Chapter 20

More PCP Theorems and the Fourier Transform Technique

The PCP Theorem has several direct applications in complexity theory, in particular showing that unless $P = NP$, many $NP$ optimization problems cannot be approximated in polynomial-time to within arbitrary precision. However, for some applications, the standard PCP Theorem does not suffice, and we need stronger (or simply different) “PCP Theorems”. In this chapter we survey some of these results and their proofs. The Fourier transform technique turned out to be especially useful in advanced PCP constructions, and in other areas in theoretical computer science. We describe the technique and show two of its applications. First, we use Fourier transforms to prove the correctness of the linearity testing algorithm of Section 19.4, completing the proof of the PCP Theorem. We then use it to prove a stronger PCP Theorem due to Håstad, showing tight inapproximability results for many important problems, including MAX 3SAT.

20.1 Parallel Repetition of PCP’s

Recall that the soundness parameter of a PCP system is the probability that the verifier may accept a false statement. Definition 19.1 specified the soundness parameter to be $1/2$, but as we noted, it can be reduced to an arbitrary small constant by increasing the number of queries. Yet for some applications we need a system with, say, three queries, but an arbitrarily small constant soundness parameter. Raz has shown that this
can be achieved if we consider systems with non binary alphabet. (For a finite set $S$, we say that a PCP verifier uses alphabet $S$ if it takes as input a proof string $\pi$ in $S^*$.) The idea is simple and natural: use parallel repetition.

That is, we take a PCP verifier $V$ and run $\ell$ independent copies of it, to obtain a new verifier $V^\ell$ such that a query of $V^\ell$ is the concatenation of the $\ell$ queries of $V$, and an answer is a concatenation of the $\ell$ answers. (So, if the original verifier $V$ used proofs over, say, the binary alphabet, then the verifier $V^\ell$ will use the alphabet $\{0,1\}^\ell$.) The verifier $V^\ell$ accepts the proof only if all the $\ell$ executions of $V$ accept. Formally, we define parallel repetition as follows:

**Definition 20.1 (Parallel repetition)**

Let $S$ be a finite set. Let $V$ be a PCP verifier using alphabet $S$ and let $\ell \in \mathbb{N}$. The $\ell$-times parallel repeated $V$ is the verifier $V^\ell$ that operates as follows:

1. $V^\ell$ uses the alphabet $\hat{S} = S^\ell$. We denote the input proof string to $V^\ell$ by $\hat{\pi}$.

2. Let $q$ denote the number of queries $V$ makes. On any input $x$, $V^\ell$ chooses $\ell$ independent random tapes $r^1, \ldots, r^\ell$ for $V$, and runs $V$ on the input and these tapes to obtain $\ell$ sets of $q$ queries:

$$
\begin{align*}
&i^1_1, i^1_2, \ldots, i^1_q \\
i^2_1, i^2_2, \ldots, i^2_q \\
&\vdots \\
i^\ell_1, i^\ell_2, \ldots, i^\ell_q
\end{align*}
$$

3. $V^\ell$ makes $q$ queries $i_1, \ldots, i_q$ to $\hat{\pi}$ where $i_j$ is \langle $i^1_j, \ldots, i^\ell_j$ $\rangle$ (under a suitable encoding of $\mathbb{N}^\ell$ into $\mathbb{N}$).

4. For $j \in [q]$, denote $\langle a^1_j, \ldots, a^\ell_j \rangle = \hat{\pi}(i_j)$. The verifier $V^\ell$ accepts if and only for every $k \in [\ell]$, the verifier $V$ on random tape $r_k$ accepts when given the responses $a^1_k, \ldots, a^\ell_k$.

**Remark 20.2**

For every input $x$, if there is a proof $\pi$ such that on input $x$, the verifier $V$ accepts $\pi$ with probability one, then there is a proof $\hat{\pi}$ such that on input $x$, the verifier $V^\ell$ accepts $\hat{\pi}$ with probability one. Namely, for every $\ell$-tuple of positions $i^1, \ldots, i^\ell$, the proof $\hat{\pi}$ contains the tuple $\langle \pi[i^1], \ldots, \pi[i^\ell] \rangle$. Note that $|\hat{\pi}| = |\pi|^\ell$. 
20.1. PARALLEL REPETITION OF PCP’S

<table>
<thead>
<tr>
<th></th>
<th>Original $V$</th>
<th>Parallel repeated $V^\ell$</th>
<th>Sequential repeated $V^{\text{seq}\ell}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabet size</td>
<td>$W$</td>
<td>$W^\ell$</td>
<td>$W$</td>
</tr>
<tr>
<td>Proof size</td>
<td>$m$</td>
<td>$m^\ell$</td>
<td>$m$</td>
</tr>
<tr>
<td>Random coins used</td>
<td>$r$</td>
<td>$\ell r$</td>
<td>$\ell r$</td>
</tr>
<tr>
<td>Number of queries</td>
<td>$q$</td>
<td>$q$</td>
<td>$\ell q$</td>
</tr>
<tr>
<td>Completeness probability</td>
<td>$1$</td>
<td>$1$</td>
<td>$(1 - \delta)^{\ell}$</td>
</tr>
<tr>
<td>soundness parameter</td>
<td>$1 - \delta$</td>
<td>$(1 - \delta^a)^{\ell b}$</td>
<td>$(1 - \delta)^{\ell}$</td>
</tr>
</tbody>
</table>

Table 20.1: Parameters of $\ell$-times parallel repeated verifier $V^\ell$ vs. parameters for sequential repetition.

**Why is it called “parallel repetition”?** We call the verifier $V^\ell$ the parallel repeated version of $V$ to contrast with sequential repetition. If $V$ is a PCP verifier and $\ell \in \mathbb{N}$, we say that $\ell$-times sequentially repeated $V$, denoted $V^{\text{seq}\ell}$, is the verifier that chooses $\ell$ random tapes for $V$, then makes the $q\ell$ queries corresponding to these tapes one after the other, and accepts only if all the instances accept. Note that $V^{\text{seq}\ell}$ uses the same alphabet as $V$, and uses proofs of the same size. The relation between the parameters of $V$, $V^\ell$ and $V^{\text{seq}\ell}$ is described in Table 20.1.

