MAT 271: Applied & Computational Harmonic Analysis Comments on Homework 2

Problem 1: It is important to notice that this shah (or comb) function is not really a function, but a *generalized function* or also known as a *distribution*. So, *you cannot use the usual Fourier transform definition*.

The best way to solve this problem is to split it into two subproblems:

(1) Show $\mathscr{F}\{\mathrm{III}_1\}(\xi) = \mathrm{III}_1(\xi)$;

(2) Apply the dilation operator $\boldsymbol{\delta}_A$ to $\mathrm{III}_1(x)$ and use the the Fourier transform formula: $\mathscr{F}\{\boldsymbol{\delta}_A f\}(\xi) = \boldsymbol{\delta}_{1/A}\mathscr{F}\{f\}(\xi)$.

As for (1), I stated this in my lecture when we discussed the sampling theorem, but this is what I really wanted you to show. The correct proof goes like this. Let us write $\mathrm{III}_1(x)$ as $\mathrm{III}(x)$. First of all, since the shah function is a generalized function, its Fourier transform is defined by pairing it with a very nice function ϕ in the Schwartz space $\mathscr{S}$ (a space of functions of infinitely many times differentiable and decaying faster than any polynomials in both space and frequency domain). With the slight abuse of notation of $\langle f, g\rangle = \int f(x)g(x)\,\mathrm{d}x$ instead of the usual $\int f(x)\overline{g(x)}\,\mathrm{d}x$, we have

$$\begin{aligned}
\langle \mathscr{F}\{\mathrm{III}\}, \phi\rangle &= \langle \mathrm{III}, \mathscr{F}\{\phi\}\rangle \\
&= \int \mathrm{III}(\xi)\widehat{\phi}(\xi)\,\mathrm{d}\xi \\
&= \sum_{k\in\mathbb{Z}} \widehat{\phi}(k) \\
&\overset{(a)}{=} \sum_{k\in\mathbb{Z}} \phi(k) \\
&= \int \mathrm{III}(\xi)\phi(\xi)\,\mathrm{d}\xi \\
&= \langle \mathrm{III}, \phi\rangle .
\end{aligned}$$

Since $\phi \in \mathscr{S}$ is arbitrary, we can conclude $\widehat{\mathrm{III}}(\xi) = \mathrm{III}(\xi)$. The equality (a) above is called the *Poisson summation formula*, and will be proved simply as follows: Consider a periodized version of $\phi(x)$, i.e.,

$$\sum_{k\in\mathbb{Z}} \phi(x-k).$$

This is clearly periodic with period 1. Thus, we can expand this into a Fourier series as:

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} \phi(x-k) &= \sum_{n\in\mathbb{Z}} c_n \mathrm{e}^{2\pi \mathrm{i} n x} \\
&= \sum_{n\in\mathbb{Z}} \left(\int_{-1/2}^{1/2} \sum_{k\in\mathbb{Z}} \phi(y-k) \mathrm{e}^{-2\pi \mathrm{i} n y} \,\mathrm{d}y \right) \mathrm{e}^{2\pi \mathrm{i} n x} \\
&= \sum_{n\in\mathbb{Z}} \left(\sum_{k\in\mathbb{Z}} \int_{-1/2}^{1/2} \phi(y-k) \mathrm{e}^{-2\pi \mathrm{i} n (y-k)} \,\mathrm{d}y \right) \mathrm{e}^{2\pi \mathrm{i} n x} \\
&= \sum_{n\in\mathbb{Z}} \left(\int_{-\infty}^{\infty} \phi(y) \mathrm{e}^{-2\pi \mathrm{i} n y} \,\mathrm{d}y \right) \mathrm{e}^{2\pi \mathrm{i} n x} \\
&= \sum_{n\in\mathbb{Z}} \widehat{\phi}(n) \mathrm{e}^{2\pi \mathrm{i} n x}.
\end{aligned}$$

