Robert Heeter
ELEC 378
1/27/2023

Problem Set #2

1. a.

$$\underline{s}_k = \begin{bmatrix} s_k[0] \\ s_k[1] \\ \vdots \\ s_k[N-1] \end{bmatrix} \leftarrow k^{th} \text{ harmonic sinusoid in } N \text{ dimensions}$$

$$s_k[n] = \frac{1}{\sqrt{N}} e^{j \frac{2\pi k}{N} n} \quad \text{for } k = 0, \dots N-1$$

To form a basis for $\mathbb{C}^N$ (complex vector space in N dimensions), a collection of vectors $v \in \mathbb{C}^N$ must be linearly independent and span $\mathbb{C}^N$ (from slides)

To show that the harmonic sinusoids form an orthogonal basis $\{s_k\}_{k=0}^{N-1}$ for $\mathbb{C}^N$, the inner product should be used:

$\langle s_k, s_\ell \rangle = 0$ for $k \neq \ell$ → this indicates s_k and s_ℓ are linearly independent for any s_k and s_ℓ $(0 \leq \ell < k \leq N-1)$

Given $\langle u, v \rangle = \sum_{n=0}^{N-1} u[n] v[n]^*$: ← complex conjugation →

$$\text{Here: } \langle s_k, s_\ell \rangle = \sum_{n=0}^{N-1} s_k[n] s_\ell[n]^* = \sum_{n=0}^{N-1} \left(\frac{1}{\sqrt{N}} e^{j \frac{2\pi k}{N} n} \right) \left(\frac{1}{\sqrt{N}} e^{-j \frac{2\pi \ell}{N} n} \right)$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \left(e^{j \frac{2\pi n}{N} (k-\ell)} \right) \leftarrow e^{ix} = \cos(x) + i\sin(x) \text{ (Euler's formula)}$$

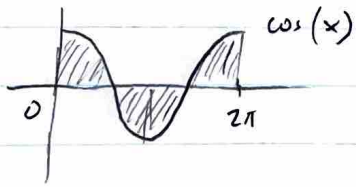
$$= \frac{1}{N} \sum_{n=0}^{N-1} \cos\left(\frac{2\pi n}{N} (k-\ell) \right) + \dots$$

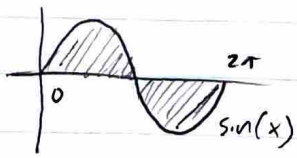
$$+ \frac{1}{N} \sum_{n=0}^{N-1} i\sin\left(\frac{2\pi n}{N} (k-\ell) \right)$$

Note that $\cos(\dots)$ and $i\sin(\dots)$ are summed from $n=0 \rightarrow n=N-1$; this is equivalent to:

$$= \frac{1}{N} \underbrace{\sum_{n'=0}^{2\pi - \frac{2\pi}{N}} \cos\left(n' (k-\ell) \right)}_{①} + \frac{1}{N} \underbrace{\sum_{n'=0}^{2\pi - \frac{2\pi}{N}} i\sin\left(n' (k-\ell) \right)}_{②}$$

When summed from $0 \rightarrow 2\pi$ (1 period), cosine and sine $= 0$:





Therefore, $\langle s_k, s_\ell \rangle = \frac{1}{N}(0) + \frac{1}{N}(0) = 0$
① ②

Continued on next page

(1). (a). Because $\langle s_k, s_\ell \rangle = 0$ for any $k, \ell = 0, 1, \ldots N-1$ (and $k \neq \ell$), the harmonic sinusoids form an orthogonal basis $\{s_k\}_{k=0}^{N-1}$ for $\mathbb{C}^N$

b. For a vector $\underline{x}$ having signal representation in this basis:

$a_k = \langle x, s_k \rangle \longrightarrow a_k$ is analysis weights

$$a_k = \langle x, s_k \rangle = \sum_{n=0}^{N-1} x[n], s_k[n]^* = \sum_{n=0}^{N-1} (x[n]) \left(\frac{1}{\sqrt{N}} e^{j \frac{2\pi k}{N} n} \right)$$

$$\boxed{a_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x[n] e^{j \frac{2\pi k}{N} n}}$$

($e^{j \frac{2\pi k}{N} n}$: sinusoids)

This equation looks familiar to that of a Fourier transform

$$\underline{x} = a_0 \underline{s}_0 + a_1 \underline{s}_1 + \ldots + a_{N-1} \underline{s}_{N-1}$$

$$\underline{s}_0 = \begin{bmatrix} s_0[0] \\ \vdots \\ s_0[n] \end{bmatrix} \qquad \underline{s}_{N-1} = \begin{bmatrix} s_{N-1}[0] \\ \vdots \\ s_N[n] \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{N}} \\ \vdots \\ \frac{1}{\sqrt{N}} e^{j \frac{2\pi k}{N} (n)} \end{bmatrix}$$

