An improved algorithm for balanced POD through an analytic treatment of impulse response tails

Jonathan H. Tu\textsuperscript{a}, Clarence W. Rowley\textsuperscript{a,*}

\textsuperscript{a}Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08544, United States

\begin{abstract}
We present a modification of the balanced proper orthogonal decomposition (balanced POD) algorithm for systems with simple impulse response tails. In this new method, we use dynamic mode decomposition (DMD) to estimate the slowly decaying eigenvectors that dominate the long-time behavior of the direct and adjoint impulse responses. This is done using a new, low-memory variant of the DMD algorithm, appropriate for large datasets. We then formulate analytic expressions for the contribution of these eigenvectors to the controllability and observability Gramians. These contributions can be accounted for in the balanced POD algorithm by simply appending the impulse response snapshot matrices (direct and adjoint, respectively) with particular linear combinations of the slow eigenvectors. Aside from these additions to the snapshot matrices, the algorithm remains unchanged. By treating the tails analytically, we eliminate the need to run long impulse response simulations, lowering storage requirements and speeding up ensuing computations. To demonstrate its effectiveness, we apply this method to two examples: the linearized complex Ginzburg-Landau equation, and the two-dimensional fluid flow past a cylinder. As expected, reduced-order models computed using an analytic tail match or exceed the accuracy of those computed using the standard balanced POD procedure, at a fraction of the cost.
\end{abstract}

\textbf{Keywords:} Balanced proper orthogonal decomposition, dynamic mode decomposition, model reduction, empirical Gramian, impulse response

1. Introduction

Model reduction is an increasingly common approach in numerical flow control studies. A typical discretization of the Navier-Stokes equation can produce a dynamical system with over $10^6$ states, making standard control design procedures prohibitive. However, the gross behavior of a fluid system can be much simpler than its state dimension would suggest. In such flows, a reduced-order model may be able to capture the dominant behavior using a relatively small number of states. For instance, the main features of vortex shedding behind a cylinder at low Reynolds numbers can be captured with a three-dimensional model \cite{1}. These low-order models can then be used for control design, and in the course of their development important underlying physical mechanisms may be discovered. Of the many model reduction techniques, balanced truncation is an especially well-suited choice for control-oriented applications. The resulting models balance the controllability and observability of a stable, linear system. (Unstable systems can be treated by decoupling the stable and unstable dynamics, as done in \cite{2,3}. ) Modes that are neither highly controllable nor highly observable are truncated. These modes are exactly those that cannot be easily affected by actuation nor easily measured with sensors. In other words, they have little effect on the input-output dynamics of the system, and as such are not useful for control design.

\textsuperscript{*}Corresponding author.
\textit{Email address:} cwrowley@princeton.edu (Clarence W. Rowley)

Unfortunately, for very high-dimensional systems, the standard balanced truncation technique is impractical, requiring the solution of large Lyapunov equations. A number of methods have been developed to iteratively solve such equations, including the classical Smith method [4] and its cyclic low-rank variant [5]. Alternatively, it is possible to avoid solving Lyapunov equations entirely. Balanced proper orthogonal decomposition (balanced POD) is a snapshot-based approximation to balanced truncation that takes this approach, making it suitable for large systems [6]. Balanced POD has been used to great effect in a variety of flow control applications. For instance, Ilak and Rowley [7] used balanced POD to accurately model the nonnormal transient growth in a linearized channel flow. Ahuja and Rowley [3] used balanced POD to design estimator-based controllers that stabilized unstable steady states of the flow past a flat plate at a high angle-of-attack. Balanced POD-based controllers were also used by Bagheri et al. [8] and Semeraro et al. [9] to suppress the growth of perturbations in a boundary layer. Dergham et al. [10] used balanced POD to model the flow over a backward-facing step, showing that a small number of input projection modes is able to capture the effect of arbitrarily placed localized actuators.

In balanced truncation, the product of the controllability and observability Gramians is used to find a transformation to a balanced coordinate system. In balanced POD, a similar computation is performed. Impulse response simulations of the direct and adjoint systems are sampled and the resulting snapshots are collected into large matrices. The product of these snapshot matrices approximates the Hankel matrix, from which an approximate balancing transformation can be found.

Two obvious sources of error are inherent in approximating the Hankel matrix this way. The first is in discretely sampling the continuously varying impulse responses. The second is in truncating the impulse responses at a finite time. In practice, both of these are dealt with by using convergence tests, with respect to the sampling frequency and simulation lengths, respectively. However, such tests can be costly for large-scale simulations.

As an alternative, we propose a method for incorporating the effect of the truncated snapshots, based on analytic considerations. For a stable, linear system, the long-time behavior of the impulse response is dominated by the system’s slowest eigenvectors alone. After enough time has elapsed, the contribution of all other eigenvectors to the state will have decayed to nearly zero. When the number of these slow eigenvectors is small, we say that the system has a simple impulse response tail. Using a snapshot-based, Arnoldi-like algorithm called dynamic mode decomposition (DMD) [11, 12], we estimate these slow eigenvectors and eigenvalues. We then express the state at the beginning of the tail as a linear combination of the slow eigenvectors. The further evolution of the state is completely characterized by the corresponding eigenvalues, so no further snapshots need to be saved, reducing the required storage space and simulation time. The contribution of the tail to the Hankel matrix can then be computed analytically, as a function of the slow eigenvectors and eigenvalues.

The remainder of this paper is organized as follows: Section 2 provides a brief introduction to empirical Gramians, balanced truncation, and the balanced POD algorithm. Section 3 builds on this theory to develop the analytic tail method. Both a complex and real formulation are derived. In Section 4 we describe a memory-efficient variant of the DMD algorithm that is appropriate for large datasets. This algorithm is used to estimate the eigenvectors and eigenvalues required to describe an impulse response tail. Finally, in Section 5 we demonstrate the effectiveness of the analytic tail method using a number of examples.

2. Background

2.1. Empirical Gramians

Consider the stable, linear system

\[
\begin{align*}
\dot{x} &= Ax + Bu \quad x \in \mathbb{R}^n, \ u \in \mathbb{R}^p \\
y &= Cx \quad y \in \mathbb{R}^q.
\end{align*}
\]

(1)

The controllability and observability Gramians are given by

\[
W_c = \int_0^\infty e^{At}BB^*e^{A^*t} \, dt \quad W_o = \int_0^\infty e^{A^*t}C^*Ce^{At} \, dt.
\]
where asterisks denote the conjugate transpose of a matrix. The controllability Gramian provides a measure of how easily a state is affected by actuation, while the observability Gramian describes how easily a state excites a sensor measurement.