Setting $x = 0$ in both sides, we get the Poisson summation formula:

$$\sum_{k\in\mathbb{Z}} \phi(k) = \sum_{k\in\mathbb{Z}} \widehat{\phi}(k).$$

Once we establish (1), i.e., $\widehat{\mathrm{III}}_1(\xi) = \mathrm{III}_1(\xi)$, it is easy to get (2) by the dilation formula. First, it is important to notice that $\delta(ax) = (1/a)\delta(x)$ or $a\delta(ax) = \delta(x)$. Now,

$$\mathrm{III}_A(x) = \sum_{k\in\mathbb{Z}} \delta(x - kA) = \sum_{k\in\mathbb{Z}} \frac{1}{A}\delta(x/A - k) = \frac{1}{\sqrt{A}} \boldsymbol{\delta}_A \mathrm{III}(x).$$

$$\begin{aligned}
\mathscr{F}\{\mathrm{III}_A\}(\xi) &= \mathscr{F}\left\{ \frac{1}{\sqrt{A}} \boldsymbol{\delta}_A \mathrm{III} \right\}(\xi) \\
&= \frac{1}{\sqrt{A}} \boldsymbol{\delta}_{1/A} \mathrm{III}(\xi) \\
&= \mathrm{III}(A\xi) \\
&= \sum_{k\in\mathbb{Z}} \delta(A\xi - k) \\
&= \sum_{k\in\mathbb{Z}} \frac{1}{A} \delta(\xi - k/A) \\
&= \frac{1}{A} \mathrm{III}_{1/A}(\xi).
\end{aligned}$$

I noticed that many of you had tried to compute the Fourier transform of the exponential function $\exp(2\pi \mathrm{i} k x/A)$ in this problem, but this does not work because such an exponential function is neither in $L^1(\mathbb{R})$ nor $L^2(\mathbb{R})$.

Problem 2: This was an easy problem. Most of you answered correctly.

Problems 3–4: (b) The point of this problem is to figure out the difference between the hand-derived Fourier series coefficients and the the DFT coefficients computed via the FFT function of MATLAB. In Part 5) here, many of you used the phrases such as "they look very similar" or "they are close". Whenever you use such words, you must be more precise and quantitative.

In order to really understand what is going on in this problem, you first need to go back to the original definition of the DFT and the Fourier coefficients.

$$
\begin{aligned}
c_k &= \int_{-\frac{1}{2}}^{\frac{1}{2}} f(x)\mathrm{e}^{-2\pi \mathrm{i} kx}\,\mathrm{d}x \\
&\approx \sum_{\ell=0}^{N-1} f(x_\ell)\mathrm{e}^{-2\pi \mathrm{i} k x_\ell}\Delta x, \quad x_\ell = -\frac{1}{2} + \ell\Delta x \text{ and } \Delta x = \frac{1}{N} \\
&= \frac{1}{N}\sum_{\ell=0}^{N-1} f\left(-\frac{1}{2}+\frac{\ell}{N}\right)\mathrm{e}^{-2\pi \mathrm{i} k(-1/2+\ell/N)} \\
&= \frac{\mathrm{e}^{-\pi \mathrm{i} k}}{N}\sum_{\ell=0}^{N-1} f\left(-\frac{1}{2}+\frac{\ell}{N}\right)\mathrm{e}^{-2\pi \mathrm{i} k\ell/N} \\
&= \frac{(-1)^k}{N}\sum_{\ell=0}^{N-1} f\left(-\frac{1}{2}+\frac{\ell}{N}\right)\mathrm{e}^{-2\pi \mathrm{i} k\ell/N} \\
&= \frac{(-1)^k}{N}\sum_{\ell=0}^{N-1} f_\ell\,\mathrm{e}^{-2\pi \mathrm{i} k\ell/N}.
\end{aligned}
$$

