In this lecture we will prove a weaker version of the PCP theorem, namely that $\mathbf{NP} \subseteq \mathbf{PCP}(poly(n), q)$ for some universal constant $q > 0$. The treatment here follows closely the exposition in [AB09], Sections 18.4 and 19.3.2. Before delivering the proof, we give a crash course in some relevant coding theory.

23.1 Communication with noise, and error-correcting codes

Imagine that Alice wants to send a message to Bob over a noisy channel. That is, her message is some string $M \in \{0,1\}^k$, and the medium by which she sends the message (e-mail over a sequence of network links, postal mail, carrier pigeon, etc.) can part of what she sends to Bob (for example, a poor Internet link dropping packets). Alice wants to send a "robust" version M' of her message M such that, even after corruption by the channel, Bob can recover M *exactly* only having access to a corrupted version of M'. M' is typically called an *encoding* of message M.

There are two main classes of noisy channels studied, due to Shannon [Sha48] and Hamming [Ham50]. In the Shannon class of channels, it is assumed that the channel corrupts transmissions probabilistically. For example, if some encoded message, or *codeword*, $C \in \{0,1\}^n$ is sent via the channel, the "Binary Symmetric Channel" BSC_p flips each bit of C independently with probability p (and sends the bit faithfully with probability $1-p$). In the Hamming world, however, the channel corrupts transmissions adversarially. That is to say, whereas a channel such as BSC_p corrupts pn bits in expectation such that bits are corrupted independently at random, in the Hamming world the channel is *adversarial* and is given a budget pn. The adversarial channel is then allowed to choose *any* subset of at most pn bits and flip them.

Some jargon: In this lecture, what will be relevant for a proof of the weak PCP theorem is the Hamming view. For us, a *code* is simply a subset $\mathcal{C} \subseteq [q]^n$, where $[q]$ denotes the set $\{0,\ldots,q-1\}$. The value q is called the *alphabet size*, and n is referred to as the *block length*. In the above introduction we discussed codewords with $q = 2$, but in the general case q may be larger (and we allow the adversarial channel to corrupt some fraction of the codeword symbols arbitrarily). For two codewords $c, c' \in \mathcal{C}$, we let $\Delta(c,c') = |\{i \in [n] : c_i \neq c'_i\}|/n$ denote the *relative Hamming distance* of c, c', and $\delta = \delta(\mathcal{C}) = \min_{c \neq c' \in \mathcal{C}} \Delta(c,c')$ denotes the *relative distance* of the code $\mathcal{C}$. Note that if a codeword c is transmitted and corrupted to some new string $\tilde{c} \in [q]^n$, then as long as $\Delta(c,\tilde{c}) < \delta/2$, then the receiver has enough knowledge to recover c uniquely. This is because if $\Delta(\tilde{c},c') < \delta/2$ for another $c' \neq c \in \mathcal{C}$ then $\Delta(c,c') \leq \Delta(c,\tilde{c}) + \Delta(\tilde{c},c') < \delta$ by the triangle inequality, which contradicts the definition of δ. Lastly, if we

eliminate some elements of $\mathcal{C}$ so that it has exactly q^k elements for some integer k, then we can injectively encode messages in $[q]^k$ to distinct codewords in $\mathcal{C}$. Then k/n is called the *rate* of the code, and gives a measure of the level of information-theoretic redundancy due to sending message encodings rather than raw messages (rate near 1 means almost no redundancy, while low rate means high redundancy).

We will specifically use the *Walsh-Hadamard* code in this lecture. Given a message $x \in \{0,1\}^k$, the Walsh-Hadamard encoding $\mathbf{WH}(x) \in \{0,1\}^{2^k}$ works as follows. We view x as indexing a linear function

$$f_x : \{0,1\}^k \to \{0,1\}^k, \text{ with } f_x(y) = \langle x, y \rangle \tag{23.1}$$

where the inner product is taken modulo 2. That is, $f_x(y) = (\sum_{i=1}^k x_i y_i) \mod 2$. We then define $\mathbf{WH}(x) = (f_x(y))_{y \in \{0,1\}^k}$ to be the truth table of the function f_x. Note the rate of this code is *terrible*, but the relative distance is the best one could hope for from a binary code. The proof of the following is left as an exercise for the reader.

