Problem 1

1.1 Gibbs Phenomenon

Gibbs’ phenomenon has to do with how poorly Fourier series converge in the vicinity of a jump or discontinuity of a signal $f$. This fact was pointed out by Gibbs in a letter to *Nature* (1899). (Actually Gibbs’ phenomenon was first described by the British mathematician Wilbraham (1848).) The function Gibbs considered was a sawtooth (1). Gibbs was replying to a letter by the physicist Michaelson to *Nature* (1898), in which the latter expressed himself doubtful as to the idea that a real discontinuity (in $f$) can replace a sum of continuous curves ($S_n(f)$).

To investigate Gibbs’ phenomenon, let us look at the function on the unit interval

$$f(t) = \begin{cases} t & 0 \leq t < 1/2 \\ t - 1/2 & 1/2 \leq t < 1. \end{cases}$$

(a) Calculate the Fourier coefficients $(c_k)$ of $f$.

(b) Let $S_n(f)$ be the partial sum $S_n = \sum_{|k| \leq n} c_k e^{i2\pi kt}$. Calculate the approximation error $\|f - S_n(f)\|_{L^2}$ as accurately as possible.

(c) Repeat (b) but where the partial Fourier series now corresponds to the sum over the $(2n + 1)$ largest Fourier coefficients.

(d) Consider now

$$f(t) = \begin{cases} 0 & 0 \leq t < 1/2 \\ 1 & 1/2 \leq t < 1. \end{cases}$$

Gibbs observed that in the vicinity of the jump, the partial sums always overshoot the mark by about 9%. Verify this assertion carefully setting up a numerical experiment.

(e) Repeat the last question on the sawtooth signal.

(f) *Bonus question.* Can you prove that in (d)

$$\lim_{n \rightarrow +\infty} \max S_n \approx 1.089.$$
1.2 Solution

(a) We just use the definition of the complex Fourier coefficients of a function $f$ of period $T$

$$c_k = \frac{1}{T} \int_0^T f(t)e^{-\frac{i2\pi kt}{T}} \, dt$$

In our case we have $T = 1$ and we get

$$c_k = \int_0^1 f(t)e^{-i2\pi kt} \, dt$$

$$= \int_0^{1/2} f(t)e^{-i2\pi kt} \, dt + \int_{1/2}^1 f(t)e^{-i2\pi kt} \, dt$$

$$= \int_0^{1/2} te^{-i2\pi kt} \, dt + \int_{1/2}^1 (t - \frac{1}{2})e^{-i2\pi kt} \, dt$$

After an integration by parts we will get

$$c_k = -\frac{1 + (-1)^k}{4\pi ik}, \quad c_0 = \frac{1}{4}$$

or we can write it (for $k \neq 0$)

$$c_{2k} = i \frac{1}{4\pi k}$$
$$c_{2k+1} = 0$$

(b) We just calculate the square of the norm and develop the terms, taking into consideration that both $f$ and $S_n(f)$ are real.

$$\|f - S_n(f)\|_{L^2}^2 = \int_0^1 |f - S_n(f)|^2 \, dt$$

$$= \int_0^1 (f^2 + S_n(f)^2 - 2fS_n(f)) \, dt$$

$$= \int_0^1 f^2 \, dt + \int_0^1 S_n(f)^2 \, dt - 2 \int_0^1 fS_n(f) \, dt$$

We calculate each term. The first term yields

$$\int_0^1 f^2 \, dt = \int_0^{1/2} t^2 \, dt + \int_{1/2}^1 (t - \frac{1}{2})^2 \, dt$$

$$= 2 \int_0^{1/2} t^2 \, dt, \quad \text{with a change of variables}$$

$$= \frac{2}{3} \left[ t^3 \right]_0^{1/2}$$

$$= \frac{1}{12}$$
To calculate the second term, \( \int_0^1 S_n(f)^2 \, dt \), we just have to use Parseval’s equality and we get

\[
\int_0^1 S_n(f)^2 \, dt = \sum_{|k| \leq n} |c_k|^2
\]

For the third and last term we just have to realise that \( \int_0^1 f S_n(f) \, dt = \langle f | S_n(f) \rangle \) (projection on the \( 2n + 1 \) first components) and we get

\[
\int_0^1 f S_n(f) \, dt = \sum_{|k| \leq n} |c_k|^2
\]

Regrouping the terms we get

\[
\|f - S_n(f)\|_{L^2}^2 = \frac{1}{12} - \sum_{|k| \leq n} |c_k|^2
\]

\[
= \frac{1}{12} - c_0^2 - 2 \sum_{k=1}^{n} |c_k|^2
\]

\[
= \frac{1}{48} - 2 \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} \frac{1}{4\pi^2 (2k)^2}
\]

\[
= \frac{1}{48} - \frac{1}{8\pi^2} \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} \frac{1}{k^2}
\]

We need the evaluate the sum, \( \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} \frac{1}{k^2} \). In order to do this we are going to say that

\[
\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} \frac{1}{k^2} = \sum_{k=1}^{\infty} \frac{1}{k^2} - \sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{\infty} \frac{1}{k^2}
\]

\[
= \frac{\pi^2}{6} - \sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{\infty} \frac{1}{k^2}
\]

Now we are going to evaluate the last sum with integrals

\[
\int_k^{k+1} \frac{dt}{t^2} \leq \frac{1}{k^2} \leq \int_{k-1}^{k} \frac{dt}{t^2}
\]

\[
\frac{1}{M} = \int_{k=M}^{\infty} \frac{dt}{t^2} \leq \sum_{k=M}^{\infty} \frac{1}{k^2} \leq \int_{k=M-1}^{\infty} \frac{dt}{t^2} = \frac{1}{M - 1}
\]

