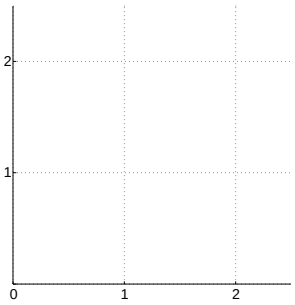
CS 70 Discrete Mathematics and Probability Theory
Summer 2016 Dinh, Psomas, and Ye Discussion 8A Sol

First two questions are rehashes of 7D. Skip if covered already.

1. Visualizing error correction Alice wants to send a message of 2 packets to Bob, and wants to guard against 1 lost packet. So working over $GF(3)$, she finds the unique polynomial $P(x)$ that passes through the points she wants to send, and sends Bob her augmented message of 3 packets: $(0,P(0)),(1,P(1)),(2,P(2))$.

One packet is lost, so Bob receives the following packets: $(0,2),(2,0)$.

1. Plot the points represented by the packets Bob received on the grid below.



2. Draw in the unique polynomial $P(x)$ that connects these two points.
3. By visual inspection, find the lost packet $(1,P(1))$.

TAs: Now, drawing on the board, expand the field (to, say, GF(7)). We now know that the points Alice wanted to send were $2,1$. Encoding these over GF(7) we get the polynomial $P(x) = 6x+2$. What if we didn't just have dropped packets, but had malicious errors? Start drawing in malicious errors so the students can see until when error-correction could work, how many points you would need to send, etc. Try to do this interactively.

2. Where are my packets?

Alice wants to send the message (a_0,a_1,a_2) to Bob, where each $a_i \in \{0,1,2,3,4\}$. She encodes it as a polynomial P of degree ≤ 2 over $GF(5)$ such that $P(0) = a_0$, $P(1) = a_1$, and $P(2) = a_2$, and she sends the packets $(0,P(0))$, $(1,P(1))$, $(2,P(2))$, $(3,P(3))$, $(4,P(4))$. Two packets are dropped, and Bob only learns that $P(0) = 4$, $P(3) = 1$, and $P(4) = 2$. Help Bob recover Alice's message.

1. Find the multiplicative inverses of $1,2,3$ and 4 modulo 5.
   Inverse pairs mod 5: $(1,1),(2,3),(4,4)$.

2. Find the original polynomial P by using Lagrange interpolation or by solving a system of linear equations.

$$\begin{aligned}
\Delta_0 &= \frac{(x-3)(x-4)}{(0-3)(0-4)} = \frac{x^2-7x+12}{(-3)(-4)} = 3(x^2+3x+2) = 3x^2+4x+1 \\
\Delta_3 &= \frac{(x-0)(x-4)}{(3-0)(3-4)} = \frac{x^2-4x}{(3)(-1)} = 3(x^2+x) = 3x^2+3x \\
\Delta_4 &= \frac{(x-0)(x-3)}{(4-0)(4-3)} = \frac{x^2-3x}{(4)(1)} = 4(x^2+2x) = 4x^2+3x
\end{aligned}$$

Thus, our original polynomial P is

$$\begin{aligned}
4\Delta_0 + 1\Delta_3 + 2\Delta_4 &= 4(3x^2+4x+1) + (3x^2+3x) + 2(4x^2+3x) \\
&= (2x^2+x+4) + (3x^2+3x) + (3x^2+x) \\
&= 3x^2+4
\end{aligned}$$

Linear equation way: Writing $P(x) = m_2x^2 + m_1x + m_0$, we solve for the m_i's by solving the linear equation

$$\begin{bmatrix} 0 & 0 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \end{bmatrix} \begin{bmatrix} m_2 \\ m_1 \\ m_0 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}$$

This gives the equation

$$\frac{1}{2}x^2 - \frac{5}{2}x + 4,$$

which, in the modulo 5 world, means $P(x) = 3x^2+4$.

