

Part 3

Discrete Fourier Transform

DFT Applications

Fast Fourier Transform

Chirp Z-Transform

Adaptive Filtering

Spectral Estimation

3.1 Discrete Fourier Transform (DFT)

- Direct DFT (p. 234)
- Inverse DFT (p. 236)
- DFT Properties (section 5.7, p. 259)
- Circular Convolution (section 5.4.2, p.246)

3.1.1 Direct DFT

Essential idea of DFT: Numerical Evaluation of Z-transform
(or DTFT) on unit circle:

- 1) Assume causal and truncate series
- 2) Sampled on unit circle (causes aliasing both in time and frequency)

$$\begin{aligned}G_k &= G(e^{j\frac{2\pi k}{N}}) \\ &= \sum_{n=0}^{N-1} g_n e^{-j\frac{2\pi kn}{N}}\end{aligned}$$

or

$$\begin{aligned}G_k &= G(W_N^k) \\ &= \sum_{n=0}^{N-1} g_n W_N^{-kn}\end{aligned}$$

where

$$W_N = e^{j\frac{2\pi}{N}}$$

Direct DFT (continued):

In matrix form: (pp. 237 – 238)

$$\begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \\ \vdots \\ G_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & \cdots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & \cdots & W_N^{-2(N-1)} \\ 1 & W_N^{-3} & \cdots & W_N^{-3(N-1)} \\ \vdots & \vdots & & \vdots \\ 1 & W_N^{-(N-1)} & \cdots & W_N^{-(N-1)^2} \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_{N-1} \end{bmatrix}$$

Note: $W_N^N = 1$

So we *define* the Discrete Fourier Transform of order N by

$$(G_k) = \mathcal{D}_N \{g_n\}$$

or, if there is only one value of N in question

$$(G_k) = \mathcal{D} \{g_n\}$$

where G_n and W_N are given by the above equations.

3.1.2 Inverse DFT (pp. 236, 238)

$$g_n = \frac{1}{N} \sum_{k=0}^{N-1} G_k e^{j \frac{2\pi kn}{N}}$$

or

$$g_n = \frac{1}{N} \sum_{k=0}^{N-1} G_k W_N^{kn}$$

In matrix form:

$$\begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & \cdots & W_N^{2(N-1)} \\ 1 & W_N^3 & \cdots & W_N^{3(N-1)} \\ \vdots & \vdots & & \vdots \\ 1 & W_N^{(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \\ \vdots \\ G_{N-1} \end{bmatrix}$$

Inverse DFT (continued):

Example: $N = 4$

$$W_N = j$$

$$\begin{pmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

and

$$\begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{pmatrix} \begin{pmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{pmatrix}$$

Inverse DFT (continued):

So if

$$(g_0, g_1, g_2, g_3) = (1, 1, 0, 0)$$

then

$$(G_0, G_1, G_2, G_3) = (2, 1 - j, 0, 1 + j)$$

3.1.3 Properties of DFT: (pp. 259 – 264)

- Linearity
- Periodicity in frequency domain: if we define

$$G_k = \sum_{n=0}^{N-1} g_n W_N^{-kn} \quad \text{for all } k$$

then G_k is periodic with period N , i.e.,

$$G_{k+N} = G_k \quad \text{for all } k$$

- Periodicity in time-domain: if

$$g_n = \frac{1}{N} \sum_{k=0}^{N-1} G_k W_N^{kn} \quad \text{for all } n$$

then g_n is periodic with period N , i.e.,

$$g_{n+N} = g_n \quad \text{for all } n$$

DFT Properties (Continued):

- Shifting:
 - *Circular* Shifting: shifting as an infinite periodic sequence of period N
 - $\mathcal{D}_N\{g_{n-l}\} = (W_N^{-kl} G_k)$
- Convolution: If (f_n) and (g_n) are sequences of length N , then

$$\mathcal{D}_N\{(f_n) \otimes (g_n)\} = (F_k G_k)$$

where \otimes denotes *circular convolution*.