Theorem 23.1 *For any integer $k \geq 1$, the Walsh-Hadamard code has rate $k/2^k$ and relative distance exactly $1/2$ (and hence allows for correcting any fraction of errors strictly less than $1/4$).*

Note the decoding algorithm is simple, and runs in exponential time: given a (potentially corrupted) codeword $\tilde{c}$, the algorithm loops over all $c \in \mathcal{C}$ and returns the first one such that $\Delta(c, \tilde{c}) < 1/4$.

Linearity testing and self-correctability. In proving the weak PCP theorem, we will use more about the Walsh-Hadamard code than it simply being a code. One of these additional properties is *self-correctability*. We now state a definition and some lemmas.

Definition 23.2 *We say two functions $f, g : \{0,1\}^k \to \{0,1\}$ are* ρ-close *if*

$$\mathbb{P}(f(x) = g(x)) \geq \rho,$$

where x is drawn from the uniform distribution over $\{0,1\}^k$. For a class of functions $\mathcal{F}$, we say f is ρ-close to $\mathcal{F}$ if there exists $g \in \mathcal{F}$ such that f is ρ-close to g.

We will be interested in the case that $\mathcal{F}$ is the class of linear functions mapping $\{0,1\}^k$ to $\{0,1\}$, i.e. $\mathcal{F} = \{f : \{0,1\}^k \to \{0,1\} : \exists u \in \{0,1\}^k \text{ s.t. } f(x) = \langle u, x \rangle\}$. Note the elements of this $\mathcal{F}$ are in 1-to-1 correspondence with Walsh-Hadamard codewords. If f is ρ-close to this particular class $\mathcal{F}$, we say that f is ρ-*close to a linear function*.

We now state a theorem about a very natural algorithm to test whether or not a function is close to linear, called the BLR linearity test, named after the authors who first suggested and analyzed the test (Blum, Luby, and Rubinfeld) [BLR93]. The BLR test works as follows: pick $x, y \in \{0,1\}^k$ independently and uniformly at random, then test whether $f(x) + f(y) = f(x+y)$ (where addition of vectors is component-wise mod 2 addition). The following provides a correctness guarantee for this probabilistic test.

Theorem 23.3 *If $\mathbb{P}_{x,y}(f(x)+f(y) = f(x+y)) \geq \rho$ for some $\rho \geq 1/2$, then f is ρ-close to a linear function.*

Recall that Walsh-Hadamard codewords are simply truth tables of linear functions. Also, any transmitted $\tilde{c} \in \{0,1\}^{2^k}$ can be viewed as a function $\tilde{f} : \{0,1\}^k \to \{0,1\}$ where $\tilde{f}(x) = \tilde{c}_x$. Thus if $\tilde{c}$ was obtained by altering an ε-fraction of entries of a Walsh-Hadamard codeword, that is equivalent to $\tilde{c}$ representing a function $\tilde{f}$ that is $(1-\varepsilon)$-close to linear. The following *self-correctability* statement is then a corollary of Theorem 23.3, and implies that such corrupted transmissions $\tilde{c}$, for small ε, can be very efficiently corrected — in constant time! That is, if $\tilde{c}$ is a corruption of a Walsh-Hadamard codeword c for small ε, then there is a randomized algorithm which, for any index $x \in \{0,1\}^k$, one can recover c_x with good probability by only looking at a constant number of entries of $\tilde{c}$. One does this by computing $\tilde{c}_y + \tilde{c}_{x+y}$ for a uniformly random $y \in \{0,1\}^k$.

