ECE2111 laboratory 2: Convolution

Prepared by: Dr. James Saunderson

Aim

The aims of this lab are

1. to understand how discrete-time linear time-invariant systems are described by their impulse response
2. to implement an LTI system via convolution with the impulse response
3. to implement an echo effect and a system that simulates playing music in rooms with different acoustic properties
4. to implement simple image filters for edge enhancement and denoising via convolution

Introduction

We have seen that a discrete-time linear time-invariant system is completely determined by its response to the discrete-time unit impulse

$$\delta[n] = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0. \end{cases}$$

The output of the system, when the input is δ, is called the *impulse response*. The impulse response is a signal. It is a notational convention to use the notation h for the impulse response.

Given the impulse response of a system, there is a formula relating the input x of the system to the output y of the system. This is the *convolution formula*

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k].$$

If x and h have finite duration, then this infinite sum simplifies to a finite sum.

Scheduling

This lab runs over two weeks (weeks four and five).

Prelab

There are three prelab questions throughout this lab document. By the end of week 3 (for the precide due date see Moodle), read through the lab document, find the prelab questions, and answer them.

Submit your answers to the prelab questions via the Moodle quiz called 'prelab2' on the Moodle page. You are expected to do this individually.

Results document

You are required to organize your code and outputs in what we will call a "results document". This must be created following the guidelines in the accompanying file "Formatting requirements for lab results document". Submit this by the end of week 5 (for the precise due date see Moodle) via the 'results document' assignment link on Moodle. **Your submission must reflect your own work!**

End-of-lab quiz

There is a timed, end-of-lab quiz that must be completed by the end of week 5 (for the precise due date see Moodle). **Please do not start this quiz until you are ready.** This quiz tests your understanding of the lab material. **It must be completed individually.**

1 Implementing an LTI system via convolution

Recall from lab 1 that we represent a discrete-time signal x via two row vectors. One of these, `nx`, gives the time instants at which the signal is (possibly) non-zero. The other, `x`, has the same length as `nx` and specifies the value of the signal at the time instants defined by `nx`.

Recall from lectures that the MATLAB `conv()` function takes two row vectors `x` and `h` and returns a third row vector `y`. The MATLAB implementation assumes that if `x` has length L then `x` represents the signal

$$x[n] = \begin{cases} 0 & \text{if } n < 0 \\ \texttt{x(n+1)} & \text{if } 0 \leq n \leq L-1 \\ 0 & \text{if } n \geq L \end{cases}$$

and similarly for `h` and `y`.

1.1 Prelab question 1

Let x be a discrete-time signal such that $x[n] = 0$ for $n < a_x$ and $n > b_x$. Let h be a discrete-time signal such that $h[n] = 0$ for $n < a_h$ and $n > b_h$. Let $y = x * h$ be the convolution of x and h. Let a_y be the largest integer, and b_y the smallest integer, such that $y[n] = 0$ for $n < a_y$ and $n > b_y$.

(a) Find an expression for a_y in terms of a_x and a_h.

(b) Find an expression for b_y in terms of b_x and b_h.

In the Moodle quiz for prelab question 1, enter the values of a_y and b_y if $a_x = 1, b_x = 5, a_h = -1$, and $b_h = 3$.

1.2 Activities

In this part of the lab you will write a MATLAB function called `myconv` that implements the convolution of two discrete-time signals.

1. Write a MATLAB function `myconv()` that takes four row vectors `x` and `nx` (representing a DT signal x) and `h` and `nh` (representing a DT signal h) as input and returns two row vectors `y` and `ny` that represent the DT signal $y = x * h$. Assume that `nx = ax:bx` for integers `ax` and `bx` and that `nh = ah:bh` for integers `ah` and `bh`.

   Use your answer to the prelab to help you figure out how to define `ny` in terms of `nx` and `nh`. Feel free to use MATLAB's built-in `conv()` function within your function, or to write your own implementation of the convolution formula from scratch.