3.1.4 Circular Convolution: (section 5.4.2, pp. 246 – 250)

- defined only for periodic sequences of the same period N
- defined by $(f_n) \otimes (g_n) = (k_n)$, where

$$k_n = \sum_{l=0}^{N-1} f_l g_{n-l}$$

- k_n is again a periodic sequence of period N
- the sum can be over any interval of length N

Circular Convolution (continued):

Linear and Circular Convolution:

- Linear Convolution
 - Ordinary discrete-time convolution
 - Z-transform converts to multiplication

- Circular Convolution:
 - Convolution as *periodic* sequences
 - DFT converts to multiplication

Circular Convolution (continued):

Comparison of Linear and Circular Convolution: (section 5.10.1, pp. 268 – 269)

Example: $N = 4$

Suppose the sequences are

$$(f_0, f_1, f_2, f_3)$$

$$(g_0, g_1, g_2, g_3)$$

and we find $(k_n) = (f_n) \otimes (g_n)$ (by interpreting the signals as periodic with period 4).

Circular Convolution (continued):

Then, for $(k_n) = (f_n) \otimes (g_n)$ we get:

$$k_0 = f_0g_0 + f_1g_3 + f_2g_2 + f_3g_1$$

$$k_1 = f_0g_1 + f_1g_0 + f_2g_3 + f_3g_2$$

$$k_2 = f_0g_2 + f_1g_1 + f_2g_0 + f_3g_3$$

$$k_3 = f_0g_3 + f_1g_2 + f_2g_1 + f_3g_0$$

while if we find $(k_n) = (f_n) * (g_n)$, (by interpreting the signals as being zero outside the range 0-3), we get:

$$k_0 = f_0g_0$$

$$k_1 = f_0g_1 + f_1g_0$$

$$k_2 = f_0g_2 + f_1g_1 + f_2g_0$$

$$k_3 = f_0g_3 + f_1g_2 + f_2g_1 + f_3g_0$$

$$k_4 = f_1g_3 + f_2g_2 + f_3g_1$$

$$k_5 = f_2g_3 + f_3g_2$$

$$k_6 = f_3g_3$$

Circular Convolution (continued):

However, if we take the signals to be of period 8 (by padding with zeros)

$$(f_0, f_1, f_2, f_3, 0, 0, 0, 0)$$

$$(g_0, g_1, g_2, g_3, 0, 0, 0, 0)$$

and now find $(k_n) = (f_n) \otimes (g_n)$, we get:

$$k_0 = f_0 g_0$$

$$k_1 = f_0 g_1 + f_1 g_0$$

$$k_2 = f_0 g_2 + f_1 g_1 + f_2 g_0$$

$$k_3 = f_0 g_3 + f_1 g_2 + f_2 g_1 + f_3 g_0$$

$$k_4 = f_1 g_3 + f_2 g_2 + f_3 g_1$$

$$k_5 = f_2 g_3 + f_3 g_2$$

$$k_6 = f_3 g_3$$

$$k_7 = 0$$

which is the same as the output from the linear convolution.

Circular Convolution (continued):

Therefore the DFT can be used to calculate linear convolution, *provided* that the signals are padded with zeros so that the sum of the lengths of the original signals is $\leq N$.

