Chapter 8
Noise and Power

DMD shares many properties with POD in space and the fast Fourier transform (FFT) in time. It is natural to extend the power spectrum from the Fourier transform to DMD, so that the importance of a DMD mode may be interpreted by its power. It is also possible to truncate low-energy modes corresponding to small singular values, as in POD.

A major issue with the standard DMD algorithm is the sensitivity of its eigenvalue spectrum to noisy data. It has been shown that there is a systematic bias in the eigenvalue spectrum with the addition of sensor noise on the data, and this bias does not disappear as more data is sampled [129]. The effect of small sensor noise on DMD and the Koopman expansion has also been carefully explored and characterized [12].

In addition to truncating the SVD of the data, there are two dominant solutions to denoise or debias the DMD spectrum. In the first method, by Dawson et al. [77], data is processed in forward and backward time and then averaged, removing the systematic bias. In the second method, by Hemati et al. [129], the DMD algorithm is solved using a total least-squares algorithm [118] instead of the standard least-squares algorithm. Dawson et al. [77] provide an excellent discussion of the sources of noise along with a comprehensive and accessible comparison of the proposed denoising algorithms.

The methods advocated in this chapter should generally be considered with every application of DMD to a new dataset. In particular, we recommend judicious rank truncation, denoising the DMD eigenvalues, and analyzing the power spectrum.

8.1 ▪ Power spectrum

The power spectrum is a central concept in frequency analysis, providing a visual representation of the frequency content of a signal. Peaks in the power spectrum occur at frequencies that are dominant in the data. Information from the power spectrum, such as the height and broadness of peaks, the magnitude of the noise floor, and resonances between integer multiples of a frequency, can all provide valuable diagnostic information about a signal.

For periodic data, where the DMD eigenvalues are on the complex unit circle, there is a direct correspondence between the power spectrum and the DMD mode amplitude. More generally, however, DMD eigenvalues may have a growth or decay component, making the computation of a DMD power spectrum more involved.

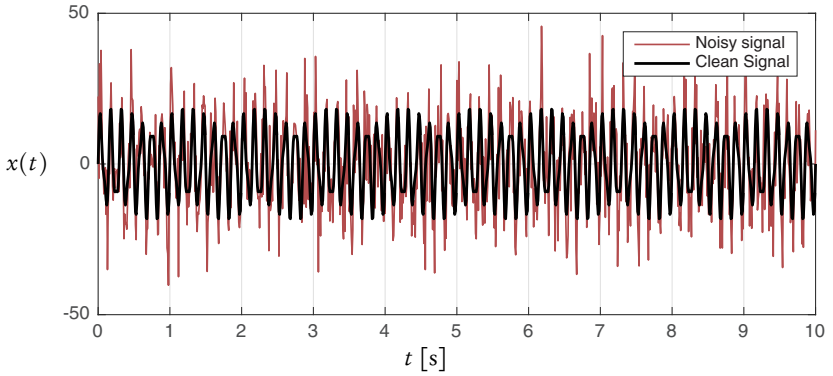

Figure 8.1. *Sum of sines signal from* (8.1).

8.1.1 ▪ Fourier transform and power spectrum

To demonstrate the relationship between the FFT power spectrum and the DMD spectrum, consider a simple example given by the sum of two sine waves:

$$x(t) = 14\sin(7 \times 2\pi t) + 5\sin(13 \times 2\pi t) + 10\eta(t), \tag{8.1}$$

where $\eta(t)$ is a zero-mean Gaussian white noise process with unit standard deviation. This signal, with and without noise, is shown in Figure 8.1 and generated in Algorithm **8.1**.

***ALGORITHM* 8.1. Generate sum of sines signal from (8.1) (`FFTDMD_spectrum.m`).**

```
dt = .01;        % time step of signal
L = 10;
t = 0:dt:L;

% signal: sum of two sines at 13 Hz and 7 Hz
xclean = (14*sin(7*2*pi*t)+5*sin(13*2*pi*t));
% add noise to signal
x = xclean + 10*randn(size(xclean));
ylim([-40 40])
plot(t,x,'Color',[.7 .3 .3],'LineWidth',.9), hold on
plot(t,xclean,'k','LineWidth',1.5)
legend('Noisy signal','Clean Signal')
```

The traditional FFT power spectrum is readily computed, as in Algorithm **8.2**. The analogous DMD spectrum is computed in Algorithm **8.3**. As in Chapter 7, to capture a standing wave, time-shifted copies of the data must be stacked to form the data matrix. The FFT and DMD power spectra are compared in Figure 8.2.

When computing the DMD mode amplitude, we use $\mathbf{b} = \mathbf{\Phi}^{\dagger}\mathbf{x}_1$, based on the amount of mode energy expressed in the first snapshot $\mathbf{x}_1$. This mode energy is then scaled by $2/\sqrt{s}$, where s is the number of rows in the shift-stacked Hankel matrix, as in (7.5) and (7.9). The power scaling $2/\sqrt{s}$ appears to hold regardless of n, the size of $\mathbf{x}$.