This means that for \( n \gg 1 \) we have

\[
\sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{\infty} \frac{1}{k^2} \sim \frac{2}{n}
\]

Finally we can estimate the distance in \( L^2 \) between \( f \) and \( S_n(f) \) as a function of \( n \) and we get

\[
\|f - S_n(f)\|_{L^2}^2 \sim \frac{1}{4\pi^2 n}
\]
(c) If we are summing over the \(2n + 1\) bigger coefficients, we have the coefficients \(c_0\) and then the first \(n\) positive and \(n\) negative even coefficients; this means we are considering the series \(S_{2n}(f)\). Using the formula we just determined we get

\[
\|f - S_{2n}(f)\|_{L^2}^2 \sim \frac{1}{8\pi^2 n}
\]

(d) It is fairly easy to do this question in MATLAB using the FFT of the signal. The program is self explanatory.

```matlab
% Matlab Program to observe Gibbs oscillations
% clear all

% Number of points
N=1024;

% Time vector
T=(0:N-1)/N;

% Initial signal
f=(t>1/2); % Unit step
%f=t.*(t<1/2)+(t-1/2).*(t>1/2); % Sawtooth

% FFT
fhat=fft(f);

% Filtering the appropriate frequencies
M=64; % half the number of filtering points
omega=(N*t<M)+(N*t>N-M);

% IFFT
Sf=ifft(fhat.*omega);
maxi=max(real(Sf));

% Plots
plot(t,f,'--',t,real(Sf))
xlabel('time in s')
ylabel('Function values')
title(['Gibbs phenomenon for finite discontinuities
and max(S_{n}(f))=' num2str(maxi)])
legend('Unit step','Fourier Series (128 coefficients)')
print -deps fig0.eps
unix('epstopdf fig0.eps');
```
We can see the formation of Gibbs’ oscillations in the figure below.

(e) We just use the same program with a different initial signal. The result can be seen in the figure below.
(f) If we consider the step function, the Fourier coefficients will be

\[ c_k = \int_0^1 f(t) e^{-i2\pi kt} \, dt \]

\[ = \int_{\frac{1}{2}}^1 e^{-i2\pi kt} \, dt \]

\[ = -\frac{1}{2\pi ik} \left[ e^{-i2\pi kt} \right]^{\frac{1}{2}} \]

\[ = (-1)^k - 1 \]

\[ \frac{1}{2\pi ik} \]

or if we prefer

\[ c_{2k} = 0 \]

\[ c_{2k+1} = -\frac{1}{\pi ik} \]

Now in order to calculate \( \lim_{n \to +\infty} \max S_n(f) \) we are going to choose \( n \) even. This will make the calculations easier and, because we know the limit exists, it will not affect the result (we could follow the same procedure with \( n \) odd).

We have

\[ S_n(f) = \sum_{k=-n}^n c_k e^{i2\pi kt} \]

\[ = -\frac{1}{\pi i} \sum_{k=-\frac{n}{2}+1}^{\frac{n}{2}} \frac{e^{i2\pi(2k-1)t}}{2k-1} \]

Before going one step further, we calculate the general formula (we are going to need it later)

\[ \sum_{k=-M}^{N} a^k = \sum_{k=0}^{N} a^k + \sum_{k=0}^{M} \left( \frac{1}{a} \right)^k - 1 \]

\[ = \frac{1 - a^{N+1}}{1 - a} + \left( \frac{1}{a} \right)^{M+1} - 1 - \frac{1 - a}{1 - a} \]

\[ = a^{N+1} + a^{-M} \]

\[ = a^{-1} \]

\[ = a^{N-M} \frac{a^{N+1} + a^{M+1} - a^{N+1}}{a^{\frac{1}{2}} - a^{-\frac{1}{2}}} \]

\[ = a^{N-M} \frac{a^{\frac{N+1 + M}{2}} - a^{-\frac{N+1+M}{2}}}{a^{\frac{1}{2}} - a^{-\frac{1}{2}}} \]
To get the maximum for $S_n(f)$ we calculate $\frac{dS_n(f)}{dt}$

$$S_n'(f) = -2 \sum_{k=\frac{-n}{2}+1}^{\frac{n}{2}} e^{i2\pi(2k-1)t}$$

$$= -2e^{-i2\pi t} \sum_{k=\frac{-n}{2}+1}^{\frac{n}{2}} e^{i4\pi kt}$$

$$= -2e^{-i2\pi t} \left[ (i4\pi t) \frac{n}{2} + \frac{n}{2} + 1 \right] \frac{\sin\left(\frac{(4\pi t) \frac{n}{2} + \frac{n}{2}}{2}\right)}{\sin\left((4\pi t) \frac{1}{2}\right)}, \text{ use of previous formula}$$

$$= -2 \frac{\sin(2\pi nt)}{\sin(2\pi t)}$$

Now, we see we get $S_n'(f)(t) = 0$ for $t_l = 1/2 + l/(2n)$ with $l \in [-n, n]$. The functions $S_n(f)$ are symmetric with respect to the point $(1/2, 1/2)$. Therefore we only need $l \in [0, n]$. With this interval we can prove (by calculating $S_n''(f)$), that for $n$ even, the maximums occur for $l$ odd. Now that we know all this we can go back to getting the limit of the maximum of $S_n(f)$.

