Nonlinear space-time model reduction in the frequency domain

Peter Frame[*1] **and Aaron Towne**[1]

[1]Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA

Abstract. We propose a space-time reduced-order model (ROM) for nonlinear dynamical systems, building upon previous work on linear systems. Whereas most ROMs are space-only in that they reduce only the spatial dimension of the state, the proposed method leverages an efficient encoding of the entire trajectory of the state on the time interval $[0, T]$, enabling significant additional reduction. Trajectories are encoded using SPOD modes, a spatial basis at each temporal frequency tailored to the structures that appear at that frequency. These modes have a number of properties that make them an ideal choice for space-time model reduction, including separability and near-optimality for long trajectories. We derive a system of algebraic equations involving the SPOD coefficients, forcing, and initial condition by projecting an implicit solution of the governing equations onto the set of SPOD modes in a space-time inner product. We therefore refer to the method as spectral solution operator projection (SSOP). The online phase of SSOP comprises solving this system for the SPOD coefficients, given the initial condition and forcing. We find that SSOP gives two orders of magnitude lower error than POD-Galerkin projection at the same number of modes and CPU time across a suite of tests, including ones that use out-of-sample forcings and affine parameter variation. In fact, the method is substantially more accurate even than the projection of the solution onto the POD modes, which is a lower bound for the error of any method based on a linear space-only encoding of the state.

Key words: space-time model reduction, spectral POD

1 Introduction

The great majority of model reduction methods use the following two-step approach: *(i)* find an efficient encoding of the state of the dynamical system, replacing the high-dimensional state space with a low-dimensional one, then *(ii)* derive a dynamical system for the encoding. There are many methods for both of these steps. Proper orthogonal decomposition (POD) modes [22, 38] are perhaps the most common choice for *(i)*, but other choices include balanced truncation modes [26, 44, 34] and nonlinear encodings like those based on autoencoders [10]. These methods all make

[*]Email address for correspondence: pframe@umich.edu

use of correlations between the points that make up the state. Common choices for *(ii)* include methods like Galerkin [2, 27] and Petrov-Galerkin [5] projections, which derive the reduced-order model (ROM) equations from the full-order model (FOM) equations, and methods like operator inference [32, 19], which obtain equations for the evolution of the encoding directly from data. The former methods are referred to as intrusive, and the latter as non-intrusive. We refer to methods that use this two-step paradigm as space-only model reduction methods — only the spatial dimension of the system is reduced.

Far less common are space-time model reduction methods [45, 10, 9, 18, 31, 16], which leverage the spatiotemporal correlations present in problems of interest, rather than solely the spatial ones. Step *(i)* is now to find an efficient encoding for the entire trajectory of the state over some time interval of interest. With the entire evolution of the state represented in the encoding, the degrees of freedom that represent the solution no longer evolve in time. Step *(ii)* becomes solving for these static degrees of freedom either by solving a system of simultaneous equations or an optimization. The potential advantage of the space-time paradigm over the space-only one is that solutions to problems of interest exhibit spatiotemporal coherent structures, which offer an extremely efficient means of describing the dynamics. In a space-only ROM, one is nonetheless solving for an encoding of the trajectory – the space-only encoding of the state at all time steps may be viewed as such – but this trajectory encoding is far from optimal. Indeed, the space-only POD coefficients, for example, are highly correlated from one time step to the next, so they provide a redundant (i.e., suboptimal) encoding of the entire trajectory. Thus, with the same total degrees of freedom, the space-time framework offers the potential for substantial additional accuracy by leveraging spatiotemporal correlations. The key questions are the following. (1) Can one solve for the space-time encoding coefficients accurately? POD-Galerkin projection (POD-G), e.g., does not exactly recover the projection of the solution onto the POD modes due to the influence of unclosed terms. To what extent does this persist for a space-time approach? (2) Can the system of equations in the space-time ROM be solved in similar time to the time integration required in the space-only ROM? In other words, does the space-time approach take the same (or less) CPU time as the space-only approach for the same number of degrees of freedom? If the answer to both of these is 'yes', then the space-time ROM will be much more accurate than a space-only ROM at the same CPU time.

The spatiotemporal basis vectors used in many space-time model reduction methods are separable, i.e., they are formed by taking the Kronecker product of a spatial vector and a temporal one, and the spatial basis vectors used are often the standard POD modes [10, 9, 18, 16]. In most cases, the time dependence for a given POD mode is tailored to that mode, which is accomplished by taking the SVD of a data matrix containing as its columns different temporal evolutions of the associated POD coefficient. In other cases [10], the same temporal basis is used for each POD mode. In either case, the temporal basis is obtained empirically. Refs. [10, 9, 18] minimize a space-time residual over the basis coefficients, and Ref. [16] trains a machine learning model that outputs the basis coefficients.

Our approach is to use spectral proper orthogonal decomposition (SPOD) modes as the space-time basis. At each temporal frequency, there is a spatial basis of modes — the SPOD modes — that are optimal for describing the structures that appear at that temporal frequency. The modes at frequency ω_k are associated, by definition, with the time dependence $e^{i\omega_k t}$. Therefore, the SPOD modes can be used as a (separable) space-time basis by adding these modes with their associated time dependence together with constant coefficients. A number of properties of the SPOD basis make them attractive for model reduction. First, for statistically stationary systems, SPOD modes are nearly the optimal linear space-time encoding for long time intervals. More precisely, for statistically stationary systems, SPOD modes approach space-time POD modes, the optimal linear encoding of trajectories, as the time interval of the encoding goes to infinity [22, 40]. Second, various properties of the analytic time dependence of the modes are useful; one of particular importance is that the Fourier time dependence leads to sparser coupling via the nonlinearity than one would achieve with a general space-time basis. Finally, the SPOD basis is separable, i.e., each mode is a function of space multiplied by a function of time. This means that relative to other space-time bases such as space-time POD, the SPOD modes are easier to converge from data, consume less memory, and inner products are far cheaper to evaluate.

In our previous work [12], we proposed *Spectral Solution Operator Projection* (SSOP), a space-time ROM that returns SPOD coefficients that represent the solution to a linear time-invariant system on a predefined time interval, given the initial condition and forcing. We showed that this method achieved orders-of-magnitude lower error than POD-G, balanced truncation [26], and even the projection of the FOM solution onto POD modes. Such accuracy was achieved at comparable (and usually slightly lower) CPU time than these benchmarks. SSOP is so named because it uses the fundamental solution to LTI systems, which can be thought of as a solution operator acting on the initial condition and forcing, and applies a space-time projection onto spectral POD modes to this solution operator. This results in an equation, at every frequency, giving the SPOD coefficients at that frequency in terms of the forcing and initial condition.

The goal of this paper is to generalize SSOP to nonlinear systems. Three new aspects must be addressed. First, the operator must be adjusted to account for the nonlinearity. To do this, we treat the nonlinear term as an additional forcing on the system and use the fundamental solution to the linear system with this additional forcing. This is consistent with an increasingly popular perspective in the fluid dynamics community on the role of nonlinearity [25, 24]. Crucially, this additional forcing depends on the solution itself, and this results in a system of nonlinear equations that must be solved for the SPOD coefficients. Second, the influence of nonlinearity at each frequency must be approximated as a function of the retained SPOD coefficients. We use two approaches, depending on the nature of the nonlinearity. The first is a DEIM-based hyper-reduction approximation of the nonlinearity, and the second, which is only applicable for quadratic nonlinearities, uses a subset of the triadic interactions between the SPOD modes. Third, the system of nonlinear equations that results from the addition of nonlinearity must be efficiently solved. We use a simple fixed-point iteration and find that for most cases, this converges in ~ 10 iterations. For cases where this does not converge, we use a pseudo-time-stepping method, which is somewhat slower but more stable.

We test the method on two Ginzburg-Landau systems, one with the standard cubic nonlinearity and the other with a Burgers'-type (quadratic) nonlinearity used to test the triadic-interaction handling of the nonlinearity. We demonstrate good results for both: at similar cost to POD-G, SSOP results in orders-of-magnitude lower error than POD-G and even the lower bound for space-only methods given by the projection of the solution onto the POD modes. We test the method both on statistically stationary forcings, where the SPOD modes provide the asymptotically optimal encoding, and on non-stochastic forcings, observing similar results for both. We also test the ability of the method to handle affine parameter dependence — we use SPOD modes from one Ginzburg-Landau system to build the ROM, then test it on a parameter sweep of Ginzburg-Landau systems.

In addition to our previous work [12], several methods have been proposed that use SPOD modes for model reduction. Refs. [21, 39] developed Galerkin and Petrov-Galekin SPOD-based ROMs, respectively, for linear systems. These models are only capable of handling systems in which the solution is known to be periodic. Recently, Ref. [20] proposed a nonlinear SPOD-based model, somewhat similar to the Galerkin models in Refs. [21, 39], but which was generalized to handle nonlinearities. Ref. [20] then applied this model to find periodic solutions in a canonical incompressible viscous flow. This latter model can be seen as a spatially reduced version of the harmonic balance method [15], a method which derives a relation between the various frequencies of a temporally periodic solution to the Navier-Stokes equations. The harmonic balance method has been applied in turbomachinery problems [15, 14], where the solution is assumed to be periodic with the fundamental frequency given by the blade-passing frequency, and has been used to study boundary layer transition [33]. In both, just a few harmonics are retained. All the methods above require a periodic solution, something that might be natural to expect of methods based on the Fourier transform. However, SPOD modes are effective at representing both periodic and aperiodic trajectories, and since the latter is far more common, there is a greater need for reduced-order models for that case. The proposed method, and its antecedent [12], can handle both periodic and aperiodic problems, owing to the solution operator projection approach.

The remainder of this paper is organized as follows. In Section 2, we describe trajectory representation, SPOD modes, and linear SSOP. We derive the general nonlinear SSOP in Section 3. We report the results of the application of the method to the Ginzburg-Landau systems in Section 4, and finally conclude the paper in Section 5.

2 Spectral POD and linear spectral solution operator projection

In this section, we discuss the preliminaries that lay the foundation for the proposed method. First, we discuss the task of trajectory encoding. Then, we review the relevant properties of spectral POD [40], the most important of which is the way SPOD modes can be used to represent a trajectory and the efficiency of this representation [13]. Finally, we review linear SSOP [12], which is the antecedent to this work.

2.1 Trajectory representation

A common task in model reduction is formulating an accurate scheme for encoding the state of a dynamical system $\boldsymbol{q} \in \mathbb{C}^{N_x}$. This entails specifying an encoder $\boldsymbol{e}^x : \mathbb{C}^{N_x} \to \mathbb{C}^r$ and a decoder $\boldsymbol{d}^x : \mathbb{C}^r \to \mathbb{C}^{N_x}$, where, ideally, $r \ll N_x$ and $\boldsymbol{q} \approx \boldsymbol{d}^x(\boldsymbol{e}^x(\boldsymbol{q}))$ for some appropriate ensemble of states $\boldsymbol{q}$. The superscript indicates that the spatial dimension is reduced. It is well known that, in a certain precise sense, the first r proper orthogonal decomposition modes can be used to formulate the optimal linear encoding scheme. Specifically, the encoder $\boldsymbol{e}^x(\boldsymbol{q}) = \boldsymbol{\Phi}^*\mathbf{W}\boldsymbol{q}$ and decoder $\boldsymbol{d}^x(\boldsymbol{a}) = \boldsymbol{\Phi}\boldsymbol{a}$ minimize $\mathbb{E}[\|\boldsymbol{q} - \boldsymbol{d}^x(\boldsymbol{e}^x(\boldsymbol{q}))\|_x^2]$ over all linear encoder/decoder pairs [28], where $\boldsymbol{\Phi} \in \mathbb{C}^{N_x \times r}$ is the matrix of the first r POD modes, and $\mathbf{W}$ is a (positive definite) weight matrix. The norm is induced by the inner product $\|\boldsymbol{q}\|_x = \sqrt{\langle \boldsymbol{q}, \boldsymbol{q}\rangle_x}$, the inner product is defined using the weight matrix as $\langle \boldsymbol{q}_1, \boldsymbol{q}_2\rangle_x = \boldsymbol{q}_2^*\mathbf{W}\boldsymbol{q}_1$, and $\mathbb{E}[\cdot]$ is the expectation operator over the ensemble that defines the POD modes. The 'x' subscript is used to denote a space-only norm or inner product, as opposed to a space-time one.

A related, though less familiar, task is formulating an encoding/decoding scheme for trajectories of the state of the dynamical system on some time interval $[0, T]$. We take the trajectory on the interval to be a vector of its values at times $0, \Delta t, \ldots, (N_\omega - 1)\Delta t$, where $N_\omega \Delta t = T$. That is, the trajectory is a vector $\boldsymbol{q}_{\mathcal{J}} = [\boldsymbol{q}_0^T, \boldsymbol{q}_1^T, \ldots, \boldsymbol{q}_{N_\omega-1}^T]^T$ where $\boldsymbol{q}_j$ is the state at time $t_j = j\Delta T$. Throughout the paper, a subscript $\mathcal{J}$ is used to indicate the set or vector that is formed by taking the subscript to be each element of the vector $\mathcal{J} = [0, 1, \ldots, N_\omega - 1]^T$. For example $t_{\mathcal{J}}$ denotes the vector $[0, \Delta t, \ldots, (N_\omega - 1)\Delta t]^T$. We denote the number of time steps N_ω for reasons related to the discrete Fourier transform, which will become clear later.