The convolution involves 3 DFT steps:

- Find the DFT of each signal ($O(N \log N)$ multiplications each)
- Multiply the transformed signals together ($O(N)$ multiplications),
- Find the inverse DFT ($O(N \log N)$ multiplications)

Linear Convolution: Figure 5.8(b), p. 248

Circular Convolution: Figure 5.8(a), p. 248

Circular Convolution with Zero-Padding: Figure 5.14(a),
p. 269

3.2 Applications of DFT:

- filtering: (section 5.10.2, pp. 269 – 274)
 - acquire M signal points
 - pad with zeros
 - use DFT to convolve with padded impulse response
 - add overlaps
 - output

Applications of DFT: (Continued)

- noncausal filtering (convolution)
 - acquire full signal
 - pad with zeros
 - use DFT to convolve with padded impulse response
- DTMF detection: p. 855
- spectral analysis: find frequency content of deterministic signal (stationary, p. 856, non-stationary (STFT), p. 864)
- calculation of system performance: calculate system performance in frequency domain; use inverse DFT to find time-domain performance. (Dangerous!)

3.2.1 Interpreting the DFT:

Several Problems:

- Relationship of numbers to real world
 - ΔT : Given by sampling rate
 - Δf :

$$\begin{aligned}\Delta\theta &= 2\pi/N \\ \Rightarrow \Delta\omega &= (1/T)\Delta\theta \\ &= 2\pi/NT \\ \Rightarrow \Delta f &= \frac{1}{NT} \\ &= \frac{1}{N}f_S\end{aligned}$$

- When finding an analog impulse response, must multiply by $1/T$.

Interpreting the DFT (Continued)

- Relationship of order to real world
 - In the frequency domain, the second half of the sequence $(G_0, G_1, \dots, G_{N-1})$ represents *negative* frequency (aliasing in frequency).
 - In the time domain, the second half of the sequence $(g_0, g_1, \dots, g_{N-1})$ represents *negative* time (aliasing in time). Specifically, g_{N-1} represents the time-instant $-T$, g_{N-2} represents $-2T$, etc..

Interpreting the DFT (Continued)

- DFT will not give correct response for causal, unstable systems. (The DFT approximates the DTFT, rather than the Z-transform.)
- Periodic signals – must *not* pad with zeros
- Extreme care is needed when dealing with nonlinear operations, even in linear systems. (May need to pad with many zeros.)

3.3 Fast Fourier Transform: Sec. 11.3

A fast algorithm ($< O(N^2)$) for calculating the DFT; usually $O(N \log N)$ or $O(N)$ operations.

Numerous algorithms.

Simplest: Cooley-Tukey power of 2, sec. 11.3.2.

Basic Calculation:

Suppose $N = 2M$. Then, for $0 \leq k \leq N - 1$,

$$\begin{aligned}
 X_k &= x_0 + x_1 W_N^{-k} + x_2 W_N^{-2k} + x_3 W_N^{-3k} \\
 &\quad + \cdots + x_{N-1} W_N^{-(N-1)k} \\
 &= x_0 + x_2 W_N^{-2k} + \cdots + x_{N-2} W_N^{-(N-2)k} \\
 &\quad + x_1 W_N^{-k} + x_3 W_N^{-3k} + \cdots + x_{N-1} W_N^{-(N-1)k} \\
 &= x_0 + x_2 W_M^{-k} + \cdots + x_{N-2} W_M^{-(M-1)k} \\
 &\quad + W_N^{-k} \left[x_1 + x_3 W_M^{-k} + \cdots + x_{N-1} W_M^{-(M-1)k} \right]
 \end{aligned}$$

FFT (continued):

The last step holds because $W_N^2 = W_M$ (since $N = 2M$).

Now if we define the sequences (x_{En}) and (x_{On}) (both of length M) by

$$\begin{aligned}(x_{E0}, x_{E1}, \dots, x_{E(M-1)}) &= (x_0, x_2, \dots, x_{N-2}) \\(x_{O0}, x_{O1}, \dots, x_{O(M-1)}) &= (x_1, x_3, \dots, x_{N-1})\end{aligned}$$

and let

$$\begin{aligned}(X_{Ek}) &= \mathcal{D}_M(x_{En}) \\(X_{Ok}) &= \mathcal{D}_M(x_{On})\end{aligned}$$

then, for $0 \leq k \leq M - 1$,

$$X_k = X_{Ek} + W_N^{-k} X_{Ok}$$

and, by the periodicity of the DFT

$$\begin{aligned}X_{k+M} &= X_{Ek} + W_N^{-(k+M)} X_{Ok} \\&= X_{Ek} - W_N^{-k} X_{Ok}\end{aligned}$$

for $0 \leq k \leq M - 1$.

