ANDREW AND ERNA VITERBI
FACULTY OF ELECTRICAL ENGINEERING

הטכניון מכון טכנולוגי לישראל
TECHNION Israel Institute of Technology

COMMUNICATION LABORATORY
המעבדה לתקשורת

קודים לתיקון שגיאות

מעבדה 2,3 בהנדסת חשמל

הפקולטה להנדסת חשמל ע"ש ויטרבי

הטכניון - מכון טכנולוגי לישראל

נכתב על־ידי: אשד רם, אוגוסט 2019.
עדכון אחרון: אשד רם, 16 בספטמבר 2019

תוכן עניינים

רשימת איורים

הניסוי

ניסוי זה עוסק בבעיית התקשורת על־גבי ערוץ רועש, ובפתרון שלה בדמות תורת הקידוד (Coding Theory). מטרת הניסוי היא לחשוף את מבצעיו לאתגרים ההנדסיים שעומדים בבסיס בעיית התקשורת, תוך כדי הכרה של מושגים בסיסיים הדרושים כדי להבין את המרכיבים השונים בסכימת הקידוד. בניסוי מודגמות טכניקות קידוד שונות המסוגלות לתקן רעשים, ובכך לאפשר תקשורת אמינה בין שני משתמשי קצה. טכניקות אלו כוללות אלגוריתמי פיענוח אשר הם חלק בלתי נפרד מטכנולוגיות מידע קיימות (Cellular, WiFi, Bluetooth, Flash, Storage, ...).

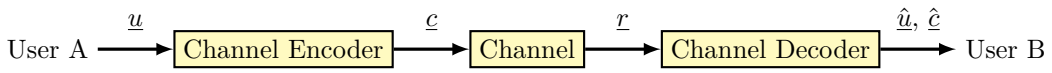
קודים לתיקון שגיאות - למה זה טוב?

כיום, כמעט כל מכשיר אלקטרוני מתקשר עם סביבתו: החל ממחשבים אשר מתקשרים עם מחשבים אחרים דרך האינטרנט באמצעות תקשורת אלחוטית (WiFi) או חוטית (Ethernet), דרך טלפונים סלולרים (BlueTooth, Cellular), לווינים בחלל (Deep Space Communication), ועד למכוניות אוטונומיות והאינטרנט של הדברים (IoT - Internet of Things). בכל אחת מהדוגמאות לעיל, מידע עובר בין משתמשים בתווך פיזיקלי מסוים - אוויר, כבלים וכדומה - אשר עלול לשבש את האות העובר דרכו; אנחנו קוראים לתווך הזה **ערוץ רועש**. כמובן שנרצה שהודעה שנשלחה תגיע ליעדה ללא שגיאות, וכאן נכנסים לתמונה קודים לתיקון שגיאות (ECC - Error Correcting Codes). למעשה, קודים לתיקון שגיאות הם ההבדל בין מערכת תקינה למערכת לא פונקציונלית, והם נמצאים בכל תקני התקשורת והתקני זיכרון.

נניח תקשורת בין שני משתמשים: משתמש א' ומשתמש ב'. תפקידה של סכימת הקידוד הינה לבצע פעולות על המידע שנשלח כדי להעביר אותו ללא שגיאות, בצורה יעילה וחסכונית ככל שניתן. לדוגמא, נניח שמשתמש א' מעוניין לשלוח למשתמש ב' אימוג'י כזה ☺, אשר מיוצג על־ידי חמישה ביטים 10001, ובזמן השליחה הערוץ הכניס שגיאה והחליף את הערך של הביט השלישי. משתמש ב' קיבל את הרצף 10101 אשר מייצג את האימוגי ☹! ברור שמצב כזה איננו יכול להתקיים, ותפקידה של סכימת הקידוד הוא להגן על המידע המועבר מפני רעשים כאלו.

איור 1 ממחיש את סכימת הקידוד שנתעסק בה: **מקודד ערוץ, ערוץ, ומפענח הערוץ**. מקודד הערוץ מקבל **מילת אינפורמציה** $\underline{u}$ (כמו הביטים שייצגו את האימוגי החייכן), מגן עליה בעזרת הוספת סימבולי יתירות (מידע נוסף) ובמוצא נותן כפלט **מילת קוד** $\underline{c}$. הערוץ עלול לשבש את מילת הקוד ומוציא את **האות המורעש** $\underline{r}$. המפענח קולט את $\underline{r}$ ומחשב את **מילת הקוד המשוערכת** $\hat{\underline{c}}$ ואת מילת האינפורמציה המתאימה $\hat{\underline{u}}$.

הרבה שאלות הנדסיות עולות מהתיאור הזה, למשל: 1) איך כדאי למקודד להוסיף את היתירות? 2) איך כדאי למפענח לשערך את מילת הקוד שנשלחה? 3) איך בחירות אלו תלויות בסוג הערוץ הרועש? התשובות לשאלות מהסוג הזה מורכבות וניידונות בקורסים השונים בפקולטה כמו **"מבוא לתורת הקידוד בתקשורת"** (046205). בניסוי זה ננסה להבין ולענות חלקית על שאלות אלה בצורה מעשית דרך דוגמאות.



איור 1: בעיית התקשורת: מקודד ערוץ, ערוץ, מפענח ערוץ.

קורסי קדם

הניסוי איננו דורש קורסי קדם מתחום התקשורת, והוא מהווה הצצה לעולם מעניין, מאתגר ומורכב. דרישות הקדם הן: מבוא להסתברות ח' (104034), ורקע בסיסי ב-MATLAB. לרשות מבצעי הניסוי נספח שמספק ידע בסיסי על־מנת להפיק את המירב מהניסוי.

מבנה הניסוי

הניסוי מורכב משני מפגשים של ארבע שעות כל אחד. בכל מפגש הסטודנטים יבצעו 4 תתי־ניסוים. כל תת־ניסוי עוסק בנושא מסוים, ובנוי משאלות מונחות הכוללות הרצת קטעי קוד MATLAB, יצירת גרפים, השלמת/כתיבת קטעי קוד MATLAB. בסוף כל תת־ניסוי ישנם שאלות לסיכום עליהם ניתן לענות בבית לאחר הניסוי במעבדה.

קבצים נלווים

יחד עם חוברת זו, ישנם קבצים נוספים לטובת משתתפי הניסוי:

1. נספח חומר רקע בקובץ CodingLabSupplementary.pdf
2. קובץ מכווץ שמכיל קבצי MATLAB בשם For Students.zip. בקובץ זה ישנם 3 תיקיות:

- Data - משתנים וקבצים שיוטענו במהלך הניסוי: BARBARA.tif, NY.gif, Stream.flac, Variables.mat.
- Functions - שמונה־עשר פונקציות MATLAB אשר נעשה בהם שימוש במהלך הניסוי: AWGN.m BECMLDecoder.m, bit2im.m, bit2str.m, BPSK.m, ConstChannel.m, Energy.m, G2H.m, gf2redref.m, im2bit.m, imCompress.m, imDecompress.m, NoisyNumWriter.m, NoisyTypeWriterChannel.m, PeelingDecoder.m, plotTanner.m, regLDPCpar.m, str2bit.m. אין לשנות את פונקציות אלו, זה עלול לשבש לכם את הניסוי!
- Scripts - לכל אחד מתשעת תתי־הניסוים יש קובץ שמכיל את הפקודות שאותם יש להריץ: Ex1.m, Ex2.m, Ex3.m, Ex4.m, Ex5.m, Ex6.m, Ex7a.m, Ex7b.m, Ex8.m. ניתן להשתמש בקבצים אלו במקום לבצע העתק־הדבק מהחוברת.

ציון

הציון בכל אחד ממפגשי הניסוי מורכב מ:

- דו"ח מכיו לכל מפגש שבו הסטודנטים יענו על שאלות הכנה - 10%
- בוחן מוכנות בכל מפגש שבו הסטודנטים ייבדקו על שאלות ההכנה - 20%
- עבודה במעבדה - 40%
- דו"ח מסכם לכל מפגש שבו הסטודנטים יענו על שאלות שעלו בזמן ביצוע הניסוי, ושאלות לסיכום שמופיעות בסוף כל תת־ניסוי. על הסטודנטים לצרף גם את קוד ה־MATLAB שהם כתבו - 30%

שאלות הכנה למפגש הראשון

לרשותכם נספח חומר רקע אשר מכיל את התשובות לרוב שאלות ההכנה.

שימו לב

קבצי הדו"ח צריכים להיות בפורמט pdf. שם קובץ הדו"ח צריך להיות בפורמט הבא: <Name1>&<Name2>–<Report Type>.pdf, כאשר <Name1>, <Name2> הם שמות משתתפי הניסוי (ללא סוגריים משולשים), ו־ <Report Type> הינו סוג הדו"ח, ויכול לקבל אחד מתוך ארבע אפשרויות: Part1Pre, Part1Post, Part2Pre, Part2Post. למשל, אם משתתפי הניסוי הם אבי ובתיה, אז **הדו"ח המכין** שלהם למפגש הראשון ייקרא Avi&Batia–Part1Pre.pdf.

---

1. חפשו באינטרנט הסבר על ספרת הביקורת במספרי הזהות הישראלים. איך מחשבים אותה? איך בודקים שמספר זהות הינו תקין? איזה סוג שגיאות היא מזהה?

   כתבו קוד MATLAB אשר מייצר ספרת ביקורת, וקד נוסף אשר בודק האם מספר זהות הינו תקין.