It is a simple exercise to show that if $V$’s soundness parameter was $1 - \delta$ then $V^{\text{seq}\ell}$ soundness parameter will be equal to $(1 - \delta)^{\ell}$. One may expect the soundness parameter of the parallel repeated verifier $V^\ell$ to also be $(1 - \delta)^{\ell}$. It turns out this is not the case (there is a known counterexample [?]), however the soundness parameter does decay exponentially with the number of repetitions:

**Theorem 20.3 (Parallel Repetition Lemma, [Raz98])**
There exist constants $a, b$ (independent of $\ell$ but depending on the alphabet size used and number of queries) such that the soundness parameter of $V^\ell$ is at most $(1 - \delta^a)^{b\ell}$

We omit the proof of Theorem 20.3 for lack of space. Roughly speaking, the reason analyzing soundness of $V^\ell$ is so hard is the following: for every tuple $(i_1, \ldots, i_\ell)$, the corresponding position in the proof for $V^\ell$ is “supposed” to consist of the values $\pi[i_1] \circ \cdots \circ \pi[i_\ell]$ where $\pi$ is some proof for $V$. However, a priori, we do not know if the proof satisfies this property. It may be that the proof is inconsistent and that two tuples containing the $i^{th}$ position “claim” a different assignment for $\pi[i]$.

**Remark 20.4**
The Gap Amplification Lemma (Lemma 19.29) of the previous chapter has a similar flavor, in the sense that it also reduced the soundness parameter...
at the expense of an increase in the alphabet size. However, that lemma assumed that the soundness parameter is very close to 1, and its proof does not seem to generalize for soundness parameters smaller than 1/2. We note that a weaker version of Theorem 20.3, with a somewhat simpler proof, was obtained by Feige and Kilian [29]. This weaker version is sufficient for many applications, including for Håstad’s 3-query PCP theorem (see Section 20.2 below).

20.2 Håstad’s 3-bit PCP Theorem

In most cases, the PCP Theorem does not immediately answer the question of exactly how well can we approximate a given optimization problem (even assuming P ≠ NP). For example, the PCP Theorem implies that if P ≠ NP then MAX 3SAT cannot be c-approximated in polynomial-time for some constant ρ < 1. But if one follows closely the proof of Theorem 19.13, this constant ρ turns out to be very close to one, and in particular it is larger than 0.999. On the other hand, as we saw in Example 19.6, there is a known 7/8-approximation algorithm for MAX 3SAT. What is the true “approximation complexity” of this problem? In particular, is there a polynomial-time 0.9-approximation algorithm for it? Similar questions are the motivation behind many stronger PCP theorems. In particular, the following theorem by Håstad implies that for every ε > 0 there is no polynomial-time (7/8 + ε)-approximation for MAX 3SAT unless P = NP:

**Theorem 20.5 (Håstad’s 3-bit PCP [29])**

For every ε > 0 and every language L ∈ NP there is a PCP-verifier V for L making three (binary) queries having completeness probability at least 1 − ε and soundness parameter at most 1/2 + ε.

Moreover, the test used by V are linear. That is, given a proof π ∈ {0, 1}^m, V chooses a triple (i_1, i_2, i_3) ∈ [m]^3 and b ∈ {0, 1} according to some distribution and accepts if \( \pi_{i_1} + \pi_{i_2} + \pi_{i_3} \equiv b \pmod{2} \).

Theorem 20.5 immediately implies that the problem MAX E3LIN is NP-hard to 1/2 + ε-approximate for every ε > 0, where MAX E3LIN is the problem of finding a solution maximizing the number of satisfied equations among a given system of linear equations over GF(2), with each equation containing at most 3 variables. Note that this hardness of approximation result is tight since a random assignment is expected to satisfy half of the equations. Also note that finding out whether there exists a solution satisfying all of the equations can be done in polynomial-time using Gaussian elimination (and hence the imperfect completeness in Theorem 20.5 is inherent).
The result for MAX 3SAT is obtained by the following corollary:

**Corollary 20.6**
For every \( \epsilon > 0 \), computing \((7/8 + \epsilon)\)-approximation to MAX 3SAT is NP-hard.

**Proof:** We reduce MAX E3LIN to MAX 3SAT. Take any instance of MAX E3LIN where we are interested in determining whether \((1 - \epsilon)\) fraction of the equations can be satisfied or at most \(1/2 + \epsilon\) are. Represent each linear constraint by four 3CNF clauses in the obvious way. For example, the linear constraint \(a + b + c = 0 \pmod 2\) is equivalent to the clauses \((\bar{a} \lor b \lor c), (a \lor \bar{b} \lor c), (a \lor b \lor \bar{c}), (\bar{a} \lor \bar{b} \lor \bar{c})\). If \(a, b, c\) satisfy the linear constraint, they satisfy all 4 clauses and otherwise they satisfy at most 3 clauses. We conclude that in one case at least \((1 - \epsilon)\) fraction of clauses are simultaneously satisfiable, and in the other case at most \(1 - (1/2 - \epsilon) \times \frac{1}{4} = \frac{7}{8} - \frac{\epsilon}{4}\) fraction are. The ratio between the two cases tends to \(7/8\) as \(\epsilon\) decreases. Since Theorem 20.5 implies that distinguishing between the two cases is NP-hard for every constant \(\epsilon\), the result follows. ■

### 20.3 Tool: the Fourier transform technique

The continuous Fourier transform is extremely useful in mathematics and engineering. Likewise, the discrete Fourier transform has found many uses in algorithms and complexity, in particular for constructing and analyzing PCP’s. The Fourier transform technique for PCP’s involves calculating the maximum acceptance probability of the verifier using Fourier analysis of the functions presented in the proof string. It is delicate enough to give “tight” inapproximability results for MAX INDSET, MAX 3SAT, and many other problems.

To introduce the technique we start with a simple example: analysis of the linearity test over GF(2) (i.e., proof of Theorem 19.23). We then introduce the Long Code and show how to test for membership in it. These ideas are then used to prove Håstad’s 3-bit PCP Theorem.

#### 20.3.1 Fourier transform over GF(2)^n

The Fourier transform over GF(2)^n is a tool to study functions on the Boolean hypercube. In this chapter, it will be useful to use the set \(\{+1, -1\} = \{\pm 1\}\) instead of \(\{0, 1\}\). To transform \(\{0, 1\}\) to \(\{\pm 1\}\), we use the mapping \(b \mapsto (-1)^b\) (i.e., 0 \(\mapsto\) +1, 1 \(\mapsto\) −1). Thus we write the hypercube as \(\{\pm 1\}^n\)
instead of the more usual \( \{0,1\}^n \). Note this maps the XOR operation (i.e., addition in GF(2)) into the multiplication operation.

The set of functions from \( \{\pm 1\}^n \) to \( \mathbb{R} \) defines a \( 2^n \)-dimensional Hilbert space (see Section ??) as follows. Addition and multiplication by a scalar are defined in the natural way: \((f + g)(x) = f(x) + g(x)\) and \((\alpha f)(x) = \alpha f(x)\) for every \( f, g : \{\pm 1\}^n \to \mathbb{R}, \alpha \in \mathbb{R} \). We define the inner product of two functions \( f, g \), denoted \( \langle f, g \rangle \), to be \( \mathbb{E}_{x \in \{\pm 1\}^n} [f(x)g(x)] \).

