Frequency-Response Identification

3.1: Transfer operator and frequency function

- The next approaches that we look at require understanding system performance in the frequency domain.
- This requires that we first define transfer functions, and frequency response functions.

The transfer operator ("transfer function")

- We introduce a linear operator "q" that performs a "forward shift." That is, $qu[k] = u[k+1]$.
- Similarly, the backward shift operator q^{-1} gives $q^{-1}u[k] = u[k-1]$.
- Can re-write a system's response as

$$y[k] = \sum_{m=0}^{\infty} g[m]u[k-m] = \sum_{m=0}^{\infty} g[m]\left(q^{-m}u[k]\right) = \left(\sum_{m=0}^{\infty} g[m]q^{-m}\right)u[k]$$
$$\triangleq G(q)u[k],$$

where the system transfer operator $G(q)$ is defined as

$$G(q) = \sum_{m=0}^{\infty} g[m]q^{-m}.$$

- Note: This is conceptually different from, but mathematically identical to the z-domain transfer function: $G(q) = G(z)|_{z=q}$.

- Strictly speaking, $G(q)$ is a transfer operator and $G(z)$ is a "transfer function," although $G(q)$ is often called a transfer function as well.
- The operator q denotes a forward shift only; the complex variable z brings in a whole bunch of system theory.
- Since q is a time *operator*, it is entirely proper to write things like $y[k] = G(q)u[k]$.
- But, since z is a complex *variable* and not a time operator, it is *not* proper to write things like $y[k] = G(z)u[k]$; however, sometimes people use this short-hand also.
- Since a system is entirely specified by its unit-pulse response, we sometimes refer to the system as "the filter $G(q)$."

- If we model disturbance as filtered white noise, we have

$$H(q) = \sum_{k=0}^{\infty} h[k]q^{-k}$$
$$v[k] = H(q)e[k],$$

  and

$$y[k] = G(q)u[k] + H(q)e[k].$$

- Note that a filter $H(q)$ is monic if its zeroth coefficient is 1.

Frequency function of a system

- LTI systems have complicated response to general inputs (via convolution), but quite simple responses to sinusoidal inputs.
- Suppose that the input to a system is a sinusoid $u[k] = \cos(\omega k)$.
- It is convenient to write this as $u[k] = \mathbb{R}[e^{j\omega k}]$.

- Then, the system output is given by

$$y[k] = \sum_{m=0}^{\infty} g[m]\mathbb{R}\left[e^{j\omega(k-m)}\right] \qquad = \mathbb{R}\left[\sum_{m=0}^{\infty} g[m]e^{j\omega(k-m)}\right]$$
$$= \mathbb{R}\left[e^{j\omega k}\sum_{m=0}^{\infty} g[m]e^{-j\omega m}\right] = \mathbb{R}\left[e^{j\omega k}G(e^{j\omega})\right]$$
$$= \left|G(e^{j\omega})\right|\cos\left(\omega k + \arg G(e^{j\omega})\right),$$

  where we have defined

$$G(e^{j\omega}) = \sum_{m=0}^{\infty} g[m]e^{-j\omega m} = G(z)|_{z=e^{j\omega}} = G(q)|_{q=e^{j\omega}}.$$

- Note that this analysis assumed that the input was a cosine from time $-\infty$. If the cosine is applied at time zero, we obtain an additional term

$$y[k] = \left|G(e^{j\omega})\right|\cos\left(\omega k + \arg G(e^{j\omega})\right) \underbrace{-\mathbb{R}\left[e^{j\omega k}\sum_{m=k}^{\infty} g[m]e^{-j\omega m}\right]}_{\text{dominated by } \sum_{m=k}^{\infty}|g[m]|}.$$

- We note that a system is stable if $\sum_{m=0}^{\infty}|g[m]| < \infty$; so, if the system is stable, this additional term is transient (tends to zero as $k \to \infty$).
- Regardless, when the input is a sinusoid, the steady-state output is a sinusoid of the same frequency, amplified by factor $|G(e^{j\omega})|$ and phase shifted by $\arg G(e^{j\omega})$.
- Therefore, the complex value $G(e^{j\omega}) = G(z)|_{z=e^{j\omega}}$ gives full information as to what happens in stationarity when the input is a sinusoid of frequency ω.
- For that reason, $G(e^{j\omega})$ is called the frequency function of the system. It is also the discrete-time Fourier transform (DTFT) of $g[k]$.

3.2: Direct identification of frequency function

- The frequency function contains the same information as the unit-pulse response, in different form.
  - The unit-pulse response can be recovered if $G(e^{j\omega})$ is known for all frequencies,
$$g[k] = \frac{1}{2\pi}\int_{-\pi}^{\pi} G(e^{j\omega})e^{j\omega k}\,\mathrm{d}\omega.$$
  - Also, some control design techniques require only $G(e^{j\omega})$ and not $g[k]$, so there is benefit to knowing how to estimate $G(e^{j\omega})$ directly.
- Let the input to a system be $u[k] = \alpha\cos(\omega k)$. Then, the output is
$$y[k] = \alpha\left|G(e^{j\omega})\right|\cos\left(\omega k + \arg G(e^{j\omega})\right) + v[k] + \text{transient}.$$
- This leads to a simple procedure, known as frequency analysis
  - Excite the system with a sinusoidal input at frequency ω_0; wait for transient to die out; collect N time samples.
  - Measure (graphically) the amplification factor and phase shift at that frequency, yielding $\widehat{G}_N(e^{j\omega_0})$.
  - Repeat for all frequencies of interest.

ISSUE: The frequency function is continuous in the frequency domain.

- This means that for a complete identification of the function, all frequencies from 0 to ∞ must be tested.
- In a practical implementation, we would need to select only a few frequencies and be willing to interpolate between the measured data.

ISSUE: We probably want to collect data over at least several wavelengths of the input frequency.

- For very low frequency inputs, this can take hours or days.
- *e.g.*, "electrochemical impedance spectroscopy" for battery cells often explores frequencies down to the sub-mHz level.
  - A single input frequency can take up to around three hours to produce a result.

ISSUE: With the presence of noise, $v[k]$, it may be difficult to accurately determine amplification and phase-shift factors.

- We can use correlations to average out the noise. Define

$$I_c(N) = \frac{1}{N}\sum_{k=1}^{N} y[k]\cos(\omega k) \quad \text{and} \quad I_s(N) = \frac{1}{N}\sum_{k=1}^{N} y[k]\sin(\omega k).$$

- Insert the formula for $y[k]$, ignoring the transient term (can wait until it dies out before collecting data),

$$\begin{aligned}
I_c(N) &= \frac{1}{N}\sum_{k=1}^{N}\left(\alpha\left|G(e^{j\omega})\right|\cos\left(\omega k + \arg G(e^{j\omega})\right) + v[k]\right)\cos(\omega k) \\
&= \frac{1}{N}\sum_{k=1}^{N}\left(\alpha\left|G(e^{j\omega})\right|\cos\left(\omega k + \arg G(e^{j\omega})\right)\right)\cos(\omega k) \\
&\qquad + \frac{1}{N}\sum_{k=1}^{N} v[k]\cos(\omega k) \\
&= \frac{\alpha}{2}\left|G(e^{j\omega})\right|\cos\left(\arg G(e^{j\omega})\right) \\
&\qquad + \frac{\alpha}{2N}\left|G(e^{j\omega})\right|\sum_{k=1}^{N}\cos\left(2\omega k + \arg G(e^{j\omega})\right) \\
&\qquad + \frac{1}{N}\sum_{k=1}^{N} v[k]\cos(\omega k),
\end{aligned}$$

noting that $\cos(a)\cos(b) = \frac{1}{2}(\cos(a+b) + \cos(a-b))$.

- The second term tends to zero as $N \to \infty$. The third term also tends to zero unless $v[k]$ has a pure periodic frequency component at frequency ω.
- Similarly, we can find that

$$I_s(N) = -\frac{\alpha}{2}\left|G(e^{j\omega})\right| \sin\left(\arg G(e^{j\omega})\right) + \frac{\alpha}{2N}\left|G(e^{j\omega})\right| \sum_{k=1}^{N} \sin\left(2\omega k + \arg G(e^{j\omega})\right) + \frac{1}{N}\sum_{k=1}^{N} v[k]\sin(\omega k),$$

noting that $\cos(a)\sin(b) = \frac{1}{2}(\sin(a+b) - \sin(a-b))$.

- By combining $I_c(N)$ and $I_s(N)$, we can come up with a frequency-response estimate

$$\left|\widehat{G}_N(e^{j\omega})\right| = \frac{\sqrt{I_c^2(N) + I_s^2(N)}}{\alpha/2}$$

$$\arg \widehat{G}_N(e^{j\omega}) = -\operatorname{atan}\left(I_s(N)/I_c(N)\right).$$

- This is known as the frequency analysis by correlation method.

3.3: Identification of frequency function via Fourier analysis

- An advantage of the direct frequency function identification approach is that a Bode plot can be obtained easily, and one can concentrate effort on an “interesting” frequency range.
- A disadvantage is that many industrial processes do not admit sinusoidal inputs in normal operation. Further, the experiment must be repeated for a number of frequencies, which may lead to long experimentation periods.
- Therefore, we would like to extend the direct frequency-function identification approach to admit inputs that are not sinusoidal, but which produce a frequency response estimate of the system.
- We will soon look at the “ETFE” method to do this, but first must clarify between two different frequency transforms.

DTFT versus DFT: An issue of clarity. . .

- Two different kinds of Fourier transform may be used to operate on discrete-time signals.
- Both transformations take discrete-time data and compute a frequency-domain interpretation of that data.
- But they are significantly different in some respects: It is important to be able to distinguish between them.
- The first is the discrete-time Fourier transform (DTFT), which we have already seen

$$H(e^{j\omega}) = \sum_{m=0}^{\infty} h[m]e^{-j\omega m}.$$

- The second is the discrete Fourier transform (DFT),[1]

$$H_N(\omega) = \frac{1}{\sqrt{N}} \sum_{m=1}^{N} h[m] e^{-j\omega m}.$$

- The DTFT requires knowing the entire infinite-length data sequence, and produces a frequency function valid at all frequencies.
- The DFT requires only a finite-length data sequence.
  - An implicit assumption is that the time data is periodic with period N, which is not generally true in practice.
  - However, with some careful attention to details, we can often work around (and nullify) this assumption.
  - The DFT then is much more valuable in practice: It is computable.

IMPORTANT POINT: The DFT is computed at specific frequencies only: $\omega = 2\pi n/N$ for $n = 1, \ldots, N$.

- It is not valid at all values of ω as is the DTFT.

- The DFT is periodic in frequency, with period N frequency points, or 2π in frequency space. Therefore,
  - We interpret DFT values near index $n = 1$ to be low frequencies,
  - We interpret DFT values near index $n = \dfrac{N}{2}$ to be high frequencies,
  - We interpret values of the DFT near index $n = N$ to be low negative frequencies.

[1] Note that this definition is not standard—the summation over the N data points usually starts with index zero. However, the extra phase contribution due to calculating it this way will cancel out in an application as we always either (1) take the magnitude only, or (2) divide by a similar quantity, with similar extra phase contribution.

- If N is a power of 2, there exists an exceptionally efficient *algorithm* to compute the DFT, called the FFT.
- In MATLAB, `fft.m` computes the DFT; `fftshift.m` performs a circular shift on the results for easier plotting against a more standard frequency axis.

MAIN POINT AGAIN: We cannot assign arbitrary frequency values to the output of the DFT. They are specified as $\omega = 2\pi n/N$.

- Similarly, since the ETFE is based on the DFT, we cannot assign arbitrary frequency values to its output. They are also specified by the underlying math.

