Notes 8: Fourier Transforms

8.1 Continuous Fourier Transform

The Fourier transform is used to represent a function as a sum of constituent harmonics. It is a linear invertible transformation between the time-domain representation of a function, which we shall denote by $h(t)$, and the frequency domain representation which we shall denote by $H(f)$. In one dimension, the Fourier transform pair consisting of the forward and inverse transforms, is often written as

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t)\, \mathrm{e}^{i\, f\, t}\, dt \\ h(t) &= \frac{1}{2\,\pi} \int_{-\infty}^{\infty} H(f)\, \mathrm{e}^{-i\, f\, t}\, df. \end{aligned}$$

Note that the Fourier transform is naturally defined in terms of complex functions. Both $h(t)$ and $H(f)$ are, in general, complex-valued. Normalisation can be a thorny issue however and there are many different conventions in different parts of the literature. In fact the transform pair

$$\begin{aligned} H(f) &= \sqrt{\frac{|b|}{(2\,\pi)^{1-a}}} \int_{-\infty}^{\infty} h(t)\, \mathrm{e}^{i\, b\, f\, t}\, dt \\ h(t) &= \sqrt{\frac{|b|}{(2\,\pi)^{1+a}}} \int_{-\infty}^{\infty} H(f)\, \mathrm{e}^{-i\, b\, f\, t}\, df. \end{aligned}$$

is equally good for any real values of the parameters a and b. In this module we shall adopt the convention $a = 0$ and $b = 2\,\pi$, which is common in the signal processing literature. With this choice the Fourier transform pair is:

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t)\, \mathrm{e}^{2\,\pi\, i\, f\, t}\, dt \\ h(t) &= \int_{-\infty}^{\infty} H(f)\, \mathrm{e}^{-2\,\pi\, i\, f\, t}\, df. \end{aligned} \tag{8.1}$$

The Fourier transform has several important symmetry properties which are often useful. The following properties are easily shown by direct calculation based on the definition:

- Parity is preserved: if $h(t)$ is even/odd then $H(f)$ is even/odd.
- Fourier transform of a real function: if $h(t)$ is real valued then $H(f)$, while still complex-valued, has the following symmetry:

  $$H(-f) = H(f)^*, \tag{8.2}$$

  where $H(f)^*$ denotes the complex conjugate of $H(f)$.

Sometimes it is possible to compute the integral involved in Eq. (8.1) analytically. Consider the important example of a Gaussian function:

$$h(t) = \frac{1}{\sqrt{2\,\pi\,\sigma^2}}\, \mathrm{e}^{-\frac{t^2}{2\,\sigma^2}}.$$

The Fourier transform can be calculated analytically using a standard trick which involves completing the square in the exponent of a Gaussian integral. From Eq. (8.1) we have

$$\begin{aligned} H(f) &= \frac{1}{\sqrt{2\,\pi\,\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{t^2}{2\,\sigma^2}}\, \mathrm{e}^{2\,\pi\,i\,f\,t}\, dt \\ &= \frac{1}{\sqrt{2\,\pi\,\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\,\sigma^2}\left[t^2 - 4\pi i\sigma^2 t f\right]} dt \\ &= \frac{1}{\sqrt{2\,\pi\,\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\,\sigma^2}\left[t^2 - 2(2\pi i\sigma^2 f)t + (2\pi i\sigma^2 f)^2 - (2\pi i\sigma^2 f)^2\right]} dt \\ &= \frac{1}{\sqrt{2\,\pi\,\sigma^2}} \left[\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\,\sigma^2}(t - 2\pi i\sigma^2 f)^2} dt\right] \mathrm{e}^{\frac{1}{2\,\sigma^2}\,(2\pi i\sigma^2 f)^2} \\ &= \frac{1}{\sqrt{2\,\pi\,\sigma^2}} \left[\sqrt{2\,\pi\,\sigma^2}\right] \mathrm{e}^{-2\pi^2\sigma^2 f^2} \\ &= \mathrm{e}^{-2\pi^2\sigma^2 f^2}. \end{aligned}$$

