9 Balanced Models for Control

Many systems of interest are exceedingly high dimensional, making them difficult to characterize. High dimensionality also limits controller robustness due to significant computational time delays. For example, for the governing equations of fluid dynamics, the resulting discretized equations may have millions or billions of degrees of freedom, making them expensive to simulate. Thus, significant effort has gone into obtaining reduced-order models that capture the most relevant mechanisms and are suitable for feedback control.

Unlike reduced-order models based on proper orthogonal decomposition (see Chapters 11 and 12), which order modes based on energy content in the data, here we will discuss a class of *balanced* reduced-order models that employ a different inner product to order modes based on input–output energy. Thus, only modes that are both highly controllable and highly observable are selected, making balanced models ideal for control applications. In this chapter we also describe related procedures for model reduction and system identification, depending on whether or not the user starts with a high-fidelity model or simply has access to measurement data.

9.1 Model Reduction and System Identification

In many nonlinear systems, it is still possible to use linear control techniques. For example, in fluid dynamics there are numerous success stories of linear model-based flow control [27, 180, 94], for example to delay transition from laminar to turbulent flow in a spatially developing boundary layer, to reduce skin-friction drag in wall turbulence, and to stabilize the flow past an open cavity. However, many linear control approaches do not scale well to large state spaces, and they may be prohibitively expensive to enact for real-time control on short timescales. Thus, it is often necessary to develop low-dimensional approximations of the system for use in real-time feedback control.

There are two broad approaches to obtain reduced-order models (ROMs): First, it is possible to start with a high-dimensional system, such as the discretized Navier–Stokes equations, and project the dynamics onto a low-dimensional subspace identified, for example, using proper orthogonal decomposition (POD; Chapter 11) [57, 251] and Galerkin projection [441, 53]. There are numerous variations to this procedure, including the discrete empirical interpolation methods (DEIM; Section 12.5) [127, 419], gappy POD (Section 12.1) [179], balanced proper orthogonal decomposition (BPOD; Section 9.2) [554, 458], and many more. The second approach is to collect data from a simulation or an experiment and identify a low-rank model using data-driven techniques. This approach is typically called system identification, and is often preferred for control design because of the relative ease of implementation. Examples include the dynamic mode decomposition

https://doi.org/10.1017/9781108380690.010 Published online by Cambridge University Press

(DMD; Section 7.2) [472, 456, 535, 317], the eigensystem realization algorithm (ERA; Section 9.3) [272, 351], the observer–Kalman filter identification (OKID; Section 9.3) [273, 428, 271], NARMAX [59], and the sparse identification of nonlinear dynamics (SINDy; Section 7.3) [95].

After a linear model has been identified, either by model reduction or system identification, it may then be used for model-based control design. However, there are a number of issues that may arise in practice, as linear model-based control might not work for a large class of systems. First, the system being modeled may be strongly nonlinear, in which case the linear approximation might only capture a small portion of the dynamic effects. Next, the system may be stochastically driven, so that the linear model will average out the relevant fluctuations. Finally, when control is applied to the full system, the attractor dynamics may change, rendering the linearized model invalid. Exceptions include the stabilization of fixed points, where feedback control rejects nonlinear disturbances and keeps the system in a neighborhood of the fixed point where the linearized model is accurate. There are also methods for system identification and model reduction that are nonlinear, involve stochasticity, and change with the attractor. However, these methods are typically advanced and they also may limit the available machinery from control theory.

9.2 Balanced Model Reduction

The high dimensionality and short timescales associated with complex systems may render the model-based control strategies described in Chapter 8 infeasible for real-time applications. Moreover, obtaining $\mathcal{H}_2$ and $\mathcal{H}_\infty$ optimal controllers may be computationally intractable, as they involve either solving a high-dimensional Riccati equation, or an expensive iterative optimization. As has been demonstrated throughout this book, even if the ambient dimension is large, there may still be a few dominant coherent structures that characterize the system. Reduced-order models provide efficient, low-dimensional representations of these most relevant mechanisms. Low-order models may then be used to design efficient controllers that can be applied in realtime, even for high-dimensional systems. An alternative is to develop controllers based on the full-dimensional model and then apply model reduction techniques directly to the full controller [209, 194, 410, 128].

Model reduction is essentially data reduction that respects the fact that the data is generated by a dynamic process. If the dynamical system is a linear time-invariant (LTI) input–output system, then there is a wealth of machinery available for model reduction, and performance bounds may be quantified. The techniques explored here are based on the singular value decomposition (SVD; Chapter 1) [212, 106, 211], and the minimal realization theory of Ho and Kalman [247, 388]. The general idea is to determine a hierarchical modal decomposition of the system state that may be truncated at some model order, only keeping the coherent structures that are most important for control.

The Goal of Model Reduction

Consider a high-dimensional system, depicted schematically in Fig. 9.1,

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{9.1a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{9.1b}$$

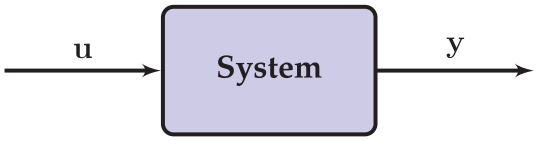

Figure 9.1 Input–output system. A control-oriented reduced-order model will capture the transfer function from **u** to **y**.

for example from a spatially discretized simulation of a PDE. The primary goal of model reduction is to find a coordinate transformation $\mathbf{x} = \mathbf{\Psi}\tilde{\mathbf{x}}$ giving rise to a related system $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}, \tilde{\mathbf{D}})$ with similar input–output characteristics,

$$\frac{d}{dt}\tilde{\mathbf{x}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}} + \tilde{\mathbf{B}}\mathbf{u}, \tag{9.2a}$$

$$\mathbf{y} = \tilde{\mathbf{C}}\tilde{\mathbf{x}} + \tilde{\mathbf{D}}\mathbf{u}, \tag{9.2b}$$

in terms of a state $\tilde{\mathbf{x}} \in \mathbb{R}^r$ with reduced dimension, $r \ll n$. Note that **u** and **y** are the same in (9.1) and (9.2) even though the system states are different. Obtaining the projection operator $\mathbf{\Psi}$ will be the focus of this section.

As a motivating example, consider the following simplified model:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 10^{-10} \end{bmatrix} u \tag{9.3a}$$

$$y = \begin{bmatrix} 1 & 10^{-10} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{9.3b}$$

In this case, the state x_2 is barely controllable and barely observable. Simply choosing $\tilde{x} = x_1$ will result in a reduced-order model that faithfully captures the input–output dynamics. Although the choice $\tilde{x} = x_1$ seems intuitive in this extreme case, many model reduction techniques would erroneously favor the state $\tilde{x} = x_2$, since it is more lightly damped. Throughout this section, we will investigate how to accurately and efficiently find the transformation matrix $\mathbf{\Psi}$ that best captures the input–output dynamics.

The proper orthogonal decomposition [57, 251] from Chapter 11 provides a transform matrix $\mathbf{\Psi}$, the columns of which are modes that are ordered based on energy content.[1] POD has been widely used to generate ROMs of complex systems, many for control, and it is guaranteed to provide an optimal low-rank basis to capture the maximal energy or variance in a data set. However, it may be the case that the most energetic modes are nearly uncontrollable or unobservable, and therefore may not be relevant for control. Similarly, in many cases the most controllable and observable state directions may have very low energy; for example, acoustic modes typically have very low energy, yet they mediate the dominant input–output dynamics in many fluid systems. The rudder on a ship provides a good analogy: although it accounts for a small amount of the total energy, it is dynamically important for control.

[1] When the training data consists of velocity fields, for example from a high-dimensional discretized fluid system, then the singular values literally indicate the kinetic energy content of the associated mode. It is common to refer to POD modes as being ordered by *energy* content, even in other applications, although *variance* is more technically correct.

Instead of ordering modes based on energy, it is possible to determine a hierarchy of modes that are most controllable and observable, therefore capturing the most input–output information. These modes give rise to *balanced* models, giving equal weighting to the controllability and observability of a state via a coordinate transformation that makes the controllability and observability Gramians equal and diagonal. These models have been extremely successful, although computing a balanced model using traditional methods is prohibitively expensive for high-dimensional systems. In this section, we describe the balancing procedure, as well as modern methods for efficient computation of balanced models. A computationally efficient suite of algorithms for model reduction and system identification may be found in [50].

A balanced reduced-order model should map inputs to outputs as faithfully as possible for a given model order r. It is therefore important to introduce an *operator norm* to quantify how similarly (9.1) and (9.2) act on a given set of inputs. Typically, we take the infinity norm of the difference between the transfer functions $\mathbf{G}(s)$ and $\mathbf{G}_r(s)$ obtained from the full system (9.1) and reduced system (9.2), respectively. This norm is given by:

$$\|\mathbf{G}\|_\infty \triangleq \max_\omega \sigma_1\left(\mathbf{G}(i\omega)\right). \tag{9.4}$$

See Section 8.8 for a primer on transfer functions. To summarize, we seek a reduced-order model (9.2) of low order, $r \ll n$, so the operator norm $\|\mathbf{G}-\mathbf{G}_r\|_\infty$ is small.

Change of Variables in Control Systems

The balanced model reduction problem may be formulated in terms of first finding a coordinate transformation

$$\mathbf{x} = \mathbf{T}\mathbf{z}, \tag{9.5}$$

that hierarchically orders the states in $\mathbf{z}$ in terms of their ability to capture the input–output characteristics of the system. We will begin by considering an invertible transformation $\mathbf{T} \in \mathbb{R}^{n\times n}$, and then provide a method to compute just the first r columns, which will comprise the transformation $\boldsymbol{\Psi}$ in (9.2). Thus, it will be possible to retain only the first r most controllable/observable states, while truncating the rest. This is similar to the change of variables into eigenvector coordinates in (8.18), except that we emphasize controllability and observability rather than characteristics of the dynamics.

