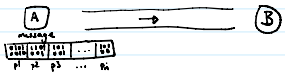
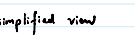
Dis 3D: Error Correcting Codes

Thursday, 9 July 2020 8:22 PM

Setting: Alice wants to send a message to Bob across an *unreliable channel*.
Define a message as group of packets and WLOG mathematize each 'packet' Of information as a number (from bitstring) mod something big.





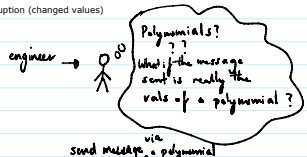
simplified view

2 Types of errors in unreliable channels:

A) Erasure errors (lost values)

a.k.a "General" errors

B) Corruption (changed values)



send message via a polynomial

A → B

using value representation

A) Erasure errors

message has n packets. A sets $m_1 = P(1)$, $m_2 = P(2)$, ..., $m_n = P(n)$

suppose channel has $\le k$ erasure errors

A needs to send $n+k$ pts for B to recover the 'message' polynomial

∵ if property "2 pts determine a line property" $d+1$ deg d

B) Corruption errors

message has n packets.

suppose channel has $\le k$ corruptions

A wants B to reconstruct the 'message' polynomial.

A needs to send $n+2k$ pts for B to recover the 'message' polynomial

How?

say B receives $r_1, r_2, \dots, r_n, r_{n+1}, \dots, r_{n+2k}$

make 'error locator polynomial' $E(x)$ of deg k

$\Rightarrow P(i)E(i) = r_i E(i)$, $1 \le i \le n+2k$

$n+2k$ pts, k corrupted

A encodes message w/ n pkts via a "message polynomial" P of deg $n-1$

$P(1) = m_1, \dots, P(n) = m_n$

$E(x) = (x-a_1)(x-a_2)\cdots(x-a_k)$ — 1)

name Bob's received pkts/pts as $r_1, r_2, \dots, r_{n+2k}$

$E(x) = x^k + b_{k-1}x^{k-1} + b_{k-2}x^{k-2} + \dots + b_1 x + b_0$ (k)

$P(x) = a_n x^n + a_{n-1}x^{n-1} + \dots + a_1 x + a_0$

deg $n+k-1$

$Q(x) = E(x)P(x)$ Q is deg n, k

$Q(i) = E(i)P(i) = E(i)r_i$, $i \in \{1, \dots, n+2k\}$

i) (i, r_i) was not corrupted

$\Rightarrow r_i = P(i)$ eqn holds

$E(i) \neq 0$

ii) (j, r_j) was corrupted

i.e $r_j \neq P(j)$

$E(j) = 0$ by defn of E.L. polynomial

i.e "j" is one of a_i's in eqn 1)

$E(i)P(i) = Q(i) = r_i E(i)$ & $n+2k$ eqns

deg $n+k-1$, deg k

$n+2k$ variables

value rep.

coeff rep

$P(x) = y_0 a_0(x) + y_1 a_1(x) + y_2 a_2(x)$

(mod 5)

$E(x) = x - b_0$

$Q = P E(x)$, deg $2+1-1 = 2$

$Q(i) = E(i) r_i$, $1 \le i \le n+2k$

$Q(x) = a_2 x^2 + a_1 x + a_0$

$P(x) = \frac{Q(x)}{E(x)}$

$\Rightarrow P(0), P(1), P(2)$

m_0 m_1 m_2

1 Berlekamp-Welch Algorithm

In this question we will send the message $(m_0, m_1, m_2) = (1, 1, 4)$ of length $n = 3$. We will use an error-correcting code for $k = 1$ general errors, doing arithmetic over GF(5).

(a) Construct a polynomial $P(x) \pmod 5$ of degree at most 2 so that

$P(0) = 1$, $P(1) = 1$, $P(2) = 4$.

What is the message $(c_0, c_1, c_2, c_3, c_4)$ that is sent?