Under the Fourier transform, the Gaussian function is mapped to another Gaussian function with a different width. If σ^2 is large/small then $h(t)$ is narrow/broad in the time domain. Notice how the width is inverted in the frequency domain. This is an example of a general principle: signals which are strongly localised in the time domain are strongly delocalised in the frequency domain and vice versa. This property of the Fourier transform is the underlying reason for the "Uncertainty Principle" in quantum mechanics where the "time" domain describes the position of a quantum particle and the "frequency" domain describes its velocity.

It is somewhat exceptional that the Fourier transform turns out to be a real quantity. In general, the Fourier transform, $H(f)$, of a real function, $h(t)$, is still complex. In fact, the Fourier transform of the Gaussian function is only real-valued because of the choice of the origin for the t-domain signal. If we would shift $h(t)$ in time, then the Fourier tranform would have come out complex. This can be seen from the following translation property of the Fourier transform. Suppose we define $g(t)$ to be a shifted copy of $h(t)$:

$$g(t) = h(t + \tau).$$

Writing both sides of this equation in terms of the Fourier transforms we have

$$\int_{-\infty}^{\infty} G(f)\, \mathrm{e}^{-2\,\pi\,i\,f\,t}\, df = \int_{-\infty}^{\infty} H(f)\, \mathrm{e}^{-2\,\pi\,i\,f\,(t+\tau)}\, df,$$

from which we conclude that

$$G(f) = \mathrm{e}^{-2\pi i\tau}\, h(f). \tag{8.3}$$

Translation in the time-domain therefore corresponds to multiplication (by a complex number) in the frequency domain.

8.1.1 Power Spectrum

Since Fourier transforms are generally complex valued, it is convenient to summarise the spectral content of a signal by plotting the squared absolute value of $H(f)$ as a function of f. This quantity,

$$P(f) = H(f)\, H^*(f), \tag{8.4}$$

is called the power spectrum of $h(t)$. The power spectrum is fundamental to most applications of Fourier methods.

8.1.2 Convolution Theorem

The convolution of two functions $f(t)$ and $g(t)$ is the function $(f * g)(t)$ defined by

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t + \tau)\, d\tau. \tag{8.5}$$

Convolutions are used very extensively in time series analysis and image processing, for example as a way of smoothing a signal or image. The Fourier transform of a convolution takes a particularly simple form. Expressing f and g in terms of

their Fourier transforms in Eq. (8.5) above, we can write

$$\begin{aligned}(f*g)(t) &= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} F(f_1)\,\mathrm{e}^{-2\,\pi\,i\,f_1\,\tau}\,df_1\right]\left[\int_{-\infty}^{\infty} g(f_2)\,\mathrm{e}^{-2\,\pi\,i\,f_2\,(t+\tau)}\,df_2\right]d\tau \\ &= \int_{-\infty}^{\infty} df_1 \int_{-\infty}^{\infty} df_2\, F(f_1)\,G(f_2)\,\mathrm{e}^{-2\pi i f_1}\int_{-\infty}^{\infty} d\tau \mathrm{e}^{-2\pi i(f_1+f_2)} \\ &= \int_{-\infty}^{\infty} df_1 \int_{-\infty}^{\infty} df_2\, F(f_1)\,G(f_2)\,\mathrm{e}^{-2\pi i f_1}\,\delta(f_1+f_2) \\ &= \int_{-\infty}^{\infty} df_1 F(f_1)\,G^*(f_1)\,\mathrm{e}^{-2\pi i f_1}.\end{aligned}$$