Typically, the Gramians are computed by solving the Lyapunov equations

\[
AW_c + W_c A^* + BB^* = 0 \quad A^*W_o + W_o A + C^*C = 0.
\]

However, for very large systems, this can be numerically prohibitive. We can instead use data from numerical simulations to compute empirical Gramians. Suppose the system (1) has \( p \) inputs. The we can write \( B \) columnwise as

\[
B = [b_1 \, \cdots \, b_p],
\]

and similarly,

\[
u = [u_1(t) \, \cdots \, u_p(t)]^T.
\]

The response to a single impulsive input \( u_j(t) = \delta(t) \) is then given by

\[
x_j(t) = e^{At}b_j,
\]

and the controllability Gramian can be rewritten as

\[
W_c = \int_0^\infty \sum_{j=1}^p x_j(t)x_j^*(t) \, dt.
\] (2)

To evaluate the right hand side, we run numerical simulations of the impulse responses, collecting snapshots of the state at discrete times \( t_1, t_2, \ldots, t_m \). We then scale each snapshot \( x_j(t_k) \) by an appropriate quadrature weight \( \delta_k \) and collect the scaled snapshots in a data matrix

\[
X = [x_1(t_1)\sqrt{\delta_1} \, \cdots \, x_1(t_m)\sqrt{\delta_m} \, \cdots \, x_p(t_1)\sqrt{\delta_1} \, \cdots \, x_p(t_m)\sqrt{\delta_m}].
\]

The integral above can then be approximated by a quadrature sum:

\[
W_c \approx XX^*.
\] (3)

We follow a similar procedure to compute the observability Gramian. Defining the adjoint system as

\[
\dot{z} = A^*z + C^*w,
\] (4)

we again sample impulse response simulations, scale the snapshots by quadrature weights, and form a data matrix \( Y \). The observability Gramian is then approximated by the quadrature sum \( W_o \approx YY^* \).

In approximating a Gramian this way, there are two clear sources of error. First, we are sampling the continuously varying impulse response at discrete points in time. However, if the sampling rate is sufficiently fast with respect to the dynamics of the system, this error should be minimal, and can be further mitigated by using appropriate quadrature weights. The second source of error comes from truncating the impulse response at \( t_m \), when the integral in (2) is evaluated to \( t \to \infty \). For a stable system, the impulse response must eventually decay to zero. Thus if \( t_m \) is large enough, the contribution of the truncated snapshots to the Gramian will be negligible. However, it is unclear how to determine an appropriate truncation point given some a priori bound on the desired accuracy of the empirical Gramian. Furthermore, any such guideline would likely require knowledge about the eigenvalues (and possibly eigenvectors) of the system. For a large system, these may not be known, and can be expensive to compute (e.g., using an Arnoldi iteration).
2.2. Balanced truncation

Balanced truncation was developed by Moore [13] as a model reduction technique for stable, linear systems. For control applications, we are interested in the input-output dynamics of a system. As such, if a mode is difficult to affect with actuation (inputs) or hard to measure using sensors (outputs), then it is not particularly useful for control. Balanced truncation builds upon this simple idea by seeking a balanced realization of the system (1), in which the most controllable states are also the most observable. To get a reduced-order model, we simply truncate those states that are neither highly controllable nor observable. It is a standard result that if a system is both controllable and observable, then such a realization always exists (for example, see [14, 15]).

In performing balanced truncation, we compute a coordinate transformation $x = Tx$ that balances the Gramians. Under this transformation, the Gramians become

$$\tilde{W}_c = T^{-1}W_c(T^{-1})^* \quad \tilde{W}_o = T^*W_oT,$$

and in particular are equal and diagonal:

$$\tilde{W}_c = \tilde{W}_o = \Sigma.$$

The elements $\sigma_i$ of the diagonal matrix $\Sigma$ satisfy $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$, and are known as the Hankel singular values.

The Hankel singular values can be used to compute a priori bounds on the error in approximating the system (1) with a reduced-order model. Let $G(s) = C(sI - A)^{-1}B$ be the transfer function of the original system, and $G_r(s)$ be that of the reduced-order system of order $r$. Then the error is bounded below by the first truncated Hankel singular value:

$$\|G(s) - G_r(s)\|_\infty > \sigma_{r+1}. \quad (5)$$

This is a lower bound for any reduced-order approximation of $G(s)$. For a balanced truncation model, we also have an upper bound given by

$$\|G(s) - G_r(s)\|_\infty < 2 \sum_{j=r+1}^{n} \sigma_j. \quad (6)$$

(These error bounds are standard results and can be found in [14, 15] or other standard texts.)

2.3. Balanced proper orthogonal decomposition

Balanced proper orthogonal decomposition (balanced POD) was developed in [6] as an approximation to balanced truncation. It is a snapshot-based method that avoids computation of the true Gramians $W_c$ and $W_o$. Instead, it makes use of the factors $X$ and $Y$ of the empirical Gramians in analyzing the Hankel matrix $H = Y^*X$. This makes balanced POD suitable for very high-dimensional systems, whereas balanced truncation is not. If we compute the singular value decomposition (SVD) of the Hankel matrix and write it as

$$H = \begin{bmatrix} U_H & \cdots & \Sigma_H & 0 & 0 & \ldots \, W_H^* \end{bmatrix},$$

then the direct balanced POD modes are then given by

$$\Phi = XW_H \Sigma_H^{-1/2}$$

and the adjoint balanced POD modes by

$$\Psi = YU_H \Sigma_H^{-1/2}.$$
To get a reduced-order model of order \( r \), we project the system (1) onto the span of the balanced POD modes:

\[
\dot{x}_r = A_r x_r + B_r u \quad A_r = \Psi_r^* A \Phi_r \\
y_r = C_r x_r \quad B_r = \Psi_r^* B \quad C_r = C \Phi_r,
\]

where \( \Phi_r \) and \( \Psi_r \) contain only the first \( r \) columns of \( \Phi \) and \( \Psi \), respectively. The entries of the diagonal matrix \( \Sigma_H \) provide an approximation to the Hankel singular values of the system (1) and can be used to estimate the error bounds given by (5) and (6). However, even if the true Hankel singular values are known, a balanced POD-based model may not satisfy the theoretical upper error bound, as balanced POD is only an approximation of balanced truncation. (The approximation comes in taking \( X \) and \( Y \) to be factors of the empirical, rather than true, Gramians. As such, the truncation and discrete sampling of the impulse responses are again to blame.)

3. Analytic tail method

3.1. Motivation

Many of the stable, linear systems studied in fluid dynamics exhibit what we refer to as a simple impulse response tail. For such a system, the long-time behavior of the impulse response is dominated by a small set of slowly decaying eigenvectors. If we can estimate these eigenvectors and their corresponding eigenvalues, then we can write down the further evolution of the impulse response analytically, neglecting the fast eigenvectors whose contributions have already decayed to nearly zero. This reduces the required storage space for snapshots, a key consideration when dealing with large datasets. Furthermore, we can use this analytic expression to evaluate the contribution of the tail to a Gramian or to the Hankel matrix. This minimizes the error due to truncation by accounting for the effect of the impulse response(s) past the truncation point. (See Section 2.1 for a brief discussion of truncation error.)