Substituting $\mathbf{Tz}$ into (9.1) gives:

$$\frac{d}{dt}\mathbf{T}\mathbf{z} = \mathbf{A}\mathbf{T}\mathbf{z} + \mathbf{B}\mathbf{u} \tag{9.6a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{T}\mathbf{z} + \mathbf{D}\mathbf{u}. \tag{9.6b}$$

Finally, multiplying (9.6a) by $\mathbf{T}^{-1}$ yields:

$$\frac{d}{dt}\mathbf{z} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{z} + \mathbf{T}^{-1}\mathbf{B}\mathbf{u} \tag{9.7a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{T}\mathbf{z} + \mathbf{D}\mathbf{u}. \tag{9.7b}$$

This results in the following transformed equations:

$$\frac{d}{dt}\mathbf{z} = \hat{\mathbf{A}}\mathbf{z} + \hat{\mathbf{B}}\mathbf{u} \tag{9.8a}$$

$$\mathbf{y} = \hat{\mathbf{C}}\mathbf{z} + \mathbf{D}\mathbf{u}, \tag{9.8b}$$

where $\hat{\mathbf{A}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$, $\hat{\mathbf{B}} = \mathbf{T}^{-1}\mathbf{B}$, and $\hat{\mathbf{C}} = \mathbf{C}\mathbf{T}$. Note that when the columns of $\mathbf{T}$ are orthonormal, the change of coordinates becomes:

$$\frac{d}{dt}\mathbf{z} = \mathbf{T}^*\mathbf{A}\mathbf{T}\mathbf{z} + \mathbf{T}^*\mathbf{B}\mathbf{u} \tag{9.9a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{T}\mathbf{z} + \mathbf{D}\mathbf{u}. \tag{9.9b}$$

Gramians and Coordinate Transformations

The controllability and observability Gramians each establish an inner product on state space in terms of how controllable or observable a given state is, respectively. As such, Gramians depend on the particular choice of coordinate system and will transform under a change of coordinates. In the coordinate system $\mathbf{z}$ given by (9.5), the controllability Gramian becomes:

$$\hat{\mathbf{W}}_c = \int_0^\infty e^{\hat{\mathbf{A}}\tau}\hat{\mathbf{B}}\hat{\mathbf{B}}^* e^{\hat{\mathbf{A}}^*\tau}\, d\tau \tag{9.10a}$$

$$= \int_0^\infty e^{\mathbf{T}^{-1}\mathbf{A}\mathbf{T}\tau}\mathbf{T}^{-1}\mathbf{B}\mathbf{B}^*\mathbf{T}^{-*}e^{\mathbf{T}^*\mathbf{A}^*\mathbf{T}^{-*}\tau}\, d\tau \tag{9.10b}$$

$$= \int_0^\infty \mathbf{T}^{-1}e^{\mathbf{A}\tau}\mathbf{T}\mathbf{T}^{-1}\mathbf{B}\mathbf{B}^*\mathbf{T}^{-*}\mathbf{T}^*e^{\mathbf{A}^*\tau}\mathbf{T}^{-*}\, d\tau \tag{9.10c}$$

$$= \mathbf{T}^{-1}\left(\int_0^\infty e^{\mathbf{A}\tau}\mathbf{B}\mathbf{B}^*e^{\mathbf{A}^\tau}\, d\tau\right)\mathbf{T}^{-*} \tag{9.10d}$$

$$= \mathbf{T}^{-1}\mathbf{W}_c\mathbf{T}^{-*}. \tag{9.10e}$$

Note that here we introduce $\mathbf{T}^{-*} := \left(\mathbf{T}^{-1}\right)^* = (\mathbf{T}^*)^{-1}$. The observability Gramian transforms similarly:

$$\hat{\mathbf{W}}_o = \mathbf{T}^*\mathbf{W}_o\mathbf{T}, \tag{9.11}$$

which is an exercise for the reader. Both Gramians transform as tensors (i.e., in terms of the transform matrix $\mathbf{T}$ and its transpose, rather than $\mathbf{T}$ and its inverse), which is consistent with them inducing an inner product on state-space.

Simple Rescaling

This example, modified from Moore 1981 [388], demonstrates the ability to balance a system through a change of coordinates. Consider the system

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -10 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 10^{-3} \\ 10^3 \end{bmatrix} u \tag{9.12a}$$

$$y = \begin{bmatrix} 10^3 & 10^{-3} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{9.12b}$$

In this example, the first state x_1 is barely controllable, while the second state is barely observable. However, under the change of coordinates $z_1 = 10^3 x_1$ and $z_2 = 10^{-3}x_2$, the system becomes balanced:

$$\frac{d}{dt}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -10 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u \tag{9.13a}$$

$$y = \begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \tag{9.13b}$$

In this example, the coordinate change simply rescales the state $\mathbf{x}$. For instance, it may be that the first state had units of millimeters while the second state had units of kilometers. Writing both states in meters balances the dynamics; that is, the controllability and observability Gramians are equal and diagonal.

Balancing Transformations

Now we are ready to derive the balancing coordinate transformation $\mathbf{T}$ that makes the controllability and observability Gramians equal and diagonal:

$$\hat{\mathbf{W}}_c = \hat{\mathbf{W}}_o = \mathbf{\Sigma}. \tag{9.14}$$

First, consider the product of the Gramians from (9.10) and (9.11):

$$\hat{\mathbf{W}}_c\hat{\mathbf{W}}_o = \mathbf{T}^{-1}\mathbf{W}_c\mathbf{W}_o\mathbf{T}. \tag{9.15}$$

Plugging in the desired $\hat{\mathbf{W}}_c = \hat{\mathbf{W}}_o = \mathbf{\Sigma}$ yields

$$\mathbf{T}^{-1}\mathbf{W}_c\mathbf{W}_o\mathbf{T} = \mathbf{\Sigma}^2 \quad \Longrightarrow \quad \mathbf{W}_c\mathbf{W}_o\mathbf{T} = \mathbf{T}\mathbf{\Sigma}^2. \tag{9.16}$$

The latter expression in (9.16) is the equation for the eigendecomposition of $\mathbf{W}_c\mathbf{W}_o$, the product of the Gramians in the original coordinates. Thus, the balancing transformation $\mathbf{T}$ is related to the eigendecomposition of $\mathbf{W}_c\mathbf{W}_o$. The expression 9.16 is valid for any scaling of the eigenvectors, and the correct rescaling must be chosen to exactly balance the Gramians. In other words, there are many such transformations $\mathbf{T}$ that make the product $\hat{\mathbf{W}}_c\hat{\mathbf{W}}_o = \mathbf{\Sigma}^2$, but where the individual Gramians are not equal (for example diagonal Gramians $\hat{\mathbf{W}}_c = \mathbf{\Sigma}_c$ and $\hat{\mathbf{W}}_o = \mathbf{\Sigma}_o$ will satisfy (9.16) if $\mathbf{\Sigma}_c\mathbf{\Sigma}_o = \mathbf{\Sigma}^2$).

We will introduce the matrix $\mathbf{S} = \mathbf{T}^{-1}$ to simplify notation.

Scaling Eigenvectors for the balancing Transformation

To find the correct scaling of eigenvectors to make $\hat{\mathbf{W}}_c = \hat{\mathbf{W}}_o = \mathbf{\Sigma}$, first consider the simplified case of balancing the first diagonal element of $\mathbf{\Sigma}$. Let $\boldsymbol{\xi}_u$ denote the unscaled first column of $\mathbf{T}$, and let $\boldsymbol{\eta}_u$ denote the unscaled first row of $\mathbf{S} = \mathbf{T}^{-1}$. Then

$$\boldsymbol{\eta}_u\mathbf{W}_c\boldsymbol{\eta}_u^* = \sigma_c \tag{9.17a}$$

$$\boldsymbol{\xi}_u^*\mathbf{W}_o\boldsymbol{\xi}_u = \sigma_o. \tag{9.17b}$$

The first element of the diagonalized controllability Gramian is thus σ_c, while the first element of the diagonalized observability Gramian is σ_o. If we scale the eigenvector $\boldsymbol{\xi}_u$ by σ_s, then the inverse eigenvector $\boldsymbol{\eta}_u$ is scaled by σ_s^{-1}. Transforming via the new scaled eigenvectors $\boldsymbol{\xi}_s = \sigma_s\boldsymbol{\xi}_u$ and $\boldsymbol{\eta}_s = \sigma_s^{-1}\boldsymbol{\eta}_u$, yields:

$$\boldsymbol{\eta}_s\mathbf{W}_c\boldsymbol{\eta}_s^* = \sigma_s^{-2}\sigma_c, \tag{9.18a}$$

$$\boldsymbol{\xi}_s^*\mathbf{W}_o\boldsymbol{\xi}_s = \sigma_s^2\sigma_o. \tag{9.18b}$$

Thus, for the two Gramians to be equal,

$$\sigma_s^{-2}\sigma_c = \sigma_s^2\sigma_o \quad \Longrightarrow \quad \sigma_s = \left(\frac{\sigma_c}{\sigma_o}\right)^{1/4}. \tag{9.19}$$

To balance every diagonal entry of the controllability and observability Gramians, we first consider the unscaled eigenvector transformation $\mathbf{T}_u$ from (9.16); the subscript u simply denotes *unscaled*. As an example, we use the standard scaling in most computational software so that the columns of $\mathbf{T}_u$ have unit norm. Then both Gramians are diagonalized, but are not necessarily equal:

$$\mathbf{T}_u^{-1}\mathbf{W}_c\mathbf{T}_u^{-*} = \mathbf{\Sigma}_c \tag{9.20a}$$
$$\mathbf{T}_u^{*}\mathbf{W}_o\mathbf{T}_u = \mathbf{\Sigma}_o. \tag{9.20b}$$

