S. Hallgren
Vazirani Class #6
September 11, 1997

# Fourier Sampling

## 1 Hadamard Transform

So far we have been using the identity between $\{0,1\}^n$ and $Z_2^n$. So to do the Hadamard Transform on $n$ bits, we use the matrix $H_n = H \bigotimes H \bigotimes \ldots \bigotimes H$, which is the $n$-fold tensor product of the following Hadamard matrix:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

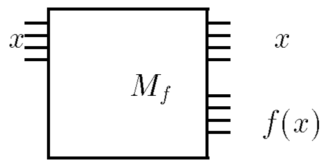
For example,

$$H_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

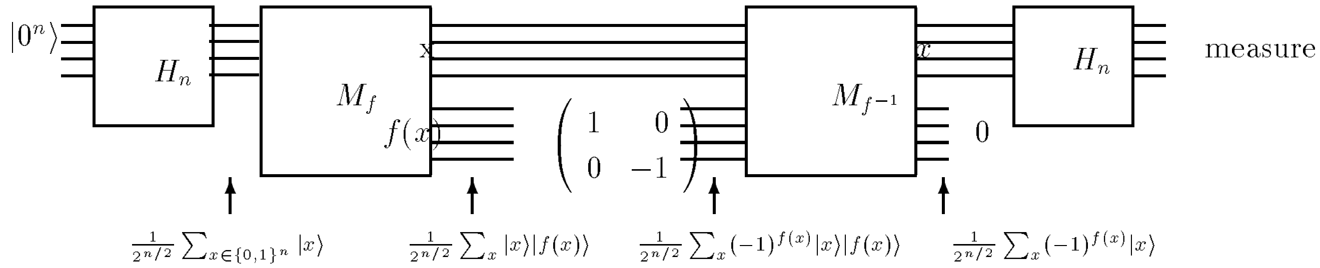
Now the columns are the new basis. Note that $H^2 = I$, so $H_n^2 = I$, since inverse in the tensor product is the inverse in each component.

## 2 The Deutsch-Josa Problem

Given a function $f : \{0,1\}^n \to \{0,1\}$ with the following promise about the function. Either $f(x) = 0$ for all $x$, or $f(x) = 0$ for exactly $2^n/2$ $x$. The algorithm must output which is the case. The function is given as the reversible circuit:




The following quantum circuit solves the problem:



If $f(x) = 0$ for all $x$, then $|0^n\rangle$ will be the final superposition. If $f(x) = 0$ for half of the $x$'s, then the final superposition will be some non-zero value. So after measurement, it can be determined which case it is.

Classically if $f$ is generated by an adversary, the algorithm may need to look at an exponential number of bits. Using randomness however, only a couple of probes are needed to to solve the problem.

## 3 Recursive Fourier Sampling

Now suppose $f$ is chosen to be one of the Hadamard basis functions (i.e., a column of the matrix). Then $f(x) = u \cdot x \pmod 2$, and the circuit will output $u$. This is counterintuitive since the circuit only makes two calls to the function, but gets $n$ bits of information. Classically $n$ bits must be read, and by making the problem recursive we will amplify this difference between 2 and $n$.

Before the structure was a $2^n$-ary tree. In the recursive version, instead of just calling the function at the leaf, a recursive problem of size $n/2$ must be solved first to get the bit. Although, the problem returns a string and we need a bit, so we define another function $g$ (for example, majority of the bits) and use $g(u)$ as the bit.

The depth of the tree is $\log n$. The top level has $2^n$ leaves, the next has $2^{n/2}$, and so on. Intuitively, a classical machine should have to make $n$ queries at the top, $n/2$ at each node one down, and so on. That is, classically the time is given by $T(n) = n \cdot T(n/2) +$ poly $= n^{\log n}$. Since the quantum circuit only makes two queries per level, it takes time $T(n) = 2 \cdot T(n/2) + O(n) = O(n \log n)$.

It can be argued that this problem is outside of BPP, thus giving evidence that BPP is strictly contained in BQP. One way to see this is to use Yao's Minimax Theorem. The columns are a finite set of deterministic algorithms (which correspond to what the BPP machine does given a random string), and the rows are inputs for a given size. The entry is one if the algorithm is wrong on the input and zero otherwise. The matrix represents the probability that a randomized algorithm is wrong on its inputs, where the probability is over random strings (i.e., over the deterministic algorithms) $D_A$ and the inputs $D_z$. Yao's theorem says that $Min_{D_A} Max_{D_z} P(incorrect) = Max_{D_z} Min_{D_A} P(incorrect)$, and we can replace $D_z$ in the lhs with the worst case $z$, and we can replace $D_A$ in the rhs with the worst case algorithm to get the following lower bound principle. To give a lower bound on a probabilistic algorithm, fix some probability distribution on inputs and prove the best deterministic algorithm performs badly. It turns out that the probability a BPP machine is correct is about 1/2.