3.4: Empirical transfer function estimates

- We wish to extend the direct frequency-function identification approach to admit inputs that are not sinusoidal.
- The most basic approach is called the ETFE, "empirical transfer function estimate" (but I think it's poorly named since it estimates a frequency response and not a transfer function).
  - Method is termed "empirical" since no assumptions are made regarding system besides LTI.
- Define the function $U_N(\omega)$ based on signal $u[k]$ to be

$$U_N(\omega) = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} u[k] e^{-j\omega k}.$$

  - Values obtained at $\omega = 2\pi n/N$, for $n = 1, \ldots, N$ are the standard discrete Fourier transform (DFT) for a finite sequence of data.
  - Note that we can invert this relationship via

$$u[k] = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} U_N(2\pi n/N) e^{j 2\pi k n/N}.$$

  - $U_N(\omega)$ is periodic with period 2π: $U_N(\omega + 2\pi) = U_N(\omega)$.
  - Also, since $u[k]$ is real, $U_N(-\omega) = \overline{U_N(\omega)}$.
- $U_N(\omega)$ represents the signal $u[k]$ as a linear combination of complex sinusoids $e^{j\omega k}$ for N different frequencies.
  - It tells us the "weight" that any specific frequency carries in the decomposition of $u[k]$.
  - Its absolute square value $|U_N(2\pi n/N)|^2$ is therefore a measure of the contribution of this frequency to the "signal power."

DEFINITION: The value $|U_N(\omega)|^2$ is known as the periodogram of the signal $u[k]$, $k = 1, 2, \ldots, N$.

- Parseval's relationship, which relates power in the time and frequency domains as,

$$\sum_{n=1}^{N} |U_N(2\pi n/N)|^2 = \sum_{k=1}^{N} u^2[k],$$

  reinforces the interpretation that the energy of a signal can be decomposed into energy contributions from different frequencies.

- Now, if $y[k] = G(q)u[k]$, then we also know $Y(e^{j\omega}) = G(e^{j\omega})U(e^{j\omega})$.
- So, we can estimate $G(e^{j\omega})$ by dividing the DFTs of $y[k]$ and $u[k]$.
  - Inject $u[k]$ into the system and measure $y[k]$.
  - Form DFT of $y[k] \Longrightarrow Y_N(\omega) = \dfrac{1}{\sqrt{N}} \displaystyle\sum_{k=1}^{N} y[k]e^{-j\omega k}$.
  - Form DFT of $u[k] \Longrightarrow U_N(\omega) = \dfrac{1}{\sqrt{N}} \displaystyle\sum_{k=1}^{N} u[k]e^{-j\omega k}$.
  - Finally, form $\widehat{G}_N(e^{j\omega}) = \dfrac{Y_N(\omega)}{U_N(\omega)}$.
- If $u[k]$ lacks frequencies, $\widehat{G}_N(e^{j\omega})$ is undefined at those frequencies.
  - Thus, method requires that $u[k]$ be fairly broadband. *e.g.*, chirp or white noise (*e.g.*, PRBS).
- As we collect more data, we get better frequency resolution (more frequency points).

Performance of the ETFE

- To see how well it works in practice, must recognize that all real systems have disturbance.
  - Actual system output of the form $y[k] = G(q)u[k] + v[k]$.
  - The disturbance plays havoc with the simple ETFE method.
- If we assume $u[k]$ is a realization of a stochastic process, and that $u[k]$ and $v[k]$ are independent, and $v[k]$ is zero-mean, then can show as $N \to \infty$,

  PROPERTY I: $\left|\mathbb{E}\left[\widehat{G}_N(e^{j\omega}) - G(e^{j\omega})\right]\right| < \dfrac{c_1}{\sqrt{N}\,|U_N(\omega)|}$.
  - That is, the ETFE is an asymptotically unbiased estimate.

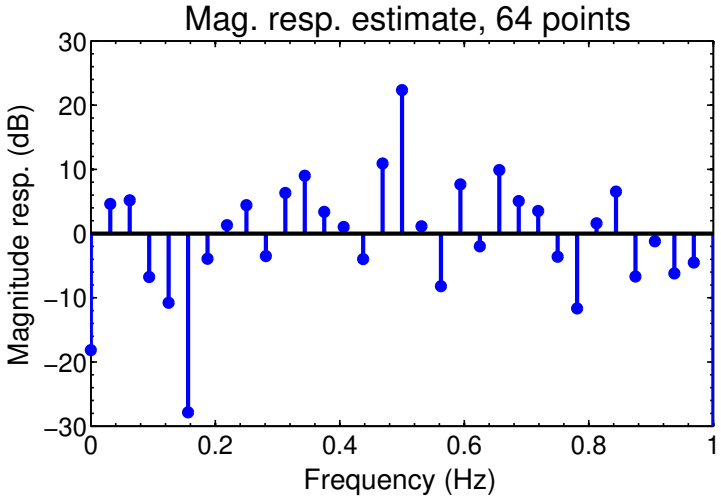
  PROPERTY II: $\mathbb{E}\left[\left|\widehat{G}_N(e^{j\omega}) - G(e^{j\omega})\right|^2\right] \approx \dfrac{\Phi_v(\omega)}{|U_N(\omega)|^2} + \dfrac{c_2}{N}$.
  - That is, the variance of ETFE decreases as N increases, but only to the fixed SNR floor at each point $\Phi_v(\omega)/\Phi_u(\omega)$.

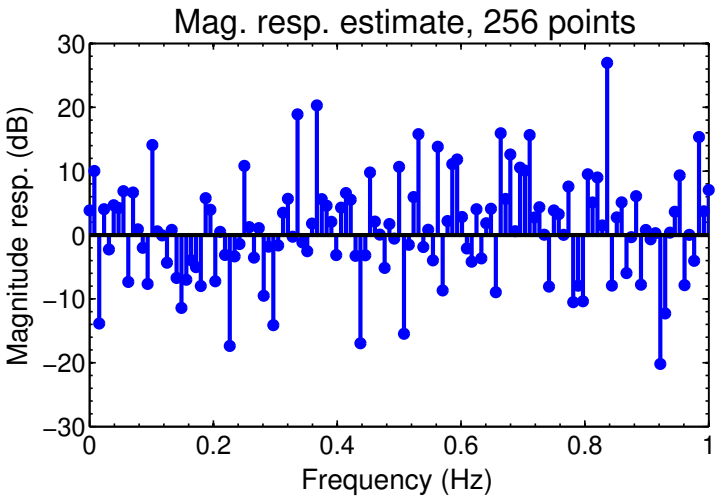
  PROPERTY III: Estimates at different (even nearby) frequencies are asymptotically uncorrelated.
  - This is bad. It means that as we collect more data, the frequency response estimate does not become smoother.

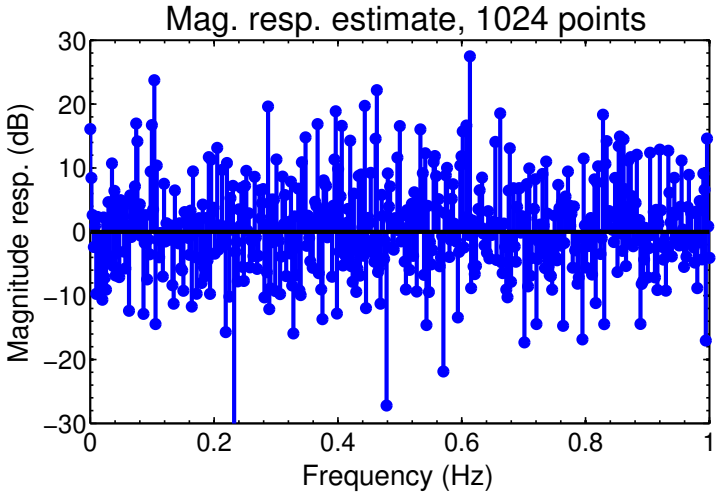
EXAMPLE: Use ETFE to estimate trivial unity-gain magnitude response using sequences of different lengths of Gaussian white noise with unit variance and $T = 0.5$.

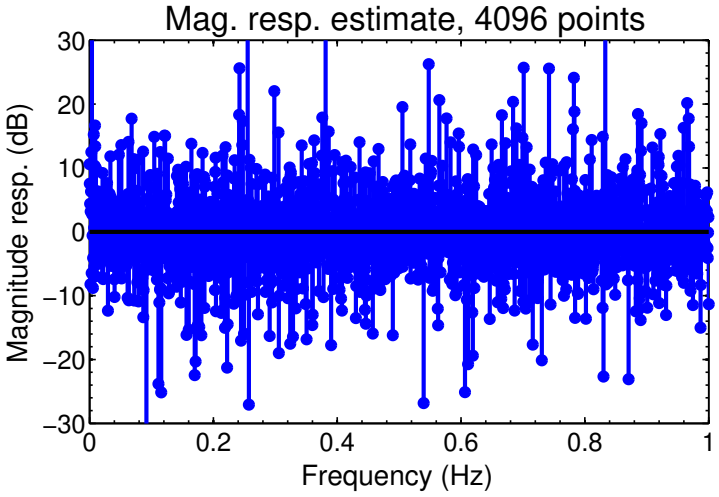
```
u = randn(8192,1); Fs = 2; % Assume 2Hz sample rate: Compute input signal
y = u(1025:end); % System = pure delay
sizes = [64, 256, 1024, 4096];
for N = 1:length(sizes),
  U = fft(u(1:sizes(N)),sizes(N))/sqrt(sizes(N)); % Compute the DFT
```

```
  Y = fft(y(1:sizes(N)),sizes(N))/sqrt(sizes(N)); % Compute the DFT
  f = Fs/2*linspace(0,1,sizes(N)/2+1);      % Corresponding freq vector

  figure(N); clf;
  Ghat = Y./U;
  stem(f,20*log10(abs(Ghat(1:sizes(N)/2+1))),'filled'); ylim([-30 30]);
  title(sprintf('Mag. resp. estimate, %d points',sizes(N)));
  xlabel('Frequency (Hz)'); ylabel('Magnitude resp. (dB)');
end
```









- Estimate shows no sign of converging to the actual flat line.
- Just more frequency points; overall picture very rough!
- To get an improved frequency response estimate, we'll have to smooth/average somehow.

3.5: Spectra of quasi-stationary signals

- The basic idea we would like to try next is to use time-domain correlation to filter the signals.
- However, we run into an immediate (theoretical) issue.
  - We know the exact values of the input signal $u[k]$ as we have generated them, so they are not technically random, even if they were generated using a random-signal generator.
  - Since the output depends in part on $u[k]$, it also is not entirely random, and hence it cannot be wide-sense stationary;
  - So autocorrelation and crosscorrelation functions, as previously defined, do not exist.
- That is, if

$$y[k] = \sum_{m=0}^{\infty} g[m]u[k-m] + v[k],$$

  and $u[k]$ is deterministic, then $\mathbb{E}\big[y[k]\big]$ is a function of the (known) $u[k]$ signal and is not constant.
  - Therefore, $y[k]$ is not stationary, and definitions of autocorrelation and cross correlation fail (at least, in principle).
- To deal with this problem, we define quasi-stationary signals.
- A signal $\{s[k]\}$ is said to be quasi-stationary if
  - $\mathbb{E}\big[s[k]\big] = \mu_s[k],\ |\mu_s[k]| \leq C_1\ \forall\, t$, and some bound C_1;
  - $\mathbb{E}\big[s[k_1]s[k_2]\big] = R_s[k_1,k_2],\ |R_s[k_1,k_2]| \leq C_2$, for some bound C_2;
  - $\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} R_s[k, k-\tau] = R_s[\tau]\ \forall \tau.$

NOTE: The expectation operation $\mathbb{E}[\cdot]$ is taken with respect to stochastic part of $s[k]$.

- That is, if $s[k]$ is deterministic, then $\mathbb{E}[\cdot]$ has no effect (the expected value of a constant is that constant).