It is interesting to note that without additive noise, the FFT and DMD spectral peaks agree perfectly (i.e., $s = 50$ and $r = 4$ in Figure 8.3). However, when we add noise, we obtain increasingly good agreement of the FFT and DMD spectra for an

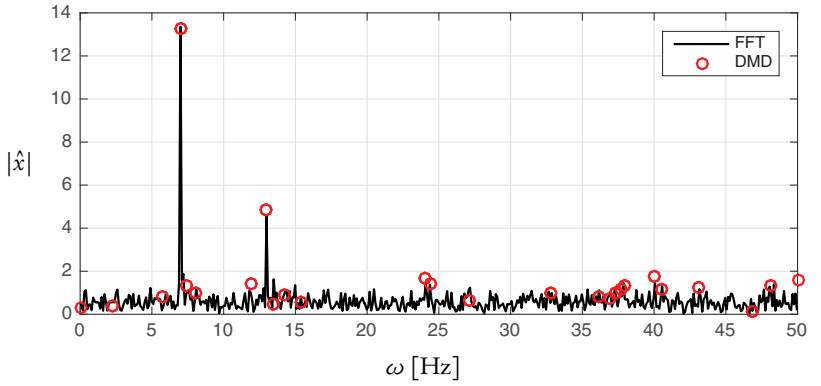

Figure 8.2. *Power spectrum for signal given by a sum of two sine waves at 7 Hz and 13 Hz with additive white noise. The FFT power spectrum is shown in black, and the DMD spectrum is shown in red circles. In this example, $r = 50$ singular values are kept for $s = 500$.*

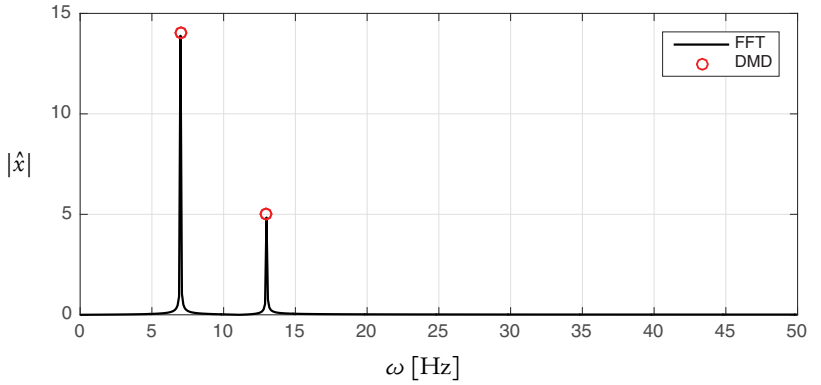

Figure 8.3. *The DMD power spectrum is accurate in the absence of sensor noise if $r = 4$ singular values are kept for $s = 50$.*

increasingly square Hankel matrix of time-delay coordinates (i.e., $s = 500$). This denoising through time-delay coordinates also becomes more important when we compute more DMD eigenvalues (e.g., $r = 50$). These issues of SVD rank truncation and number of time-delay coordinates are explored in Figures 8.2 and 8.4. Failure to rank-truncate judiciously may produce unpredictable and undesired results, where the DMD power spectrum becomes corrupted by too many singular values. A principled approach to singular value truncation will be presented in the next section.

***ALGORITHM* 8.2. Code to compute the FFT power spectrum of the signal in (8.1) (FFTDMD_spectrum.m).**

```
xhat = fft(x);
N = length(t);  % number of samples
xpower = abs(xhat(1:N/2+1))*2/N;
Fs = 1/dt;      % sampling frequency
freqs = Fs*(0:(N/2))/N;
plot(freqs,xpower,'k','LineWidth',1.2)
```

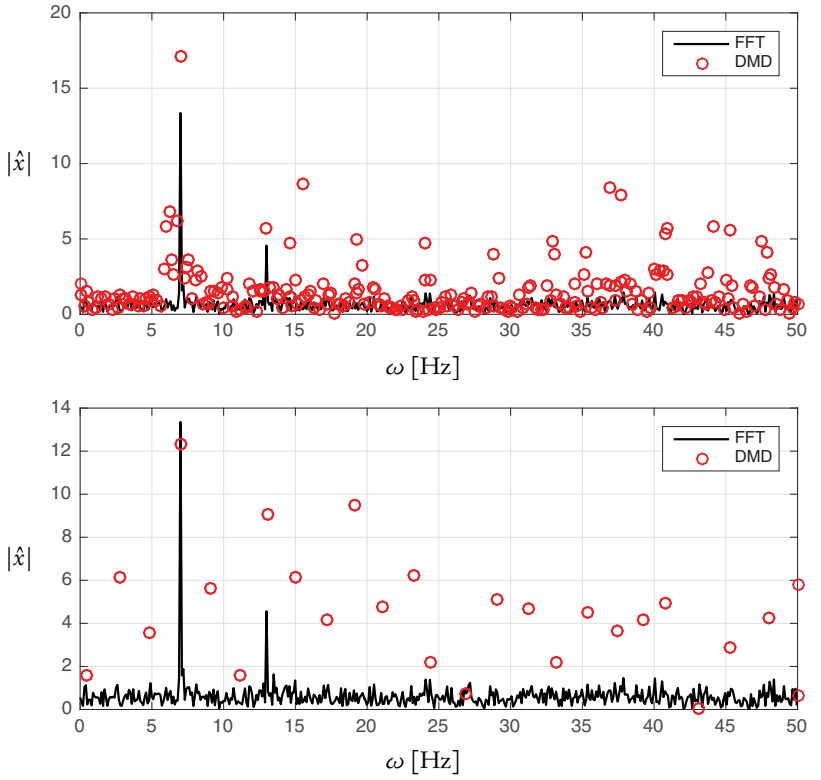

Figure 8.4. *The DMD power spectrum is corrupted if too many singular values are kept. (top) $r = 500$ modes are kept with $s = 500$. (bottom) $r = 50$ modes are kept with $s = 50$.*