The standard basis for this space is the set \( \{e_x\}_{x \in \{\pm 1\}^n} \), where \( e_x(y) \) is equal to \( k \) if \( y = x \), and equal to \( 0 \) otherwise. This is an orthonormal basis, and every function \( f : \{\pm 1\}^n \to \mathbb{R} \) can be represented in this basis as \( f = \sum_x a_x e_x \). For every \( x \in \{\pm 1\}^n \), the coefficient \( a_x \) is equal to \( f(x) \). The Fourier basis for this space is the set \( \{\chi_\alpha\}_{\alpha \subseteq [n]} \) where \( \chi_\alpha(x) = \prod_{i \in \alpha} x_i \) (\( \chi_\emptyset \) is the constant 1 function). These correspond to the linear functions over GF(2). To see this, note that every linear function of the form \( b \mapsto a \odot b \) (with \( a, b \in \{0,1\}^n \)) is mapped by our transformation to the function taking \( x \in \{\pm 1\}^n \) to \( \prod_{i \text{ s.t. } a_i = 1} x_i \).

The Fourier basis is indeed an orthonormal basis for the Hilbert space. Indeed, the random subsum principle implies that for every \( \alpha, \beta \subseteq [n] \), \( \langle \chi_\alpha, \chi_\beta \rangle = \delta_{\alpha, \beta} \) where \( \delta_{\alpha, \beta} \) is equal to 1 iff \( \alpha = \beta \) and equal to 0 otherwise. This means that every function \( f : \{\pm 1\}^n \to \mathbb{R} \) can be represented as \( f = \sum_{\alpha \subseteq [n]} \hat{f}_\alpha \chi_\alpha \). We call \( \hat{f}_\alpha \) the \( \alpha \)-th Fourier coefficient of \( f \).

We will often use the following simple lemma:

**Lemma 20.7**

Every two functions \( f, g : \{\pm 1\}^n \to \mathbb{R} \) satisfy

1. \( \langle f, g \rangle = \sum_\alpha \hat{f}_\alpha \hat{g}_\alpha \).

2. (Parseval’s Identity) \( \langle f, f \rangle = \sum_\alpha \hat{f}_\alpha^2 \).

**Proof:** The second property follows from the first. To prove the first we expand

\[
\langle f, g \rangle = \left( \sum_\alpha \hat{f}_\alpha \chi_\alpha \right) \left( \sum_\beta \hat{g}_\beta \chi_\beta \right) = \sum_{\alpha, \beta} \hat{f}_\alpha \hat{g}_\beta \langle \chi_\alpha, \chi_\beta \rangle = \sum_{\alpha, \beta} \hat{f}_\alpha \hat{g}_\beta \delta_{\alpha, \beta} = \sum_\alpha \hat{f}_\alpha \hat{g}_\alpha
\]
Example 20.8
Some examples for the Fourier transform of particular functions:

1. If \( f(u_1, u_2, \ldots, u_n) = u_i \) (i.e., \( f \) is a coordinate function, a concept we will see again soon) then \( f = \chi_{\{i\}} \) and so \( \hat{f}_{\{i\}} = 1 \) and \( \hat{f}_\alpha = 0 \) for \( \alpha \neq \{i\} \).

2. If \( f \) is a random boolean function on \( n \) bits, then each \( \hat{f}_\alpha \) is a random variable that is a sum of \( 2^n \) binomial variables (equally likely to be 1, -1) and hence looks like a normally distributed variable with standard deviation \( 2^{n/2} \) and mean 0. Thus with high probability, all \( 2^n \) Fourier coefficients have values in \( \left[-\frac{\text{poly}(n)}{2^{n/2}}, \frac{\text{poly}(n)}{2^{n/2}}\right] \).

The connection to PCPs: High level view

In the PCP context we are interested in Boolean-valued functions, i.e., those from \( GF(2)^n \) to \( GF(2) \). Under our transformation these are mapped to functions from \( \{\pm 1\}^n \) to \( \{\pm 1\} \). Thus, we say that : \( f : \{\pm 1\}^n \rightarrow \mathbb{R} \) is Boolean if \( f(x) \in \{\pm 1\} \) for every \( x \in \{\pm 1\}^n \). Note that if \( f \) is Boolean then \( \langle f, f \rangle = \mathbb{E}_x[f(x)^2] = 1 \).

On a high level, we use the Fourier transform in the soundness proofs for PCP’s to show that if the verifier accepts a proof \( \pi \) with high probability then \( \pi \) is “close to” being “well-formed” (where the precise meaning of “close-to” and “well-formed” is context dependent). Technically, we will often be able to relate the acceptance probability of the verifier to an expectation of the form \( \langle f, g \rangle = \mathbb{E}_x[f(x)g(x)] \), where \( f \) and \( g \) are Boolean functions arising from the proof. We then use techniques similar to those used to prove Lemma 20.7 to relate this acceptance probability to the Fourier coefficients of \( f, g \), allowing us to argue that if the verifier’s test accepts with high probability, then \( f \) and \( g \) have few relatively large Fourier coefficients. This will provide us with some nontrivial useful information about \( f \) and \( g \), since in a “generic” or random function, all the Fourier coefficient are small and roughly equal.

20.3.2 Analysis of the linearity test over \( GF(2) \)

We will now prove Theorem 19.23, thus completing the proof of the PCP Theorem. Recall that the linearity test is provided a function \( f : GF(2)^n \rightarrow \)
GF(2) and has to determine whether \( f \) has significant agreement with a linear function. To do this it picks \( x, y \in GF(2)^n \) randomly and accepts iff \( f(x + y) = f(x) + f(y) \).

Now we rephrase this test using \( \{\pm 1\} \) instead of \( GF(2) \), so linear functions turn into Fourier basis functions. For every two vectors \( x, y \in \{\pm 1\}^n \), we denote by \( xy \) their componentwise multiplication. That is, \( xy = (x_1y_1, \ldots, x_ny_n) \).

For two Boolean functions \( f, g \), \( \langle f, g \rangle \) is equal to the fraction of inputs on which they agree minus the fraction of inputs on which they disagree. It follows that for every \( \epsilon \in [0, 1] \) and functions \( f, g : \{\pm 1\}^n \rightarrow \{\pm 1\} \), \( f \) has agreement \( \frac{1}{2} + \frac{\epsilon}{2} \) with \( g \) iff \( \langle f, g \rangle = \epsilon \). Thus, if \( f \) has a large Fourier coefficient then it has significant agreement with some Fourier basis function, or in the GF(2) worldview, \( f \) is close to some linear function. This means that Theorem 19.23 can be rephrased as follows:

**Theorem 20.9**

Suppose that \( f : \{\pm 1\}^n \rightarrow \{\pm 1\} \) satisfies \( \Pr_{x,y}[f(xy) = f(x)f(y)] \geq \frac{1}{2} + \epsilon \). Then, there is some \( \alpha \subseteq [n] \) such \( f_\alpha \geq 2\epsilon \).