2. Next page

2. Every real, symmetric matrix is diagonalizable:

If $B = B^H$, then $\begin{cases} W \rightarrow \text{unitary matrix of orthonormal eigenvectors} \\ D \rightarrow \text{diagonal matrix of eigenvalues} \end{cases}$

and $B = WDW^H$

a. Given $B_1 = AA^H \rightarrow B_1^H = (AA^H)^H = (A^H)^H A^H = AA^H = B_1$ ✓

Because $B_1 = B_1^H$, B_1 must be symmetric. Because B_1 is symmetric, it must have an orthonormal basis of eigenvectors and has eigendecomposition $B_1 = WDW^H$ where D is a diagonal matrix of eigenvalues and W is a unitary matrix of orthonormal eigenvectors.

$\boxed{B_1 = WDW^H}$

b. Given $A = U\Sigma V^H$ [From slides] →

- $A \rightarrow$ real or complex matrix
- $U \rightarrow$ unitary matrix containing eigenvectors of AA^H (complex)
- $\Sigma \rightarrow$ diagonal matrix containing singular values (square roots of eigenvalues of AA^H and A^HA; real & non-neg.)
- $V^H \rightarrow$ unitary matrix containing eigenvectors of A^HA (complex)

Note that a unitary matrix is one such that $A^HA = AA^H = I$.

$$B_1 = AA^H = (U\Sigma V^H)(U\Sigma V^H)^H = (U\Sigma V^H)(V^H)^H\Sigma^H U^H$$
$$= U\Sigma V^H V\Sigma^H U^H = U\Sigma I\Sigma^H U^H$$
$V^HV = I$ b/c V is unitary
$$= U\Sigma\Sigma^H U^H$$
$= \Sigma^2$ b/c Σ is diagonal, so $\Sigma = \Sigma^H$ given Σ has real, non-neg. elements
$$= U\Sigma^2 U^H$$

$B_1 = WDW^H = U\Sigma^2U^H \rightarrow \therefore U = W$ ← W is matrix of orthonormal eigenvectors of B_1

$D = \Sigma^2$

$\boxed{U = W}$

Continued on next page

(2).(b). U exists for every A, since any complex matrix A multiplied by its Hermitian transpose must be real, so $B_1 = AA^H \rightarrow B$ must be real and symmetric, so $B_1 = WDW^H = U\Sigma^2 U^H$. U contains the eigenvectors of $B_1 = AA^H$

c. Given $B_2 = A^HA \rightarrow B_2^H = (A^HA)^H = A^H(A^H)^H = A^HA = B_2$ ✓

Because $B_2 = B_2^T$, B_2 must be symmetric. Because B_2 is symmetric, it must have an orthonormal basis of eigenvectors and has eigendecomposition $B_2 = WDW^H$ where D is a diagonal matrix of eigenvalues and W is a unitary matrix of orthonormal eigenvectors

$$\boxed{B_2 = WDW^H}$$

d. Given $A = U\Sigma V^H$ as in part B.

$$B_2 = A^HA = (U\Sigma V^H)^H(U\Sigma V^H) = (V^H)^H\Sigma^H U^H U\Sigma V^H$$
$$= V\Sigma^H U^H U\Sigma V^H = V\Sigma^H\Sigma V^H$$
$U^HU = I$ b/c U is unitary
$$= V\Sigma^H\Sigma V^H$$
$= \Sigma^2$ b/c Σ is diagonal, so $\Sigma^H = \Sigma^H$ given Σ has real, non-neg elements
$$= V\Sigma^2V^H$$

$B_2 = WDW^H = V\Sigma^2V^H \rightarrow \therefore V = W$ ← W is matrix of orthonormal eigenvectors of B_2

$D = \Sigma^2$

$$\boxed{V = W}$$

V exists for every A, following the same reasoning as in part B; B must be real and symmetric, so $B_2 = WDW^H = V\Sigma^2V^H$. V contains the eigenvectors of $B_2 = A^HA$.

e. From parts B and D: ← D is the diagonal matrix of eigenvalues of B_1 or B_2, so Σ is the diagonal matrix of the square root of the eigenvalues of B_1 or B_2

$D = \Sigma^2 \rightarrow \boxed{\Sigma = D^{1/2}}$

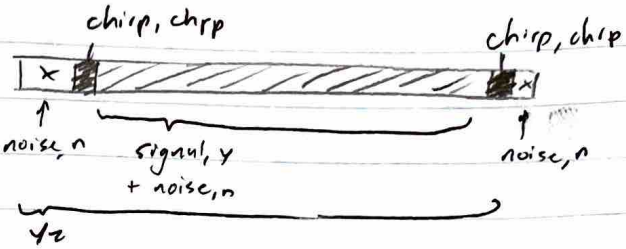
Continued on next page.