We write $S_n(f)$ in its real Fourier series decomposition. We easily get that

$$S_n(f) = \frac{1}{2} - \frac{1}{\pi} \sum_{k=0}^{n-1} \sin\left[2\pi(2k+1)(t - \frac{1}{2})\right] \frac{1}{2k+1}$$

Now we replace $t$ by $t_l$ and we get

$$S_n(f) = \frac{1}{2} - \frac{1}{\pi} \sum_{k=0}^{n-1} \sin\left(\frac{2\pi(2k+1)t_l}{2n}\right) \frac{1}{2k+1}$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} \sin\left(\frac{\pi(2k+1)t_l}{n}\right) \frac{1}{n} \sum_{k=0}^{\frac{n-1}{2}} \sin\left(\frac{\pi k}{2}\right)$$

Let’s note what a Riemann integral is. If a function $f$ is Riemann-integrable for $x \in [a, b]$ then

$$\int_a^b f(x) \, dx = \lim_{n \to +\infty} \left[ \frac{b-a}{n} \sum_{k=0}^{n-1} f\left(\frac{(b-a)k}{n}\right) \right]$$

Now that we know this we easily get

$$\lim_{n \to +\infty} S_n(f)(t_l) = \frac{1}{2} + \frac{1}{2\pi} \int_0^\pi \frac{\sin(lx)}{x} \, dx$$

and for $l$ odd and $l \in [0, N]$ we get the maximum value for $l = 1$ and $(1/\pi) \int_0^\pi \sin(x)/x \, dx \approx 1.1789$. We finally get

$$\lim_{n \to +\infty} \max S_n(f) = \frac{1}{2} + \frac{1}{2\pi} \int_0^\pi \frac{\sin(x)}{x} \, dx \approx 1.089$$
Problem 2

2.1 Problem 3.14 on page 65

We want to compute numerically the Fourier transform of \( f(t) \). Let \( f_d[n] = f(nT) \), and \( f_p[n] = \sum_{p=-\infty}^{+\infty} f_d[n - pN] \).

(a) Prove that the DFT of \( f_p[n] \) is related to the Fourier series of \( f_d[n] \) and to the Fourier transform of \( f(t) \) by

\[
\hat{f}_p[k] = \hat{f}_d\left(\frac{2\pi k}{N}\right) = \frac{1}{T} \sum_{l=-\infty}^{+\infty} \hat{f}\left(\frac{2k\pi}{NT} - \frac{2l\pi}{T}\right).
\]

(b) Suppose that \( |f(t)| \) and \( |\hat{f}(\omega)| \) are negligible when \( t \not\in [-t_0, t_0] \) and \( \omega \not\in [-\omega_0, \omega_0] \). Relate \( N \) and \( T \) to \( t_0 \) and \( \omega_0 \) so that one can compute an approximation value of \( \hat{f}(\omega) \) at all \( \omega \in \mathbb{R} \) by interpolating the samples \( \hat{f}_p[k] \). Is it possible to compute exactly \( \hat{f}(\omega) \) with such an interpolation formula?

(c) Let \( f(t) = \left(\frac{\sin(\pi t)}{\pi t}\right)^4 \). What is the support of \( \hat{f} \)? Sample \( f \) appropriately and compute \( \hat{f} \) with the FFT algorithm of MATLAB.

2.2 Solution

(a) Let’s start with the first equality

\[
\hat{f}_p[k] = \sum_{n=0}^{N-1} f_p[n]e^{-\frac{i2\pi kn}{N}} = \sum_{n=0}^{N-1} \left( \sum_{p=-\infty}^{+\infty} f_d[n - pN] \right) e^{-\frac{i2\pi kn}{N}}
\]

\[
= \sum_{p=-\infty}^{+\infty} \left( \sum_{n=0}^{N-1} f_d[n - pN] \right) e^{-\frac{i2\pi kn}{N}}
\]

\[
= \sum_{p=-\infty}^{+\infty} \left( \sum_{n=-pN}^{-pN+N-1} f_d[n] \right) e^{-\frac{i2\pi kn}{N}}
\]

\[
= \sum_{n=-\infty}^{\infty} f_d[n]e^{-\frac{i2\pi kn}{N}}
\]

\[
= \hat{f}_d\left(\frac{2\pi k}{N}\right)
\]

For the second equality we use Proposition 3.1 with a little twist. In this exercise \( \hat{f}_d(\omega) \) is the Fourier series of the samples \( f(nT) \) renormalized to unit sampling rate, i.e. \( T \) does not appear in \( \hat{f}_d(\omega) \). This is unlike proposition 3.1 where \( \hat{f}_d(\omega) \) was the Fourier transform of a train of Diracs at the sampling rate \( T \). They are the image of each other by a dilation of a factor \( T \), and we substitute into formula (3.3) to obtain

\[
\hat{f}_d\left(\frac{2\pi k}{N}\right) = \frac{1}{T} \sum_{l=-\infty}^{+\infty} \hat{f}\left(\frac{2k\pi}{NT} - \frac{2l\pi}{T}\right).
\]
If we want to recover the values of $\hat{f}(\omega)$ the easiest thing to do is to keep only one component of the previous infinite sum in our frequency window $\omega \in [-\omega_0, \omega_0]$. First, we use the time window: if $|f(t)|$ has a support $t \in [-t_0, t_0]$, so if we want no aliasing when we sample $\hat{f}(\omega)$ we need to choose an anti-aliasing time (yes, time, because we are sampling the Fourier transform and not the original function, so $2\pi/\omega \geq 2t_0$ with $\omega = 2\pi/NT$). This means we can choose

$$NT = 2t_0$$

We could also pick $NT \geq 2t_0$ but this will only give us more samples and those extra samples would come from the negligible region.