**Proof:** We can rephrase the hypothesis as \( E_{x,y}[f(xy)f(x)f(y)] \geq \left(\frac{1}{2} + \epsilon\right) - (\frac{1}{2} - \epsilon) = 2\epsilon \). We note that from now on we do not need \( f \) to be Boolean, but merely to satisfy \( \langle f, f \rangle = 1 \).

Expressing \( f \) by its Fourier expansion,

\[
2\epsilon \leq E_{x,y}[f(xy)f(x)f(y)] = E_{x,y}[(\sum_\alpha f_\alpha \chi_\alpha(xy))(\sum_\beta f_\beta \chi_\beta(x))(\sum_\gamma f_\gamma \chi_\gamma(y))].
\]

Since \( \chi_\alpha(xy) = \chi_\alpha(x)\chi_\alpha(y) \) this becomes

\[
= E_{x,y}[(\sum_\alpha f_\alpha \hat{f}_\beta \hat{f}_\gamma \chi_\alpha(x)\chi_\alpha(y)\chi_\beta(x)\chi_\gamma(y))].
\]

Using linearity of expectation:

\[
= \sum_\alpha \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma E_{x,y}[\chi_\alpha(x)\chi_\alpha(y)\chi_\beta(x)\chi_\gamma(y)]
\]

\[
= \sum_\alpha \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma E_x[\chi_\alpha(x)\chi_\beta(x)] E_y[\chi_\alpha(y)\chi_\gamma(y)]
\]

(because \( x, y \) are independent).
By orthonormality $E_x[\chi_\alpha(x)\chi_\beta(x)] = \delta_{\alpha,\beta}$, so we simplify to

$$= \sum_\alpha \hat{f}_\alpha^3$$

$$\leq (\max_\alpha \hat{f}_\alpha) \times (\sum_\alpha \hat{f}_\alpha^2)$$

Since $\sum_\alpha \hat{f}_\alpha^2 = \langle f, f \rangle = 1$, this expression is at most $\max_\alpha \{ \hat{f}_\alpha \}$. Hence $\max_\alpha \hat{f}_\alpha \geq 2\epsilon$ and the theorem is proved. ■

20.3.3 Coordinate functions, Long code and its testing

Let $W \in \mathbb{N}$. We say that $f : \{\pm 1\}^W \rightarrow \{\pm 1\}$ is a coordinate function if there is some $w \in [W]$, such that $f(x_1, x_2, \ldots, x_W) = x_w$; in other words, $f = \chi_{\{w\}}$.

**Definition 20.10 (Long Code)**

The long code for $[W]$ encodes each $w \in [W]$ by the table of all values of the function $\chi_{\{w\}} : \{\pm 1\}^W \rightarrow \{\pm 1\}$.

**Remark 20.11**

Note that $w$, normally written using $\log W$ bits, is being represented using a table of $2^W$ bits, a doubly exponential blowup! This inefficiency is the reason for calling the code “long.”

Similar to the test for the Walsh-Hadamard code, when testing the long code, we are given a function $f : \{\pm 1\}^W \rightarrow \{\pm 1\}$, and want to find out if $f$ has good agreement with $\chi_{\{w\}}$ for some $w$, namely, $\hat{f}_{\{w\}}$ is significant. Such a test is described in Exercise 16 of the previous chapter, but it is not sufficient for the proof of Håstad’s Theorem, which requires a test using only three queries. Below we show such a three query test albeit at the expense of achieving the following weaker guarantee: if the test passes with high probability then $f$ has a good agreement with a function $\chi_{\alpha}$ with $|\alpha|$ small (but not necessarily equal to 1). This weaker conclusion will be sufficient in the proof of Theorem 20.5.

Let $\rho > 0$ be some arbitrarily small constant. The test picks two uniformly random vectors $x, y \in \{\pm 1\}^W$ and then a vector $z \in \{\pm 1\}^W$ according to the following distribution: for every coordinate $i \in [W]$, with probability $1 - \rho$ we choose $z_i = +1$ and with probability $\rho$ we choose
Thus with high probability, about $\rho$ fraction of coordinates in $\mathbf{z}$ are $-1$ and the other $1-\rho$ fraction are $+1$. We think of $\mathbf{z}$ as a “noise” vector. The test accepts iff $f(x)f(y) = f(xy\mathbf{z})$. Note that the test is similar to the linearity test except for the use of the noise vector $\mathbf{z}$.

Suppose $f = \chi_{\{w\}}$. Then

$$f(x)f(y)f(xy\mathbf{z}) = x_w y_w (x_w y_w z_w) = 1 \cdot z_w$$

Hence the test accepts iff $z_w = 1$ which happens with probability $1-\rho$. We now prove a certain converse:

**Lemma 20.12**

If the test accepts with probability $\frac{1}{2} + \epsilon$ then $\sum_{\alpha} \hat{f}^3_{\alpha} (1 - 2\rho)^{|\alpha|} \geq 2\epsilon$.

**Proof:** If the test accepts with probability $\frac{1}{2} + \epsilon$ then $E[f(x)f(y)f(xy\mathbf{z})] = 2\epsilon$. Replacing $f$ by its Fourier expansion, we have

$$2\epsilon \leq \sum_{\alpha} \hat{f}^3_{\alpha} \sum_{\beta} \hat{f}_{\beta} \sum_{\gamma} \hat{f}_{\gamma} \chi_{\alpha}(x) \chi_{\beta}(y) \chi_{\gamma}(x) \chi_{\gamma}(y) \chi_{\gamma}(z)$$

Orthonormality implies the expectation is 0 unless $\alpha = \beta = \gamma$, so this is

$$= \sum_{\alpha,\beta,\gamma} \hat{f}_{\alpha} \hat{f}_{\beta} \hat{f}_{\gamma} E_{x,y,z} [\chi_{\alpha}(x) \chi_{\beta}(y) \chi_{\gamma}(x) \chi_{\gamma}(y) \chi_{\gamma}(z)]$$

Now $E_{\mathbf{z}}[\chi_{\alpha}(\mathbf{z})] = E_{\mathbf{z}} [\prod_{w \in \alpha} z_w]$ which is equal to $\prod_{w \in \alpha} E[z_w] = (1-2\rho)^{|\alpha|}$ because each coordinate of $\mathbf{z}$ is chosen independently. Hence we get that

$$2\epsilon \leq \sum_{\alpha} \hat{f}^3_{\alpha} (1 - 2\rho)^{|\alpha|}$$

The conclusion of Lemma 20.12 is reminiscent of the calculation in the proof of Theorem 20.9, except for the extra factor $(1 - 2\rho)^{|\alpha|}$. This factor depresses the contribution of $\hat{f}_{\alpha}$ for large $\alpha$, allowing us to conclude that the small $\alpha$’s must contribute a lot. This formalized in the following corollary (left as Exercise 2).
Corollary 20.13
If \( f \) passes the long code test with probability \( \frac{1}{2} + \delta \) then
\[
\sum_{\alpha : |\alpha| \leq k} \hat{f}_\alpha^3 \geq 2\delta - \epsilon,
\]
where \( k = \frac{1}{2p} \log \frac{1}{\gamma} \).