An encoding scheme for trajectories comprises an encoder $\boldsymbol{e}^{x,t} : \mathbb{C}^{N_x N_\omega} \to \mathbb{C}^{r N_\omega}$ and a decoder $\boldsymbol{d}_{\mathcal{J}}^{x,t} : \mathbb{C}^{r N_\omega} \to \mathbb{C}^{N_x N_\omega}$. The encoding scheme is effective if $\boldsymbol{q}_{\mathcal{J}} \approx \boldsymbol{d}_{\mathcal{J}}^{x,t}(\boldsymbol{e}^{x,t}(\boldsymbol{q}_{\mathcal{J}}))$ for trajectories of the system. More precisely, the expected error averaged over the trajectory, $\mathbb{E}[\|\boldsymbol{q}_{\mathcal{J}} - \boldsymbol{d}_{\mathcal{J}}^{x,t}(\boldsymbol{e}^{x,t}(\boldsymbol{q}_{\mathcal{J}}))\|_{x,t}^2]$, should be small. The space-time norm is induced in the usual way from a space-time inner product $\|\boldsymbol{q}_{\mathcal{J}}\|_{x,t} = \sqrt{\langle \boldsymbol{q}_{\mathcal{J}}, \boldsymbol{q}_{\mathcal{J}} \rangle_{x,t}}$, and this inner product is given by

$$\langle \boldsymbol{q}_{\mathcal{J}}^1, \boldsymbol{q}_{\mathcal{J}}^2 \rangle_{x,t} = \sum_{j=0}^{N_\omega - 1} \langle \boldsymbol{q}_j^1, \boldsymbol{q}_j^2 \rangle_x. \tag{2.1}$$

The analog of POD for the case of trajectories is space-time POD [22, 13] — it is the optimal linear encoding / decoding scheme for trajectories in the sense that it minimizes $\mathbb{E}[\|\boldsymbol{q}_{\mathcal{J}} - \boldsymbol{d}_{\mathcal{J}}^{x,t}(\boldsymbol{e}^{x,t}(\boldsymbol{q}_{\mathcal{J}}))\|_{x,t}^2]$. As discussed in the following subsection, SPOD modes can be used to represent trajectories, and a primary motivation for this paper is that the SPOD representation approaches the accuracy of the space-time POD representation for statistically stationary systems as $T \to \infty$. Another important point of comparison is the POD representation of trajectories, wherein each time step of the trajectory is encoded using POD modes, i.e., where $\boldsymbol{d}_j^{x,t}(\boldsymbol{e}^{x,t}(\boldsymbol{q}_{\mathcal{J}})) = \boldsymbol{\Phi}\boldsymbol{\Phi}^*\mathbf{W}\boldsymbol{q}_j$.

2.2 Spectral proper orthogonal decomposition

Spectral POD can be viewed as a ‘POD of the frequency domain’ for statistically stationary flows: for every temporal frequency, there is a basis of SPOD modes that optimally represent the spatial structures exhibited at that frequency. To make this precise, we define the trajectory at frequency ω_k using the discrete Fourier transform (DFT) as

$$\hat{\boldsymbol{q}}_k = \mathrm{DFT}_k\left[\boldsymbol{q}_{\mathcal{J}}\right] = \sum_{j=0}^{N_\omega - 1} \boldsymbol{q}_j e^{-i\omega_k t_j}, \tag{2.2}$$

where $\omega_k = 2\pi(k - N_\omega \Theta(k - N_\omega/2))/T$, where Θ is 0 if its argument is negative, and is 1 otherwise. If the DFT is applied to a new trajectory of the same system (or a different interval of the same long trajectory), the Fourier components will likely be different — if all $\hat{\boldsymbol{q}}_k$ were the same, the trajectories would be identical. The SPOD modes optimally capture this variability of $\hat{\boldsymbol{q}}_k$ just as the POD modes capture the variability of $\boldsymbol{q}$. Specifically, the SPOD modes at frequency ω_k can be defined to maximize the statistical energy captured at frequency ω_k, which is defined as

$$\lambda_k(\boldsymbol{\psi}) = \frac{\mathbb{E}\left[|\langle \hat{\boldsymbol{q}}_k, \boldsymbol{\psi} \rangle_x|^2\right]}{\|\boldsymbol{\psi}\|_x^2}. \tag{2.3}$$

The first SPOD mode at frequency ω_k is defined to maximize (2.3), and the latter modes are defined to maximize the same function subject to orthogonality with respect to previous modes,

$$\begin{aligned} \boldsymbol{\psi}_{km} &= \arg\max \lambda_k(\boldsymbol{\psi}) \\ \text{subject to} \quad \langle \boldsymbol{\psi}_{km}, \boldsymbol{\psi}_{kn} \rangle_x &= \delta_{mn} \quad \text{for} \quad 1 \leq m \leq n, \end{aligned} \tag{2.4}$$

where $\boldsymbol{\psi}_{km}$ denotes the m-th mode at the k-th frequency. We denote the energies of the modes as $\lambda_{km} = \lambda_k(\boldsymbol{\psi}_{km})$. Note the similarity to (space-only) proper orthogonal decomposition: the SPOD modes at ω_k are to $\hat{\boldsymbol{q}}_k$ as the POD modes are to $\boldsymbol{q}$.

With the modes defined, $\hat{\boldsymbol{q}}_k$ may be represented approximately by a linear combination of the first r_k modes,

$$\hat{\boldsymbol{q}}_k \approx \boldsymbol{\Psi}_k \boldsymbol{a}_k. \tag{2.5}$$

Here, $\boldsymbol{\Psi}_k = [\boldsymbol{\psi}_{k1}, \boldsymbol{\psi}_{k2}, \ldots, \boldsymbol{\psi}_{kr_k}] \in \mathbb{C}^{N_x \times r_k}$ is the truncated matrix of modes at frequency ω_k and $\boldsymbol{a}_k \in \mathbb{C}^{r_k}$ is the vector of expansion coefficients. It will become clear below why it is advantageous to retain different numbers of modes at different frequencies. The error in the representation can be expressed in terms of the truncated SPOD energies as

$$\mathbb{E}[\|\hat{\boldsymbol{q}}_k - \boldsymbol{\Psi}_k \boldsymbol{a}_k\|_x^2] = \sum_{m=r_k+1}^{N_x} \lambda_{km}, \tag{2.6}$$

and the first r_k SPOD modes minimize this quantity over all rank-r_k bases. If $\hat{\boldsymbol{q}}_k$ is known, the SPOD coefficients may be found using

$$\boldsymbol{a}_k = \boldsymbol{\Psi}_k^* \mathbf{W} \hat{\boldsymbol{q}}_k. \tag{2.7}$$

This may be written in terms of a space-time encoder $\boldsymbol{a}_k = \boldsymbol{e}_k^{x,t}(\boldsymbol{q}_{\mathcal{J}})$, where the k-th frequency of the encoder is given by

$$\boldsymbol{e}_k^{x,t}(\boldsymbol{q}_{\mathcal{J}}) = \boldsymbol{\Psi}_k^* \mathbf{W} \ \mathrm{DFT}_k[\boldsymbol{q}_{\mathcal{J}}]. \tag{2.8}$$

To obtain the SPOD modes at frequency ω_k from data, a set of realizations of $\hat{\boldsymbol{q}}_k$ is needed. These realizations could come from separate runs of the system, or, more commonly, from possibly overlapping blocks of a single long run [43]. Grouping N_d of these realizations into a data matrix $\hat{\mathbf{Q}}_k = [\hat{\boldsymbol{q}}_k^1, \hat{\boldsymbol{q}}_k^2, \ldots, \hat{\boldsymbol{q}}_k^{N_d}] \in \mathbb{C}^{N_x \times N_d}$, the modes can be obtained by first taking a singular value decomposition (SVD) of the weighted data matrix $\frac{1}{N_d}\mathbf{W}^{1/2}\hat{\mathbf{Q}}_k = \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^*$. The first N_d SPOD modes are then given by $\boldsymbol{\Psi}_k^{N_d} = \mathbf{W}^{-1/2}\mathbf{U}_k \in \mathbb{C}^{N_x \times N_d}$. Likewise, the diagonal matrix of the first N_d SPOD energies is given by the squares of the singular values, $\boldsymbol{\Lambda}_k^{N_d} = \boldsymbol{\Sigma}_k^2$.

2.3 SPOD modes for trajectory representation

Our interest in using SPOD modes stems from the fact that they provide a near-optimal representation of trajectories. The SPOD modes at each frequency form an approximate representation of the trajectory in the frequency domain. Accordingly, they can be used to give an approximate space-time representation of the trajectory in the time domain by taking an inverse DFT,

$$\tilde{\boldsymbol{q}}_j = \frac{1}{N_\omega} \sum_{k=0}^{N_\omega - 1} \boldsymbol{\Psi}_k \boldsymbol{a}_k e^{i\omega_k t_j}, \tag{2.9}$$

where $\tilde{\boldsymbol{q}}_{\mathcal{J}}$ is the approximate trajectory. It may be written in terms of a space-time decoder as $\tilde{\boldsymbol{q}}_{\mathcal{J}} = \boldsymbol{d}^{x,t}_{\mathcal{J}}(\boldsymbol{a}_{\mathcal{K}})$, where the j-th time of the decoder is given by

$$\boldsymbol{d}^{x,t}_j(\boldsymbol{a}_{\mathcal{K}}) = \frac{1}{N_\omega}\sum_{k=0}^{N_\omega-1} \boldsymbol{\Psi}_k \boldsymbol{a}_k e^{i\omega_k t_j}. \tag{2.10}$$

Hereafter, we use a subscript $\mathcal{K}$ to denote all frequencies, analogous to our use of the subscript $\mathcal{J}$ for all times. The relevant metric for accuracy is the spatial error averaged over the times in the interval, which is the square of the space-time norm of the difference. Due to Parseval's theorem, this is proportional to the error summed over frequencies,

$$\|\tilde{\boldsymbol{q}}_{\mathcal{J}} - \boldsymbol{q}_{\mathcal{J}}\|^2_{x,t} = \frac{1}{N_\omega}\sum_{k=0}^{N_\omega-1} \|\boldsymbol{\Psi}_k \boldsymbol{a}_k - \hat{\boldsymbol{q}}_k\|^2_x. \tag{2.11}$$

Taking the expected value of both sides and using (2.6), we have

$$\mathbb{E}\left[\|\tilde{\boldsymbol{q}}_{\mathcal{J}} - \boldsymbol{q}_{\mathcal{J}}\|^2_{x,t}\right] = \frac{1}{N_\omega}\sum_{k=0}^{N_\omega-1}\sum_{m=r_k+1}^{N_x} \lambda_{km}. \tag{2.12}$$

Thus, the expected accuracy of the trajectory representation is determined by the energies of the truncated SPOD modes. This informs the best allocation of modes across frequencies: for the best accuracy given a total number of modes, one should retain the modes with the highest energies over all frequencies. We denote the average number of modes retained by r, and denote the j-th largest SPOD energy at any frequency by $\tilde{\lambda}_j$. The number of modes retained at the k-th frequency r_k is then the number of modes at this frequency with energy greater than or equal to $\tilde{\lambda}_{(rN_\omega)}$,

$$r_k = |\{m : \lambda_{km} \geq \tilde{\lambda}_{(rN_\omega)}\}|, \tag{2.13}$$

where $|\cdot|$ denotes the cardinality of the set, and the parentheses in the subscript are a reminder that r and N_ω are multiplied and are not separate indices.

The SPOD trajectory representation (2.9) may be viewed as a rank-rN_ω space-time representation, i.e., it is a sum of spatiotemporal modes with constant coefficients of the form $\sum_{i=1}^{rN_\omega} \boldsymbol{\xi}_i a_i$ with $\boldsymbol{\xi}_i \in \mathbb{C}^{N_\omega N_x}$. In the SPOD representation, each spatiotemporal mode is an SPOD mode at some frequency ω_k multiplied by the associated oscillatory time dependence $e^{i\omega_k t}$. The most accurate space-time representation is space-time POD modes — these lead to lower trajectory representation error than any other space-time basis. The fact that motivates our use of SPOD modes is that as the time interval $T \to \infty$, the SPOD modes approach space-time POD modes [22, 23, 13]. This limit is approached fairly quickly in practice, so when T is long compared to the correlation time of the system, the SPOD modes provide a representation of trajectories that is nearly optimally accurate given the total number of coefficients used (see Figure 2 in Ref. [12]).

The efficacy of the SPOD modes in representing trajectories can be compared to that of POD modes by calculating the relative error produced by both encoding/decoding. The relative error is

$$\frac{\|\tilde{\boldsymbol{q}}_{\mathcal{J}} - \boldsymbol{q}_{\mathcal{J}}\|^2_{x,t}}{\|\boldsymbol{q}_{\mathcal{J}}\|^2_{x,t}}, \tag{2.14}$$

where $\tilde{\boldsymbol{q}}_{\mathcal{J}}$ is given by either the POD or SPOD encoding/decoding. In Figure 1, we show these errors as a function of r, the number of modes, averaged over 30 trajectories for the cubically

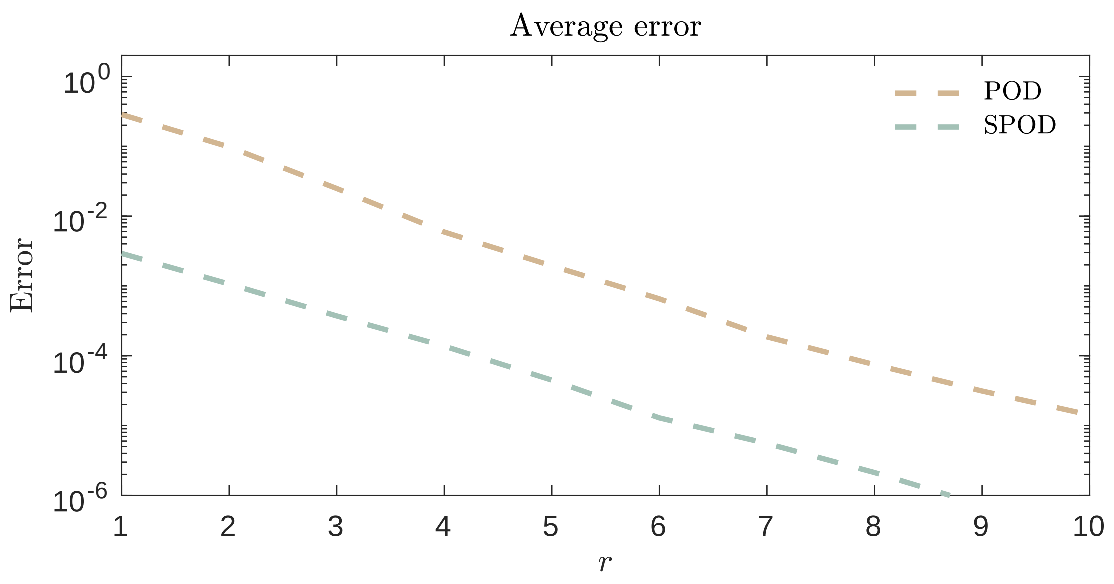

Figure 1: The relative error for the SPOD and POD trajectory encoders as a function of r, the number of modes used.

nonlinear Ginzburg-Landau system described later in the paper. Figure 1 demonstrates the superior representational power of SPOD modes over POD modes; with the same number of total coefficients, the SPOD representation results in more than two orders of magnitude lower error. The goal of the paper is to develop a reduced-order model for nonlinear systems that solves for the SPOD coefficients quickly and accurately, so as to recover the accuracy shown in Figure 1 at low cost.

2.4 Linear spectral solution operator projection

Here, we review our previous work on SSOP in the linear setting. The full details are given in Ref. [12].