Now in the frequency domain, we just use the aliasing reasoning but we also want only one sample of $\hat{f}$ in our support $\omega \in [-\omega_0, \omega_0]$. We just have to pick the normal anti-aliasing frequency

$$\frac{\pi}{T} = \omega_0 \quad \text{(or} \quad \frac{\pi}{T} \geq \omega_0)$$

If we evaluate the argument of $\hat{f}$ for this value we are going to get the support we wanted

$$-\omega_0 < \frac{2\pi}{T}(\frac{k}{N} - l) < \omega_0$$

$$-\omega_0 < 2\omega_0(\frac{k}{N} - l) < \omega_0$$

$$-\frac{1}{2} < \frac{k}{N} - l < \frac{1}{2}$$

and because $0 \leq k < N$ we get $l = 0$ and $\hat{f}_p[k] = \frac{1}{T}\hat{f}(\frac{2k\pi}{NT})$. If we forget about the negligible region, we can apply Shannon’s theorem to $\hat{f}(\omega)$: $f$ has a finite support and we are sampling $\hat{f}$ at an anti-aliasing frequency so we can recover $\hat{f}$ in $\mathbb{R}$. We can therefore calculate an approximate value of $\hat{f}(\omega)$, that we will call $\hat{f}_{num}(\omega)$.

$$\hat{f}(\omega) \approx \hat{f}_{num}(\omega) = \sum_{k=-\infty}^{+\infty} \hat{f}(\frac{2k\pi}{NT})h(\omega - \frac{2k\pi}{NT})$$

$$= T \sum_{k=-\infty}^{+\infty} \hat{f}_p[k]h(\omega - \frac{2k\pi}{NT}).$$

In general we cannot recover $\hat{f}$ exactly using this procedure. This can be understood at an intuitive level by noticing that a nonzero function $f(t)$ and its Fourier transform $\hat{f}(\omega)$ cannot be both compactly supported at the same time (Paley-Wiener theory). Since periodization occurs in both space and frequency, there is overlap in at least one domain, resulting in possible aliasing and loss of information.

This question is very tricky if one insists on giving a rigorous answer to it. The following discussion is not expected as an answer to the homework assignment, and does not solve the problem completely either. One should start by reading the problem so as to make it interesting. By ‘computing exactly $\hat{f}(\omega)$’, let us understand that if $\hat{f}(\omega)$ is compactly supported in some interval, we only require the interpolation to give us the exact answer inside that interval. So we allow ourselves to indicate the result of the interpolation by some rectangular window. Otherwise, the problem is trivial. Indeed, it is easy to see that $\hat{f}(\omega)$ defined by formula (1) is periodic with period $\pi/T$, so if we were to recover it for all $\omega \in \mathbb{R}$, it would mean that we can only start from ‘functions’ $f(t)$ that are sums of Diracs.
equispaced with spacing $T$. Those, in turn, cannot be evaluated at the points $nT$ in a classical sense, so $f_d[n]$ would not be defined for them.

Now it turns out that there exist functions $f(t)$, arbitrarily small outside some interval $[-t_0,t_0]$, and compactly supported in frequency, so that the proposed interpolation scheme recovers $\hat{f}(\omega)$ exactly on its support. There are very few such examples, however. Consider the following sum

$$f(t) = \sum_{n=0}^{N-1} c_n h_T(t-nT),$$

with $h_T(t) = \frac{\sin(\pi t/T)}{\pi t/T}$, and the $c_n$ are arbitrary real numbers. Since this is Whittaker interpolation (in time), we have $f(nT) = c_n$ for $0 \leq n < N$, and 0 for all other values of $n$. Periodization with period $NT$ will result in overlap for the function $f(t)$, but not for the samples $f(nT)$:

$$f_p[n] = f_d[n] = f(nT)$$

for $0 \leq n < N$. The Fourier transform $\hat{f}(\omega)$ is compactly supported in $[-\pi/T, \pi/T]$ and given by

$$\hat{f}(\omega) = T \sum c_n e^{-inT\omega}$$

for $-\pi/T \leq \omega \leq \pi/T$. This expression can be obtained by observing that $f(t)$ is obtained from its samples from a convolution with $h_T(t)$, whose Fourier transform is $T \mathbf{1}_{[-\pi/T, \pi/T]}$. On the other hand, the DFT of the finite sequence $f_p[n]$ is

$$\hat{f}_p[k] = \sum_{n=0}^{N-1} c_n e^{-2\pi i k n N},$$

and its Whittaker interpolation by means of formula (1) can be checked to be (replace $2\pi k NT$ by $\omega$)

$$T \sum_{n=0}^{N-1} c_n e^{-inT\omega},$$

this time on the whole line $\omega \in \mathbb{R}$. Windowing to the interval $[-\pi/T, \pi/T]$ therefore allows to recover $\hat{f}(\omega)$ exactly. As for the behavior of $f(t)$ at infinity, notice that a sum of sinc’s can be made arbitrarily small outside of a sufficiently large interval $[-t_0, t_0]$.

We leave to the reader to check that this is the only type of function which can be recovered exactly by the proposed method.
(c) In order to calculate the FT of \((\sin(\pi t)/(\pi t))^4\) we use the Fourier transform of a product of functions

\[
\mathcal{F}\left(\left(\frac{\sin(\pi t)}{\pi t}\right)^4\right) = \frac{1}{2\pi} \mathcal{F}\left(\left(\frac{\sin(\pi t)}{\pi t}\right)^2\right) \ast \mathcal{F}\left(\left(\frac{\sin(\pi t)}{\pi t}\right)^2\right) = \frac{1}{8\pi^3} \left(\mathcal{F}\left(\frac{\sin(\pi t)}{\pi t}\right) \ast \mathcal{F}\left(\frac{\sin(\pi t)}{\pi t}\right) \ast \mathcal{F}\left(\frac{\sin(\pi t)}{\pi t}\right)\right)
\]