Corollary 23.4 *Suppose $\tilde{f}$ is $(1-\varepsilon)$-close to some linear function f. Then for any $x \in \{0,1\}^k$, $\mathbb{P}_y(f(x) = \tilde{f}(y) + \tilde{f}(x+y)) \geq 1 - 2\varepsilon$.*

Proof: The values $y, x+y$ are uniformly random in $\{0,1\}^k$ (although they are not independent). Thus $\mathbb{P}(\tilde{f}(y) \neq f(y)), \mathbb{P}(\tilde{f}(x+y) \neq f(x+y)) \leq \varepsilon$. Thus by a union bound, the probability that either differs from the corresponding f values is at most 2ε, i.e. the probability that neither differs is at least $1-2\varepsilon$. But by linearity, $f(y)+f(x+y) = f(y+x+y) = f(x)$ (since component-wise addition is taken modulo 2). ■

23.1.1 Proof of Theorem 23.3

In this subsection we prove Theorem 23.3. You are not responsible for the material in this subsection for this course, and hence you may feel free to skip reading this subsection and go straight to Section 23.2 if you wish. Regardless, the proof of Theorem 23.3 is included here for the interested reader.

For the analysis of the BLR test, it is convenient to work over $\{-1,1\}^k$ instead of $\{0,1\}^k$, by interpreting -1 as 1 and 1 as 0 (i.e. a bit $b \in \{0,1\}$ is mapped to $(-1)^b$). Now if we have a function $f : \{-1,1\}^k \to \mathbf{R}$, we can write f as a 2^k-dimensional vector, which is the evaluation table of f. A typical way to write vectors is in the standard

basis: $f = \sum_{x\in\{-1,1\}^k} f(x)\cdot \mathbf{e}_x$, where $\mathbf{e}_x$ is the standard basis vector corresponding to x (or, as a function, it is the function satisfying $\mathbf{e}_x(z) = 0$ if $z \neq x$, and $\mathbf{e}_x(x) = 1$). We can also work over do other bases. Define an inner product $\langle . \rangle$ on functions defined by $\langle f, g\rangle = \mathbb{E}_{x\in\{-1,1\}^k} f(x)g(x)$. Consider the set of linear functions χ_S for $S \subseteq [n]$, defined by $\chi_S(x) = \prod_{i\in S} x_i$. One can check as an exercise that the following lemma holds, i.e. the set of linear functions $\{\chi_S\}$ forms an orthonormal basis for functions mapping $\{-1,1\}^k$ to $\mathbf{R}$ (or equivalently for the set of all vectors in $\mathbf{R}^{2^k}$).

Lemma 23.5 *For all $S \neq T \subseteq [n]$, $\langle \chi_S, \chi_S\rangle = 1$ and $\langle \chi_S, \chi_T\rangle = 0$.*

The set $\{\chi_S\}_{S\subseteq[n]}$ is known as the *Fourier basis* over the boolean hypercube $\{-1,1\}^k$. As it is an orthonormal basis, we can write any function $f : \{-1,1\}^k \to \mathbf{R}$ as

$$f = \sum_{S\subseteq[n]} \langle f, \chi_S\rangle \chi_S.$$

The coefficients $\langle f, \chi_S\rangle$ are known as the *Fourier coefficients* of f, and $\langle f, \chi_S\rangle$ is usually denoted $\hat{f}_S$. The following corollary is then immediate from definitions and Lemma 23.5.

Corollary 23.6 . *For any $f, g : \{-1,1\}^k \to \mathbf{R}$, $\langle f, g\rangle = \sum_{S\subseteq[n]} \hat{f}_S\hat{g}_S$, and $\langle f, f\rangle = \sum_{S\subseteq[n]} \hat{f}_S^2$. The latter is known as Parseval's identity.*

The key insight is then that $\langle f, g\rangle$ equals $\varepsilon > 0$ is equivalent to the statement that f, g agree on an a $(1/2+\varepsilon/2)$-fraction of vectors in $\{-1,1\}^k$. Thus Theorem 23.3 is equivalent to the statement that

$$f : \{-1,1\}^n \to \{-1,1\} \text{ satisfies } \mathop{\mathbb{P}}_{x,y\in\{-1,1\}^k}(f(xy) = f(x)f(y)) \geq 1/2+\varepsilon \Longrightarrow \exists S \subseteq [n], \hat{f}_S \geq 2\varepsilon. \tag{23.2}$$

We now prove (23.2).