- For notational simplicity, let $\overline{\mathbb{E}}\big[f[k]\big] = \lim_{N\to\infty} \frac{1}{N} \sum_{m=1}^{N} \mathbb{E}\big[f[m]\big]$.
  - Note: $f[k]$ here refers to the whole signal, since $N \to \infty$. It does not refer only to a kth element of the signal.

- Then,

$$\mu_u = \overline{\mathbb{E}}\big[u[k]\big] = \lim_{N\to\infty} \frac{1}{N} \sum_{m=1}^{N} \mathbb{E}\big[u[m]\big]$$

$$R_u[\tau] = \overline{\mathbb{E}}\big[u[k]u^T[k-\tau]\big] = \lim_{N\to\infty} \frac{1}{N} \sum_{m=1}^{N} \mathbb{E}\big[u[m]u^T[m-\tau]\big].$$

- These definitions "work" for both deterministic and stochastic $u[k]$.
- Will almost always assume quasi-stationarity.

Power spectrum

- When $R_u[\tau]$ exists (*i.e.*, the signal is stationary or quasi-stationary), we define its (power) spectrum as

$$\Phi_u(\omega) = \sum_{\tau=-\infty}^{\infty} R_u[\tau] e^{-j\tau\omega}.$$

- Similarly, we define the cross spectrum between two signals as

$$\Phi_{uw}(\omega) = \sum_{\tau=-\infty}^{\infty} R_{uw}[\tau] e^{-j\tau\omega}.$$

- The spectrum $\Phi_u(\omega)$ is always real, but the cross spectrum $\Phi_{uw}(\omega)$ is complex-valued in general.
  - Its real part, $\mathbb{R}(\Phi_{uw}(\omega))$, is the cospectrum, and its imaginary part, $\mathbb{I}(\Phi_{uw}(\omega))$, is the quadrature spectrum.
  - Its argument, $\arg(\Phi_{uw}(\omega))$ is the phase spectrum and its magnitude, $|\Phi_{uw}(\omega)|$, is the amplitude spectrum.
- By definition of the inverse DTFT, the power in a signal is

$$\overline{\mathbb{E}}\big[u^2[k]\big] = R_u[0] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_u(\omega)\,\mathrm{d}\omega.$$

EXAMPLE: $u[k]$ is white, so $R_u[\tau] = \sigma^2\delta[\tau]$. Find $\Phi_u(\omega)$.

$$\Phi_u(\omega) = \sum_{\tau=-\infty}^{\infty} \sigma^2\delta[\tau]e^{-j\tau\omega} = \sigma^2 \;\forall\; \omega.$$

EXAMPLE: If scalar $u[k]$ is quasi-stationary, $G(q)$ stable, and scalar $y[k] = G(q)u[k]$, then $y[k]$ is quasi-stationary. Find $\Phi_y(\omega)$ in terms of $G(e^{j\omega})$ and $\Phi_u(\omega)$.

- First, note a needed result regarding cross-correlations:

$$\begin{aligned} R_{uy}[\tau] &= \mathbb{E}\big[u[k]y[k-\tau]\big] \\ &= \mathbb{E}\left[u[k]\left(\sum_{i=-\infty}^{\infty} g[i]u[k-\tau-i]\right)\right] \\ &= \sum_{i=-\infty}^{\infty} \underbrace{\mathbb{E}\Big[u[k]u[k-\tau-i]\Big]}_{R_u[\tau+i]} g[i]. \end{aligned}$$

- This looks a lot like convolution, but the sign on i is wrong in R_u.

- Define $\tilde{g}[i] = g[-i]$

$$\begin{aligned} R_{uy}[\tau] &= \sum_{i=-\infty}^{\infty} \tilde{g}[-i] R_u[\tau + i] \\ &= \sum_{i'=+\infty}^{-\infty} \tilde{g}[i'] R_u[\tau - i'] \\ &= g[-\tau] * R_u[\tau]. \end{aligned}$$

- Then, we use this result in the following

$$\begin{aligned} R_y[\tau] &= \mathbb{E}\Big[y[k] y[k-\tau] \Big] \\ &= \mathbb{E}\left[\left(\sum_{i=-\infty}^{\infty} g[i] u[k-i] \right) \left(\sum_{j=-\infty}^{\infty} g[j] u[k-\tau-j] \right) \right] \\ &= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} g[i] g[j] \mathbb{E}\Big[u[k-i] u[k-\tau-j] \Big] \\ &= \sum_{i=-\infty}^{\infty} g[i] \underbrace{\sum_{j=-\infty}^{\infty} g[j] R_u[\tau - i + j]}_{R_{uy}[\tau - i]} \\ &= \sum_{i=-\infty}^{\infty} g[i] R_{uy}[\tau - i] \\ &= g[\tau] * R_{uy}[\tau]. \end{aligned}$$

- So,

$$R_y[\tau] = g[\tau] * R_{uy}[\tau] = g[\tau] * R_u[\tau] * g[-\tau]$$

$$\Phi_y(z) = G(z) \Phi_u(z) G(z^{-1})$$

$$\Phi_y(\omega) = G(e^{j\omega}) G(e^{-j\omega}) \Phi_u(\omega) = |G(e^{j\omega})|^2 \Phi_u(\omega).$$

EXAMPLE: If $v[k] = H(q)e[k]$, and $e[k]$ is white (zero-mean, variance σ^2) then

$$\Phi_v(\omega) = \left|H(e^{j\omega})\right|^2 \sigma^2.$$

- We see that $H(q)$ shapes the spectrum of $v[k]$.
- However, the inverse process is of greater interest to us. That is, given $\Phi_v(\omega)$, we will wish to find the $H(q)$ that would have created it.
  - This process is called spectral factorization.

FACT: If $\Phi_v(\omega)$ is a rational function of $\cos(\omega)$ and $\Phi_v(\omega) > 0 \ \forall \omega$ then there exists an H such that

- $\left|H(e^{j\omega})\right|^2 = \Phi_v(\omega)$,
- $H(e^{j\omega})$ is stable,
- $1/H(e^{j\omega})$ is stable.

- Most useful when we measure or estimate $\Phi_v(\omega)$: spectral estimation.

EXAMPLE: If the system $G(q)$ has more than one input and/or output, we can redo the prior example to find

$$\Phi_y(\omega) = G(e^{j\omega})\Phi_u(\omega)G^T(e^{-j\omega}).$$

- Also,

$$\Phi_{yu}(\omega) = G(e^{j\omega})\Phi_u(\omega).$$

KEY POINT: If we can estimate $\hat{\Phi}_{yu}(\omega)$ and $\hat{\Phi}_u(\omega)$, then we can develop an estimate for $\widehat{G}(e^{j\omega}) = \hat{\Phi}_{yu}(\omega)/\hat{\Phi}_u(\omega)$.

3.6: Smoothed periodograms

IDEA: Might consider estimating $\widehat{G}(e^{j\omega}) = \hat{\Phi}_{yu}(\omega)/\hat{\Phi}_u(\omega)$ using in turn periodogram estimators of the spectra, where we have already noted that the periodogram of signal $u[k]$ is defined as

$$|U_N(\omega)|^2 = \left| \frac{1}{\sqrt{N}} \sum_{k=1}^{N} u[k] e^{-j\omega k} \right|^2 .$$

PROBLEM: Some properties of the spectrum with respect to the periodogram include:

1. $\lim_{N\to\infty} \mathbb{E}\left[|Y_N(\omega)|^2 \right] \to \Phi_y(\omega) \ \forall \omega.$
   - Asymptotically unbiased estimator of $\Phi_y(\omega)$.
   - *Does not* mean that the bias is small for finite N.
2. $\lim_{N\to\infty} \mathbb{E}\left[\left(|Y_N(\omega)|^2 - \Phi_y(\omega) \right)^2 \right] \to \Phi_y^2(\omega) > 0$ in general!
   - Variance (scatter around mean) *does not* decrease as we collect more data!
3. $\lim_{N\to\infty} \mathbb{E}\left[\left(|Y_N(\omega_1)|^2 - \Phi_y(\omega_1) \right) \left(|Y_N(\omega_2)|^2 - \Phi_y(\omega_2) \right) \right] \to 0 \ \forall \omega_1 \neq \omega_2.$
   - Therefore, the estimate has zero covariance (*i.e.*, no strong relationship exists), even between nearby frequency points.
   - Not consistent with what we would expect.
   - Periodogram is very rough and scattered, therefore "naïve."

- Could form the estimate using these biased, noisy estimates, but it would be quite worthless.
- One approach (Welch's method) to reduce the variance of the spectral estimate is to average several independent estimates.

PROCEDURE: Collect data $y[k]$, $1 \leq k \leq N$.

- Divide data into n segments of m data points each ($N = nm$).
- Form $y^j[k] = y[k + m(j-1)]$ for $1 \leq k \leq m-1$ and $1 \leq j \leq n$.
- Form n periodogram-based spectral estimates

$$\hat{\Phi}_m^j(\omega) = \frac{1}{m}\left|\sum_{k=1}^{m} y^j[k]e^{-j\omega k}\right|^2 ; \ 1 \leq j \leq n.$$

- Provided that measurements separated by m points are weakly correlated, then the n spectral estimates $\hat{\Phi}_m^j$ are nearly independent.
  - Therefore, can improve estimate of $\Phi(\omega)$ by averaging.
- With $\hat{\Phi}_m^A(\omega) = \frac{1}{n}\sum_{j=1}^{n} \hat{\Phi}_m^j(\omega)$, $\text{var}\left[\hat{\Phi}_m^A(\omega)\right] = \frac{1}{n}\text{var}\left[\hat{\Phi}_m^j(\omega)\right]$, where $\text{var}\left[\hat{\Phi}_m^j(\omega)\right]$ is effectively the same as $\text{var}\left[\hat{\Phi}_N(\omega)\right]$.
  - Therefore, variance decreases as we average.
- Primary trade here is that (for N fixed), larger n leads to smaller m, which leads to larger $\Delta\omega \sim 1/n$.
  - Therefore, get a better estimate, but at fewer frequency points.
  - Could be a problem if the spectrum is known to have narrow peaks.
    - Need $\Delta\omega$ small enough to see the peak (3–5 points in the half-power bandwidth).
- Method works well when you have long data streams (“excess data”).

EXAMPLES: Tried this averaging approach on three signals

1. Spectrally flat via pseudo-random binary sequence (PRBS: p. 418)

- Periodic, deterministic sequence with white-noise like properties
- Most easily created using feedback around a shift register. Length N register gives $2^N - 1$ period.
- Very flat spectral content for large register length: `idinput`.