(b) Suppose the message is corrupted by changing c_0 to 0. Set up the system of linear equations in the Berlekamp-Welch algorithm to find $Q(x)$ and $E(x)$.

$(0, 1, a, 0, a^4)$

(c) Assume that after solving the equations in part (b) we get $Q(x) = 4x^3 + x^2 + x$ and $E(x) = x$. Show how to recover the original message from Q and E.

2 Secret Veto

In the usual secret-sharing scenario we consider (for instance) a secret vault at the United Nations, which we want to design with the property that any k representatives can pool their information and open it, but any smaller number has no hope of doing so. Assume that the solution in the notes has been implemented, so that the key is some number s, and each member has been assigned a number $f(i) \mod q$ for some degree $k-1$ polynomial f with coefficients in GF(q) and satisfying $f(0) = s$.

(a) A group of $k + \ell$ representatives get together to discuss opening the vault. What will happen if ℓ representatives are opposed to opening the vault and, instead of revealing their true numbers, secretly reveal some *different* numbers from GF(q)? Will the group be able to open the vault? If so, how long will it take?

(b) Repeat part (a) in the event that only $\ell/2$ of the ℓ representatives in opposition reveal different numbers than they were assigned—assume that ℓ is even.

3 Berlekamp-Welch Algorithm with Fewer Errors

In class we derived how the Berlekamp-Welch algorithm can be used to correct k general errors, given $n+2k$ points transmitted. In real life, it is usually difficult to determine the number of errors that will occur. What if we have less than k errors? This is a follow up to the exercise posed in the notes.

Suppose Alice wants to send a message to Bob and wants to guard against 1 general error. She decides to encode the message with $P(x) = 4x$ (on GF(7)) such that $P(0) = 0$ is the message she want to send. She then sends $P(0), P(1), P(2) = (0, 4, 1)$ to Bob.

(a) Suppose Bob receives the message $(4, 0, 4)$. Without performing Gaussian elimination explicitly, find $E(x)$ and $Q(x)$.

(b) Now, suppose there were no general errors and Bob receives the original message $(4, 4, 4)$. Show that the $Q(x), E(x)$ that you found in part (a) still satisfies $Q(i) = r_i E(i)$ for all $i = 0, 1, 2$.

(c) Verify that $E(x) = x$, $Q(x) = 4x$ is another possible set of polynomials that satisfies $Q(i) = r_i E(i)$ for all $i = 0, 1, 2$.

(d) Suppose you're actually trying to decode the received message $(4, 4, 4)$. Based on what you showed in the previous two parts, what will happen during row reduction when you try to solve for the unknowns?

(e) Prove that in general, no matter what the solution of $Q(x)$ and $E(x)$ are though, the recovered $P(x)$ will always be the same.

A more interesting question is this: how do we know that the $n+2k$ equations are *independent*, i.e., how do we know that there aren't other spurious solutions in addition to the real solution that we are looking for? Put more mathematically, suppose that the solution we construct is $Q'(x), E'(x)$; how do we know that this solution satisfies the property that $E'(x)$ divides $Q'(x)$ and that $\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)} = P(x)$?

To see that this is true, we note first that, based on our method for calculating $Q'(x), E'(x)$, we know that $Q'(i) = r_i E'(i)$ for $1 \le i \le n+2k$; and of course we also have, by definition, $Q(i) = r_i E(i)$ for the same values of i. Multiplying the first of these equations by $E(i)$ and the second by $E'(i)$, we get

$$Q'(i)E(i) = Q(i)E'(i) \qquad \text{for } 1 \le i \le n+2k, \tag{3}$$

since both sides are equal to $r_i E(i)E'(i)$. Equation (3) tells us that the two polynomials $Q(x)E'(x)$ and $Q'(x)E(x)$ are equal at $n+2k$ points. But these two polynomials both have degree $n+2k-1$, so they are completely determined by their values at $n+2k$ points. Therefore, since they agree at $n+2k$ points, they must be the same polynomial, i.e., $Q(x)E'(x) = Q'(x)E(x)$ for all[4] x. Now we may divide through by the polynomial $E(x)E'(x)$ (which by construction is not the zero polynomial) to obtain $\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)} = P(x)$, which is what we wanted. Hence we can be sure that any solution we find is correct.