3. Berlekamp–Welch algorithm

In this question we will go through an example of error-correcting codes with general errors. We will send a message (m_0, m_1, m_2) of length $n = 3$. We will use an error-correcting code for $k = 1$ general error, doing arithmetic modulo 5.

(a) Suppose $(m_0, m_1, m_2) = (4, 3, 2)$. Use Lagrange interpolation to construct a polynomial $P(x)$ of degree 2 (remember all arithmetic is $\bmod 5$) so that $(P(0), P(1), P(2)) = (m_0, m_1, m_2)$. Then extend the message to lengeth $n+2k$ by appending $P(3), P(4)$. What is the polynomial $P(x)$ and what is the message $(c_0, c_1, c_2, c_3, c_4) = (P(0), P(1), P(2), P(3), P(4))$ that is sent?

We use Lagrange interpolation to construct the unique quadratic polynomial $P(x)$ such that $P(0) = m_0 = 4, P(1) = m_1 = 3, P(2) = m_2 = 2$.

$$\begin{aligned}
\Delta_0(x) &= \frac{(x-1)(x-2)}{(0-1)(0-2)} = \frac{x^2-3x+2}{2} \\
\Delta_1(x) &= \frac{(x-0)(x-2)}{(1-0)(1-2)} = \frac{x^2-2x}{-1} \\
\Delta_2(x) &= \frac{(x-0)(x-1)}{(2-0)(2-1)} = \frac{x^2-x}{2} \\
P(x) &= m_0\Delta_0(x) + m_1\Delta_1(x) + m_2\Delta_2(x) \\
&= 4\Delta_0(x) + 3\Delta_1(x) + 2\Delta_2(x) \\
&= -x+4
\end{aligned}$$

[Note that all arithmetic is $\bmod 5$, so for example $2^{-1} \equiv 3 \pmod 5$]. Then we compute $P(3) = 1$ and $P(4) = 0$, so our message is 43210.

(b) Suppose the message is corrupted by changing c_0 to 0. We will locate the error using the Berlekamp–Welsh method. Let $E(x) = x + b_0$ be the error-locator polynomial, and $Q(x) = P(x)E(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ be a polynomial with unknown coefficients. Write down the system of linear equations (involving unknowns a_0, a_1, a_2, a_3, b_0) in the Berlekamp–Welsh method. You need not solve the equations.

The message received is $(c_0', c_1', c_2', c_3', c_4') = (0,3,2,1,0)$. Let $R(x)$ be the function such $R(i) = c_i'$ for $0 \le i < 5$. Let $E(x) = x + b_0$ be the error-locator polynomial, and $Q(x) = P(x)E(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Since $Q(i) = P(i)E(i) = R(i)E(i)$ for $1 \le i < 5$, we have the following equalities $\pmod 5$:

$$\begin{aligned} Q(0) &= 0E(0) \\ Q(1) &= 3E(1) \\ Q(2) &= 2E(2) \\ Q(3) &= 1E(3) \\ Q(4) &= 0E(4) \end{aligned}$$

They lead to the following system of linear equations:

$$\begin{array}{rcrcrcrcrcl} & & & & & & a_0 & & & = & 0 \\ a_3 & + & a_2 & + & a_1 & + & a_0 & - & 3b_0 & = & 3 \\ 8a_3 & + & 4a_2 & + & 2a_1 & + & a_0 & - & 2b_0 & = & 4 \\ 27a_3 & + & 9a_2 & + & 3a_1 & + & a_0 & - & b_0 & = & 3 \\ 64a_3 & + & 16a_2 & + & 4a_1 & + & a_0 & & & = & 0 \end{array}$$

(c) The solution to the equations in part (b) is $b_0 = 0, a_0 = 0, a_1 = 4, a_2 = 4, a_3 = 0$. Show how the recipient can recover the original message (m_0, m_1, m_2).