First we need to calculate \(\mathcal{F}(\sin(\pi t)/(\pi t))^2\). By using the above formula we get

\[
\mathcal{F}\left(\left(\frac{\sin(\pi t)}{\pi t}\right)^2\right) = \frac{1}{2\pi} \mathcal{F}\left(\frac{\sin(\pi t)}{\pi t}\right) \ast \mathcal{F}\left(\frac{\sin(\pi t)}{\pi t}\right) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} 1_{[-\pi,\pi]}(\omega-\pi,\omega+\pi) d\omega = \frac{1}{2\pi} \int_{\omega-\pi}^{\omega+\pi} 1_{[-\pi,\pi]} d\omega = \frac{1}{2\pi} (2\pi - \omega \text{sign}(\omega)) 1_{[-2\pi,2\pi]}
\]

this means we have

\[
\mathcal{F}\left(\left(\frac{\sin(\pi t)}{\pi t}\right)^4\right) = \frac{1}{8\pi^3} \left[ (2\pi - \omega \text{sign}(\omega)) 1_{[-2\pi,2\pi]} \ast [(2\pi - \omega \text{sign}(\omega)) 1_{[-2\pi,2\pi]}] \right] = \frac{1}{8\pi^3} \left[ \int_{\omega-2\pi}^{\omega+2\pi} (2\pi - u \text{sign}(u))(2\pi - (\omega - u) \text{sign}(\omega - u)) 1_{[-2\pi,2\pi]} d\omega \right]
\]

First of all, we have to take into account the fact that this Fourier transform is even. Now, we calculate the result for two different intervals \(\omega \in [0,2\pi]\) and \(\omega \in [2\pi,4\pi]\).

For \(\omega \in [0,2\pi]\) we get

\[
\mathcal{F}\left(\left(\frac{\sin(\pi t)}{\pi t}\right)^4\right)(\omega) = \frac{1}{8\pi^3} \left[ \int_{\omega-2\pi}^{2\pi} (2\pi - u \text{sign}(u))(2\pi - (\omega - u) \text{sign}(\omega - u)) d\omega \right] = \frac{1}{8\pi^3} \left[ \int_{\omega-2\pi}^{0} (2\pi + u)(2\pi + u - \omega) d\omega + \int_{0}^{\omega} (2\pi - u)(2\pi + u - \omega) d\omega + \int_{\omega}^{2\pi} (2\pi - u)(2\pi + \omega - u) d\omega \right] = \frac{1}{8\pi^3} \left( \frac{\omega^3}{2} - 2\pi\omega^2 + \frac{16\pi^3}{3} \right)
\]

Now for \(\omega \in [2\pi,4\pi]\) we get

\[
\mathcal{F}\left(\left(\frac{\sin(\pi t)}{\pi t}\right)^4\right)(\omega) = \frac{1}{8\pi^3} \left[ \int_{\omega-2\pi}^{2\pi} (2\pi - u \text{sign}(u))(2\pi - (\omega - u) \text{sign}(\omega - u)) d\omega \right] = \frac{1}{8\pi^3} \int_{\omega-2\pi}^{2\pi} (2\pi - u)(2\pi + u - \omega) d\omega = \frac{1}{8\pi^3} \left( \frac{\omega^3}{6} + 2\pi\omega^2 - 8\pi^2\omega + \frac{32\pi^3}{3} \right)
\]
Now we only need to build an even function to get

\[
\mathcal{F}\left(\frac{\sin \frac{\pi t}{\pi} \sin \frac{4\pi}{\omega}}{\pi t}\right)(\omega) = \frac{1}{8\pi^3} \left[ \frac{\text{sign}(\omega)}{2} \omega^3 - 2\pi\omega^2 + \frac{16\pi^3}{3} \right] 1[-2\pi, 2\pi] \\
+ \frac{1}{8\pi^3} \left[ -\frac{\omega^3}{6} \text{sign}(\omega) + 2\pi\omega^2 - 8\pi^2\omega \text{sign}(\omega) + \frac{32\pi^3}{3} \right] 1[-4\pi, -2\pi] \cup [2\pi, 4\pi]
\]

Now that we have the support of \( f \), \([-4\pi, 4\pi]\), we can use MATLAB to compute an approximation to \( \hat{f} \) and compare with the exact analytical solution. We just have to choose \( T = \frac{\pi}{\omega_0} = \frac{1}{4} \) and then we take a total time \( t_0 = 16 \) (at this point the function is already very small) which means we have \( N = 2t_0/T = 128 \) samples. We can now compare our calculated FT and the numerical FFT to see how good an approximation our guesses were. Here is what we get with our numerical calculation of the Fourier transform

We can repeat this procedure choosing different times \( t_0 \) after which we consider the function to be negligible. Here is a plot of the results we obtain (the parameters keep their definitions which means that \( N = 2t_0/T = 8t_0 \) will change every time).
Here is the MATLAB code for this question.

```matlab
% Computing FTs numerically (for f(t)(sin(pi*t)/(pi*t))^4)

clear all

t0=1; % time domain
w0=4*pi; % frequency domain
T=pi/w0; % minimal period of sampling
N=2*t0/T; % minimal number of samples

% Approximation
n=(-N/2:N/2-1)*T;
fd(1:N/2)=(sin(pi*n(1:N/2))/(pi*n(1:N/2))).^4; % sampled fd
fd(N/2+1)=1; % value at the origin (problem with divide by zero)
fd(N/2+2:N)=(sin(pi*n(N/2+2:N))/(pi*n(N/2+2:N))).^4; % sampled fd
fphat=fft(fd); % interpolation points function fp
fhat(1:N/2)=T*abs(fphat(N/2+1:N)); % building and
fhat(N/2+1:N)=T*abs(fphat(1:N/2)); % shifting fhat

% Here we have to be careful and to get rid
% of the small imaginary parts by taking the
% modulus of the FFT coefficients.

% Reconstructing FT with 1024 points
M=1024;
om=2*pi/(N*T);
omega=(-M/2:M/2-1)*2*w0/M;
vec=(-N/2:N/2-1)*om;
fhatre=zeros(1,M);
for i=1:M,
    for j=1:N,
        if (omega(i)-vec(j)==0),
            fhatre(i)=fhatre(i)+fhat(j);
        else
            fhatre(i)=fhatre(i)+
            fhat(j)*sin(pi*(omega(i)-vec(j))/om)/(pi*(omega(i)-vec(j))/om);
        end
    end
end

% Comparing with the real FT
for i=1:M/4,
    ft(i)=((1/(8*pi^3))*(omega(i)^3/6+2*pi*omega(i)^2
    +8*pi^2*omega(i)+32*pi^3/3);
end
```