The scaling that exactly balances these Gramians is then given by $\mathbf{\Sigma}_s = \mathbf{\Sigma}_c^{1/4}\mathbf{\Sigma}_o^{-1/4}$. Thus, the exact balancing transformation is given by

$$\mathbf{T} = \mathbf{T}_u\mathbf{\Sigma}_s. \tag{9.21}$$

It is possible to directly confirm that this transformation balances the Gramians:

$$(\mathbf{T}_u\mathbf{\Sigma}_s)^{-1}\mathbf{W}_c\,(\mathbf{T}_u\mathbf{\Sigma}_s)^{-*} = \mathbf{\Sigma}_s^{-1}\mathbf{T}_u^{-1}\mathbf{W}_c\mathbf{T}_u^{-*}\mathbf{\Sigma}_s^{-1} = \mathbf{\Sigma}_s^{-1}\mathbf{\Sigma}_c\mathbf{\Sigma}_s^{-1} = \mathbf{\Sigma}_c^{1/2}\mathbf{\Sigma}_o^{1/2} \tag{9.22a}$$
$$(\mathbf{T}_u\mathbf{\Sigma}_s)^{*}\mathbf{W}_o\,(\mathbf{T}_u\mathbf{\Sigma}_s) = \mathbf{\Sigma}_s\mathbf{T}_u^{*}\mathbf{W}_o\mathbf{T}_u\mathbf{\Sigma}_s = \mathbf{\Sigma}_s\mathbf{\Sigma}_o\mathbf{\Sigma}_s = \mathbf{\Sigma}_c^{1/2}\mathbf{\Sigma}_o^{1/2}. \tag{9.22b}$$

Manipulations 9.22a and 9.22b rely on the fact that diagonal matrices commute, so that $\mathbf{\Sigma}_c\mathbf{\Sigma}_o = \mathbf{\Sigma}_o\mathbf{\Sigma}_c$, etc.

Example of the Balancing Transform and Gramians

Before confronting the practical challenges associated with accurately and efficiently computing the balancing transformation, it is helpful to consider an illustrative example.

In Matlab, computing the balanced system and the balancing transformation is a simple one-line command:

```
[sysb,g,Ti,T] = balreal(sys); % Balance system
```

In this code, **T** is the transformation, **Ti** is the inverse transformation, **sysb** is the balanced system, and **g** is a vector containing the diagonal elements of the balanced Gramians.

The following example illustrates the balanced realization for a two-dimensional system. First, we generate a system and compute its balanced realization, along with the Gramians for each system. Next, we visualize the Gramians of the unbalanced and balanced systems in Fig. 9.2.

Code 9.1 Obtaining a balanced realization.

```
A = [-.75 1; -.3 -.75];
B = [2; 1];
C = [1 2];
D = 0;

sys = ss(A,B,C,D);

Wc = gram(sys,'c'); % Controllability Gramian
Wo = gram(sys,'o'); % Observability Gramian
```

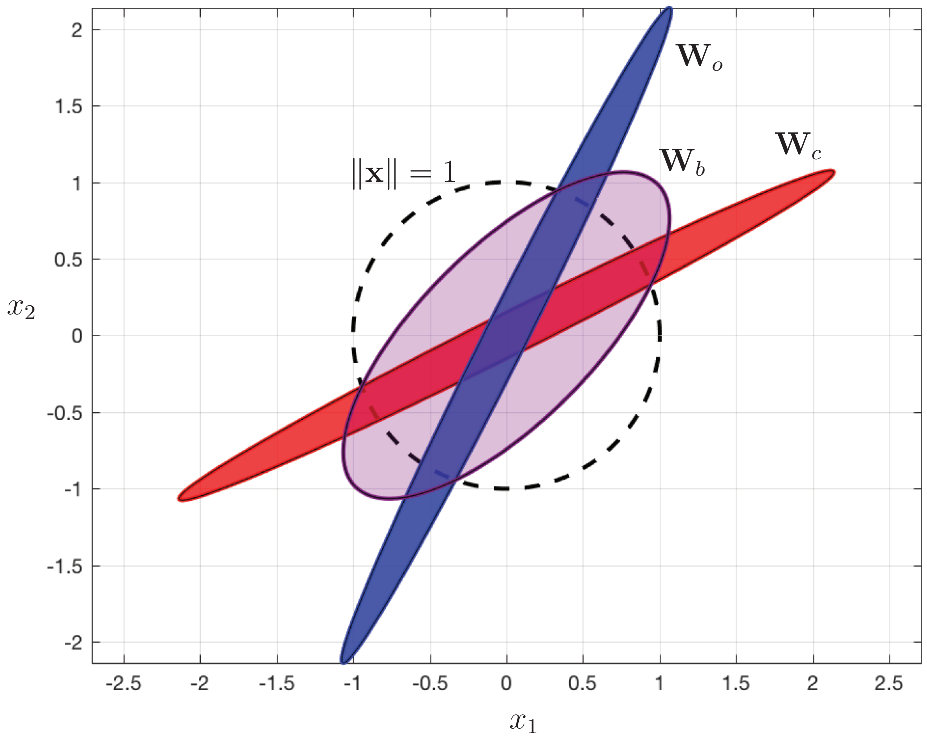

Figure 9.2 Illustration of balancing transformation on Gramians. The reachable set with unit control input is shown in red, given by $\mathbf{W}_c^{1/2}\mathbf{x}$ for $\|\mathbf{x}\| = 1$. The corresponding observable set is shown in blue. Under the balancing transformation $\mathbf{T}$, the Gramians are equal, shown in purple.

```
 [sysb,g,Ti,T] = balreal(sys); % Balance the system

BWc = gram(sysb,'c') % Balanced Gramians
BWo = gram(sysb,'o')
```

The resulting balanced Gramians are equal, diagonal, and ordered from most controllable/observable mode to least:

```
>>BWc =
    1.9439   -0.0000
   -0.0000    0.3207

>>BWo =
    1.9439    0.0000
    0.0000    0.3207
```

To visualize the Gramians in Fig. 9.2, we first recall that the distance the system can go in a direction $\mathbf{x}$ with a unit actuation input is given by $\mathbf{x}^*\mathbf{W}_c\mathbf{x}$. Thus, the controllability Gramian may be visualized by plotting $\mathbf{W}_c^{1/2}\mathbf{x}$ for $\mathbf{x}$ on a sphere with $\|\mathbf{x}\| = 1$. The observability Gramian may be similarly visualized.

In this example, we see that the most controllable and observable directions may not be well aligned. However, by a change of coordinates, it is possible to find a new direction that is the most jointly controllable and observable. It is then possible to represent the system in this one-dimensional subspace, while still capturing a significant portion of the input–output energy. If the red and blue Gramians were exactly perpendicular, so that the most controllable direction was the least observable direction, and vice versa, then the balanced Gramian would be a circle. In this case, there is no preferred state direction, and both directions are equally important for the input–output behavior.

Instead of using the **balreal** command, it is possible to manually construct the balancing transformation from the eigendecomposition of $\mathbf{W}_c\mathbf{W}_o$, as described earlier and provided in code available online.

Balanced Truncation

We have now shown that it is possible to define a change of coordinates so that the controllability and observability Gramians are equal and diagonal. Moreover, these new coordinates may be ranked hierarchically in terms of their joint controllability and observability. It may be possible to truncate these coordinates and keep only the most controllable/observable directions, resulting in a reduced-order model that faithfully captures input–output dynamics.

Given the new coordinates $\mathbf{z} = \mathbf{T}^{-1}\mathbf{x} \in \mathbb{R}^n$, it is possible to define a reduced-order state $\tilde{\mathbf{x}} \in \mathbb{R}^r$, as

$$\mathbf{z} = \begin{bmatrix} \left.\begin{matrix} z_1 \\ \vdots \\ z_r \end{matrix}\right\} \tilde{\mathbf{x}} \\ z_{r+1} \\ \vdots \\ z_n \end{bmatrix} \tag{9.23}$$

in terms of the first r most controllable and observable directions. If we partition the balancing transformation $\mathbf{T}$ and inverse transformation $\mathbf{S} = \mathbf{T}^{-1}$ into the first r modes to be retained and the last $n - r$ modes to be truncated,

$$\mathbf{T} = \begin{bmatrix} \mathbf{\Psi} & \mathbf{T}_t \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \mathbf{\Phi}^* \\ \mathbf{S}_t \end{bmatrix}, \tag{9.24}$$

then it is possible to rewrite the transformed dynamics in (9.7) as:

$$\frac{d}{dt}\begin{bmatrix} \tilde{\mathbf{x}} \\ \hline \mathbf{z}_t \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{\Phi}^*\mathbf{A}\mathbf{\Psi} & \mathbf{\Phi}^*\mathbf{A}\mathbf{T}_t \\ \hline \mathbf{S}_t\mathbf{A}\mathbf{\Psi} & \mathbf{S}_t\mathbf{A}\mathbf{T}_t \end{array}\right]\begin{bmatrix} \tilde{\mathbf{x}} \\ \hline \mathbf{z}_t \end{bmatrix} + \begin{bmatrix} \mathbf{\Phi}^*\mathbf{B} \\ \hline \mathbf{S}_t\mathbf{B} \end{bmatrix}\mathbf{u} \tag{9.25a}$$

$$\mathbf{y} = \left[\begin{array}{c|c} \mathbf{C}\mathbf{\Psi} & \mathbf{C}\mathbf{T}_t \end{array}\right]\begin{bmatrix} \tilde{\mathbf{x}} \\ \hline \mathbf{z}_t \end{bmatrix} + \mathbf{D}\mathbf{u}. \tag{9.25b}$$