2. Shaped, but smooth, $N = 2048$.
3. Shaped, but peaked.

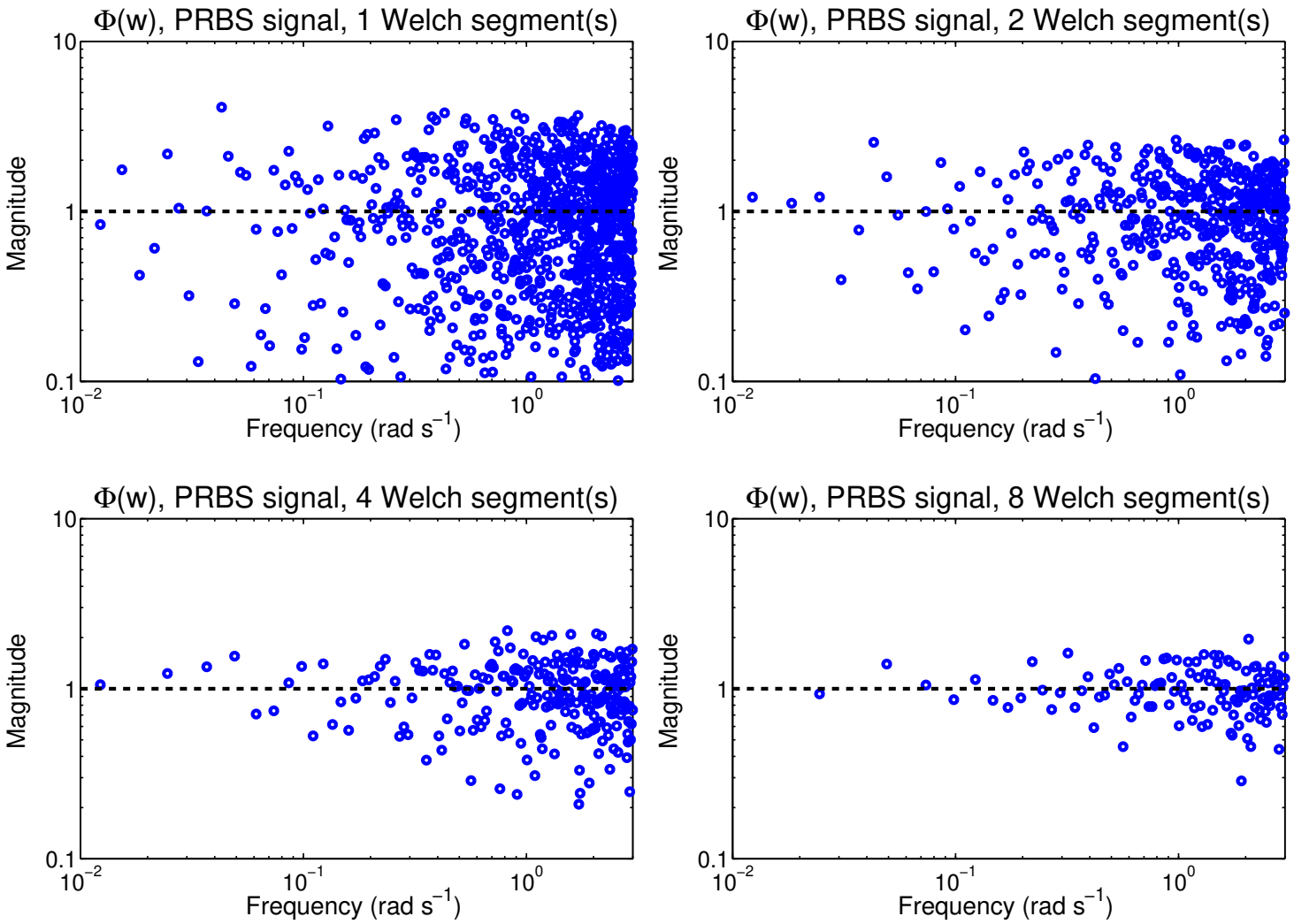
- First defined a "welch" function

```
function G=welch(U,M)
  %U data, M number of segments
  L=1:round(length(U)/M); N = max(L);

  wk = (L-1)*2*pi/max(L); wk = wk(:);
  G = []; G.frequency = wk; G.SpectrumData = 0*wk;
  for k=1:M
    G1 = abs(fft(U(L+(k-1)*N))).^2/N;
    G.SpectrumData=G.SpectrumData+G1/M;
  end
end
```

- Results below for spectrally flat PRBS input:

```
N = 2048; U = idinput(N,'prbs'); % virtually white sequence

for M = [1,2,4,8],
  G = welch(U,M); figure(10+M); clf;
  loglog(G.frequency,G.SpectrumData,'bo',[.01 5],[1 1],'k--');
  title(sprintf('\\Phi(w), PRBS signal, %d Welch segment(s)',M))
  xlabel('Frequency (rad s^{-1})'); ylabel('Magnitude');
end
```

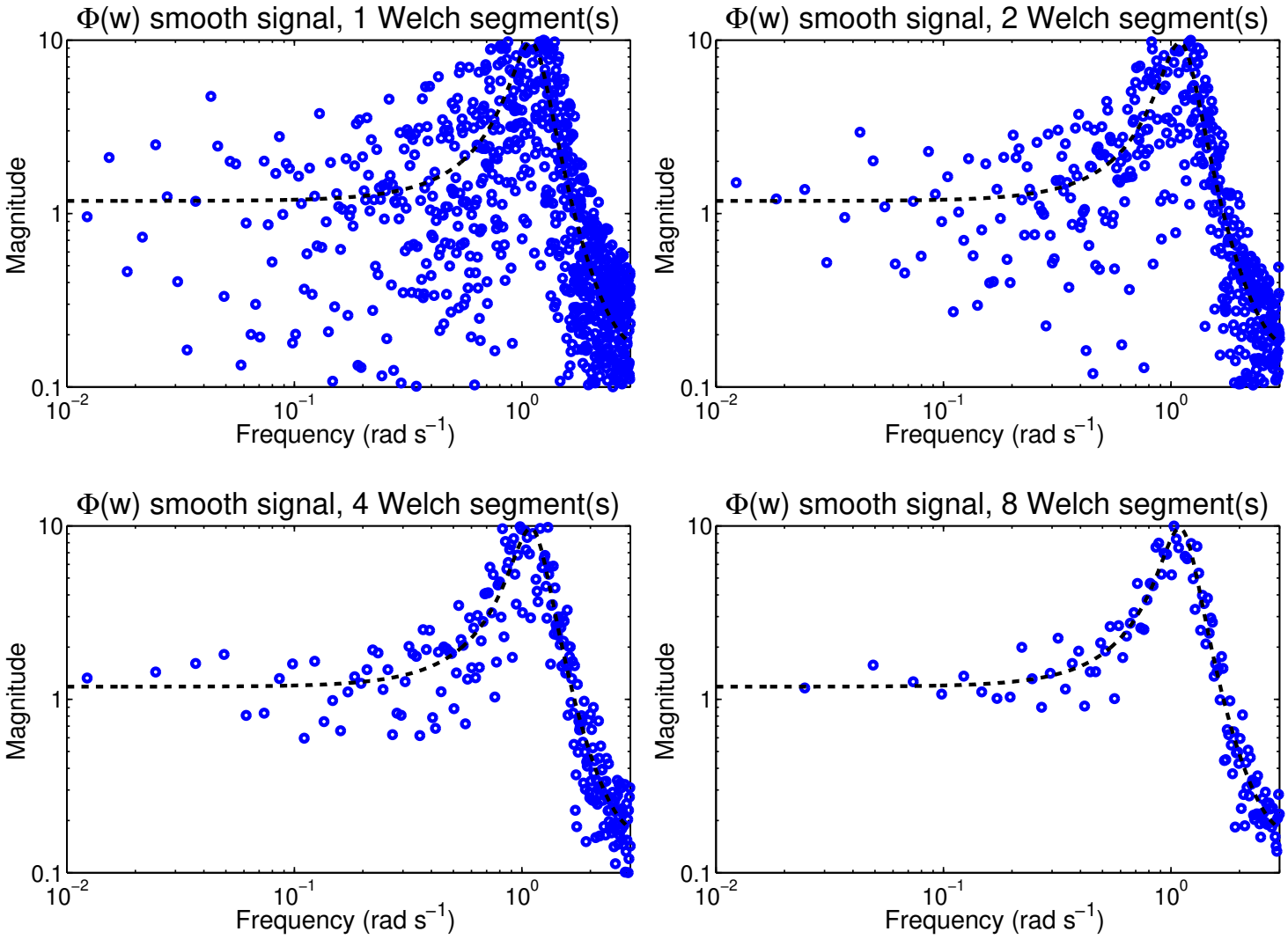
- First plot is very poor, but steady improvement with increasing n.
- $\Delta\omega$ increases by factor of four. Only 18 points between 0.1 to 1 rad s^{-1} in final plot.



- Results below for second signal, smoothed PRBS:

```
a01=-0.8; b01=1; c01=1;
sys = idpoly([1 0.9*a01 a01^2],[0 0 b01],[c01],[1]);
Y=idsim([0*U U],sys);
Gact = []; Gact.frequency = logspace(-2,pi,1000);
Gact.data = squeeze(bode(sys,Gact.frequency));

% Note that we need to compare \Phi with |G|^2
for M = [1,2,4,8],
  G = welch(Y,M); figure(10+M); clf;
  loglog(G.frequency,G.SpectrumData,'bo',...
    Gact.frequency,Gact.data.^2,'k--');
  title(sprintf('\\Phi(w) smooth signal, %d Welch segment(s)',M))
  xlabel('Frequency (rad s^{-1})'); ylabel('Magnitude');
end
```


- Very hard to see $G(e^{j\omega})$ in first plot. Improves with n and is quite obvious in the case $n = 8$.



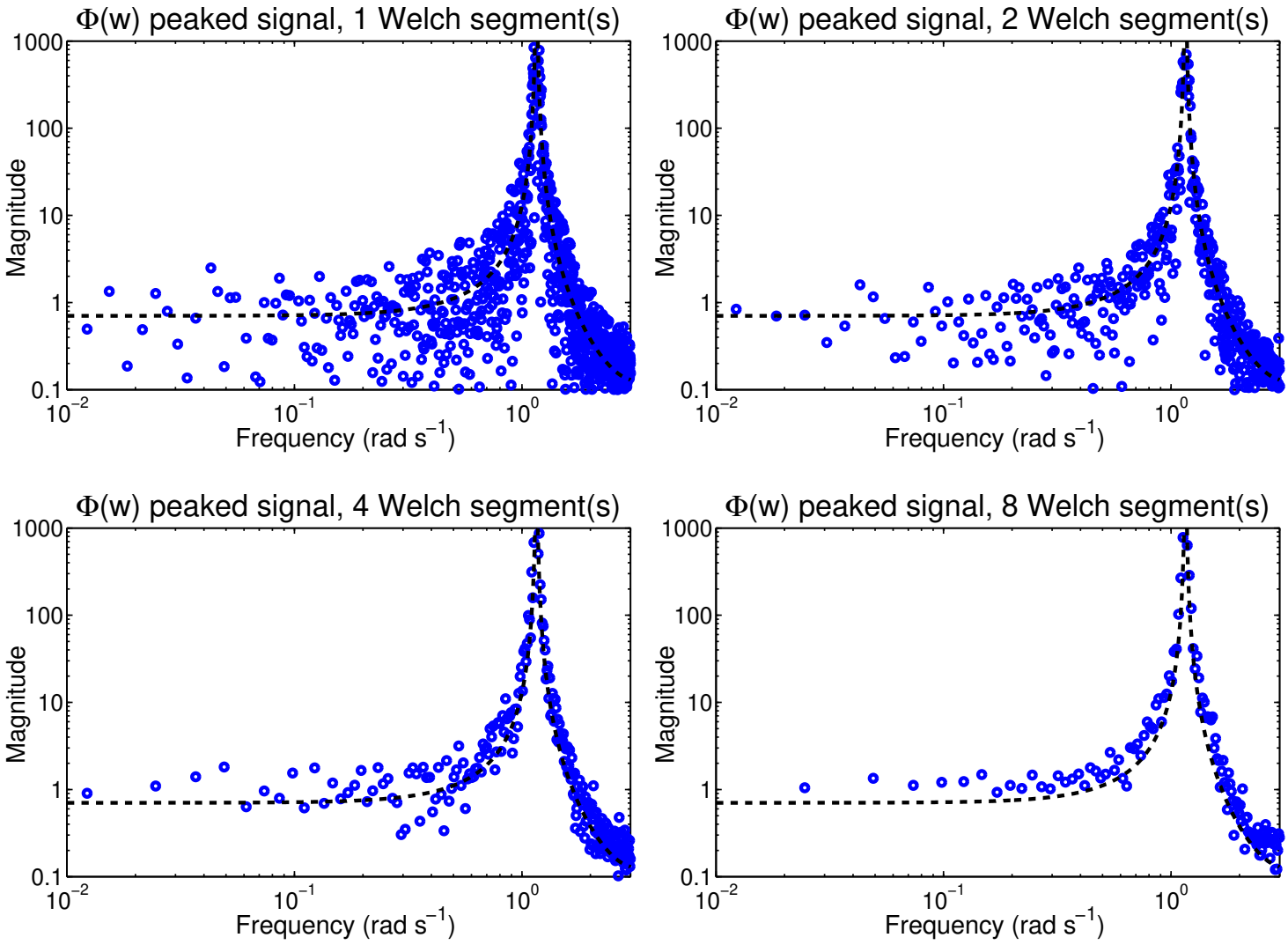
- Results below for third signal, peaked output:

```
a01=-0.8;b01=1;c01=1;
sys = idpoly([1 a01 1.55*a01^2],[0 0 b01],[c01],[1]);
Y=idsim([0*U U],sys);
Gact = []; Gact.frequency = logspace(-2,pi,1000);
Gact.data = squeeze(bode(sys,Gact.frequency));

% Note that we need to compare \Phi with |G|^2
for M = [1,2,4,8],
  G = welch(Y,M); figure(10+M); clf;
  loglog(G.frequency,G.SpectrumData,'bo',...
    Gact.frequency,Gact.data.^2,'k--');
```



```
  title(sprintf('\\Phi(w) peaked signal, %d Welch segment(s)',M))
  xlabel('Frequency (rad s^{-1})'); ylabel('Magnitude');
end
```

- Trend similar, but we can see that $\Delta\omega$ too large impacts our ability to see the peak.