The method aims to find, as accurately as possible, the SPOD coefficients of the solution $\boldsymbol{q}_{\mathcal{J}}$ (defined on a temporal grid) to the LTI system

$$\dot{\boldsymbol{q}} = \mathbf{A}(\boldsymbol{\mu})\boldsymbol{q} + \mathbf{B}\boldsymbol{f}, \tag{2.15}$$

where $\boldsymbol{f} \in \mathbb{C}^{N_f}$ is a known time-dependent forcing. $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{C}^{N_x \times N_x}$ and $\mathbf{B} \in \mathbb{C}^{N_x \times N_f}$ are the usual system matrices, the former of which depends affinely on the parameter vector $\boldsymbol{\mu} \in \mathbb{R}^{N_\mu}$. We suppress this parameter dependence for now; after deriving the nonlinear reduced-order model in Section 3, we reintroduce it and show how the reduced matrices involved may be computed quickly given $\boldsymbol{\mu}$.

The method derives from the fact that the SPOD coefficients are given by linear functionals of the initial condition and forcing. Specifically, the m-th SPOD coefficient at the k-th frequency is

$$a_{km} = \langle \boldsymbol{L}[\boldsymbol{q}_0, \mathbf{B}\boldsymbol{f}(t); t_{\mathcal{J}}], \boldsymbol{\psi}_{km} e^{i\omega_k t_{\mathcal{J}}} \rangle_{x,t}, \tag{2.16}$$

where $\boldsymbol{a}_k = [a_{k1}, \ldots, a_{kr_k}]^T$, and where the space-time inner product is (2.1). $\boldsymbol{L}[\boldsymbol{q}_0, \boldsymbol{g}(t); t_j]$ is a solution operator that maps the initial condition, forcing, and a time t_j to $\boldsymbol{q}(t_j)$ as

$$\boldsymbol{L}[\boldsymbol{q}_0, \boldsymbol{g}(t); t_j] = e^{\mathbf{A}t_j}\boldsymbol{q}_0 + \int_0^{t_j} e^{\mathbf{A}(t_j - t')}\boldsymbol{g}(t')\,\mathrm{d}t'. \tag{2.17}$$

Analytic progress can be made by breaking (2.16) into a DFT of the solution operator and a spatial inner product as

$$\hat{\boldsymbol{q}}_k = \sum_{j=0}^{N_\omega-1} \left(e^{\mathbf{A} j\Delta t}\boldsymbol{q}_0 + \int_0^{j\Delta t} e^{\mathbf{A}(j\Delta t - t')}\mathbf{B}\boldsymbol{f}(t')\, dt' \right) e^{-i\omega_k j \Delta t}. \tag{2.18a}$$

$$\boldsymbol{a}_k = \boldsymbol{\Psi}_k^* \mathbf{W} \hat{\boldsymbol{q}}_k \tag{2.18b}$$

where we the left multiplication by $\boldsymbol{\Psi}_k^*\mathbf{W}$ effects the inner product for all modes at the k-th frequency. The sum in (2.18a) can be computed analytically assuming that *(i)* the forcing is of the form $\boldsymbol{f}(t) = \sum_{l=0}^{N_\omega-1} \hat{\boldsymbol{f}}_l e^{i\omega_l t}$, and *(ii)* that $i\omega_l \mathbf{I} - \mathbf{A}$ is invertible for all included l (this latter assumption can be relaxed). Under these assumptions, (2.18a) can be written

$$\hat{\boldsymbol{q}}_k = \mathbf{R}_k \mathbf{B} \hat{\boldsymbol{f}}_k + \left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k \mathbf{I})\Delta t} \right)^{-1} \left(\mathbf{I} - e^{\mathbf{A}T} \right) \left(\boldsymbol{q}_0 - \frac{1}{N_\omega} \sum_{l=0}^{N_\omega-1} \mathbf{R}_l \mathbf{B} \hat{\boldsymbol{f}}_l \right). \tag{2.19}$$

Here, $\mathbf{R}_k = (i\omega\mathbf{I} - \mathbf{A})^{-1}$ is the resolvent operator. Though assumption *(i)* will seldom hold in practice, we have found in our numerical experiments that this introduces minimal error. Using (2.7), we obtain an equation for the SPOD coefficients,

$$\boldsymbol{a}_k = \boldsymbol{\Psi}_k^* \mathbf{W} \mathbf{R}_k \mathbf{B} \hat{\boldsymbol{f}}_k + \boldsymbol{\Psi}_k^* \mathbf{W} \left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k \mathbf{I})\Delta t} \right)^{-1} \left(\mathbf{I} - e^{\mathbf{A}T} \right) \left(\boldsymbol{q}_0 - \frac{1}{N_\omega} \sum_{l=0}^{N_\omega-1} \mathbf{R}_l \mathbf{B} \hat{\boldsymbol{f}}_l \right). \tag{2.20}$$

At this point, a number of approximations must be made to make the operators in (2.20) tractable to compute for large systems, i.e., to reduce the offline cost, and to make the equation fast to evaluate for all frequencies online. If the system in question is small enough for direct computation of the matrices in (2.20), then only the last approximation described in this subsection is necessary.

First, the operator $\boldsymbol{\Psi}_k^* \mathbf{W} \mathbf{R}_k \mathbf{B}$ must be approximated. We do this by making use of the data matrices used to compute the SPOD modes to obtain the action of the resolvent operator within a subspace, then form an approximation by projecting into this subspace. Specifically, we form the matrix $\mathbf{G}_k = \mathbf{L}_k \hat{\mathbf{Q}}_k \in \mathbb{C}^{N_x \times N_d}$, where $\mathbf{L}_k = i\omega_k \mathbf{I} - \mathbf{A}$. This implies

$$\hat{\mathbf{Q}}_k = \mathbf{R}_k \mathbf{G}_k, \tag{2.21}$$

meaning we have the action of the resolvent operator on the subspace spanned by the columns of $\mathbf{G}_k$. Using (2.21), we approximate the resolvent using

$$\mathbf{R}_k \approx \hat{\mathbf{Q}}_k \mathbf{G}_k^+, \tag{2.22}$$

where $\mathbf{G}_k^+ = (\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \mathbf{G}_k^* \mathbf{W}$ is the Moore-Penrose pseudoinverse of $\mathbf{G}_k$. It may be shown of this approximation that

$$\hat{\mathbf{Q}}_k \mathbf{G}_k^+ = \mathbf{R}_k \mathbf{P}_{G_k}, \tag{2.23}$$

where $\mathbf{P}_{G_k} = \mathbf{G}_k(\mathbf{G}_k^*\mathbf{W}\mathbf{G}_k)^{-1}\mathbf{G}_k^*\mathbf{W}$ is the orthogonal projection operator into the column space of $\mathbf{G}_k$. This implies that the approximation (2.22) of the resolvent is accurate to the extent that it is applied to vectors near the column space of $\mathbf{G}_k$. The approximation of $\mathbf{\Psi}_k^*\mathbf{W}\mathbf{R}_k\mathbf{B}$ is then

$$\mathbf{E}_k = \mathbf{\Psi}_k^*\mathbf{W}\hat{\mathbf{Q}}_k\mathbf{G}^+\mathbf{B} \approx \mathbf{\Psi}_k^*\mathbf{W}\mathbf{R}_k\mathbf{B}. \tag{2.24}$$

Next, we approximate $\mathbf{\Psi}_k^*\mathbf{W}\left(\mathbf{I} - e^{(\mathbf{A}-i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\mathbf{A}T}\right)$. This is accomplished by the Galerkin approximation

$$\mathbf{P}_k\left(\mathbf{I} - e^{(\tilde{\mathbf{A}}-i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\tilde{\mathbf{A}}T}\right)\mathbf{\Psi}_k^{N_d*}\mathbf{W} \approx \mathbf{\Psi}_k^*\mathbf{W}\left(\mathbf{I} - e^{(\mathbf{A}-i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\mathbf{A}T}\right), \tag{2.25}$$

where $\tilde{\mathbf{A}} = \mathbf{\Psi}_k^{N_d*}\mathbf{W}\mathbf{A}\mathbf{\Psi}_k^{N_d} \in \mathbb{C}^{N_d\times N_d}$ and $\mathbf{P}_k = \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix} \in \mathbb{R}^{r_k\times N_d}$ selects the first r_k rows of the matrix it operates on. In Appendix C, we detail a more accurate, but more costly, approximation of $\mathbf{\Psi}_k^*\mathbf{W}\left(\mathbf{I} - e^{(\mathbf{A}-i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\mathbf{A}T}\right)$.

The purpose of the previous two approximations is to alleviate the offline cost of building the operators in (2.20). In order to avoid poor online scaling, we approximate the term $\boldsymbol{q}_0 - \frac{1}{N_\omega}\sum_{l=0}^{N_\omega-1}\mathbf{R}_l\mathbf{B}\hat{\boldsymbol{f}}_l$ by representing it in a rank-p_1 reduced spatial basis $\mathbf{\Phi} \in \mathbb{C}^{N_x\times p}$. We refer to $\mathbf{\Phi}$ as the intermediary basis, and we take it to be the (space-only) POD modes. This is helpful because with it, evaluating the impact of $\boldsymbol{q}_0 - \frac{1}{N_\omega}\sum_{l=0}^{N_\omega-1}\mathbf{R}_l\mathbf{B}\hat{\boldsymbol{f}}_l$ for each frequency scales with p_1 rather than with N_x. The approximation is

$$\mathbf{\Phi}\mathbf{\Phi}^*\boldsymbol{q}_0 - \frac{1}{N_\omega}\sum_{l=0}^{N_\omega-1}\mathbf{\Phi}\mathbf{J}_l\hat{\boldsymbol{f}}_l \approx \boldsymbol{q}_0 - \frac{1}{N_\omega}\sum_{l=0}^{N_\omega-1}\mathbf{R}_l\mathbf{B}\hat{\boldsymbol{f}}_l, \tag{2.26}$$

where $\mathbf{J}_k \in \mathbb{C}^{p_1\times N_f}$ is an approximation of $\mathbf{\Phi}^*\mathbf{W}\mathbf{R}_k\mathbf{B}$. Using the same approximation of the resolvent from before, it is defined as

$$\mathbf{J}_k = \mathbf{\Phi}^*\mathbf{W}\hat{\mathbf{Q}}_k\mathbf{G}_k^+\mathbf{B}. \tag{2.27}$$

We also define the matrix $\mathbf{H}_k \in \mathbb{C}^{r_k\times p_1}$ as

$$\mathbf{H}_k = \mathbf{P}_k\left(\mathbf{I} - e^{(\tilde{\mathbf{A}}-i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\tilde{\mathbf{A}}T}\right)\mathbf{\Psi}_k^{N_d*}\mathbf{W}\mathbf{\Phi}. \tag{2.28}$$

The ROM equations are then

$$\tilde{\boldsymbol{a}}_k = \mathbf{E}_k\hat{\boldsymbol{f}}_k + \mathbf{H}_k\left(\mathbf{\Phi}^*\boldsymbol{q}_0 - \frac{1}{N_\omega}\sum_{l=0}^{N_\omega-1}\mathbf{J}_l\hat{\boldsymbol{f}}_l\right), \tag{2.29}$$

where the $\tilde{(\cdot)}$ indicates the coefficients above are an approximation of the SPOD coefficients for the trajectory — they are approximate due to the operator approximations introduced above. In (2.29), the term in parentheses need only be computed once, and the online calculations scale as $\mathcal{O}\big(N_\omega(rN_f + rp_1 + N_f\log N_\omega)\big)$. Pseudocode for the offline and online phases of the method can be found in Ref. [12]. The pseudocode given in Appendix D can also be used for the linear problem by disregarding the nonlinearity.

3 Nonlinear spectral solution operator projection

We now consider systems of the form

$$\dot{\boldsymbol{q}} = \mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{f} + \boldsymbol{n}(\boldsymbol{q}), \tag{3.1}$$

where $\boldsymbol{n} : \mathbb{C}^{N_x} \to \mathbb{C}^{N_x}$ is some nonlinear function of the state. The parameter dependence in the linear operator $\mathbf{A}$ is temporarily suppressed, and we assume the nonlinear operator is not parameter-dependent. We again assume that the system has been transformed such that the solution has zero mean. An additional transformation may be useful if the linear operator has unstable eigenvalues. The transformation is $\mathbf{A} \to \mathbf{A} - \alpha\mathbf{I}$, $\boldsymbol{n}(\boldsymbol{q}) \to \boldsymbol{n}(\boldsymbol{q}) + \alpha\boldsymbol{q}$, where α is to the right of the spectrum of $\mathbf{A}$, i.e., $\alpha > \max \Re(\lambda(\mathbf{A}))$. Without this transformation, the terms related to the linear operator may dominate the solution.

Our aim is to generalize SSOP to nonlinear systems of the form (3.1). This is accomplished by treating the nonlinear term as an additional forcing on the system, but one that depends on the trajectory. The linear ROM (2.29) is amended by adding a term corresponding to the nonlinearity at frequency ω_k that results from the SPOD coefficients at all frequencies. As we show in Section 3.1, this results in a system that must be solved for all the SPOD coefficients at once. We then discuss two methods for approximating the effect of nonlinearity in Section 3.2. Affine parameter dependence is discussed in Section 3.3. We present a method for solving the nonlinear system of equations in Section 3.4, and discuss scaling in Section 3.5.

The method is shown schematically in Figure 2, and the details of the components therein will be discussed in the remainder of Section 3. We note here that in the bottom-right box in the figure, an inner product is equated to a vector, and the second argument of the inner product is a matrix. In this shorthand notation, the m-th element of the vector is the inner product of the first argument with the m-th column of the matrix.