FFT (continued):

Therefore, X_k can be found for $0 \leq k \leq N - 1$ by the following two equations

$$\begin{aligned}X_k &= X_{Ek} + W_N^{-k} X_{Ok} \\X_{k+M} &= X_{Ek} - W_N^{-k} X_{Ok}\end{aligned}$$

for $0 \leq k \leq M - 1$.

Therefore, a DFT of order $2M$ can be calculated by using *two* (rather than four) DFT's of order M , M complex multiplications, and $2M$ complex additions.

If M is even, this idea can be applied again; and if M is a power of 2, it can be used recursively to calculate the entire transform.

FFT (continued):

Signal flow graphs:

For one value of k :

$$\begin{aligned}X_k &= X_{Ek} + W_N^{-k} X_{Ok} \\X_{k+M} &= X_{Ek} - W_N^{-k} X_{Ok}\end{aligned}$$

we get the basic 'butterfly': figures 11.22, 11.23, pp. 614 – 615.

One full stage ($N = 8$): figure 11.17, p.612.

Full Signal Flow-Graph ($N = 8$): figure 11.24, p. 615.

Remaining Problem: Order of input values:

Bit-Reversed Addressing (p. 616)

3.3.1 Goertzel's Algorithm:

Alternative if only some frequency points are needed
(section 11.3.1, pp. 607 – 610):

Since

$$X_k = \sum_{l=0}^{N-1} x_l W_N^{-kl} = \sum_{l=0}^{N-1} x_l W_N^{k(N-l)}$$

which is a convolution, the DFT at each point k can be found as the output of a first-order recursive (complex) filter whose impulse response is $h_n = W_N^{kn}$.

Starting with zero initial conditions ($y_k(-1) = 0$; $x(N) = 0$), we get

$$y_k(n) = x(n) + W_N^k y_k(n-1) \quad 0 \leq n \leq N$$

$$X(k) = y_k(N)$$

where $X(k)$ is the k -th DFT coefficient, and N is the length of the DFT.

If only the amplitude of $X(k)$ is needed, the computation can be done with real numbers. (p. 610)

3.4 Chirp Z-Transform:

Calculates Z-transform on any set of points of the form:

$$z_k = az_0^k$$

where a and z_0 are complex numbers, and

$$0 \leq k \leq M - 1.$$

This corresponds to a uniformly sampled line segment in the s -plane.

It can be used to evaluate the Z-transform on a segment of the unit circle (i.e., to find the frequency response over a limited range), or on a segment of the real axis (i.e., to evaluate the Laplace transform).

Chirp Z-Transform (continued):

Assuming that the signal is finite ($0 \leq n \leq N - 1$), we want to find

$$\begin{aligned}
 G_k &= G(z_k) \\
 &= G(az_0^k) \\
 &= \sum_{n=0}^{N-1} g_n (az_0^k)^{-n} \\
 &= \sum_{n=0}^{N-1} g_n a^{-n} z_0^{-kn}
 \end{aligned}$$

Now let $W = z_0^{-1/2}$ and use the fact that

$$-nk = \frac{1}{2} [(k-n)^2 - n^2 - k^2]$$

to get

$$\begin{aligned}
 G_k &= \sum_{n=0}^{N-1} g_n a^{-n} W^{n^2} W^{k^2} W^{-(k-n)^2} \\
 &= W^{k^2} \sum_{n=0}^{N-1} [g_n a^{-n} W^{n^2}] W^{-(k-n)^2}
 \end{aligned}$$