20.4 Proof of Theorem 20.5

Recall that our proof of the PCP Theorem implies that there are constants \( \gamma > 0, s \in \mathbb{N} \) such that \( (1-\gamma)\)-GAP 2CSP is \( \text{NP} \)-hard (see Claim 19.36). This means that for every \( \text{NP} \)-language \( L \) we have a PCP-verifier for \( L \) making two queries over alphabet \( \{0, \ldots, s-1\} \) with perfect completeness and soundness parameter \( 1 - \gamma \). Furthermore this PCP system has the property that the verifier accepts the answer pair \( z_1, z_2 \) iff \( z_2 = h_r(z_1) \) where \( h_r \) is a function (depending on the verifier’s randomness \( r \)) mapping \( \{0, \ldots, s-1\} \) to itself (see Exercise 3). We call this the projection property. Using the Raz’s parallel repetition lemma (Theorem 20.3), we can reduce the soundness parameter to an arbitrary small constant at the expense of increasing the alphabet. Note that parallel repetition preserves the projection property.

Let \( L \) be an \( \text{NP} \)-language and \( \epsilon > 0 \) an arbitrarily small constant. By the above there exists a constant \( W \) and PCP-verifier \( V_{Raz} \) (having the projection property) that makes two queries to a polynomial-sized PCP proof \( \pi \) with alphabet \( \{1, \ldots, W\} \) such that for every \( x \) if \( x \in L \) then there exists \( \pi \) such that \( \Pr[V_{Raz}^\pi(x) = 1] = 1 \) and if \( x \notin L \) then \( \Pr[V_{Raz}^\pi(x) = 1] < \epsilon \) for every \( \pi \).

Now we describe Håstad’s verifier \( V_H \). It essentially follows \( V_{Raz} \), but it expects each entry in the PCP proof \( \pi \) to be encoded using the long code. It expects these encodings to be bifolded, a technical property we now define and is motivated by the observation that coordinate functions satisfy \( \chi_{\{w\}}(-u) = -\chi_{\{w\}}(u) \), where \( -u \) is the vector \((-u_1, \ldots, -u_W)\).

**Definition 20.14**
A function \( f : \{\pm 1\}^W \to \{\pm 1\} \) is bifolded if for all \( u \in \{\pm 1\}^W \), \( f(-u) = -f(u) \).

Whenever the PCP proof is supposed to contain a longcode codeword then we may assume without loss of generality that the function is bifolded. The reason is that the verifier can identify, for each pair of inputs \( u, -u \), one designated representative —say the one whose first coordinate is \(+1\)— and
just define $f(u)$ to be $-f(u)$. One benefit — though of no consequence in the proof — of this convention is that bifolded functions require only half as many bits to represent. We will use the following fact:

**Lemma 20.15**

If $f : \{\pm 1\}^W \to \{\pm 1\}$ is bifolded and $\hat{f}_\alpha \neq 0$ then $|\alpha|$ must be an odd number (and in particular, nonzero).

**Proof:** By definition,

$$\hat{f}_\alpha = \langle f, \chi_\alpha \rangle = \frac{1}{2^n} \sum_u f(u) \prod_{i \in \alpha} u_i.$$  

If $|\alpha|$ is even then $\prod_{i \in \alpha} u_i = \prod_{i \in \alpha} (-u_i)$. So if $f$ is bifolded, the terms corresponding to $u$ and $-u$ have opposite signs and the entire sum is 0. ■

**Hästad’s verifier.** Recall that $V_{Raz}$ uses its randomness to select a function two entries $i, j$ in the table $\pi$ and a function $h : [W] \to [W]$, and accepts iff $\pi(j) = h(\pi(i))$. Hästad’s verifier, denoted $V_H$, expects the proof $\tilde{\pi}$ to consist of bifolded longcode encodings of each entry of $\pi$. The verifier $V_H$ emulates $V_{Raz}$ to pick two locations $i, j$ in the table and a function $h : [W] \to [W]$ such that $V_{Raz}$’s test is to accept iff $\pi[j] = h(\pi[i])$. The proof $\tilde{\pi}$ contains in the locations $i$ and $j$ two functions $f$ and $g$ respectively (which may or may not be the longcode encoding of $\pi(i)$ and $\pi(j)$). Instead of reading the long codes $f, g$ in their entirety, the verifier $V_H$ performs a simple test that is reminiscent of the long code test. For a string $y \in \{\pm 1\}^W$ we denote by $h^{-1}(y)$ the string such that for every $w \in [W]$, $h^{-1}(y)_w = y_{h(w)}$. In other words, for each $u \in [W]$, the bit $y_u$ appears in all coordinates of $h^{-1}(y)$ that are indexed by integers in the subset $h^{-1}(u)$. This is well defined because $\{h^{-1}(u) : u \in [W]\}$ is a partition of $[W]$. $V_H$ chooses uniformly at random $u, y \in \{\pm 1\}^W$ and chooses $z \in \{\pm 1\}^W$ by letting $z_i = +1$ with probability $1 - \rho$ and $z_i = -1$ with probability $\rho$. It then accepts if

$$f(u)g(y) = f(h^{-1}(y)uz)$$  

(1)

Translating back from $\{\pm 1\}$ to $\{0, 1\}$, note that $V_H$’s test is indeed linear, as it accepts iff $\hat{\pi}[i_1] + \hat{\pi}[i_2] + \hat{\pi}[i_3] = b$ for some $i_1, i_2, i_3 \in [m2^W]$ and $b \in \{0, 1\}$. (The bit $b$ can indeed equal 1 because of the way $V_H$ ensures the bifolding property.)
Completeness of $V_H$. Suppose $f, g$ are long codes of two integers $w, u$ satisfying $h(w) = u$ (in other words, $V_{raz}$ would have accepted the assignments represented by these integers). Then

$$f(u)g(y)f(h^{-1}(y)uz) = u_wy_u(h^{-1}(y)uz_w) = u_wy_u(y_{h(w)w}z_w) = z_w.$$ 

Hence $V_H$ accepts iff $z_w = 1$, which happens with probability $1 - \rho$.

Soundness of $V_H$. We now show that if $V_H$ accepts $f, g$ with probability significantly more than $1/2$, then the Fourier transforms of $f, g$ must be correlated. To formalize this we define for $\alpha \subseteq [W],$

$$h_2(\alpha) = \{ u \in [W] : |h^{-1}(u) \cap \alpha| \text{ is odd} \}$$

Notice in particular that for every $u \in h_2(\alpha)$ there is at least one $w \in \alpha$ such that $h(w) = u$.