***ALGORITHM* 8.3. Code to compute the DMD spectrum of the signal in (8.1) (FFTDMD_spectrum.m).**

```
% better denoising with larger kshift
s = 500;  % number of times to shift-stack signal
for k = 1:s
    X(k,:) = x(k:end-s+k);
end
[U,S,V] = svd(X(:,1:end-1),'econ');

% keep 50 modes and compute DMD spectrum Lambda
r = 50;
Atilde = U(:,1:r)'*X(:,2:end)*V(:,1:r)*inv(S(1:r,1:r));
[W,Lambda] = eig(Atilde);
% convert eigenvalues to continuous time
DMDfreqs = log(diag(Lambda))/dt/2/pi;
Phi = X(:,2:end)*V(:,1:r)*inv(S(1:r,1:r))*W;

% mode amplitude  (based on first snapshot)
b = Phi\X(:,1);
%  need to scale power by 2/sqrt(s)
DMDpower = abs(b)*2/sqrt(s)
```

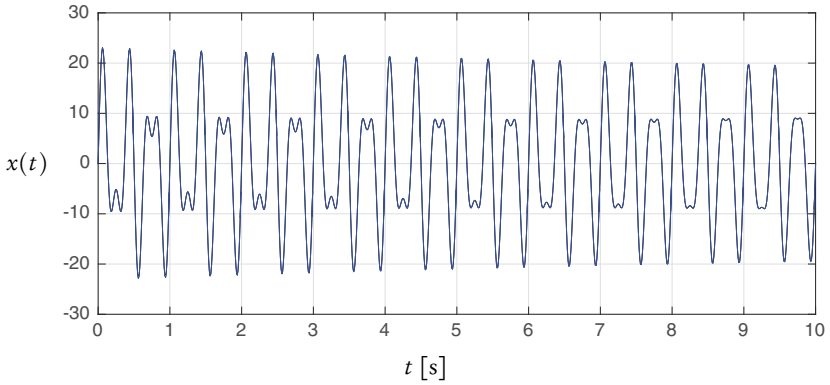

Figure 8.5. *Sum of sines with exponential decay signal from* (8.2).

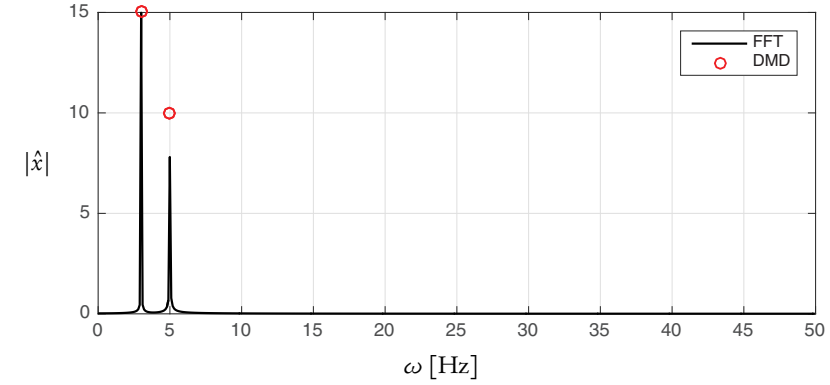

Figure 8.6. *Power spectrum for signal given by a sum of two sine waves with decay. The FFT power spectrum is shown in black, and the DMD spectrum is shown in red circles.*

```
scatter(abs(imag(DMDfreqs)),DMDpower,'r')
legend('FFT','DMD')
```

8.1.2 ▪ DMD power spectrum for growth and decay modes

Now, consider a signal that includes the sum of two sine waves, where one of the sine waves is multiplied by an exponential decay term:

$$x(t) = 15\sin(3 \times 2\pi t) + 10\sin(5 \times 2\pi t)\exp(-t/20). \tag{8.2}$$

This signal is shown in Figure 8.5. As shown in Figure 8.6, the DMD spectrum, obtained by computing $\mathbf{b} = \mathbf{\Phi}^{\dagger}\mathbf{x}_1$, accurately captures the amplitude of the two modes, whereas the FFT spectrum underestimates the amplitude of the decaying mode.

Since the mode is decaying, it is natural to look at whether or not there is a difference in the mode amplitudes if computed using other snapshots $\mathbf{x}_k$. Figure 8.7 shows $\mathbf{\Phi}^{\dagger}\mathbf{x}_k$, which decays over time; the average value of this amplitude matches the FFT amplitude. Instead, it is necessary to scale this amplitude by the eigenvalue, as $\mathbf{\Phi}^{\dagger}\mathbf{x}_k/|\lambda|^k$. The scaled mode amplitude accurately captures the signal in (8.2).

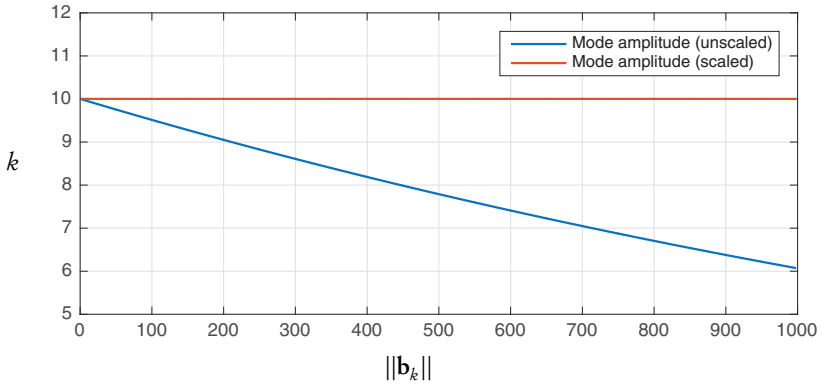

Figure 8.7. *DMD mode amplitude corresponding to damped mode with and without scaling by* $|\lambda|^k$.