2. Your implementation of the convolution formula should satisfy basic properties of convolution. Test your implementation by writing a MATLAB script that carries out the following tasks:

   (a) Use the function call `[x, nx] = dtimpulse(0,0,0);` to construct the DT unit impulse (the function `dtimpulse()` is from lab 1).

   (b) Let `h = [1:1:3, 3:-1:1];` and let `nh = -2:3;`.

   (c) What does `[y, ny] = myconv(x,nx,h,nh)` return? Output the result to the command window (by leaving the semicolon off the line of your script). Why do you expect this to happen?

   (d) What relationship do you expect between `myconv(x,nx,h,nh)` and `myconv(h,nh,x,nx)`?

(e) Construct the signal defined by `x = [1 1 1];` and `nx = 2:4;`. Using the signal defined by `h` and `nh` from part (b), output both `[y1, ny1] = myconv(x,nx,h,nh)` and `[y2, ny2] = myconv(h,nh,x,nx)` to the command window.

Once you have finished this section:

- copy your code (your function `myconv` and your script for item 2) into your results document
- copy the output of your script from part 2 into your results document.
- Answer the questions in 2(c) and 2(d) in you results document.

2 Modelling an echo effect as a discrete-time LTI system

One context in which discrete-time LTI systems are used is to model sound effects. In this part of the lab you will develop the impulse reponse corresponding to a simple 'echo' effect.

Consider a discrete-time LTI system with impulse response of the form:

$$h[n] = \delta[n] + \alpha\delta[n - D]$$

where D is a positive integer and $0 < \alpha < 1$. If the input to this system is a signal x, the output is the the signal

$$(x * h)[n] = x[n] + \alpha x[n - D].$$

So the system output consists of x itself, together with an echo, which is scaled down by a factor of α.

2.1 Prelab question 2

Suppose the sampling frequncy of our audio file is F_s Hz and we want an inter-echo delay of τ seconds. Find a formula for D in terms of F_s and τ. In the prelab Moodle quiz, enter the value of D if the sampling frequency is $F_s = 8192$ Hz and the desired delay is $\tau = 0.75$ seconds.

2.2 Activities

1. Write a MATLAB function `echoIR(D,alpha)` that creates an impulse response with given inter-echo delay `D` (in samples), given amplitude decay `alpha` (scalar between 0 and 1). The output should be a pair of row vectors of length $D + 1$, one corresponding to the signal values, and another to the associated time indices.

2. Write a MATLAB script that does the following:

   (a) Let `D` be the number of samples you found in prelab question 2, and let `alpha = 0.5` Construct the corresponding impulse response using your function `[h, nh] = echoIR(D,alpha)`.

   (b) Make a stem plot of the impulse response for times $n = 0, 1, 2, \ldots, D$.

   (c) Enter `load handel;` (see lab 1) to load a particular audio signal. Recall that the signal `y` in your workspace is the corresponding sound signal. Let `ny = 0:(length(y)-1);` be the corresponding vector of time indices.

   (d) Using the function `myconv()` you wrote in part 1 of this lab, compute the output of the discrete-time LTI system with *input* given by the signal `y` and impulse reponse `h`. Call the output `yecho`.

   (e) Play the output using `soundsc()` (this is a version of the command `sound()` that rescales the inputs to have values between -1 and 1.) What happens if you increase or decrease the parameters `D` and `alpha`?

   (f) Pass the signal `yecho` through the echo system again and call the output `yecho2`. This is equivalent to taking the signal `y` and passing it through the series composition of two echo systems. Play the resulting sound using `soundsc()`. How many echoes do you hear?

Once you have finished this section:

- copy your code (your function `echoIR()` and your script for item 2) into your results document
- copy your plot from item 2(b) into your results document.
- Briefly answer the questions in part 2(e) and 2(f) in your results document.

3 Implementing realistic reverb effects via convolution

In this part of the lab you will implement a sound effect that makes a given audio signal sound as if it were played in a gymnasium and then as if it were played in a cave. You will do this by using impulse responses measured from data. Such impulse responses are obtained by playing an impulse-like sound (such as a recording of a starter's pistol in an anechoic chamber) in a room, and then measring the resulting signal.