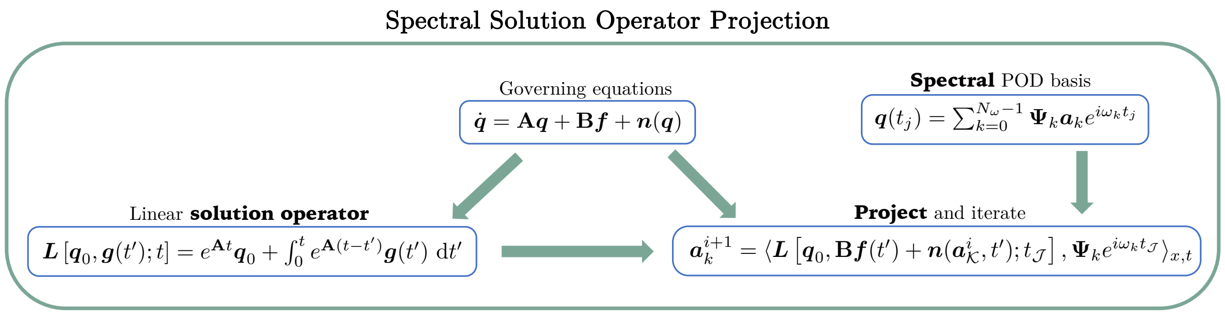
Spectral Solution Operator Projection
Governing equations
$\dot{\boldsymbol{q}} = \mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{f} + \boldsymbol{n}(\boldsymbol{q})$
Spectral POD basis
$\boldsymbol{q}(t_j) = \sum_{k=0}^{N_\omega - 1} \boldsymbol{\Psi}_k \boldsymbol{a}_k e^{i\omega_k t_j}$
Linear **solution operator**
$\boldsymbol{L}\left[\boldsymbol{q}_0, \boldsymbol{g}(t'); t\right] = e^{\mathbf{A}t}\boldsymbol{q}_0 + \int_0^t e^{\mathbf{A}(t-t')}\boldsymbol{g}(t')\, \mathrm{d}t'$
Project and iterate
$\boldsymbol{a}_k^{i+1} = \langle \boldsymbol{L}\left[\boldsymbol{q}_0, \mathbf{B}\boldsymbol{f}(t') + \boldsymbol{n}(\boldsymbol{a}_{\mathcal{K}}^i, t'); t_{\mathcal{J}}\right], \boldsymbol{\Psi}_k e^{i\omega_k t_{\mathcal{J}}} \rangle_{x,t}$

Figure 2: **Spectral Solution Operator Projection**: A system of nonlinear algebraic equations is obtained by performing a space-time projection of a linear solution operator onto the set of SPOD modes used to encode trajectories. The linear solution operator returns the solution to the linear system given the initial condition and forcing. Nonlinearities are handled in this framework as a state-dependent forcing. The system is solved via a fixed-point iteration, which may be viewed as a perturbation series around the linear solution.

3.1 Formulation

To handle nonlinearity within the framework described above, we begin with the implicit equation

$$\boldsymbol{q}_j = \boldsymbol{L}\left[\boldsymbol{q}_0, \mathbf{B}\boldsymbol{f}(t) + \boldsymbol{n}(\boldsymbol{q}(t)); t_j\right], \tag{3.2}$$

where $\boldsymbol{L}$ is the solution operator given in (2.17). This formulation treats the nonlinearity as an additional forcing on the linear part of the system, but, crucially, one that depends on the state. The m-th SPOD coefficient at the k-th frequency is now

$$a_{km} = \langle \boldsymbol{L}\left[\boldsymbol{q}_0, \mathbf{B}\boldsymbol{f}(t) + \boldsymbol{n}(\boldsymbol{q}(t)); t_{\mathcal{J}}\right], \boldsymbol{\psi}_{km} e^{i\omega_k t_{\mathcal{J}}}\rangle_{x,t}. \tag{3.3}$$

In order to arrive at a closed system of equations, $\boldsymbol{n}(\boldsymbol{q}(t))$ must be approximated using the SPOD coefficients, and we return to this closure in Section 3.2. First, as in the linear case, we evaluate the space-time inner product by taking a temporal DFT of the solution operator, followed by a spatial inner product:

$$\hat{\boldsymbol{q}}_k = \sum_{j=0}^{N_\omega - 1}\left(e^{\mathbf{A}j\Delta t}\boldsymbol{q}_0 + \int_0^{t_j} e^{\mathbf{A}(t_j - t')}\mathbf{B}\boldsymbol{f}(t') + \boldsymbol{n}(\boldsymbol{q}(t'))\, dt'\right)e^{-i\omega_k t_j}. \tag{3.4a}$$

$$\boldsymbol{a}_k = \mathbf{\Psi}_k^* \mathbf{W}\hat{\boldsymbol{q}}_k. \tag{3.4b}$$

With the two assumptions from Section 2.4, as well as the additional assumption *(iii)* that the nonlinearity can be written in the form $\boldsymbol{n}(\boldsymbol{q}(t)) = \sum_{l=0}^{N_\omega - 1}\hat{\boldsymbol{n}}_l e^{i\omega_l t}$ where $\hat{\boldsymbol{n}}_l \in \mathbb{C}^{N_x}$, we show in Appendix A that the k-th frequency of the solution is given by

$$\begin{aligned}\hat{\boldsymbol{q}}_k =& \mathbf{R}_k\left(\mathbf{B}\hat{\boldsymbol{f}}_k + \hat{\boldsymbol{n}}_k^q(\hat{\boldsymbol{q}}_{\mathcal{K}})\right)\\ &+ \left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\mathbf{A}T}\right)\left(\boldsymbol{q}_0 - \frac{1}{N_\omega}\sum_{l=0}^{N_\omega - 1}\mathbf{R}_l\left(\mathbf{B}\hat{\boldsymbol{f}}_l + \hat{\boldsymbol{n}}_l^q(\boldsymbol{q}_{\mathcal{K}})\right)\right).\end{aligned} \tag{3.5}$$

Here, $\hat{\boldsymbol{n}}_k^q : \mathbb{C}^{N_x N_\omega} \to \mathbb{C}^{N_x}$ returns the nonlinearity at frequency ω_k that results from $\hat{\boldsymbol{q}}_{\mathcal{K}}$, the concatenation of all frequencies of the trajectory. The superscript is used to distinguish this function from a similar one introduced in the next subsection. One way to compute $\hat{\boldsymbol{n}}_k^q$ is to first take the inverse DFT of $\hat{\boldsymbol{q}}_{\mathcal{K}}$, giving the trajectory in the time domain $\boldsymbol{q}_{\mathcal{J}}$. The k-th frequency of the nonlinearity $\hat{\boldsymbol{n}}_k^q$ is given by then computing $\boldsymbol{n}(\boldsymbol{q}_{\mathcal{J}})$ and taking the k-th component of the DFT of the result. Similar to the assumption of the form of the forcing, assumption *(iii)* will likely not hold in practice, but we have found that the error introduced is minimal.

Inserting (3.5) into (3.4b), we have

$$\begin{aligned}\tilde{\boldsymbol{a}}_k =& \mathbf{\Psi}_k^*\mathbf{W}\mathbf{R}_k\left(\mathbf{B}\hat{\boldsymbol{f}}_k + \hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})\right)\\ &+ \mathbf{\Psi}_k^*\mathbf{W}\left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k)\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\mathbf{A}T}\right)\left(\boldsymbol{q}_0 - \frac{1}{N_\omega}\sum_{l=0}^{N_\omega - 1}\mathbf{R}_l\left(\mathbf{B}\hat{\boldsymbol{f}}_l + \hat{\boldsymbol{n}}_l(\tilde{\boldsymbol{a}}_{\mathcal{K}})\right)\right).\end{aligned} \tag{3.6}$$

Here, $\tilde{\boldsymbol{a}}_{\mathcal{K}}$ is the (ROM approximation of the) vector of retained SPOD coefficients at all frequencies concatenated together. With the reduction, the nonlinearity at each frequency must be computed as a function of the vector $\tilde{\boldsymbol{a}}_{\mathcal{K}}$, not $\hat{\boldsymbol{q}}$, i.e., $\hat{\boldsymbol{n}}_k : \mathbb{C}^{rN_\omega} \to \mathbb{C}^{N_x}$. Two different methods for approximating the nonlinearity at each frequency, given the SPOD coefficients, are described in the following subsection.

The operators in (3.6) must be approximated if the system is large. With the same approximations used in the linear case, we have,

$$\tilde{\boldsymbol{a}}_k = \mathbf{E}_k \hat{\boldsymbol{f}}_k + \boldsymbol{\Psi}_k^* \mathbf{W} \mathbf{R}_k \hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}}) + \mathbf{H}_k \left(\boldsymbol{\Phi}^* \boldsymbol{q}_0 - \sum_{l=0}^{N_\omega - 1} \mathbf{J}_l \hat{\boldsymbol{f}}_l + \boldsymbol{\Phi}^* \mathbf{W} \mathbf{R}_k \hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}}) \right). \tag{3.7}$$

3.2 Constructing the nonlinearity

To close the reduced-order model equations (3.7), we must compute the k-th frequency of the nonlinearity $\hat{\boldsymbol{n}}_k$ that results from the SPOD coefficients. With no approximation, the nonlinearity is given in terms of the k-th frequency of the nonlinear function acting on the decoding of the coefficients

$$\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}}) = \mathrm{DFT}_k \left[\boldsymbol{n}(\boldsymbol{d}^{x,t}_{\mathcal{J}}(\tilde{\boldsymbol{a}}_{\mathcal{K}})) \right] = \mathrm{DFT}_k \left[\boldsymbol{n} \left(\sum_{l=0}^{N_\omega - 1} \boldsymbol{\Psi}_l \tilde{\boldsymbol{a}}_l e^{i\omega_l t_{\mathcal{J}}} \right) \right]. \tag{3.8}$$

However, computing the nonlinearity this way is expensive since it involves expanding to the N_x-dimensional space. The problem of obtaining $\hat{\boldsymbol{n}}_k$ without doing computations in this space is entirely analogous to approximating the nonlinearity at a given time step given the POD coefficients at that time step. Accordingly, we use hyper-reduction, specifically, the discrete empirical interpolation method [6] (DEIM) in the case that the nonlinearity is not a quadratic form. If the nonlinearity is a quadratic form, we use (sparsified) triadic interactions. We note that it may be appropriate to apply a dealiasing technique [29] in computing the nonlinearity, such as truncating the sum in (3.8). The details and effectiveness of dealiasing depend on the nature of the nonlinearity. We therefore ignore this issue in dealing with general nonlinearities, but implicitly employ a standard 2/3-rule dealiasing for the quadratic nonlinearity. We note here that the function $\boldsymbol{n}(\tilde{\boldsymbol{a}}_{\mathcal{K}}, t)$ in Figure 2 is $\hat{\boldsymbol{n}}_{\mathcal{K}}(\tilde{\boldsymbol{a}}_{\mathcal{K}})$ transformed to the time domain.

3.2.1 Hyper-reduction

Here, we describe a DEIM-based approximation of $\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})$. We assume that the nonlinearity is a local function of the state, i.e., each component of the nonlinearity is a function of only the same component of the state $n_i(\boldsymbol{q}) = n(q_i)$. The approach can trivially be extended to cases where the nonlinearity is a local function of the state and some number of linear operations on it, such as $n_i(\boldsymbol{q}) = n(q_i, (\mathbf{D}^1 \boldsymbol{q})_i)$, where $\mathbf{D}^1$ is a differentiation matrix.

We build the matrices $\mathbf{U}^n \in \mathbb{C}^{N_x \times p_2}$ and $\mathbf{P}^{nT} \in \mathbb{R}^{p_2 \times N_x}$ following the standard DEIM algorithm using snapshots of the nonlinearity and p_2 sample points. The matrix $\mathbf{U}^n$ is the POD modes for the nonlinearity and the matrix $\mathbf{P}^n = [\boldsymbol{e}_{s_1}, \ldots, \boldsymbol{e}_{s_{p_2}}]^T$ is a the sampling matrix, where $\boldsymbol{e}_j$ is the j-th canonical unit vector and s_i is the i-th sample point given by DEIM. The standard DEIM approximation of the nonlinearity at time t given the sampled state is $\boldsymbol{n}(\boldsymbol{q}) \approx \mathbf{U}^n (\mathbf{P}^{nT} \mathbf{U}^n)^{-1} \boldsymbol{n}(\mathbf{P}^{nT} \boldsymbol{q})$, where $\boldsymbol{n}(\mathbf{P}^{nT} \boldsymbol{q}) \in \mathbb{C}^{p_2}$.

We adapt this methodology to approximating $\hat{\boldsymbol{n}}_k$ as follows. The nonlinearity at each time t_j is approximated using DEIM and the state at t_j that is implied by the SPOD coefficients. Then,

$\hat{\boldsymbol{n}}_k$ is obtained using the DFT of the nonlinearity at all times as

$$\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}}) \approx \mathrm{DFT}_k \left[\mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1} \mathbf{P}^{nT} \boldsymbol{n} \left(\sum_{l=0}^{N_\omega - 1} \boldsymbol{\Psi}_l \tilde{\boldsymbol{a}}_l e^{i\omega_l t_{\mathcal{J}}} \right) \right]. \tag{3.9}$$

The right-hand side of (3.9) is expensive to evaluate as written. However, leveraging the locality of the nonlinearity, the sampling operator $\mathbf{P}^{nT}$ may be applied directly to the SPOD modes, and the matrix $\mathbf{U}^n(\mathbf{P}^{nT}\mathbf{U}^n)^{-1}$ may be moved outside the DFT, as it is independent of time. Then, we have

$$\begin{aligned} &\mathrm{DFT}_k \left[\mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1} \mathbf{P}^{nT} \boldsymbol{n} \left(\sum_{l=0}^{N_\omega - 1} \boldsymbol{\Psi}_l \tilde{\boldsymbol{a}}_l e^{i\omega_l t_{\mathcal{J}}} \right) \right] \\ =&\mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1} \mathrm{DFT}_k \left[\boldsymbol{n} \left(\sum_{l=0}^{N_\omega - 1} \mathbf{P}^{nT} \boldsymbol{\Psi}_l \tilde{\boldsymbol{a}}_l e^{i\omega_l t_{\mathcal{J}}} \right) \right]. \end{aligned} \tag{3.10}$$

With the appropriate precomputed operators, evaluating the expression on the right does not scale with N_x, as desired. Three operators must be precomputed. First, the sampling matrix applied to the SPOD modes, $\mathbf{S}_k = \mathbf{P}^{nT}\boldsymbol{\Psi}_k$, must be precomputed for each frequency. Next, the matrix $\boldsymbol{\Psi}_k^* \mathbf{W} \mathbf{R}_k \mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1}$ must be both precomputed and approximated for each frequency. Using the approximation $\mathbf{R}_k \approx \hat{\mathbf{Q}}_k \mathbf{G}_k^+$ established in Section 2.4, we have

$$\mathbf{N}_k = \boldsymbol{\Psi}_k^* \mathbf{W} \hat{\mathbf{Q}}_k \mathbf{G}_k^+ \mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1} \approx \boldsymbol{\Psi}_k^* \mathbf{W} \mathbf{R}_k \mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1} \in \mathbb{C}^{r_k \times p_2}. \tag{3.11}$$

Similarly, the matrix $\boldsymbol{\Phi}^* \mathbf{W} \mathbf{R}_k \mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1}$ must be precomputed and approximated at each frequency. Using the same approximation of the action of the resolvent, we have

$$\mathbf{M}_k = \boldsymbol{\Phi}^* \mathbf{W} \hat{\mathbf{Q}}_k \mathbf{G}_k^+ \mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1} \approx \boldsymbol{\Phi}^* \mathbf{W} \mathbf{R}_k \mathbf{U}^n (\mathbf{P}^{nT}\mathbf{U}^n)^{-1} \in \mathbb{C}^{p_1 \times p_2}. \tag{3.12}$$

With these approximations in place, the nonlinear system with the DEIM approximation of the nonlinearity is

$$\tilde{\boldsymbol{a}}_k = \mathbf{E}_k \hat{\boldsymbol{f}}_k + \mathbf{N}_k \hat{\boldsymbol{n}}_k^s(\tilde{\boldsymbol{a}}_{\mathcal{K}}) + \mathbf{H}_k \left(\boldsymbol{\Phi}^* \boldsymbol{q}_0 - \sum_{l=0}^{N_\omega - 1} \mathbf{J}_l \hat{\boldsymbol{f}}_l + \mathbf{M}_l \hat{\boldsymbol{n}}_l^s(\tilde{\boldsymbol{a}}_{\mathcal{K}}) \right), \tag{3.13}$$

where the sampled nonlinearity $\hat{\boldsymbol{n}}_k^s \in \mathbb{C}^{p_2}$ is calculated as $\hat{\boldsymbol{n}}_k^s = \mathrm{DFT}_k \left[\boldsymbol{n} \left(\sum_{l=0}^{N_\omega - 1} \mathbf{S}_l \tilde{\boldsymbol{a}}_l e^{i\omega_l t_{\mathcal{J}}} \right) \right]$.