3.7: Frequency filtering

- Another method smooths estimate in frequency domain.

APPROACH: Assume that (true) $\Phi_y(\omega)$ is relatively constant in the neighborhood of a given frequency ω_k.

- That is, $\Phi_y(\omega_{k-m}) \approx \cdots \approx \Phi_y(\omega_k) \approx \cdots \approx \Phi_y(\omega_{k+m})$.
- Have already seen that $\text{cov}\left[\hat{\Phi}_y(\omega_1), \hat{\Phi}_y(\omega_2)\right] \approx 0 \ \forall \ \omega_1 \neq \omega_2$.
  - Can use all of $\hat{\Phi}_y(\omega_{k-m}), \ldots, \hat{\Phi}_y(\omega_k), \ldots, \hat{\Phi}_y(\omega_{k+m})$ as uncorrelated estimates of $\hat{\Phi}_y(\omega_k)$.
  - Can improve our estimate of $\hat{\Phi}_y(\omega_k)$ by averaging (smoothing over frequency) these independent estimates.

RESULT: We get

$$\bar{\hat{\Phi}}_y(\omega_k) = \frac{1}{2m+1} \sum_{j=-m}^{m} \hat{\Phi}_m(\omega_{k-j}).$$

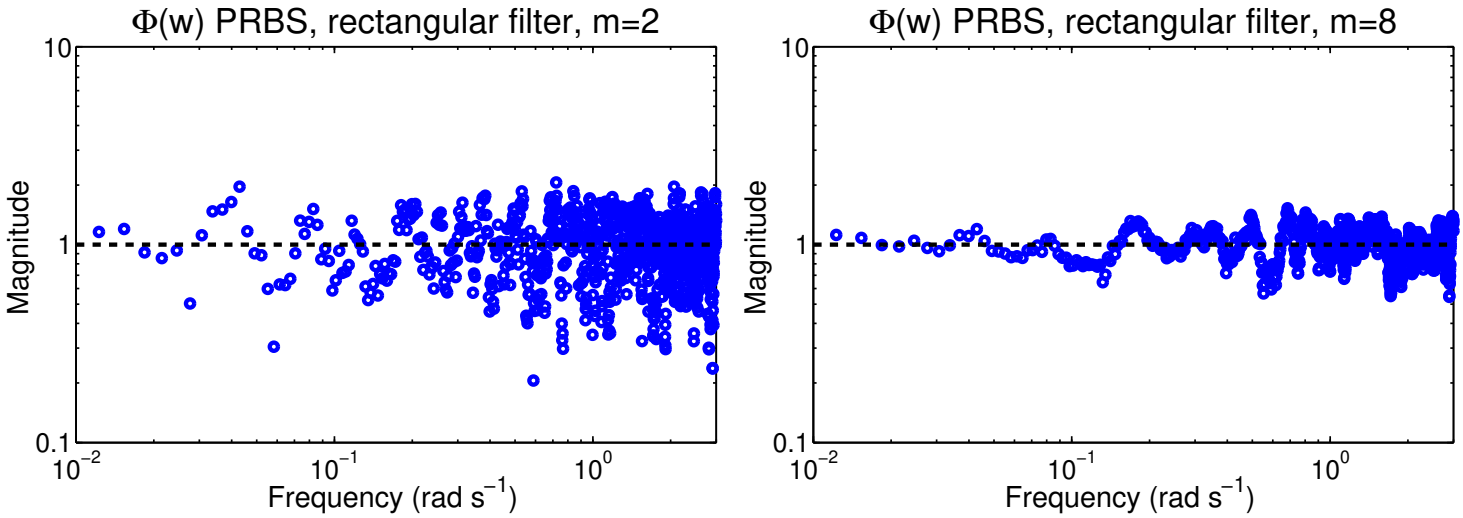
- Then,

$$\text{var}\left[\bar{\hat{\Phi}}_y(\omega_k)\right] \approx \frac{1}{2m+1} \text{var}\left[\hat{\Phi}_y(\omega_k)\right].$$

- Therefore, by choosing a large m, we can substantially reduce the variance of the estimate.
- Results below for PRBS input:

```
G = welch(U,1); AMP = G.SpectrumData;
for M = [1,2,4,8],
  figure(10+M); clf;
  AMPfilt = flipud(filter(ones(1,2*M+1)/(2*M+1),1,flipud(AMP)));
  loglog(G.frequency,AMPfilt,'bo',[0.01 5],[1 1],'k--');
  title(sprintf('\\Phi(w) PRBS, rectangular filter, m=%d',M));
  xlabel('Frequency (rad s^{-1})'); ylabel('Magnitude');
end
```

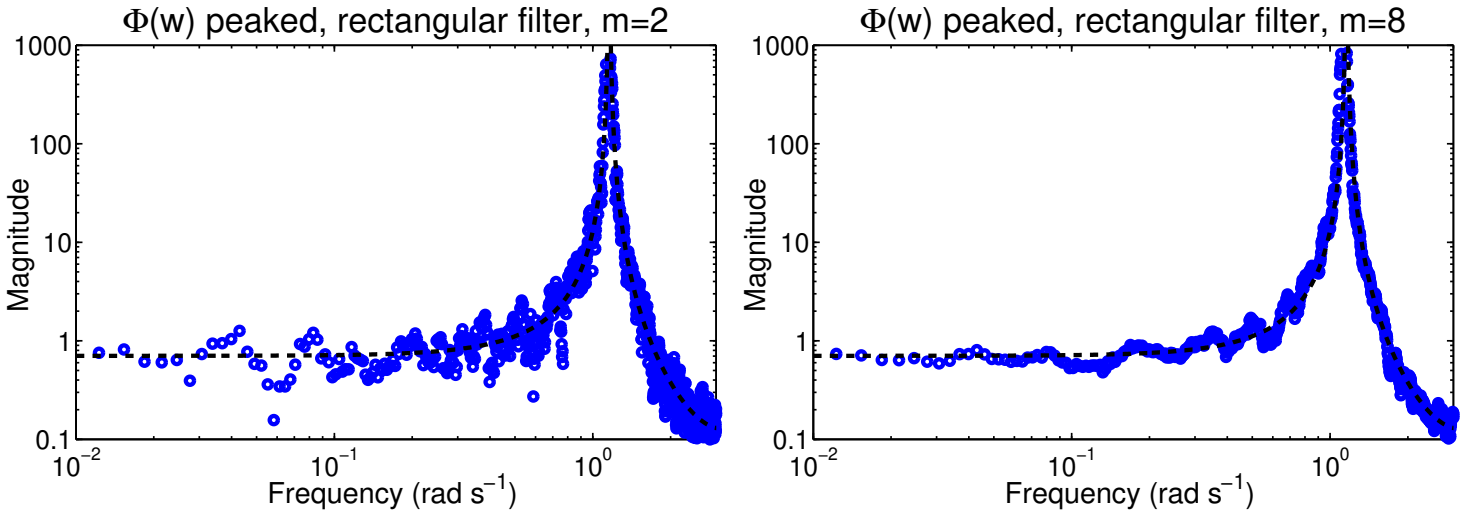
- Smoothing clear (variance reduced), but not great!



- Results below for peaked input:

```
G = welch(Y,1); AMP = G.SpectrumData;
for M = [1,2,4,8],
  figure(10+M); clf;
  AMPfilt = flipud(filter(ones(1,2*M+1)/(2*M+1),1,flipud(AMP)));
  loglog(G.frequency,AMPfilt,'bo',Gact.frequency,Gact.data.^2,'k--');
  title(sprintf('\\Phi(w) peaked, rectangular filter, m=%d',M));
  xlabel('Frequency (rad s^{-1})'); ylabel('Magnitude');
end
```

- Not bad, but lose definition of peak when m too big.



- There are much better frequency filters to use, and the results show that we need them! That is, can generalize this result to the form

$$\bar{\hat{\Phi}}_y(\omega) = \int_{-\omega_N}^{\omega_N} w_\gamma(\omega - \phi)\hat{\Phi}_y(\phi)\,\mathrm{d}\phi,$$

where γ is a "width parameter," and where the convolution integral smooths the spectral estimate $\hat{\Phi}_y(\phi)$ over a frequency band $[\omega - \omega_N, \omega + \omega_N]$ using the weighting function w_γ.

  - Convolution in frequency domain = multiplication in time domain.
  - Equivalent to multiplying $R_y[\tau]$ by a windowing function. See "Blackman–Tukey" topic, later.

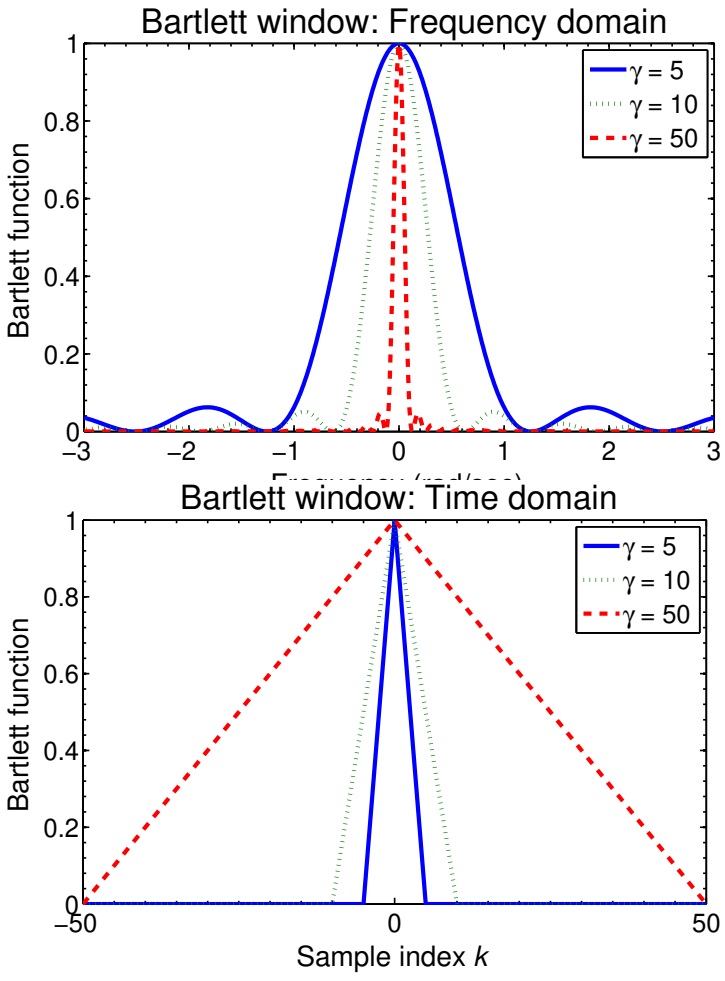
- Prior example used "rectangular" weighting function:

$$w_\gamma = \frac{1}{2m+1}.$$

- Another example is the "Bartlett" weighting function:

$$w_\gamma(\omega) = \frac{1}{\gamma}\left(\frac{\sin(\gamma\omega/2)}{\sin(\omega/2)}\right)^2.$$

- Three cases shown in figures.
- In frequency domain, we see a main lobe and side-lobe ripples.
- Degree of smoothing a direct function of central-lobe width.