In balanced truncation, the state $\mathbf{z}_t$ is simply truncated (i.e., discarded and set equal to zero), and only the $\tilde{\mathbf{x}}$ equations remain:

$$\frac{d}{dt}\tilde{\mathbf{x}} = \mathbf{\Phi}^*\mathbf{A}\mathbf{\Psi}\tilde{\mathbf{x}} + \mathbf{\Phi}^*\mathbf{B}\mathbf{u} \tag{9.26a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{\Psi}\tilde{\mathbf{x}} + \mathbf{D}\mathbf{u}. \tag{9.26b}$$

Only the first r columns of $\mathbf{T}$ and $\mathbf{S}^* = \mathbf{T}^{-*}$ are required to construct $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$, and thus computing the entire balancing transformation $\mathbf{T}$ is unnecessary. Note that the matrix $\boldsymbol{\Phi}$ here is different than the matrix of DMD modes in Section 7.2. The computation of $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ without $\mathbf{T}$ will be discussed in the following sections. A key benefit of balanced truncation is the existence of upper and lower bounds on the error of a given order truncation:

$$\textbf{Upper bound:} \quad \|\mathbf{G}-\mathbf{G}_r\|_\infty \leq 2\sum_{j=r+1}^{n}\sigma_j, \tag{9.27a}$$

$$\textbf{Lower bound:} \quad \|\mathbf{G}-\mathbf{G}_r\|_\infty > \sigma_{r+1}, \tag{9.27b}$$

where σ_j is the jth diagonal entry of the balanced Gramians. The diagonal entries of $\boldsymbol{\Sigma}$ are also known as *Hankel singular values*.

Computing Balanced Realizations

In the previous section we demonstrated the feasibility of obtaining a coordinate transformation that balances the controllability and observability Gramians. However, the computation of this balancing transformation is nontrivial, and significant work has gone into obtaining accurate and efficient methods, starting with Moore in 1981 [388], and continuing with Lall, Marsden, and Glavaški in 2002 [321], Willcox and Peraire in 2002 [554] and Rowley in 2005 [458]. For an excellent and complete treatment of balanced realizations and model reduction, see Antoulas [17].

In practice, computing the Gramians $\mathbf{W}_c$ and $\mathbf{W}_o$ and the eigendecomposition of the product $\mathbf{W}_c\mathbf{W}_o$ in (9.16) may be prohibitively expensive for high-dimensional systems. Instead, the balancing transformation may be approximated from impulse-response data, utilizing the singular value decomposition for efficient extraction of the most relevant subspaces.

We will first show that Gramians may be approximated via a snapshot matrix from impulse-response experiments/simulations. Then, we will show how the balancing transformation may be obtained from this data.

Empirical Gramians

In practice, computing Gramians via the Lyapunov equation is computationally expensive, with computational complexity of $\mathcal{O}(n^3)$. Instead, the Gramians may be approximated by full-state measurements of the discrete-time direct and adjoint systems:

$$\textit{direct}: \quad \mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \tag{9.28a}$$

$$\textit{adjoint}: \quad \mathbf{x}_{k+1} = \mathbf{A}_d^*\mathbf{x}_k + \mathbf{C}_d^*\mathbf{y}_k. \tag{9.28b}$$

(9.28a) is the discrete-time dynamic update equation from (8.21), and (9.28b) is the adjoint equation. The matrices $\mathbf{A}_d$, $\mathbf{B}_d$, and $\mathbf{C}_d$ are the discrete-time system matrices from (8.22). Note that the adjoint equation is generally nonphysical, and must be simulated; thus the methods here apply to analytical equations and simulations, but not to experimental data. An alternative formulation that does not rely on adjoint data, and therefore generalizes to experiments, will be provided in Section 9.3.

Computing the impulse-response of the direct and adjoint systems yields the following discrete-time snapshot matrices:

$$\mathcal{C}_d = \begin{bmatrix} \mathbf{B}_d & \mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{A}_d^{m_c-1}\mathbf{B}_d \end{bmatrix} \qquad \mathcal{O}_d = \begin{bmatrix} \mathbf{C}_d \\ \mathbf{C}_d\mathbf{A}_d \\ \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1} \end{bmatrix}. \tag{9.29}$$

Note that when $m_c = n$, $\mathcal{C}_d$ is the discrete-time controllability matrix and when $m_o = n$, $\mathcal{O}_d$ is the discrete-time observability matrix; however, we generally consider $m_c, m_o \ll n$. These matrices may also be obtained by sampling the continuous-time direct and adjoint systems at a regular interval Δt.

It is now possible to compute *empirical* Gramians that approximate the true Gramians without solving the Lyapunov equations in (8.42) and (8.43):

$$\mathbf{W}_c \approx \mathbf{W}_c^e = \mathcal{C}_d\mathcal{C}_d^*, \tag{9.30a}$$
$$\mathbf{W}_o \approx \mathbf{W}_o^e = \mathcal{O}_d^*\mathcal{O}_d. \tag{9.30b}$$

The empirical Gramians essentially comprise a Riemann sum approximation of the integral in the continuous-time Gramians, which becomes exact as the time-step of the discrete-time system becomes arbitrarily small and the duration of the impulse response becomes arbitrarily large. In practice, the impulse-response snapshots should be collected until the lightly-damped transients die out. The method of empirical Gramians is quite efficient, and is widely used [388, 320, 321, 554, 458]. Note that p adjoint impulse responses are required, where p is the number of outputs. This becomes intractable when there are a large number of outputs (e.g., full state measurements), motivating the output projection in the next section.

Balanced POD

Instead of computing the eigendecomposition of $\mathbf{W}_c\mathbf{W}_o$, which is an $n \times n$ matrix, it is possible to compute the balancing transformation via the singular value decomposition of the product of the snapshot matrices,

$$\mathcal{O}_d\mathcal{C}_d, \tag{9.31}$$

reminiscent of the method of snapshots from Section 1.3 [490]. This is the approach taken by Rowley [458].

First, define the generalized Hankel matrix as the product of the adjoint ($\mathcal{O}_d$) and direct ($\mathcal{C}_d$) snapshot matrices from (9.29), for the discrete-time system:

$$\mathbf{H} = \mathcal{O}_d\mathcal{C}_d = \begin{bmatrix} \mathbf{C}_d \\ \mathbf{C}_d\mathbf{A}_d \\ \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1} \end{bmatrix} \begin{bmatrix} \mathbf{B}_d & \mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{A}_d^{m_c-1}\mathbf{B}_d \end{bmatrix} \tag{9.32a}$$

$$= \begin{bmatrix} \mathbf{C}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c-1}\mathbf{B}_d \\ \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^2\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c}\mathbf{B}_d \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1}\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{m_o}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+m_o-2}\mathbf{B}_d \end{bmatrix}. \tag{9.32b}$$

Next, we factor **H** using the SVD:

$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^* = \begin{bmatrix} \tilde{\mathbf{U}} & \mathbf{U}_t \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_t \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{V}}^* \\ \mathbf{V}_t^* \end{bmatrix} \approx \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^*. \tag{9.33}$$

For a given desired model order $r \ll n$, only the first r columns of **U** and **V** are retained, along with the first $r \times r$ block of $\boldsymbol{\Sigma}$; the remaining contribution from $\mathbf{U}_t\boldsymbol{\Sigma}_t\mathbf{V}_t^*$ may be truncated. This yields a bi-orthogonal set of modes given by:

$$\textit{direct modes}: \quad \boldsymbol{\Psi} = \mathcal{C}_d\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1/2}, \tag{9.34a}$$

$$\textit{adjoint modes}: \quad \boldsymbol{\Phi} = \mathcal{O}_d^*\tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}^{-1/2}. \tag{9.34b}$$

The direct modes $\boldsymbol{\Psi} \in \mathbb{R}^{n\times r}$ and adjoint modes $\boldsymbol{\Phi} \in \mathbb{R}^{n\times r}$ are bi-orthogonal, $\boldsymbol{\Phi}^*\boldsymbol{\Psi} = \mathbf{I}_{r\times r}$, and Rowley [458] showed that they establish the change of coordinates that balance the truncated empirical Gramians. Thus, $\boldsymbol{\Psi}$ approximates the first r-columns of the full $n \times n$ balancing transformation, **T**, and $\boldsymbol{\Phi}^*$ approximates the first r-rows of the $n \times n$ inverse balancing transformation, $\mathbf{S} = \mathbf{T}^{-1}$.

Now, it is possible to project the original system onto these modes, yielding a balanced reduced-order model of order r:

$$\tilde{\mathbf{A}} = \boldsymbol{\Phi}^*\mathbf{A}_d\boldsymbol{\Psi}, \tag{9.35a}$$

$$\tilde{\mathbf{B}} = \boldsymbol{\Phi}^*\mathbf{B}_d, \tag{9.35b}$$

$$\tilde{\mathbf{C}} = \mathbf{C}_d\boldsymbol{\Psi}. \tag{9.35c}$$

It is possible to compute the reduced system dynamics in (9.35a) without having direct access to $\mathbf{A}_d$. In some cases, $\mathbf{A}_d$ may be exceedingly large and unwieldy, and instead it is only possible to evaluate the action of this matrix on an input vector. For example, in many modern fluid dynamics codes the matrix $\mathbf{A}_d$ is not actually represented, but because it is sparse, it is possible to implement efficient routines to multiply this matrix by a vector.

It is important to note that the reduced-order model in (9.35) is formulated in discrete time, as it is based on discrete-time empirical snapshot matrices. However, it is simple to obtain the corresponding continuous-time system:

```
>>sysD = ss(Atilde,Btilde,Ctilde,D,dt);   % Discrete-time
>>sysC = d2c(sysD);                       % Continuous-time
```

In this example, **D** is the same in continuous time and discrete time, and in the full-order and reduced-order models.

Note that a BPOD model may not exactly satisfy the upper bound from balanced truncation (see (9.27)) due to errors in the empirical Gramians.