Thus the Fourier transform of the convolution of f and g is simply the product of the individual Fourier transforms of f and g. This fact is sometimes called the Convolution Theorem. One of the most important uses of convolutions is in time-series analysis. The convolution of a signal, $h(t)$, (now thought of as a time-series) with itself is known as the auto-correlation function (ACF) of the signal. It is usually written

$$R(\tau) = \int_{-\infty}^{\infty} h(t+\tau)\,h(t)\,dt. \tag{8.6}$$

The ACF quantifies the memory of a time-series. Using the Convolution Theorem we can write

$$R(\tau) = \int_{-\infty}^{\infty} df\, H(f)\,H^*(f)\,\mathrm{e}^{-2\pi i f} = \int_{-\infty}^{\infty} df\, P(f)\,\mathrm{e}^{-2\pi i f}. \tag{8.7}$$

The ACF of a signal is the inverse Fourier transform of the power spectrum. This fundamental result is known as the Wiener-Kinchin Theorem. It reflects the fact that the frequency domain and time-domain representations of a signal are just different ways of looking at the same information.

The Convolution Theorem means that convolutions are very easy to calculate in the frequency domain. In order for this to be practically useful we must be able to do the integral in Eq. (8.1) required to return from the frequency domain to the time domain. As usual, the set of problems for which this integral can be done using analytic methods is a set of measure zero in the space of all potential problems. Therefore a computational approach to computing Fourier transforms is essential. Such an approach is provided by the Discrete Fourier Transform which we now discuss.

8.2 Discrete Fourier Transform

8.2.1 Time-domain discretisation and the Sampling Theorem

In applications, a time-domain signal, $h(t)$, is approximated by a set of discrete values, $\{h_n = h(t_n),\ n = \ldots -2, -1, 0, 1, 2, \ldots\}$ of the signal measured at a list of consecutive points $\{t_n,\ n = \ldots -2, -1, 0, 1, 2, \ldots\}$. We shall assume that the sample points are uniformly spaced in time: $t_n = n\,\Delta$. The quantity Δ is called the sampling interval and its reciprocal, $1/\Delta$, is the sampling rate. The quantity

$$f_c = \frac{1}{2\Delta}. \tag{8.8}$$

is known as the Nyquist frequency. The importance of the Nyquist frequency will become apparent below.

Clearly, the higher the sampling rate, the more closely the sampled points, $\{h_n\}$, can represent the true underlying function. Of principal importance in signal processing is the question of how high the sampling rate should be in order to represent a given signal to a given degree of accuracy. This question is partially answered by a fundamental result known as the Sampling Theorem (stated here without derivation):

Theorem 8.2.1. *Let $h(t)$ be a function and $\{h_n\}$ be the samples of $h(t)$ obtained with sampling interval Δ. If $H(f) = 0$ for all $|f| \geq f_c$ then $h(t)$ is* completely *determined by its samples. An explicit reconstruction is provided by the Whittaker-Shannon interpolation formula:*

$$h(t) = \Delta \sum_{n=-\infty}^{\infty} h_n \frac{\sin\left[2\pi f_c\,(t-n\Delta)\right]}{\pi\,(t-n\Delta)}.$$

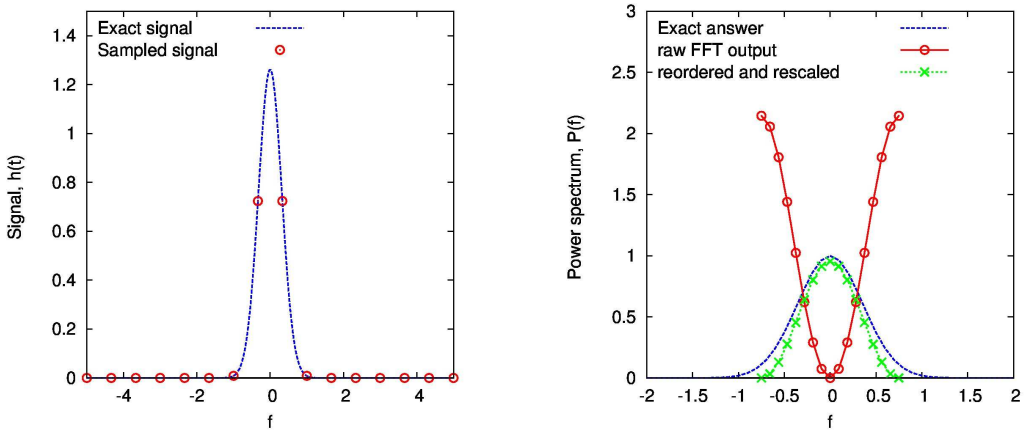

Figure 8.1: Discrete Fourier Transform of a Gaussian sampled with 16 points in the domain [-5:5] using the fftw3 implementation of the Fast Fourier Transform.