8.1.3 ▪ Generalized DMD power spectrum

In general, DMD will be applied to high-dimensional data containing modes that both oscillate and either grow or decay, so that eigenvalues are either real or come in complex conjugate pairs (for real data): $\lambda = \alpha \pm i\beta$.

In the early DMD papers [235, 64], the matrix $\mathbf{X}$ was decomposed into the product of scaled DMD modes and a Vandermonde matrix that captured the iterative exponentiation of the DMD eigenvalues:

$$\mathbf{X} \approx \underbrace{\begin{bmatrix} | & | & \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots \\ | & | & \end{bmatrix}}_{\text{DMD modes}} \underbrace{\begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{m-2} \\ 1 & \lambda_2 & \cdots & \lambda_2^{m-2} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}}_{\text{Vandermonde matrix}}. \tag{8.3}$$

In this formulation, the norm of a mode $\mathbf{v}_k$ yields the mode amplitude: $b_k = \|\mathbf{v}_k\|_2$. In Jovanović et al. [149], the amplitude of the DMD modes is explicitly separated into a product of the normalized DMD modes $\mathbf{\Phi}$, a diagonal matrix of mode amplitudes, and the Vandermonde eigenvalue matrix:

$$\mathbf{X} \approx \begin{bmatrix} | & | & \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} b_1 & 0 & \cdots \\ 0 & b_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{m-2} \\ 1 & \lambda_2 & \cdots & \lambda_2^{m-2} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}. \tag{8.4}$$

In Chapter 9, we will use this formulation in conjunction with the sparsity-promoting ℓ_1-minimization to find a small subset of important modes.

From the expression in (8.4), we see that we may use the first column of $\mathbf{X}$ to estimate the mode amplitudes:

$$\mathbf{x}_1 \approx \begin{bmatrix} | & | & \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \end{bmatrix}. \tag{8.5}$$

In MATLAB, we solve for $\mathbf{b}$ by

```
>> b = Phi\X(:,1);
```

The pseudoinverse is used because the columns of $\mathbf{\Phi}$ are not orthogonal, so it is impossible to *project* the data onto $\mathbf{\Phi}$ using the inner product. Note that the pseudoinverse of $\mathbf{\Phi}$ may be quite computationally expensive, especially if the number of DMD modes r is large. This may be bypassed if the Schmid definition of DMD modes [247], $\mathbf{\Phi} = \mathbf{UW}$, is used, by applying the inverse of the square matrix $\mathbf{W}$ to $\mathbf{x}_1$ in POD coefficients $\boldsymbol{\alpha}_1$:

$$\mathbf{x}_1 \approx \mathbf{\Phi b} \tag{8.6a}$$
$$\implies \mathbf{U}\boldsymbol{\alpha}_1 \approx \mathbf{UWb} \tag{8.6b}$$
$$\implies \mathbf{b} \approx \mathbf{W}^{-1}\boldsymbol{\alpha}_1. \tag{8.6c}$$

In MATLAB, this becomes

```
Phi = U(:,1:r)*W;  % DMD Mode from Schmid JFM, 2010
b = Phi\X(:,1);  % mode amplitude

% approximation using POD subspace
alpha1 = S(1:r,1:r)*V(1,1:r)';
bPOD = W\alpha1;

norm(U(:,1:r)'*X(:,1) - S(1:r,1:r)*V(1,1:r)')
norm(b-bPOD,2)
```

It is also possible to determine $\mathbf{b}$ using inexpensive computations on the POD subspace with the new formulation of DMD modes from Tu et al. [290], $\mathbf{\Phi} = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$, although this is more involved:

$$\mathbf{x}_1 \approx \mathbf{\Phi b} \tag{8.7a}$$
$$\implies \mathbf{U}\boldsymbol{\alpha}_1 \approx \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{Wb} \tag{8.7b}$$
$$\implies \boldsymbol{\alpha}_1 \approx \mathbf{U}^*\mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{Wb} \tag{8.7c}$$
$$\implies \boldsymbol{\alpha}_1 \approx \tilde{\mathbf{A}}\mathbf{Wb} \tag{8.7d}$$
$$\implies \boldsymbol{\alpha}_1 \approx \mathbf{W}\mathbf{\Lambda}\mathbf{b} \tag{8.7e}$$
$$\implies \mathbf{b} \approx (\mathbf{W}\mathbf{\Lambda})^{-1}\boldsymbol{\alpha}_1. \tag{8.7f}$$

Again, in MATLAB, this is given by

```
Phi = X(:,2:end)*V(:,1:r)*inv(S(1:r,1:r))*W;

% mode amplitude  (based on first snapshot)
b = Phi\X(:,1);

% approximation using POD subspace
alpha1 = S(1:r,1:r)*V(1,1:r)';
bPOD = (Atilde*W)\alpha1;
bPOD = (W*Lambda)\alpha1;  % equivalent

norm(b-bPOD,2)
```

It is interesting to note that the mode amplitude $\mathbf{b}$ is based on the first snapshot $\mathbf{x}_1$ if $\mathbf{x}_1$ is not approximately in the column space of $\mathbf{X}'$. In this case $||\mathbf{\Phi b} - \mathbf{x}_1||_2$ will be large. However, for most data sets, such as the flow past a cylinder of video data, the mode amplitude $\mathbf{b}$ will be a good approximation.

