\,W(N/8) + 6N$$

$$\ldots = 16\,W(N/16) + 8N$$

$$= 2^L \cdot \underbrace{W(N/2^L)}_{=O(1)} + 2 \cdot L \cdot N = N + 2\log_2(N) \cdot N = O(N \log N)$$

Details:

let $\omega = e^{-i\frac{2\pi}{N}}$ and we want $c_k = \sum_{j=0}^{N-1} y_j \cdot \omega^{jk}$, $k = 0, 1, \ldots, N-1$

$\omega^N = 1$

$\omega^{N/2} = -1$

$$c_k = \underbrace{\sum_{m=0}^{N/2-1} y_{2m}\, \omega^{(2m)k}}_{\text{even}} + \underbrace{\sum_{m=0}^{N/2-1} y_{2m+1}\, \omega^{(2m+1)k}}_{\text{odd}}$$

split j into even or odd: $2m$, $2m+1$

let $\omega_2 = \omega^2 = e^{-i\frac{2\pi}{N}\cdot 2} = e^{-i\frac{2\pi}{N/2}}$ ... the ω we'd use in a $N/2$ FFT!

$$c_k = \underbrace{\sum_{m=0}^{N/2-1} y_{2m}\, \omega_2^{mk}}_{E_k} + \omega^{-k} \cdot \underbrace{\sum_{m=0}^{N/2-1} y_{2m+1}\, \omega_2^{mk}}_{O_k} \quad (*) \qquad k = 0, 1, \ldots, N/2-1 \qquad \omega_2^{N/2} = 1$$

(recursion)

If $k \le N/2 - 1$, E_k and O_k come from FFT's of size $N/2$ done to the even and odd indices of $\vec{y}$

Small size $N/2$ FFTs only output $N/2$ coefficients, so this doesn't quite work if $k \ge N/2$ — i.e. the formula $(*)$ holds but can't use little $N/2$ FFTs

If $k \ge N/2$, write as $c_k = c_{k'+N/2}$ so $k' < N/2$

$$= \sum_{m=0}^{N/2-1} y_{2m}\, \omega_2^{mk'} \cdot \underbrace{\omega_2^{m(N/2)}}_{(\omega_2^{N/2})^m = 1^m = 1} + \omega^{-(k'+N/2)} \sum_{m=0}^{N/2-1} y_{2m+1}\, \omega_2^{mk'} \underbrace{\omega_2^{m(N/2)}}_{=1}$$

$$= E_{k'} + \omega^{-N/2} \cdot \omega^{k'} O_{k'} = E_{k'} - \omega^{k'} O_{k'}$$

Tuesday, October 28, 2025 8:54 PM

Still assuming $N = 2^L$ for simplicity

to summarize, let $\omega = e^{-\frac{2\pi i}{N}}$, $\vec{y} \in \mathbb{C}^N$ then $\vec{c} \in \mathbb{C}^N$ defined by

$k = 0, 1, \ldots, N/2-1$

$$c_k = E_k + \omega^k O_k \qquad \text{where } E_k = \text{FFT}([y_0, y_2, y_4, \ldots, y_{N-2}])$$

$$c_{k+N/2} = E_k - \omega^k O_k \qquad O_k = \text{FFT}([y_1, y_3, \ldots, y_{N-1}])$$

via recursion

"twiddle factors"

The FFT has revolutionized / enabled a lot of signal processing

skip in lecture

Around 2000, "Computing in Science & Engineering" (published by American Inst. of Phys. + IEEE Computing society) published an influential "Top Ten Algorithms of the Century"

- 1946 von Neumann, Stan Ulam, Nick Metropolis — Metropolis Algo / Monte Carlo
  (Ulam math 60's until '75)
- 1948 Dantzig's Simplex algo. for linear programming
- 1950 Krylov subspace methods
- 1951 Matrix decompositions formalized
- 1957 Fortran compiler, IBM
- 1959-61, QR Algo for eigenvalues
- 1962 Quicksort
- 1965 Cooley (IBM) & Tukey (Princeton) FFT
- 1977 integer relation detection
- 1987 Leslie Greengard, Vladimir Rokhlin (Yale) Fast Multipole Method "FMM"