13
for i=M/4+1:M/2,
    ft(i)=(1/(8*pi^3))*(-omega(i)^3/2-2*pi*omega(i)^2+16*pi^3/3);
end
for i=M/2+1:3*M/4,
    ft(i)=(1/(8*pi^3))*(omega(i)^3/2-2*pi*omega(i)^2+16*pi^3/3);
end
for i=3*M/4+1:M,
    ft(i)=(1/(8*pi^3))*(-omega(i)^3/6+2*pi*omega(i)^2
    -8*pi^2*omega(i)+32*pi^3/3);
end

% Plots
% plot(omega,ft,omega,fhatre+0.01);
% xlabel('Frequency range [-4\pi,4\pi]')
% ylabel('Fourier transform')
% legend('Analytical FT','Numerical FT lifted by 0.01')
% title('Comparing exact and numerical FTs for t_\{0\}=16')
% print -deps fig2.eps
% unix('epstopdf fig2.eps');

% figure
% plot(omega,abs(fhatre-ft))
% xlabel('Frequency range [-4\pi,4\pi]')
% ylabel('|FT(f)-FT(f)_{\text{num}}|')
% title('Error between exact and numerical FTs for t_\{0\}=16')
% print -deps fig3.eps
% unix('epstopdf fig3.eps');

% Error in l^2
error=norm(fhatre-ft);

% Table of errors
tv=[1 2 4 8 16 32];
er=[0.0275 0.0035 3.5199e-4 3.2424e-5 2.8967e-6 2.5720e-7];
plot(tv,er,'-o')
text(1,0.0285,'0.0275'),text(2.05,0.0045,'0.0035')
text(4,0.0015,'3.52e-4'),text(8,0.001,'3.24e-5')
text(16,0.001,'2.9e-6'),text(32,0.001,'2.57e-7')
xlabel('times t_\{0\}')
ylabel('error in l^2')
title('Error for different maximum times t_\{0\}=2^\{n\}
    with \omega=4\pi, T=\pi/\omega_\{0\} and N=2t_\{0\}/T')
print -deps fig4.eps
unix('epstopdf fig4.eps');
Problem 3

3.1 Frequency selective filters

Let \( Y = X + Z \) where \( X \) is the process to be recovered and \( Z \) is a white noise with noise level \( \sigma \). As we have seen in class, the Mean-Squared Error (MSE) is obtained by the Wiener filter. A frequency selective \( D \) filter has weights which can only take the values 0 or 1.

(a) Find the frequency selective weights that minimize the MSE.

(b) Prove that

\[
MSE(\text{Wiener}) \leq MSE(D) \leq 2MSE(\text{Wiener}).
\]

3.2 Solution

First of all, let’s write the estimate using the weights defined by the Wiener filter and its MSE. We know from class that the estimate minimizing the MSE is of the form

\[
\hat{X} = E(X|Y) = \sum_{k=0}^{T-1} w_k < Y, \phi_k > \phi_k,
\]

with \( w_k = \frac{\lambda_k^2}{\lambda_k^2 + \sigma^2} \)

where \( \phi_k \)'s are the eigenvectors of \( X \) and \( \lambda_k \) its eigenvalues.

We also need the MSE for a weighted estimator. By definition we have

\[
MSE = E(\|X - \hat{X}\|^2) = \sum_{k=0}^{T-1} E(\|X_k - \hat{X}_k\|^2)
\]

\[
= \sum_{k=0}^{T-1} E(\|X_k - w_k(X_k + Z_k)\|^2)
\]

\[
= \sum_{k=0}^{T-1} E(\|(1 - w_k)X_k - w_kZ_k\|^2)
\]

\[
= \sum_{k=0}^{T-1} E((1 - w_k)^2X_k^2 + w_k^2Z_k^2 - 2w_k(1 - w_k)X_kZ_k)
\]

\[
= \sum_{k=0}^{T-1} \left( (1 - w_k)^2E(X_k^2) + w_k^2E(Z_k^2) - 2w_k(1 - w_k)E(X_kZ_k) \right)
\]

\[
= \sum_{k=0}^{T-1} \left( (1 - w_k)^2\lambda_k^2 + w_k^2\sigma^2 \right) \quad \text{because } E(X_kZ_k) = 0
\]

and if we use this formula with the Wiener filter (we just have to replace \( w_k \) by its value we find

\[
MSE(\text{Wiener}) = \sum_{k=0}^{T-1} \frac{\lambda_k^2\sigma^2}{\lambda_k^2 + \sigma^2}
\]

Now that this has been established we can start with the exercise.
(a) This question is fairly easy: we just have to find the best way to approximate the Wiener weights with zeroes and ones. We get

\[
   w_k = \begin{cases} 
   1 & \text{if } \frac{\lambda_k^2}{\lambda_k^2 + \sigma^2} \geq \frac{1}{2} \quad (\text{or } \sigma \leq |\lambda_k|) \\
   0 & \text{if } \frac{\lambda_k^2}{\lambda_k^2 + \sigma^2} < \frac{1}{2} \quad (\text{or } \sigma > |\lambda_k|) 
   \end{cases}
\]

This is indeed the choice which minimizes each term \((1 - w_k)^2 \lambda_k^2 + w_k^2 \sigma^2\) in the expression of the MSE.