8.2 ▪ Truncating data and singular value thresholding

The DMD algorithm is based on the SVD of the data. Deciding how many singular values to keep and how many to truncate is one of the most important choices. This choice depends on many factors, including the origin and quality of the data and the dynamic importance of low-energy modes. It has been shown that in many problems, such as control of acoustic tones in a fluid flow over an open cavity, low-energy fluid coherent structures are important for balanced input-output models suitable for control [233, 236, 238, 140]. Typically, numerical simulation data is of a high enough fidelity to extract these low-energy modes, whereas experimental data may be corrupted with measurement noise, making it unclear where to truncate.

In the past, many different truncation criteria have been presented, including looking for "elbows" in the singular value plot on a logarithmic scale or choosing a singular value threshold to retain 99% or 99.9% of the total variance in the data. These methods, known as *hard threshold* techniques, retain the r largest singular values and singular vectors, discarding the remaining data. Another alternative is *soft thresholding* [81], where low-energy mode amplitudes are smoothly reduced and eventually truncated entirely.

A recent breakthrough by Gavish and Donoho [106] provides a theoretical foundation for the *optimal* truncation of singular values in the case of a data matrix of measurements, $\mathbf{Y}$, that contains signal $\mathbf{X}$ plus additive white noise error $\mathbf{X}_n$:

$$\mathbf{Y} = \mathbf{X} + \eta \mathbf{X}_n, \tag{8.8}$$

where $\mathbf{X}_n$ is a matrix composed of identical, independently distributed Gaussian variables with zero mean and unit standard deviation.

In the case that $\mathbf{Y}$ is a square matrix of size $n \times n$ and the noise magnitude η is known, [106] shows that the optimal singular value threshold τ to denoise is given by

$$\tau = \frac{4}{\sqrt{3}} \sqrt{n} \eta. \tag{8.9}$$

When $\mathbf{Y}$ is a rectangular matrix of size $n \times m$, then the threshold is given by

$$\tau = \lambda(\beta) \sqrt{n} \eta, \tag{8.10}$$

where $\beta = m/n$ is the aspect ratio of the matrix $\mathbf{Y}$ and $\lambda(\beta)$ is given by

$$\lambda(\beta) = \left(2(\beta + 1) + \frac{8\beta}{(\beta + 1) + (\beta^2 + 14\beta + 1)^{1/2}} \right)^{1/2}. \tag{8.11}$$

In most cases, the noise magnitude η is *unknown* and must be estimated directly from the SVD of $\mathbf{Y}$. The median singular value σ_{median} is used to estimate the SVD of the white noise matrix $\eta \mathbf{X}_n$. The singular values of $\mathbf{Y}$ that are larger than the largest-noise singular value are kept, while any singular values below the noise threshold are discarded. In this case, the optimal threshold τ is given by

$$\tau = \omega(\beta) \sigma_{\text{median}}, \tag{8.12}$$

where $\omega(\beta) = \lambda(\beta)/\mu_\beta$ and μ_β is the median of the Marčenko–Pastur distribution [106]. Conveniently, the value of $\omega(\beta)$ is computed by a function in a MATLAB code supplement to [106] at http://purl.stanford.edu/vg705qn9070.

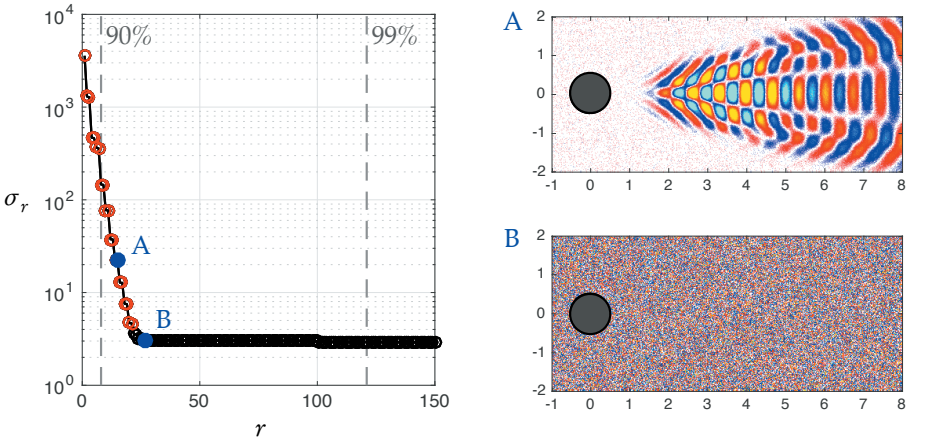

Figure 8.8. *Singular values for cylinder wake with additive noise (left). The singular values that are kept by the optimal hard threshold are shown in red. The* 90 % *and* 99 % *thresholds are shown in gray. An SVD mode that is kept* (A) *is compared with an SVD mode that is discarded by truncation* (B).

8.2.1 ▪ Singular value threshold for flow past a cylinder

We demonstrate the optimal singular value hard threshold on the fluid cylinder wake example from Chapter 2. In particular, a small amount of white noise is added to the data matrix $\mathbf{X}$, and we demonstrate that the threshold τ from (8.12) identifies a reasonable singular value threshold.