Note $\mathbb{E} f(xy)f(x)f(y) = \mathbb{P}(f(xy) = f(x)f(y)) - \mathbb{P}(f(xy) \neq f(x)f(y)) \geq 2\varepsilon$. Thus

$$\begin{aligned}
2\varepsilon \leq \mathop{\mathbb{E}}_{x,y} f(xy)f(x)f(y) &= \mathop{\mathbb{E}}_{x,y}(\sum_S \hat{f}_S\chi_S(xy))(\sum_T \hat{f}_T\chi_T(x))(\sum_R \hat{f}_R\chi_R(y)) \\
&= \mathop{\mathbb{E}}_{x,y} \sum_{S,T,R} \hat{f}_S\hat{f}_T\hat{f}_R\chi_S(x)\chi_S(y)\chi_T(x)\chi_R(y) \\
&= \sum_{S,T,R} \hat{f}_S\hat{f}_T\hat{f}_R(\mathop{\mathbb{E}}_x \chi_S(x)\chi_T(x))(\mathop{\mathbb{E}}_y \chi_S(y)\chi_R(y)) \\
&= \sum_S \hat{f}_S^3 \text{ (unless } S = T = R\text{, expectation is 0)} \\
&\leq (\max_S \hat{f}_S)\cdot(\sum_S \hat{f}_S^2) \\
&= \max_S \hat{f}_S.
\end{aligned}$$

The last line uses Parseval's identity, since $\sum_S \hat{f}_S^2 = \langle f, f\rangle = 1$, since the range of f is $\{-1,1\}$.

23.2 A weak PCP theorem

We will provide the proof system for a particular **NP**-complete problem: $\mathsf{QUADEQ} = \{\langle A,b\rangle : \exists U \in \{0,1\}^{n^2}$ with $U_{i,j} = u_i u_j$ for some $u \in \{0,1\}^n$ s.t. $AU = b\}$. Here $\langle A,b\rangle$ is an encoding of an $m \times n^2$ binary matrix A and m-dimensional binary vector b, and the language consists of all such A,b pairs such that the quadratic equation $AU = b$ has a solution. That is, we have a system of m equations each of the for $\sum_{i,j} A^{(r)}_{i,j} u_i u_j = b_r$ for $r = 1,\ldots,m$.

Lemma 23.7 QUADEQ *is* **NP***-complete.*

Proof: It is in **NP** since a witness that can be checked in polynomial-time is a solution $u \in \{0,1\}^n$ to $AU = b$ (recall $U_{i,j} = u_i u_j$). To show that it is **NP**-hard, we reduce from 3SAT. Suppose we have a 3SAT instance φ with m clauses whose rth clause C_r is, say, $(x_{i_1} \vee x_{i_2} \vee \bar{x}_{i_3})$. From this we derive a quadratic equation $a_r x_{i_1} + b_r x_{i_2} + c_r(1 - x_{i_3}) = 1$, and in this way we create m equations from the m clauses. We claim this system of equations $AX = b$ has a solution x iff φ is satisfiable. If $AX = b$ has a solution, then the same vector x itself is a solution to φ. For the converse, if φ has a solution x^*, we use the same x^* to solve $AX = b$. In addition, for the rth clause for each r, we set a_r, b_r, c_r so that exactly one of the three products in the sum is 1 and the other two are zero; this ensures that two 1's do not sum to a 0 and yield 0 modulo 2. For example, if the rth clause is $x_1 \vee \bar{x}_2 \vee x_4$ and a satisfying assignment x^* has $x^*_1 = 1, x^*_2 = 0, _4 = 0$, then this clause is satisfied by both x_1 and x_2. We then pick one of these literals that satisfies the clause arbitrarily, say x_1, and set the corresponding a_r to 1. We then set $b_r = c_r = 0$ (we could have also set $b_r = 1$ and $a_r = c_r = 0$). ■

Corollary 23.8 *If there is a proof system for* QUADEQ *in which the verifier flips* $poly(mn)$ *bits and checks* $q = O(1)$ *bits of the proof to have completeness* 1 *and soundness* $1/2$*, then* $\mathbf{NP} \subseteq \mathbf{PCP}(poly(n), q)$.