3.2. Complex formulation

Consider the stable, linear system (1). Without loss of generality we assume a single-input system. If there are multiple inputs, the following procedure can be applied to each independently. Let \( x(t) \) be the response to an impulse in the input \( u(t) \). Suppose that at some time \( T \), we can approximate the state as a linear combination of \( M \) slow eigenvectors:

\[
x(T) = \sum_{j=1}^{M} v_j,
\]

where \( A v_j = \lambda_j v_j \) and we scale the eigenvectors \( v_j \) to subsume any multiplicative constants. Then for \( t \geq T \), the state is given by

\[
x(t) = \sum_{j=1}^{M} e^{\lambda_j (t-T)} v_j,
\]

or in matrix notation,

\[
x(t) = \begin{bmatrix} v_1 & \cdots & v_M \end{bmatrix} \begin{bmatrix} e^{\lambda_1 (t-T)} \\
\vdots \\
\vdots \\
\vdots \\
e^{\lambda_M (t-T)} \end{bmatrix}.
\tag{7}
\]

Suppose we want to compute the empirical controllability Gramian, as in (2). For our single-input system, we have

\[
W_c = \int_0^\infty x(t)x^*(t) \, dt.
\]
For $t \geq T$, we can substitute for $\mathbf{x}(t)$ from (7), yielding

$$\mathbf{x}(t)\mathbf{x}^*(t) = \mathbf{V}_M \mathbf{M}(t) \mathbf{V}_M^*,$$

where

$$\mathbf{V}_M = [\mathbf{v}_1 \cdots \mathbf{v}_M]$$

and the elements of $\mathbf{M}(t)$ are given by

$$\mathbf{M}_{j,k}(t) = e^{(\lambda_j + \lambda_k)(t-T)}.$$

Splitting the integral at $t = T$, we can rewrite the controllability Gramian using our simple tail approximation:

$$\mathbf{W}_c = \int_0^T \mathbf{x}(t)\mathbf{x}^*(t) \ dt + \mathbf{V}_M \left( \int_T^\infty \mathbf{M}(t) \ dt \right) \mathbf{V}_M^*.$$

The integral of $\mathbf{M}(t)$ can be performed element-wise, recalling that the eigenvalues of $\mathbf{A}$ all have negative real part:

$$\int_0^\infty \mathbf{M}_{j,k}(t) \ dt = \int_0^\infty e^{(\lambda_j + \lambda_k)(t-T)} \ dt$$

$$= \lim_{t_f \to \infty} \frac{1}{\lambda_j + \lambda_k} \left[ e^{(\lambda_j + \lambda_k)(t_f-T)} \right]$$

Then we can write

$$\mathbf{W}_c = \int_0^T \mathbf{x}(t)\mathbf{x}^*(t) \ dt + \mathbf{V}_M \mathbf{N} \mathbf{V}_M^*,$$

where the elements of $\mathbf{N}$ are given by

$$\mathbf{N}_{j,k} = -\frac{1}{\lambda_j + \lambda_k}.$$

We wish to express (8) in a form that lends itself to the snapshot-based formulation. In other words, we seek an expression $\mathbf{W}_c = \mathbf{X} \mathbf{X}^*$ for some data matrix $\mathbf{X}$. If we collect impulse response snapshots at discrete times $t_1, \ldots, t_m = T$, then the integral in (8) is given by

$$\int_0^T \mathbf{x}(t)\mathbf{x}^*(t) \ dt = \mathbf{X}_T \mathbf{X}_T^*,$$

where

$$\mathbf{X}_T = [\mathbf{x}(t_1)\sqrt{\delta_1} \cdots \mathbf{x}(T)\sqrt{\delta_m}].$$

Then

$$\mathbf{W}_c = \mathbf{X}_T \mathbf{X}_T^* + \mathbf{V}_M \mathbf{N} \mathbf{V}_M^*.$$

We observe that $\mathbf{N}$ is Hermitian, and as such has a unitary diagonalization $\mathbf{N} = \mathbf{U}_N \mathbf{\Lambda}_N \mathbf{U}_N^*$. If we define

$$\mathbf{\Gamma} = \mathbf{U}_N \mathbf{\Lambda}_N^{1/2},$$

we can write

$$\mathbf{V}_M \mathbf{N} \mathbf{V}_M^* = \mathbf{V}_M \mathbf{\Gamma} \mathbf{\Gamma}^* \mathbf{V}_M^*,$$

allowing us to rewrite (10) as

$$\mathbf{W}_c = \left[ \mathbf{X}_T \quad \mathbf{V}_M \mathbf{\Gamma} \right] \left[ \mathbf{X}_T \quad \mathbf{V}_M \mathbf{\Gamma} \right]^*.$$

This procedure can be applied in the same way to an impulse response of the adjoint system, yielding an improved approximation of the observability Gramian.
3.3. Real formulation

In some applications, it may not be desirable to append the snapshot matrix with complex-valued vectors, as is done in (11). For instance, in an application where the state is always real-valued (often the case in numerical simulations), post-processing codes for computing empirical Gramians or balanced POD modes may already exist, but may not be equipped to deal with complex-valued vectors. While we must always consider complex-valued vectors when computing the eigenvector matrix \( V_M \), this process will in general be handled by a different code than the one that computes Gramians or balanced POD modes. As such, a real factorization of \( V_M N V_M^* \) may be desirable.

We break \( V_M \) and \( N \) into their real and imaginary parts:

\[
V_M = V_M^R + iV_M^I, \\
N = N^R + iN^I.
\]

For a real-valued system, the product \( V_M N V_M^* \) must also be real-valued, so we can simply collect the real terms in computing

\[
= [V_M^R V_M^I] [N^R N^I] [V_M^R V_M^I]^T.
\]

(12)

(The imaginary terms can be shown to equal the zero matrix individually, if one considers the form of \( N \) itself, as well as the fact that the columns of \( V_M \) come in conjugate pairs for a real-valued system.)

We recall from (9) that

\[
N_{j,k} = -\frac{1}{\lambda_j + \bar{\lambda}_k}.
\]

Since (11) is a stable system, we let \( \lambda_j = -\alpha_j + i\beta_j \), with \( \alpha_j > 0 \). Then

\[
N_{j,k} = \frac{\alpha_j + \alpha_k}{(\alpha_j + \alpha_k)^2 + (\beta_j - \beta_k)^2} + \frac{\beta_j - \beta_k}{(\alpha_j + \alpha_k)^2 + (\beta_j - \beta_k)^2},
\]

giving us

\[
N_{j,k}^R = \frac{\alpha_j + \alpha_k}{(\alpha_j + \alpha_k)^2 + (\beta_j - \beta_k)^2} \\
N_{j,k}^I = \frac{\beta_j - \beta_k}{(\alpha_j + \alpha_k)^2 + (\beta_j - \beta_k)^2}.
\]

From this we see that \( N^R \) is symmetric and \( N^I \) is skew-symmetric, making

\[
Q = \begin{bmatrix}
N^R & N^I \\
-\bar{N}^I & \bar{N}^R
\end{bmatrix}
\]

a symmetric matrix. Then it has a unitary diagonalization \( Q = UQ\Lambda U^* \). Letting

\[
R = UQ\Lambda^{1/2},
\]

(12) can then be rewritten as

\[
V_M N V_M^* = \left( [V_M^R V_M^I] R \right) \left( [V_M^R V_M^I] R \right)^T,
\]

and the controllability Gramian (see (10)) as

\[
W_c = \left[ X_T [V_M^R V_M^I] R \right] \left[ X_T [V_M^R V_M^I] R \right]^T,
\]

(13)

a product of real matrices.
3.4. Application to balanced proper orthogonal decomposition

While (11) was derived for an impulse response of the direct system (1), the same method can be applied to an impulse response of the adjoint system (4). Applying the analytic tail method to both sets of impulse responses, we can factor the controllability and observability Gramians as

\[
W_c = [X_T \ V^c_M \Gamma^c] [X_T \ V^c_M \Gamma^c]^*, \\
W_o = [Y_T \ V^o_M \Gamma^o] [Y_T \ V^o_M \Gamma^o]^*.
\]

For balanced POD, we construct the Hankel matrix by multiplying factors of the controllability and observability matrices. With the analytic tail, this gives us

\[
H = [Y_T \ V^o_M \Gamma^o]^* [X_T \ V^c_M \Gamma^c].
\] (14)

From here, the rest of the balanced POD algorithm is the same.