Figure 8.8 shows the SVD for the noisy flow data, and the singular values retained after applying the hard threshold criterion are colored red. An example left singular vector (POD mode) that is kept is compared with a mode that is discarded. There is a clear difference in quality and signal-to-noise ratio in the modes that are kept versus the modes that are discarded. Algorithm **8.4** determines the optimal threshold value and plots the singular values.

***ALGORITHM* 8.4. Code to compute optimal hard singular value threshold from Gavish and Donoho [106] (`SVHT_cylinder.m`).**

```
Y = VORTALL + .01*randn(size(VORTALL));

[U,S,V] = svd(Y,'econ');      % SVD matrix
beta = size(Y,2)/size(Y,1);   % aspect ratio of matrix
sigma = diag(S);              % extract singular values
% optimal threshold tau
tau = optimal_SVHT_coef(beta,0)*median(sigma);

semilogy(sigma,'k-o','LineWidth',1.2)
hold on
semilogy(sigma(sigma>tau),'ro','LineWidth',1.2)
```

8.3 ▪ Compensating for noise in the DMD spectrum

As mentioned earlier, the DMD spectrum has been shown to be quite sensitive to measurement noise [77, 129]. In numerical simulations, it is often possible to obtain high-fidelity snapshot data, but in experiments measurement noise must be accounted for. This section highlights the excellent work of Dawson et al. [77] and Hemati et al. [129] to remove the systematic bias introduced in the DMD eigenvalues due to noisy measurements. DMD has also been used to remove noise in robotic applications [26].

In general, we consider a set of direct measurements $\mathbf{y}$ of the state $\mathbf{x}$ with additive noise $\mathbf{n}_k$:

$$\mathbf{y}_k = \mathbf{x}_k + \mathbf{n}_k. \tag{8.13}$$

In the standard DMD formulation, both data snapshot matrices $\mathbf{Y}$ and $\mathbf{Y}'$ have noise and may be expressed in terms of the true noiseless data $\mathbf{X}$ and $\mathbf{X}'$ plus noise matrices $\mathbf{X}_n$ and $\mathbf{X}'_n$:

$$\left.\begin{aligned}\mathbf{Y} &= \mathbf{X} + \mathbf{X}_n \\ \mathbf{Y}' &= \mathbf{X}' + \mathbf{X}'_n\end{aligned}\right\} \quad \Longrightarrow \quad \mathbf{A}_\mathbf{Y} = \mathbf{Y}'\mathbf{Y}^\dagger. \tag{8.14}$$

The original DMD algorithm solves for $\mathbf{A}_\mathbf{Y}$ using standard regression, which assumes noise on the $\mathbf{Y}'$ matrix but *not* on the $\mathbf{Y}$ matrix. This introduces a systematic bias, making the DMD eigenvalues of $\mathbf{A}_\mathbf{Y}$ differ from the true eigenvalues of the noiseless system $\mathbf{A}_\mathbf{X}$.

In Dawson et al. [77], the matrix $\mathbf{A}_\mathbf{Y}$ is related to the underlying true system $\mathbf{A}_\mathbf{X}$ by the expression

$$\mathbb{E}(\mathbf{A}_\mathbf{Y}) = \mathbf{A}_\mathbf{X}\left(\mathbf{I} - \mathbb{E}(\mathbf{X}_n\mathbf{X}_n^*)(\mathbf{X}\mathbf{X}^*)^{-1}\right)^{-1} \tag{8.15a}$$

$$\approx \mathbf{A}_\mathbf{X}\left(\mathbf{I} - m\eta^2\Sigma_\mathbf{Y}^{-2}\right)^{-1} \tag{8.15b}$$

$$\approx \mathbf{A}_\mathbf{X}\left(\mathbf{I} + m\eta^2\Sigma_\mathbf{Y}^{-2}\right). \tag{8.15c}$$

Recall that m is the number of snapshots and η^2 is the variance of the noise. Each of the approximations in (8.15) relies on a small noise magnitude, and it is also assumed that the noise matrix $\mathbf{X}_n$ is composed of identical, independently distributed Gaussian entries, as in § 8.2. It is noted in [77] that when the number of measurements is greater than the number of snapshots, $n > m$, then $\mathbf{X}\mathbf{X}^*$ is not invertible, so it is necessary to work in a truncated POD subspace where $r < m$. If the noise variance η^2 is known, it is possible to manually debias the expectation of the matrix $\mathbf{A}_\mathbf{Y}$ and remove the extra $\mathbf{I} + m\eta^2\Sigma_\mathbf{Y}^{-2}$ term. However, this removes the expectation bias but does not reduce the variance of the DMD spectrum due to noise. This is addressed in the following sections.

To see the systematic bias, we have constructed a simple linear dynamical system, inspired by the system in Dawson et al. [77]:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} \\ -1/\sqrt{3} & 2/\sqrt{3} \end{bmatrix}\mathbf{x}_k. \tag{8.16}$$

The eigenvalues of this discrete-time system form a complex conjugate pair on the complex unit circle, $\lambda = \sqrt{3}/2 \pm 1/2$, indicating that the system is neutrally stable. Figure 8.9 shows the eigenvalues obtained by the standard DMD method (blue dots)

