Lab 8: Fourier series: Gibbs phenomenon and filtering

1 Background

In class we used the Fourier theorem to construct a Fourier series representation of a periodic square wave. If the periodic square wave is written as an odd function, then the Fourier series is

\[ g(t) = \frac{1}{2} + \frac{2}{\pi} \sin t + \frac{2}{3\pi} \sin 3t + \frac{2}{5\pi} \sin 5t + \cdots. \]

Since odd numbers can be written as \(2n - 1\), where \(n\) is an integer, we can re-write this series as

\[ g(t) = \frac{1}{2} + \sum_{n=1}^{\infty} \frac{2}{(2n-1)\pi} \sin((2n-1)t). \tag{1} \]

There are two important properties of periodic square waves for bench top researchers. The first is Gibbs’s phenomenon. Figure 1 shows the Gibbs phenomenon that arises when we increase the number of terms in (1). Sharp transitions, such as the edge of the square wave, generate high frequency components in the Fourier series. Gibbs’s phenomena produces artifacts in JPG images (see Section 8.4) and also produces aliasing (see Lab 9) that makes the determination of power spectral descriptions of neural spike trains [?] and the EKG of the heart beat difficult.

The second important property concerns the relationship between square and sine waves. At first sight this does not seem to be that important. However, in electronics, communication signals are often transmitted as square wave pulses and on arrival to their destination are converted to sine waves. Moreover in an electronic circuit it is often easier to create a train of periodic square waves rather than to generate a sine wave. An example in biology arose in the measurement of the Bode plot for the hyperosmolar response of yeast cells (Sections 7.4.2 and
From (1) we see that a square wave can be thought of as a sine wave to which terms with higher frequencies have been added. Thus an attractive idea to generate a sine wave is to low-pass filter the train of periodic square waves to remove the higher frequency terms, leaving the pure sine wave component behind.

Figure 1: Comparison of a square wave to the (1) using 50 terms. The little spikes at the corners of the square wave (↓) represent the Gibbs phenomenon.

The purpose of this lab is to demonstrate Gibb’s phenomenon and then to use the convolution integral to filter the square wave to make a sine wave. It is important to keep in mind that this problem corresponds to the mathematical problem

\[
\text{output}(t) = \int_{-\infty}^{\infty} \text{impulse response}(u) \text{ input}(t - u) \, du,
\]

where the input is the periodic square wave and the impulse response is that for a RC circuit, namely,

\[
I(t, t') = ke^{-kt}.
\]

It is observed that at least a two-pole low pass filter is required.

In Lab 9 we will show that we can also accomplish these goals using the fast fourier transform (FFT). In this case we have

\[
B(f) = I(f) \text{ filter}(f).
\]

**Browser use:** We use the function `scipy.signal.square()` to generate a periodic train of square waves. The article written by R. Mark Stitt,
provides background into the importance of converting square waves into sine waves in industry. A number of animations which illustrate the act of convolution are available on the Internet. In particular, we recommend the Wikipedia-Convolution entry since it includes animations of the two convolutions studied in this Lab. Information concerning defining a function in Python and constructing a `while` loop can either be obtained on the Internet or by consulting `intro_python.pdf` on our website.

**House keeping:** It is useful to place the programs related to convolution into a directory labeled, for example, `convol`. This is because we will use these programs in later labs and hence it is advantageous to be able to locate them quickly.

## 2 Exercise 1: Gibbs phenomenon

It is often necessary to evaluate a sum in a computer program, such as occurs in (21). In Python sums can be evaluated using a `while` loop. For example to determine $y$ where

\[ y = \sum_{i=1}^{10} i^2 \]

we can write

```python
sum=0
n=1
while n <11:
    sum=sum+nˆ2
    n=n+1
print sum
```

It is possible to use a `for` loop in Python; however, a `while` loop construct is typically much faster.

**Questions to be answered:**

1. Write a Python program using a `while` loop to compare $g(t)$ to a periodic square wave as a function of the number of terms $n$. Compare the solution to a periodic $2\pi$-periodic square wave.
2. Rewrite your program by constructing a function that evaluates $g(t)$ for an inputted number of terms.

3. Modify this program by defining a function so that you can easily plot $g(t)$ for different numbers of $n$ on the same graph. This plot is useful to see the emergence of the square waves as the number of terms in the sum increases.

4. What happens to the Gibbs phenomenon when $n$ becomes large?

3 Exercise 3: Convolution: Graphical interpretation

Note: This section has been moved to Lab 7.

4 Exercise 4: Making a sine wave from a square wave

A sine wave can be created by low-pass filtering a square wave. It is necessary to use at least a two pole low-pass filter to make wave that looks sinusoidal. A RC circuit corresponds to a one pole low-pass filter. Thus it should not be surprising that one way a sine wave is created from a square wave is to pass the square wave signal through two consecutive RC-type circuits. Let’s use the convolution programs we developed in Lab 7 to see how well this works.

We give you two useful hints:

1. A useful way to use scipy.signal.square() to generate a periodic train of square waves is to write

   ```python
t = np.linspace(0, 1, 100, endpoint=False)
plt.plot(t, signal.square(2 * np.pi * 3 * t))
```

To illustrate, these lines of code have been written to generate a 3 Hz periodic train of square waves sampled at 100 Hz for 1 s.

2. The cut-off frequency for a RC circuit whose impulse response is $ke^{kt}$ is $1/k$. 
Figure 2: a) An input signal consisting of a 5 Hz periodic train of square waves. b) the output signal after passing through a one pole low-pass filter, and c) the output signal after passing through two consecutive one pole low-pass filters. Note that the signals shown correspond to steady state solutions, i.e., a large number of transient have been allowed to die out. It should be noted that using a one pole low-pass filter that it is possible to get other output waveforms. We have used parameters that generate the “most sinusoidal shaped one”.

Questions to be answered

1. Write a computer program that uses the convolution program in Lab 7 to generate a sine wave from a square wave. remember to include the modification to the convolution program that we discovered in Exercise 3. The basic idea is to convolve the periodic square wave train with a RC filter and then convolve the output with a second RC circuit. Figure 2 shows our attempt at this strategy. Clearly it is clearly better to use two RC circuits, namely a two-pole low-pass filter, in order to get an output that resembles a sine wave. In our case we choose \( k \) for each RC circuit to be the same. Can you do better by choosing different \( k \)’s for each low-pass filter?

2. According to the principle of linear superposition one would expect that the
result of passing a signal through a two-pole filter would be the same as passing a signal through two consecutive one-pole filters having the same poles. To test this hypothesis consider the impulse function for a second-order over damped differential equation (Equation 6 in Lab 7). Pick two poles, $\lambda_1$ and $\lambda_2$ and write a computer program to convolve the square wave used in the previous question with this impulse response. Compare this result to that obtained by passing the square wave signal through two consecutive RC circuits, the first with pole $\lambda_1$ and the second with pole $\lambda_2$. Are they the same?

**Deliverables:** Use Lab8_template.tex to prepare the lab assignment.

**References**