In the next Lemma $\delta$ is allowed to be negative.

**Lemma 20.16**

Let $f, g : \{\pm 1\}^W \to \{\pm 1\}$, $h : [W] \to [W]$ be bifooled functions passing $V_H$’s test (1) with probability at least $1/2 + \delta$. Then

$$\sum_{\alpha \subseteq [W], \alpha \neq \emptyset} \hat{f_\alpha}^2 \hat{g_{h_2(\alpha)}}(1 - 2\rho)^{\alpha} \geq 2\delta$$

**Proof:** By hypothesis, $f, g$ are such that $E[f(u)f(uh^{-1}(y)z)g(y)] \geq 2\delta$. Replace $f, g$ by their Fourier expansions. We get that

$$2\delta = E_{u,y,z} \left( \sum_{\alpha} \hat{f_\alpha} \chi_\alpha(x) \left( \sum_{\beta} \hat{g_\beta} \chi_\beta(y) \left( \sum_{\gamma} \hat{f_\gamma} \chi_\gamma(uh^{-1}(y)z) \right) \right) \right)$$

By orthornormality this simplifies to

$$= \sum_{\alpha, \beta, \gamma} \hat{f_\alpha} \hat{g_\beta} \hat{f_\gamma} E_{u,y,z} \left[ \chi_\alpha(u) \chi_\beta(y) \chi_\gamma(h^{-1}(y)) \chi_\gamma(z) \right]$$

By orthornormality this simplifies to

$$= \sum_{\alpha, \beta} \hat{f_\alpha}^2 \hat{g_\beta} E_{y,z} \left[ \chi_\beta(y) \chi_\alpha(h^{-1}(y)) \chi_\alpha(z) \right]$$

$$= \sum_{\alpha, \beta} \hat{f_\alpha}^2 \hat{g_\beta} (1 - 2\rho)^{\alpha} E_{y} \left[ \chi_\alpha(h^{-1}(y)) \chi_\beta(y) \right] \quad (2)$$
since $\chi_\alpha(z) = (1-2\rho)^{||\alpha||}$, as noted in our analysis of the long code test. Now we have
\[
E_y[\chi_\alpha(h^{-1}(y))\chi_\beta(y)] = E_y[\prod_{w \in \alpha} h^{-1}(y)_w \prod_{u \in \beta} y_u] = E_y[\prod_{w \in \alpha} y_{h(w)} \prod_{u \in \beta} y_u],
\]
which is 1 if $h_2(\alpha) = \beta$ and 0 otherwise. Hence (2) simplifies to
\[
\sum_\alpha \hat{f}_\alpha^2 \hat{g}_{h_2(\alpha)} (1-2\rho)^{||\alpha||}.
\]
Finally we note that since the functions are assumed to be bifolded, the Fourier coefficients $\hat{f}_\emptyset$ and $\hat{g}_\emptyset$ are zero. Thus those terms can be dropped from the summation and the Lemma is proved. ■

The following corollary of Lemma 20.16 completes the proof of Håstad’s 3-bit PCP Theorem.

Corollary 20.17
Let $\epsilon$ be the soundness parameter of $V_{Raz}$. If $\rho, \delta$ satisfy $\rho \delta^2 > \epsilon$ then the soundness parameter of $V_H$ is at most $1/2 + \delta$.

Proof: Suppose $V_H$ accepts a proof $\tilde{\pi}$ with probability at least $1/2 + \delta$. We give a probabilistic construction of a proof $\pi$ causing $V_{Raz}$ to accept the same statement with probability at least $\rho \delta^2$.

Suppose that $V_{Raz}$ uses proofs $\pi$ with $m$ entries in $[W]$. We can think of $\tilde{\pi}$ as providing, for every $i \in [m]$, a function $f_i : \{\pm 1\}^W \rightarrow \{\pm 1\}$. We will use $\tilde{\pi}$ to construct a proof $\pi$ for $V_{Raz}$ as follows: we first use $f_i$ to come up with a distribution $D_i$ over $[W]$. We then let $\pi[i]$ be a random element from $D_i$.

The distribution $D_i$. Let $f = f_i$. The distribution $D_i$ is defined by first selecting $\alpha \subseteq [W]$ with probability $\hat{f}_\alpha^2$ and then selecting $w$ at random from $\alpha$. This is well defined because $\sum_\alpha \hat{f}_\alpha^2 = 1$ and (due to bifolding) $\hat{f}_\emptyset = 0$.

Recall that $V_{Raz}$ picks using its random tape a pair $i, j$ of locations and a function $h : [W] \rightarrow [W]$ and then verifies that $\pi[j] = h(\pi[i])$. Let $r$ be some possible random tape of $V_{Raz}$ and let $i, j, h$ be the pair of entries in $\pi$ and function that are determined by $r$. We define the indicator random variable $I_r$ to be 1 if for $w \in R D_i$ and $u \in R D_j$ it holds that $w = h(u)$ and to be 0 otherwise. Thus, our goal is to show that
\[
E_{\pi=D_1,\ldots,D_m} [E_r[I_r]] \geq \rho \delta^2
\]
20.4. PROOF OF THEOREM ??

since that would imply that there exists a table $\pi$ causing $V_{Raz}$ to accept with probability at least $\rho \delta^2$, proving the corollary.

To prove (3) we first notice that linearity of expectation allows us to exchange the order of the two expectations and so it is enough to bound $E_r[E_{D_i,D_j}[I_r]]$ where $i,j$ are the entries determined by the random tape $r$. For every $r$ denote by $\delta_r$ the probability that $V_H$ accepts $\pi$ when it uses $r$ as the random tape for $V_{Raz}$. The acceptance probability of $V_H$ is $E_r[\frac{1}{2} + \delta_r]$ and hence $E_r[\delta_r] = \delta$.

Let $i,j,h$ be the pair and function determined by $r$ and denote by $f = f_i$ and $g = f_j$ where $f_i$ (resp. $f_j$) is the function at the $i^{\text{th}}$ (resp. $j^{\text{th}}$) entry of the table $\tilde{\pi}$. What is the chance that a pair of assignments $w \in R D_i$ and $v \in R D_j$ will satisfy the constraint? (i.e., will satisfy $v = h(w)$?). Recall that we pick $w$ and $u$ by choosing $\alpha$ with probability $\hat{f}_\alpha^2$ and choosing $w \in R \alpha, v \in R \beta$. Now if $\beta = h_2(\alpha)$ then for every $v \in \beta$ there exists $w \in \alpha$ with $h(w) = v$ and hence the probability the constraint is satisfied is at least $1/|\alpha|$. Thus, we have that

$$\sum_{\alpha} \frac{1}{|\alpha|} \hat{f}_\alpha^2 \hat{g}_{h_2(\alpha)}^2 \leq E_{D_i,D_j}[I_r] \quad (4)$$