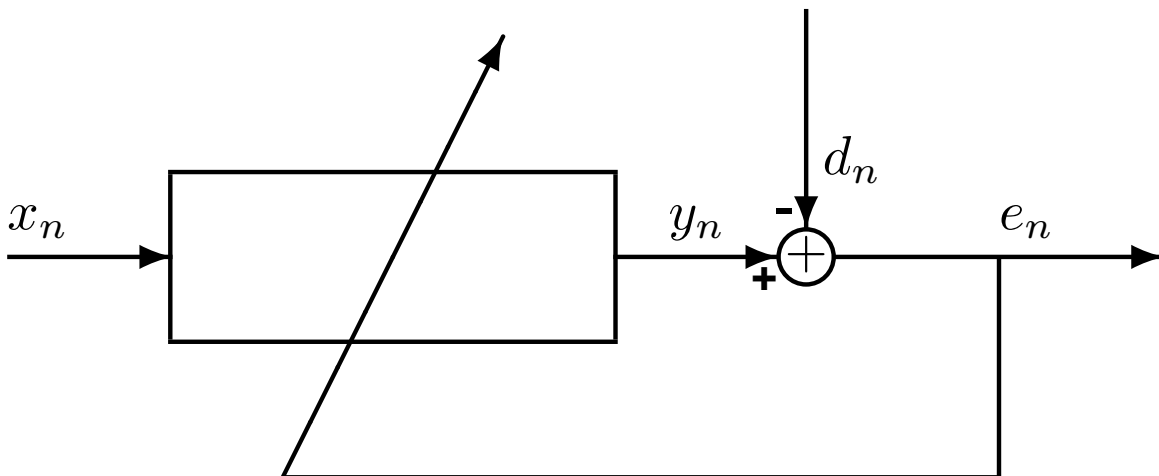
Chirp Z-Transform (continued):

Therefore the numbers G'_k can be evaluated by

- multiplying the original signal by $a^{-n}W^{n^2}$ ($O(N)$ operations)
- convolving with $W^{-(k-n)^2}$ (Two DFT calculations, and $O(2N + M)$ operations).
- multiplying the result by W^{k^2} ($O(K)$ operations)

3.5 Adaptive Filtering:

Basic system:



3.5.1 LMS Algorithm:

Approach: Least Squares

Assume that the filter is a transversal FIR filter with $N + 1$ taps (w_j).

Then the problem becomes: Minimize

$$\begin{aligned} J_L &= \sum_{k=0}^L e_k^2 \\ &= \sum_{k=0}^L (y_k - d_k)^2 \\ &= \sum_{k=0}^L \left(d_k - \sum_{j=0}^N w_j x_{k-j} \right)^2 \end{aligned}$$

LMS Algorithm (continued):

In vector terms:

Let \mathbf{e} denote the error vector (e_0, e_1, \dots, e_L) , \mathbf{w} denote the weight vector (w_0, w_1, \dots, w_N) , and \mathbf{X} denote the data matrix

$$\begin{pmatrix} x_0 & 0 & \dots & 0 \\ x_1 & x_0 & \dots & 0 \\ \vdots & \vdots & & \\ x_N & x_{N-1} & \dots & x_0 \\ x_{N+1} & x_N & \dots & x_1 \\ \vdots & \vdots & & \\ x_L & x_{L-1} & \dots & x_{L-N} \end{pmatrix}$$

LMS Algorithm (continued):

Then

$$\begin{aligned} J_L &= \mathbf{e}^T \mathbf{e} \\ &= (\mathbf{d} - \mathbf{X}\mathbf{w})^T (\mathbf{d} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{d}^T \mathbf{d} - 2\mathbf{d}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} \\ &= \mathbf{d}^T \mathbf{d} - 2\mathbf{p}^T \mathbf{w} + \mathbf{w}^T \mathbf{R}\mathbf{w} \end{aligned}$$

where

$$\mathbf{R} = \mathbf{X}^T \mathbf{X}$$

is the *data covariance matrix* and

$$\mathbf{p}^T = \mathbf{d}^T \mathbf{X}$$

is the *cross covariance vector*.