3.2.2 Sparse triadic interactions

Many systems of interest have quadratic nonlinearities, i.e., ones of the form $\boldsymbol{n}(\boldsymbol{q}) = \boldsymbol{b}(\boldsymbol{q}, \boldsymbol{q})$, where $\boldsymbol{b} : \mathbb{C}^{N_x} \times \mathbb{C}^{N_x} \to \mathbb{C}^{N_x}$ is a bilinear function. In these systems, $\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})$ can be computed exactly in the reduced space by summing over triadic interactions. These sums can be costly, so we discuss a strategy for discarding many of these triadic interactions. The resulting approximation of $\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})$ is likely to be more accurate in many applications than the DEIM-based approximation described above.

Expressing (3.8) in terms of $\boldsymbol{b}$ gives

$$\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}}) = \mathrm{DFT}_k\left[\boldsymbol{b}\left(\sum_{l=0}^{N_\omega-1}\boldsymbol{\Psi}_l\tilde{\boldsymbol{a}}_l e^{i\omega_l t_{\mathcal{J}}}, \sum_{i=0}^{N_\omega-1}\boldsymbol{\Psi}_i\tilde{\boldsymbol{a}}_i e^{i\omega_i t_{\mathcal{J}}}\right)\right]. \tag{3.14}$$

After leveraging the bilinearity, we have

$$\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}}) = \mathrm{DFT}_k\left[\sum_{l=0}^{N_\omega-1}\sum_{i=0}^{N_\omega-1} e^{i(\omega_l+\omega_i)t_{\mathcal{J}}}\boldsymbol{b}\left(\boldsymbol{\Psi}_l\tilde{\boldsymbol{a}}_l, \boldsymbol{\Psi}_i\tilde{\boldsymbol{a}}_i\right)\right]. \tag{3.15}$$

$\mathrm{DFT}_k[\cdot]$ only depends on pairs of frequencies where $l+i \equiv k \pmod{N_\omega}$, however some of these terms are generated by aliasing. After expanding the SPOD mode sums, making use of the bilinearity, and dropping the aliasing terms, the nonlinear term becomes

$$\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}}) = \sum_{\omega_l+\omega_i=\omega_k}\sum_{m=1}^{r_l}\sum_{n=1}^{r_i} a_{lm}a_{in}\boldsymbol{b}\left(\boldsymbol{\psi}_{lm}, \boldsymbol{\psi}_{in}\right). \tag{3.16}$$

The terms involving $\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})$ in the spatially reduced equation for the SPOD coefficients (3.7) are $\boldsymbol{\Psi}_k^*\mathbf{W}\mathbf{R}_k\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})$ and $\boldsymbol{\Phi}^*\mathbf{W}\mathbf{R}_k\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})$. To evaluate these terms quickly online, we precompute the vectors

$$\boldsymbol{n}_{klmn} = \boldsymbol{\Psi}_k^*\mathbf{W}\hat{\mathbf{Q}}_k\mathbf{G}_k^+\boldsymbol{b}\left(\boldsymbol{\psi}_{lm}, \boldsymbol{\psi}_{in}\right) \in \mathbb{C}^{r_k}, \tag{3.17a}$$

$$\boldsymbol{m}_{klmn} = \boldsymbol{\Phi}^*\mathbf{W}\hat{\mathbf{Q}}_k\mathbf{G}_k^+\boldsymbol{b}\left(\boldsymbol{\psi}_{lm}, \boldsymbol{\psi}_{in}\right) \in \mathbb{C}^{p_1}, \tag{3.17b}$$

where $\hat{\mathbf{Q}}_k\mathbf{G}_k^+$ is the approximation of $\mathbf{R}_k$ used elsewhere in the paper. In (3.17), we omit an 'i' subscript because the frequency ω_i is determined by ω_k and ω_l.

In many physical systems, the great majority of the triadic interactions are of negligible magnitude [36]. Accordingly, many of the terms in the sum in (3.16) may have little impact on $\hat{\boldsymbol{n}}_k(\tilde{\boldsymbol{a}}_{\mathcal{K}})$, due to either $|a_{lm}a_{in}|$ or $\|\boldsymbol{b}\left(\boldsymbol{\psi}_{lm}, \boldsymbol{\psi}_{in}\right)\|_x$ being small. In order to accelerate the computation of $\hat{\boldsymbol{n}}_k$, it is useful to ignore these irrelevant terms in the online stage of the method. We set a threshold ϵ and only retain terms for which

$$\|\boldsymbol{n}_{klmn}\|_x^2\lambda_{lm}\lambda_{in} > \epsilon. \tag{3.18}$$

The quantity $\|\boldsymbol{n}_{klmn}\|_x^2\lambda_{lm}\lambda_{in}$ is an easy-to-evaluate proxy for $\mathbb{E}[\|\boldsymbol{n}_{klmn}a_{lm}a_{in}\|_x^2]$, and the two are equivalent in the case that $\mathbb{E}[|a_{lm}a_{in}|^2] = \mathbb{E}[|a_{lm}|^2]\mathbb{E}[|a_{in}|^2]$. We denote the set of tuples (l,m,n) that meet the criterion (3.18) by

$$\mathcal{N}_k = \{(l,m,n) : \|\boldsymbol{n}_{klmn}\|_x^2\lambda_{lm}\lambda_{in} > \epsilon\}. \tag{3.19}$$

The nonlinear system with the nonlinearity approximated using sparse triadic interactions is

$$\tilde{\boldsymbol{a}}_k = \mathbf{E}_k\hat{\boldsymbol{f}}_k + \sum_{(l,m,n)\in\mathcal{N}_k}\boldsymbol{n}_{klmn}a_{lm}a_{in} + \mathbf{H}_k\left(\boldsymbol{\Phi}^*\boldsymbol{q}_0 - \sum_{l=0}^{N_\omega-1}\mathbf{J}_l\hat{\boldsymbol{f}}_l + \sum_{(p,m,n)\in\mathcal{N}_l}\boldsymbol{m}_{lpmn}a_{lm}a_{in}\right), \tag{3.20}$$

where the index 'i' is implied by l via $\omega_i = \omega_k - \omega_l$.

A practical matter bears mentioning: the nonlinear term in a quadratically nonlinear full-order model will likely not be written in quadratic form. That is, the function $\boldsymbol{n}(\boldsymbol{q})$ will be available, but not the function $\boldsymbol{b}(\boldsymbol{q}_1, \boldsymbol{q}_2)$. In fact, the latter is not uniquely defined. Defining $\boldsymbol{b}$ to also be symmetric, i.e., $\boldsymbol{b}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \boldsymbol{b}(\boldsymbol{q}_2, \boldsymbol{q}_1)\ \forall \boldsymbol{q}_1, \boldsymbol{q}_2$, makes it both unique and easy to calculate from $\boldsymbol{n}$. The unique symmetric bilinear function with $\boldsymbol{b}(\boldsymbol{q}, \boldsymbol{q}) = \boldsymbol{n}(\boldsymbol{q})$ is

$$\boldsymbol{b}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \frac{1}{2}\left[\boldsymbol{n}(\boldsymbol{q}_1 + \boldsymbol{q}_2) - \boldsymbol{n}(\boldsymbol{q}_1) - \boldsymbol{n}(\boldsymbol{q}_2)\right]. \tag{3.21}$$

3.3 Affine parametric dependence

Until now, we have suppressed the dependence of $\mathbf{A}$ on the parameter vector $\boldsymbol{\mu} \in \mathbb{R}^{N_\mu}$. Here, we show that by precomputing certain operators, the matrices in the ROM can quickly be computed given a new parameter vector. We assume this parameter dependence is affine, i.e., $\mathbf{A}(\boldsymbol{\mu})$ can be expressed as a linear combination of M_μ matrices as

$$\mathbf{A}(\boldsymbol{\mu}) = \sum_{j=1}^{M_\mu} \zeta_j(\boldsymbol{\mu})\mathbf{A}_j, \tag{3.22}$$

with scalar coefficients ζ_j that depend on $\boldsymbol{\mu}$. The components of the ROM that inherit this parameter dependence are $\mathbf{E}_k$, $\mathbf{J}_k$, $\mathbf{N}_k$, $\mathbf{M}_k$, $\boldsymbol{n}_{klmn}$, $\boldsymbol{m}_{klmn}$, $\mathbf{H}_k$, and $\tilde{\mathbf{A}}_k$. The parameter dependence of all but the latter two is through the matrix $\mathbf{G}_k^+$.

We define the matrices

$$\mathbf{G}_{kj} = \left(\delta_{1j} i\omega_k \mathbf{I} - \mathbf{A}_j\right)\hat{\mathbf{Q}}_k, \tag{3.23}$$

for $j \in \{1, \dots, M_\mu\}$, where δ is the Kronocker delta. The weighted sum of these matrices is $\mathbf{G}_k$, i.e., $\sum_{j=1}^{M_\mu} \zeta_j(\boldsymbol{\mu})\mathbf{G}_{kj} = \mathbf{G}_k$, and they are elements of $\mathbb{C}^{N_x \times N_d}$. We then define the following quantities,

$$\mathbf{G}^G_{kij} = \mathbf{G}^*_{ki}\mathbf{W}\mathbf{G}_{kj} \in \mathbb{C}^{N_d \times N_d}, \tag{3.24a}$$

$$\mathbf{B}^G_{kj} = \mathbf{G}^*_{kj}\mathbf{W}\mathbf{B} \in \mathbb{C}^{N_d \times N_f}, \tag{3.24b}$$

$$\mathbf{U}^G_{kj} = \mathbf{G}^*_{kj}\mathbf{W}\mathbf{U}^n \in \mathbb{C}^{N_d \times p_2}, \tag{3.24c}$$

$$\boldsymbol{b}^G_{kjlmin} = \mathbf{G}^*_{kj}\mathbf{W}\boldsymbol{b}(\boldsymbol{\psi}_{lm}, \boldsymbol{\psi}_{in}) \in \mathbb{C}^{N_d}, \tag{3.24d}$$

$$\tilde{\mathbf{A}}_{kj} = \boldsymbol{\Psi}^*_k\mathbf{W}\mathbf{A}_j\boldsymbol{\Psi}_k. \tag{3.24e}$$

Here, the superscript 'G' and subscript 'kj' are reminders that the quantity comes from taking inner products with the columns of $\mathbf{G}^*_{kj}$. With these quantities precomputed, the ROM components may be computed quickly online given a parameter vector $\boldsymbol{\mu}$ using the formulae

$$\mathbf{E}_k = \boldsymbol{\Psi}^*_k\mathbf{W}\hat{\mathbf{Q}}_k\left(\mathbf{G}^*_k\mathbf{W}\mathbf{G}_k\right)^{-1}\sum_{j=1}^{M_\mu}\zeta_j(\boldsymbol{\mu})\mathbf{B}^G_{kj}, \tag{3.25a}$$

$$\mathbf{J}_k = \boldsymbol{\Phi}^*\mathbf{W}\hat{\mathbf{Q}}_k\left(\mathbf{G}^*_k\mathbf{W}\mathbf{G}_k\right)^{-1}\sum_{j=1}^{M_\mu}\zeta_j(\boldsymbol{\mu})\mathbf{B}^G_{kj}, \tag{3.25b}$$

$$\mathbf{N}_k = \boldsymbol{\Psi}^*_k\mathbf{W}\hat{\mathbf{Q}}_k\left(\mathbf{G}^*_k\mathbf{W}\mathbf{G}_k\right)^{-1}\left(\sum_{j=1}^{M_\mu}\zeta_j(\boldsymbol{\mu})\mathbf{U}^G_{kj}\right)\left(\mathbf{P}^{nT}\mathbf{U}^n\right)^{-1}, \tag{3.25c}$$

$$\mathbf{M}_k = \boldsymbol{\Phi}^*\mathbf{W}\hat{\mathbf{Q}}_k\left(\mathbf{G}^*_k\mathbf{W}\mathbf{G}_k\right)^{-1}\left(\sum_{j=1}^{M_\mu}\zeta_j(\boldsymbol{\mu})\mathbf{U}^G_{kj}\right)\left(\mathbf{P}^{nT}\mathbf{U}^n\right)^{-1}, \tag{3.25d}$$

$$\boldsymbol{n}_{klmn} = \boldsymbol{\Psi}_k^* \mathbf{W} \hat{\mathbf{Q}}_k \left(\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k \right)^{-1} \sum_{j=1}^{M_\mu} \zeta_j(\boldsymbol{\mu}) \boldsymbol{b}_{kjlmin}^G, \tag{3.25e}$$

$$\boldsymbol{m}_{klmn} = \boldsymbol{\Phi}^* \mathbf{W} \hat{\mathbf{Q}}_k \left(\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k \right)^{-1} \sum_{j=1}^{M_\mu} \zeta_j(\boldsymbol{\mu}) \boldsymbol{b}_{kjlmin}^G, \tag{3.25f}$$

$$\tilde{\mathbf{A}}_k = \sum_{j=1}^{M_\mu} \zeta_j(\boldsymbol{\mu}) \tilde{\mathbf{A}}_{kj}, \tag{3.25g}$$

where $(\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \in \mathbb{C}^{N_d \times N_d}$ is computed quickly using

$$\left(\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k \right)^{-1} = \left(\sum_{j=1}^{M_\mu} \sum_{l=1}^{M_\mu} \zeta_j(\boldsymbol{\mu}) \zeta_l(\boldsymbol{\mu}) \mathbf{G}_{kjl}^G \right)^{-1}. \tag{3.26}$$

Note that though $\mathbf{H}_k$ depends on $\boldsymbol{\mu}$, this dependence is only through $\tilde{\mathbf{A}}_k$, so after $\tilde{\mathbf{A}}_k$ is computed with a given parameter, $\mathbf{H}_k$ can be computed quickly using (2.28). With the formulae given in (3.25), all ROM components may be computed given a parameter $\boldsymbol{\mu}$ without performing operations that scale with N_x, as desired.