From the solution, we know

$$\begin{aligned} Q(x) &= a_3x^3 + a_2x^2 + a_1x + a_0 = -x^2 + 4x \\ E(x) &= x + b_0 = x \end{aligned}$$

Since $Q(x) = P(x)E(x)$, the recipient can compute $P(x) = Q(x)/E(x) = -x + 4$ [note that this is the same polynomial $P(x)$ from part (a) used by the sender]. The recipient may deduce the location of the error from $E(x)$ as follows. There is only one error at location e_1, we have $E(x) = (x - e_1) = x$, so $e_1 = 0$ and the error is at position 0. To correct the error we evaluate $P(0) = 4$. Since the other two positions m_1, m_2 of the message are uncorrupted, we recover the original message $(m_0, m_1, m_2) = (4,3,2)$.

4. Po(l)lynomial Pranks Alex and Barb talk to each other via Polly. Knowing her tendency to prank, they use polynomials to ensure they can recover their original messages in case she decides to erase (i.e., replace with a blank) or change some of the packets.

Throughout this question, let x_i and y_i denote the i^{th} x and y values the sender sends, and x_i' and y_i' denote the i^{th} x and y values the receiver receives. We use one-based indexing. Although we only show the solutions for interpolation-encoding Reed-Solomon scheme, similar techniques can be applied to coefficient-encoding codewords as well.

1. Never the one to run out of ideas, Polly adds an integer offset c to the x values of the packets instead, i.e., each packet (x, y) becomes $(x + c, y)$. Will Alex and Barb be able to get their original messages

back without knowing c beforehand? Do they need to modify their scheme to handle this prank? If so, describe the method briefly.

They can get their original message back without having to change anything.

When c is added, each packet (x_i, y_i) becomes $(x_i + c, y_i)$. The y_i values are unaffected, and the ordering of the packets doesn't change, i.e., if $x_i < x_j$, then $x'_i = x_i + c < x_j + c = x'_j$. The receiver can just read the received values as normal. □

2. Realizing what you just showed, Polly adds the integer offset c to the y values of the packets instead, i.e., each packet (x, y) becomes $(x, y + c)$. Can Alex and Barb get their original messages back using their current scheme? If not, propose a modified scheme that will work.

   No, they can't. There are at least a few possible schemes.

   Method 1: Send one additional packet at the end with $y = 0$ to figure out the offset c to subtract back.

   - **Sender:** Interpolate n data points to find $P(x)$. Send $(x_1, P(x_1)), (x_2, P(x_2)), \ldots, (x_n, P(x_n)), (x_{n+1}, 0)$.
   - **Receiver:** Let $c = y'_{n+1}$. Get $y'_1 - c$, $y'_2 - c$, ..., $y'_n - c$ as the original message.

   Method 2: Shift $P(x)$'s degree by one place and let Polly's offset occupy the coefficient of x_0.

   - **Sender:** Interpolate n data points to find $P(x)$. Create a degree-n polynomial $\hat{P}(x) = xP(x)$ to let c occupy the coefficient of x^0. Send $n+1$ packets, $(x_1, \hat{P}(x_1)), (x_2, \hat{P}(x_2)), \ldots, (x_n, \hat{P}(x_n))$, and $(x_{n+1}, \hat{P}(x_{n+1}))$.
   - **Receiver:** Interpolate the points to find $\hat{P}(x) = b_n x^n + \ldots + b_1 x + b_0$. Let $c = b_0$ and recover the original message $(y'_1 - c)(x'_1)^{-1}$, $(y'_2 - c)(x'_2)^{-1}$, ..., $(y'_n - c)(x'_n)^{-1}$.

   Both of the schemes need 1 additional packet = $(n+1)$ minimum packets in total.

   Method 3: (Student's solution from HW Party) Swap x's and y's so the noise perturbs x values instead.

   - **Sender:** Interpolate n data points to find $P(x)$. Send $(P(x_1), x_1), (P(x_2), x_2), \ldots, (P(x_n), x_n)$.
   - **Receiver:** Get the original message $x'_1, x'_2, \ldots, x'_n$.