Output Projection

Often, in high-dimensional simulations, we assume full-state measurements, so that $p = n$ is exceedingly large. To avoid computing $p = n$ adjoint simulations, it is possible instead to solve an output-projected adjoint equation [458]:

$$\mathbf{x}_{k+1} = \mathbf{A}_d^* \mathbf{x}_k + \mathbf{C}_d^* \tilde{\mathbf{U}} \mathbf{y} \tag{9.36}$$

where $\tilde{\mathbf{U}}$ is a matrix containing the first r singular vectors of $\mathcal{C}_d$. Thus, we first identify a low-dimensional POD subspace $\tilde{\mathbf{U}}$ from a direct impulse response, and then only perform adjoint impulse response simulations by exciting these few *POD coefficient* measurements. More generally, if $\mathbf{y}$ is high dimensional but does not measure the full state, it is possible to use a POD subspace trained on the measurements, given by the first r singular vectors $\tilde{\mathbf{U}}$ of $\mathbf{C}_d \mathcal{C}_d$. Adjoint impulse responses may then be performed in these output POD directions.

Data Collection and Stacking

The powers m_c and m_o in (9.32) signify that data must be collected until the matrices $\mathcal{C}_d$ and $\mathcal{O}_d^*$ are full rank, after which the controllable/observable subspaces have been sampled. Unless we collect data until transients decay, the true Gramians are only approximately balanced. Instead, it is possible to collect data until the Hankel matrix is full rank, balance the resulting model, and then truncate. This more efficient approach is developed in [533] and [346].

The snapshot matrices in (9.29) are generated from impulse-response simulations of the direct (9.28a) and adjoint (9.36) systems. These time-series snapshots are then interleaved to form the snapshot matrices.

Historical Note

The balanced POD method described in the previous subsection originated with the seminal work of Moore in 1981 [388], which provided a data-driven generalization of the minimal realization theory of Ho and Kalman [247]. Until then, minimal realizations were defined in terms of idealized controllable and observable subspaces, which neglected the subtlety of degrees of controllability and observability.

Moore's paper introduced a number of critical concepts that bridged the gap from theory to reality. First, he established a connection between principal component analysis (PCA) and Gramians, showing that information about degrees of controllability and observability may be mined from data via the SVD. Next, Moore showed that a balancing transformation exists that makes the Gramians equal, diagonal, and hierarchically ordered by balanced controllability and observability; moreover, he provides an algorithm to compute this transformation. This set the stage for principled model reduction, whereby states may be truncated based on their joint controllability and observability. Moore further introduced the notion of an empirical Gramian, although he didn't use this terminology. He also realized that computing $\mathbf{W}_c$ and $\mathbf{W}_o$ directly is less accurate than computing the SVD of the empirical snapshot matrices from the direct and adjoint systems, and he avoided directly computing the eigendecomposition of $\mathbf{W}_c \mathbf{W}_o$ by using these SVD transformations. In 2002, Lall, Marsden, and Glavaški in 2002 [321] generalized this theory to nonlinear systems.

One drawback of Moore's approach is that he computed the entire $n \times n$ balancing transformation, which is not suitable for exceedingly high-dimensional systems. In 2002, Willcox and Peraire [554] generalized the method to high-dimensional systems, introducing a variant based on the rank-r decompositions of $\mathbf{W}_c$ and $\mathbf{W}_o$ obtained from the direct and adjoint snapshot matrices. It is then possible to compute the eigendecomposition of $\mathbf{W}_c \mathbf{W}_o$ using efficient eigenvalue solvers without ever actually writing down the full $n \times n$ matrices. However, this approach has the drawback of requiring as many adjoint impulse-response simulations as the number of output equations, which may be exceedingly large for full-state measurements. In 2005, Rowley [458] addressed this issue by introducing the output projection, discussed previously, which limits the number of adjoint simulations to the number of relevant POD modes in the data. He also showed that it is possible to use the eigendecomposition of the product $\mathcal{O}_d\mathcal{C}_d$. The product $\mathcal{O}_d\mathcal{C}_d$ is often smaller, and these computations may be more accurate.

It is interesting to note that a nearly equivalent formulation was developed twenty years earlier in the field of system identification. The so-called eigensystem realization algorithm (ERA) [272], introduced in 1985 by Juang and Pappa, obtains equivalent balanced models without the need for adjoint data, making it useful for system identification in experiments. This connection between ERA and BPOD was established by Ma et al. in 2011 [351].

Balanced Model Reduction Example

In this example we will demonstrate the computation of balanced truncation and balanced POD models on a random state-space system with $n = 100$ states, $q = 2$ inputs, and $p = 2$ outputs. First, we generate a system in Matlab:

```
q = 2;    % Number of inputs
p = 2;    % Number of outputs
n = 100;  % State dimension
sysFull = drss(n,p,q); % Discrete random system
```

Next, we compute the Hankel singular values, which are plotted in Fig. 9.3. We see that $r = 10$ modes captures over 90% of the input–output energy.

```
hsvs = hsvd(sysFull); % Hankel singular values
```

Now we construct an exact balanced truncation model with order $r = 10$:

```
%% Exact balanced truncation
sysBT = balred(sysFull,r);   % Balanced truncation
```

The full-order system, and the balanced truncation and balanced POD models are compared in Fig. 9.4. The BPOD model is computed using Code 9.2. It can be seen that the

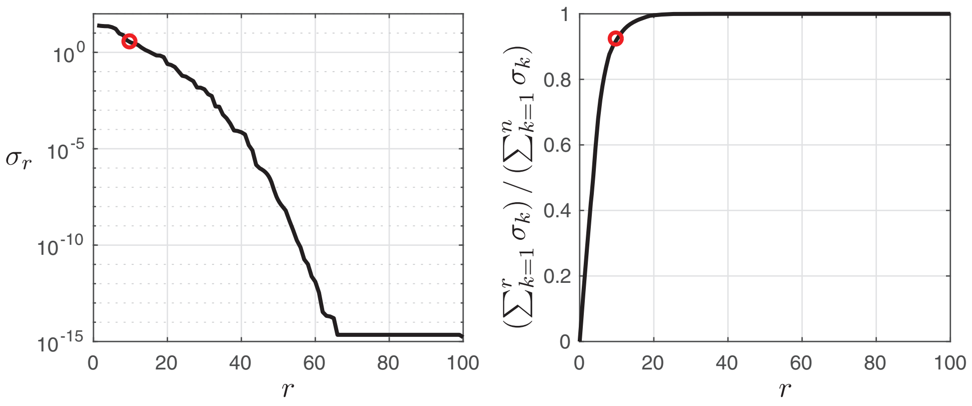

Figure 9.3 Hankel singular values (left) and cumulative energy (right) for random state space system with $n = 100$, $p = q = 2$. The first $r = 10$ HSVs contain 92.9% of the energy.

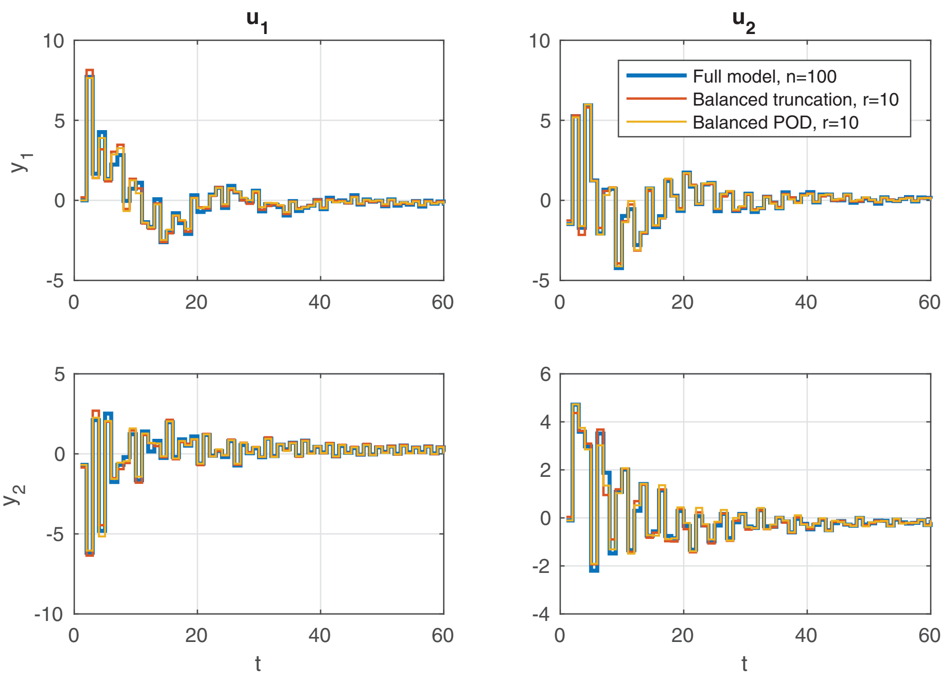

Figure 9.4 Impulse response of full-state model with $n = 100$, $p = q = 2$, along with balanced truncation and balanced POD models with $r = 10$.

balanced model accurately captures the dominant input–output dynamics, even when only 10% of the modes are kept.