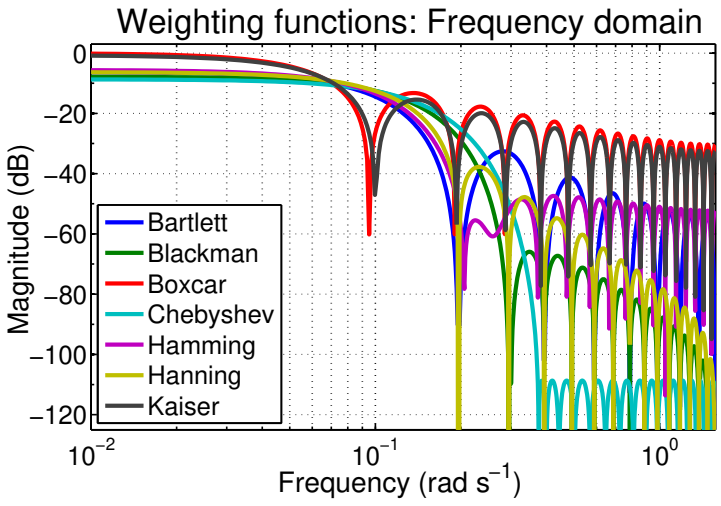


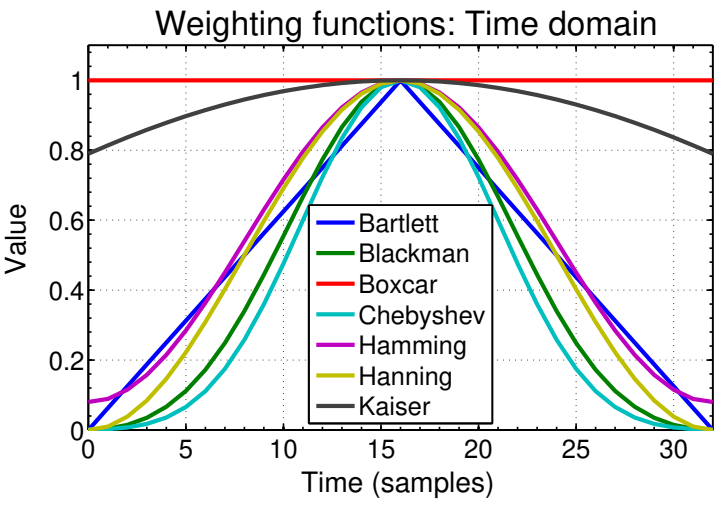
- Wide lobe smooths over larger local frequency window (variance goes down).

3.8: Blackman–Tukey estimate

- Various common “named” weighting functions exist:

```
N = 33; Nfft = (N-1) * 64;
w1 = bartlett(N); w2 = blackman(N); w3 = boxcar(N);   w4 = chebwin(N,100);
w5 = hamming(N);  w6 = hann(N);     w7 = kaiser(N,1);
w = [w1 w2 w3 w4 w5 w6 w7];
W = abs(fft(w,Nfft))/N; f = 1/2*linspace(0,pi,Nfft/2+1);

figure(1); clf;
plot(0:N-1,w); grid
legend('Bartlett','Blackman','Boxcar','Chebyshev','Hamming','Hanning',...
       'Kaiser','location','south')
title('Weighting functions: Time domain');
xlabel('Time (samples)'); ylabel('Value');

figure(2); clf;
semilogx(f(1:Nfft/2),20*log10(W(1:Nfft/2,:))); grid
legend('Bartlett','Blackman','Boxcar','Chebyshev','Hamming','Hanning',...
       'Kaiser','location','southwest')
title('Weighting functions: Frequency domain');
xlabel('Frequency (rad s^{-1})'); ylabel('Magnitude (dB)');
```





- All trade-off central lobe width with the reduction in the side lobes, just do it different ways.

- Choice of γ more important than choice of which window (weighting function).
- Want large lobe (small γ) to smooth $\hat{\Phi}$ to reduce the variance.
- Large lobe means that we include many points in our average. This blurs the features (peaks).
- Want small lobe (large γ) to preserve features.
- The selection of the "right" γ requires that we have a good idea of what Φ_y looks like.

KEY POINT: We can substantially improve the variance of our spectral estimate by reducing γ and make the central lobe wider. But, this will distort the estimate too, resulting in a bias.

- These weighting functions can be applied in the frequency domain, as we have just seen.
- They can also be applied in the time domain. We previously had the result

$$\Phi_y(\omega) = \sum_{\tau=-\infty}^{\infty} R_y[\tau] e^{-j\omega\tau}.$$

- We can use this to form a new estimate.
- Given data $y[k]$, we can form an estimate for $\hat{R}_y[\tau]$ as

$$\hat{R}_y^N[\tau] = \frac{1}{N} \sum_{k=\tau}^{N-1} y[k]y[k-\tau].$$

  - Accurate estimator for τ small compared to N.
- Would expect this accuracy to degrade for larger τ since there are fewer terms in the sum.

- One possible spectral estimator is:

$$\hat{\Phi}_y(\omega) = \sum_{\tau=-\infty}^{\infty} \hat{R}_y[\tau] e^{-j\omega\tau}.$$

- Not going to use this many terms ($\tau \ll \infty$).
- Would like to de-weight $\hat{R}_y[\tau]$ for large τ ➠ window operation, but in the time domain now (called a lag window).
  - Directly analogous to the frequency-domain windows.
  - Just the IDFT of the frequency window $W(\omega) \Longleftrightarrow w[k]$.
- New estimate is

$$\hat{\Phi}_{BT}(\omega) = \sum_{\tau=-(\gamma-1)}^{\gamma-1} w[\tau] \hat{R}_y[\tau] e^{-j\omega\tau}.$$

- Filter width given by γ. For example, the Bartlett window is defined by

$$w[\tau] = \begin{cases} 1 - |\tau|/\gamma, & 0 \leq |\tau| \leq \gamma \\ 0 & \text{else.} \end{cases}$$

- Increasing γ includes more $\hat{R}_y[\tau]$ terms. Reduces effect of window.
- Window gives reduced weight for $\hat{R}_y[\tau]$: goes to zero for large τ.
- With careful analysis of the new estimate (indices of the various summations), can show that these two approaches are equivalent.
- So, can think about smoothing in frequency or time domains.

3.9: ETFE and SPA toolbox commands

- Can automate the frequency-response estimation process using `etfe.m` and `spa.m`.
  - ETFE computes the ratio of DFTs of output and input sequences.
    - If a window size M_{input} is supplied, also does some frequency-domain smoothing with a Hamming window.
  - SPA ("spectral analysis") computes the ratio of smoothed spectra of output and input sequences, using $\hat{\Phi} = \sum_k w[k]\hat{R}[k]e^{-j\omega k}$ approach with the Hanning window $w[k]$.
    - Width of window in frequency domain is $\alpha_E + 1$, where $\alpha_E \approx \#\text{points}/M_{\text{input}}$, the number of points is usually around 1024, and M_{input} is the "M" in the calling statement.
    - With the number of points equal to 1024, we get

M_{input}	64	128	256	512
α_E	16	8	4	2

      which shows that as M_{input} increases, α_E decreases and the degree of smoothing is reduced (variance increases).

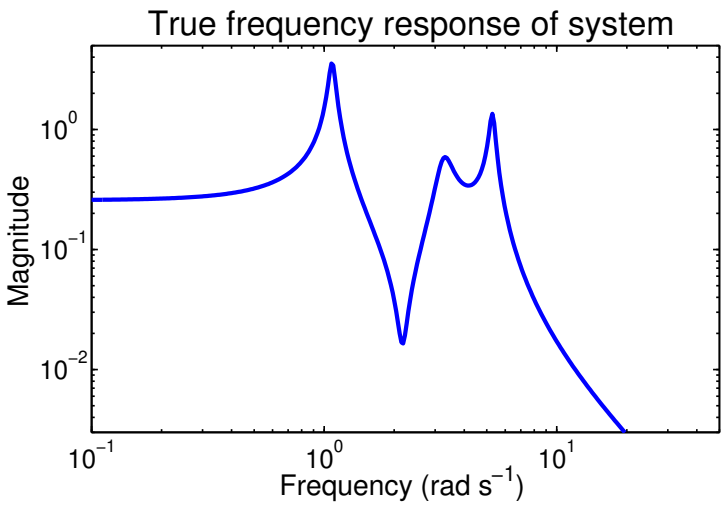
- Results shown in figures for several cases. All have $N = 1024$ points.
- Actual dynamics are quite complex with three poles and zeros (large dynamic range) and fairly large amount of sensor noise.

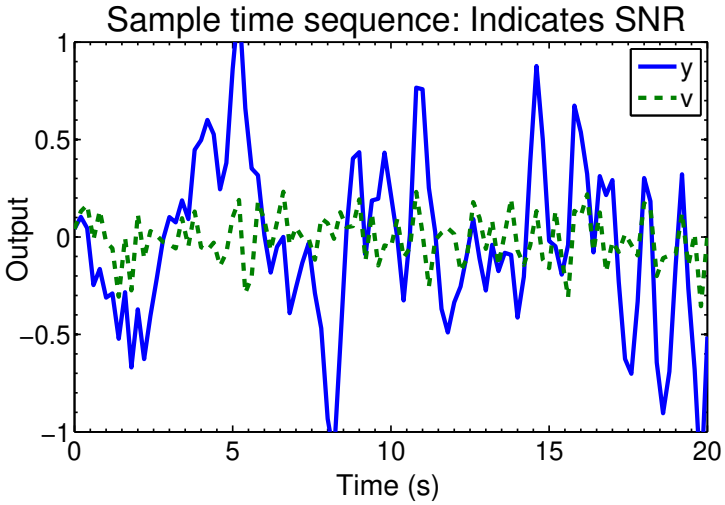
```
% Set up lightly damped system with high dynamic range
m1=5;m2=4;c1=4.0;c2=0.1;k1=35;k2=40; M=diag([m1 m2 2*m2]);
K=[k1+k2 -k2 0;-k2 2* k2 -k2;0 -k2 k2];
C=[c1+c2 -c2 0;-c2 2*c2 -c2;0 -c2 c2];
a=[zeros(3) eye(3);-M\[K C]]; b=[zeros(3,1);M\[1;0;0]];
c=[m1 0 m2 0 m2 0]; d=0;
```

```
% sample at 5 Hz
Npts=1024; Ts=0.2; Tf=(Npts-1)*Ts; t=(0:Ts:Tf)';

sigma=1/8; % set sensor noise size compared to process noise
u=idinput(length(t),'prbs'); % pseudo random noise input
w=[u sigma*randn(size(t))]; % add sensor noise
% simulate system with both PRBS and sensor noise
y=lsim(a,[b 0*b],c,[d 1],w,t);
z=[y u];
zf=idfilt(z,4,.8); % filtering of both y and u to chop out HF stuff
```