Proof: Let L be any language in **NP**, and let f be a poly-time computable reduction from L to QUADEQ. Then a proof system for L is as follows: a verifier for L first computes $f(x)$, then runs the verifier for QUADEQ on $f(x)$. ■

Corollary 23.8 tells us that in order to show $\mathbf{NP} \subseteq \mathbf{PCP}(poly(n), q)$ for some constant q, it suffices to specify such a proof system for QUADEQ. We do so now.

The verifier V for QUADEQ is given A,b and expects a proof of the form $(c,c') = (\mathbf{WH}(u), \mathbf{WH}(U))$ for a solution $u \in \{0,1\}^n$ to some system of quadratic equations $AU = b$ (recall $U \in \{0,1\}^{n^2}$ is defined to have $U_{i,j} = u_i u_j$). Thus $c \in \{0,1\}^{2^n}$ and $c' \in \{0,1\}^{2^{n^2}}$. Given such a proof, V works in three phases as follows.

1. First verify that c, c' are both .999-close to being Walsh-Hadamard codewords (that is, make sure that the functions $\tilde{f}, \tilde{g}$ that c, c' represent, respectively, are both .999-close to linear functions f, g). If not, the verifier rejects.

2. Then, verify that g is faithful to the format of U, i.e. that g is (close to) the function $f_W(z) = \langle W, z \rangle$ for $W_{i,j} = w_i w_j$, for w being such that $f = f_w$ as defined in Equation 23.1. If not, the verifier rejects.

3. Finally, verify that the W represented by g is a solution to $AU = b$ (i.e. verify that $AW = b$) and reject if not.

For all the phases (1)-(3) above, a verifier that can only read a constant number of entries in c, c' cannot perform the verification precisely, but will do so using random probes such that the completeness is 1 and soundness is $1/2$.

We now show how to implement the phases.

Phase 1. Given the BLR linearity test, this is the easiest phase to implement. If $\tilde{f}$ is not .999-linear, then the BLR test will fail with probability at least $1/1000$. Thus if we repeat the BLR test $1000 \cdot \lceil \ln(2/\eta) \rceil$ times, the probability that the BLR test never outputs "fail" is at most $(1 - 1/1000)^{1000 \ln(2/\eta)} \leq \eta/2$ (using that $(1 - 1/x)^x \leq 1/e$ for any $x > 0$). The same holds true for testing $\tilde{g}$. Thus by a union bound, if either of $\tilde{f}, \tilde{g}$ are not .999-linear, the BLR test will output "fail" with probability at least $1 - \eta$. If η is a fixed constant, then the number of queries to c, c' is a fixed constant, and we can set η to be as small a constant as we like (we will set $\eta = 1/8$).

Phase 2. For phases 2 and 3, we will assume that we have query access to f and g. This can be implemented using self-correctability (Corollary 23.4). Since $\tilde{f}, \tilde{g}$ are .999-close to linear, the probability that any query to f or g fails is at most $.002 = 1/500$. Thus as long as our total number of queries to f, g is at most $\lfloor 500/8 \rfloor = 62$ in these two phases, which we will ensure, the probability that all our queries to f, g succeed is at least $7/8$ (by a union bound).

Now for phase 2: since f is linear, we have $f(x) = f_u(x)$ for some $u \in \{0,1\}^n$ and $g(x) = g_w(x)$ for some $w \in \{0,1\}^{n^2}$ (for f_u, g_w as defined in Equation 23.1). We now wish to make sure that $w_{i,j} = u_i u_j$ for all i, j (viewing the pair (i, j) as an index into $\{1, \ldots, n^2\}$). That is, if W is defined to be the matrix whose (i, j) entry is $w_{i,j}$, then we wish to verify that $W = U$. The verifier will do the following t times for some t to be specified: pick $r, r' \in \{0,1\}^n$ independently and uniformly at random, and reject if for any of these t trials it holds that $f(r)f(r') \neq g(rr'^T)$, treating rr'^T as an n^2-dimensional vector (note that rr^T is the $n \times n$ matrix whose (i, j) entry is $r_i r'_j$).