4. Dynamic mode decomposition

4.1. Snapshot-based eigenvector estimation

The analytic tail method described in Section 3 requires a knowledge of certain eigenvalues and eigenvectors of \(A\). For a very large system, computing the eigenvectors of \(A\) directly may be numerically intractable. In some cases, for instance in fluid simulations, the exact form of \(A\) is not even known. Iterative methods such as the Arnoldi algorithm provide a means for estimating the eigenvalues and eigenvectors of large systems, taking a “black box” approach that does not require an explicit knowledge of \(A\). Instead, they simply require the ability to compute the evolution of an initial condition under the dynamics defined by \(A\). However, for our purposes Arnoldi-like methods are less than ideal. In addition to requiring additional simulations, which may be expensive, they typically estimate the eigenvectors of \(A\) whose corresponding eigenvalues lie on the periphery of the spectrum [10], whereas we are only interested in those that dominate the impulse response tail. For instance, the slowest eigenvalue of \(A\) may correspond to an eigenvector that is not excited by the impulse at all.

Instead, we turn to dynamic mode decomposition (DMD), a variant of the Arnoldi algorithm [11, 12]. The DMD algorithm is completely snapshot-based, and requires no direct knowledge of \(A\). All that it requires is a set of snapshots from a simulation of the dynamics defined by \(A\). The resulting DMD modes will capture only the behavior observed in this snapshot set. For the analytic tail method, we can run an impulse response simulation until a small number of eigenvectors begins to dominate the state, at which point we stop the simulation. (This cut-off can be detected, for example, by plotting the norm of the state and waiting until only a few frequencies are present in the signal.) DMD modes can then be computed from a small number of snapshots collected at the end of the impulse response. By using DMD, we eliminate the need for any additional simulations and guarantee that only those eigenvectors with a measurable presence in the tail are estimated.

The number of snapshots necessary for such a DMD computation depends on the number of eigenvectors that are active in the impulse response tail. At minimum, the rank of the snapshot set must be equal the number of eigenvectors to be estimated. Since this number is assumed to be small (a simple tail is assumed), the DMD computation is quite cheap. Furthermore, there is no benefit in using additional snapshots if they do not increase the rank of the snapshot set. In a simple impulse response tail, all snapshots will be linear combinations of the same few eigenvectors, so there is no need to extend the impulse response past the beginning of the tail.

It was shown in [12] that if one uses DMD to estimate eigenvalues \(\lambda_j\) and eigenvectors \(v_j\) from a set of snapshots \(\{k_j\}_{j=0}^N\), the modes can be scaled such that

\[
k_j = \sum_{k=1}^{N} \lambda_k^j v_k \quad j = 0, \ldots, N - 1.
\] (15)
Thus the norm of each mode gives some indication of its contribution to a given snapshot. For example, the first snapshot is simply equal to the sum of the DMD modes:

\[ k_0 = \sum_{k=1}^{N} v_k. \]  

(16)

As such, in addition to an estimate of the eigenvectors and eigenvalues that dominate the impulse response tail, DMD analysis also provides us with a way to quantify the relative importance of each eigenvector/eigenvalue pair, based on the norm \( \|v_k\| \). This can be used to determine how many eigenvectors are necessary to characterize an impulse response tail.

4.2. Limitations of the standard algorithm

The algorithms presented in [11, 12] are equivalent for infinite-precision arithmetic, but here we focus on the algorithm described in [11] due to its improved numerical stability. A typical implementation consists of the following steps:

1. Collect \( N + 1 \) snapshots \( k_0, \ldots, k_N \) from the simulation of a linear system.
2. Compute the singular value decomposition (SVD) \( K = U_K \Sigma_K W_K^* \) where \( K \) is the matrix of snapshots \( K = [k_0 \ldots k_{N-1}] \).
3. Form the matrix \( \tilde{A} = U_K^* K' W_K \Sigma_K^{-1} \), where \( K' = [k_1 \ldots k_N] \).
4. Solve the eigenvalue problem \( \tilde{A} \tilde{V} = \tilde{V} \Lambda \).
5. Compute the unscaled modes \( \hat{V} = U_K \tilde{V} \).
6. Solve \( \hat{V} d = k_0 \).
7. Compute the scaled DMD modes \( V = \hat{V} \cdot \text{diag}(d) \), which approximate the eigenvectors of the system whose dynamics yield \( k_0, \ldots, k_N \). (\( \text{diag}(d) \) is a diagonal matrix with \( d \) along its main diagonal.)

(The rescaling of the DMD modes is only necessary to ensure that (15) is satisfied, though the norms of the rescaled modes can be used as a measure of their importance [12].)

For very large systems, it may be inefficient or even impossible to implement this algorithm. If the snapshots are large and there are sufficiently many of them, it may be impossible to form the snapshot matrix \( K \) at all, due to a lack of memory. As such, we would like to formulate the DMD algorithm in a way that is memory efficient, allowing flexibility for cases when only a small number of snapshots can be stored in memory at any given time. Furthermore, we would like to generalize the formulation to allow for the use of any inner product on the space of snapshots. In performing the SVD of \( K \) directly, the matrices \( U_K \) and \( W_K \) will be orthogonal with respect to the standard \( L_2 \) inner product. To change this, we would have to pre- and post-process the snapshot matrix appropriately, for instance scaling and unscaling by grid weights (e.g., in a fluid flow simulation).

4.3. Memory-efficient algorithm

The most memory intensive steps in the standard DMD algorithm are those involving matrices whose columns are snapshots, or the size of snapshots: computing the SVD of \( K \), computing the product \( U_K^* K' \) (in the formation of \( \tilde{A} \)), and solving the least-squares problem \( k_0 = \tilde{V} d \). Here we present a formulation of the DMD algorithm that allows each of these operations to be done in a memory-efficient manner, requiring as few as two snapshots to be loaded in memory at any given time. This flexibility allows the algorithm to be applied to very large datasets.

The SVD of \( K \) can be computed efficiently by observing that the columns of the left singular matrix \( U_K \) are the same as the proper orthogonal decomposition (POD) modes of the dataset \( \{k_j\}_{j=0}^{N-1} \). As such, we can compute them in an efficient manner using the method of snapshots, as proposed by Sirovich [17].

We first form the correlation matrix \( K^*K \), noting that we use matrix multiplication notation on the space of snapshots as a shorthand for general inner products. That is, in general the elements of the matrix \( K^*K \) are given by

\[ (K^*K)_{j,k} = \langle v_k, v_j \rangle. \]
In the case of an $L_2$ inner product, this simplifies to $(\mathbf{K}^*\mathbf{K})_{j,k} = \mathbf{v}_j^*\mathbf{v}_k$, as usual. We observe that the elements of $\mathbf{K}^*\mathbf{K}$ can be computed one at a time if necessary, loading only two snapshots in memory at any given time. (Of course, when possible more snapshots should be loaded simultaneously for efficiency.) In addition, this is a symmetric matrix, so only the upper triangular portion needs to be computed. Next we compute the eigenvectors of the correlation matrix:

$$(\mathbf{K}^*\mathbf{K})\mathbf{W}_K = \mathbf{W}_K\hat{\Sigma}_K.$$  

Alternatively, for numerical considerations we can compute the SVD

$$\hat{\mathbf{K}}^*\mathbf{K} = \mathbf{W}_K\hat{\Sigma}_K\hat{\mathbf{W}}_K^*.$$  

The POD modes of the original snapshot set are then given by

$$\mathbf{U}_K = \mathbf{K}\mathbf{W}_K\hat{\Sigma}_K^{-1/2},$$  

where it can be shown that $\hat{\Sigma}_K = \Sigma_{K}^2$.