- First, we note the true frequency response of the system, and the sample output sequence (and noise sequence).

```
f=logspace(-2,1,400); g=freqresp(a,b,c,d,1,1j*2*pi*f);
phg=unwrap(angle(g))*180/pi;

figure(1); clf; loglog(2*pi*f,abs(g));
xlabel('Frequency (rad s^{-1})'); ylabel('Magnitude')
title('True frequency response of system');

figure(2); clf; plot(t(1:101),z(1:101,1),t(1:101),w(1:101,2),'--');
xlabel('Time (s)'); ylabel('Output')
title('Sample time sequence: Indicates SNR'); legend(['y';'v']);
```



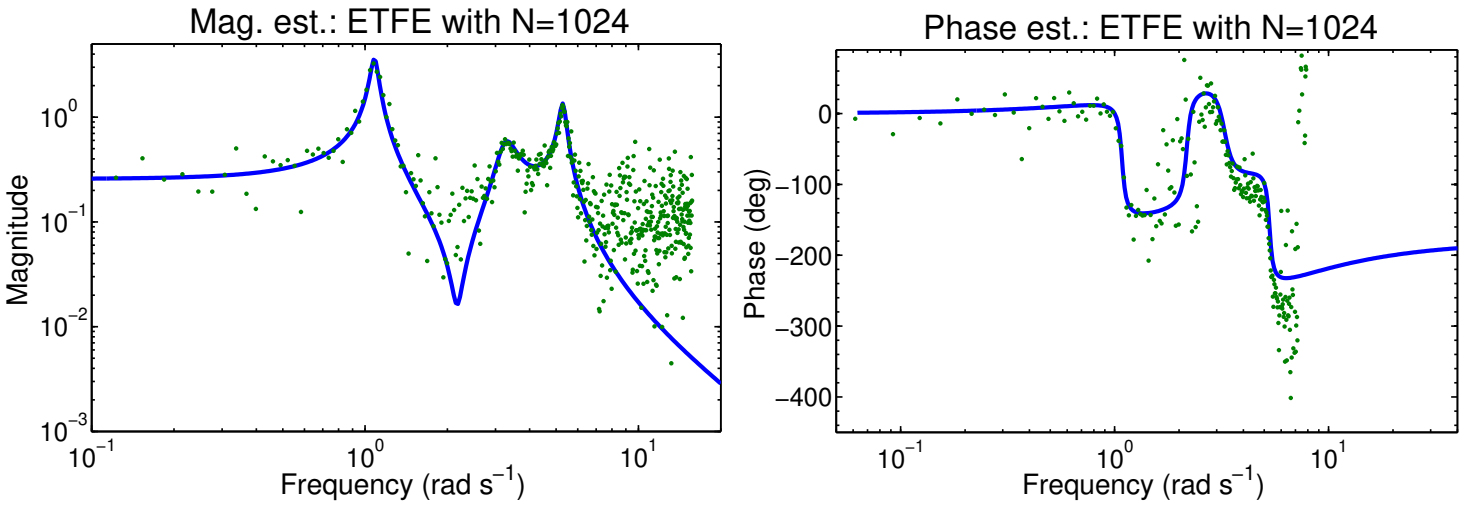


- Raw ETFE: Somewhat correct but useless.

```
ghat=etfe([y u],[],Npts/2,Ts); [what,ghm,ghp]=getff(ghat,1,1);
```



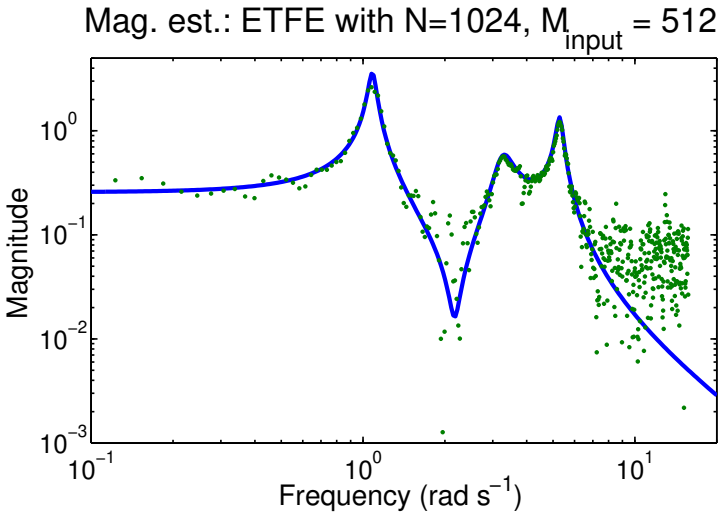
```
figure(3);clf; loglog(2*pi*f,abs(g),what,ghm,'.','markersize',10);
xlabel('Frequency (rad s^{-1})');ylabel('Magnitude')
title(sprintf('Mag. est.: ETFE with N=%d',Npts))

figure(4); clf; semilogx(2*pi*f,phg,what,ghp,'.','markersize',10);
xlabel('Frequency (rad s^{-1})');ylabel('Phase (deg)')
title(sprintf('Phase est.: ETFE with N=%d',Npts))
```

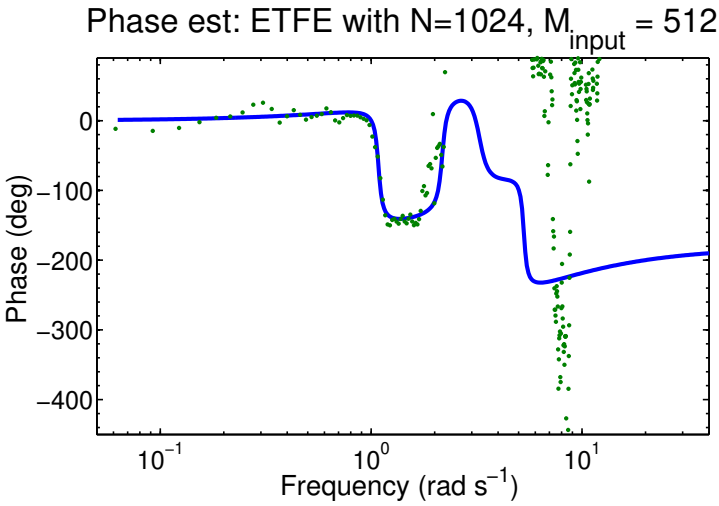


- Code for ETFE with different M_{input} values:

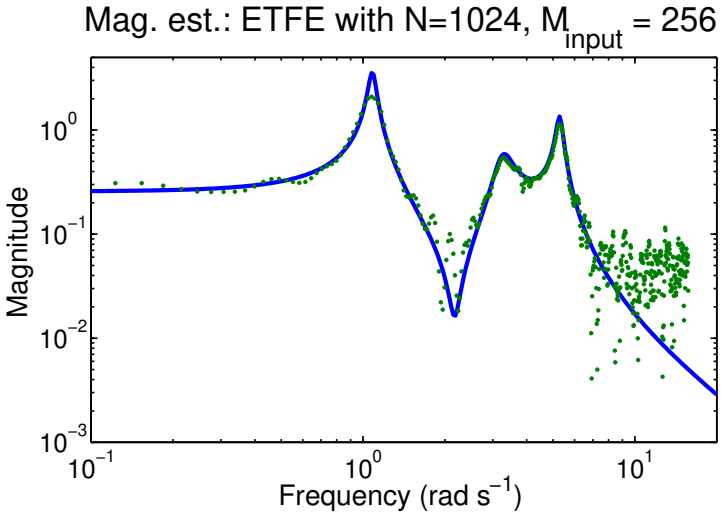
```
Minps = [512,256,128,64];
for theMinp = 1:length(Minps),
  Minp=Minps(theMinp);
  ghat=etfe([y u],Minp,Npts/2,Ts); [what,ghm,ghp]=getff(ghat,1,1);

  figure(4+theMinp);clf; loglog(2*pi*f,abs(g),what,ghm,'.');
  xlabel('Frequency (rad s^{-1})');ylabel('Magnitude')
  title(sprintf('Mag. est.: ETFE with N=%d, M_{input} = %d',Npts,Minp))

  figure(5+theMinp); clf; semilogx(2*pi*f,phg,what,ghp,'.');
  xlabel('Frequency (rad s^{-1})');ylabel('Phase (deg)')
  title(sprintf('Phase est: ETFE with N=%d, M_{input} = %d',Npts,Minp))
end
```

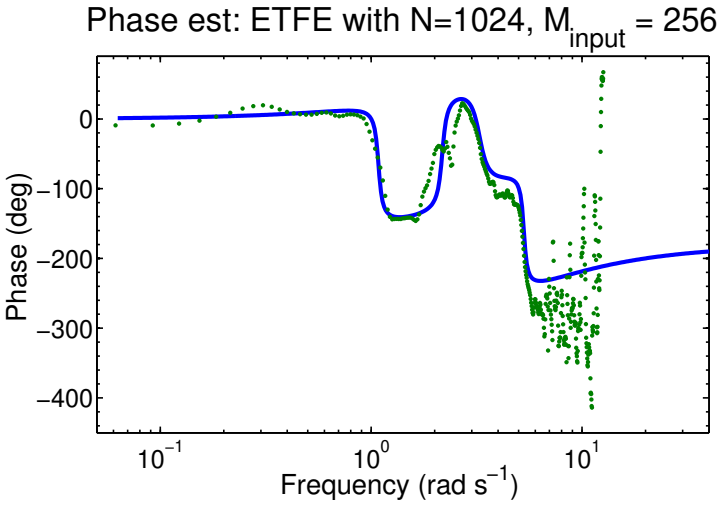
- $M_{\text{input}} = 512$: True dynamics clearer, but still too much scatter.



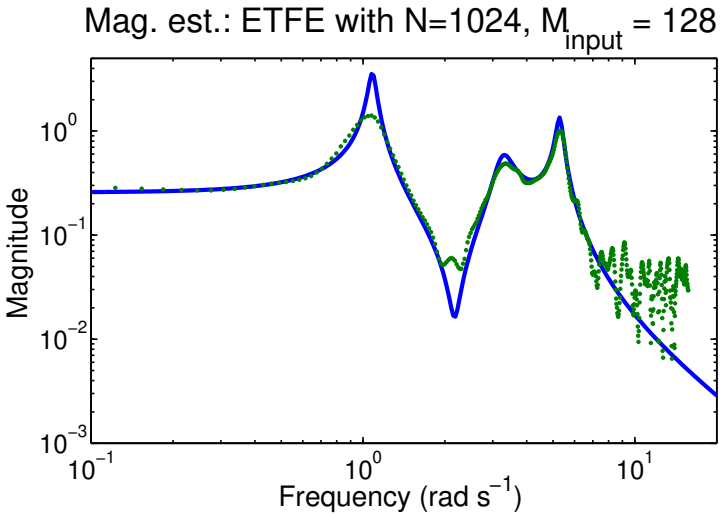


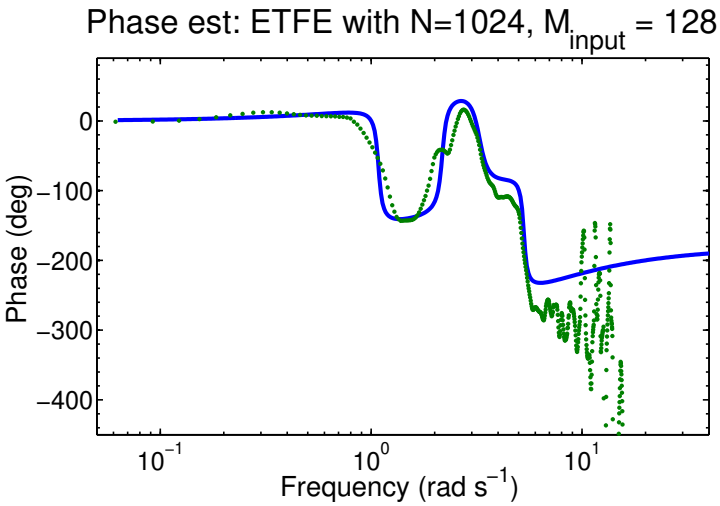
- $M_{\text{input}} = 256$: Much better definition of peaks and zero.