This is similar to (but not quite the same as) the expression in Lemma 20.16, according to which

$$2\delta_r \leq \sum_{\alpha} \hat{f}_\alpha^2 \hat{g}_{h_2(\alpha)}^2 (1 - 2\rho)^{|\alpha|}.$$  

However, since one can easily see that $(1 - 2\rho)^{|\alpha|} \leq \frac{2}{\sqrt{\rho |\alpha|}}$ we have

$$2\delta_r \leq \sum_{\alpha} \hat{f}_\alpha^2 \left| \hat{g}_{h_2(\alpha)} \right| \frac{2}{\sqrt{\rho |\alpha|}}$$

Or

$$\delta_r \sqrt{\rho} \leq \sum_{\alpha} \hat{f}_\alpha^2 \frac{1}{\sqrt{|\alpha|}}$$

Applying the Cauchy-Schwartz inequality, $\sum_i a_i b_i \leq (\sum_i a_i^2)^{1/2}(\sum_i b_i^2)^{1/2}$, with $\hat{f}_\alpha \left| \hat{g}_{\pi_2(\alpha)} \right| \frac{1}{\sqrt{|\alpha|}}$ playing the role of the $a_i$’s and $\hat{f}_\alpha$ playing that of the $b_i$’s, we obtain

$$\delta_r \sqrt{\rho} \leq \sum_{\alpha} \hat{f}_\alpha^2 \left| \hat{g}_{h_2(\alpha)} \right| \frac{1}{\sqrt{|\alpha|}} \leq \left( \sum_{\alpha} \hat{f}_\alpha^2 \right)^{1/2} \left( \sum_{\alpha} \hat{f}_\alpha^2 \hat{g}_{h_2(\alpha)}^2 \frac{1}{|\alpha|} \right)^{1/2} \quad (5)$$
Since \( \sum_\alpha \hat{\alpha}^2 = 1 \), by squaring \( 5 \) and combining it with \( 4 \) we get that for every \( r \),

\[
\delta_r^2 \leq E_{D_i,D_j}[I_r]
\]

taking expectation over \( r \) and using \( E[X]^2 \leq E[X^2] \) we get that

\[
\delta^2 \leq E_r[\delta_r^2] \leq E_r[E_{D_i,D_j}[I_r]]
\]

proving \( 3 \).

### 20.5 Learning Fourier Coefficients

Suppose that you are given random access to a Boolean function \( f : \{\pm 1\}^n \rightarrow \{\pm 1\} \) and want to find the high Fourier coefficients of \( f \). Of course, we can compute all of the coefficients in time polynomial in \( 2^n \), but is there a faster algorithm? By the Parseval equality (Lemma 20.7) we know that there can be at most \( 1/\epsilon^2 \) coefficients with absolute value larger than \( \epsilon \), and so we can hope to learn these coefficients in time polynomial in \( n \), and \( 1/\epsilon \). It turns out we can (almost) achieve this goal:

**Theorem 20.18** ([?])

There is an algorithm \( A \) that given input \( n \in \mathbb{N}, \epsilon \in (0,1) \) and random access to a function \( f : \{\pm 1\}^n \rightarrow \{\pm 1\} \), runs in \( \text{poly}(n,1/\epsilon) \) time and with probability at least 0.9 outputs a set \( L \) of size at most \( O(1/\epsilon^2) \) such that for every \( \alpha \subseteq [n] \), if \( |\hat{\alpha}| > \epsilon \) then \( \alpha \in L \).

**Proof:** We identify subsets of \( [n] \) with strings in \( \{0,1\}^m \) in the obvious way. For \( k \leq n \) and \( \alpha \in \{0,1\}^k \) denote

\[
\tilde{f}_{\alpha*} = \sum_{\beta \in \{0,1\}^{n-k}} \tilde{f}_{\alpha \beta}^2,
\]

where \( \circ \) denotes concatenation. By Parseval (Lemma 20.7) \( \tilde{f}_{\ast} = 1 \). Note also that for every \( k < n \) and \( \alpha \in \{0,1\}^k \), \( \tilde{f}_{\alpha*} = \tilde{f}_{\alpha0*} + \tilde{f}_{\alpha1*} \). Therefore, if we think of the full depth-\( n \) binary labeled by binary strings of length \( \leq n \) (with the root being the empty word and the two children of \( \alpha \) are \( \alpha0 \) and \( \alpha1 \)), then at any level of this tree there can be at most \( 1/\epsilon^2 \) strings \( \alpha \) such that \( \tilde{f}_{\alpha*} > \epsilon^2 \) (the \( k \)th level of the tree corresponds to all strings of length \( k \)). Note that if a string \( \alpha \) satisfies \( \tilde{f}_{\alpha*} < \epsilon^2 \) then the same holds for every string of the form \( \alpha \circ \beta \). Our goal will be to find all these strings at all levels,
and then output all the strings that label leaves in the tree (i.e., all n-bit strings).

The heart of the algorithm is a procedure Estimate that given $\alpha$ and
oracle access to $f(\cdot)$, outputs an estimate of $f_\alpha$ within $\epsilon/4$ accuracy with
probability $1 - \frac{e^2}{100n}$. Using this procedure we work our way from the root
down, and whenever Estimate($\alpha$) gives a value smaller than $\epsilon/2$ we “kill”
this node and will not deal with it and its subnodes. Note that unless the
output is more than $\epsilon/4$-far from the real value (which we will
ensure by the union bound happens with probability less than 0.1 over all
the levels) at most $4/\epsilon$ nodes will survive at any level. The algorithm will
output the $4/\epsilon$ leaves that survive.

The procedure Estimate uses the following claim:

**Claim 20.19**

For every $\alpha$,

$$\hat{f}_{\alpha^*} = E_{x,x' \in \{0,1\}^n, y \in \{0,1\}^n} [f(x \circ y)f(x' \circ y)\chi_\alpha(x)\chi_\alpha(x')]$$