(b) We already know the first part of this question: the Wiener filter is the best possible approximation; therefore \(MSE(\text{Wiener}) \leq MSE(D)\). For the second part we just have to use the formulas we calculated at the beginning of the exercise. We know the expression of \(MSE(\text{Wiener})\), so we just have to calculate the difference \(2MSE(\text{Wiener}) - MSE(D)\) and show it is positive.

\[
2MSE(\text{Wiener}) - MSE(D) = \sum_{k=0}^{T-1} \left[ \frac{2\lambda_k^2 \sigma^2}{\lambda_k^2 + \sigma^2} - ((1 - w_k)^2 \lambda_k^2 + w_k^2 \sigma^2) \right] 
\]

\[
= \sum_{k=0}^{T-1} \frac{\lambda_k^2 (\sigma^2 - \lambda_k^2) H_{\text{e}}(\sigma - |\lambda_k|) + \sigma^2 (\lambda_k^2 - \sigma^2) H_{\text{e}}(|\lambda_k| - \sigma)}{\lambda_k^2 + \sigma^2} 
\]

\[
= \sum_{k=0}^{T-1} \frac{\lambda_k^2}{\lambda_k^2 + \sigma^2} \min(\lambda_k^2, \sigma^2) \geq 0 \quad (H_{\text{e}} \text{ was the Heaviside function}) 
\]
Problem 4

4.1 Gaussian Processes

Consider the simple Gaussian process defined by

\[ X_t = \rho X_{t-1} + \epsilon_t, \quad t = 1, \ldots, T. \]

where \(-1 < \rho < 1\) and the \(\epsilon_t\)’s are iid standard normal (we will take \(X_0 = 0\) so that \(X_1 = \epsilon_1\)).

(a) Find the covariance matrix \(\Sigma\) of \(X\).

(b) Explain how you would draw a realization of \(X\).

(c) In the remainder of this exercise, we will take \(\rho = 0.8\). Generate a sample from the observed process \(Y\)

\[ Y_t = X_t + \frac{1}{4} W_t, \quad t = 1, \ldots, T = 1024, \]

where \(X\) is as before and \(W\) is a standard white noise.

(d) Construct the Wiener filter and plot \(X\) and your estimate \(\hat{X}\) on a same figure. Calculate the estimation error

\[ 1/T \sum (X_t - \hat{X}_t)^2. \]

(e) Repeat (d) a thousand times (without plotting the results each time) and summarize your results with a histogram. Compute the average estimation error.

(f) Calculate the exact Mean Squared Error for this estimation problem and compare this value with your findings in the previous question. Comment.

(g) Bonus question: What is the standard deviation of the average you calculated in (e)?

4.2 Solution

(a) For this first question we just have to find the expression of \(X_t\) as a function of \(\rho\) and the \(\epsilon_t\)’s. It is easy to find that

\[ X_t = \sum_{k=1}^{t} \rho^{t-k} \epsilon_k \]

Now that we know this we just use the definition of the covariance matrix to find its components (remember that because the \(\epsilon_t\)’s are iid (independent identically distributed) standard normal we have that \(E(\epsilon_t) = 0, E(\epsilon_t^2) = 1\) and for \(i \neq j, E(\epsilon_i \epsilon_j) = 0\).
The calculations look like this

\[ \Sigma_{i,j} = \text{cov}(X)_{i,j} = (E(X_iX_j) - E(X_i)E(X_j)) \]

\[ = E(X_iX_j) \]

\[ = E(\left(\sum_{k=1}^{i} \rho^{i-k}\epsilon_k\right)\left(\sum_{l=1}^{j} \rho^{j-l}\epsilon_l\right)) \]

\[ = \sum_{1 \leq k \leq i, 1 \leq l \leq j} \rho^{i+j-k-l}E(\epsilon_k\epsilon_l) \]

\[ = \sum_{k=1}^{\min(i,j)} \rho^{i+j-2k} \quad \text{we needed } k = l \text{ and then } E(\epsilon_k^2) = 1 \]

\[ = \rho^{i+j} \left[ \sum_{k=0}^{\min(i,j)} \left( \frac{1}{\rho^2} \right)^k - 1 \right] \]

\[ = \rho^{i+j} \left[ \frac{\left( \frac{1}{\rho^2} \right)^{\min(i,j)} - 1}{1 - \rho^2} \right] \]

and this will give us the covariance matrix.

(b) To draw a realization of \( X \) we just have to generate the \( \epsilon_t \)'s and pick a value for \( \rho \) and we will have created the values \( X_t \).

(c) This question is treated in the MATLAB program (at the end of the solutions for this problem).

(d) This is done in the MATLAB program. Here are the plots for \( X \) and \( \hat{X} \) between times 500 and 600.
(e) Here we have the histogram of the estimated errors (the 1000 samples have been divided into a 100 intervals for the plot). We get 0.056892 for the average error.

(f) The exact MSE for the Wiener filter has been derived in question 3 and is given by

$$MSE = \sum_k \frac{\lambda_k^2 \sigma^2}{\lambda_k^2 + \sigma^2}.$$  

In our case, the numerical value is 0.056926. We can note that the values of the average estimated error and the MSE are different (even if the order of magnitude is similar). We would need an infinity of samples to reproduce the MSE because by using the expectation we are taking into account all the possible white noise values.