We can use (17) to compute the product $\mathbf{U}^*_K\mathbf{K}'$ efficiently. Suppose that $\mathbf{U}_K$ has $N_U$ columns. By definition, $\mathbf{K}'$ has $N$ columns. Then computing the product $\mathbf{U}^*_K\mathbf{K}'$ would require $N \times N_U$ inner products. (Again, we emphasize that because this matrix product involves columns whose size is equal to that of a snapshot, it is only a shorthand for general inner products.) But if we use (17) to expand this product, we see that

$$\mathbf{U}^*_K\mathbf{K}' = \hat{\Sigma}_K^{-1/2}\mathbf{W}_K^*\hat{\mathbf{K}}'\mathbf{K}'$$

$$= \hat{\Sigma}_K^{-1/2}\mathbf{W}_K^*[k_0 \ldots k_{N-1}]^* [k_1 \ldots k_N]$$

In this form, we see that we can reduce the computation of $\mathbf{U}^*_K\mathbf{K}'$ to a set of inner products of the snapshots $\{k_j\}_{j=1}^N$ with the snapshots $\{k_j\}_{j=0}^{N-1}$, modulo scaling by $\hat{\Sigma}_K^{-1/2}\mathbf{W}_K^*$. But except for those involving $k_N$, all of these inner products were already computed as elements of the matrix $\mathbf{K}^*\mathbf{K}$. Then in this step we only have to compute $N$ new inner products $\langle k_N, k_j \rangle$ for $j = 0, \ldots, N-1$:

$$\mathbf{U}^*_K\mathbf{K}' = \hat{\Sigma}_K^{-1/2}\mathbf{W}_K^*[(\mathbf{K}^*\mathbf{K})_{0,N-1},[1,N-1] [k_0 \ldots k_{N-1}]^* k_N].$$  

(18)

By noting this overlap with the correlation matrix, we have reduced the number of necessary inner products from $N \times N_U$ to $N$. Again, we can compute the additional inner products one at a time if necessary, requiring as few as two snapshots in memory at a given time.

Finally, we can solve $\mathbf{k}_0 = \hat{\mathbf{V}}\hat{\mathbf{d}}$ efficiently by using a pseudoinverse. The columns of $\hat{\mathbf{V}}$ are estimated eigenvectors of $\mathbf{A}$, and as such are the same dimension as a snapshot. Thus it may not be possible to hold $\hat{\mathbf{V}}$ in memory. Instead, we consider the solution

$$\hat{\mathbf{d}} = (\hat{\mathbf{V}}^*\hat{\mathbf{V}})^{-1}\hat{\mathbf{V}}^*\mathbf{k}_0.$$  

The elements of $\hat{\mathbf{V}}^*\mathbf{k}_0$ are just inner products of $\mathbf{k}_0$ with the unscaled DMD modes, which can be computed with as few as two snapshots in memory. To compute the matrix $\hat{\mathbf{V}}^*\hat{\mathbf{V}}$, we recall that $\hat{\mathbf{V}} = \mathbf{U}_K\hat{\mathbf{V}}$, where the columns of $\hat{\mathbf{V}}$ are the eigenvectors of $\hat{\mathbf{A}}$ (see Section 4.2). Then we can write

$$\hat{\mathbf{V}}^*\hat{\mathbf{V}} = \hat{\mathbf{V}}^*\mathbf{U}_K\mathbf{U}_K\hat{\mathbf{V}}$$

$$= \hat{\mathbf{V}}^*\hat{\mathbf{V}},$$

since $\mathbf{U}_K$ is unitary. The product of high-dimensional vectors $\hat{\mathbf{V}}^*\hat{\mathbf{V}}$ is thus reduced to an $N \times N$ matrix multiplication, where $N$, the number of snapshots, is much smaller than the dimension of the snapshots. As a result, we can write the solution to our least-squares problem as

$$\hat{\mathbf{d}} = (\hat{\mathbf{V}}^*\hat{\mathbf{V}})^{-1}\hat{\mathbf{V}}^*\mathbf{k}_0.$$  

(19)
where the only manipulations involving snapshot-sized vectors are the (at most) \( N_U \) inner products of \( k_0 \) with the columns of \( \mathbf{V} \).

The low-memory DMD algorithm is summarized below:

1. Collect \( N + 1 \) snapshots \( k_0, \ldots, k_N \) from the simulation of a linear system.
2. Stack the first \( N \) snapshots into a matrix \( \mathbf{K} = [k_0 \cdots k_{N-1}] \) and compute the correlation matrix \( \mathbf{K}^* \mathbf{K} \) (inner products).
3. Solve the eigenvalue problem \( (\mathbf{K}^* \mathbf{K}) W_k = W_k \Sigma_k \) or the SVD \( \mathbf{K}^* \mathbf{K} = W \hat{\Sigma} W^* \).
4. Compute the matrix \( \mathbf{U}_k = \mathbf{K} W_k \hat{\Sigma}_k^{-1/2} \) (linear combination).
5. Compute \( [k_0 \cdots k_{N-1}]^* k_N \) (inner products).
6. Stack the above matrix with elements of the correlation matrix to compute the product
\[
\mathbf{\hat{A}} = \hat{\Sigma}_k^{-1/2} W_k^* (K^* K)_{[0, N-1], [1, N-1]} [k_0 \cdots k_{N-1}]^* k_N W_k \Sigma_k^{-1/2}.
\]
7. Solve the eigenvalue problem \( \mathbf{\hat{A}} \hat{\mathbf{V}} = \Lambda \hat{\mathbf{V}} \).
8. Compute the unscaled modes \( \hat{\mathbf{V}} = \mathbf{U}_k \hat{\mathbf{V}} \) (linear combination).
9. Compute the elements of \( \hat{\mathbf{V}}^* k_0 \) (inner products).
10. Compute the vector \( d = (\hat{\mathbf{V}}^* \hat{\mathbf{V}})^{-1} \hat{\mathbf{V}}^* k_0 \).
11. The scaled DMD modes are given by \( \mathbf{V} = \hat{\mathbf{V}} \cdot \text{diag}(d) \) (linear combination).

All of the steps involving inner products can be done with as few as two snapshots in memory at a time. The steps involving linear combinations can be done with a single snapshot in memory at a given time. All other steps are matrix operations involving matrices whose dimension is small relative to the dimension of a snapshot. The key departures from the standard DMD algorithm are the use of the correlation matrix to compute \( \mathbf{U}_k \), the reuse of the correlation matrix to compute \( \hat{\mathbf{A}} \) efficiently, and the computation of \( d \) using a pseudoinverse.