Also, please read the problem carefully. The interval specified was $[-1/2, 1/2)$, not $(-1/2, 1/2]$. Hence, the discretization point should starts at $x = -1/2$, and ends at $x = 1/2 - \Delta x$, not $x = 1/2$. This makes a difference from my definition of the DFT in my lecture where I used $(-1/2, 1/2]$ (i.e., $\ell = -N/2+1 : N/2$). Hence, in this case, in order to best match the MATLAB `fft` and the hand-computed Fourier coefficients over $[-1/2, 1/2)$, you do *not* need to do:

`F=circshift(fftshift(fft(fftshift(circshift(f,1)))),-1)/sqrt(length(f));`

Instead, you can simply do: `F=fft(fftshift(f))/sqrt(length(f));` We all need to appreciate the subtlety of the discretization! See my example MATLAB script below.

Finally, the summation portion can be computed by `fft` (non-unitary original version) in MATLAB. Here is my MATLAB script for Problem 3 if we want to match the hand-computed Fourier coefficients and the MATLAB `fft` output as much as possible. I also put my codes online, so please download them and run them to see how much these two sets of coefficients agree. By the way, when you submit your figures by hardcopies, you need to be a bit more considerate for the graders/readers. If you plot data using different colors, you should submit the color hardcopy! If you submit B&W printouts, then you should not use color plots; instead, you should use different plot symbols, such as 'o', '*', '+', etc. Otherwise, your intention does not convey to the graders/readers. This is also absolutely true when you submit your journal papers!!

```
% Problem 3

% define the basic parameters.
N=1024;
a=-0.5;
b=0.5;

% Create an array of N equidistant points over [a,b).
% Trying to exclude the point x=b=0.5 from the samples.
x=linspace(a,b,N+1);
x=x(1:end-1);

% Create a function.
y = x; % In problem 4, this should be y=x.^2, of course.

% Normalize the function to have a unit L^2 norm.
ey = norm(y);
y = y/ey;

% Do the fft to approximate the Fourier series coefficients over this
% interval.  Note that we need to have 1/N here.  You need to go back
% to the original definition of the Fourier coefficients and its
% approximation by the trapezoidal rule.
% Note that fft essentially view the input data is defined over the
% interval on [0,1), instead of [-1/2,1/2).  So you need to do either
% of the following two:
% 1) Apply fftshift to the input vector before taking fft; or
% 2) Apply the complex exponential factor exp(pi*k)=(-1)^k to the output
%   of fft, which is equivalent to changing the signum of the fft results
%   alternatively as fy(2:2:N)=-fy(2:2:N), where fy=fft(y)/N.

fy = fft(y)/N;
fy(2:2:N)=-fy(2:2:N);

% Now, prepare the analytical Fourier coefficients you derived by
% hand.

c = zeros(1,N);
% c(1) = 1/12.0; % for problem 4.
for k=1:N-1
  c(k+1)=i*(-1)^k/(2*pi*k); % c(k+1)=(-1)^k/(2*(pi*k)^2); % for problem 4.
end
```

```
% Normalize the coefficients
c = c/ey;

% Now plot real and imaginary part separately using the semilog plot.

figure(1)
clf;
subplot(1,2,1);
plot(real(c(1:N/2)));
grid
hold on
plot(real(fy(1:N/2)),'r.');
title('Real Part')
hold off

subplot(1,2,2);
plot(imag(c(1:N/2)));
grid
hold on
plot(imag(fy(1:N/2)),'r.');
title('Imaginary Part')
hold off

% Let's look at the more details around the origin.
figure(2)
clf;
subplot(1,2,1);
stem(real(c(1:N/16)),'o');
grid
hold on
stem(real(fy(1:N/16)),'r.');
title('Real Part')
hold off

subplot(1,2,2);
stem(imag(c(1:N/16)),'o');
grid
hold on
stem(imag(fy(1:N/16)),'r.');
title('Imaginary Part')
hold off
```

Do the similar computation for Problem 4. You can see that they match closely, but not exact due to the approximation error by the trapezoidal rule and sampling. What happens if we increase the number of samples, e.g., to $N = 2^{15}$?