Code 9.2 Balanced proper orthogonal decomposition (BPOD).

```
sysBPOD = BPOD(sysFull,sysAdj,r)

[yFull,t,xFull] = impulse(sysFull,0:1:(r*5)+1);
sysAdj = ss(sysFull.A',sysFull.C',sysFull.B',sysFull.D',-1);
[yAdj,t,xAdj] = impulse(sysAdj,0:1:(r*5)+1);
% Not the fastest way to compute, but illustrative
% Both xAdj and xFull are size m x n x 2
HankelOC = [];  % Compute Hankel matrix H=OC
for i=2:size(xAdj,1) % Start at 2 to avoid the D matrix
    Hrow = [];
    for j=2:size(xFull,1)
        Ystar = permute(squeeze(xAdj(i,:,:)),[2 1]);
        MarkovParameter = Ystar*squeeze(xFull(j,:,:));
        Hrow = [Hrow MarkovParameter];
    end
    HankelOC = [HankelOC; Hrow];
end
[U,Sig,V] = svd(HankelOC);
Xdata = [];
Ydata = [];
for i=2:size(xFull,1)  % Start at 2 to avoid the D matrix
    Xdata = [Xdata squeeze(xFull(i,:,:))];
    Ydata = [Ydata squeeze(xAdj(i,:,:))];
end
```

```
Phi = Xdata*V*Sig^(-1/2);
Psi = Ydata*U*Sig^(-1/2);
Ar = Psi(:,1:r)'*sysFull.a*Phi(:,1:r);
Br = Psi(:,1:r)'*sysFull.b;
Cr = sysFull.c*Phi(:,1:r);
Dr = sysFull.d;
sysBPOD = ss(Ar,Br,Cr,Dr,-1);
```

9.3 System Identification

In contrast to model reduction, where the system model ($\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$) was known, system identification is purely data-driven. System identification may be thought of as a form of machine learning, where an input–output map of a system is learned from training data in a representation that generalizes to data that was not in the training set. There is a vast literature on methods for system identification [271, 338], and many of the leading methods are based on a form of dynamic regression that fits models based on data, such as the DMD from Section 7.2. For this section, we consider the eigensystem realization algorithm (ERA) and observer-Kalman filter identification (OKID) methods because of their connection to balanced model reduction [388, 458, 351, 535] and their successful application in high-dimensional systems such as vibration control of aerospace structures and closed-loop flow control [27, 26, 261]. The ERA/OKID procedure is also applicable to multiple-input, multiple-output (MIMO) systems. Other methods include the autoregressive moving average (ARMA) and autoregressive moving average with exogenous inputs (ARMAX) models [552, 72], the nonlinear autoregressive-moving average with exogenous inputs (NARMAX) [59] model, and the SINDy method from Section 7.3.

Eigensystem Realization Algorithm

The eigensystem realization algorithm produces low-dimensional linear input–output models from sensor measurements of an impulse response experiment, based on the “minimal realization" theory of Ho and Kalman [247]. The modern theory was developed to identify structural models for various spacecraft [272], and it has been shown by Ma *et al.* [351] that ERA models are equivalent to BPOD models[2]. However, ERA is based entirely on impulse response measurements and does not require prior knowledge of a model.

We consider a discrete-time system, as described in Section 8.2:

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k \tag{9.37a}$$

$$\mathbf{y}_k = \mathbf{C}_d\mathbf{x}_k + \mathbf{D}_d\mathbf{u}_k. \tag{9.37b}$$

A discrete-time delta function input in the actuation $\mathbf{u}$:

$$\mathbf{u}_k^\delta \triangleq \mathbf{u}^\delta(k\Delta t) = \begin{cases} \mathbf{I}, & k = 0 \\ \mathbf{0}, & k = 1, 2, 3, \cdots \end{cases} \tag{9.38}$$

[2] BPOD and ERA models both balance the empirical Gramians and approximate balanced truncation [388] for high-dimensional systems, given a sufficient volume of data.

gives rise to a discrete-time impulse response in the sensors $\mathbf{y}$:

$$\mathbf{y}_k^\delta \triangleq \mathbf{y}^\delta(k\Delta t) = \begin{cases} \mathbf{D}_d, & k = 0 \\ \mathbf{C}_d\mathbf{A}_d^{k-1}\mathbf{B}_d, & k = 1, 2, 3, \cdots . \end{cases} \tag{9.39}$$

In an experiment or simulation, typically q impulse responses are performed, one for each of the q separate input channels. The output responses are collected for each impulsive input, and at a given time-step k, the output vector in response to the j-th impulsive input will form the j-th column of $\mathbf{y}_k^\delta$. Thus, each of the $\mathbf{y}_k^\delta$ is a $p \times q$ matrix $\mathbf{CA}^{k-1}\mathbf{B}$. Note that the system matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ don't actually need to exist, as the method in the next section is purely data-driven.

The Hankel matrix $\mathbf{H}$ from (9.32), is formed by stacking shifted time-series of impulse-response measurements into a matrix, as in the HAVOK method from Section 7.5:

$$\mathbf{H} = \begin{bmatrix} \mathbf{y}_1^\delta & \mathbf{y}_2^\delta & \cdots & \mathbf{y}_{m_c}^\delta \\ \mathbf{y}_2^\delta & \mathbf{y}_3^\delta & \cdots & \mathbf{y}_{m_c+1}^\delta \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{m_o}^\delta & \mathbf{y}_{m_o+1}^\delta & \cdots & \mathbf{y}_{m_c+m_o-1}^\delta \end{bmatrix} \tag{9.40a}$$

$$= \begin{bmatrix} \mathbf{C}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c-1}\mathbf{B}_d \\ \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^2\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c}\mathbf{B}_d \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1}\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{m_o}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+m_o-2}\mathbf{B}_d \end{bmatrix}. \tag{9.40b}$$

The matrix $\mathbf{H}$ may be constructed purely from measurements $\mathbf{y}^\delta$, without separately constructing $\mathcal{O}_d$ and $\mathcal{C}_d$. Thus, we do not need access to adjoint equations.

Taking the SVD of the Hankel matrix yields the dominant temporal patterns in the time-series data:

$$\mathbf{H} = \mathbf{U\Sigma V}^* = \begin{bmatrix} \tilde{\mathbf{U}} & \mathbf{U}_t \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_t \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{V}}^* \\ \mathbf{V}_t^* \end{bmatrix} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{9.41}$$

The small small singular values in $\mathbf{\Sigma}_t$ are truncated, and only the first r singular values in $\tilde{\mathbf{\Sigma}}$ are retained. The columns of $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are *eigen*-time-delay coordinates.

Until this point, the ERA algorithm closely resembles the BPOD procedure from Section 9.2. However, we don't require direct access to $\mathcal{O}_d$ and $\mathcal{C}_d$ or the system $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ to construct the direct and adjoint balancing transformations. Instead, with sensor measurements from an impulse-response experiment, it is also possible to create a second, shifted Hankel matrix $\mathbf{H}'$:

$$\mathbf{H}' = \begin{bmatrix} \mathbf{y}_2 & \mathbf{y}_3^\delta & \cdots & \mathbf{y}_{m_c+1}^\delta \\ \mathbf{y}_3^\delta & \mathbf{y}_4^\delta & \cdots & \mathbf{y}_{m_c+2}^\delta \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{m_o+1}^\delta & \mathbf{y}_{m_o+2}^\delta & \cdots & \mathbf{y}_{m_c+m_o}^\delta \end{bmatrix} \tag{9.42a}$$

$$= \begin{bmatrix} \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^2\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c}\mathbf{B}_d \\ \mathbf{C}_d\mathbf{A}_d^2\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^3\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+1}\mathbf{B}_d \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o}\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{m_o+1}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+m_o-1}\mathbf{B}_d \end{bmatrix} = \boldsymbol{\mathcal{O}}_d\mathbf{A}\boldsymbol{\mathcal{C}}_d. \tag{9.42b}$$

Based on the matrices $\mathbf{H}$ and $\mathbf{H}'$, we are able to construct a reduced-order model as follows:

$$\tilde{\mathbf{A}} = \tilde{\boldsymbol{\Sigma}}^{-1/2}\tilde{\mathbf{U}}^*\mathbf{H}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1/2}; \tag{9.43a}$$

$$\tilde{\mathbf{B}} = \tilde{\boldsymbol{\Sigma}}^{1/2}\tilde{\mathbf{V}}^* \begin{bmatrix} \mathbf{I}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}; \tag{9.43b}$$

$$\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{I}_q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}^{1/2}. \tag{9.43c}$$

Here $\mathbf{I}_p$ is the $p \times p$ identity matrix, which extracts the first p columns, and $\mathbf{I}_q$ is the $q \times q$ identity matrix, which extracts the first q rows. Thus, we express the input–output dynamics in terms of a reduced system with a low-dimensional state $\tilde{\mathbf{x}} \in \mathbb{R}^r$:

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{B}}\mathbf{u} \tag{9.44a}$$

$$\mathbf{y} = \tilde{\mathbf{C}}\tilde{\mathbf{x}}_k. \tag{9.44b}$$

$\mathbf{H}$ and $\mathbf{H}'$ are constructed from impulse response simulations/experiments, without the need for storing direct or adjoint snapshots, as in other balanced model reduction techniques. However, if full-state snapshots are available, for example, by collecting velocity fields in simulations or PIV experiments, it is then possible to construct direct modes. These full-state snapshots form $\mathcal{C}_d$, and modes can be constructed by:

$$\boldsymbol{\Psi} = \mathcal{C}_d\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1/2}. \tag{9.45}$$

These modes may then be used to approximate the full-state of the high-dimensional system from the low-dimensional model in (9.44) by:

$$\mathbf{x} \approx \boldsymbol{\Psi}\tilde{\mathbf{x}}. \tag{9.46}$$

If enough data is collected when constructing the Hankel matrix $\mathbf{H}$, then ERA balances the empirical controllability and observability Gramians, $\boldsymbol{\mathcal{O}}_d\boldsymbol{\mathcal{O}}_d^*$ and $\boldsymbol{\mathcal{C}}_d^*\boldsymbol{\mathcal{C}}_d$. However, if less data is collected, so that lightly damped transients do not have time to decay, then ERA will only approximately balance the system. It is instead possible to collect just enough data so that the Hankel matrix $\mathbf{H}$ reaches numerical full-rank (i.e., so that remaining singular values are below a threshold tolerance), and compute an ERA model. The resulting ERA model will typically have a relatively low order, given by the numerical rank of the controllability and observability subspaces. It may then be possible to apply exact balanced truncation to this smaller model, as is advocated in [533] and [346].