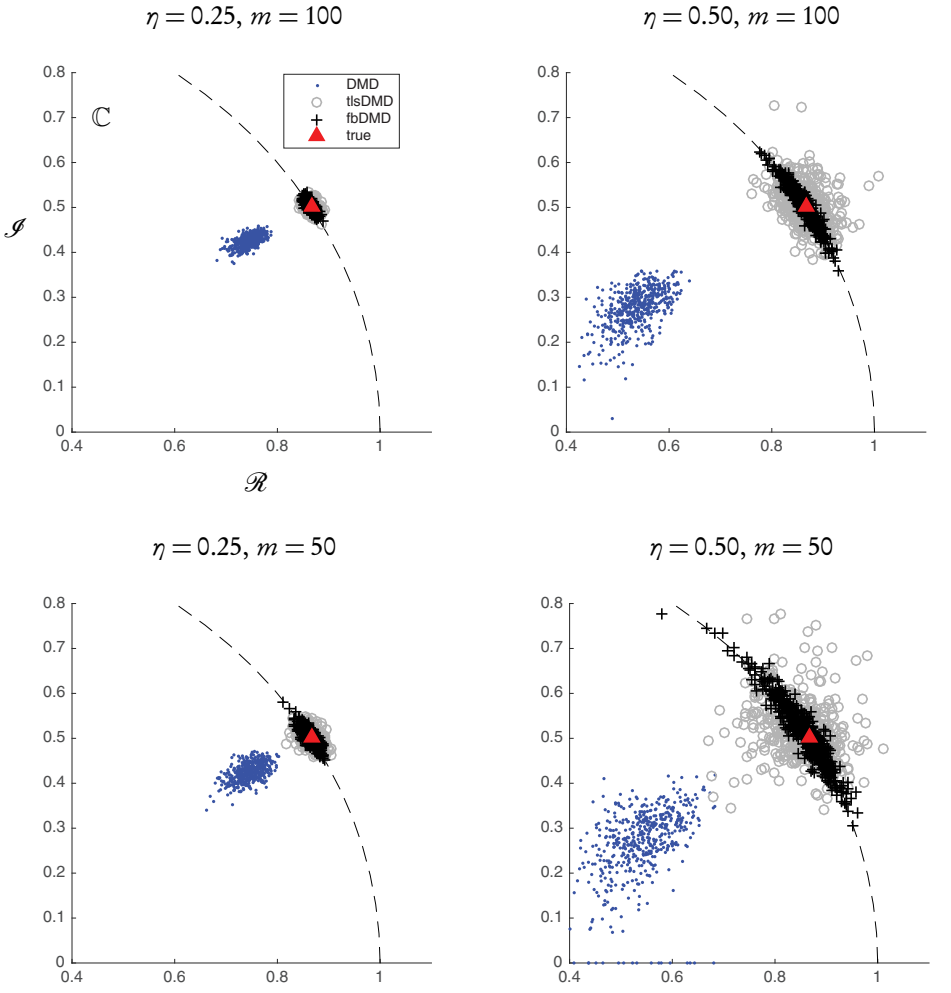

Figure 8.9. *Ensembles of DMD eigenvalue spectra using standard DMD, total-least-squares DMD (tlsDMD), and forward-backward DMD (fbDMD) denoising for various noise magnitudes η and numbers of snapshots m.*

with additive sensor noise. A systematic bias is introduced, forcing the ensemble eigenvalue cloud into the unit circle, making the eigenvalues appear stable. Increasing the number m of snapshots collected decreases the variance but does not change the bias in the mean.

***ALGORITHM* 8.5. Generate dynamics (`LDS_DMD_eig.m`).**

```
% parameters
sig = 0.5;  % noise standard deviation
niterations = 500;  % size of random ensemble
m = 100;  % number of snapshots
```

```
A = [1, 1; -1, 2]/sqrt(3);
n = 2;

X = zeros(n, m);
X(:, 1) = [0.5, 1]';

for k = 2:m,
    X(:, k) = A * X(:, k-1);
end;
```

8.3.1 ▪ Forward/backward DMD

In the first method to debias the DMD spectrum, we consider the forward-backward DMD method of Dawson et al. [77]. A strength of this approach is its simple implementation and interpretation. The main observation is that if we swap the data matrices $\mathbf{X}$ and $\mathbf{X}'$ and compute DMD on the pair $(\mathbf{X}', \mathbf{X})$, we obtain a new propagator matrix $\mathbf{A}_{\mathbf{X}}^{b}$ for the *backward* dynamical system in reverse time. This backward propagator matrix should be the inverse of the direct matrix if this data was generated by a linear dynamical system, so that $\mathbf{A}_{\mathbf{X}} = \left(\mathbf{A}_{\mathbf{X}}^{b}\right)^{-1}$.

However, when noise is added, and this procedure is repeated on the noisy measurement matrices $(\mathbf{Y}, \mathbf{Y}')$ and $(\mathbf{Y}', \mathbf{Y})$, the resulting matrices $\mathbf{A}_{\mathbf{Y}}$ and $\mathbf{A}_{\mathbf{Y}}^{b}$ are only *approximately* inverses. In fact, both of these matrices will have the same type of eigenvalue bias, and Dawson et al. [77] go on to show that it is possible to obtain a debiased estimate of the underlying matrix $\mathbf{A}_{\mathbf{X}}$ from

$$\mathbf{A}_{\mathbf{X}}^{2} \approx \mathbf{A}_{\mathbf{Y}} \left(\mathbf{A}_{\mathbf{Y}}^{b}\right)^{-1} \tag{8.17a}$$

$$\implies \mathbf{A}_{\mathbf{X}} \approx \left(\mathbf{A}_{\mathbf{Y}} \left(\mathbf{A}_{\mathbf{Y}}^{b}\right)^{-1}\right)^{1/2}. \tag{8.17b}$$

This is implemented in Algorithm **8.6**, and the results are shown in Figure 8.9. The matrix square root is not a unique operation, so there is a freedom of choice in (8.17b).