The data we use in this section have been obtained from 'The Open Acoustic Impulse Response Library'[1] The audio files are `.wav` files. If we had a file called `mywavfile.wav` we could load it into MATLAB by running the following code:

```
[y, Fs] = audioread('mywavfile.wav');
```

This creates a vector `y` consisting of audio sample values (scaled between -1 and 1) as well as returning the sampling rate `Fs`. Please pay attention to the sampling rates during this section, especially when using the command `soundsc()`.

3.1 Activities

Write a MATLAB script to carry out the following tasks:

1. Load an audio file corresponding to a trumpet sound played in an anechoic chamber (`trumpet.wav`). Name the signal `trumpet`. Play this sound. What is the sampling frequency? What is the length of the signal? [2]

2. Load the impulse response for a system that represents the reverberation in the sports centre at the university of York (`sportscentre.wav`). Name the impulse response `hSports`. What is the sampling frequency? [3]

   (a) Use `conv()` to alter the signal `trumpet` to sound as though it is being played in the sports centre at the University of York. Call the output signal `trumpetSports`. This will take quite some time to run (a few minutes). This is because we are working with signals sampled at a high sampling frequency, so the signals are very long.

   A couple of comments to keep in mind:

   - It turns out that there are significantly faster algorithms for computing the convolution of two long sequences, based on an algorithm called the *fast fourier transform (FFT)*. We will briefly discuss this later in the unit.
   - Once you have computed `trumpetSports` once, and correctly, **save the result** using the `save` command, and don't recompute it ever again. (This will save you lots of frustration waiting around.) If you need it later, load it again using the `load` command.

   (b) Play the resulting output sound. (Hint, use the command `soundsc()` rather than `sound` to automatically rescale the sound so that it takes values between -1 and 1 before playback. Also, make sure you specify the correct sampling frequency. Read `help soundsc` for more explanation.)

[1] The url for the database is currently down (August 2020) but contact details for the project are available at `https://pure.york.ac.uk/portal/en/datasets/openair--the-open-acoustic-impulse-response-library(c2fec4e3-7e29-4ca9-a9b2-875494fc311e).html`

[2] The original file was once available at `http://www.openairlib.net/sites/default/files/anechoic/data/helena-daffern/modern-piccolo-trumpet/mono/tr-1967-piece7-t-32.wav`

[3] The original file was once available at `http://www.openairlib.net/sites/default/files/auralization/data/audiolab/sports-centre-university-york/mono/sportscentre_omni.wav`

(c) Using the `subplot` command, plot both `trumpet` and `trumpetSports` on separate axes, one above the other. (Use `plot` rather than `stem` because otherwise the plots look a bit messy!) Use `xlim()` to ensure that both plots have the same limits for the horizontal axis, to aid comparison. Make sure you label the axes appropriately.

3. Load the impulse response for a system that represents the reverberation in a cave (`cavemono.wav`). Name the impulse response `hCave`.[4]

   (a) Use `conv()` to alter the signal `trumpet` to sound as though it is being played in a small cave. Again this will take quite some time to run. Call the output signal `trumpetCave`.

   (b) Play the resulting output sound using `soundsc()`.

   (c) Using the `subplot` command, plot both `trumpet` and `trumpetCave` on separate axes, one above the other. Use `xlim()` to ensure that both plots have the same limits for the horizontal axis, to aid comparison. Make sure you label the axes appropriately.

Once you have finished this section:

- copy your MATLAB script into your results document.
- copy your plots from item 2(c) and 3(c) into your results document.

[4] This file is modified from the original, which was once available at http://www.openairlib.net/sites/default/files/auralization/data/fstevens/creswell-crags/b-format/2_r_rhcbottom_s_rhc_mouth.wav

4 Images in MATLAB

MATLAB makes it convenient to load, manipulate, and display images. We will use this functionality throughout the labs. In MATLAB, a grayscale image that is q pixels wide and p pixels high can be represented by

- a $p \times q$ matrix with entries of type `double` that take values between 0 (black) and 1 (white); or
- a $p \times q$ matrix with entreis of type `uint8` (8 bit unsigned integer) that take values between 0 (black) and 255 (white).