(2). (e). Σ must exist for every A, since for every A, B_1 or B_2 are real and symmetric following the reasoning in part B, so B_1 or B_2 must have a diagonal matrix of eigenvalues of D where $\Sigma = D^{1/2}$.

3. Next page

3. a. $x(t) + n(t) = y_2(t)$

↑ sent signal ↑ noise signal ↑ received signal



Convolution:

$$y[n] = \sum_{k=-\infty}^{+\infty} h[k]\, x[n-k] = \sum_{k=-\infty}^{+\infty} h[k]\, \hat{x}_n[k] = \langle h, \hat{x}_n \rangle$$

$x[n-k]$: time-reversed version of x shifted by n samples $= \hat{x}_n[k]$

n^{th} sample of convolution is inner product b/w h and time-reversed x, shifted by n samples

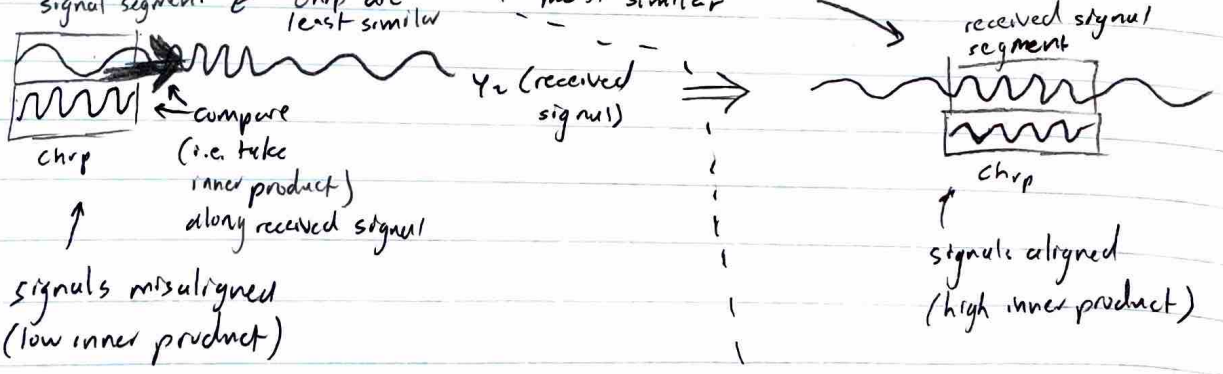
Allows efficient computation of inner products

The pre-processing system must extract the desired signal (+ noise), $y(t)$, from the received signal $y_2(t)$, shown in the diagram above. A convolution can be performed between the received signal and the known chirp signal to identify where the chirp signal lies in the received signal. A convolution is a method of iteratively calculating the inner product between two signals; here, a convolution is used to calculate the inner product between the chirp signal and every equivalent-length signal segment along the received signal. Following the Cauchy-Schwartz inequality, when the chirp signal aligns with a segment of the received signal, the inner product is large:

$$0 \leq |\text{chirp} \cdot y_2| \leq \|\text{chirp}\|_2 \, \|y_2\|_2$$

↑ lower bound when y_2 and chirp are least similar

↑ upper bound when chirp and y_2 are most similar



(where the chirps on both signals are aligned)

(3). (a). Along the received signal, there should be two indices where the inner product between chirp and the received signal are large, indicating the beginning and end of the received signal. The received signal can be trimmed and decoded following the process in Problem set #1. Also note that for the convolution, the received signal should be time-reversed following the equation above, and (I believe) the convolution actually operates by "padding" the regions before and after the chirp signal with zeros as it iterates along the received signal (such that both have the same length) so the received signal out-of-frame has no contribution to the inner product calculations.

b. See code and output. The system decodes the corrupt signal correctly. A graph of the convolution magnitude vs. index along the received signal is shown as well, with the two peaks where the chirp signal and chirp on the received signal are aligned. The decoding of the trimmed received signal ($y(t)$) was copied from Problem set #1.