Problem 5: 1)–3) Several people use the normal distribution factor as the parameter a, i.e., $a = 1/(\sigma\sqrt{2\pi})$. But my intention was to use a as the normalization constant so that the ℓ^2 norm of the input vector becomes 1. Once you do that, then you can follow the same strategy here as above.

4) You got mixed results. In fact, it is true that the larger the value of σ (i.e., the wider the Gaussian is), the faster the the decay of its Fourier *transform* because of it is proportional to $\exp(-2\pi\sigma^2\xi^2)$ in the Fourier domain. But I was asking the decay of the Fourier *coefficients* of the Gaussian on the *finite* interval $[-1/2, 1/2)$. So, the boundary effects at $x = \pm 1/2$ becomes more prominent compared to the smoothness. The Fourier coefficient magnitudes follow more like $\exp(-2\pi\sigma^2\xi^2)$ in the low frequency region. But then, the boundary effects start dominating. This fact was obscured if you use the wrong normalization a.

5) It seems that the decay of the Fourier coefficients of the Gaussian functions are faster than that of the polynomials such as ax or ax^2, which is the case in the low frequency region. However, the periodized Gaussian over the interval $[-1/2, 1/2)$ is not a C^∞ function. It's simply a continuous function, not even C^1 function because the derivatives do not match at the boundary. Of course this derivative mismatch is less pronounced for small values of σ or extremely large σ. Hence, in the finite length DFT, the quadratic polynomial behaves similarly to the Gaussian with appropriate value of σ for the higher frequency part. Thus, the decay of the Fourier coefficients of ax^2 is slower than those of the Gaussian, but the decay curve in the high frequency range looks similar to that of the Gaussian with $\sigma = 1$.

6) The Fourier transform of the Gaussian in this case is the following using Problem 3 of HW #1:

$$\mathscr{F}\{a\mathrm{e}^{-x^2/(2\sigma^2)}\} = \mathscr{F}\{a\sqrt{2\pi}\sigma g(x;\sigma)\} = a\sqrt{2\pi}\sigma \int_{-\infty}^{\infty} g(x;\sigma)\mathrm{e}^{-2\pi\mathrm{i}\xi x}\,\mathrm{d}x = a\sqrt{2\pi}\sigma\mathrm{e}^{-2\pi^2\sigma^2\xi^2}.$$

On the other hand, using the MATLAB `fft` function, we can only approximate the Fourier coefficients of the periodized Gaussian (or mutilated Gaussian) on $[-1/2, 1/2)$:

$$c_k = \int_{-\frac{1}{2}}^{\frac{1}{2}} g(x;\sigma)\mathrm{e}^{-2\pi\mathrm{i}kx}\,\mathrm{d}x.$$

Actual output is even different from this c_k due to the error by the trapezoidal rule used to obtain FFT/DFT. Therefore, there are two errors involved here: 1) Truncation of the interval; and 2) error due to the trapezoidal rule. For more details, I strongly recommend to read [1, Chap. 6].

Problem 6: (a) Please read the problem carefully. $D_N\boldsymbol{f}$ represents the FFT operation in MATLAB `fft(f)`. Hence,

$$D_N = \sqrt{N}W_N^*$$

Note that $W_N^*\boldsymbol{f}$ is the unitary (i.e., normalized) version of the DFT.

(b), (d) If you follow my `lecture07.pdf`, you should be able to get the solution easily as follows:

$$\widetilde{W}_N^* \boldsymbol{f} = T_N^* S_N D_N S_N T_N \boldsymbol{f} / \sqrt{N}.$$

Note that $T_N^* = T_N^{\mathsf{T}} = T_N^{-1}$ whereas $S_N^* = S_N^{\mathsf{T}} = S_N^{-1} = S_N$.

References

[1] W. L. BRIGGS AND V. E. HENSON, *The DFT: An Owner's Manual for the Discrete Fourier Transform*, SIAM, Philadelphia, PA, 1995.