5. Results and discussion

5.1. Computing the controllability Gramian

Here we present two examples that demonstrate the effectiveness of the analytic tail method in computing empirical Gramians. In each, we compute the impulse response of a real system \( \dot{x} = \mathbf{A} x + \mathbf{B} u \), collecting snapshots of the state \( x \) every \( \Delta t = 0.01 \). The empirical controllability Gramian is first computed using what we will refer to as the “standard” method. For varying \( T \), we stack snapshots spanning the interval \( t = [0, T] \) as columns of a matrix, using a uniform quadrature weight \( \sqrt{\Delta t} \):

\[
\mathbf{X}_T = [x(0) \ x(\Delta t) \ \ldots \ x(T)] \sqrt{\Delta t}.
\]

The empirical Gramian is then given by \( \mathbf{W}_c = \mathbf{X}_T^* \mathbf{X}_T \). For the analytic tail method, we use DMD to compute the slow eigenvalues and eigenvectors and form the matrices \( \mathbf{V} \) and \( \mathbf{\Gamma} \) as in (11). The modified snapshot matrix is then

\[
\mathbf{X} = [\mathbf{X}_T \ \mathbf{V} \mathbf{\Gamma}]
\]

and the controllability Gramian is given by \( \mathbf{W}_c = \mathbf{X}^* \mathbf{X} \). We compare each of these computations against the controllability Gramian as computed using Matlab. The error is measured using the Frobenius matrix norm:

\[
\| \Delta \mathbf{W}_c \|_2 = \left[ \sum_j \sum_k \left( \mathbf{W}_{c,j,k}^{(\text{matlab})} - \mathbf{W}_{c,j,k}^{(\text{empirical})} \right)^2 \right]^{1/2}.
\]
5.1.1. Non-normal 3 × 3 system

In our first example, we consider the system (1) with

\[
\begin{bmatrix}
-1 & 0 & 100 \\
0 & -2 & 100 \\
0 & 0 & -5
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
\]

Though this system is stable, it exhibits non-normal transient growth before undergoing exponential decay (Figure 1 left). This non-normality is caused by the fact that the slow eigenvector \([-0.6 -0.8 -0.02]^T\) is nearly parallel to the span of the other two eigenvectors, \([1 0 0]^T\) and \([0 1 0]^T\). Using the standard method, we must sample the impulse response to \(T = 6\) before \(W_c\) converges to its final value (Figure 1 right). In contrast, if we estimate the slow eigenvector and treat the tail analytically, we observe convergence in \(W_c\) by \(T = 4\). (For a given simulation length \(T\), the last four snapshots are used for the DMD computation.)

In addition to the 33% reduction in required storage space for snapshots, we see that the analytic tail method produces a more accurate controllability Gramian for all \(T\), up to the point where both methods have converged. That the same amount of error is eventually observed for both methods is expected and encouraging. Though the analytic tail method should increase accuracy by accounting for the truncated snapshots, for a stable impulse response the contribution of those snapshots will eventually be negligible. Thus for \(T\) large enough, the two methods should produce nearly identical results. The observed agreement suggests that the analytic tail method is indeed enhancing accuracy in the manner intended. The remaining error in computing \(W_c\) is due to the fact that we are using snapshots to compute the Gramian empirically, and can be reduced by sampling the impulse response faster and/or using higher-order quadrature weights.

5.1.2. Pseudorandom 100 × 100 system

For a larger example, we construct a pseudorandom, 100 × 100 matrix \(A\) using the following procedure:

1. Start with a matrix of zeroes. Place ten stable, slow oscillators of the form

\[
\begin{bmatrix}
-\alpha & \beta \\
-\beta & -\alpha
\end{bmatrix}
\]

along the diagonal of \(A\). The values \(\alpha\) and \(\beta\) are chosen randomly subject to the restrictions \(\alpha \in (0, 1]\) and \(\beta \in [0, 10]\).

2. Place up to 40 stable, fast oscillators (same form as above) along the diagonal of \(A\). The number of fast oscillators is chosen randomly, modulo the restrictions \(\alpha \in (1, 5]\) and \(\beta \in [0, 10]\).
3. Place ten slow, stable, real eigenvalues along the diagonal of $A$. These eigenvalues are of the form $\lambda = -\alpha$ with $\alpha \in (0, 1]$.

4. The rest of the entries on the diagonal are filled with fast, stable, real eigenvalues $\lambda = -\alpha$ with $\alpha \in (1, 5]$.

5. Fill in the upper triangular portion of $A$ with random values lying in the interval $[0, 0.25]$.

By constructing $A$ in this way, we are able to specify its eigenvalues, guaranteeing a stable system with oscillatory dynamics and multiple timescales of interest.

Here we consider a particular choice of $A$ with a much more complex impulse response than was seen in the $3 \times 3$ example discussed previously. After an initial period of non-normal transient growth, the system simultaneously decays and oscillates (Figure 2). The decay rate is fairly constant from $t = 20$ to $t = 100$, though there is growing evidence of multifrequency interaction, in the form of beating (amplitude modulation). Around $t = 100$, the decay rate begins to slow down and the presence of beating is clear. The beating behavior begins to fade as the decay rate slows down to its final value, and by $t = 200$, it appears that we have returned to a constant decay rate and oscillation at a single, fixed frequency.

Using DMD analysis, we can corroborate this behavior. We consider impulse responses ending at $T = 50$, $T = 150$ and $T = 250$. For each case, we use the last 20 snapshots of the simulation for DMD. At $T = 50$, the spectrum is dominated by a real eigenvector with a decay rate $\alpha = 0.046$ (Figure 3 left). The beating, oscillatory behavior is caused by the interaction of two pairs of complex conjugate eigenvectors, at $\alpha = 0.004$ and $\alpha = 0.016$. For $T = 150$, the spectrum is instead dominated by the complex conjugate pair at $\alpha = 0.004$, corresponding to the change in decay rate discussed previously. The eigenvector pair at $\alpha = 0.016$ still has a significant, though reduced, norm here, corresponding to the reduced evidence of beating. Once we reach $T = 250$, the slow eigenvector pair at $\alpha = 0.004$ completely dominates the DMD spectrum, corresponding to the constant exponential decay and single frequency oscillation observed at the end of the impulse response.

Motivated by this DMD analysis, we compute the empirical controllability Gramian using a five-dimensional and two-dimensional analytic tail. (The DMD spectra suggest that five and two eigenvectors should accurately describe the impulse tails at $T = 150$ and $T = 250$, respectively.) Indeed, we see that for the five-dimensional tail, the controllability Gramian converges by $T = 150$ (Figure 3 right). With a two-dimensional tail, the computation converges by $T = 250$. Surprisingly, a standard computation of $W_c$ does not converge until we pass $T = 700$. This is despite the fact that past $T = 250$ the impulse response is dominated by a single pair of eigenvectors. The surprisingly slow convergence of the standard method highlights the fact that for many systems, an impulse response must be sampled well into the tail before convergence is achieved, even if by that point the dynamics are very simple.
Figure 3: (Left) DMD spectra for the impulse response of a pseudorandom 100 \times 100 system. The norms of the estimated eigenvectors are plotted against the corresponding decay rates. For $T = 50$ the dominant decay rate corresponds to a real eigenvector at $\alpha = 0.046$ whereas at $T = 150$ it corresponds to a complex conjugate pair at $\alpha = 0.004$. The beating observed in the impulse response results from the interaction of the $\alpha = 0.004$ pair with another complex conjugate eigenvector pair at $\alpha = 0.016$. (Right) Error in computing the controllability Gramian empirically for a pseudorandom 100 \times 100 system. As predicted by DMD analysis, a five-dimensional analytic tail leads to convergence by $T = 150$, while a two-dimensional tail converges at $T = 250$. In contrast, without an analytic tail, the empirical controllability Gramian does not converge until $T > 700$. By applying the analytic tail method we achieve a savings of 65\% (2-D tail), or even 79\% (5-D tail), in storage space for snapshots.