Lemma 23.9 *If $W \neq U$, then $\mathbb{P}_{r,r'}(f(r)f(r') \neq g(rr'^T)) \geq 1/4$.*

Proof: We have $f(r)f(r') = (\sum_i u_i r_i) \cdot (\sum_i u_i r'_i) = r^T U r'$. We also have $g(rr'^T) = \sum_{i,j} w_{i,j} r_i r'_j = r^T W r'$. Thus $f(r)f(r') = g(rr'^T)$ iff $r^T P r' = 0$ for $P = W - U$, which we show happens with probability at most $3/4$ if $P \neq 0$. If $P \neq 0$ then $\mathbb{P}_{r'}(Pr' \neq 0) \geq 1/2$ — we already proved this in Lecture 18 in our analysis of Freivalds' algorithm (see Lemma 18.7 in the Lecture 18 notes)! Then conditioned on $v = Pr' \neq 0$, $\mathbb{P}_r(r^T v) \neq 0$ is exactly $1/2$. This is because if $v \neq 0$ then $v_{i^*} = 1$ for some i^*. But then the probability $r^T v$ is zero is exactly $1/2$ over just the randomness of r_{i^*}, even if we condition on all the other r_i values for $i \neq i^*$. Thus $\mathbb{P}_{r,r'}(r^T P r' = 0) \geq (1/2)^2 = 1/4$. ■

We thus set $t = 7$ so that we make $7 \times 3 = 21$ queries to f, g combined in phase 2. Then if $W \neq U$, our test fails to declare so with probability at most $(1 - 1/4)^7 < 1/8$.

Phase 3. Having used 21 queries to f, g in phase 2, we can afford to make at most $62 - 21 = 41$ queries in phase 3 (as mentioned in the beginning of the phase 2 discussion); we will fit well within this budget (we will only make 3 queries on phase 3!).

Recall we want to make sure that $AU = b$. That is, we want to make sure that for every $k \in \{1, \ldots, m\}$

$$\sum_{i,j} A^{(k)}_{i,j} u_i u_j = b_k$$

for A, b known to the verifier. $(A^{(k)}_{i,j})_{(i,j) \in [n] \times [n]}$ is just some n^2-dimensional vector a_k, so this just amounts to checking a single query $g(a_k)$ to verify that it comes out to b_k. However, we cannot afford to check all a_k, since there are m of them and we only want to make a constant number of queries. Instead, we pick $\beta_1, \ldots, \beta_m$ independent and uniformly at random in $\{0, 1\}$ and query $g(\sum_{k=1}^m \beta_k a_k)$, and verify that it equals $\sum_k \beta_k b_k$. If all equations are satisfied, this test succeeds with probability 1. If even one equation, say equation k^*, is unsatisfied, then this test fails with probability exactly $1/2$. This is because, even conditioned on all the β_k for $k \neq k^*$, the probability that this test fails just over the randomness of β_{k^*} is exactly $1/2$ (**exercise:** justify this to yourself!). Thus U does not satisfy $AU = b$, the probability that we do not output "fail" after three independent tests is $1/2^3 = 1/8$.

Overall, we have perfect completeness in all three phases. For soundness, $\tilde{f}, \tilde{g}$ are not each .999-linear, then we output fail with probability at least $7/8$ in phase 1. Since we only make $21 + 3 = 24 < 62$ queries to f, g in the rest of the algorithm, the probability we succeed in all these queries is also at least $7/8$ by the discussion at the beginning of the phase 2 analysis. Phase 2 then succeeds with probability at least $7/8$, and phase 3 also with probability at least $7/8$. Thus by a union bound, the probability that we do not output "fail" is at most $1/8 + 1/8 + 1/8 + 1/8 = 1/2$.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[Ham50] Richard W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.