2. **קוד חזרות** - פרק 4 בנספח

   נתון קוד חזרות באורך n: המקודד מקבל בכניסה ביט יחיד ומשכפל אותו n פעמים. מהו קצב הקוד? נניח שליחה של ביט אינפורמציה בודד, ונניח שהערוץ עלול להפוך ביטים. כמה שגיאות (היפוכי ביטים) ניתן לתקן בעזרת קוד החזרות?

   ממשו ב־ MATLAB מקודד חזרות באורך $n = 3$: המקודד יקבל וקטור שורה של ביטים באורך כלשהו, ישכפל כל ביט n פעמים, ויפלוט וקטור שורה ארוך פי $n = 3$. קודדו את השמות הפרטיים שלכם בעזרת המקודד: ראשית המירו את רצף התווים לביטים (היעזרו בקוד ASCII ובפונקציה de2bi), ולאחר מכן שכפלו כל ביט 3 פעמים. הציגו את התוצאה.

3. **הערוץ הגאוסי**

   - נניח שידור קוד לינארי בקצב R על גבי הערוץ הגאוסי עם שונות σ^2, ועם מיפוי BPSK $'0' \to +A,\ '1' \to -A$. מהי האנרגיה לביט E_b? מהו יחס האות לרעש E_b/N_0? (פרקים 2 ו־4)
   - מהו פיענוח Hard Decision? מהו פיענוח Soft Decision? מי מביניהם בעל הסתברות שגיאה נמוכה יותר? (פרק 6)

4. קראו בדוקומנטציה של MATLAB על הפקונציות bsc, qfunc. הסבירו מהם ערכי הקבלה וההחזרה שלהן ומה הן עושות.

5. **פרמטרים של קוד** - פרקים 4־6 בנספח

   - מה זה ספר קוד? מהו קוד ליניארי? (פרק 4)
   - מהו מרחק Hamming? מהו משקל Hamming? (פרק 4)
   - הסבירו מה זה מרחק קוד, ואיך מרחק הקוד קשור ליכולת לתקן שגיאות. (פרק 6)
   - מה מרחק קוד החזרות? האם קוד החזרות הוא לינארי? אם כן, כתבו מהו מימד הקוד, וכתבו מטריצה יוצרת עבור קוד החזרות. (פרקים 4 ו־5)
   - נתון קוד בינארי באורך n המכיל את כל המילים בעלות משקל Hamming זוגי (קוד זה נקרא קוד זוגיות). מהו מרחק קוד זוגיות באורך n? האם הקוד הזה לינארי? אם כן, מצאו את מימד הקוד. (פרקים 4 ו־5)

מפגש ראשון

הכנת סביבת העבודה

פתחו את תוכנת MATLAB, וצרו לעצמכם תיקיה בה תממשו את הניסויים. הורידו מאתר labadmin את הקובץ ForStudents.zip וחלצו ממנו את התיקיות Data, Functions, Scripts (ראו הסבר על תיקיות אלו למעלה). על־מנת שתוכלו להריץ פונקציות ולקרוא למשתנים, עליכם להוסיף את תיקיות Data, Functions לסביבת העבודה: לחצו עם הלחצן הימני של העכבר על תיקיית Functions, ובתפריט שייפתח לחצו על Add to Path ⇒ Selected Folders and Subfolders. בצעו את התהליך הנ"ל גם על תיקיית Data.

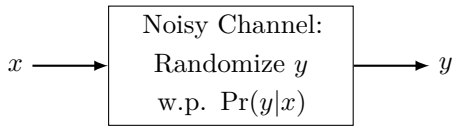
בתיקיית Scripts תמצאו קטעי קוד המתאימים לכל אחד מהניסויים. במקום לבצע העתק־הדבק מחוברת הניסוי, אתם יכולים לבצע העתק־הדבק מהקבצים של תיקיה זו (Acrobat Reader מוסיף תווים לא רצוים בזמן העתקה).

שימו לב

זכרו את פורמט שם קובץ הדו"ח <Name1>&<Name2>–<Report Type>.pdf.

ניסוי 1 - ערוצים רועשים

הדבר הראשון שיש להבין בעת תכנון מערכת לתיקון שגיאות הינו מודל הרעש/השגיאה שמולו אנו מתגוננים. המודל בו אנו נעסוק הינו מודל הסתברותי, כלומר נניח שהרעש במערכת התקשורת הינו אקראי ולא זדוני.



איור 2: ערוץ רועש

יש מגוון רחב של ערוצים, כל אחד עם התכונות והמאפיינים שלו. בחלק זה של הניסוי נעשה היכרות עם הערוץ הבינארי הסימטרי (BSC), והערוץ הגאוסי האדיטיבי (AWGN).

ערוץ BSC – Binary Symmetric Channel

1. הכניסו את שמותיכם הפרטים למחרוזת והשתמשו בפונקציה `str2bit` כדי להמירה לביטים. וודאו בעזרת הפונקציה ההופכית `bit2str` שאכן קיבלתם בחזרה את המחרוזת המקורית. הכניסו לדו"ח את המחרוזת ורצף הביטים. העבירו את מחרוזת הביטים בערוץ bsc(0.1) והציגו את התוצאה: הן את רצף הביטים המורעש והן את המחרוזת המתאימה לו. כמה ביטים התהפכו בערוץ?

2. פתחו את המשתנה `bitStrErr1` אשר נמצא בקובץ Variables.mat. משתנה זה הוא הרעשה של הודעת טקסט בערוץ bsc(0.1). המירו אותו למחרוזת תווים בעזרת `bit2str`. האם אתם יכולים לנחש מה היתה הודעת הטקסט? אם כן, קראו למדריך והסבירו לו איך גיליתם.

3. פתחו את המשתנה `bitStrErr2`. משתנה זה הוא הרעשה של הודעת טקסט (אחרת) בערוץ $\mathrm{bsc}(1)$. חזרו על הסעיף הקודם.

ערוץ AWGN – Aditive White Gaussian Noise

4. בחרו קטע שיר באנגלית של לפחות מאה תווים והעתיקו אותו לתוך משתנה (השתמשו ב־`strcat` אם השיר ארוך מאוד). המירו את המחרוזת לביטים: `bitStream = str2bit(song);`.
5. בצעו מיפוי $'0' \to +1,\ '1' \to -1$ לרצף הביטים שקיבלתם והעבירו את האות המתקבל בערוץ גאוסי עם שונות $\sigma = 1$. הציגו את האות המורעש בעזרת `plot`.

```
A=1; bitStrModulated = BPSK(bitStream,A);
sigma=1; bitStrErr = AWGN(bitStrModulated,sigma);
figure(); plot(bitStrErr);
```

6. בצעו hard decision לאות המורעש: `bitStrHard = double(bitStrErr<0);`. כמה שגיאות נפלו בתהליך? מהו המספר היחסי של שגיאות שנפלו?
7. חזרו על 2 הסיעיפים האחרונים עם מיפוים $'0' \to +A,\ '1' \to -A$, עבור ערכי A שנעים לוגריתמית, והציגו בגרף את המספר היחסי של שגיאות שנופלות עבור כל ערך של A:

```
A = logspace(-1,2,100); BitErrorRate = zeros(1,numel(A));
for aa = 1:numel(A)
  a = A(aa);
  ... BPSK + AWGN + Hard Decision + Count Errors
end
```

שימו לב

- כאשר אתם מתבקשים להשלים קטעי MATLAB, אז השלימו במקומות שמסומנים בשלוש נקודות $(\ldots)$.
- לאורך כל הניסוי, כאשר תתבקשו להציג גרף של הסתברות שגיאה, השתמשו בפקודה `semilogy` אשר יוצרת גרף עם סקאלה אנכית לוגריתמית. הסיבה לשימוש בסקלאה לוגריתמית נובעת מכך שהסתברות השגיאה בדרך כלל יורדת מהר (בצורה מעריכית).
- כאשר משתמשים בפקודה `semilogy`, ערכים של 0 נעלמים מהגרף.

---

8. חזרו על הניסוי בסעיף 7 100 פעמים, מצעו את התוצאות והציגו על גרף.

שאלות מחשבה לסיכום

1. מה הסיבה, לדעתכם, שניתן לשחזר שגיאות במחרוזת תווים בשפה האנגלית (כלומר מילים תקינות בשפה)? נסו לחשוב מה אתם (המוח שלכם) עושים כשאתם מתקנים שגיאות אלו.
2. הסבירו איך להפוך ערוץ BSC עם פרמטר p מסוים, לערוץ BSC עם פרמטר $1-p$.
3. האם הגרף שקיבלתם בסעיף 7 הינו מונוטוני יורד? הסבירו.
4. האם הגרף שקיבלתם בסעיף 8 הינו מונוטוני יורד? הסבירו.
5. מה הקשר בין ממוצע המספר היחסי של השגיאות שנופלות אחרי ביצוע hard decision בערוץ הגאוסי עם מיפוי $'0' \to +A,\ '1' \to -A$, לבין פרמטר הערוץ σ? בתשובתכם התייחסו לפונקציה `qfunc`.

ניסוי 2 - קוד חזרות

אחת הדרכים הכי פשוטות לקודד מידע הינה בעזרת קוד חזרות. בשיטה זו, כל ביט אינפורמציה משוכפל n פעמים ליצירת מילת קוד אשר נשלחת בערוץ. הפיענוח של קוד החזרות הינו פשוט מאוד. אם רוב הביטים המתקבלים הם '1' אז ננחש שנשלח ביט האינפורמציה שערכו '1', ולהפך. מפענח זה נקרא "מפענח רוב".

קוד החזרות אמנם פשוט מאוד, ויכול לתקן הרבה שגיאות, אבל הקצב שלו נמוך מאוד, מה שגורם לו להיות לא אטרקטיבי להרבה אפליקציות. עם זאת, ישנן אפליקציות אשר משתמשות בקוד החזרות, בעיקר בגלל הפיענוח הפשוט שלו. למשל, בחוות שרתי אכסון יש לעיתים שכפול מידע כדי להתמודד עם נפילות של שרתים.

איור 3: קוד חזרות באורך 3 עם מפענח רוב, מעל הערוץ הבינארי הסימטרי

BSC

1. הריצו את הקטע הבא:

```
emojis = '<:-) 8-D ;-O :-X'; msg = str2bit(emojis);
msgLength = length(msg);
```

msg הינה ההודעה (בביטים) שאנחנו רוצים להעביר.

2. העבירו את רצף הביטים בערוץ $\mathrm{BSC}(p)$ עם הסתברות היפוך $p = 0.1$. כמה שגיאות בינאריות נפלו? הציגו את המחרוזת המתקבלת בעזרת bit2str.

3. קודדו את רצף הביטים שמייצג את המשתנה emojis בעזרת קוד חזרות באורך $n = 9$, והעבירו את המידע המקודד דרך ערוץ $\mathrm{BSC}(p)$ עם הסתברות היפוך $p = 0.1$. פענחו את המידע לפי מפענח רוב:

$$\mathrm{Decode}(b_1, b_2, \ldots, b_n) = \begin{cases} 1 & \sum_{i=1}^{n} b_i > \frac{n}{2} \\ 0 & \text{else} \end{cases} ,$$

כאשר $b_1, b_2, \ldots, b_n$ הם רצף הביטים המורעשים בכניסה למפענח. האם הצלחתם לקבל 0 שגיאות? הציגו את המחרוזת שמתקבלת.

4. חזרו על הסעיף הקודם עם הסתברות היפוך בערוץ $p = 0.25$. האם הצלחתם לקבל 0 שגיאות? הציגו את המחרוזת שמתקבלת והסבירו.

5. הביטו בקטע הבא:

```
nVec=9:4:33; numErr = zeros(1,length(nVec));
for jj =1:length(nVec)
  n=nVec(jj);
  ... encode msg + simulate BSC(0.25) + Majority Decoding
end
```

לכל ערך של n קודדו את רצף הביטים שמייצג את המשתנה `emojis` בעזרת קוד חזרות בארוך n, העבירו אותו בערוץ $\mathrm{BSC}(p)$ עם הסתברות היפוך $p = 0.25$, פענחו את האות המורעש בעזרת מפענח רוב, והציגו כמה שגיאות נפלו בפיענוח לכל ערך של n.

6. הריצו את הניסוי מהסעיף הקודם 100 פעמים ומצעו את התוצאות. בצעו את הפעולה הזו כאשר פרמטר הערוץ נע על ערכים: `p = 0:0.05:0.5`. הציגו בגרף אחד את **מיצוע** המספר **היחסי** של שגיאות (כלומר מספר השגיאות לחלק למספר ביטי האינפורמציה) כפונקציה של `p` לכל ערך של n (עליכם לקבל 7 עקומות).

שאלות מחשבה לסיכום

1. סעיף 5: האם קיבלתם מונוטוניות ב־n של מספר השגיאות (n יותר גדול מוביל תמיד למספר שגיאות יותר קטן)? הסבירו.
2. סעיף 6: האם לכל p קיבלתם מונוטוניות ב־n של ממוצע מספר השגיאות? מה המחיר של הקטנת הסתברות השגיאה? הסבירו.
3. סעיף 6: האם לכל n קיבלתם מונוטוניות ב־p של ממוצע מספר השגיאות? הסבירו מה קורה כאשר $p = 0.5$.

ניסוי 3 - קודים

כמסקנה ישירה מהניסוים הקודמים שערכתם, יש צורך בקודים לתיקון שגיאות. השאלה היא איך כדאי לתכנן אותם? מה הופך קוד לטוב? האתגר הגדול של תחום הקודים לתיקון שגיאות הוא למצוא קודים עם מרחק גדול וקצב גבוה, אשר ניתנים לקידוד ופיענוח בסיבוכיות יעילה!

שימו לב

לאורך כל חוברת הניסוי שימוש במילה "קוד" מתייחס לקוד תיקון שגיאות, ולא לקטע קוד בשפת תכנות.

נתון קוד בינארי באורך $n = 7$ עם $M = 4$ מילות קוד:

```
n = 7; Codebook = {
[1,0,1,1,0,0,1],...% 00
[1,1,0,0,1,1,0],...% 01
[1,1,1,0,1,1,0],...% 10
[0,0,1,1,1,0,1]};  % 11
M=numel(Codebook);
```

המקודד שנבנה מבצע את המיפוי הבא:

$$00 \to 1011001, \quad 01 \to 1100110, \quad 10 \to 1110110, \quad 11 \to 0011101$$

נקודד בעזרת קוד זה שני ביטי אינפורמציה:

```
msg = [0 1]; msgEnc = Codebook{bi2de(msg,'left-msb')+1};
```

1. מילת הקוד לעיל עברה בערוץ שהפך את הביט השלישי בה:

```
msgNoisy = msgEnc; msgNoisy(3) = 1-msgNoisy(3);
```

המפענח מבצע פיענוח כזה:

```
dBest = n;
for mm = 1:M
  if(n*pdist2(Codebook{mm},msgNoisy,'hamming')<dBest)
    mBest=mm; dBest = n*pdist2(Codebook{mm},msgNoisy,'hamming');
  end
end
msgDec = de2bi(mBest-1,2,'left-msb')
```

האם הפיענוח הצליח? אם לא, מה הסיבה לכך?

2. הריצו את הקטע הבא:

```
dmin=n;
for i=1:(M-1)
  for j=(i+1):M
    if(n*pdist2(Codebook{i},C{j},'hamming')<dmin)
      dmin=n*pdist2(Codebook{i},Codebook{j},'hamming');
    end
  end
end
```

מהו המרחק המינימלי של הקוד? מי הן מילות הקוד הכי קרובות אחת לשניה? קשרו את הממצא הזה לפיענוח מסעיף קודם.

3. שנו את הקוד כך שניתן יהיה לפענח את ההיפוך בביט השלישי, והראו את הצלחת הפיענוח.

נשאלת השאלה, האם הקוד שיצרתם בסעיף הקוד עמיד בפני היפוך ביט בודד, ללא תלות במיקום ההיפוך או בזהות מילת הקוד המשודרת.

4. הריצו ניסוי שבו כל מילת קוד מורעשת בהיפוך ביט בודד, עם כל האפשרויות.(4 מילות קוד ולכל אחת 7 היפוכים אפשריים - 28 נסיונות). האם הפיענוח מצליח תמיד?

5. נגדיל את אורך הקוד להיות באורך $n = 15$. מצאו קוד עם $M = 4$ מילים שיש לו מרחק מינימלי גדול (לפחות 7). כתבו את מילות ומרחק הקוד, והסבירו איך בניתם את הקוד.

6. כעת נשתמש בקוד מפורסם: קוד BCH. הריצו את הקטע הבא:

```
msg = str2bit('EE:)').'; msgSize = length(msg);
n = 15; k = 5; bchEncoder = comm.BCHEncoder(n,k);
bchDecoder = comm.BCHDecoder(n,k);
msgPadSize = k*ceil(msgSize/k) - msgSize; %padding with zeros
%reshape for encoding by columns
msgPadReshape = reshape([msg zeros(1,msgPadSize)],k,[]);
msgEnc = zeros(n,size(msgPadReshape,2));
for j = 1:size(msgPadReshape,2)
  msgEnc(:,j) = bchEncoder(msgPadReshape(:,j));
end
```

מה האורך הכולל של המידע המקודד?

7. העבירו את msgEnc בערוץ $\mathrm{BSC}(p)$ עם הסתברות היפוך $p = 0.05$, פענחו את האות הנקלט בעזרת האובייקט bchDecoder והציגו את המחרוזת המתקבלת (שימו לב המפענח מצפה לקבל ווקטור עמודה). האם הפיענוח הצליח?

שאלות מחשבה לסיכום

1. הסבירו במילים מה המפענח בסעיף 1 עושה. למה לדעתכם זו צורת הפיענוח?

2. סעיף 6: כמה מילות קוד יש בספר הקוד שמוגדר על ידי המקודד bchEncoder? מה מרחק הקוד? השוו עם סעיף 5.

3. למה יש צורך בריפוד אפסים בסעיף 6?

ניסוי 4 - קידוד בערוץ הגאוסי

בניסוי הקודם ראינו את החשיבות של בניית קוד טוב. כעת נבחן את הסכימות השונות כאשר התקשורת מבוצעת בערוץ הגאוסי עם מודולוציית BPSK שבו הביטים בכניסה לערוץ ממופים לפי $'0' \to +A, '1' \to -A$, והערוץ מוסיף משתנה רעש גאוסי עם תוחלת אפס ושונות σ^2.

מכיוון שהגדלת עוצמת האות המשודדר (A) מורידה את הסתברות השגיאה אך מגדילה את האנרגיה שצריך להשקיע בשידור, ישנו trade-off בין השניים. על־מנת לבצע השוואה הוגנת בין סכימות שונות, נהוג להציג את הסתברות השגיאה כפונקציה של יחס האות לרעש E_b/N_0.

מבחינת פיענוח, יש שתי גישות עיקריות: פיענוח קשיח (Hard Decoding) שבו מוצא הערוץ קודם עובר דה־מודולציה , מקבל ערכים בדידים, ואז מפוענח לפי אלגוריתם פיענוח כלשהו; ופיענוח רך (Soft Decision) שבו הפיענוח מתבצע ישירות על האות הרציף ממוצא הערוץ. נשאלת השאלה, מה יותר עדיף מבחינת מזעור הסתברות השגיאה? בניסוי ננסה לענות על שאלה זו.

ההודעה אותה נעביר (`msg`) הינה באורך 5000 ביטים, אמפליטודות מיפוי ה־BPSK מסומנות ב־`A`, פרמטר הערוץ הגאוסי הינו `sigma`, ונמצע את הניסוי על פני `NumOfTrials` נסיונות.

```
msgLength = 5000; msg = randi([0 1],1,msgLength); A = logspace(-1,2,100);
AA = numel(A); NumOfTrials = 1e2; sigma = sqrt(0.5);
```

בשלב הראשון, ננסה לשדר בערוץ ללא קוד כלל:

איור 4: הערוץ הגאוסי עם מיפוי BPSK ללא קידוד

1. השלימו את הקטע הבא:

```
BER_Uncoded = zeros(NumOfTrials,AA);
for aa=1:AA
  ... perform BPSK
  for tt=1:NumOfTrials
    ... simulate AWGN + perform Hard Decision
    BER_Uncoded(tt,aa) = ... calculate Bit Error Rate
  end
end
... semilogy Average BER vs. Eb[dB]
```

שימו לב

- קטע זה זהה לנעשה בניסוי הראשון של המפגש. רצוי להסתכל בקטע הקוד שם.
- נהוג להציג תוצאות בערוץ הגאוסי כאשר הציר האופקי הינו יחס אותו לרעש $E_b/N_0[\text{dB}]$, כאשר N_0 הינה צפיפות ההספק הספקטרלית של הרעש הלבן. מכיוון שאנחנו מחזיקים את הרעש קבוע, אז נציג את התוצאות כפונקציה של $E_b[\text{dB}]$ בלבד.
- כאשר מציגים משתנה מסוים `var` בדציבל ([dB]), אז יש להוציא לוגריתם על בסיס 10 ולהכפיל ב־10 - `10*log10(var)`.

כעת נוסיף את קוד החזרות למערכת התקשורת:

איור 5: הערוץ הגאוסי עם מיפוי BPSK עם קוד חזרות ופיענוח קשיח

2. נתחיל עם פיענוח קשיח: המרת האות הנקלט לביטים ואז ביצוע פיענוח רוב.

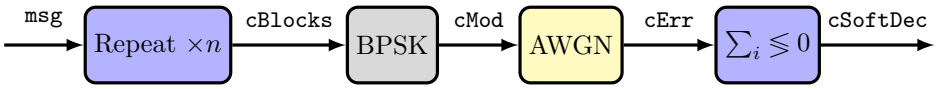
```
for n = [3 15]
  BER_Rep = zeros(NumOfTrials,AA); c = ...  encode msg
  for aa = 1:AA
    ... perform BPSK
    for tt=1:NumOfTrials
      ...  AWGN + Hard Decision + Majority Decoding
      BER_Rep(tt,aa) = ...  calculate Bit Error Rate
    end
  end
  ... plot Average BER vs. Eb[dB]
end
```

הוסיפו את ה־BER של הסכימה הזו לגרף הקודם. האם הביצועים השתפרו?

שימו לב

זכרו לחשב את E_b מחדש עבור כל קוד, כפי שמוסבר בנספח (פרק 5).

נשנה את שיטת הפיענוח של קוד החזרות לפיענוח רך: פיענוח האות הנקלט ישירות.



איור 6: הערוץ הגאוסי עם מיפוי BPSK עם קוד חזרות ופיענוח רך

3. הריצו את הקטע הבא:

```
for n = [3 15]
  BER_Rep = zeros(NumOfTrials,AA); c = ... encode msg
  for aa = 1:AA
    ... perform BPSK
    for tt=1:NumOfTrials
      ...  AWGN + soft Decision
      BER_Rep(tt,aa) = ...  calculate Bit Error Rate
    end
  end
  ... semilogy Average BER vs. Eb[dB]
end
```

הוסיפו את ה־BER לגרף. האם הביצועים השתפרו?

כעת נשתמש בקוד תיקון שגיאות אחר - קוד BCH בינארי בקצב $R = \frac{99}{127}$.

איור 7: הערוץ הגאוסי עם מיפוי BPSK עם קוד BCH ופיענוח קשיח

4. השלימו את הקטע הבא:

```
NumOfTrials= 1e1; n = 127; k = 99; bchEnc = comm.BCHEncoder(n,k);
bchDec = comm.BCHDecoder(n,k); msgPadSize = k*ceil(msgLength/k)-msgLength;
uBlocks = reshape([msg zeros(1,msgPadSize)],k,[]);
codewordsSent = size(uBlocks,2); cBlocks = zeros(n,codewordsSent);
for j = 1:codewordsSent %encode by columns
  cBlocks(:,j) = bchEnc(uBlocks(:,j));
end
BER_BCH = zeros(NumOfTrials,AA);
for aa = 1:AA
  ... perform BPSK
  for tt=1:NumOfTrials
    ...  simulate AWGN + Hard decision
    cHardBCHDec = zeros(k,codewordsSent);
    for j = 1:codewordsSent
      cHardBCHDec(:,j) = bchDec(... hard decidion column vector);
    end
    cHardBCHDec = cHardBCHDec(:).'; %rearange to a bit stream
    msgHardbchDec = cHardBCHDec(1:msgLength);%remove padding
    BER_BCH(tt,aa) = ...  calculate Bit Error Rate
  end
end
...  semilogy Average BER vs. Eb[dB]
```

הוסיפו את ה־BER לגרף וצרפו את הגרפים לדו"ח המסכם.

שאלות מחשבה לסיכום

1. סעיף 1: מה משמעו המשתנה `NumOfTrials`, ולמה מתבצע מיצוע? איך ייתכן שהסתברות השגיאה הולכת וקטנה עם הגדלת אמפליטודת מיפוי ה־BPSK למרות שלא משתמשים בקוד לתיקון שגיאות?

2. סעיפים 1-3: מדוע קוד חזרות עם פיענוח קשיח נותן ביצועים גרועים מהסכימה ללא קידוד? מדוע קוד חזרות עם פיענוח רך נותן ביצועים זהים לסכימה ללא קידוד בכלל?

01000011 01101111 01101101 01110000 01110101 01110100 01100101 01110010 01110011 00100000
01100100 01101111 01101110 01110100 00100000 01101101 01100001 01101011 01100101 00100000
01101101 01101001 01110011 01110100 01100001 01101011 01100101 01110011 00101100 00100000
01110000 01100101 01101111 01110000 01101100 01100101 00100000 01100100 01101111 00100001

בהצלחה!

שאלות הכנה למפגש השני

1. קראו בודוקומנטציה של MATLAB על:

- `containers.Map` - הסבירו מה זה אובייקט מסוג מפה, איך מאתחלים אותו ואיך ניגשים לאברים שבו.
- `bchnumerr` - הסבירו מה ערכי הקבלה וההחזרה שלה. בחרו ערך $N<100$ כלשהו והריצו `bchnumerr(N)`. הסבירו מה רואים.
- `comm.RSEncoder,comm.RSDecoder,comm.BCHEncoder,comm.BCHDecoder` -
הסבירו מה הפקודות הנ"ל עושות, ומה הפרמטרים שהם מקבלות. בפרט, הסבירו על הדגלים `'BitInput'`, `'ErasuresInputPort'`.
- `hist` - הסבירו מה הפונקציה עושה, ומה ערכי הקבלה שלה.

2. **Data Interleaving** - פרק 8 בנספח

- מה זה פרץ שגיאות (error burst)?
- מה עושה פעולת השזירה (interleaving)? מה מטרתה?
- כתבו קוד MATLAB שעושה שזירת בלוקים לתווים בגודל 3×5 ובצעו שזירה למחרוזת תווים כלשהי באורך 15. כתבו קוד MATLAB שמבצע את הפעולה ההופכית לשזירה ומחזיר את המחרוזת להיות בסדר נכון. וודאו כי 2 הפעולות עובדות והציגו את המחרוזת לאורך כל התהליך (לפני שזירה, אחרי שזירה, ואחרי דה־שזירה).

3. מה יכולות תיקון **מחיקות** של קוד Reed-Solomon באורך n ומימד k?

4. **קודי גרף ופעינוח איטרטיבי** - פרק 7 בנספח

- קראו בדוקומנטציה של MATLAB על משתנים מסוג `sparse`. מה המוטבציה לשימוש בהם?
- הגרילו ב־ MATLAB מטריצת בדיקת זוגיות H לקוד בינארי שהינה בגודל 5×10, וציירו את גרף הטאנר המתאים לה. האם הווקטור $(0,0,0,0,0,0,0,0,0,0)$ הינו מילת קוד חוקית בקוד אשר H הינה מטריצה בודקת עבורו?
- פענחו בעזרת גרף הטאנר שהגרלתם את הווקטור הנקלט הבא: $(0,?,0,0,?,0,0,0,0,0)$. הסבירו את מהלך האלגוריתם. האם הפעינוח הצליח?
- פענחו בעזרת גרף הטאנר שהגרלתם את הווקטור הנקלט הבא: $(0,?,?,?,?,0,0,0,0,0)$. הסבירו את מהלך האלגוריתם. האם הפעינוח הצליח?

מפגש שני

שימו לב

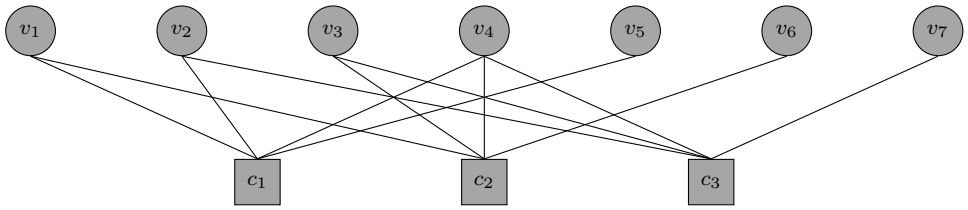
במפגש זה עליכם לבצע **4** ניסוים: ניסוי 5, ניסוי 6, ניסוי 8, **ואחד** מהניסוים 7.א/7.ב. על מדריך הניסוי לקבוע לכם איזה מהניסוים 7.א ו־7.ב לבצע. כמובן, מי שמעוניין יכול לבצע את כל הניסוים!

ניסוי 5 - קודי גרף

עד כה ראינו קודים מצוינים לתיקון שגיאות כמו קודי Reed-Solomon וקודי BCH. האם זה הכל? האם אפשר להשתמש בקודים האלו לכל מטרה? התשובה היא לא, בין השאר מכיוון שקשה להגדיל את אורך הקוד איתם (סיבוכיות פיענוח גדלה), והגדלת אורך הבלוק משפרת את הביצועים - מקטינה את הסתברות שגיאת הפיענוח! המסקנה היא שיש צורך במשפחות קודים אחרות, אשר סיבוכיות הפיענוח שלהם נמוכה יותר.

דוגמא נפוצה למשפחת קודים כזו, היא קודי LDPC – Low-Density Parity-Check. אשר נבחרו למספר רב של תקני תקשורת (WiFi, 5G, Deep-Space Communications) כמו גם התקני זכרון (SSD, Flash).

קודים לינאריים אלו מוגדרים בעזרת גרף דו־צדדי (גרף Tanner), מאופיינים במטריצה בודקת שהינה דלילה (sparse) , והפיענוח שלהם מתבצע על־גבי הגרף. בניסוי זה נתמקד



איור 8: דוגמא לגרף Tanner

בקודי LDPC בינארים המשודרים על גבי ערוץ המחיקה הבינארי (BEC). אלגוריתם הפיענוח בו נשתמש נקרא Peeling Decoder. שמו ניתן לו מכיוון שהוא מקלף את המחיקות אחת אחת (כמו בצל) - בכל פעם הוא חושף מחיקה אחת ומסיר אותה מהגרף, עד אשר לא נשארים יותר צמתים בגרף.

1. הגדירו את המטריצה הבודקת הבאה, והציגו אותה כגרף Tanner:

```
H = [
  1 1 1 1 0 1 1 0 0 0 ; 0 0 1 1 1 1 1 1 0 0 ;
  0 1 0 1 0 1 0 1 1 1 ; 1 0 1 0 1 0 0 1 1 1 ;
  1 1 0 0 1 0 1 0 1 1 ];
plotTanner(H);
```

צרפו לדו"ח את הגרף, והסבירו מה רואים. בתשובתכם, התייחסו למספר הצמתים, סוגי הצמתים, מספר הקשתות, וחיבורי הקשתות.

2. הריצו את הקטע הבא:

```
G=G2H(H); [k,n] = size(G); u = randi([0 1],1 ,k); rng(2,'twister');
t=2; erasureLocations = [ones(1,t) zeros(1,n-t)];
erasureLocations = erasureLocations(randperm(n));
y = mod(u*G,2); y(erasureLocations==1) = 2; %'2' is an erasure
```

G הינה מטריצה יוצרת המתאימה למטריצה הבודקת H. הציגו את האות המרועש y. האם אתם יכולים לנחש מה ערכי האינדקסים המחוקים (מבלי להסתכל במילת הקוד הלא מורעשת)?

3. הריצו את הפקודה:

```
[cDecPeeling,successPeeling] = PeelingDecoder(H,y,'PlotTanner',true);
```

הגרפים המופיעים מתארים את התקדמות אלגוריתם הפיענוח. שמרו את הגרפים בעזרת:

```
FigList = flip(findobj(allchild(0), 'flat', 'Type', 'figure'));
for Fig = 1:numel(FigList)
  saveas(FigList(Fig), sprintf('Peeling_Example_%d.jpg',Fig));
end
```

צרפו לדו"ח את הגרפים לפי סדר הופעתם והסבירו מה קורה בכל שלב: כמה צמתי משתנה יש? מה הערכים בצמתי הבדיקה? אילו צמתי משתנה אפשר להסיר?

4. הריצו את הפקודה: `[cDecML,successML] = BECMLDecoder(G,y);`. האם הפיענוח הצליח? האם cDecML זהה ל־ cDecPeeling. הסבירו.

5. כעת נגדיל את מספר המחיקות ל־ t=3. חזרו על סעיפים 2־4,

נרצה להגדיל את אורך הבלוק (ולהישאר באותו קצב).

6. הריצו את הקטע הבא, וענו על השאלות אחריו:

```
CNDegree = 6; VNDegree = 3; blockLength = 6*1e4;
H = regLDPCpar(CNDegree,VNDegree,blockLength);
```

- כמה ערכים שאינם אפס יש במטריצה H? מה מספר האלמנטים הכללי שיש בה?
- מה הטיפוס של המשתנה H? הריצו `whos H`; כמה זכרון תופס המשתנה H?
- הריצו `HFull = full(H)`. כמה זכרון תופס המשתנה HFull? מה המסקנה?

7. הריצו את הקטע הבא:

```
[m,n] = size(H); p = 0.4;
erasureLocations = rand(1,n)<p; y = 2*erasureLocations;
[cDecPeeling,successPeeling,fracErasedBits] = ...
PeelingDecoder(H,y,'PlotTanner',false);
```

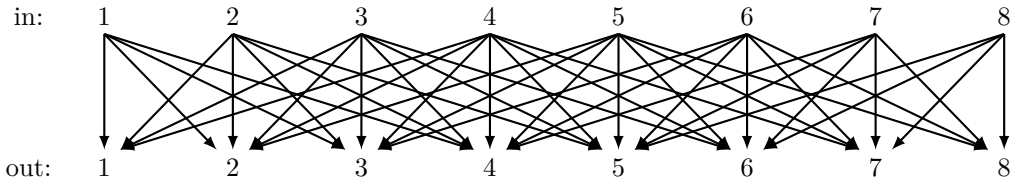
כמה מחיקות נפלו בערוץ? מה המספר היחסי של מחיקות? האם הפיענוח הצליח?

ניסוי 6 - תיקון טעויות הקלדה

טעויות הקלדה הן תופעה נפוצה שמטרידה הרבה משתמשים באינספור אפליקציות. בחלק זה, נראה כיצד אפשר להיעזר בקידוד על־מנת להתגבר על הבעיה.

בסכימות הקידוד שבנינו במפגש הראשון, הסתברות השגיאה הייתה קיימת תמיד; לקוד תיקון שגיאות הייתה יכולת תיקון כלשהי, ובהסתברות מסוימת (בתקווה קטנה מאוד!) השגיאה שמכניס הערוץ לא מאפשרת תיקון (Uncorrectable Error). בניסוי זה, נבחן מערכת שבה ניתן לקיים תקשורת ללא שגיאות כלל (אפילו לא בהסתברות קטנה!).

נניח, בשלב ראשון, שהמקלדת שלנו היא התווים '1' עד '8', ושטעות ההקלדה מתרחשת **בהסתברות קטנה מ־**$\frac{1}{2}$, ויכולה להוביל לתו שכן שרחוק עד כדי 3 מקומות (אם קיים שכן כזה). למשל, אם ניסיתם להקליד את התו '5', יכול להיות שהקלדתם בטעות את אחד התווים מ־'2' עד־'8', ואם ניסיתם להקליד '2', אז אפשר לטעות ולהקליד תו מ־'1' עד '5'. הערוץ הרועש נראה כך:



איור 9: ערוץ טעויות הקלדה למקלדת '1' - '8'

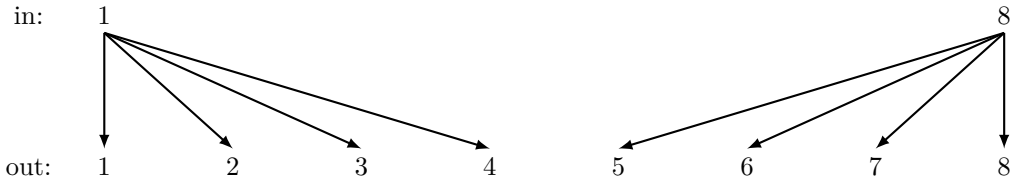
1. הריצו את הקטע הבא:

```
strIn = '46733'; strOut = NoisyNumWriter(strIn);
```

הציגו את `strOut` והסבירו.

2. נסו להשתמש בקוד החזרות באורך 7 ובמפענח רוב על־מנת להגן על `strIn` מפני השגיאות בערוץ (היעזרו בפונקציה `mode`). האם הצלחתם לתקן את כל השגיאות? חזרו על ההרעשה והפיענוח מספר פעמים ובדקו האם אתם **תמיד** מצליחים לפענח את את המידע ששודר.

נשים לב שאם נגביל את השימוש לתווים '1' ו־'8', אז ניתן יהיה לדעת **בוודאות** מה שודר:



איור 10: '1' ו־'8' הם תווים בני הבחנה בערוץ

כעת אם ביציאה מהערוץ יש '1' '2' '3' או '4', אז בוודאות בכניסה היה '1', ואם יש '5' '6' '7' או '8', אז בכניסה היה '8'.

3. נגדיר את המקודד הבא:

```
encoderIn =  {'1',  '2',  '3',  '4',  '5',  '6',  '7',  '8'  };
encoderOut = {'111','118','181','188','811','818','881','888'};
encoder = containers.Map(encoderIn,encoderOut);
```

מה אורך מילות הקוד? מה קצב המקודד (כמה המקודד "מנפח" את האינפורמציה)? קודדו את strIn כך

```
strEnc = []; i=1;
while i<= numel(strIn)
  strEnc = [strEnc, encoder(strIn(i))]; i = i+1;
end
```

והציגו את האות המקודד strEnc.

4. העבירו את האות שקיבלתם בסעיף קודם בערוץ והציגו את מוצא הערוץ.

5. השתמשו באובייקט מסוג containers.Map, הגדירו מיפוי שהופך את פעולת הערוץ (בהינתן שמשתשמים רק ב־'1' או '8' בכניסה, ראו איור 10) ותקנו את רעש הערוץ (denoising). הציגו את המחרוזת המתקבלת.

6. השתמשו באובייקט מסוג containers.Map כדי להגדיר מיפוי הפוך לפעולת המקודד encoder, וחלצו את מילת האינפורמציה ששודרה strIn.

כעת נעבור למודל יותר מציאותי. נניח שימוש במקלדת qwerty, ונניח שטעויות הקלדה מתרחשות בהסתברות קטנה מחצי, וגורמות להחלפת תו מקלדת מסוים בתו שכן לו (מימין או משמאל). למשל, נסיון להקליד 't' עלול להוביל לשגיאת הקלדה של 'r' או 'y'. על־מנת לפשט את הניסוי, נניח שימוש ב־41 תוים בלבד:

```
qwerty = {
  '1','2','3','4','5','6','7','8','9','0',... %10 elements
  'q','w','e','r','t','y','u','i','o','p',...%10 elements
  'a','s','d','f','g','h','j','k','l',';'...  %10 elements
  'z','x','c','v','b','n','m',',','.','/'...%10 elements
  ' '}; totalNumOfChars = numel(qwerty);
```

כאשר מובטח שהתו רווח ' ' לא גורם לשגיאה. בנוסף, תווים בקצה - שאין להם שכן מצד מסוים - עלולים להשתנות לתו השכן היחיד שלהם (ויכולים גם לא להשתנות).

7. הריצו את הקטע הבא:

```
strIn ='hello, my cellphone number is 555046205, thanks.';
strOut = NoisyTypeWriterChannel(strIn);
```

והציגו את המחרוזת ביציאה מהערוץ.

8. מצאו 16 תווים מתוך הרשימה qwerty ששידורם מבטיח שחזור מושלם (כמו שמצאנו בסעיפים למעלה עם '1' ו־'8'). ענו על השאלות הבאות:

- כמה מילות קוד שונות ניתן ליצור בעזרת תוו בר־הבחנה בודד (יכול לקבל אחד מתוך 16 אפשרויות)? למה אי־אפשר בעזרת תוו בר הבחנה בודד לקודד כל תו מקלדת?

- כמה מילות קוד שונות ניתן ליצור בעזרת שני תווים בני־הבחנה (כל אחד מהם יכול לקבל אחד מתוך 16 אפשרויות)? מה יהי אורך מילות הקוד (בתווים) במקרה זה? מה קצב המקודד?

כדי לשפר את קצב הקוד ולחסוך בתווים, נשתמש בקוד עם אורך משתנה (variable-length coding). טענו את המשתנה `encoderOut` אשר נמצא בקובץ Variables.mat. משתנה זה מחזיק את מילות הקוד המתאימות לכל אחד מהתווים ב־`qwerty`.

9. הציגו את `encoderOut`, וענו על השאלות הבאות:
   - מה האורך הממוצע של מילת קוד?
   - איך המקלט יידע אם מילת הקוד הינה באורך תו בודד או שני תווים?
10. הגדירו מקודד הממיר בין `qwerty` ל־`encoderOut` בעזרת אובייקט `containers.Map`, וקודדו בעזרת המקודד את המחרוזת `strIn`. כמה תווים יש במחרוזת המקודדת? הציגו את המחרוזת המקודדת.
11. העבירו את האות המקודד בערוץ והציגו את התוצאה.
12. חשבו על אלגוריתם פיענוח לקוד ולערוץ הנ"ל. לפני המימוש, הסבירו את הרעיון שלכם למדריך. **רמז -** חלקו את הפיענוח לשני שלבים: הסרת הרעש (denoising), והיפוך פעולת הקידוד.
13. פענחו את מוצא הערוץ וודאו שקיבלתם את מחרוזת האינפורמציה `strIn` ללא שגיאות.

שאלות מחשבה לסיכום

1. למה לדעתכם קוד החזרות נכשל בסעיף 2? האם יש אורך חזרות מסוים שבו **מובטח** שלא ייפלו שגיאות פיענוח בכלל?
2. סעיף 8: מה מייחד קבוצת תווים בני־הבחנה? למה רצוי שתיהיה לנו קבוצת תווים בני־הבחנה כמה שיותר גדולה?
3. סעיף 8: מה ממוצע אורך מילות הקוד ב־ `encoderOut`? השוו לממוצע אורך המילים אם היינו משתמשים בקוד בלוק (כלומר בקוד שבו כל המילים בעלי אורך זהה).
4. סעיף 8: נניח שלא הייתם נוקטים בגישה של תווים בני־הבחנה, אלא הייתם מקודדים תווים בעזרת קוד תיקון שגיאות (כמו למשל קוד BCH). האם היה מובטח פיענוח מוצלח?

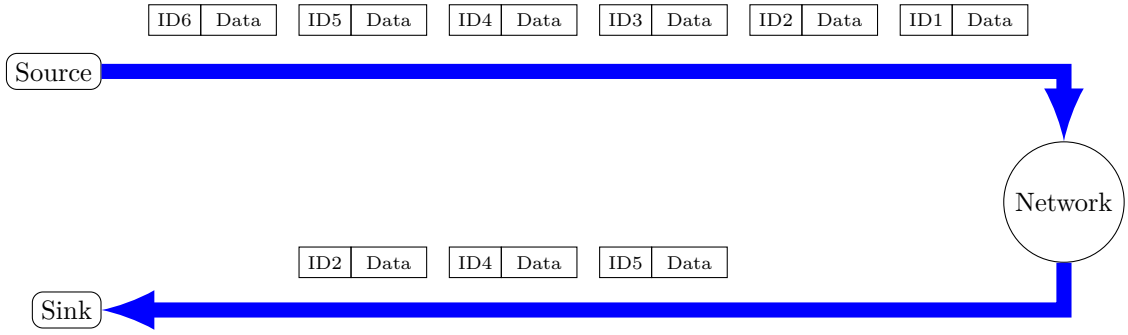
ניסוי 7.א - קידוד עבור מחיקות ושזירה (interleaving)

שימו לב

זכרו לשאול את מדריך הניסוי האם עליכם לבצע את ניסוי 7.א או 7.ב.

תיקון מחיקות שונה בעיקרו מתיקון שגיאות (אם כי לרוב משתמשים באותם הקודים), מכיוון שעל המפענח לדעת איפה בדיוק נפלו מחיקות! באפליקציות מסוימות, מידע זה זמין למפענח ולכן אפשר לתכנן את המערכת בהתאם. למשל, חשבו על חוות שרתים ענקית שמאוכסן בה מידע של חברת טכנולוגיה גדולה. לעיתים קרובות שרת כזה נופל והמידע בו אובד. מכיוון שידוע איזה שרת נפל, אז ניתן להתיחס עליו כאל מחיקה!

דוגמא נוספת היא פרוטוקול האינטרנט UDP – User Datagram Protocol, שבו חבילות packets עוברות במהירות גבוהה, על חשבון אמינות. לכל חבילה מוצמד מספר סידורי, וכך, אם חבילה אבדה בדרך, אז ניתן להתיחס על המידע בה כאל מחיקה. האיור הבא ממחיש את הבעיה:



איור 11: נפילה של חבילות באינטרנט. אם חבילות 1 עד 6 היו מילה בקוד שיכול לתקן את המחיקות, אז היה ניתן לשחזר את כל החבילות.

נגדיר מקודד ומפענח Reed-Solomon כך שידע להתחשב במחיקות:

```
k = 16; n = 24; rsEnc = comm.RSEncoder(n,k,'BitInput',false);
rsDec = comm.RSDecoder(n,k,'BitInput',false,'ErasuresInputPort',true);
u = randi([0 n],k,1); c = rsEnc(u);
```

1. מחקו שני סימובולים מתוך מילת הקוד ופענחו:

```
tErase = 2; erasureLocations = [ones(tErase,1);zeros(n-tErase,1)];
erasureLocations = erasureLocations(randperm(n));
y = c; y(erasureLocations==1) = 0; m = rsDec(y,erasureLocations);
```

האם הפיענוח הצליח? מצאו ערך סף `t` כך שאם `tErase<=t` אז הפיענוח מוצלח, ואם `tErase>t` אז לא ניתן לפענח.

שימו לב

`comm.RSDecoder` מחזיר הודעת שגיאה אם מספר המחיקות גדול מיכולת התיקון של הקוד.

כעת נבצע סימולציה של streaming דרך האינטרנט. המדיה שאותה נזרים תהיה שיר. הריצו את הקטע הבא:

```
[samples,Freq] = audioread('Stream.flac');
player = audioplayer(samples,Freq);
```

השמיעו את השיר בעזרת play(player), והתרשמו ממנו למשך של מספר שניות. אתם יכולים לעצור את ההשמעה בעזרת הפקודה stop(player).

2. נדמה שימוש (פשוט) בפרוטוקול אינטרנט, ונארוז את המידע בתוך חבילות (packets). נניח שגודל כל חבילה הינו 32 בתים (32B), ומכיוון שדגימות השיר הן 8 בתים כל אחת, אז בכל חבילה יש 4 דגימות.

```
packetByteSize = 32; samplesPerPacket = packetByteSize/8;
numPackets = numel(samples)/samplesPerPacket;
packets = cell(numPackets,1);
for i = 1:numPackets
  packets{i} = samples((i-1)*samplesPerPacket+1:i*samplesPerPacket);
end
```

בעת שידור החבילות, חלקם לא הגיעו, ומכיוון שאנחנו יודעים להגיד מי לא הגיעה, אז ניתן להתייחס אליהן כאל מחיקות:

```
p=0.25; packetErasureLocation = binornd(1,p,numPackets,1);
packetErasureIndices = find(packetErasureLocation);
packetsRecieved = packets;
for i = 1:numel(packetErasureIndices)
  packetsRecieved{packetErasureIndices(i)}=zeros(samplesPerPacket,1);
end
```

מה מספר החבילות שנמחקו? מה המספר היחסי של חבילות שנמחקו (ביחס למספר החבילות שנשלחו)?

3. בצד המקבל צריך "לפרוק" את החבילות

```
samplesReconstruct = zeros(numPackets*samplesPerPacket,1);
for i = 1:numPackets
  samplesReconstruct((i-1)*samplesPerPacket+1:i*samplesPerPacket) = ...
  packetsRecieved{i};
end
```

השמיעו את השיר שמתקבל. האם האיכות נפגעה?

4. כדי לאפשר בכל זאת הזרמת מדיה איכותית, נשתמש בקוד לתיקון שגיאות. הריצו את הקטע הבא:

```
numInformationPackets = 5; numRedudantPackets = 2;
numCodewordPackets = numInformationPackets + numRedudantPackets;
k = packetByteSize*numInformationPackets;
n = packetByteSize*(numInformationPackets+numRedudantPackets);
rsEnc = comm.RSEncoder(n,k,'BitInput',false);
rsDec = comm.RSDecoder(n,k,'BitInput',false,...
```

```
  'ErasuresInputPort',true);
% encode
samplesInt = typecast(samples(:),'uint8');
msg = reshape(samplesInt,k,[]); msgEnc = uint8(zeros(n,size(msg,2)));
for j = 1:size(msg,2)
  msgEnc(:,j) = rsEnc(msg(:,j));
end
msgEnc = msgEnc(:);
```

הקוד בו השתמשנו הינו קוד Reed-Solomon באורך $n = (5+2)\cdot 32 = 224$, ומימד $k = 5\cdot 32 = 160$. כל דגימת שמע (בגודל 1B) הינה סימבול במילת קוד. כלומר, המקודד ממיר 160 דגימות (5 חבילות) ל־224 דגימות (7 חבילות).

5. אירזו את המידע בחבילות ושלחו אותן על־גבי הרשת

```
% pack
numEncPackets = numel(msgEnc)/packetByteSize;
packetsEnc = cell(numEncPackets,1);
for i = 1:numEncPackets
  packetsEnc{i} = msgEnc((i-1)*packetByteSize+1:i*packetByteSize);
end
% send
packetErasureLocation = ...
  zeros(numCodewordPackets,numEncPackets/numCodewordPackets);
for j = 1:size(packetErasureLocation,2)
  erasures = [ones(numRedudantPackets,1);...
          zeros(numInformationPackets,1)];
  packetErasureLocation(:,j) = erasures(randperm(numCodewordPackets));
end
packetErasureLocation = packetErasureLocation(:);
packetErasureIndices = find(packetErasureLocation);
packetsRecieved = packetsEnc;
numTotalErasedPackets = numel(packetErasureIndices);
for i = 1:numTotalErasedPackets
  packetsRecieved{packetErasureIndices(i)} = ...
        uint8(zeros(packetByteSize,1));
end
```

מה מספר החבילות שנמחקו? מה המספר היחסי של חבילות שנמחקו?

6. כעת נפענח את המידע שהתקבל:

```
% unpack
msgRecieved = uint8(zeros(numEncPackets*packetByteSize,1));
for i = 1:numEncPackets
  msgRecieved((i-1)*packetByteSize+1:i*packetByteSize) = ...
  packetsRecieved{i};
end
msgRecieved = reshape(msgRecieved,n,[]);
% translate packet erasures into symbol erasures
```

```
erasedSymbolsLocation = repmat(packetErasureLocation',packetByteSize,1);
erasedSymbolsLocation = reshape(erasedSymbolsLocation,n,[]);
% decode
msgDecoded = uint8(zeros(k,size(msgRecieved,2)));
for j = 1:size(msgRecieved,2)
  msgDecoded(:,j) = rsDec(msgRecieved(:,j),erasedSymbolsLocation(:,j));
end
```

האם הפיענוח מושלם?

7. כדי לקבל בחזרה את השיר, נמיר את הדגימות להיות בפורמט נכון:

```
samplesDecoded = msgDecoded(:);
samplesDecoded = reshape(samplesDecoded,8,[]).';
samplesReconstruct = zeros(numel(samplesDecoded)/8,1);
for i = 1:numel(samplesReconstruct)
  samplesReconstruct(i) = typecast(samplesDecoded(i,:),'double');
end
```

השמיעו את השיר המפוענח ובדקו האם השיחזור הצליח.

הרבה פעמים מחיקות (וגם שגיאות) מגיעות בפרצים (bursts), למשל, כאשר הקשר בין שתי נקודות הקצה נעלם לזמן מסוים ואז חוזר. בחלק זה נדמה מצב שבו קו התקשורת איכותי ויציב לאורך כל השידור, פרט לרגע אחד שבו הוא נופל לזמן ארוך ואז חוזר. בזמן שהקו היה למטה, "נמחקו" 10000 חבילות!

8. הריצו את הקטע הבא:

```
burstLength = 10000; burstStart = randi([1 numEncPackets-1000]);
packetErasureLocation = zeros(numEncPackets,1);
packetErasureLocation(burstStart:(burstStart+burstLength))=1;
packetErasureIndices = find(packetErasureLocation);
packetsRecieved = packetsEnc;
numTotalErasedPackets = numel(packetErasureIndices);
for i = 1:numTotalErasedPackets
  packetsRecieved{packetErasureIndices(i)} = ...
          uint8(zeros(packetByteSize,1));
end
```

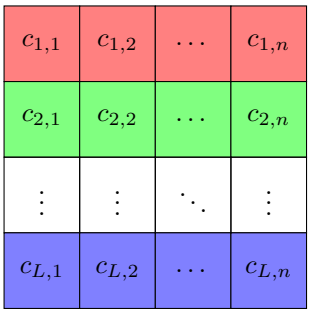
מה מספר החבילות שנמחקו? מה המספר היחסי של חבילות שנמחקו? השוו עם סעיף 5.

9. נסו לפענח את המידע שהתקבל. האם הפיענוח הצליח? הסבירו.

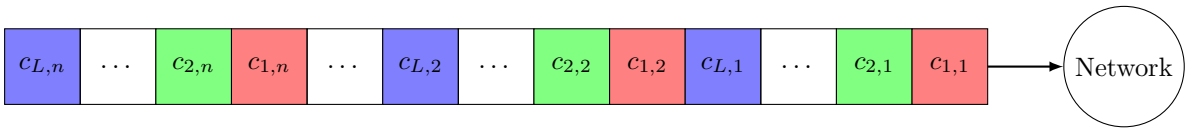
כדי להתגבר על בעיית הפרצים נשתמש בשיטת השזירה (Interleaving). מטרת השזירה היא לחלק את הפרץ באופן אחיד בין מילות קוד שונות, ובכך למנוע מצב שמילות קוד שלמות נמחקות. יש מספר שיטת לשזירה ואנחנו נתמקד בפשוטה מכולם (אך לא היעילה ביותר), שנקראת Block Interleaving. בשיטה זו, מכניסים L מילות קוד באורך n לתוך מערך מלבני בגודל $L \times n$ בצורה הבאה:

ומוציאים אותם לפי עמודות בצורה הבאה (מימין לשמאל):

10. הריצו את הקטע הבא:



איור 12: L מילות קוד באורך n בכניסה לשוזר



איור 13: L מילות קוד באורך n ביציאה מהשוזר

```
packetsEncInter =  packetsEnc;
packetsEncInter = reshape(packetsEncInter,numCodewordPackets,[]);
packetsEncInter = packetsEncInter.';
packetsEncInter = packetsEncInter(:);
```

העבירו את החבילות השזורות בערוץ עם פרץ השגיאות הארוך, ובצעו את הפעולה ההפוכה לשזירה עבור המידע המורעש ועבור מיקומי המחיקות. לאחר מכן, פענחו את המידע. האם יש פיענוח מושלם? הסבירו מה השתנה לעומת סעיף קודם.

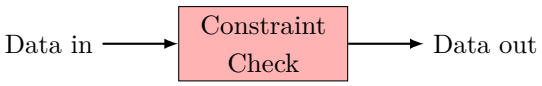
שימו לב
זכרו לבצע דה־שזירה גם למיקומי המחיקות!

שאלות מחשבה לסיכום

1. סעיף 1: מה הקשר בין ערך הסף `tErase` לבין פרמטרי הקוד `n,k`?
2. למה הפיענוח בסעיף 6 הצליח והפיענוח בסעיף 9 נכשל? האם מפני שנמחקו יותר חבילות?
3. למה הפיענוח בסעיף 9 נכשל והפיענוח בסעיף 10 נכהצליח, למרות ששידרנו על אותו ערוץ ועם אותו הקוד?

ניסוי 7.ב - קידוד עם אילוצים

בהרבה תרחישים ערוץ התקשורת שעליו יש להעביר מידע איננו אקראי לגמרי, ואם המידע שמועבר בו עומד באילוצים מסוימים, אז מובטח שהוא לא ישתבש בכלל. בניסוי זה נתרכז בערוץ בינארי נפוץ אשר איננו סובל רצף של יותר מביט אחד שערכו '1'. כלומר המחרוזת `101001` בכניסה לערוץ עומדת בתנאים (מותר רצף אפסים), אבל המחרוזת `101101` איננה תקינה. כמוטיבציה לערוץ זה חשבו על התקן שבו ביט שערכו '1' מיוצג על־ידי פולס אלקרטו־מגנטי, ושדרוש להתקן זמן התאוששות של לפחות מחזור שעון אחד בין פולס לפולס.



איור 14: ערוץ עם אילוצים

1. נתון קובץ בינארי באורך 6000 ביטים. `binFile = randi([0 1],1,6000);`. היעזרו בפונקציה `ConstChannel` שסופקה לכם, והעבירו את הקובץ דרך הערוץ. למה יש הודעת שגיאה?

על־מנת להעביר את הקובץ בערוץ, נמנע מצב שבו יש רצף של יותר מ־'1' יחיד על ידי הכנסת '0' אחרי כל ביט. כלומר, רצף הביטים

$$b_1,\, b_2,\, b_3, \ldots, b_{n-2},\, b_{n-1},\, b_n$$

יקודד ל־

$$b_1,\, 0,\, b_2,\, 0,\, b_3, \ldots, b_{n-2},\, 0,\, b_{n-1}, 0,\ b_n,\, 0$$

.

2. ממשו את המקודד הנ"ל, וקודדו את `binFile` בעזרתו. מה קצב הקוד? כמה ביטים יש באות המקודד?

3. העבירו את האות המקודד בערוץ. האם ישנה הודעת שגיאה? אם כן, בידקו את עצמכם. אם לא, חלצו מתוך האות מהערוץ את ביטי האינפורמציה, וודאו שקיבלתם בחזרה אות שזהה ל־ `binFile`.

נרצה לייעל את התהליך ולקבל קוד בקצב גבוה יותר.

4. מצאו קוד עם 8 מילים בינאריות באורך 5 ביטים אשר מקיים: 1) אין רצף אחדים באף מילת קוד, ו־2) אף מילת קוד לא מסתיימת ב־'1'. כתבו בדו"ח מיפוי כניסה ויציאה עבור הקוד מהצורה הבאה:

in	out
000	
001	
⋮	
111	

מה קצב המקודד? כמה ביטים יהיו באות המקודד אם נקודד את binFile בעזרת המקודד הנ"ל? השוו עם סעיף 2.

שימו לב

אין צורך לקודד את binFile, אלא רק לציין את מספר הביטים באות המקודד.

נציע מקודד אחר בו מילות הקוד באורך משתנה:

in	out
0	0
1	10

5. קודדו את binFile בעזרת הקוד עם אורך משתנה. השתמשו בקוד מהצורה הבאה:

```
binFileEnc = []; i=1;
while (i<= numel(binFile))
  if (...)
    binFileEnc = [binFileEnc ...];
  else
    binFileEnc = [binFileEnc ...];
  end
end
```

כמה ביטים יש באות המקודד binFileEnc? השוו עם סעיף 4.

6. העבירו את האות המקודד בערוץ וחלצו ממוצא הערוץ את האינפורמציה ששודרה. וודאו שקיבלתם בחזרה אות שמזדהה עם binFile.

ניסוי 8 - רעש בתמונות

לתמונות יש מבנה שונה מזה של תווים שמתארים שפה, למשל מכיוון שערכי פיקסלים שכנים צפוים להיות דומים. MATLAB מייצג תמונות בעזרת מטריצה בגודל התמונה, שבה כל פיקסל זה אלמנט במטריצה. אנחנו נתמקד בתמונות בגווני אפור, שבהם כל פיקסל מקבל ערך בין 0 (שחור לגמרי) עד 255 (לבן לגמרי). במצב כזה, האלמנטים של המטריצה הינם מטיפוס `'uint8'`, כלומר כל פיקסל מיוצג על ידי 8 ביטים כאשר 00000000 מייצג ערך 0, ו־11111111 מייצג 255.

1. הציגו את התמונה: `im = imread('NY.gif'); figure(); imshow(im,[]);`. הפכו את התמונה לרצף ביטים בעזרת `msg = im2bit(im);`. כמה ביטים יש בתמונה?

2. המירו את הביטים חזרה: `imHeight = size(im,1); imRe = bit2im(msg,imHeight);`, וודאו שקיבלתם בחזרה את התמונה המקורית.

3. העבירו את רצף הביטים שמייצג את התמונה בערוץ $\mathrm{BSC}(p = 10^{-4})$, והציגו את התמונה המתקבלת אחרי הרעשה זו. מצאו פיקסל שעבר הרעשה וענו על השאלות הבאות:

   - איך יכול להיות שהצלחתם למצוא פיקסל מורעש למרות שהערוץ לא סיפק מידע זה?
   - האם אתם יכולים לשער מה היה ערך הפיקסל המקורי (לפני הרעשה)?
   - איך יכול להיות שניתן לדעת מה היה ערך הפיקסל המקורי למרות שלא הגנתם על המידע בעזרת קוד תיקון שגיאות?

   קראו למדריך והסבירו לו את התשובות שלכם.

בפועל, יש מבנה ויתירות רבה בתמונות, וכדי לחסוך במשאבים דוחסים אותן ושולחים את המידע הדחוס. הריצו את הפקודה הבאה: `CI = imCompress(im);`. המשתנה `CI` הינו `struct` אשר מהווה את הקבוץ הדחוס. השדה העיקרי שלו הינו `CI.data` אשר מכיל את המידע הדחוס בצורה של רצף ביטים.

4. מה גודל `CI.data`? פי איזה פקטור הדחיסה הקטינה את התמונה (התעלמו משאר השדות של `CI`)? שחזרו את התמונה בעזרת הפקודות הבאות:

```
figure(); imshow(imDecompress(CI.data, CI.height, CI.dict),[]);
```

5. העבירו את `CI.data` בערוץ $\mathrm{BSC}(p = 10^{-4})$, והציגו את התמונה המתקבלת. האם ניתן לתקן עכשיו?

כדי להעביר בבטחה את התמונה הדחוסה בערוץ, נגן עליה בעזרת קוד לתיקון שגיאות. לצורך כך ישנו תקציב של לכל היותר 20% תוספת של **ביטי** יתירות למידע הדחוס. נרצה להשתמש בקוד BCH בינארי (ראו ניסוי 4 במפגש הראשון). השאלה היא איך לבחור את פרמטרי הקוד n, k?

6. בחרו `n=2^4-1` והציגו את `bchnumerr(n)`. מה ערכי `k` האפשריים? כמה אחוז תוספת יתירות יש לכל אחד מערכים אלו? מה יכולת התיקון של כל אחד מערכים אלו?

7. תכננו מערכת תקשורת - העומדת בתקציב היתירות שהוגדר - אשר תקודד את התמונה הדחוסה, תעביר אותה בערוץ $\mathrm{BSC}(p = 10^{-4})$, תתקן את השגיאות שהערוץ הכניס, ולבסוף תציג את התמונה. הראו את המערכת למדריך. מה קצב הקוד שבו השתמשתם? כמה ביטים בסך־הכחל נשלחו בערוץ?

שימו לב

- אורך הקוד n מוגבל להיות מהצורה $2^m - 1$, עבור m שלם שנע בין 3 ל־16.
- אובייקטים מסוג `comm.BCHEncoder`,`comm.BCHDecoder` פועלים על ווקטורי עמודה בלבד.
- כדי לקודד את התמונה הדחוסה, יש צורך לפרק את `CI.data` לבלוקים בגודל k. ייתכן ש־`CI.data` לא מתחלק בדיוק ב־k, ולכן יש צורך בריפוד אפסים. ראו ניסוי 4 במפגש הראשון.

שאלות מחשבה לסיכום

1. סעיף **??**: מה משמעות המשתנים `m,k,n`, ומה סוג קוד תיקון השגיאות שהשתמשנו בו?
2. סעיף **??**: מה ניתן להסיק על יכולות התיקון של הקוד?
3. סעיף **??**: מה מרחק הקוד בו נעשה שימוש. מה הקשר בינו לבין יכולת התיקון של הקוד?
4. סעיף 3: איך יכול להיות שניתן לתקן את השגיאות מהערוץ ללא קוד לתיקון שגיאות? באיז מידע השתמשתם כש"ניחשתם" את ערך הפיקסלים בתמונת השחזור?
5. סעיף 5: מדוע לא ניתן לשחזר את התמונה למרות שהערוץ הרועש בן השתמשתם הינו אותו ערוץ כמו בסעיף 3?
6. סעיף 7: השוו את מספר הביטים הכולל שנשלחו בערוץ, עם מספר הביטים שהועברו ללא דחיסה וללא קוד בסעיף 3. מה מסקנותיכם?
7. סעיף 7: האם לדעתכם עדיף היה קודם לקודד את התמונה ואחר־כך לדחוס את התמונה המקודדת? הסבירו.

01000010 01101001 01110100 01110011 00100000 01100001 01110010 01100101 00100000 01110100
01101000 01100101 00100000 01100010 01100001 01110011 01101001 01100011 01110011 00100000
01101111 01100110 00100000 01101001 01101110 01100110 01101111 01110010 01101101 01100001
01110100 01101001 01101111 01101110

בהצלחה!