3.4 Solving the nonlinear system

Equations (3.13) and (3.20), pertaining to the DEIM-based and triadic interaction-based approximations of the nonlinearity, respectively, are both systems of rN_ω nonlinear equations in rN_ω unknowns. The online phase of the proposed reduced-order model consists of solving these equations for the coefficients $\tilde{\boldsymbol{a}}_{\mathcal{K}}$. For the method to be viable, these equations must be solved as quickly as possible, and to this end, we propose a fixed-point iteration technique. In the majority of our tests, the fixed-point iteration converged in a few (~ 10) iterations. We also describe a slower but more stable pseudo-time-stepping method that we used in the few cases where the fixed-point iteration did not converge. We analyze the convergence of both in Appendix B.

Both systems (3.13) and (3.20) may be written abstractly as

$$\tilde{\boldsymbol{a}}_{\mathcal{K}} = \boldsymbol{c}_{\mathcal{K}} + \boldsymbol{w}_{\mathcal{K}}(\tilde{\boldsymbol{a}}_{\mathcal{K}}), \tag{3.27}$$

where $\boldsymbol{c}_{\mathcal{K}} \in \mathbb{C}^{rN_\omega}$ is the grouping of terms in the system (either (3.13) or (3.20)) that do not depend on $\tilde{\boldsymbol{a}}_{\mathcal{K}}$, and where $\boldsymbol{w}_{\mathcal{K}} : \mathbb{C}^{rN_\omega} \to \mathbb{C}^{rN_\omega}$ is the grouping of terms that do depend on $\tilde{\boldsymbol{a}}_{\mathcal{K}}$. The fixed-point iteration is simply

$$\begin{aligned} \boldsymbol{a}_{\mathcal{K}}^0 &= \mathbf{0}, \\ \boldsymbol{a}_{\mathcal{K}}^{i+1} &= \boldsymbol{c}_{\mathcal{K}} + \boldsymbol{w}_{\mathcal{K}}(\boldsymbol{a}_{\mathcal{K}}^i). \end{aligned} \tag{3.28}$$

With the initial guess of $\mathbf{0}$, the first iterate is $\boldsymbol{a}_{\mathcal{K}}^1 = \boldsymbol{c}_{\mathcal{K}}$, the solution to the system without the presence of the nonlinearity. The solution $\tilde{\boldsymbol{a}}_{\mathcal{K}}$ to (3.27) is a fixed point of the iteration above.

A necessary condition for convergence to this fixed point is that the eigenvalues of the Jacobian about it must be within the unit circle. In our numerical examples, we have found that this condition is usually met, and the iteration converges. The exceptions have been in cases where the nonlinearity is strong, and the solution to the system with no nonlinearity bears little resemblance to that with the nonlinearity. In Appendix B, we present some analysis of the fixed point iteration that is consistent with this observation. We also note that Anderson acceleration [1, 42] may be used to accelerate the convergence and possibly make it more robust.

In cases where the fixed point iteration does not converge, a slower but more stable alternative is the following pseudo-time-stepping method used by Ref. [15] to solve similar problems,

$$\begin{aligned} &\boldsymbol{a}_{\mathcal{K}}(0) = \boldsymbol{c}_{\mathcal{K}}, \\ &\frac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{a}_{\mathcal{K}}(\tau) = \boldsymbol{c}_{\mathcal{K}} + \boldsymbol{w}_{\mathcal{K}}(\boldsymbol{a}_{\mathcal{K}}(\tau)) - \boldsymbol{a}_{\mathcal{K}}(\tau). \end{aligned} \tag{3.29}$$

The stability of this method also depends on the eigenvalues of the same Jacobian. In this case, however, the stability condition is that the real part of the eigenvalues must be less than 1 (with an infinitesimal time step), so the pseudo-time-stepping method is more stable than the fixed point iteration. In our numerical examples, this method never failed to converge. Note that the pseudo-time-stepping method is equivalent to the fixed point iteration if an explicit Euler integrator is used with a time step of 1.

3.5 Scaling analysis

Algorithms for the offline and online phases of the method are given in Appendix D. Here we give the scalings for both, depending on the handling of the nonlinearity. With either DEIM or the sparse triadic interactions, the dominant online cost comes from calculating the nonlinear term $\boldsymbol{w}_{\mathcal{K}}$ at each iteration. In totaling the online scaling of the method in this subsection and timing the method in Section 4, we count all operations beginning from the initial condition and forcing and ending at the SPOD coefficients.

The online phase of the method consists of repeating the iteration (3.28) until the SPOD coefficients are converged. The constant $\boldsymbol{c}_{\mathcal{K}}$ is computed once at the beginning; computing this constant is equivalent to solving for the SPOD coefficients with no nonlinearity present. The nonlinear term is computed for each iteration, and, in practice, it is the dominant cost of the method.

CPU cost due to the constant term scales as

$$\mathcal{O}(N_\omega(N_f \log N_\omega + p_1 N_f + rN_f + rp_1) + p_1 N_x). \tag{3.30}$$

The first term is due to the FFT of the forcing; the second to calculating the effect of the forcing in the intermediary basis; the third to calculating the effect of the forcing directly on each SPOD coefficient; and the fourth to calculating the effect of each coefficient in the intermediary basis on each SPOD coefficient. The last term, which does not scale with the number of frequencies, comes from calculating the initial condition in the intermediary basis.

The cost of the nonlinear term at each iteration, i.e., each evaluation of $\boldsymbol{w}_{\mathcal{K}}(\tilde{\boldsymbol{a}}_{\mathcal{K}})$, depends on the handling of the nonlinearity. If DEIM is used, each iteration scales as

$$\mathcal{O}\left(N_\omega(rp_2 + p_2 \log N_\omega + p_1 p_2 + rp_1)\right). \tag{3.31}$$

The first term is due to sampling the trajectory at the p_2 sample points at every frequency; the second to taking the IFFT of this sampling, computing the nonlinearty, then taking the FFT; the third to computing the effect of the nonlinearity at p_2 sample points in the intermediary basis; and the fourth to calculating the effect of the each coefficient in the intermediary basis on each SPOD coefficient.

If, instead, sparse triadic interactions are used, the scaling of each iteration is

$$\mathcal{O}\left(N_\omega p_1 r + \sum_{k\in\mathcal{K}} |\mathcal{N}_k|(r_k + p_1)\right). \tag{3.32}$$

The first term accounts for the effect of each coefficient of the intermediary basis on each SPOD coefficient. The sum accounts for the number of nonlinear terms that are retained at each frequency.

The total online cost, therefore, scales as

$$\mathcal{O}\left(N_\omega\left(N_f \log N_\omega + p_1 N_f + r N_f + N_{iter}\left(r p_2 + p_2 \log N_\omega + p_1 p_2 + r p_1\right)\right) + p_1 N_x\right) \tag{3.33}$$

if DEIM is used and

$$\mathcal{O}\left(N_\omega\left(N_f \log N_\omega + p_1 N_f + r N_f\right) + N_{iter}\left(r p_1 N_\omega + \sum_{k\in\mathcal{K}} |\mathcal{N}_k|(r_k + p_1)\right) + p_1 N_x\right) \tag{3.34}$$

if the sparse triadic interactions are used.

In practice, in the DEIM case, r, p_1, and p_2 will likely be of similar magnitudes, and the dominant cost will come from the iteration to solve the system. The scaling will then be

$$\mathcal{O}\left(N_\omega N_{iter} r^2\right). \tag{3.35}$$

If the sparse triadic interactions are used, the dominant cost will likely come from evaluating the nonlinear term at every iteration. A rough estimate of the scaling in this case is

$$\mathcal{O}(N_{iter} N_{int} r), \tag{3.36}$$

where $N_{int} = \sum_{k\in\mathcal{K}} |\mathcal{N}_k|$. In practice, we have observed the fixed-point iteration to converge in ~ 10 iterations or fewer. If the pseudo-time-stepping method is used, N_{iter} is the number of time steps needed for convergence, and in the cases where we used this method, N_{iter} was a few hundred using an adaptive ttime-steppingmethod (though we made no attempt to optimize this).

Assuming the governing equations are sparse, i.e., computing the action $\mathbf{A}$ and $\boldsymbol{n}$ both scale as $\mathcal{O}(N_x)$, the offline scalings are as follows. Obtaining the SPOD modes and computing the operators necessary for the constant term scale as

$$\mathcal{O}(N_\omega N_d^2 N_x). \tag{3.37}$$

Constructing the operators needed for the nonlinearity in the DEIM case scales as

$$\mathcal{O}((r + p_1 + p_2) N_d N_x N_\omega). \tag{3.38}$$

In the case of sparse triadic interactions, constructing the necessary vectors scales as

$$\mathcal{O}\left(\sum_{k\in\mathcal{K}} \sum_{l\in\mathcal{K}} r_l r_i r_k N_x\right). \tag{3.39}$$

So long as the offline cost is feasible, the online cost is the salient quantity. In practice, computing the effect of the nonlinearity for each iteration ((3.31) or (3.32)) dominates the online cost. In our numerical examples, we find that DEIM is faster (at the same accuracy) than the sparsified triadic interactions, though we do not expect this to hold in general.

4 Results

We use the 1-dimensional Ginzburg-Landau equations to test the method. We use the standard form of these equations for testing the DEIM-based handling of nonlinearity and a modified Ginzburg-Landau system with a quadratic nonlinearity to test the triadic-interaction-based treatment of the nonlinearity. For both systems, we compare the relative error and online CPU time of SSOP to those of POD-G. The results are encouraging: for the same number of modes, SSOP gives two orders of magnitude lower error than does POD-G. For the DEIM-based treatment of the nonlinearity, this accuracy improvement comes at slightly lower CPU time than POD-G, whereas the triadic-interaction-based ROM is slower than POD-G. We also test the accuracy of the method on out-of-sample data and find that the method trained on one GL system generalizes well to a new GL system.

4.1 Standard Ginzburg-Landau system

The complex Ginzburg-Landau equation is a common test case for model reduction methods [17, 4, 8, 30]. It is

$$\dot{q}(x,t) = \left[-\nu\frac{\partial}{\partial x} + \gamma\frac{\partial^2}{\partial x^2} + \mu_0 - c_\mu^2 + \frac{\mu_2}{2}x^2 \right] q(x,t) - \alpha q(x,t)|q(x,t)|^2 + f(x,t), \tag{4.1}$$

where $f(x,t)$ is a forcing and where we set $\nu = 2 + 0.4i$, $\gamma = 1 - i$, $c_\mu = 0.2$, $\mu_2 = -0.01$, and $\alpha = 1$, which are standard values [3, 40]. In most of our tests, we set μ_0, a bifurcation parameter in the system, to $\mu_0 = 0.229$ [3, 40]. The terms in the system generate advection, diffusion, and local growth or decay depending on the sign of $\mu_0 - c_\mu^2 + \frac{\mu_2}{2}x^2$. For $\mu_0 = 0.229$, $\mu_0 - c_\mu^2 + \frac{\mu_2}{2}x^2 > 0$ for $x \in [-6.15, 6.15]$, so solutions grow in this region. For this value of μ_0, the system is linearly stable, i.e., the eigenvalues of the linear operator about the steady solution $q(x) = 0$ are all in the stable half-plane. There is a bifurcation at $\mu_0 = 0.397$; above this value, the system is linearly unstable, and the dynamics are characterized by a competition between the amplifying effect of the linear terms and the damping effect of the nonlinear term. While most of our tests use $\mu_0 = 0.229$, we also test the range $\mu_0 \in [0.079, 0.499]$, which envelopes the bifurcation. We refer to (4.1) as the 'standard' Ginzburg-Landau system to distinguish it from the modified system introduced in the next subsection.

The full-order model consists of a pseudo-spectral Hermite discretization [3, 7] of (4.1) with $N_x = 220$ collocation points [40] that we integrate with `ode45` in MATLAB. To generate data, we integrate for 3000 time steps with $\Delta t = 0.8$. We use the method described in Ref. [40] to obtain the modes from the single long trajectory by forming 44 trajectories of length $N_\omega = 256$, each overlapping 75% with the next. The forcing generating this data is spread over the range $x \in [-12, -8]$ and is strongest at $\overline{x} = 10$. It is stochastic with a spatial correlation length of 1,

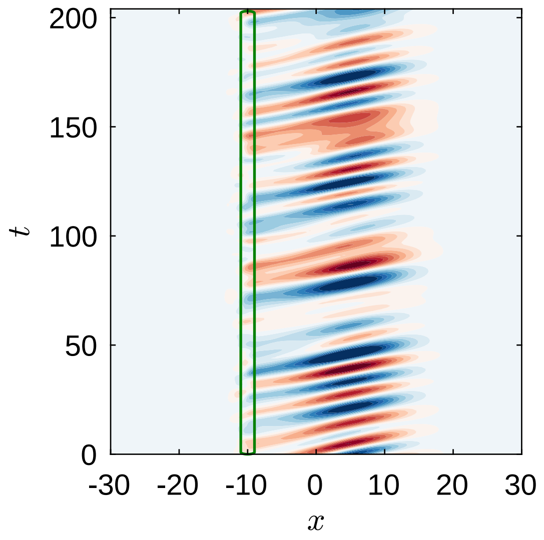

Figure 3: A trajectory of the (standard) Ginzburg-Landau system with $\mu_0 = 0.229$. The green lines demarcate the location of the forcing. The spatiotemporal structure enables space-time modes, like SPOD modes, to represent the trajectory far more efficiently than space-only modes.

a temporal correlation length of 3.33, and a Gaussian support. The spatiotemporal correlation is given by

$$\mathbb{E}[f(x_1,t_1)f^*(x_2,t_2)] \propto \exp\left[-\left((x_1-\overline{x})^2+(x_2-\overline{x})^2+(x_2-x_1)^2+(0.3(t_2-t_1))^2\right)\right], \tag{4.2}$$

and we take it to be zero outside $x \in [-12, -8]$.

Figure 3 shows a space-time diagram of a trajectory of the Ginzburg-Landau system with $\mu_0 = 0.229$. The advective behavior of the system is evident in the diagonally oriented streaks, with the upward inclination indicating advection in the positive direction. There are spatial correlations here, but, crucially, there are strong spatiotemporal correlations as well. These spatiotemporal correlations — the fact that there is structure in this space-time diagram beyond the spatial structure — are what allow space-time modes to encode the trajectory substantially more efficiently than POD modes.