5.2. Model reduction

The analytic tail method is especially useful for balanced POD, as there are tails associated with both the direct and adjoint impulse responses. We present two examples demonstrating the benefits of the method. First, we consider the complex Ginzburg-Landau (CGL) equation. A discretization of these dynamics yields a system that can be analyzed directly using numerical packages such as Matlab, allowing us to compare the models derived using balanced POD against those generated from exact balanced truncation. We then consider the two-dimensional flow past a cylinder. This is a much larger computation and clearly demonstrates the savings achieved with the analytic tail method, as well as its applicability for the types of large systems that are likely to be encountered in practice. Unfortunately, due to the size of the problem, exact balanced truncation cannot be performed, and as such, we use convergence tests to compare the results of balanced POD with and without analytic tails. This is in contrast to the CGL system, for which a direct comparison to balanced truncation is done.

5.2.1. Complex Ginzburg-Landau system

The linearized, complex Ginzburg-Landau (CGL) equation is given by

$$\dot{q} = -\nu \frac{\partial q}{\partial x} + \mu(x)q + \gamma \frac{\partial^2 q}{\partial x^2}. \tag{22}$$

The evolution of $q$ can be thought of as a model for the growth and decay of a velocity perturbation in a fluid flow. For a control-oriented review of the CGL equation, see [18]. To put (22) in the state-space form \([11]\), we discretize as described in \([14]\). We choose a state dimension $n = 100$, which is large enough to accurately represent \([22]\) but small enough to perform exact balanced truncation (using Matlab), which we use as a reference for our empirical methods. We choose a subcritical value $\mu_0 = 0.38$, place a single actuator at $x = -1$, and place a single sensor at $x = 1$. All other parameters are set to the default values used in \([13]\).

The direct impulse response initially decays before undergoing non-normal transient growth, with $\|x\|_2$ reaching a peak around $t = 11$ (Figure 4 left). Past this point, there is exponential decay at a constant rate. Looking at the real part of the state, we can see that the state also oscillates, with a single, fixed frequency.
As such, we use a single, complex eigenvector to describe the direct impulse response tail. We assume that the same can be done for the adjoint impulse response.

To form reduced-order models of (22), we collect snapshots of the direct and adjoint impulse responses every $\Delta t = 0.01$. We vary the truncation point $T$ from 20 to 100, at each point using the last 20 snapshots for a DMD computation of the slow eigenvector and eigenvalue, direct and adjoint respectively. The snapshots are scaled with fourth-order quadrature weights [20] so that the quadrature sum (3) more accurately approximates the integral expression (2). Balanced POD modes are then computed, both with and without an analytic tail. We project the dynamics (22) onto these modes to get reduced-order models.

Figure 4 (right) shows the error in computing 10-state reduced-order models of the linearized CGL equation, as a function of $T$. In addition to comparing the models to each other, we also compare the transfer function errors to the analytic bounds given by (5) and (6). We recall that because balanced POD is only an approximation of balanced truncation, the theoretical upper bound (6) may not be satisfied. This is indeed the case for the standard balanced POD models, up to $T = 85$. In contrast, the analytic tail models meet this criterion as early as $T = 25$. To check that this is not a peculiarity for a model order $r = 10$, we fix $T$ and plot the transfer function error as a function of the model order $r$. Figure 5 shows that with $T = 25$, the analytic tail models meet the theoretical upper bound for all model orders, while the standard balanced POD models fail to meet the theoretical upper bound for any. However, as expected, with increasing $T$ the performance of the standard balanced POD models begins to approach that of the analytic tail models. For $T = 100$, the errors are nearly indistinguishable (not pictured, for clarity) and closely approximate the error for exact balanced truncation (within 8% for the cases shown).

### 5.2.2. Two-dimensional cylinder flow

To investigate the flow past a two-dimensional cylinder, we use the fast immersed boundary method developed by Colonius and Taira [21]. In this formulation, the forces on the surface of a body are modeled as a set of delta-functions, yielding the governing equations

\[
\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla) \vec{u} = -\nabla p + \frac{1}{\text{Re}} \nabla^2 \vec{u} + \int f(\vec{x}) \delta(\vec{x} - \vec{x}) d\vec{x}
\]

\[
\nabla \cdot \vec{u} = 0.
\]

(We use arrows to denote the vectors in these equations, to avoid confusion with the variables defined previously in the discussion of linear systems.) The magnitude of the delta forces at a given point is chosen
to enforce the no-slip condition.

The fast immersed boundary method uses nested domains, each with increasing mesh resolution. For the finest domain we consider \((x, y) \in [-15, 15] \times [-5, 5]\), with a cylinder centered at \((0, 0)\). The large upstream region is necessary for the adjoint simulations, for which the flow moves in the reverse direction. With three nested grids, the full computational domain spans a region \((x, y) \in [-60, 60] \times [-20, 20]\). (See Figure 6 for an illustration of the computational domain.) Convergence tests show that this domain is sufficiently large, avoiding blockage effects and fully capturing the features of the wake. In terms of grid cells, each of the nested domains has dimension 1500 \times 500, corresponding to \(dx = dy = 0.02\). Only the data from the innermost domain, with the finest resolution, are used for the balanced POD analysis. The outer domains are used only to ensure an accurate simulation.

We consider the flow past a cylinder of diameter 1 at a Reynolds number of 100, for which the flow is globally unstable. This instability leads to an oscillatory wake, where vortices alternately shed from the upper and lower shear layers, yielding the familiar Kármán vortex street. As the vortices shed, they generate unsteady forces on the cylinder, which can be undesirable. To eliminate these oscillations, we can design and implement feedback controllers based on reduced-order models. Here we will investigate the benefits of using the analytic tail method in constructing such models using balanced POD.

The balanced POD computation requires simulations of the direct and adjoint dynamics, which are based on the linearization of Equation (23). (For details on this linearization, see [3].) To simulate these dynamics, we must first identify the unstable equilibrium. We do so using selective frequency damping [22], yielding the steady solution shown in Figure 7. Furthermore, because balanced POD can be applied only to stable systems, we must also decouple the stable and unstable dynamics. For the cylinder flow at Re = 100, the direct and adjoint systems each have a single pair of unstable global modes, which we compute using a standard Arnoldi iteration [16]. These global modes are shown in Figure 8 and are used to project the linearized and adjoint dynamics onto their stable subspaces, respectively. It is these restricted, stable dynamics on which we perform balanced POD.

To control the cylinder wake, we actuate the flow using a disk of vertical force located downstream of the cylinder. The forcing covers a spatial region equal in size to the body, and is placed two diameters downstream (Figure 9). This choice of actuation is based on the work of Noack et al. [23], which can be used as a benchmark for cylinder control applications. Though it does not model a physical actuator, the volume forcing is a convenient choice for this example as it is easy to implement and has an obvious effect on the wake. The output signals used for feedback control are collected using point sensors placed at \(x = 2, 3,\) and 6. Each sensor measures the vertical component of the velocity alone.