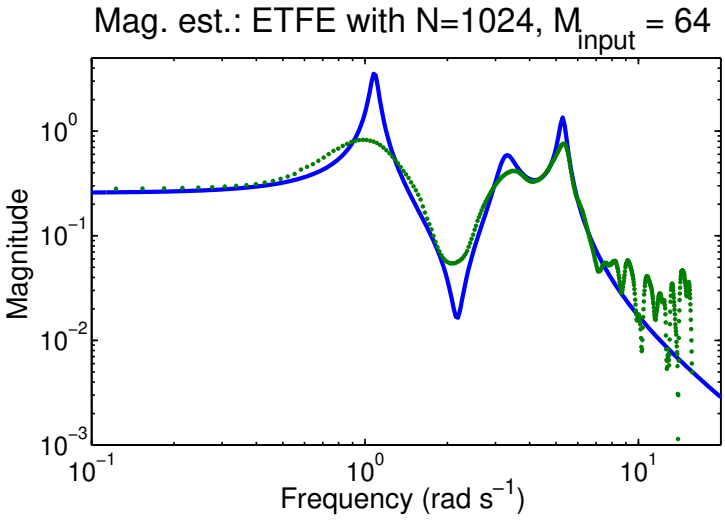


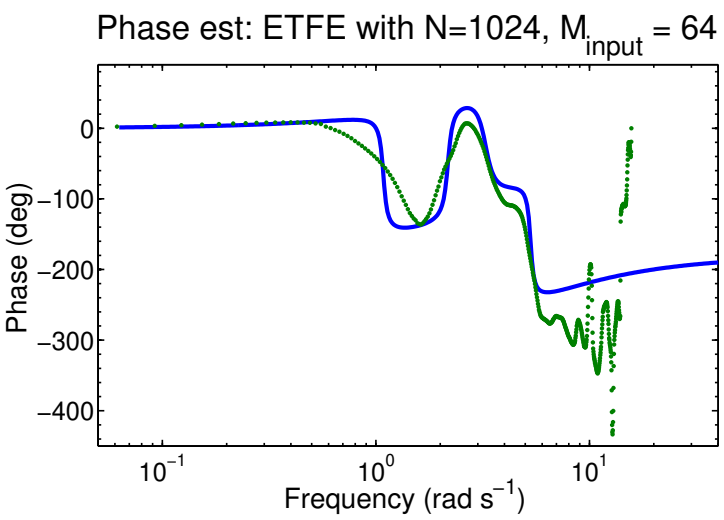
- $M_{\text{input}} = 128$: Very good smooth match. Losing height of first peak?





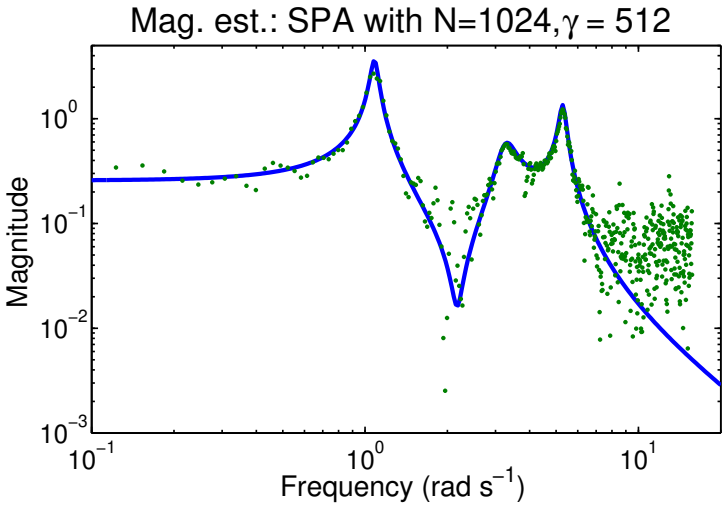
- $M_{\text{input}} = 64$: Now clearly over-smoothed. Losing height of all peaks.

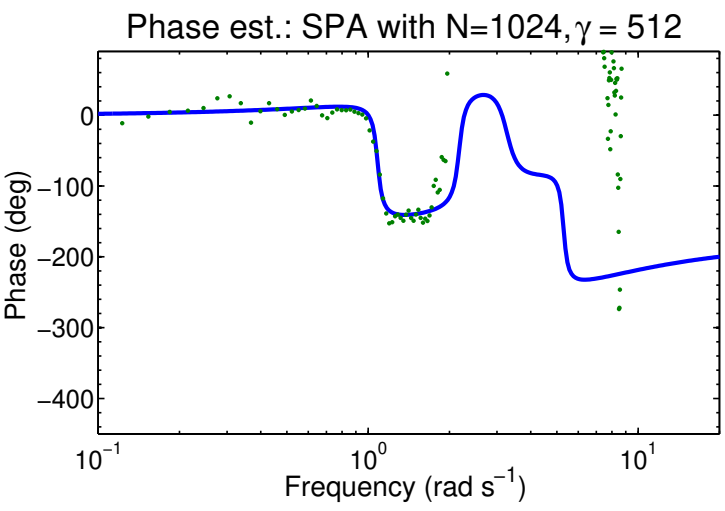




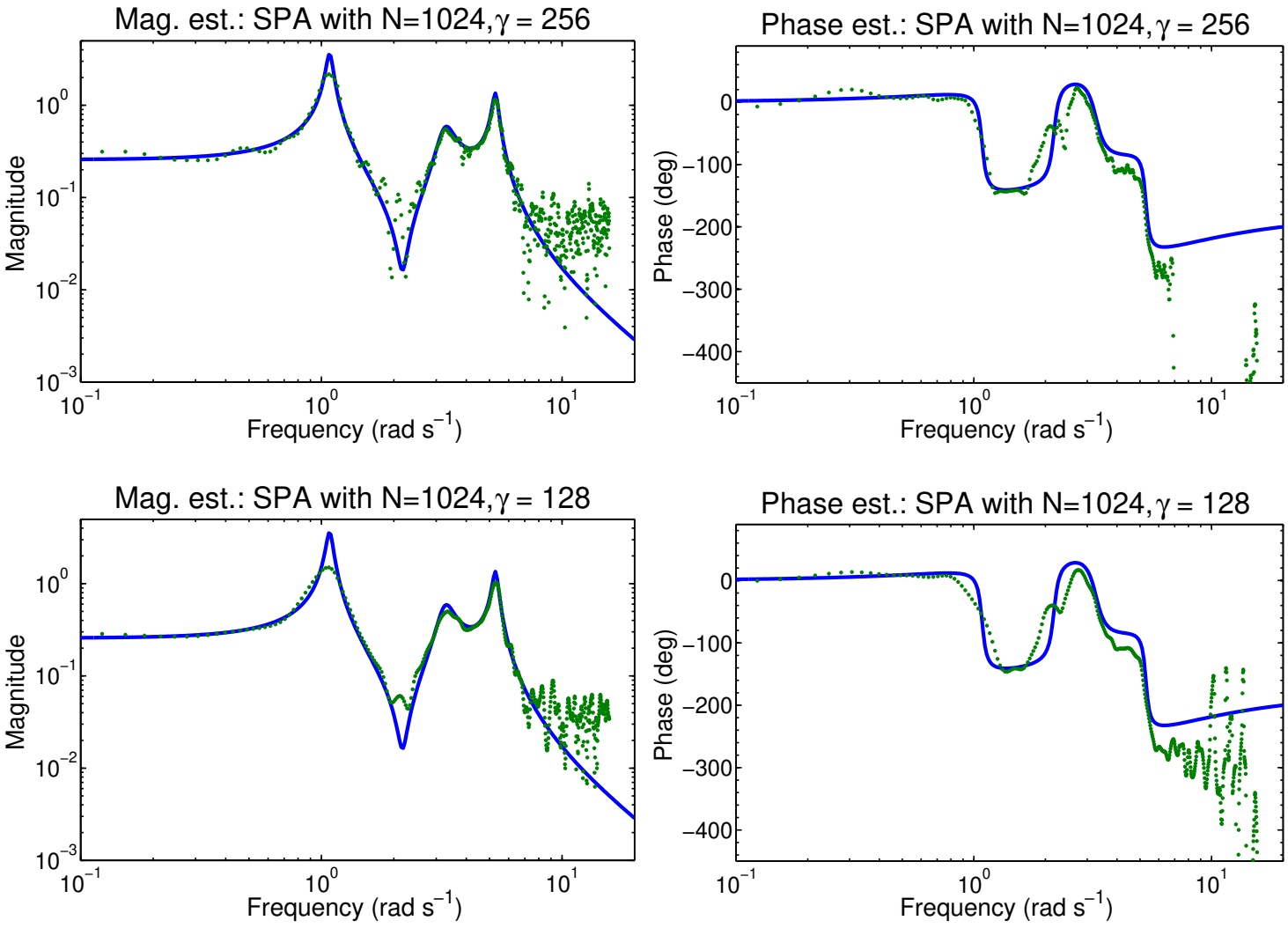
- Code for SPA with different γ values:

```
gammas = [128,256,512];
for theGamma = 1:length(gammas),
  winGam=gammas(theGamma);
  ghats=spa([y u],winGam,what,[],Ts); [what,ghms,ghps]=getff(ghats,1,1);

  figure(12+theGamma);clf; loglog(2*pi*f,abs(g),what,ghms,'.');
  xlabel('Frequency (rad s^{-1})');ylabel('Magnitude')
  title(sprintf('Mag. est.: SPA with N=%d, \\gamma = %d',Npts,winGam))

  figure(13+theGamma); clf; semilogx(2*pi*f,phg,what,ghps,'.');
  xlabel('Frequency (rad s^{-1})');ylabel('Phase (deg)')
  title(sprintf('Phase est.: SPA with N=%d, \\gamma = %d',Npts,winGam))
end
```





- $\gamma = 512$: Very similar to ETFE results for $M_{\text{input}} = 256$.
- $\gamma = 256$ and $\gamma = 128$: See effect of over-smoothing.



- Give pretty good estimates. Definitely something we can work with.
- Note that `etfe.m` and `spa.m` can produce frequency-dependent error bounds on the frequency-response estimate.
  - Very valuable when considering robust control.
  - See code in notes chapter 1 for an example using `spa.m`.

Quality of transfer function

- Without knowing truth, how can we tell if estimate is any good?
- Often use coherency spectrum between $y[k]$ and $u[k]$, defined as

$$\hat{\kappa}_{yu}^{N}(\omega) \triangleq \sqrt{\frac{\left|\hat{\Phi}_{yu}^{N}(\omega)\right|^2}{\hat{\Phi}_{y}^{N}(\omega)\hat{\Phi}_{u}^{N}(\omega)}}$$

  as a measure of the "goodness" of the frequency response function.
- Can show that $\hat{\Phi}_{v}^{N}(\omega) = \hat{\Phi}_{y}^{N}(\omega)\left[1 - (\hat{\kappa}_{yu}^{N}(\omega))^2\right]$.
- Thus, a value of $\hat{\kappa}_{yu}^{N} \approx 1$ at some frequency implies that there is (nearly) perfect correlation between input and output ➠ Very little noise interference, so that frequency-response value should be quite accurate.

Where from here?

- We've now looked at "nonparametric" methods of system ID, resulting in unit-pulse response and frequency-response estimates.
- Disturbance is an issue for both of these approaches. Must find ways to average out noise better than we have done so far.
- One reason noise remains an issue is that we essentially allow an infinite number of values to describe the system (*i.e.,* unit-pulse response time-domain values, or frequency-response frequency-domain values).
  - Need to reduce number of "parameters" describing system, so that we can average out more of the noise when creating model.
- So, we move on to look at "parametric" system ID.