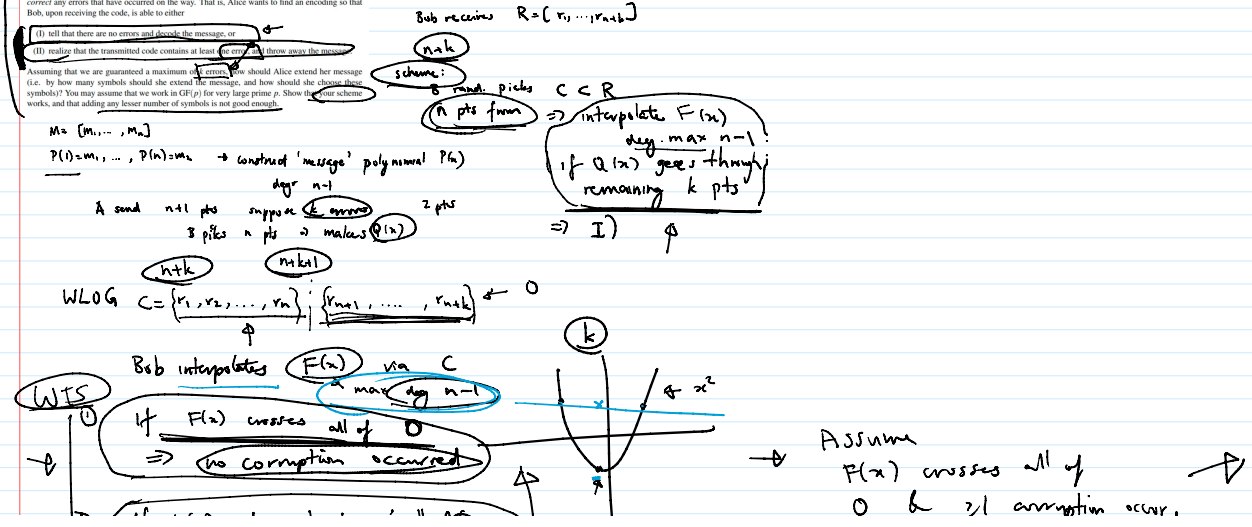
4 Error-Detecting Codes

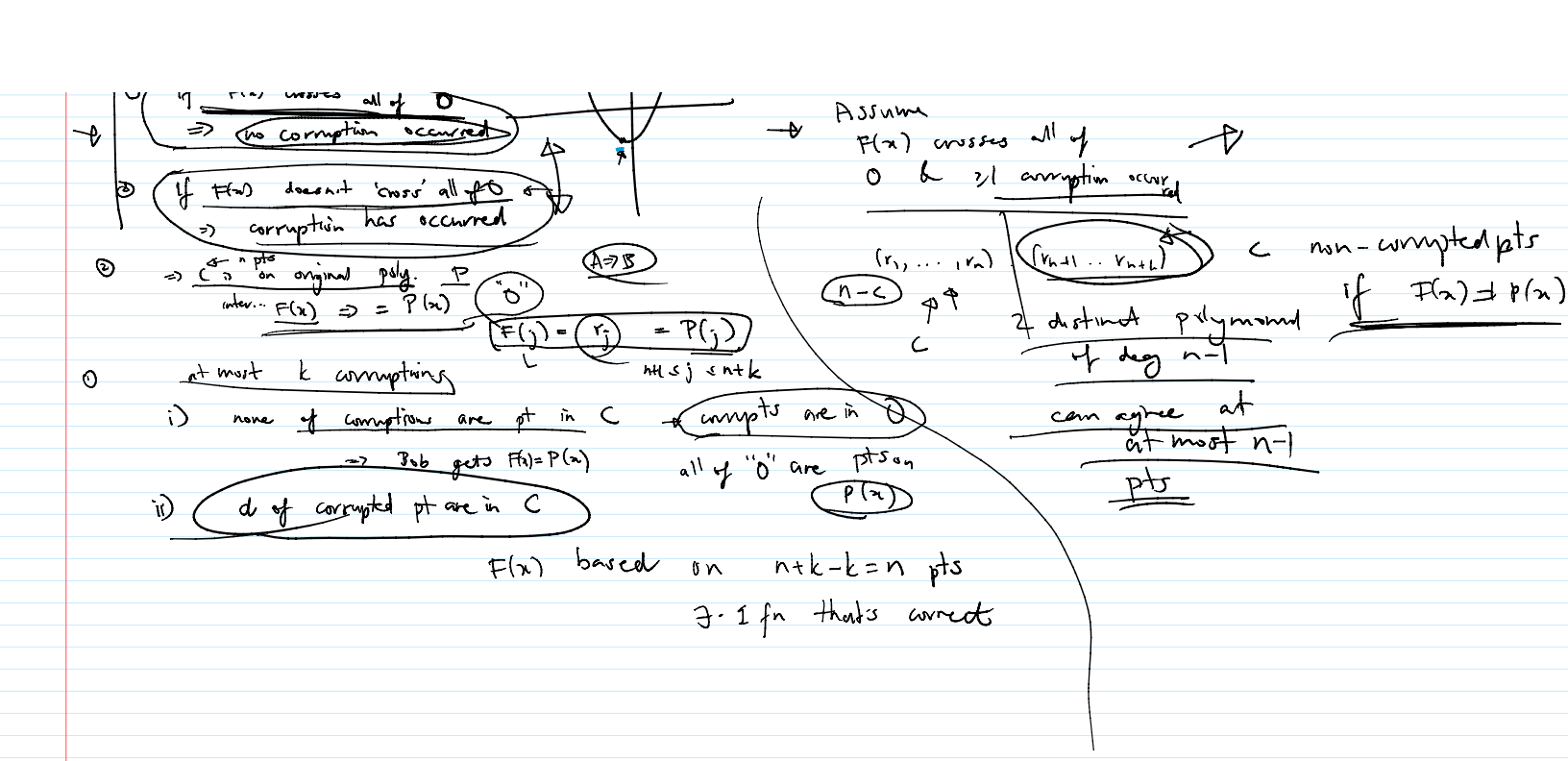
Suppose Alice wants to transmit a message of n symbols, so that Bob is able to *detect* rather than *correct* any errors that have occurred on the way. That is, Alice wants to find an encoding so that Bob, upon receiving the code, is able to either

(I) tell that there are no errors and decode the message, or

(II) realize that the transmitted code contains at least one error and throw away the message.

Assuming that we are guaranteed a maximum of k errors, how should Alice extend her message (i.e. by how many symbols should she extend the message, and how should she choose these symbols)? You may assume that we work in GF(p) for very large prime p. Show that your scheme works, and that adding any lesser number of symbols is not good enough.




if F(x) crosses all of O
⇒ no corruption occurred
If F(x) doesn't 'cross' all of O
⇒ corruption has occurred
A ⇒ B
⇒ C "n pts" on original poly. P
interp. F(x) ⇒ = P(x)
"O"
F(j) = r_j = P(j)
1 ≤ j ≤ n+k
at most k corruptions
i) none of corruptions are pt in C
⇒ Bob gets F(x) = P(x)
ii) d of corrupted pt are in C
corrupts are in O
all of "O" are pts on P(x)
Assume
P(x) crosses all of
O & ≥1 corruption occurred
(r_1, ... r_n)
(r_{n+1} .. r_{n+k})
C
non-corrupted pts
if F(x) ≠ P(x)
n-c
pts
C
2 distinct polynomial of deg n-1
can agree at at most n-1 pts
F(x) based on n+k-k=n pts
∃! 1 fn that's correct