ELEC378-HW2

January 27, 2023

[1]:
```
# ROBERT HEETER
# ELEC 378 Machine Learning
# 27 January 2023

# PROBLEM SET 2
```

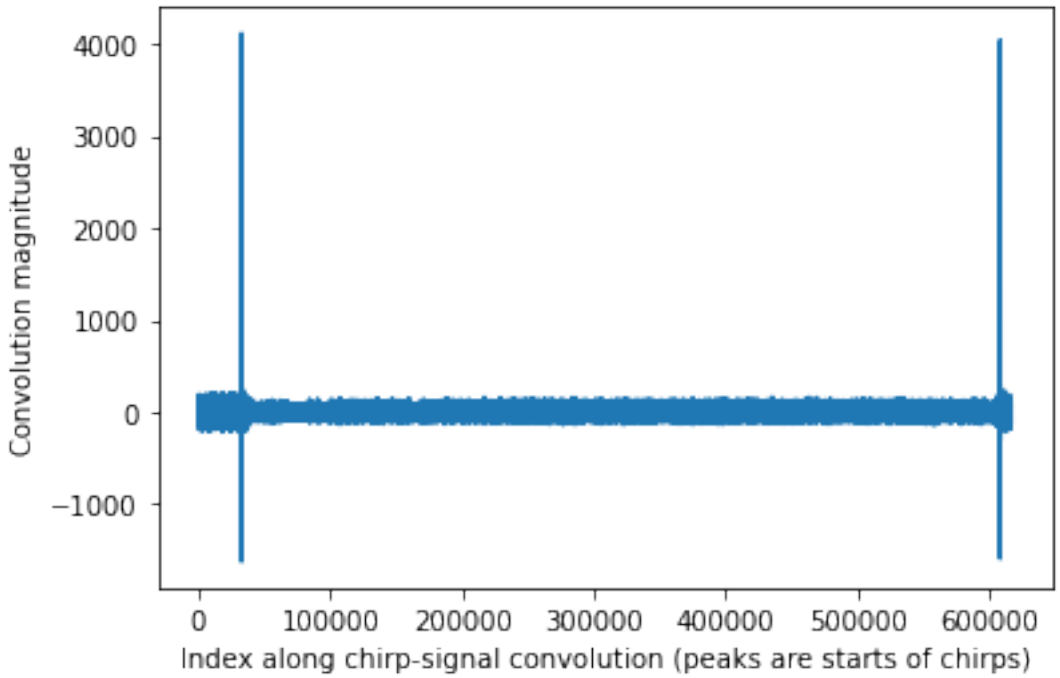
[2]:
```
import scipy.io as sc
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt
import time
import struct
```

[3]:
```
a = sc.loadmat('cauchy_schwarz_decoding.mat')
b = sc.loadmat('cauchy_schwarz_decoding_2.mat')
```

[4]:
```
chrp = np.squeeze(b['chrp'])
y2 = np.squeeze(b['y2'])
L_chrp = np.shape(chrp)[0]

# find the sequence of inner products between chirp and received signal via␣
 ↪convolution
# must time flip signal for convolution and time flip convolution back to␣
 ↪forward direction
conv = np.flip(signal.convolve(chrp,np.flip(y2),'valid'))
```

[5]:
```
# find peaks in conv to locate chirp start/stop tones and extract the desired␣
 ↪signal
plt.plot(conv)
plt.xlabel('Index along chirp-signal convolution (peaks are starts of chirps)')
plt.ylabel('Convolution magnitude')
plt.show()

# find indices of the 2 largest values in the convolution (see graph)
end_chrp,start_chrp = np.argpartition(conv,-2)[-2:]
```

```
# signal starts at the end of the first chirp and ends at the start of the␣
 ↪second chirp
y = y2[(start_chrp + L_chrp):end_chrp]

print('start of first chirp: ' + str(start_chrp)) # start of first chirp
print('start of second chirp: ' + str(end_chrp)) # start of second chirp
print('chirp length: ' + str(L_chrp)) # chirp length
print('signal length: ' + str(np.shape(y)[0])) # signal length (divisible by 40)
```




```
start of first chirp: 32768
start of second chirp: 608320
chirp length: 8192
signal length: 567360
```

[6]:
```
# from PROBLEM SET #1

c0 = a['c0']
c1 = a['c1']

# construct the matrix C which contains as columns the carriers c0 and c1
C = np.transpose(np.array([c0[0],c1[0]]))

# construct the matrix Y which contains as rows the received (noisy) carrier
# tones corresponding to each transmitted bit
# the width of Y should be equal to the length of one carrier tone
```

```
Y = y.reshape(-1,np.shape(C)[0])

# use matrix multiplication of C and Y to compute the sequence of inner products
# between each received (noisy) carrier tone and each known carrier tone (c0␣
 ↪and c1)
S = np.matmul(Y,C)

# use argmax to decode according to cauchy schwarz
# bits should have shape (N,) where N is the number of decoded bits
bits = np.argmax(np.abs(S), axis=1)

# conversion from binary to uint8
strResult = ''.join(str(n) for n in bits)
byteResult = list(int(strResult[i : i+8][::-1], 2) for i in range(0,␣
 ↪len(strResult), 8))
arrayResult = np.asarray([byteResult]).astype('uint8')

# writing decoded bits as a .jpg
f = open('decoded.jpg','wb')
f.write(arrayResult)
f.close()
```

[ ]: