1 Administrative Instructions

1. The take home exam must be submitted individually.

2. For any clarification or doubts, please contact both the TA, Weiqi Sun (email: wsun23 AT ncsu DOT edu), and instructor, Dror Baron (email: barondror AT ncsu DOT edu).

3. The test will start on May 4 at noon, and is due 24 hours later. We may add extra time if needed. (Students who needed accommodation emailed us by 5 pm on April 27.)

4. You should submit an electronic copy via Moodle. In case of late submissions caused, for example, by internet issues, we will deduct 1% per minute.

5. Please submit a report in one pdf file. The report can be comprised of screenshots, handwritten derivations, typed solutions, and so on.

6. There is a WebWork component to the take home exam. You will receive separate instructions for that.

7. Please justify your answers carefully.
2 Background

Group testing involves looking at $N$ objects, and trying to identify a small number of special objects, $K \ll N$, that have some special property. For example, the objects could be car parts, and you are testing whether a part is defective or not. A second example involves coins; you are examining $N = 1000$ coins, and based on your experience $p = 1\%$ of the objects are fake; this ratio is often called the prevalence. You expect $Np = 1000 \times 0.01 = 10$ fake coins; the specific number might be a bit larger or smaller. A third example involves people who might have a disease. The flu, for example.

How many measurements, $M$, do you need to identify the $K$ special objects? If you have to test every single object, you need $N$ measurements or tests. That is simple enough. But can we do better? Can we identify the $K \ll N$ special objects using $M < N$ measurements?

2.1 Adaptive group testing

In some applications, we can test multiple objects at once. For example, suppose that a regular coin weighs 12 grams, and a fake one is 13 grams. If you weigh batches of $B = 20$ coins at a time, and a batch weighs $12B = 240$ grams, then the coins are all genuine. Else if the batch weighs 241 grams, we have a fake coin.\footnote{With 242 grams, there are 2 fakes. And so on.} Because $p = 1\%$, most batches of $B = 20$ have no fake coins, and a single measurement shows that. Occasionally the weight will be 241 (or more) grams, and we can then weigh all $B$ coins in the batch individually.\footnote{Students who enjoy math riddles can surely identify a single fake coin out of $B = 20$ using less than $B$ measurements; we ignore this for now.} This approach is called adaptive group testing, because we only decide who to test individually after seeing which batches contain fake coins. The adaptive component allows us to focus individual tests on objects (in this case coins) more likely to be special (fake).

For this example, it can be shown that roughly 18\% of batches will contain (at least one) fake coin. That is, for $N = 1000$ coins, we have $N/B = 1000/20 = 50$ batches. In Part 1 of this group testing approach, we measure each one, which requires $M_1 = 50$ measurements. Next, $18\% \times 50 = 9$, and we have (around) 9 batches containing a fake. In Part 2 of this group testing approach, we need $B = 20$ measurements for each of these 9, which is $M_2 = 9 \times 20 = 180$ measurements. Combining the $M_1 = 50$ measurements of Part 1 and $M_2 = 180$ of Part 2, we need a total of $M = M_1 + M_2 = 50 + 180 = 230$ measurements.

Tasks related to adaptive group testing appear in Section 3.1. In particular, we will study the trade-off between the batch size, $B$, and number of measurements, $M$. 
2.2 Why is group testing useful?

Testing coins with known weights is relatively straightforward, because we know the weights of genuine and fake coins, and the weight of a batch is informative. There are other settings where group testing is also useful.

Group testing originated during World War 2, when the military wanted to screen potential soldiers for syphilis. Testing for syphilis required a blood sample. Because the syphilis test was expensive, it seemed beneficial to pool together $B$ soldiers’ blood samples and evaluate the pooled blood samples together in one test. Group testing is most effective when the prevalence probability is low, because many objects can be ruled out in each batch.

A modern application of group testing is to search for HIV in blood donations. Because blood donations are not collected from people who have tested positive for HIV, the prevalence is very low.

Group testing offers multiple potential advantages in applications where it can be used. First, if tests are costly and pooling multiple objects into a single test is relatively simple, group testing can identify special objects at lower cost. Second, if testing equipment is being run in a pipelined fashion, reducing the number of tests required for some population can improve throughput, and even reduce latency. *All things being equal, we want to reduce the number of measurements, $M$, as much as possible.*

2.3 Non-adaptive group testing

Adaptive group testing uses Part 1 to test suspicious objects individually in Part 2, thus adapting the testing structure. However, adaptive group testing requires (at least) two round trips of test delays, because we must wait for the test results of Part 1 before deciding who to test in Part 2. This problem is further exacerbated in more sophisticated adaptive group testing procedures with 3 or more parts.

You can envision an application such as testing for COVID19 where some testing procedures take several hours; we want to get results to people quickly, and waiting for two (or more) rounds of tests could be troublesome. An alternative to adaptive group testing is *non-adaptive group testing*, where all tests are conducted in one round. Although the number of measurements, $M$, may increase mildly, the faster response time is often worth it.

**Problem formulation.** We now describe a non-adaptive problem formulation. As before, consider a vector $x$ of length $N$. For $n \in \{1, \ldots, N\}$, entry $n$ of $x$ is denoted by $x(n)$, and its value is 1 if object $n$ is special, else $x(n) = 0$. Next, we have a matrix $A$ with $M$ rows and $N$ columns. We denote the entry of $A$ in row $m \in \{1, \ldots, M\}$ and column $n \in \{1, \ldots, N\}$ by $A_{mn}$. Row $m$ of $A$ describes the structure of measurement $m$, and in particular the ones in row $m$ appear in columns $n$ that correspond to objects being tested by measurement $m$. Similarly, the ones in column $n$ correspond to measurement that test object $n$. In summary, if $A_{mn} = 1$, then object $n$ is tested by measurement $m$, else $A_{mn} = 0$, and object $n$ is not tested by measurement $m$. 
In addition to the objects vector \( x \) and matrix \( A \), our \( M \) measurements are arranged in a vector denoted by \( y \). Entry \( m \) of \( y \) is denoted by \( y(m) \), and it is 1 if at least one of the objects being tested, i.e., the objects being tested by measurement \( m \) correspond to indices \( n \in \{1, \ldots, N\} \) in row \( m \) where \( A_{mn} = 1 \), is special. If none of the objects being tested by measurement \( m \) is special, then \( y(m) = 0 \).

**Example.** Consider \( N = 5 \) objects and the vector \( x = [0 \ 1 \ 0 \ 0 \ 0]^T \), where \((\cdot)^T \) denotes vector transpose, meaning that \( x \) is a column vector. You can see that \( x(n) = 1 \) only for \( n = 2 \), and only object 2 is special.

Next, our matrix is

\[
A = \begin{bmatrix}
1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0
\end{bmatrix}.
\]

For example, row 2 is \([0 \ 1 \ 0 \ 1 \ 1]\), and it tests objects 2, 4, and 5. Because object 2 is special, \( y(2) = 1 \), meaning that measurement 2 is positive. Similarly, \( y(1) = 1 \) and \( y(3) = 0 \).

**From \((y, A)\) to \(x\).** In practice, you do not know \( x \). Instead, you are given the measurements vector \( y \) and matrix \( A \), and are asked to estimate \( x \).

How can we solve \( x \) from \( A \) and \( y \)? In our example, \( y(3) = 0 \), and none of the objects it tests are special. Row 3 is \([1 \ 0 \ 1 \ 0 \ 1]\), meaning that it tests objects 1, 3, and 5. None of them are special. The remaining objects that have not been ruled out are objects 2 and 4. Object 2 was tested by rows 1 and 2, and both tested positive, i.e., \( y(1) = 1 \) and \( y(2) = 1 \). Therefore, it is plausible (and indeed seems likely) that object 2 is special. What about object 4? It was only tested by measurement 2, which was positive. Therefore, object 4 could also be special. Because object 4 was only tested once, whereas object 2 was tested twice, it seems more plausible that object 2 is special, in which case \( y(2) \) would be positive because of object 2, and not object 4.

If we were given another measurement corresponding to a row of the form \([1 \ 0 \ 0 \ 1 \ 1]\), then this extra measurement would evaluate object 4 (along with objects 1 and 5, which have already been ruled out). Because object 4 is not special, this extra measurement would be negative, and we could rule out object 4. The main point here is that negative measurements (zeros in \( y \)) rule out objects they test from being special, and positive measurements (ones in \( y \)) indicate that the corresponding objects might be special. (If these ideas seem to resemble those discussed for compressed sensing, you are correct. Indeed, some researchers have proposed to use compressed sensing algorithms or variations thereof for group testing.)

**Challenges.** Our discussion shows that it is imperative for \( A \) to test each object multiple times. There are other considerations in matrix design for non-adaptive group testing, and this is an active research area.

Another challenge is how to deal with errors in tests. For example, in our example, if \( y(1) \) mistakenly tested negative, and only \( y(2) \) was positive, then object 2 would be ruled out by \( y(1) \), and suddenly object 4 would be the only object that could be special while satisfying all the measurements. Group
testing with erroneous measurements is also an active research area. Tasks related to non-adaptive group testing appear in Section 3.2.

3 Tasks

For the tasks below, make sure to include relevant MATLAB code, plots, and explanations as needed. Responses such as, “B should be -7,” will not be helpful.

3.1 Adaptive group testing

Recall that adaptive group testing pools batches of B objects in Part 1, with a single test often ruling out an entire pool, and Part 2 tests individual objects in pools that tested positively. What is the optimal batch size, B? To evaluate this, simulate a large number of coins, \( N = 10^5 \) or more. To keep things simple, denote a genuine coin by the label 0, and a fake one by 1. For your convenience, below is MATLAB code that generates a vector of \( N \) labels, where \( p = 1\% \) are 1, the rest 0.

\[
\begin{align*}
N &= 1e5; \quad \% \text{ number of coins} \\
p &= 0.01; \quad \% \text{ prevalence} \\
x &= \text{rand}(N,1) < p; \quad \% \text{ see explanation in text below} \\
\text{fprintf('Min(x) = %1d\n',min(x))}; \quad \% \text{ prints 0} \\
\text{fprintf('Max(x) = %1d\n',max(x))}; \quad \% \text{ prints 1} \\
\text{fprintf('Sum(x) = %6d\n',sum(x))}; \quad \% \text{ prints number close to 1000} \\
\text{fprintf('Number ones = %6d\n',sum(x==1))}; \\
\text{fprintf('Number zeros= %6d\n',sum(x==0))};
\end{align*}
\]

We now explain how we generated the binary vector \( x \), which is comprised of \( N \) bits (each entry of \( x \) is 0 or 1). Matlab’s \text{rand(\cdot)} command returns a vector of \( N \) random numbers whose values are between 0 and 1. Any interval of width \( \Delta \) between 0 and 1 will contain roughly \( N\Delta \) of the random numbers. By evaluating which of these \( N \) random numbers is less than the prevalence, \( p \), i.e., these numbers are elements of the interval \((0, p)\), which is of width \( p \), we assign 1 to (approximately) \( Np \) positions in the vector \( x \).

Simulation. Having generated \( N \) numbers, evaluate them in batches of several sizes. Using \( N = 10^5 \), \( x \) can be partitioned into an integer number of blocks when \( B \) divides \( N \) nicely; we recommend \( B \in \{5, 10, 20, 40\} \). For each such \( B \) (can use other values in light of Footnote 4), simulate the number of batches that contain at least one fake coin.

To do so, partition \( N \) into \( N/B \) non-overlapping batches. For each batch, evaluate whether it contains (at least one) fake coin. Count the number

---

\[\text{(Footnote 4)}\] Some of you may want to consider \( B \) that divides \( N \) with a remainder, where the last batch contains less than \( B \) coins.

\[\text{(Footnote 3)}\] When the random number is less than \( p \), the condition, \( \text{rand}(N,1) < p \), is true and returns 1. Else the random number is larger or equal to \( p \), the condition is false, and returns 0.
of batches that contain a fake coin, and multiply by $B$ to obtain $M_2$, the number of individual measurements required for Part 2. Compute the total number of measurements required for these two parts: (i) Part 1 is when you evaluate each of the $N/B$ batches using a single measurement; (ii) Part 2 measures all $B$ coins individually within each problematic batch. The total number of measurements is the sum of the number of measurements required in Parts 1 and 2.

**Numerical illustration.** Plot the number of measurements required for Part 1, Part 2, and in total; these should be plotted as a function of $B$. For your convenience, here is possible MATLAB code for plotting this.

```matlab
% Before this code we computed:
% M1 = number of measurements required for Part1
% M2 = number of measurements required for Part2
% M3 = M1+M2
plot(B,M1,'*-',B,M2,'o-',B,M3,'+-');
xlabel('Batch size B');
ylabel('Number of measurements');
legend('Part1','Part2','Total');
```

**Discussion of $M_1$ and $M_2$.** Discuss $M_1$, the number of measurements required for Part 1, as a function of $B$. Does it increase with $B$, decrease with $B$, other? Please explain why. Similarly, discuss $M_2$, the number of measurements required for Part 2.

**Discussion of $M$.** Define $M = M_1 + M_2$, which is the total number of measurements required for both parts. Does it increase with $B$, decrease with $B$, other? Do you see trade-offs in the selection of $B$?

**Optional.** Feel free to refer to Footnote 4 and run your code on a larger range of possible batch sizes, $B$. You can even change the prevalence probability, $p$, and evaluate how it affects the trade-off between $B$ and $M$. (Note that we may provide modest extra credit for creative responses. We advise you to ignore this part unless you finish the exam quickly.)

### 3.2 Non-adaptive group testing

**Main idea.** In this task, we will give you data for the matrix, $A$, and measurements, $y$. You will need to solve $x$. We have tested the problem carefully, and are confident that there is one unique solution (one possible vector $x$) that explains $y$ given the testing dependencies $A$ between them. Your role is to download data containing $y$ and $A$, and then compute $x$.

**How to compute $x$?** Recall our example in Section 2.3, where we discussed two properties. First, if $y(m) = 0$, all objects tested by that measurement, i.e., objects $m$ such that $A_{mn} = 1$, are negative. This allows us to rule these objects out. Similarly, if $y_m = 1$ and only one object being measured has not been ruled out, then that object must be special.

There are different ways how you can use these observations. Some of you will want to approach the problem iteratively, while gradually reducing the size of the remaining problem. Others may have some special insight. You
can solve this problem by hand or using MATLAB code. *We will be happy to read through any thoughtful approach!*

**Problem setting.** The problem you will download has the following parameters. We have \( N = 30 \) objects, and \( K = 2 \) of them are special. We evaluate these objects using \( M = 15 \) measurements (note that \( M/N = 0.5 \)). In the matrix, \( A \), each row contains 6 ones. Note that the average number of ones per row (6) multiplied by the number of rows (\( M = 15 \)) equals the average number of ones per column times the number of columns (\( N = 30 \)). Therefore, the average number of ones per column is \( 15 \cdot 6/30 = 3 \). To simplify things, each column of \( A \) contains exactly 3 ones.

With \( K = 2 \) special objects, your role is to derive the two indices \( n_1, n_2 \in \{1, \ldots, N\} \) that correspond to the special objects. Your final answer could have the form, “the two special objects are 13 and 22.”

**Downloading data.** A file containing data, ece421_data.mat, will be available. After downloading the file, you can load it to MATLAB using the following code snippet.

```
load ece421_data
```

Once you load this file, you will see 3 data structures. Each of them contains 100 versions of the data. *Use the version of \( y \) and \( A \) that corresponds to the last 2 digits of your student id;* the last 2 digits select between 100 versions. Example MATLAB code for doing so appears below.

- The matrix \( y_{\text{all}} \) is of size 100 \( \times \) 15. Each of the 100 columns contain a different length-15 column.

- The matrix \( A_{\text{all}} \) is of size 100 \( \times \) 15 \( \times \) 30. This is a 3-dimensional matrix, and each of the 100 versions is a matrix of size 15 \( \times \) 30.

- For your convenience, we have also included another matrix, \( A_{\text{list all}} \), of size 100 \( \times \) 30 \( \times \) 3. Recall that each of the \( N = 30 \) columns of your \( A \) matrix (accessible in \( A_{\text{all}} \)) contains exactly 3 ones; the positions of these 3 ones appear in your version of the 30 \( \times \) 3 matrix.

Some of you may want to work with \( A \) directly, while others may prefer the list approach. Again, we will be happy to read through any thoughtful approach.

To help you load your version of the problem, below is example MATLAB code that does so.

```
id=37; % suppose that my student id ends with 37
y=squeeze(y_all(id,:)); % squeeze returns length-15 vector
A=squeeze(A_all(id,:,:)); % returns 15x30 matrix
A_list=squeeze(A_list_all(id,:,:)); % returns 30x3 matrix
```

(If by chance your id ends with the digits 00, you should use \( \text{id}=100 \) in the MATLAB snippet above.)

**What do we expect?** Walk us through your steps in determining \( x \) from \( y \) and \( A \). Again, there should be one unique \( x \) that explains \( y \) and \( A \).
Figure 1 shows a sequence $x(n)$ where the value of $x(3)$ is an unknown constant, $c$. (The sample with amplitude $c$ is not necessarily drawn to scale.)

Let $X(k)$ be the 5-point *discrete Fourier transform* (DFT) of $x(n)$. Consider a second DFT, $X_1(k)$,

$$X_1(k) = X(k)e^{j2\pi 3k/5}.$$

The sequence $x_1(n)$ is obtained by applying the inverse DFT to $X_1(k)$; $x_1(n)$ is plotted in Figure 1. What is the value of $c$?

![Figure 1: Sequences $x(n)$ and $x_1(n)$.](image-url)
Consider a complex valued continuous-time signal $x_c(t)$. Whereas real valued signals have a conjugate symmetric Fourier response, complex valued signals need not. Indeed, the Fourier transform shown in Figure 2 is not symmetric. (Note that $\Omega_2 > \Omega_1$, and $\Omega_2 - \Omega_1 = \Delta \Omega$.) This continuous time signal, $x_c(t)$, is sampled, yielding the complex valued discrete time signal, $x(n) = x_c(nT)$.

![Fourier transform of $x_c(t)$](image)

**Figure 2:** Fourier transform of $x_c(t)$.

(a) Suppose that the sampling period is $T = \pi / \Omega_2$. Please sketch the Fourier transform, $X(e^{j\omega})$, of the discrete time sequence, $x(n)$. Make sure to label your axes appropriately.

(b) What is the Nyquist rate? In words, please derive the lowest sampling frequency that can be used for sampling $x_c(t)$ such that it can later be reconstructed perfectly from $x(n)$. 
(c) How would you reconstruct $x_c(t)$ from $x(n)$? You can assume that the sampling rate exceeds the Nyquist rate computed in Part (b), and that ideal filters are available, even if they are complex valued, non-causal, and so on. Please sketch a block diagram of a system that can reconstruct $x_c(t)$ perfectly from $x(n)$. Make sure to justify your design carefully.