MIDDLE EAST TECHNICAL UNIVERSITY (METU)
Department of Electrical and Electronics Engineering

EE 798 Theory of Remote Image Formation
Spring 2021

Problem Set 2

Issued: Tuesday, May 10, 2022 **Due:** 23:59, Sunday, May 22, 2022

Problem 2.1 (Model for a CCD detector)
In this problem we will model the signal measured by a CCD device. A charge-coupled device (CCD) is a modern light-density to electron-density converter that is widely used as an optical sensor.A CCD is divided into nonoverlapping cells called "pixels". In each cell, the total incident light intensity received in a time duration of Δt over the area of that cell is converted to a voltage measurement.

Let $h(x, y)$ describe the shape of a pixel (e.g. a properly scaled rectangle or circle function). Suppose $s(x, y)$ is the signal incident on the CCD detector.

a. Prove that signal measured by the CCD device can be modeled as the sampled version of

$$r(x, y) = h(-x, -y) \ast\ast\ s(x, y)$$

That is, CCD measurements can be modeled as $r(x, y)$ followed by a perfect uniform sampler with the samples centered on pixels.

b. Suppose $s(x, y) = \text{circ}\left(\frac{x}{a}, \frac{y}{a}\right)$ and the pixel shape is also $\text{circ}\left(\frac{x}{a}, \frac{y}{a}\right)$. The CCD is not necessarily aligned with the signal $s(x, y)$. Describe the CCD measurements and how the CCD can degrade the signal. (Hint: In this part, an analytical reasoning is asked, but if you cannot proceed with an analytical discussion you can provide your answer based on Matlab simulations.)

Problem 2.2 (Tomography with unique solution)
Suppose that $s(x, y)$ is a separable function:

$$s(x, y) = s_1(x)s_2(y)$$

Prove that $s(x, y)$ is uniquely determined by two of its projections, given the side information that it is separable.

Problem 2.3 (Adjoint of operators)
As we will learn a bit later in the term, adjoint of operators appear throughout reconstruction

and inverse theory. For example, the celebrated "backprojection" operation of tomography is based on an adjoint. In this problem we will study such adjoint operators. Given a linear operator $L : \mathcal{X} \mapsto \mathcal{Y}$ mapping the vector space $\mathcal{X}$ to the space $\mathcal{Y}$, the adjoint L^* is defined by the relationship:

$$< y, Lx >_{\mathcal{Y}} \quad = \quad < L^* y, x >_{\mathcal{X}} \tag{1}$$

for all x, y, where $< ., . >_{\mathcal{X}}$ denotes the inner product in the space $\mathcal{X}$ and $< ., . >_{\mathcal{Y}}$ denotes the inner product in the space $\mathcal{Y}$.

(a) Suppose a linear operator $L : \mathcal{X} \mapsto \mathcal{Y}$ is given by a real integral equation of the first kind of the form :

$$Lx(t) = \int_a^b K(t, \tau) x(\tau) d\tau \tag{2}$$

with $\mathcal{X} = L_2[a, b]$, $\mathcal{Y} = L_2[a, b]$ (i.e., L_2 space consisting of finite energy signals). In both the input and output spaces we use the usual inner product between two real functions $u(t)$, $v(t)$:

$$< u, v > \quad = \int_a^b u(t) v(t) dt \tag{3}$$

What is the corresponding adjoint operator to the linear operator L given in Eq. (2)? An operator is termed "self-adjoint" when it equals its adjoint. What property must the kernel satisfy for the linear operator to be self-adjoint?

(b) Now, we specialize our general results to the matrix case. Consider a linear operator L between finite dimensional spaces equipped with the standard inner product $< u, v >= u^T v$. This operator can always be represented by a matrix: $Lx(i) = Ax$ for some matrix A and vector x. Prove that in this case $L^* = A^T$, i.e. adjoint is given by the transpose of the matrix A. What form must A have in order to correspond to a self-adjoint operator?

(c) A very important example of Eq.(2) corresponds to the case of LSI filtering or convolution:

$$Lx(t) = \int_{-\infty}^{\infty} h(t - \tau) x(\tau) d\tau \tag{4}$$

What is the adjoint operator to convolution? What condition does the impulse response $h(t)$ have to satisfy for the corresponding convolution operation to be self-adjoint?

Problem 2.4 (Matrix representation of 1D convolution)
In inverse problems where the observed quantity can be viewed as the output of an LSI system, one is often lead to consider discrete representations with a corresponding convolutional

structure. In this problem we investigate how to model such discrete LSI inverse problems. You will write Python or Matlab code to generate the matrices that perform these operations and start to understand their structure. One-dimensional discrete convolutional problems are of the form:

$$y(i) = \sum_{j=1}^{L} h(i-j)x(j) = h(i) * x(i) \tag{5}$$

where the nonzero portion of $h(i)$ is of length P and that of $x(i)$ is of length L and * denotes linear convolution.

(a) Let y be the vector of $y(i)$ elements and x be the corresponding vector of $x(i)$ elements. The elements of y and x are related through 1D convolution with $h(i)$. Then what are the entries of the matrix C which models this linear convolution as $y = Cx$? Find it analytically. What special matrix form does it have? How is the size of C related to the lengths of the signal $x(i)$ and the filter impulse response $h(i)$? Use the Python function **scipy.linalg.toeplitz** or Matlab function **toeplitz** to generate the linear convolution matrix C for a given $h(i)$ and problem size. Generate a few such C matrices (be careful of the difference between row and column vectors when using the function). These matrices represent the kernel of linear convolution. View them as an image or mesh plot. Notice that they are constant along their diagonals, as they must be for a shift-invariant problem.

(b) Since (3) represents a convolution we know that Fourier techniques should be useful. In particular, since the problem is discrete, the appropriate tool is the discrete Fourier transform (DFT), which may be efficiently implemented using the FFT algorithm. But recall that the product of the DFT coefficients of two sequences actually corresponds to the DFT of the *circular* convolution of the two sequences. In general, what length-N circular convolution must be used to ensure that the circular convolution of $h(i)$ and $x(i)$ produce the same results as the linear convolution of these sequences? How are the corresponding sequences $\tilde{h}$ and $\tilde{x}$ used in the circular convolution related to h and x? Write a Python or Matlab function **cconv** to perform the N-point circular convolution of two sequences (Hint: Use the **numpy.fft.fft** routine in Phyton or **fft** routine in Matlab).

(c) What is the matrix $\tilde{C}$ that performs the N-point circular convolution of the sequence $h(i)$ with a length N vector (assuming $N > P$)? Find it analytically. What special matrix form does it have? How is it related to the linear convolution matrix C? Using your function **cconv**, write another Python/Matlab function **cconvmtx** to create $\tilde{C}$ for an arbitrary h and N. For this, you should notice that the columns of $\tilde{C}$ are equal to the circular convolution of h with the unit coordinate vectors. (The unit coordinate vectors are all zero except for a single 1, i.e. $e_1 = [1\,0\ldots0]$, $e_2 = [0\,1\,0\ldots0]$, and such.). Note that if N is chosen large enough we will have that $y = \tilde{C}\tilde{x}$. At this point you have represented the linear convolution output as an equivalent circulant convolution. We are now ready to apply the DFT.

(d) The DFT pair of a discrete sequence $x(n)$ of length N is most commonly defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}; k = 0, ..., N-1 \tag{6}$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N}; n = 0, ..., N-1 \tag{7}$$

As usual, let X denote the vector of coefficients $X(k)$ and x denote the vector of the points $x(n)$. Then X and x are related by a matrix F through $X = Fx$, i.e. the matrix F takes the DFT of a sequence. What are the entries of the matrix F - i.e. write out the general form of matrix F analytically. What is F^{-1} in terms of F? The Python function **scipy.linalg.dft** and MATLAB function **dftmtx** generate the N-point DFT matrix F for a given value of N.

If your function **cconv** is correct, then for any two length-N column vectors x and y you should have that $\mathbf{cconv}(\mathbf{x}, \mathbf{y}, \mathbf{N})$ is equal to `np.linalg.inv(F) @ ((F @ x) * (F @ y))` in Phyton or $F^{-1} * ((F * x). * (F * y))$ in Matlab, i.e. the time and frequency domain results should match. Verify this for your function.

(e) Let $Y = Fy$, $\tilde{H} = F\tilde{h}$ and $\tilde{X} = F\tilde{x}$ be vectors of DFT coefficients. Using the fact that $y = Cx = \tilde{C}\tilde{x}$, where $\tilde{C}$ represents a circular convolution, together with the relationship between the DFT and circular convolution, show that the matrix $\tilde{\mathbf{C}} = F\tilde{C}F^H/N$ is a diagonal matrix, where F^H denotes the complex conjugate transpose of F. It is a general fact that such circulant matrices are diagonalized by the DFT. In terms of operations on the rows and columns of $\tilde{C}$ what does $F\tilde{C}F^H/N$ represent? Verify these relationships for a numerical example, i.e. show that $y = \tilde{C}\tilde{x} = \tilde{h} \odot \tilde{x}$ (where $\odot$ is circular convolution), $Fy = (F\tilde{C}F^H/N)F\tilde{x} = \tilde{H}. * \tilde{X}$, and that $F\tilde{C}F^H/N$ is indeed diagonal.

(f) What is the relationship between the elements on the diagonal of $\tilde{\mathbf{C}}$ and the DTFT of the original impulse response h? How are these values related to the DFT of the first column of $\tilde{C}$? (Try an experiement). How are the diagonal elements of $\tilde{\mathbf{C}}$ related to the eigendecomposition of the circular convolutional matrix $\tilde{C}$? What must the eigenfunctions of $\tilde{C}$ be?

(g) In what way would things change if the problem were *not* shift-invariant?

Problem 2.5 (Separable convolution)
In this problem we will study systems described by *separable* convolution. That is, problems in which the convolutional kernel can be expressed as the product of functions involving each coordinate direction alone. In particular, consider the general 2-D convolution of an $L_1 \times L_2$

image $x(n_1, n_2)$ with a $P_1 \times P_2$ convolutional kernel $h(n_1, n_2)$:

$$y(n_1, n_2) = \sum_{k_1=1}^{L_1} \sum_{k_2=1}^{L_2} h(n_1 - k_1 + 1, n_2 - k_2 + 1)x(k_1, k_2) \tag{8}$$

In separable problems the convolutional kernel can be expressed as follows:

$$h(n_1, n_2) = h_1(n_1)h_2(n_2)$$

where $h_1(n_1)$ is a 1-D convolutional kernel along the first index (i.e. the "y" direction) and $h_2(n_2)$ is a 1-D convolutional kernel along the second index (i.e. the "x" direction). As a result, for these problems we can write:

$$y(n_1, n_2) = \sum_{k_2=1}^{L_2} h_2(n_2 - k_2 + 1)[\sum_{k_1=1}^{L_1} h_1(n_1 - k_1 + 1)x(k_1, k_2)] \tag{9}$$

Thus, for separable problems, the 2-D convolution is equivalent to a 1-D convolution along each column followed by a 1-D convolution along each row.

(a) Write a Python/Matlab function **sepconv2** to implement separable kernel convolution which exploits the separable structure of the problem. Inputs to the function are the 2-D input image (i.e. matrix) X and the 1-D convolutional kernels for the individual coordinate directions h_1 and h_2 and the output is the resulting 2-D convolved image (i.e. matrix) Y.

Verify your code using the "cameraman" image and the following example convolution kernel:

$$h(n_1, n_2) = \frac{1}{P_1 P_2} \quad 1 \le n_1 \le P_1, \quad 1 \le n_2 \le P_2$$

This kernel is often used as a separable approximation of the kernel for the out-of-focus blur in image restoration. The cameraman image can be obtained in Python using the **scikit-image** library:

```
from skimage import data
X = data.camera()
```

or in Matlab using I = imread('cameraman.tif'). Note: You can use the 1D convolution routine **scipy.signal.convolve** in Phyton or **conv** in Matlab.

(b) Not only can we exploit the separable structure in performing the convolution, but we can also use it to simply obtain the corresponding convolution matrix in Python/Matlab.

Let x and y be vectors representing the 2-D images (i.e. matrices) X and Y through lexicographic ordering. Further, let H_1 be the matrix that performs the 1-dimensional

linear convolution of a length L_1 sequence with the kernel $h_1(n)$. Similarly, let H_2 be the matrix that performs the 1-dimensional linear convolution of a length L_2 sequence with the kernel $h_2(n)$. For such separable convolutional problems it is then straightforward to show based on Eq. (9) that the following matrix relationship holds :

$$Y = H_1 X H_2^T$$

From linear algebra we additionally have that the following matrix-vector relationship holds for arbitrary comparable matrices A, B, and X:

$$\text{vec}(AXB^T) = (B \otimes A)\text{vec}(X)$$

where $\otimes$ denotes the Kronecker product of two matrices and vec denotes the vector obtained from a matrix by stacking its columns.

Use these relationships to write a Python/Matlab function **sepconvmtx2** that generates the overall convolution matrix C for separable problems given the separable components of the impulse response $h_1(n_1)$, $h_2(n_2)$ and the input image dimensions L_1, L_2. Verify this function by comparing its convolution output obtained through matrix multiplication to that obtained from **sepconv2** for a small scale example.