This theorem explains why the Nyquist frequency is important. There are two aspects to this. On the one hand, if the signal $h(t)$ is such that the power spectrum is identically zero for large enough absolute frequencies then the Sampling Theorem tells us that it is possible to reconstruct the signal perfectly from its samples. The Nyquist frequency tells us the minimal sampling rate which is required in order to do this. On the other hand, we know that most generic signals have a power spectrum whose tail extends to all frequencies. For generic signals, the amount of power outside the Nyquist interval, $[-f_c, f_c]$, limits the accuracy with which we can reconstruct a signal from its samples. The Nyquist frequency is then used in conjunction with an error tolerance criterion to decide how finely a function must be sampled in order to reconstruct the function from its samples to a given degree of accuracy.

8.2.2 Aliasing

The above discussion about using the Nyquist frequency to determine the sampling rate required to reconstruct a signal to a given degree of accuracy implicitly assumes that we know the power spectrum of $h(t)$ to begin with in order to determine how much power lies outside the Nyquist interval. In practice, we usually only have access to the samples of $h(t)$. From the samples alone, it is *in principle* impossible to determine how much power is outside the Nyquist interval. The reason for this is that the process of sampling moves frequencies which lie *outside* of the interval $[-f_c, f_c]$ into this interval via a process known as aliasing.

Aliasing occurs because two harmonics $\mathrm{e}^{2\pi i f_1 t}$ and $\mathrm{e}^{2\pi i f_2 t}$ give the same samples at the points $t_n = n\Delta$ if $f_1 - f_2$ is an integer multiple of Δ^{-1} (which is the width of the Nyquist interval). This can be seen by direct computation:

$$
\begin{aligned}
& \mathrm{e}^{2\pi i f_1 n\Delta} = \mathrm{e}^{2\pi i f_1 n\Delta} && \forall n \\
\Rightarrow & \mathrm{e}^{2\pi i (f_1 - f_2) n\Delta} = 1 && \forall n \\
\Rightarrow & \mathrm{e}^{2\pi i (f_1 - f_2)\Delta} = 1 && \\
\Rightarrow & (f_1 - f_2)\,\Delta = m && m \in \mathbb{Z} \\
\Rightarrow & (f_1 - f_2) = \frac{m}{\Delta} && m \in \mathbb{Z}
\end{aligned}
$$

Due to aliasing, it is impossible to tell the difference between the signals $h_1(t) = \mathrm{e}^{2\pi i f_1 t} + \mathrm{e}^{2\pi i f_2 t}$ and $h_2(t) = 2\,\mathrm{e}^{2\pi i f_1 t}$ from the samples alone. Aliasing means that under-sampled signals cannot be identified as such from looking at the sampled values. Care must therefore be taken in practice ensure that the sampling rate is sufficient to avoid aliasing errors.

8.2.3 Derivation of the Discrete Fourier Transform

Let us suppose we now have a finite number, N, of samples of the function $h(t)$: $\{h_n = h(t_n),\ n = 0, \ldots N-1\}$. For convenience we will assume that N is even. We now seek to use these values to estimate the integral in Eq. (8.1) at a set of discrete frequency points, f_n. What values should we choose for the f_n? We know from the Sampling Theorem that if the sampling rate is sufficiently high, the Nyquist interval contains the information necessary to reconstruct $h(t)$. If the sampling rate is too low, the power outside the Nyquist interval will be mapped back into the Nyquist interval by aliasing. In