MATLAB can also represent and work with colour images by using three different matrices (for R and G and B values, for instance). We do not consider colour images in this lab.

We can display an image represented by a matrix `X` with entries of type `double`, by using the command

```
imshow(X);
```

Any entry of `X` with value larger than 1 is displayed as white, and any entry with value smaller than 0 is displayed as black. The alternative command

```
imshow(X,[]);
```

automatically rescales the matrix `X` before display so that the smallest entry of `X` displays as black and the largest entry displays as white. Type `help imshow` for more information and other options.

Given an image file, stored in JPEG format (for example), with filename `filename.jpg`, we can load it into MATLAB using the following commands:

```
X = imread('filename.jpg','jpeg');
X = double(X)/255;
```

the first line reads the image into the matrix `X`. The second line converts the matrix `X` from consisting of entries of type `uint8` with values between 0 and 255, to a matrix with entries of type `double` with values between 0 and 1. This later type is a little more convenient for manipulation (even though it does require more memory).

4.1 Prelab question 3

If we run the MATLAB code

```
X = [0.5  2; -0.5  1];
imshow(X);
```

how many pixels will display as white?

4.2 Activities

Write a MATLAB script to carry out the following tasks:

1. Copy the file `echart.mat` from the ECE2111 Moodle page into your working directory. Enter

   ```
   load echart;
   ```

   to load a matrix called `echart` into your workspace.

2. Display the image represented by the matrix `echart` using the command `imshow()`.
3. How many pixels wide and how many pixels high is the image?
4. Add noise to the image by using the command

```
echartnoisy = echart + 0.8*rand(size(echart));
```

5. Display the image `echartnoisy` using `imshow(echartnoisy);`
6. Display the image `echartnoisy` using `imshow(echartnoisy,[]);`
7. What is the difference between the displayed images in items 5 and 6?

Once you have finished this section:

- copy your MATLAB script into your results document
- copy the images you displayed from items 2, 5, and 6 into your results document
- answer the questions in items 3 and 7 in your results document.

5 Image processing via 2D convolution

So far we have implemented discrete-time LTI systems for *one-dimensional* signals (like audio) using convolution with a one-dimensional impulse response. We can use these systems to process *two-dimensional* signals (like images) by regaring each row (or column) of the image as a one-dimensional signal, and passing each row (or column) through the system.

For example, if `hrow` is the impulse response of an FIR discrete-time LTI system, we can apply the system to all the rows of an image represented by a $p \times q$ matrix `X` with the MATLAB code:

```
Y = zeros(size(X,1),size(X,2)+length(hrow)-1);
for i=1:size(X,1)
    Y(i,:)  = conv(hrow,X(i,:));
end
```

The ith row of `Y` contains the output of the system when the input is the ith row of `X`, and so we filters each row of the image (separately). The first line of the code just initializes `Y` to be the zero image of the right size. (Why is this the right size?) This speeds up the code because the memory required for `Y` can be preallocated.

We could then filter each *column* of the resulting image `Y` by passing it through an LTI system with impulse response `hcol` (a row vector) as follows:

```
Z = zeros(size(Y,1) + length(hcol)-1,size(Y,2));
for j=1:size(Y,2)
    Z(:,j) = conv(Y(:,j)',hcol)';
end
```

The characters `'` on the third line convert between column and row vectors as appropriate.

We can apply LTI systems designed for one-dimensional signals to two-dimensional signals, by applying a system to each row (separately), and a system to each column (separately). Rather than using the two `for` loops, as in the example code above, MATLAB has a special built-in function called `conv2()` to do this. We can construct `Z` from `X` with the single command:

```
Z = conv2(hcol,hrow,X);
```

We could also filter just the rows of `X` with the code

```
Z = conv2(1,hrow,X);
```

Why do you think this works? If you read `help conv2` you will see that the function `conv2()` can perform a more general family of two-dimensional convolution operations. We will not use this additional functionality in the labs.

5.1 Activities

Write a MATLAB script to carry out the following tasks.