   This method only needs n packets. It is worth noting that this only works because the offset is constant and doesn't affect the packet ordering. □

3. Polly changes her mind again. Adding a constant offset c is too simple. She now picks a random po(l)lynomial $N(x)$ of degree $\leq d$ and adds it to the y values instead, i.e., each packet (x, y) becomes $(x, y + N(x))$. Note that $N(x)$ can have higher degree than $P(x)$. Suppose Alex and Barb know d, provide a scheme for them to reliably communicate using the minimum number of packets. You only need to give a brief justification.

   This is an extension of the solutions for part 2.

   Method 1: Send $d+1$ additional packets at the end with $y = 0$ to figure out $N(x)$ to subtract back.

   - **Sender:** Interpolate n data points to find $P(x)$.
     Send $(x_1, P(x_1)), (x_2, P(x_2)), \ldots, (x_n, P(x_n)), (x_{n+1}, 0), (x_{n+2}, 0), \ldots, (x_{n+d+1}, 0)$.
   - **Receiver:** Interpolate for $N(x)$ from points $(x'_{n+1}, y'_{n+1}), (x'_{n+2}, y'_{n+2}), \ldots, (x'_{n+d+1}, y'_{n+d+1})$.
     Recover the original message $y'_1 - N(x'_1), y'_2 - N(x'_2), \ldots, y'_n - N(x'_n)$.

   Method 2: Shift $P(x)$'s degree by $d+1$ places.

   - **Sender:** Interpolate n data points to find $P(x)$. Create a degree-$n+d$ polynomial $\hat{P}(x) = x^{d+1}P(x)$ to let $N(x)$ occupy the coefficients of x^d to x^0.
     Send $n+d+1$ packets, $(x_1, \hat{P}(x_1)), (x_2, \hat{P}(x_2)), \ldots, (x_{n+d}, \hat{P}(x_{n+d}))$, and $(x_{n+d+1}, \hat{P}(x_{n+d+1}))$.
   - **Receiver:** Interpolate the points to find $\hat{P}(x) = b_{n+d}x^{n+d} + \ldots + b_1 x + b_0$. Let $N(x) = b_d x^d + \ldots + b_1 x + b_0$ and recover the original message $y_i = (y'_i - N(x'_i))(x'^{d+1}_i)^{-1}$.

Note that Method 3 from part 2 doesn't work because it can give wrong or undefined ordering of the message. For example, let $P(x) = 2x$, $N(x) = -x$, and $n = 3$. Let the original packets from sender before the x-y swap be $(1,2),(2,4),(3,6)$. The packets that arrive Polly will be $(2,1),(4,2),(6,3)$. Polly then delivers $(2,1+N(2)),(4,2+N(4)),(6,3+N(6)) = (2,-1),(4,-2),(6,-3)$, effectively reversing the order of the message. The receiver swaps x and y back, sorts by ascending x values and gets 6, 4, 2 as the message instead of 2, 4, 6.

Next, we show that $n+d+1$ *is indeed the minimum number of packets*. Let

$$P(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x + a_0, \tag{1}$$

$$N(x) = b_dx^d + b_{d-1}x^{d-1} + \ldots + b_1x + b_0. \tag{2}$$

Let $Q(x)$ denote the received polynomial, i.e., $Q(x) = P(x) + N(x)$,

$$Q(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x + a_0 + b_dx^d + b_{d-1}x^{d-1} + \ldots + b_1x + b_0. \tag{3}$$

$Q(x)$ consists of $n+d+1$ unknown variables, $a_0, a_1, \ldots, a_{n-1}$, and $b_0, b_1, \ldots, b_d$. Although we only need to know either $a_0, a_1, \ldots, a_{n-1}$ or $b_0, b_1, \ldots, b_d$, we need at least $n+d+1$ pairs of $(x, Q(x))$ to generate $n+d+1$ equations to solve the system. Hence, the minimum number of packets is $n+d+1$.