Figure 4 shows the energy of the modes. Specifically, Figure 4 (a) shows the fraction of energy that is excluded as a function of r, which is computed by summing the energies of the modes that are not among the rN_ω most energetic, normalized by the sum of all of the energies. This is equivalent to the average relative SPOD reconstruction error over the training data. This gives a lower bound of the relative error of the method applied to the training data — the error of the SSOP ROM cannot be lower than the SPOD reconstruction error. Figure 4 (b) shows the SPOD energies of the modes as a function of frequency for $r = 5$. The top curve is $\lambda_{\mathcal{K},1}$, the first SPOD mode as a function of frequency, and the lower curves are higher mode numbers. The red and blue curves represent retained and truncated modes, respectively. The threshold is determined by r as $\tilde{\lambda}_{(rN_\omega)}$ and is shown in orange. The number of modes at each frequency that meet the threshold and are thus retained is shown in green on the right axis. For the least energetic frequencies, as few as 3 modes are retained, whereas for the most energetic frequencies, as many as 8 are retained. The average value is $r = 5$, by definition.

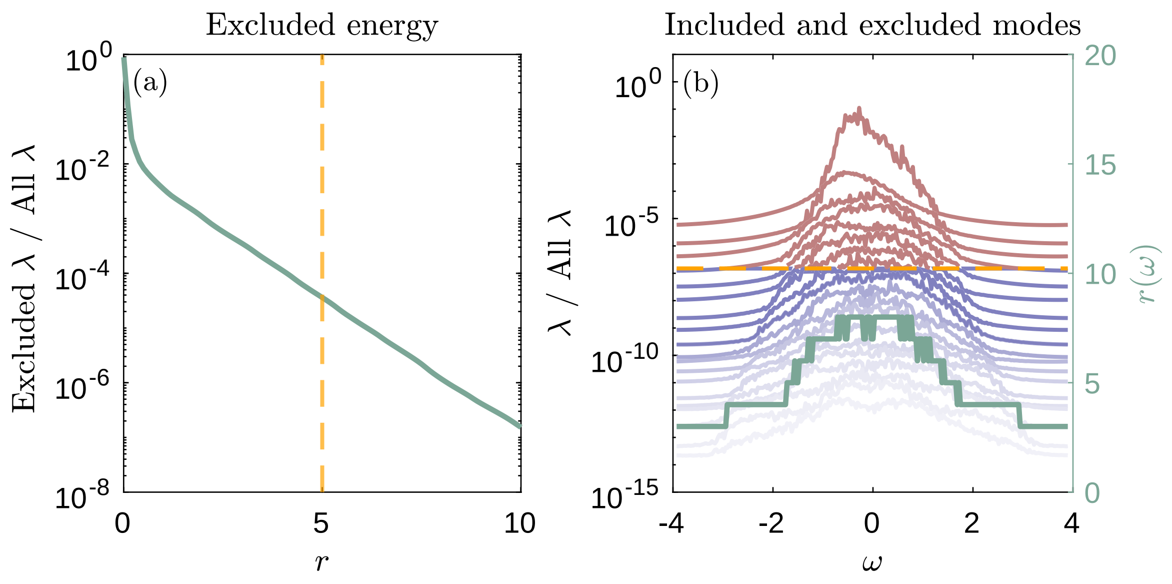

Figure 4: SPOD energies. (a) The energy not captured by the first rN_ω modes, calculated as the ratio of the sum of the energies of the excluded modes to the sum of the energies of all the modes. The dashed line corresponds to the r value used to determine the cutoff in (b). (b) The energy of the included (red) and excluded (blue) modes as a function of ω for $r = 5$. Inclusion of a given mode $\boldsymbol{\psi}_{km}$ is determined by whether the associated energy is among the rN_ω largest energies over all frequencies i.e., whether $\lambda_{km} \geq \tilde{\lambda}_{(rN_\omega)}$ (see (2.13)). The cutoff energy $\tilde{\lambda}_{(rN_\omega)}$ is shown in orange. The number of modes at each frequency $r(\omega)$ is shown in green on the right axis; the mean is $r = 5$.

Throughout the remainder of this section, the (relative) errors are defined as follows. The error at time t_j is the square norm of the difference between the ROM and FOM solutions normalized by the mean square norm of the FOM solution averaged over time and over test trajectories. This may be written as

$$e_j = \frac{\|\tilde{\boldsymbol{q}}_j - \boldsymbol{q}_j\|_x^2}{\frac{1}{N_\omega N_{test}} \sum_{i=1}^{N_{test}} \|\boldsymbol{q}_{\mathcal{J}}^i\|_{x,t}^2}, \tag{4.3}$$

where $\boldsymbol{q}$ is the FOM solution, $\tilde{\boldsymbol{q}}$ is the ROM approximation thereof, and $\boldsymbol{q}_{\mathcal{J}}^i$ is the i-th trajectory in the test data. This error averaged over time is

$$e = \frac{\|\tilde{\boldsymbol{q}}_{\mathcal{J}} - \boldsymbol{q}_{\mathcal{J}}\|_{x,t}^2}{\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \|\boldsymbol{q}_{\mathcal{J}}^i\|_{x,t}^2}. \tag{4.4}$$

For the most part, we report these quantities averaged over the test trajectories below.

We first test the method on 30 new trajectories, i.e., trajectories not included in the training data. The forcing is stochastic in each case and is drawn from the same distribution as the stochastic forcing used to generate the training data (but the realizations are different). The 30 initial conditions in the test data are taken from a single long run of the system and are also not in the data used to compute the SPOD modes, but come from the same statistical distribution as the states in the training data. These conditions together mean that the SPOD modes from the training data are good at representing the trajectories in the test data.

In Figure 5, we show the relative error of the proposed method with $r = 5$ modes along with that of POD-G, also with 5 modes. The dashed curves are the projection errors of the respective bases. In other words, they represent the discrepancy between the full-order solution and the projection thereof onto the POD and SPOD bases. The projection error is a lower bound for the model error; if the model recovers the exact coefficients, then it achieves the projection error. All errors are calculated as the mean over the 30 realizations on which we test the method, and the shaded regions represent data within one standard deviation of the mean. Most notably, SSOP outperforms POD-G by nearly three orders of magnitude for most of the interval. It also significantly outperforms the projection of the FOM solution onto the POD modes (dashed brown). Since POD modes are the optimal linear encoder, the POD projection error is a lower bound for any method based on a space-only linear encoding, such as a space-only Petrov-Galerkin method or operator inference. That SSOP and the SPOD projection (solid and dashed green, respectively) nearly overlap indicates that SSOP solves nearly exactly for the SPOD coefficients. A notable feature of Figure 5 is that the error is largest at the ends of the interval, and it bears mentioning that this feature is not due to Gibbs phenomena — indeed, it is too small. Instead, it is due to a cyclic property of the SPOD encoder/decoder pair, and as more modes are included, the effect diminishes.

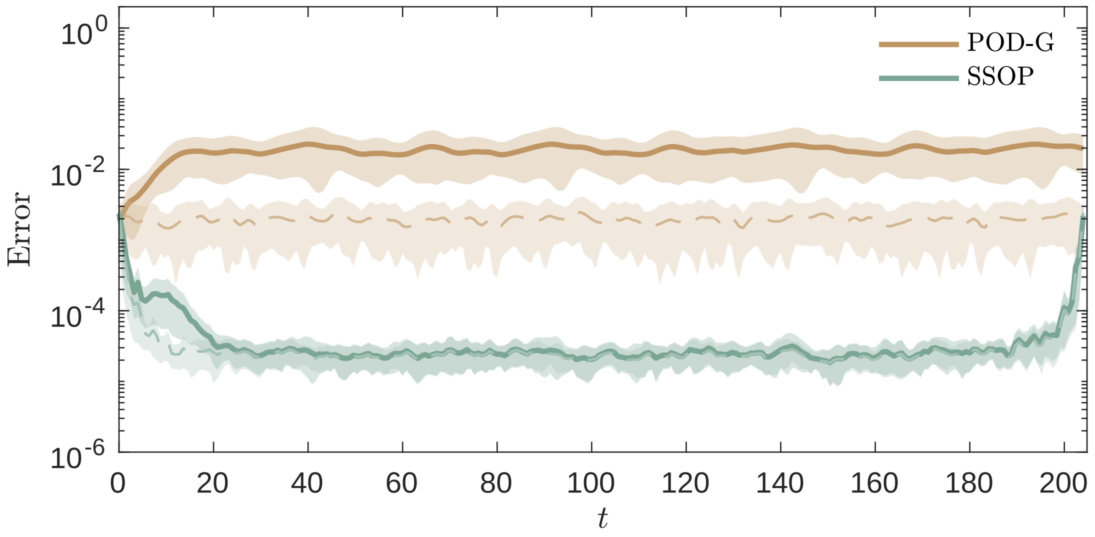

Figure 5: Error as a function of time for the proposed method and for POD-G, both with $r = 5$. The curves represent the mean error over the 30 test trajectories, and the shaded regions indicate the standard deviation of these errors. The dashed curves show the respective projection errors, which serve as lower bounds for the two methods.

Figure 6 shows the relative error, averaged over time, as a function of the number of modes retained. For all the values shown, SSOP again gives multiple orders of magnitude lower error than POD-G. The dashed curves are again the projection error. Another notable feature of the method is that it nearly achieves the SPOD mode projection error for relatively few modes. The error asymptote is likely due to the fact that the forcing and nonlinearity are not exactly represented by a finite Fourier series, as was assumed in deriving the method.

Figure 7 shows the CPU time of the proposed method in comparison to that of POD-G. The substantial error reduction shown does not come at an increase in computational cost; indeed, the

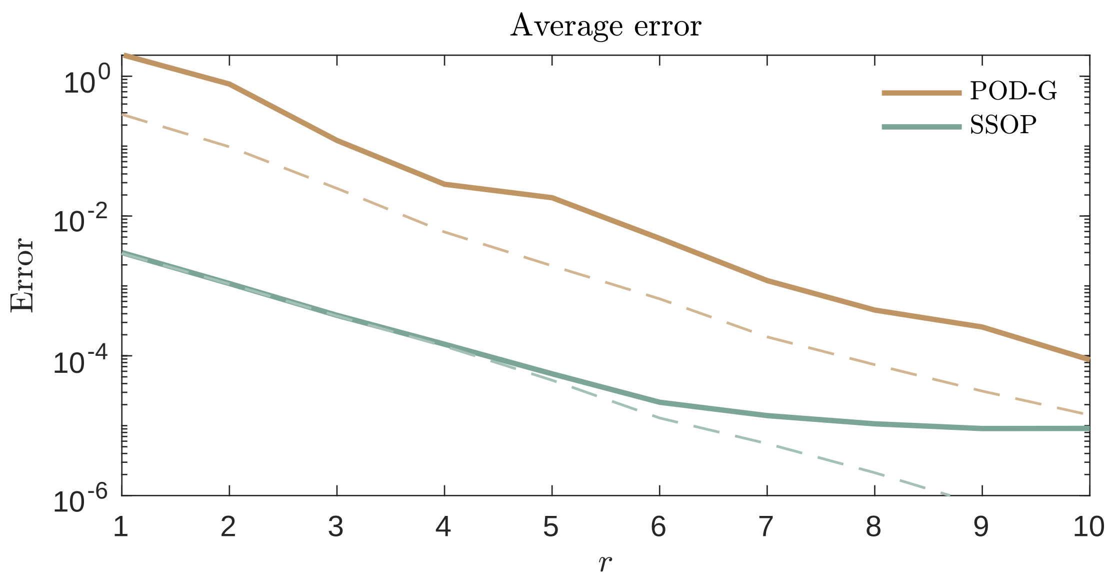

Figure 6: Error, averaged over time, for SSOP and for POD-G as a function of r, the number of modes used in the ROMs. The dashed curves again represent the projection error onto the POD and SPOD modes.

method is faster for most of the range considered. In what follows, we use $r = 5$ for all tests.

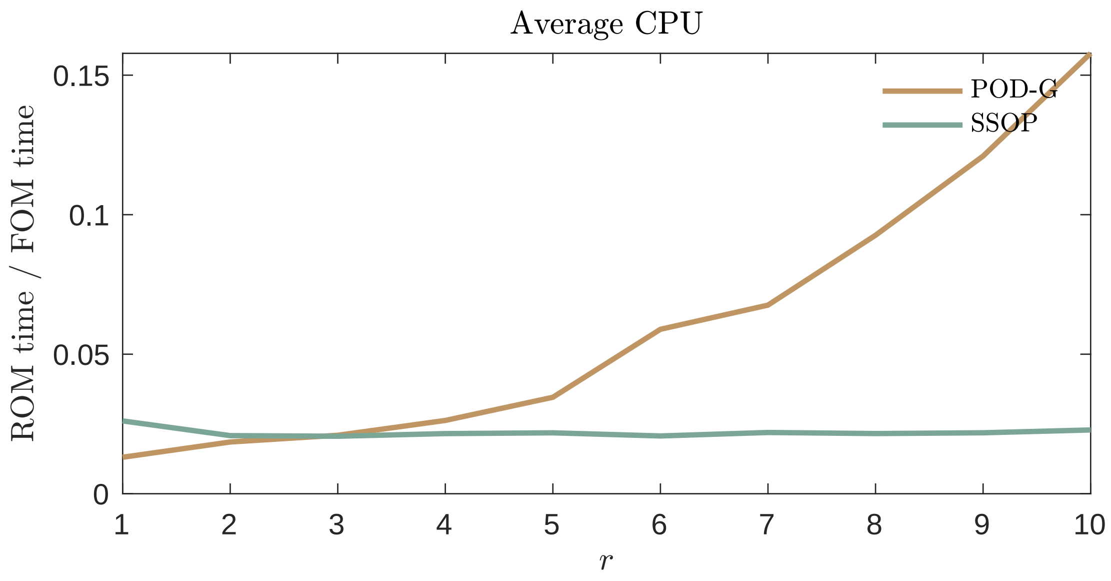

Figure 7: CPU time for SSOP in comparison to that of POD-G as a function of the number of modes used in the ROMs.

Next, we investigate how the method performs for non-stationary forcings. We again use the SPOD modes from data generated by a stochastic, statistically stationary forcing, but test the resulting model on a variety of non-stationary forcings. The forcing is active in the same spatial region as before, and is given in this region by

$$f(x,t) = f(t)\sqrt{\exp[-(x-\overline{x})^2]}, \tag{4.5}$$

The forcing is again strongest at $\overline{x} = 10$. We test four choices of the function $f(t)$: a periodic function, a pulse function, a quasiperiodic function, and a series of pulses, steps, and quasiperiodic functions. The initial condition is zero in each case.