The *gradient* is then given by

$$\frac{\partial J}{\partial \mathbf{w}} = \nabla = 2(\mathbf{R}\mathbf{w} - \mathbf{p})$$

LMS Algorithm (continued):

Minimum:

$$\nabla = 2(\mathbf{R}\mathbf{w} - \mathbf{p}) = 0$$

or

$$\mathbf{w}^* = \mathbf{R}^{-1}\mathbf{p}$$

This is not adaptive, and there are two problems with it:

- \mathbf{R} is at best an estimate
- inversion is both costly and unstable

LMS Algorithm (continued):

For an adaptive algorithm, rewrite \mathbf{w}^* in terms of the gradient ∇ :

Since $\nabla = 2(\mathbf{R}\mathbf{w} - \mathbf{p})$, we can solve to get

$$\mathbf{R}^{-1}\mathbf{p} = \mathbf{w} - \frac{1}{2}\mathbf{R}^{-1}\nabla$$

and so

$$\mathbf{w}^* = \mathbf{w} - \frac{1}{2}\mathbf{R}^{-1}\nabla$$

This is still a non-adaptive, single-step formula. To make it adaptive, we could write:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu\mathbf{R}^{-1}\nabla_k$$

but this still leaves the problem with \mathbf{R}^{-1} .

LMS Algorithm (continued):

To avoid this, we use the *method of steepest descent*, which takes a step in the direction opposite to the gradient:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla_k$$

In order not to overshoot, we must make sure that our step-size is at most the smallest that \mathbf{R}^{-1} would give:

$$\mu < 1/\lambda_{MAX} \leq N/\text{Trace}(\mathbf{R}) = 1/\sigma^2$$

The only remaining problem is that ∇_k is still dependent on \mathbf{R} . We obtain the *LMS adaptive filtering algorithm* by replacing ∇ by an instantaneous estimate of ∇ .

LMS Algorithm (continued):

Instead of taking the gradient of the total error $\mathbf{e}^T \mathbf{e}$, we use the gradient of the present error e_k^2 :

$$\begin{aligned}\nabla_k &= \frac{\partial}{\partial \mathbf{w}} e_k^2 \\ &= 2e_k \frac{\partial}{\partial \mathbf{w}} (d_k - \mathbf{w}_k^T \mathbf{x}_k) \\ &= -2e_k \mathbf{x}_k\end{aligned}$$

where $\mathbf{x}_k^T = (x_k, x_{k-1}, \dots, x_{k-N})$ denotes the current data vector.

LMS Algorithm (continued):

Using this in place of the gradient in the steepest descent algorithm, we get the final form of the LMS algorithm:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \frac{2u}{\sigma^2} e_k \mathbf{X}_k$$

where $0 < u < 1$ and σ^2 is the average input power. This can be estimated by using a 'fading memory update'

$$\hat{\sigma}_k^2 = \alpha x_k^2 + (1 - \alpha) \hat{\sigma}_{k-1}^2$$

3.6 Spectral Estimation:

The FFT can be applied to the problem of determining the frequency content of a deterministic signal. When the signal is random, the problem is somewhat different: it is the spectrum which is to be estimated.

We will assume that we have a zero-mean, stationary, Gaussian stochastic process (x_n) . Such a process is characterized by its *spectrum*:

$$P_x(\omega) = \sum_{k=-\infty}^{\infty} r_x(k) e^{-j\omega k}$$

where $r_x(k)$ are the *autocorrelation coefficients* of the process:

$$r_x(k) = E[x(n)x(n+k)]$$

Equivalently:

$$P_x(\omega) = \lim_{M \rightarrow \infty} E \left[\frac{1}{2M+1} \left| \sum_{n=-M}^M x(n) e^{-j\omega n} \right|^2 \right]$$

The simplest spectral estimator is the *periodogram*, and is derived from the above

$$\hat{P}(\omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-j\omega n} \right|^2$$

Problem: it is an inconsistent estimator; the variance (noise) doesn't decrease as we take more points!