The code to compute ERA is provided in Code 9.3.

Code 9.3 Eigensystem realization algorithm.

```
function [Ar,Br,Cr,Dr,HSVs] = ERA(YY,m,n,nin,nout,r)
 for i=1:nout
     for j=1:nin
         Dr(i,j) = YY(i,j,1);
         Y(i,j,:) = YY(i,j,2:end);
     end
 end

% Yss = Y(1,1,end);
% Y = Y-Yss;
% Y(i,j,k)::
% i refers to i-th output
% j refers to j-th input
% k refers to k-th timestep

% nin,nout number of inputs and outputs
% m,n dimensions of Hankel matrix
% r, dimensions of reduced model

assert(length(Y(:,1,1))==nout);
assert(length(Y(1,:,1))==nin);
assert(length(Y(1,1,:))>=m+n);

for i=1:m
    for j=1:n
        for Q=1:nout
            for P=1:nin
                H(nout*i-nout+Q,nin*j-nin+P) = Y(Q,P,i+j-1);
                H2(nout*i-nout+Q,nin*j-nin+P) = Y(Q,P,i+j);
            end
        end
    end
end

[U,S,V] = svd(H,'econ');
Sigma = S(1:r,1:r);
Ur = U(:,1:r);
Vr = V(:,1:r);
Ar = Sigma^(-.5)*Ur'*H2*Vr*Sigma^(-.5);
Br = Sigma^(-.5)*Ur'*H(:,1:nin);
Cr = H(1:nout,:)*Vr*Sigma^(-.5);
HSVs = diag(S);
```

Observer Kalman Filter Identification

OKID was developed to complement the ERA for lightly damped experimental systems with noise [273]. In practice, performing isolated impulse response experiments is challenging, and the effect of measurement noise can contaminate results. Moreover, if there is a large separation of timescales, then a tremendous amount of data must be collected to use ERA. This section poses the general problem of approximating the impulse response from arbitrary input–output data. Typically, one would identify reduced-order models according to the following general procedure:

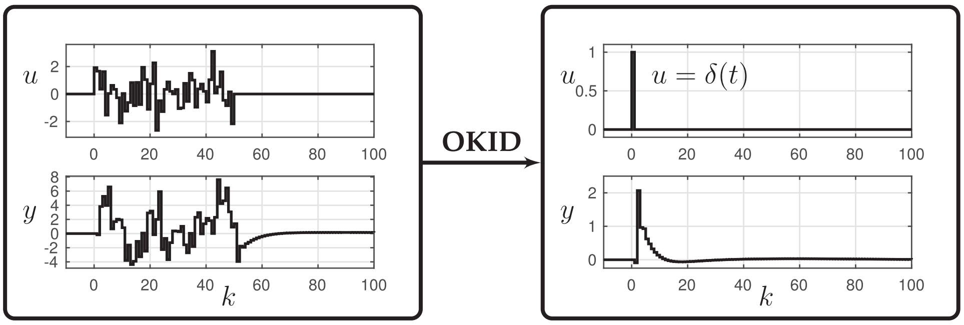

Figure 9.5 Schematic overview of OKID procedure. The output of OKID is an impulse response that can be used for system identification via ERA.

1. Collect the output in response to a pseudo-random input.
2. This information is passed through the OKID algorithm to obtain the de-noised linear impulse response.
3. The impulse response is passed through the ERA to obtain a reduced-order state-space system.

The output $\mathbf{y}_k$ in response to a general input signal $\mathbf{u}_k$, for zero initial condition $\mathbf{x}_0 = \mathbf{0}$, is given by:

$$\mathbf{y}_0 = \mathbf{D}_d\mathbf{u}_0 \tag{9.47a}$$

$$\mathbf{y}_1 = \mathbf{C}_d\mathbf{B}_d\mathbf{u}_0 + \mathbf{D}_d\mathbf{u}_1 \tag{9.47b}$$

$$\mathbf{y}_2 = \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d\mathbf{u}_0 + \mathbf{C}_d\mathbf{B}_d\mathbf{u}_1 + \mathbf{D}_d\mathbf{u}_2 \tag{9.47c}$$

$$\cdots$$

$$\mathbf{y}_k = \mathbf{C}_d\mathbf{A}_d^{k-1}\mathbf{B}_d\mathbf{u}_0 + \mathbf{C}_d\mathbf{A}_d^{k-2}\mathbf{B}_d\mathbf{u}_1 + \cdots + \mathbf{C}_d\mathbf{B}_d\mathbf{u}_{k-1} + \mathbf{D}_d\mathbf{u}_k. \tag{9.47d}$$

Note that there is no $\mathbf{C}$ term in the expression for $\mathbf{y}_0$ since there is zero initial condition $\mathbf{x}_0 = \mathbf{0}$. This progression of measurements $\mathbf{y}_k$ may be further simplified and expressed in terms of impulse-response measurements $\mathbf{y}_k^\delta$:

$$\underbrace{\begin{bmatrix}\mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_m\end{bmatrix}}_{\mathcal{S}} = \underbrace{\begin{bmatrix}\mathbf{y}_0^\delta & \mathbf{y}_1^\delta & \cdots & \mathbf{y}_m^\delta\end{bmatrix}}_{\mathcal{S}^\delta} \underbrace{\begin{bmatrix}\mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_m \\ \mathbf{0} & \mathbf{u}_0 & \cdots & \mathbf{u}_{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{u}_0\end{bmatrix}}_{\mathcal{B}}. \tag{9.48}$$

It is often possible to invert the matrix of control inputs, $\mathcal{B}$, to solve for the Markov parameters $\mathcal{S}^\delta$. However, $\mathcal{B}$ may either be un-invertible, or inversion may be ill-conditioned. In addition, $\mathcal{B}$ is large for lightly damped systems, making inversion computationally expensive. Finally, noise is not optimally filtered by simply inverting $\mathcal{B}$ to solve for the Markov parameters.

The OKID method addresses each of these issues. Instead of the original discrete-time system, we now introduce an optimal observer system:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}_d\hat{\mathbf{x}}_k + \mathbf{K}_f\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right) + \mathbf{B}_d\mathbf{u}_k \tag{9.49a}$$

$$\hat{\mathbf{y}}_k = \mathbf{C}_d\hat{\mathbf{x}}_k + \mathbf{D}_d\mathbf{u}_k, \tag{9.49b}$$

which may be re-written as:

$$\hat{\mathbf{x}}_{k+1} = \underbrace{(\mathbf{A}_d - \mathbf{K}_f\mathbf{C}_d)}_{\bar{\mathbf{A}}_d}\hat{\mathbf{x}}_k + \underbrace{\begin{bmatrix}\mathbf{B}_d - \mathbf{K}_f\mathbf{D}_d, & \mathbf{K}_f\end{bmatrix}}_{\bar{\mathbf{B}}_d}\begin{bmatrix}\mathbf{u}_k \\ \mathbf{y}_k\end{bmatrix}. \tag{9.50}$$

Recall from earlier that if the system is observable, it is possible to place the poles of $\mathbf{A}_d - \mathbf{K}_f\mathbf{C}_d$ anywhere we like. However, depending on the amount of noise in the measurements, the magnitude of process noise, and uncertainty in our model, there are *optimal* pole locations that are given by the *Kalman filter* (recall Section 8.5). We may now solve for the *observer Markov parameters* $\bar{\mathcal{S}}^\delta$ of the system in (9.50) in terms of measured inputs and outputs according to the following algorithm from [273]:

1. Choose the number of observer Markov parameters to identify, l.

2. Construct the data matrices here:

$$\mathcal{S} = \begin{bmatrix}\mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_l & \cdots & \mathbf{y}_m\end{bmatrix} \tag{9.51}$$

$$\mathcal{V} = \begin{bmatrix}\mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_l & \cdots & \mathbf{u}_m \\ \mathbf{0} & \mathbf{v}_0 & \cdots & \mathbf{v}_{l-1} & \cdots & \mathbf{v}_{m-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{v}_0 & \cdots & \mathbf{v}_{m-l}\end{bmatrix} \tag{9.52}$$

where $\mathbf{v}_i = \begin{bmatrix}\mathbf{u}_i^T & \mathbf{y}_i^T\end{bmatrix}^T$.

The matrix $\mathcal{V}$ resembles $\mathcal{B}$, except that is has been augmented with the outputs $\mathbf{y}_i$. In this way, we are working with a system that is augmented to include a Kalman filter. We are now identifying the observer Markov parameters of the *augmented* system, $\bar{\mathcal{S}}^\delta$, using the equation $\mathcal{S} = \bar{\mathcal{S}}^\delta\mathcal{V}$. It will be possible to identify these observer Markov parameters from the data and then extract the impulse response (Markov parameters) of the original system.

3. Identify the matrix $\bar{\mathcal{S}}^\delta$ of observer Markov parameters by solving $\mathcal{S} = \bar{\mathcal{S}}^\delta\mathcal{V}$ for $\bar{\mathcal{S}}^\delta$ using the right pseudo-inverse of $\mathcal{V}$ (i.e., SVD).

4. Recover system Markov parameters, $\mathcal{S}^\delta$, from the observer Markov parameters, $\bar{\mathcal{S}}^\delta$:

(a) Order the observer Markov parameters $\bar{\mathcal{S}}^\delta$ as:

$$\bar{\mathcal{S}}_0^\delta = \mathbf{D}, \tag{9.53}$$

$$\bar{\mathcal{S}}_k^\delta = \begin{bmatrix}(\bar{\mathcal{S}}^\delta)_k^{(1)} & (\bar{\mathcal{S}}^\delta)_k^{(2)}\end{bmatrix} \text{ for } k \geq 1, \tag{9.54}$$

where $(\bar{\mathcal{S}}^\delta)_k^{(1)} \in \mathbb{R}^{q\times p}$, $(\bar{\mathcal{S}}^\delta)_k^{(2)} \in \mathbb{R}^{q\times q}$, and $\mathbf{y}_0^\delta = \bar{\mathcal{S}}_0^\delta = \mathbf{D}$.