Then what about just sending $d+1$ extra packets, constructing $n+d+1$ equations and solving them? There will be cases where the $n+d+1$ packets offer redundant data, leading to degenerated system of equations, and we won't be able to figure out the unique original message. For example, consider $N(x) = -P(x)$, the received packets will be $(x_1, P(x_1)+N(x_1)), \ldots, (x_{n+d+1}, P(x_{n+d+1}) + N(x_{n+d+1})) = (x_1, 0), \ldots, (x_{n+d+1}, 0)$, which are not useful at all. □

4. Extend the scheme in part 3 to account for an additional k_e erasure errors. Does your scheme still use the minimum number of packets? If not, come up with a new scheme that does. You should give a brief justification.

   From part 3, if we use Method 1 , we would need $2k_e$ extra packets, k_e for $N(x)$ and k_e for the actual message. But we know from the lecture we can guard against k_e erasures using just k_e extra packets. Can we send less? Yes, by merging all the information into one polynomial. We can do that for Method 1 with a slight modification (or just a different interpretation of the packets, really.) Method 2 works right away.

   Method 1: (Student's solution from HW Party) Add $d+1$ data points with $y = 0$.

   - **Sender:** Add $d+1$ data points, $(x_{n+1}, 0), \ldots, (x_{n+d+1+k_e}, 0)$. Interpolate to find $P(x)$ of degree up to $n+d$ that passes through all $n+d+1$ data points. Use $P(x)$ to generate and send $n+d+1+k_e$ packets, $(x_1, P(x_1)), (x_2, P(x_2)), \ldots, (x_n, P(x_n)), (x_{n+1}, 0), (x_{n+2}, 0), \ldots, (x_{n+d+1+k_e}, 0)$.
   - **Receiver:** The received packets will be of form $(x'_i, P(x'_i) + N(x'_i))$. Let $R(x) = P(x) + N(x)$. Since $R(x)$ can be of at most degree $n+d$, we can interpolate the received $n+d+1$ packets to find $R(x)$. Then we can generate all missing packets, then use $(x'_{n+1}, y'_{n+1}), \ldots, (x'_{n+d+1}, y'_{n+d+1}) = (x'_{n+1}, N(x'_{n+1})), \ldots, (x'_{n+d+1}, N(x'_{n+d+1}))$ to generate $N(x)$. Finally, recover the original message $y'_1 - N(x'_1), \ldots, y'_n - N(x'_n)$.

   Method 2: Shift $P(x)$'s degree by $d+1$ places.

   - **Sender:** Interpolate n data points to find $P(x)$. Create a degree-$n+d$ polynomial $\hat{P}(x) = x^{d+1}P(x)$ to let $N(x)$ occupy the coefficients of x^d to x^0.
     Send $n+d+1+k_e$ packets, $(x_1, \hat{P}(x_1)), \ldots, (x_{n+d+k_e}, \hat{P}(x_{n+d+k_e}))$, and $(x_{n+d+k_e+1}, \hat{P}(x_{n+d+k_e+1}))$.

- **Receiver:** Interpolate the $n+d+1$ points received to find $\hat{P}(x) = b_{n+d}x^{n+d} + \ldots + b_1 x + b_0$. Let $N(x) = b_d x^d + \ldots + b_1 x + b_0$ and recover the original message $y_i = (y_i' - N(x_i'))(x_i'^{d+1})^{-1}$.

□

5. Modify the scheme in part 4 to account for an additional k_g general errors instead of erasure errors.

   Again, from the lecture, we should need $2k_g$ extra packets to recover from general errors. Fortunately, both methods in part 4 work. We can just send $n+d+1+2k_g$ packets instead of $n+d+1+k_e$ and then run Berlekamp-Welch algorithm to detect the error and recover the right polynomial. □