(g) First of all, the standard deviation of a mean of N samples is equal to

$$Var(\bar{X}) = \frac{Var(X)}{N}.$$  

Now we just have to calculate the $Var(X)$ to get what we want. In our case we are going to have (see MATLAB code)

$$Var_{\text{rest}}\left(\frac{1}{T} \sum (X_t - \hat{X}_t)^2\right) = Var\left(\frac{1}{T} \sum (X_t - \hat{X}_t)^2\right) = 5.9286 \times 10^{-9}.$$  

and the estimated standard deviation is just equal to $\text{stdest} = 7.6997e-05$ which seems correct because the MSE and the average error differ by something smaller than the estimated standard deviation $\text{stdest}$. We can also compute the exact value of the variance of the error $\|X - \hat{X}\|^2$. First, we are going to calculate the values of $E(X_k^4)$ and $E(Z_k^4)$ because will need them later. We already know that, because of the eigenvalue decomposition, we will have $E(X_k^4) = E(Z_k^4)\sigma^2 = \lambda_k^2$ (we just exchange the eigenvalues). The same happens for $E(X_k^2)$ and $E(Z_k^2)$, so we only need to calculate one of them. It
will also be useful to remember that, if we call $\phi(y)$ the probability density function (with a variance $\sigma$ and for simplicity we suppose we have a centered variable), we have $\sigma^2 d\phi/dy = -y\phi(y)$. This comes from the fact that $\phi(y)$ is of the form $\frac{1}{\sqrt{2\pi} \sigma} e^{-\frac{y^2}{2\sigma^2}}$. So, 

$$E(Y^4) = \int_{-\infty}^{+\infty} y^4 \phi(y) dy = -\sigma^2 \int_{-\infty}^{+\infty} y^3 \phi'(y) dy = -\sigma^2 \left[ y^3 \phi(y) \right]_{-\infty}^{+\infty} + 3 \int_{-\infty}^{+\infty} y^2 \phi(y) dy = 3\sigma^4$$

and in our case we will have $E(X_k^4) = 3\lambda_k^4$ and $E(Z_k^4) = 3\sigma^4$. We can now calculate the variance of a weighted filter decomposed on the Karhunen-Loève basis.

$$Var(||X - \hat{X}||^2) = E(||X - \hat{X}||^4) - (E(||X - \hat{X}||^2))^2$$

we already know the second term

$$= \sum_{k=0}^{T-1} E((1 - w_k)^4 X_k^4 - w_k^4 Z_k^4) - (E(||X - \hat{X}||^2))^2$$

$$= \sum_{k=0}^{T-1} \left[ (1 - w_k)^4 E(X_k^4) - 4(1 - w_k)^3 w_k E(X_k^3 Z_k) + 2(1 - w_k)^2 w_k^2 E(X_k^2 Z_k^2) \right.$$

$$- 4(1 - w_k) w_k^3 E(X_k Z_k^2) + w_k^4 E(Z_k^4) \left. \right] - (E(||X - \hat{X}||^2))^2$$

$$= \sum_{k=0}^{T-1} \left[ 3(1 - w_k)^4 \lambda_k^4 + 2(1 - w_k)^2 w_k^2 \lambda_k^2 \sigma^2 + 3w_k^4 \sigma^4 \right] - (E(||X - \hat{X}||^2))^2$$

$$= \sum_{k=0}^{T-1} \left[ \frac{\sigma^4 \lambda_k^4}{(\sigma^2 + \lambda_k^2)^4} (3(\sigma^4 + \lambda_k^4) + 2\sigma^2 \lambda_k^2) \right] - (E(||X - \hat{X}||^2))^2$$

This result, again, should be divided by $N$ to account for the fact that we take an average over $N$ samples. Taking the square root, we obtain the exact value of the standard deviation: $\text{stdexact} = 5.6526e - 05$ which is of the same order as the value we found above empirically.

Here’s the matlab code for this last question.
clear all;

% Constant
rho=0.8;
T=1024;
sigma=0.25;
N=1000;

% Covariance matrix
for i=1:T,
    for j=1:i,
        F(i,j)=rho^(i-j);
    end
end

Sig=F*transpose(F);

% Eigenvalues and eigenvectors
[phi,lambdasq]=eig(Sig);

%load phi;
%load lambdasq;

%Initializing
x=zeros(T,1);

% Weight matrix
Wei=lambdasq./(lambdasq+sigma^2);
W=phi*Wei*transpose(phi);

% MSE
mse=(1/T)*(trace(Wei))*sigma^2;

% Loop
for i=1:N,

% Generating the epsilons
eps=randn(T,1);

% Generating the x
x=F*eps;

% Generating the white noise
w=randn(T,1);
% Generating the observed process
y = x + sigma * w;

% Generating the Wiener estimator
xhat = W * y;

% Calculating the estimation error
esterror(i) = (1/T) * (x - xhat)' * (x - xhat);

% End of loop
end

% Average estimation error
avesterror = mean(esterror);

% Estimated Variance of the average estimation error
varest = var(esterror);
varestbar = varest / N;
stdest = sqrt(varestbar);

% Exact Variance
varexact = 0;
for k = 1:T,
    varexact = varexact + lambdasq(k)^2 * (3*lambdasq(k)^2 + 3*sigma^4 + 2*sigma^2*lambdasq(k)) / (sigma^2 + lambdasq(k))^4;
end
varexact = (sigma^4 * varexact - mse^2) / T^2;
stdexact = sqrt(varexact);

% Plots
figure(1)
plot((1:T), x, 'o', (1:T), xhat, '--')
xlabel('time')
ylabel('value')
legend('Gaussian process X', 'Wiener estimator Xhat')
title('Comparison between X and its estimator Xhat')
print -deps fig0.eps
unix('epstopdf fig0.eps');
figure(2)
hist(esterror, 100)
xlabel('estimated error')
ylabel('draws')
title(['Estimated error with mean=', num2str(avesterror), ' and MSE=', num2str(mse)])
print -deps fig1.eps
unix('epstopdf fig1.eps');