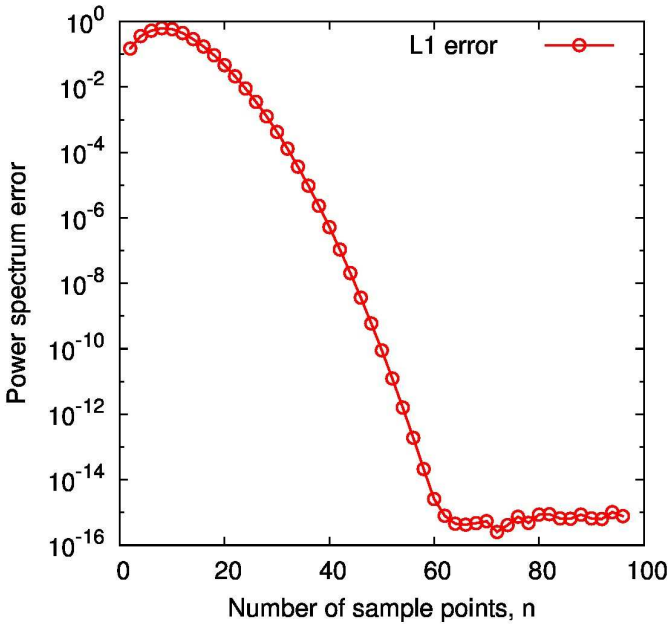

Figure 8.2: L1 error in the power spectrum of a Gaussian approximated using the Discrete Fourier Transform plotted as a function of the number of sample points in the domain [-5:5]. The fftw3 implementation of the Fast Fourier Transform was used.

either case, we should choose our frequency points to discretise the Nyquist interval. Let us do this uniformly by choosing $f_n = \frac{n}{\Delta N}$ for $n = -\frac{N}{2}, \ldots + \frac{N}{2}$. The extremal frequencies are clearly $\pm f_c$. Those paying close attention will notice that there are $N+1$ frequency points to be estimated from N samples. It will turn out that not all $N+1$ frequencies are independent. We will return to this point in a moment.

Let us now estimate the integral $H(f)$ in Eq. (8.1) at the discrete frequency points f_n by approximating the integral by a Riemann sum:

$$\begin{aligned} H(f_n) &\approx \sum_{k=0}^{N-1} h_k \, \mathrm{e}^{2\pi i f_n t_k} \Delta \\ &= \Delta \sum_{k=0}^{N-1} h_k \, \mathrm{e}^{2\pi i \left(\frac{n}{N\Delta}\right)(k\Delta)} \\ &= \Delta \sum_{k=0}^{N-1} h_k \, \mathrm{e}^{\frac{2\pi i n k}{N}} \\ &\equiv \Delta \, H_n, \end{aligned}$$

where the array of numbers, $\{H_n\}$, defined as

$$H_n = \sum_{k=0}^{N-1} h_k \, \mathrm{e}^{\frac{2\pi i n k}{N}} \tag{8.9}$$

is called the Discrete Fourier Transform (DFT) of the array $\{h_n\}$. The inverse of the transform is obtained by simply changing the sign of the exponent. Note that the DFT is simply a map from one array of complex numbers to another. All information about the time and frequency scales (including the sampling rate) have been lost. Thought of as a function of the index, n, the DFT is periodic with period N. That is

$$H_{n+N} = H_n.$$

We can see this by direct computation:

$$\begin{aligned} H_{n+N} &= \sum_{k=0}^{N-1} h_k \, \mathrm{e}^{\frac{2\pi i(n+N)k}{N}} \\ &= \sum_{k=0}^{N-1} h_k \, \mathrm{e}^{\frac{2\pi ink}{N}} \, \mathrm{e}^{2\pi ik} \\ &= \sum_{k=0}^{N-1} h_k \, \mathrm{e}^{\frac{2\pi ink}{N}} \\ &= H_n. \end{aligned}$$