**Proof:** We start with the case that $\alpha = 0^k$. To get some intuition, suppose that $\hat{f}_{0^k^*} = 1$. This means that $f$ can be expressed as a sum of functions of
the form $\chi_{0^k \circ \beta}$ and hence it does not depend on its first $k$ variables. Thus
$f(x \circ y) = f(x' \circ y)$ and we’ll get that $E[f(x \circ y)f(x' \circ y)] = E[f(x)^2] = 1$. More
generally, if $\hat{f}_{0^k^*}$ is large then that means that in the Fourier representation,
the weight of functions not depending on the first $k$ variables is large and
hence we expect large correlation between $f(x' \circ y)$ and $f(x \circ y)$. This is
verified by the following calculations:

$$2^{-n-k} \sum_{x,x',y} f(x \circ y)f(x' \circ y) \overset{\text{basis change}}{=} 2^{-n-k} \sum_{x,x',y} \left( \sum_{\gamma \circ \beta} \hat{f}(\gamma \circ \beta)\chi_{\gamma \circ \beta}(x \circ y) \right) \left( \sum_{\gamma' \circ \beta'} \hat{f}(\gamma' \circ \beta') \chi_{\gamma' \circ \beta'}(x' \circ y) \right) \overset{\chi_{\gamma \circ \beta}(x \circ y) = \chi_\gamma(x)\chi_\beta(y)}{=} \sum_{\gamma \circ \beta} \hat{f}(\gamma \circ \beta)\chi_\gamma(x)\chi_\beta(y) \cdot \sum_{\gamma' \circ \beta'} \hat{f}(\gamma' \circ \beta')\chi_{\gamma'}(x')\chi_{\beta'}(y)$$

$$= \sum_{\gamma \circ \beta, \gamma' \circ \beta'} \hat{f}(\gamma \circ \beta) \hat{f}(\gamma' \circ \beta') 2^{-k} \sum_{x} \chi_\gamma(x) 2^{-k} \sum_{x'} \chi_{\gamma'}(x') \sum_{y} \chi_\beta(y) \chi_{\beta'}(y)$$

$$= \sum_{\gamma \circ \beta, \gamma' \circ \beta'} \hat{f}(0^k \circ \beta) \hat{f}(0^k \circ \beta') \delta_{\beta, \beta'} = \sum_{\beta} \hat{f}(0^k \circ \beta)^2 = \hat{f}_{0^k^*}$$
For the case $\alpha \neq 0^k$, we essentially add these factors to translate it to the case $\alpha = 0^k$. Indeed one can verify that if we define $g(x \circ y) = f(x \circ y) \chi_\alpha(x)$ then for every $\beta \in \{0, 1\}^{n-k}$. $g_{\alpha \circ \beta} = f_{\alpha \circ \beta}$. ■

By the Chernoff bound, we can estimate the expectation of Claim 20.19 (and hence $f_{\alpha \star}$) using repeated sampling, thus obtaining the procedure Estimate and completing the proof. ■

20.6 Other PCP Theorems: A Survey

The following variants of the PCP Theorem have been obtained and used for various applications.

20.6.1 PCP’s with sub-constant soundness parameter.

Because $\ell$-times parallel repetition transforms a proof of size $m$ to a proof of size $m^\ell$, we cannot use it with $\ell$ larger than a constant and still have a polynomial-sized proof. Fortunately, there have been direct constructions of PCP’s achieving low soundness using larger alphabet size, but without increasing the proof’s size. Raz and Safra [?] show that there is an absolute constant $q$ such that for every $W \leq \sqrt{\log n}$, every NP language has a $q$-query verifier over alphabet $\{0, \ldots, W - 1\}$ that uses $O(\log n)$ random bits, and has soundness $2^{-\Omega(\log W)}$.

20.6.2 Amortized query complexity.

Some applications require binary-alphabet PCP systems enjoying a tight relation between the number of queries (that can be an arbitrarily large constant) and the soundness parameter. The relevant parameter here turns out to be the free bit complexity [?, ?]. This parameter is defined as follows. Suppose the number of queries is $q$. After the verifier has picked its random string, and picked a sequence of $q$ addresses, there are $2^q$ possible sequences of bits that could be contained in those addresses. If the verifier accepts for only $t$ of those sequences, then we say that the free bit parameter is $\log t$ (note that this number need not be an integer). In fact, for most applications it suffices to consider the amortized free bit complexity [?]. This parameter is defined as $\lim_{s \to 0} f_s / \log(1/s)$, where $f_s$ is the number of free bits needed by the verifier to ensure the soundness parameter is at most $s$. Håstad constructed systems with amortized free bit complexity tending to zero [?]. That is, for every $\epsilon > 0$, he gave a PCP-verifier for NP that
uses $O(\log n)$ random bits and $\epsilon$ amortized free bits. He then used this PCP system to show (using tools from [?, ?, ?]) that $\text{MAX INDEGREE}$ (and so, equivalently, $\text{MAX CLIQUE}$) is $\text{NP}$-hard to approximate within a factor of $n^{1-\epsilon}$ for arbitrarily small $\epsilon > 0$.

### 20.6.3 Unique games.

**Exercises**

§1 Prove that there is a polynomial-time algorithm that given a satisfiable $\text{2CSP}_W$ instance $\varphi$ over $\{0..W-1\}$ where all the constraints are permutations (i.e., $\varphi_i$ checks that $u_{j'} = h(u_j)$ for some $j, j' \in [n]$ and permutation $h : \{0..W-1\} \rightarrow \{0..W-1\}$) finds a satisfying assignment $u$ for $\varphi$.

§2 Prove Corollary 20.13.

§3 Prove that the PCP system resulting from the proof of Claim 19.36 (Chapter 19) satisfies the projection property.

§4 Let $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ and let $I \subseteq [n]$. Let $M_I$ be the following distribution: we choose $z \in_R M_I$ by for $i \in I$, choose $z_i$ to be $+1$ with probability $1/2$ and $-1$ with probability $1/2$ (independently of other choices), for $i \notin I$ choose $z_i = +1$. We define the variation of $f$ on $I$ to be $\Pr_{x \in_R \{-1,1\}^n, z \in_R M_I}[f(x) \neq f(xz)]$.

Suppose that the variation of $f$ on $I$ is less than $\epsilon$. Prove that there exists a function $g : \{\pm 1\}^n \rightarrow \mathbb{R}$ such that (1) $g$ does not depend on the coordinates in $I$ and (2) $g$ is $10\epsilon$-close to $f$ (i.e., $\Pr_{x \in_R \{-1,1\}^n}[f(x) \neq g(x)] < 10\epsilon$). Can you come up with such a $g$ that outputs values in $\{\pm 1\}$ only?

§5 For $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ and $x \in \{\pm 1\}^n$ we define $N_f(x)$ to be the number of coordinates $i$ such that if we let $y$ to be $x$ flipped at the $i^{th}$ coordinate (i.e., $y = xe^i$ where $e^i$ has $-1$ in the $i^{th}$ coordinate and $+1$ everywhere else) then $f(x) \neq f(y)$. We define the average sensitivity of $f$, denoted by $as(f)$ to be the expectation of $N_f(x)$ for $x \in_R \{\pm 1\}^n$.

(a) Prove that for every balanced function $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ (i.e., $\Pr[f(x) = +1] = 1/2$), $as(f) \geq 1$.

(b) Let $f$ be balanced function from $\{\pm 1\}^n$ to $\{\pm 1\}$ with $as(f) = 1$. Prove that $f$ is a coordinate function or its negation (i.e., $f(x) = x_i$ or $f(x) = -x_i$ for some $i \in [n]$ and for every $x \in \{\pm 1\}^n$).
20.6. OTHER PCP THEOREMS: A SURVEY