The direct impulse response for this system, restricted to the stable subspace, is qualitatively similar to
Figure 6: Domain used for simulating flow past a two-dimensional cylinder. Each of the nested domains contains $1500 \times 500$ grid points, giving the finest grid a grid spacing $dx = dy = 0.02$. The large upstream region is necessary for adjoint simulations, which flow from right to left.

Figure 7: Unstable equilibrium for flow past a cylinder at $Re = 100$. The flow field is depicted using contours of vorticity overlaid with velocity vectors.

that of the CGL system. Figure 10 (left) shows that there is an initial period of non-normal growth during which the norm of the state grows by over four orders of magnitude. This is followed by a relatively slow decay. After 1200 convective times (60,000 time steps) the state is still over six times as energetic as the initial condition. During the initial period of decay ($t \in [100, 300]$), there are slow oscillations in the kinetic energy. While the slow oscillations die out, there are also fast oscillations that are present through the end of the impulse response (see the enlarged inset in Figure 11). By $t = 500$, this fast frequency is the only oscillatory behavior that can be observed. All other oscillatory behavior has died away. This, in addition to the fact that the decay rate is perfectly logarithmic, suggests that the remainder of the impulse response can be modeled using an analytic tail.

To test this hypothesis, we compute a series of reduced order models using balanced POD, with and without an analytic tail. For such a high-dimensional system, we cannot compute an exact balanced truncation, so we instead check for convergence. As more snapshots are used in the standard balanced POD computations, more and more of the long-time behavior is captured, and the models should converge. If the analytic tail method is correctly capturing the long-time behavior, then the models computed using an analytic tail will converge to the same answer, but using fewer snapshots.

We run our direct and adjoint impulse response simulations to $t = 1200$, collecting a snapshot every 50 timesteps (once every convective time unit). (Collecting snapshots at this rate resolves the fastest frequency observed in the impulse responses.) All of these snapshots are used to compute a balanced POD model of order 16, using Riemann sum approximations for all integrals. We take this model as the best approximation of exact balanced truncation. Figure 11 (left) shows that the output predicted by this model does in fact
match the output from the full simulation, validating this approximation. (A close inspection reveals small discrepancies between simulation and model outputs. However, these are expected and result from the fact that in this multi-domain scheme, the Laplacian operator is not self-adjoint to numerical precision, as described in [3].)

We then compute 16-state models with direct and adjoint impulse responses truncated at various $T < 1200$. Each of these models is compared to the $T = 1200$ model to check for convergence. The results of this analysis are shown in Figure 11 (right). As before, for each choice of $T$, using the analytic tail method improves the accuracy of the model. Furthermore, we see that the models computed with an analytic tail converge must faster than those computed without. With an analytic tail of dimension two, snapshots only need to be collected up to $T = 400$. The computation of the eigenvectors dominating the tail is fairly cheap, requiring a DMD computation using only the last seven available snapshots. Further savings could potentially be achieved by considering more vectors in the tail, at little additional cost.

Table 1 gives a quantitative summary of the savings achieved by implementing the analytic tail method. The computation time is dominated by the impulse responses (one direct, three adjoint), which are each done in serial. Using analytic tails, we get a linear speedup in the simulation time (67% savings), which for this particular computation corresponds to a savings of nearly 300 CPU hours. In computing the Hankel matrix, we achieve a savings of nearly 85%, or about 70 CPU hours. While the absolute savings in this

Figure 8: Unstable global modes for the two-dimensional cylinder flow at $Re = 100$. Flow fields are depicted using contours of vorticity overlaid with velocity vectors. (a) Direct system, real part; (b) adjoint system, real part; (c) direct system, imaginary part; (d) adjoint system, imaginary part.

Figure 9: Schematic of input and output for two-dimensional cylinder flow. Actuation is implemented as a disk of vertical force two cylinder diameters downstream of the body (blue). Point sensors measuring the vertical velocity are placed at $x = 2$, 3, and 6 (red, ×).
step is smaller, it scales roughly quadratically. This is critical, as constructing the Hankel matrix can easily dominate the computation time. For instance, using a parallel solver could decrease the simulation time, while large datasets and/or larger snapshot ensembles would increase the cost of assembling the Hankel matrix. Computing the SVD of the Hankel matrix (97% savings) scales cubically, but the SVD time is such a small part of the total cost that this savings is insignificant. Finally, we also achieve a linear speedup (66%, 33 CPU hours) in constructing the balanced POD modes, which is a linear operation. In total, by implementing the analytic tail method, we save nearly 400 CPU hours without any sacrifice in model accuracy.

6. Conclusions

We have presented a method for analytically treating the tail of an impulse response, improving accuracy and efficiency when computing empirical Gramians or using balanced proper orthogonal decomposition (balanced POD) to compute reduced-order models. When the long-term behavior of an impulse response is governed by a small number of eigenvectors, we can account for the effect of these eigenvectors on the empirical Gramian analytically. In doing so, we no longer need to sample the impulse response past the beginning of the tail. This lowers the storage requirement for snapshots and speeds up ensuing computations. These effects are especially useful for balanced POD, as benefits are gained in treating both the direct and adjoint impulse responses this way. We estimate the eigenvectors that dominate the tail using dynamic mode decomposition (DMD). By using this snapshot-based method, we minimize the additional cost in applying the analytic tail method, requiring no additional simulations. In particular, we develop a low-memory implementation of DMD that is appropriate for large datasets.
These methods were applied to a number of examples, demonstrating their effectiveness. For two linear systems, the analytic tail method was used to aid in computing the controllability Gramian empirically. It was also used to more efficiently compute reduced-order models of the linearized, complex Ginzburg-Landau equation and the linearized flow past a two-dimensional cylinder at a Reynolds number of 100. In all cases, the use of an analytic tail produced highly accurate results with significantly fewer snapshots than would be required otherwise. The controllability Gramians and balanced POD-based models, respectively, converged to the values that were obtained when the impulse responses were sampled far into their tails. These examples verify that the analytic tail method correctly accounts for the long-term behavior of the impulse response tails with little additional cost.

We note that though these last two examples focused on the accuracy of the resulting low-order models, the main benefit of the analytic tail method is in computing the balanced POD modes themselves. If the goal is to compute an accurate input/output model, the eigensystem realization algorithm (ERA) provides an alternative to balanced BPOD. ERA models are equivalent to balanced POD modes [2], and because ERA makes use of input/output data rather than snapshots, it is a much faster method. However, using ERA one cannot compute the balanced modes of the system. (At best, ERA can be used to compute the direct modes, if snapshots of the impulse response are available, but not the adjoint modes.) In some applications, a knowledge of the modal structures is desirable, as it may lend insight into the underlying flow physics that an input/output model alone could not. For these purposes, balanced POD is an appropriate method.

7. Acknowledgments

This work was supported by the AFOSR grant FA9550-09-1-0257 and the National Science Foundation Graduate Research Fellowship Program (NSF GRFP). The authors acknowledge productive conversations with Mark Luchtenburg regarding the low-memory DMD algorithm.

8. References