We can therefore shift the index, n, used to define the frequencies f_n, by $\frac{N}{2}$ in order to obtain an index which runs over the range $n = 0 \ldots N$ rather than $n = -\frac{N}{2}, \ldots + \frac{N}{2}$. This is more convenient for a computer where arrays are usually indexed by positive numbers. Most implementations of the DFT adopt this convention. One consequence of this convention is that if we want to plot the DFT as a function of the frequencies (for example to compare against an analytic expression for the corresponding continuous Fourier transform) we need to obtain the negative frequencies from $H_{-n} = HN - n$. Another consequence of this periodicity is that $H_{-\frac{N}{2}} = H_{\frac{N}{2}}$ so $H_{\frac{N}{2}}$ corresponds to both frequencies $f = \pm f_c$ at either end of the Nyquist interval. Thus, as mentioned above, only N of the $N + 1$ values of n chosen to discretise the Nyquist interval are independent. THe DFT, Eq. (8.9), is therefore a map between arrays of length N which can be written as a matrix multiplication:

$$H_n = \sum_{k=0}^{N-1} W_{nk} \, h_k, \tag{8.10}$$

where W_{pq} are the elements of the $N \times N$ matrix

$$W_{pq} = \mathrm{e}^{\frac{2\pi ipq}{N}}.$$

The application of the DFT to a sampled version of the Gaussian function is shown in Fig. 8.1. The frequency domain plot on the right of Fig. 8.1 shows the power spectrum obtained from the output of the DFT plotted simply as a function of the array index (in red). Notice how the frequencies appear in the "wrong" order as a result of the shift described above. Notice also how the amplitude of the power spectrum obtained from the DFT is significantly larger than the analytic expression obtained for the continuous Fourier transform. This is due to the fact that the pre-factor of Δ in the approximation of $H(f)$ derived above is not included in the definition of the DFT. When both of these facts are taken into account by rescaling and re-ordering the output of the DFT, the green points are obtained which are a much closer approximation to the true power spectrum. The difference between the two is due to the low sampling rate. As the sampling rate is increased the approximation obtained from the DFT becomes better. This is illustrated in Fig. 8.2 which plots the error as a function of the number of sample points. Why do you think the error saturates at some point?

8.3 The Fast Fourier Transform algorithm

From Eq. (8.10), one might naively expect that the computation of the DFT is an $O(N^2)$ operation. In fact, due to structural regularities in the matrix W_{pq} it is possible to perform the computation in $O(N \log_2 N)$ operations using a divide-and-conquer algorithms which is known as the Fast Fourier Transform (FFT). In all that follows we will assume that the input array has length $N = 2^L$ elements. The FFT algorithm is based on the Danielson-Lanczos lemma which states that a DFT of length N can be written as a sum of 2 DFTs of length $N/2$. This can be seen by direct computation where the trick is to build the sub-transforms of lenght $N/$ from the elements of the original transform whose indices are odd/even

respectively:

$$\begin{aligned} H_k &= \sum_{j=0}^{N-1} h_j \, \mathrm{e}^{\frac{2\pi i n j}{N}} \\ &= \left[\sum_{j=0}^{\frac{N}{2}-1} h_{2j} \, \mathrm{e}^{\frac{2\pi i n (2j)}{N}} \right] + \left[\sum_{j=0}^{\frac{N}{2}-1} h_{2j+1} \, \mathrm{e}^{\frac{2\pi i n (2j+1)}{N}} \right] \\ &= \left[\sum_{j=0}^{\frac{N}{2}-1} h_j^{(e)} \, \mathrm{e}^{\frac{2\pi i n j}{N/2}} \right] + \mathrm{e}^{\frac{2\pi i k}{N}} \left[\sum_{j=0}^{\frac{N}{2}-1} h_j^{(o)} \, \mathrm{e}^{\frac{2\pi i n j}{N/2}} \right]. \end{aligned}$$