(b) Reconstruct system Markov parameters:

$$\mathbf{y}_k^\delta = (\bar{\mathcal{S}}^\delta)_k^{(1)} + \sum_{i=1}^{k}(\bar{\mathcal{S}}^\delta)_i^{(2)}\mathbf{y}_{k-i}^\delta \text{ for } k \geq 1. \tag{9.55}$$

Thus, the OKID method identifies the Markov parameters of a system augmented with an asymptotically stable Kalman filter. The system Markov parameters are extracted from the observer Markov parameters by (9.55). These system Markov parameters approximate the impulse response of the system, and may be used directly as inputs to the ERA algorithm. A code to compute OKID is provided in Code 9.4.

ERA/OKID has been widely applied across a range of system identification tasks, including to identify models of aeroelastic structures and fluid dynamic systems. There are numerous extensions of the ERA/OKID methods. For example, there are generalizations for linear parameter varying (LPV) systems and systems linearized about a limit cycle.

Code 9.4 Observer Kalman filter identification (OKID).

```
function H = OKID(y,u,r)
% Inputs: y (sampled output), u (sampled input), r (order)
% Output: H (Markov parameters)

% Step 0, check shapes of y,u
p = size(y,1);  % p is the number of outputs
m = size(y,2);  % m is the number of output samples
q = size(u,1);  % q is the number of inputs

% Step 1, choose impulse length l (5 times system order r)
l = r*5;

% Step 2, form y, V, solve for observer Markov params, Ybar
V = zeros(q + (q+p)*l,m);
for i=1:m
    V(1:q,i) = u(1:q,i);
end
for i=2:l+1
    for j=1:m+1-i
        vtemp = [u(:,j);y(:,j)];
        V(q+(i-2)*(q+p)+1:q+(i-1)*(q+p),i+j-1) = vtemp;
    end
end
Ybar = y*pinv(V,1.e-3);

% Step 3, isolate system Markov parameters H
D = Ybar(:,1:q);  % Feed-through term (D) is first term
for i=1:l
    Ybar1(1:p,1:q,i) = Ybar(:,q+1+(q+p)*(i-1):q+(q+p)*(i-1)+q);
    Ybar2(1:p,1:q,i) = Ybar(:,q+1+(q+p)*(i-1)+q:q+(q+p)*i);
end
Y(:,:,1) = Ybar1(:,:,1) + Ybar2(:,:,1)*D;
for k=2:l
    Y(:,:,k) = Ybar1(:,:,k) + Ybar2(:,:,k)*D;
    for i=1:k-1
        Y(:,:,k) = Y(:,:,k) + Ybar2(:,:,i)*Y(:,:,k-i);
    end
end

H(:,:,1) = D;
for k=2:l+1
    H(:,:,k) = Y(:,:,k-1);
end
```

Combining ERA and OKID

Here we demonstrate ERA and OKID on the same model system from Section 9.2. Because ERA yields the same balanced models as BPOD, the reduced system responses should be the same.

First, we compute an impulse response of the full system, and use this as an input to ERA:

```
%% Obtain impulse response of full system
[yFull,t] = impulse(sysFull,0:1:(r*5)+1);
YY = permute(yFull,[2 3 1]); % Reorder to be size p x q x m
                             % (default is m x p x q)

%% Compute ERA from impulse response
mco = floor((length(yFull)-1)/2);  % m_c = m_o = (m-1)/2
[Ar,Br,Cr,Dr,HSVs] = ERA(YY,mco,mco,numInputs,numOutputs,r);
sysERA = ss(Ar,Br,Cr,Dr,-1);
```

Next, if an impulse response is unavailable, it is possible to excite the system with a random input signal and use OKID to extract an impulse response. This impulse response is then used by ERA to extract the model.

```
%% Compute random input simulation for OKID
uRandom = randn(numInputs,200);  % Random forcing input
yRandom = lsim(sysFull,uRandom,1:200)'; % Output

%% Compute OKID and then ERA
H = OKID(yRandom,uRandom,r);
mco = floor((length(H)-1)/2);  % m_c = m_o
[Ar,Br,Cr,Dr,HSVs] = ERA(H,mco,mco,numInputs,numOutputs,r);
sysERAOKID = ss(Ar,Br,Cr,Dr,-1);
```

Figure 9.6 shows the input–output data used by OKID to approximate the impulse response. The impulse responses of the resulting systems are computed via

```
[y1,t1] = impulse(sysFull,0:1:200);
[y2,t2] = impulse(sysERA,0:1:100);
[y3,t3] = impulse(sysERAOKID,0:1:100);
```

Finally, the system responses can be seen in Fig. 9.7. The low-order ERA and ERA/OKID models closely match the full model and have similar performance to the BPOD models described previously. Because ERA and BPOD are mathematically equivalent, this agreement is not surprising. However, the ability of ERA/OKID to extract a reduced-order model from the random input data in Fig. 9.6 is quite remarkable. Moreover, unlike BPOD, these methods are readily applicable to experimental measurements, as they do not require nonphysical adjoint equations.

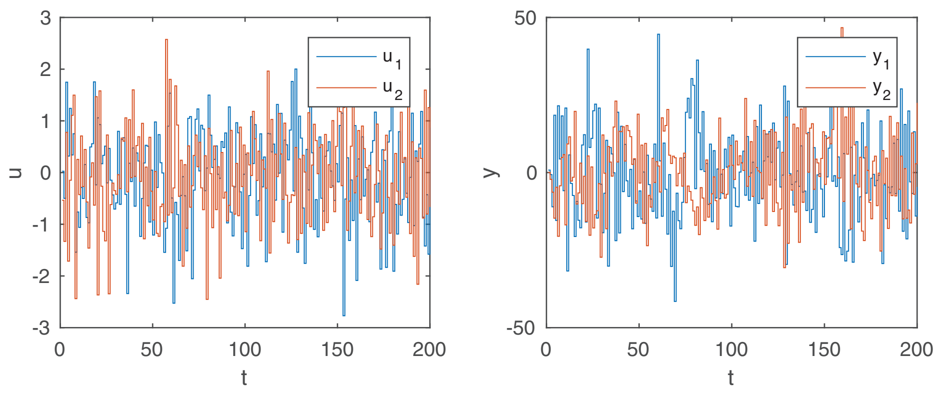

Figure 9.6 Input–output data used by OKID.

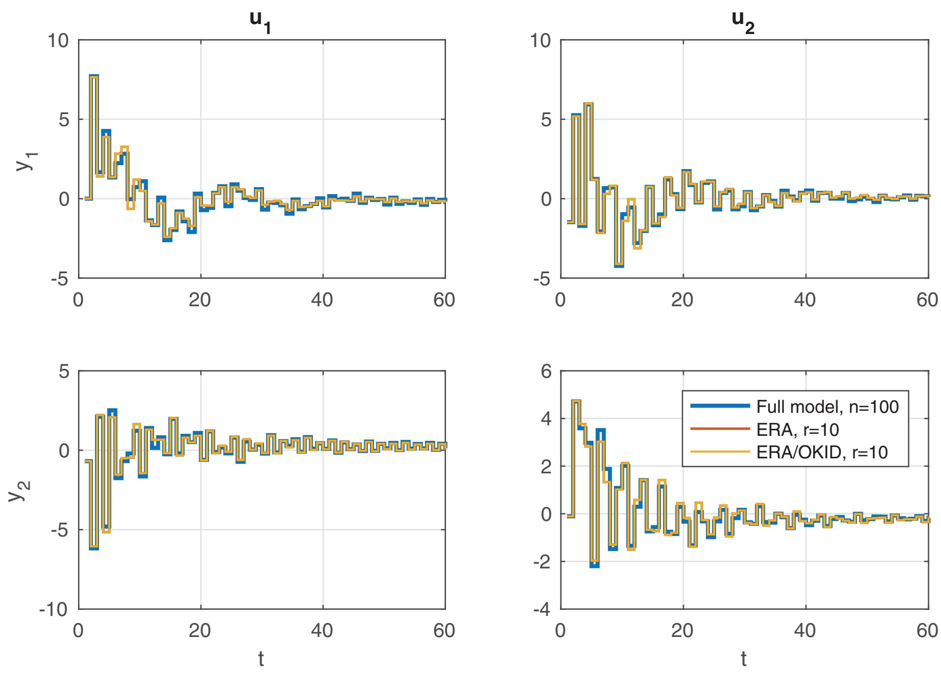

Figure 9.7 Impulse response of full-state model with $n = 100$, $p = q = 2$, along with ERA and ERA/OKID models with $r = 10$.

Suggested Reading

Papers and Reviews

(1) **Principal component analysis in linear systems: Controllability, observability, and model reduction**, by B. C. Moore, *IEEE Transactions on Automatic Control*, 1981 [388].

(2) **Identification of linear parameter varying models**, by B. Bamieh and L. Giarré, *International Journal of Robust and Nonlinear Control*, 2002 [34].

(3) **Balanced model reduction via the proper orthogonal decomposition**, by K. Willcox and J. Peraire, *AIAA Journal*, 2002 [554].

(4) **Model reduction for fluids using balanced proper orthogonal decomposition**, by C. W. Rowley, *International Journal of Bifurcations and Chaos*, 2005 [458].

(5) **An eigensystem realization algorithm for modal parameter identification and model reduction**, by J. N. Juang and R. S. Pappa, *Journal of Guidance, Control, and Dynamics*, 1985 [272].