It is important to note that not only does this algorithm reduce the expectation bias of the DMD eigenvalues with noisy data (i.e., the mean of the eigenvalue cloud is properly centered on the true eigenvalue), but the variance is significantly reduced. Again, it is important to note that if the state dimension is large, $n \gg m$, it may be advantageous to work entirely in a low-dimensional POD subspace.

***ALGORITHM* 8.6. Compute forward-backward DMD, as in Dawson et al. [77] (`DMD_eig.m`).**

```
    case 'fbdmd',
        % compute forward DMD
        [U, S, V] = svd(Y, 'econ');
        f_Atilde = U' * Yp * V / S;

        % compute backward DMD
        [U, S, V] = svd(Yp, 'econ');
```

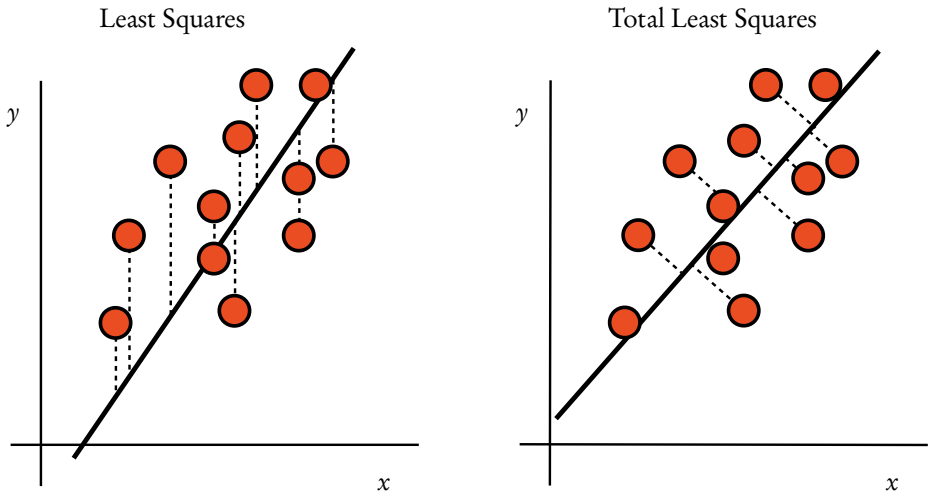

Figure 8.10. *Schematic illustrating the geometric difference between the least-squares algorithm (left) and the total least-squares algorithm (right).*

```
b_Atilde = U' * Y * V / S;

% estimate forward/backward DMD
Atilde = (f_Atilde * inv(b_Atilde)) ^ 0.5;
```

8.3.2 ▪ Total least-squares DMD

Now, we consider the total least-squares DMD algorithm of Hemati et al. [129]. This algorithm is based on the above observation that standard DMD solves for $\mathbf{A_Y}$ using regression, which assumes noise on $\mathbf{Y}$ but not on $\mathbf{Y}'$.

The total least-squares algorithm [117, 118, 94, 95, 291, 115] is an extension of the standard SVD-based least-squares regression [114, 54, 116, 119] that assumes noise on both matrices:

$$\mathbf{Y}' = \mathbf{A_Y}\mathbf{Y} \tag{8.18a}$$

$$\implies \mathbf{X}' + \mathbf{X}'_n = \mathbf{A_Y}(\mathbf{X} + \mathbf{X}_n). \tag{8.18b}$$

In particular, the total least-squares algorithm solves an optimization that finds the best-fit solution $\mathbf{A_Y}$ that simultaneously minimizes the Frobenius norms of $\mathbf{X}'_n$ and $\mathbf{X}_n$. This is shown schematically in Figure 8.10.

This optimization turns out to be relatively straightforward. We stack the matrices $\mathbf{Y}$ and $\mathbf{Y}'$ into a new matrix $\mathbf{Z}$ and take the SVD:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Y} \\ \mathbf{Y}' \end{bmatrix} = \mathbf{U_Z}\mathbf{\Sigma_Z}\mathbf{V_Z^*}. \tag{8.19}$$

Now, the left singular vectors in $\mathbf{U_Z}$ are partitioned into four equal quadrants:

$$\mathbf{U_Z} = \begin{bmatrix} \mathbf{U}_{\mathbf{Z},a} & \mathbf{U}_{\mathbf{Z},b} \\ \mathbf{U}_{\mathbf{Z},c} & \mathbf{U}_{\mathbf{Z},d} \end{bmatrix}. \tag{8.20}$$

Finally, the new total least-squares estimate for $\mathbf{A_Y}$ is given by

$$\mathbf{A_Y} = \mathbf{U}_{Z,c}\mathbf{U}_{Z,a}^{\dagger}. \tag{8.21}$$

This algorithm is shown in Algorithm **8.7**, and the resulting debiased eigenvalues are shown in Figure 8.9.

***ALGORITHM* 8.7. Code to compute total least-squares DMD, as in Hemati et al. [129] (`DMD_eig.m`).**

```
    case 'tlsdmd',
        % stack
        Z = [Y; Yp];

        % tls DMD
        [U, ~, ~] = svd(Z, 'econ');
        U11 = U(1:r, 1:r);
        U21 = U(r+1:end, 1:r);

        Atilde = U21 * inv(U11);
```