where $\left\{ h_j^{(o/e)}, j = 0, \ldots \frac{N}{2} - 1 \right\}$ are the arrays of length $\frac{N}{2}$ constructed from the odd/even elements of the original input array respectively. The last line is simply a linear combination of the DFTs of these two arrays:

$$H_k = H_k^{(e)} + \alpha_k \, H_k^{(o)}, \tag{8.11}$$

where $\alpha_k = \mathrm{e}^{\frac{2\pi i k}{N}}$. Note that since the index k on the LHS ranges over $k = 0, \ldots, N-1$, whereas the arrays on the RHS are only of length $N/2$, the periodicity of the DFT must be invoked to make sense of this formula. The FFT algorithm works by recognising that this can be applied recursively. After $L = \log_2 N$ subdivisions, we are left with arrays of length 1, each of which contains one of the elements of the original input array. Thus at the lowest level of the recursion, we just have a re-ordered copy of the original array. This is illustrated schematically in Fig. 8.3. The DFT of an array of length one is trivially just the array itself. The DFT of the original array can therefore be reassembled by going back up the tree linearly combining pairs of arrays of length n at each level to form arrays of length $2n$ at the next level up according to the rule given in Eq. (8.11 until we arrive back at the top level. At each level, there are clearly N operations required and there are $L = \log_2 N$ levels. Therefore the total complexity of this process is $O(N \, \log_2 N)$.

The key insight of the FFT algorithm is to construct the re-ordered array *first* and then go back up the tree. Notice how each position in the array at the bottom level can be indexed by a string of e's and o's depending on the sequence of right and left branches which are followed when descending the tree to reach that position. This sequence can be reversed to find out which element of the input array should be placed at which position at the bottom of the recursion tree. This is done using the bit-reversal rule. Starting with the string corresponding to a particular position at the bottom of the recursion tree we proceed as follows:

1. reverse the order of the string of e's and o's.
2. replace each e with 0 and each o with 1.
3. interpret the resulting string of zeros and ones as the binary representation of an integer.

The integer obtained in this way is the index of the element in the original input array which should be placed at this position at the bottom of the recursion tree. It is worth convincing oneself that this really works by checking some explicit examples in Fig. 8.3. This re-ordering can be done in $O(N)$ operations.

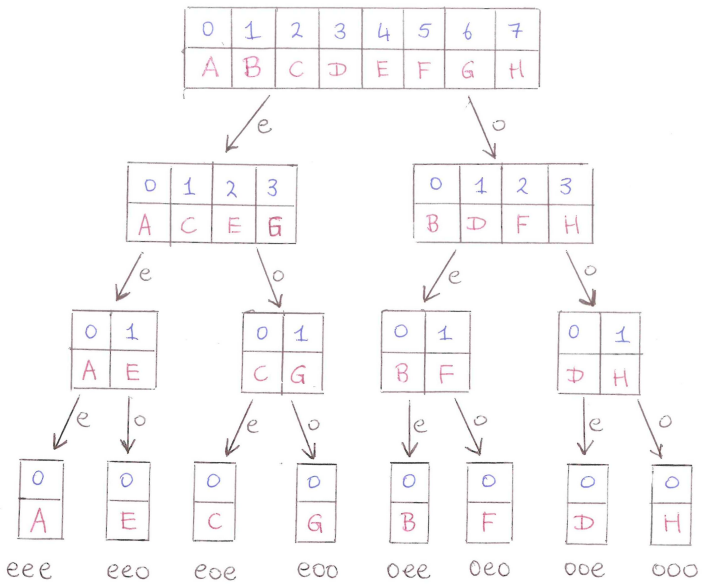

Figure 8.3: Graphical illustration of the recursive structure of the FFT algorithm for an array of length 8. At each level, each array is divided into two sub-arrays containing the elements with odd and even indices respectively. At the lowest level, the position of an element is indexed by a unique string of e's and o's. This string is determined by the sequence of right and left branches which are followed when descending the tree to reach that position. The bit reversal trick then identifies which element of the original array ends up in each position.