3.6.1 Averaged Periodogram:

Solution:

Use the *averaged periodogram*, defined by splitting the data into K non-overlapping pieces of length L each, and taking the average of the resulting periodogram.

The variance decreases as $1/K$, but the estimator is *biased*: its expected value is not correct. It smears the actual spectrum by applying a Bartlett window to it; the bias decreases as L increases.

The result is that we need both K and L large, i.e., a lot of data.

3.6.2 Other Nonparametric Spectral Estimators:

Blackman-Tukey Estimator:

Estimate the autocorrelation coefficients, $r_x(k)$, of the process by

$$\hat{r}_x(k) = \frac{1}{N} \sum_{n=0}^{N-1-|k|} x(n)x(n+|k|)$$

and then let

$$\hat{P}_x(\omega) = \sum_{k=-(N-1)}^{N-1} w_M(k) \hat{r}_x(k) e^{-j\omega k}$$

where $w_M(k)$ is a window function of order $M \leq N - 1$.

A rectangular window on the data gives a triangular window on the autocorrelation sequence, and is equivalent to the periodogram.

In this application, the transform of the autocorrelation window should be non-negative, to avoid negative power.

Nonparametric Spectral Estimators (continued)

Capon Estimator:

$$\hat{P}_x(\omega) = \frac{p}{\mathbf{e}^* \mathbf{R}^{-1} \mathbf{e}}$$

where \mathbf{R} is the p by p autocorrelation matrix whose (i, j) element is r_{i-j} ,

$$\mathbf{e} = (1, e^{j\omega}, e^{j2\omega}, \dots, e^{j(p-1)\omega})^T$$

and $*$ denotes conjugate transpose.

In practice, the autocorrelation coefficients must be estimated, and p should be much less than N to yield good estimates.

3.6.3 Parametric Spectral Estimators:

Assume a model: performance is better than nonparametric if model is correct, and worse if model is incorrect.

The process $x(n)$ is normally modelled as the output of a filter with transfer function

$$\begin{aligned} H(z) &= \frac{B(z)}{A(z)} \\ &= \frac{1 + \sum_{k=1}^q b_k z^{-k}}{1 + \sum_{k=1}^p a_k z^{-k}} \end{aligned}$$

driven by white noise with variance σ^2 . Then

$$P_x(\omega) = \sigma^2 |H(e^{j\omega})|^2$$

and so the spectral estimation problem becomes one of finding σ , a_k , and b_k .

Parametric Spectral Estimators (continued):

This is called an *autoregressive moving-average* (ARMA(p, q)) model.

There are two important special cases:

- autoregressive (AR(p)): $b_k = 0$ for all k
- moving average (MA(q)): $a_k = 0$ for all k

Note: the full ARMA model is *much* more difficult to handle than either the AR or MA cases.

3.6.4 AR Spectral Estimator:

A maximum-likelihood criterion applied to the AR(p) model yields a spectral estimator whose coefficients are found by solving the equation:

$$\mathbf{C}\mathbf{a} = \mathbf{c}$$

where

$$\mathbf{a} = (a_1, a_2, \dots, a_p)^T$$

$$\mathbf{c} = (c_{10}, c_{20}, \dots, c_{p0})^T$$

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ c_{p1} & c_{p2} & \dots & c_{pp} \end{pmatrix}$$

and

$$c_{jk} = \frac{1}{N-p} \sum_{n=p}^{N-1} x(n-j)x(n-k)$$

σ^2 is given by

$$\sigma^2 = c_{00} + \sum_{k=1}^p a_k c_{0k}$$