Math 563, Spring 2020
Last updated: 2/8/20

HOMEWORK 4 (DUE WED. FEB. 12)

Reading: Some further reading is noted below:

- Review material from previous weeks (see Piazza post for the list)
- Suggested: Section 10.7 and Section 10.9 of Quarteroni. Note that 10.9.1 has the fix (Lanczos smoothing) for the Gibbs' phenomenon problem.

Code to turn in:

- None required; you can turn in the code for Q2 if completed.

1. Problems (some with, some without code)

Q1 (Hermite polynomials). Starting with $p_0(x) = 1$, derive the next three (up to deg. 3) **Hermite polynomials**, orthogonal with respect to

$$\langle f, g\rangle = \int_{-\infty}^{\infty} f(x)g(x)e^{-x^2}\,dx.$$

Then estimate the integral

$$I = \int_{-\infty}^{\infty} \sin^4(x)e^{-x^2}\,dx = 0.34270348108793888\cdots$$

using 3 points. How close is it to the exact solution? What about with five points?[1]

Q2 (discrete least squares). Let $x_j = jh$ with $h = 2\pi/N$ (for $j = 0, \cdots N-1$) be the usual nodes for the DFT.

a) Verify the discrete orthogonality relation

$$\langle e^{ijx}, e^{ikx}\rangle_d = 0 \text{ for } j \neq k$$

where $\langle f, g\rangle_d = \sum_{j=0}^{N-1} f(x_j)\overline{g(x_j)}$.

b) Suppose we know the values of f at the x_j's and seek an approximation

$$f(x) \approx p_m(x) = \sum_{k=-m}^{m} c_k e^{ikx}$$

where $m < N/2$ that minimizes the **discrete least squares error**

$$E(\vec{c}) = \sum_{j=0}^{N-1} |p_m(x_j) - f_j|^2$$

Show directly that the c_k's that solve this problem are just the DFT coefficients.

[1]You can look up the nodes/coefficients at https://dlmf.nist.gov/3.5. This site is a good resource for looking up any properties of special functions you may need.

Q3 (Parseval. discrete version). Recall that Parseval's identity says

$$\int_0^{2\pi} |f(x)|^2\,dx = 2\pi \sum_{k\in\mathbb{Z}} |c_k|^2$$

if $f \in L^2([0,2\pi])$ has a Fourier series $\sum_{k\in\mathbb{Z}} c_k e^{ikx}$ (proven by considering $\langle f, f\rangle$).

Derive a discrete analogue of this identity for the DFT, relating $\sum |f_j|^2$ to $\sum_k |F_k|^2$. Can the DFT/IDFT be defined so that the two quantities are equal?

Q4 (discrete cosine transform).
Suppose now that $f(x)$ is a function defined on $[0,\pi]$. Derive the **discrete cosine transform** (DCT) (possibly up to a factor of $1/N$)

$$F_k = \frac{1}{2}(f_0 + (-1)^k f_N) + \sum_{j=1}^{N-1} f_j \cos(\pi jk/N), \quad k = 0,\cdots,N-1$$

by the following method:

- Extend f to $[0,2\pi]$ by 'even extension' with $f(x) = f(2\pi - x)$.
- Take the DFT of f using $2N$ points ($x_j = jh$ for $j = 0,\cdots,2N-1$).
- Simplify the DFT so that it only involves values in $[0,\pi]$.

Also derive a formula for the inverse transform by simplifying the IDFT in the same way.

Remark: The discrete cosine transform is used, for instance, in data compression by cutting off high frequencies in the transform (here the fact that it is all real numbers is a computational advantage). The JPEG compression algorithm uses this approach.

Q5 (Testing Fourier series). (AC) Implement the discrete Fourier transform using the direct formula (not the FFT) (note: complex arithmetic is built into python and numpy!).

Check it by computing the real valued least-squares approximations for $f(x) = 1/(2 + e^{\sin x})$ and plotting the max. error between f and the number of terms N. *Hint: compute the DFT coefficients for a large number of points once; then you can construct all the approximations from this one set of coefficients.*

Bonus (no credit, but a good exercise): Check your calculations against the stock fft algorithm provided by numpy (beware - check the documentation for the convention).