1. Let `h = [-1 1];` be the impulse reponse of a DT LTI system.

   (a) Filter just the rows of `echart` using `h` and `conv2()` to create a new image signal called `Yrow`.

   (b) Filter just the columns of `echart` using `h` and `conv2()` to create a new image signal called `Ycol`.

(c) Filter the rows and columns of `echart` using `h` (for both) and `conv2()` to create a new image signal called `Yboth`.

(d) Use `imshow(abs(Yrow),[]);` to display the image represented by the absolute value of `Yrow`. Repeat this to display the images represented by the absolute values of `Ycol` and `Yboth`.

(e) Based on the images you made in part (d), briefly explain how convolution of the rows and/or columns of the image with `h`, followed by an absolute value operation, enhances the edges of an image.

2. Let `echartnoisy` be the noisy signal from part 4 of section 4.2.

   (a) Filter the rows of `echartnoisy` using `h` and `conv2()` to create a new image signal called `Zrow`.

   (b) Use `imshow(abs(Zrow),[]);` to display the image represented by the absolute value of `Zrow`.

   (c) How does noise affect the result of the edge enhancement operation?

3. One way to ensure the system performs better in the presence of noise is to first 'blur' the image before passing it through the edge enhancement filter. Such 'blurring' is an example of low-pass filtering, a topic we will discuss more later in the unit. In image processing, a typical 'blurring' filter has impulse response given by sampling a Gaussian function

$$p(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-t^2/(2\sigma^2)}$$

at equally spaced points over some range. For example if we sample at integers over the range $-3\sigma \leq n \leq 3\sigma$, the impulse response is

$$h_\sigma[n] = \frac{1}{\sqrt{2\pi}\sigma} e^{-n^2/(2\sigma^2)} \quad \text{for } -3\sigma \leq n \leq 3\sigma.$$

   (a) Create signals `h1` and `h2` and `h3` that represent the signals h_1, h_2, and h_3 respectively (i.e., h_σ with $\sigma = 1, 2$, and 3).

   (b) Using the `subplot()` command, make stem plots of `h1`, `h2`, and `h3`, one above the other. What is the effect of changing σ?

   (c) Filter the rows and columns of `echartnoisy` using the impulse response `h1` (for both rows and columns) and the command `conv2()`. Repeat using `h2` nd `h3`. Call the resulting images `Zblur1`, `Zblur2` and `Zblur3` respectively and display these three images.

   (d) Now filter the rows of `Zblur1` using the impulse response `h` and the command `conv2()`. Display the absolute value of the resulting image. Repeat with the input image `Zblur2` and `Zblur3`.

   (e) How does first applying a Gaussian blur (with different values of σ) affect the performance of subsequent edge enhancement filtering?

Once you have finished this section:

- copy your MATLAB script into your results document.
- copy the images you displayed in items 1(d), 2(b), 3(c), and 3(d) into your results document.
- copy your plots from item 3(b) into your results document.
- answer the questions in items 1(e), 2(c), 3(b), and 3(e) in your results document.

Once you have completed the lab tasks and understand them, you are ready for the end-of-lab Moodle quiz. This quiz is closely based on the lab tasks. It must be completed **individually**.You may use MATLAB. Unlike the prelab, you only have **one attempt at each question**.

Assessment

This lab is marked out of 12. Your mark is based on the following:

- **Prelab:** Correct responses to the three prelab questions, submitted via the prelab Moodle quiz (3 marks, 1 per question)
- **Results document:** (4 marks, 3 marks for content and 1 mark for presentation)
  Marks per section: Each of the 5 sections is marked out of 0.6 (3 marks):
  - 0 marks if not attempted
  - 0.4 marks if a reasonable attempt is made, but has clear flaws
  - 0.6 marks if no errors or possibly very minor errors

  Presentation: (1 mark)
  - Results document adheres to the formatting requirements (1 mark)
  - Results document mostly adhere to formatting requirements, but not completely (0.5 mark)
  - Results document rarely adheres to formatting requirements (0 marks)
- **End-of-lab quiz:** Correct responses to the end-of-lab Moodle questions (5 marks, 1 per question).