These functions, the relative error of SSOP and POG-G, and the respective projection errors are shown in Figure 8. The salient takeaways are the same as in the stochastic forcing case: SSOP achieves substantially lower error than POD-G and than the POD projection error, and nearly achieves the SPOD projection error. With the periodic (a) and quasiperiodic (c) forcings, the solution at the beginning of the interval is different than the solution at the end of the interval, so the SPOD projection error, and thus SSOP, is high at the beginning and end of the interval. Conversely, because the pulse (b) and series (d) forcing are both zero for long enough at the end of the interval for the solution to decay, the error of the SPOD projection and SSOP remains low at the beginning and end of the interval. Note that because the error is normalized by the average square norm of the solution over the interval, the pulse error is substantially higher than the others. We also note that none of the forcings here (or in the stochastic forcing case) meet assumption *(ii)*, but this has little effect on the accuracy of the method.

Finally, we test the method on a range of Ginzburg-Landau systems by varying μ_0 over the range $\mu_0 \in [0.079, 0.499]$ in increments of 0.03. The behavior of the Ginzburg-Landau system changes significantly over this range. At the lower end, the system is linearly stable, i.e., all of the eigenvalues of $\mathbf{A}$ are in the stable half plane, and the behavior is entirely modal, i.e., the eigenvectors are nearly orthogonal. At $\mu_0 = 0.229$, the value we have used in the tests so far, the system is slightly non-modal — one measure of this is the optimal transient growth [41, 35], which is nearly 5. At $\mu_0 = 0.379$, the system is linearly stable but is strongly non-modal, with an optimal transient growth of nearly 200. At $\mu_0 > 0.397$, the system is linearly unstable, and the dynamics feature a competition between the growth caused by the linear instability and the damping of the nonlinear term.

We perform two tests over this range of μ_0. First, we build an SSOP model with SPOD modes from training data at each μ_0 with the stochastic forcing (4.2), then test the model at the same μ_0 with different realizations from the same distribution of forcings. This test verifies that the method can perform well across a range of Ginzburg-Landau systems. Second, we take SPOD modes from $\mu_0^* = 0.229$, then use them to build an SSOP model for each μ_0. This second test is designed to see if the method can be used to take data from one system and use it to predict behavior of a different one.

Figure 9(a) shows the results of the first test using $r = 5$ modes — the average relative error over both time and the 30 trajectories is shown as a function of μ_0. Again, SSOP substantially outperforms POD-G and the projection error of the POD basis, and nearly achieves its own projection error. The SSOP error is remarkably uniform over the range of parameters, in contrast to the POD-G error. The SSOP performance past the bifurcation at $\mu_0 = 0.397$ shows the efficacy of the strategy described in Section 3.1 of diverting some of the linear term to the nonlinear term.

Figure 9(b) shows the second test with $r = 5$ modes. Again, the relative error averaged over time and trajectories is shown, but, in contrast to (a), the model is built using modes educed using data from a system with $\mu_0^* = 0.229$. The projection errors (dashed) are also computed using the modes from μ_0^*. That the projection error is substantially lower near μ_0^* than at other values indicates that the SPOD modes change significantly as μ_0 changes. Despite this, the model error

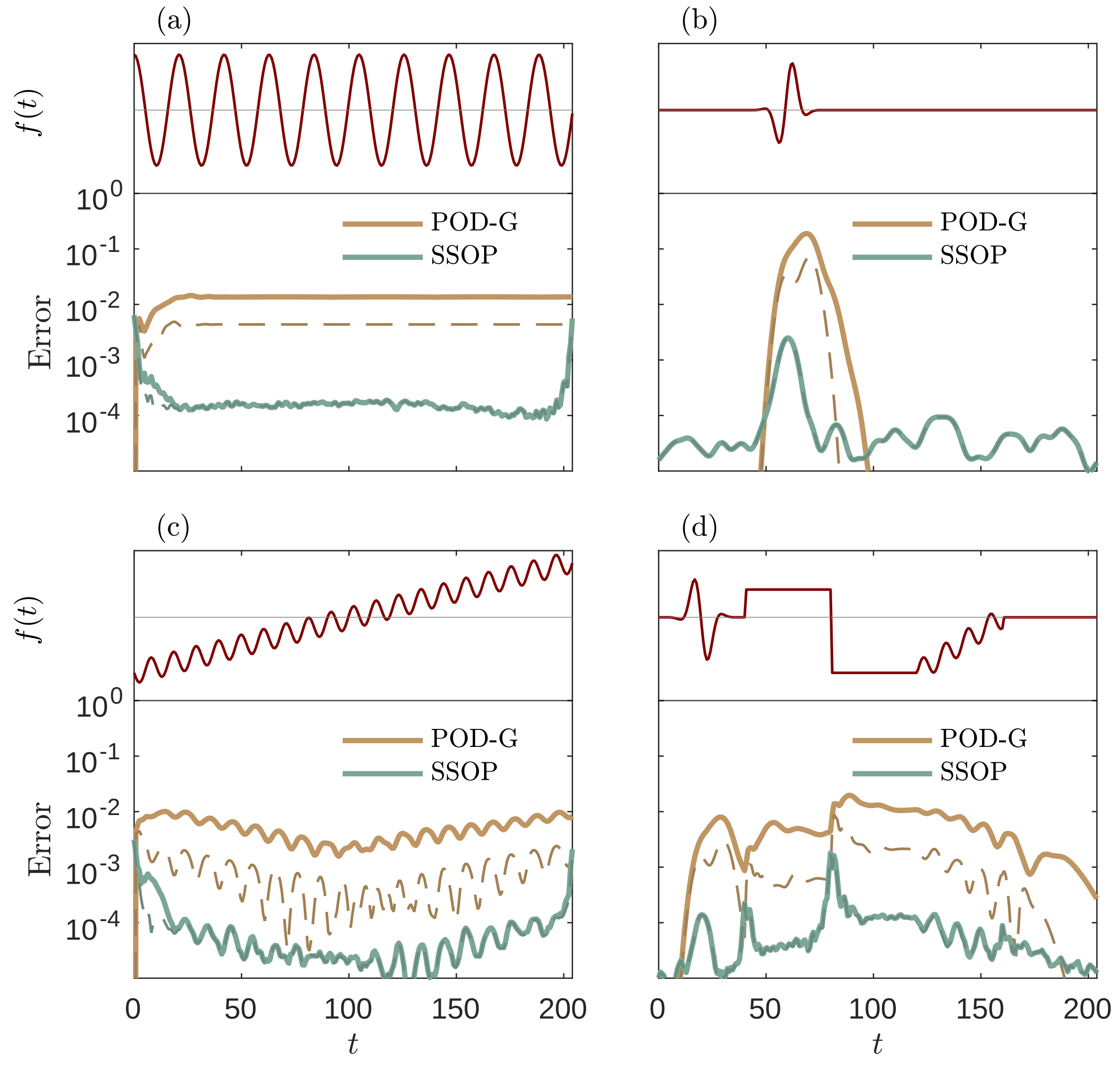

Figure 8: The error as a function of time for a variety of non-stochastic forcings for both SSOP and POD-G with $r = 5$ modes, along with the respective projection errors (dashed). The SSOP performance is robust for these out-of-sample forcings.

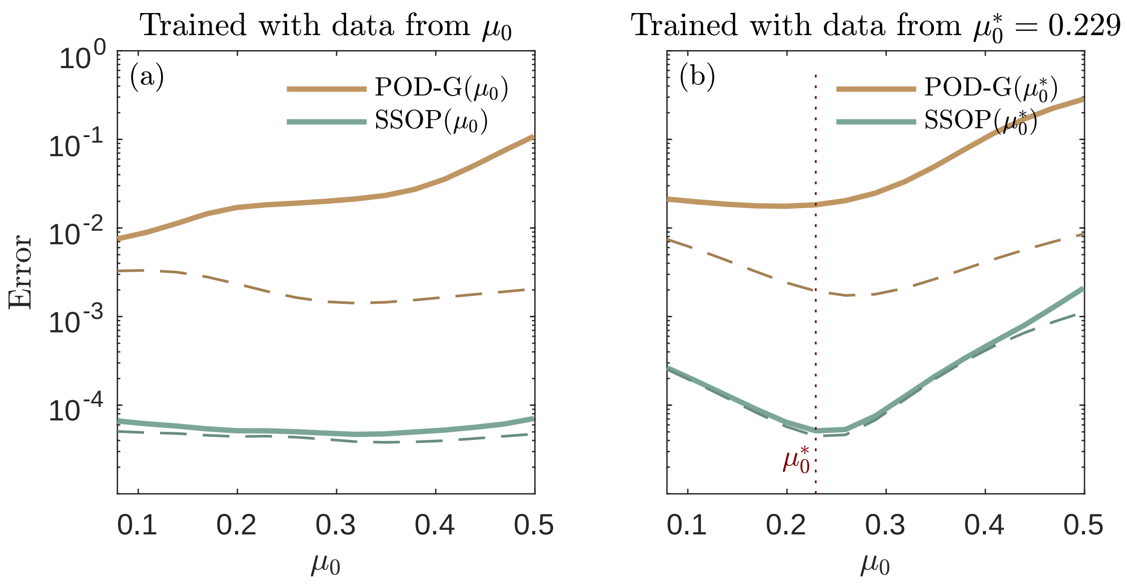

Figure 9: The methods tested on a range of Ginzburg-Landau systems parameterized by μ_0 with $r = 5$. In (a), the model is trained using data from the same GL system. In (b), the models are trained using data from $\mu_0^* = 0.229$. The dashed curves correspond to the projection error for the modes from the training system.

results (solid) show SSOP can generate accurate predictions for new systems. In fact, the SSOP model built using data from a different Ginzburg-Landau system ((b), solid green) produces lower error in this example than the projection error onto the POD modes of the test system ((a), dashed brown). We view this as a strong result showcasing the robustness of the proposed method.

4.2 Quadratic Ginzburg-Landau system

Here, we test the method described in Subsection 3.2.2 for handling quadratic nonlinearities. We do this by replacing the standard nonlinear term with a Burgers'-type (quadratic) nonlinearity and find that most of the triadic interactions have a negligible impact. We exclude roughly 99% of the interactions, which leads to substantial speedup with no meaningful increase in the error relative to retaining all of them. Even with excluding these interactions, however, the method is slower than using DEIM on the same problem. Our outlook is that using the triadic-interaction-sum method of handling nonlinearities will be more effective than DEIM for more complex problems based on the findings in Ref. [37].

The quadratically nonlinear Ginzburg-Landau equation is

$$\frac{\partial}{\partial t}q(x,t) = \left[-\nu\frac{\partial}{\partial x} + \gamma\frac{\partial^2}{\partial x^2} + \mu_0 - c_\mu^2 + \frac{\mu_2}{2}x^2 \right] q(x,t) + \kappa q(x,t)\frac{\partial}{\partial x}q(x,t). \tag{4.6}$$

We set $\kappa = 5$ and otherwise use the same parameters from before (including $\mu_0 = 0.229$).

We generate data following the same procedure as before — using the stochastic forcing (4.2) to generate 3000 time steps of training data with $\Delta t = 0.8$, forming this data into 44 overlapping blocks of length $N_\omega = 256$, and computing the SPOD modes and operators. We then again use

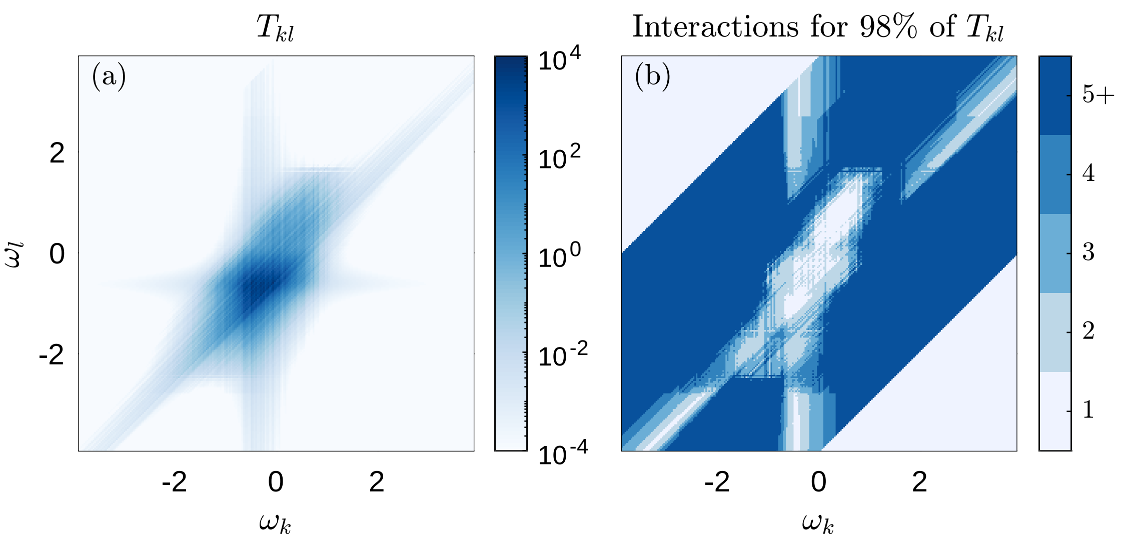

Figure 10: The impact of ω_l on ω_k as defined by (4.7) via triadic interaction with $\omega_i = \omega_k - \omega_l$. (a) shows T_{kl}, a proxy for the strength of all interactions between the SPOD modes at ω_l with those at ω_i. (b) shows how many of these interactions are needed to account for 98% of T_{kl}.

different realizations from the same forcing distribution to generate the data used for testing.

The criterion (described in Section 3.2.2) for including the triadic interaction between mode m at ω_l and mode n at frequency $\omega_i = \omega_k - \omega_l$ is $\|\boldsymbol{n}_{klmn}\|_x^2 \lambda_{lm}\lambda_{in} \geq \epsilon$ for a user-defined value of ϵ. We define the matrix $\mathbf{T} \in \mathbb{R}^{N_\omega \times N_\omega}$ as

$$T_{kl} = \sum_{m=1}^{r_l} \sum_{n=1}^{r_i} \|\boldsymbol{n}_{klmn}\|_x^2 \lambda_{lm}\lambda_{in}. \tag{4.7}$$

T_{kl} is a proxy for the total impact of ω_l on ω_k via triadic interactions. We plot T_{kl} for all pairs of frequencies in Figure 10(a). In Figure 10(b), we show the number of interactions at a given frequency pair ω_k, ω_l that is required to account for 98% of T_{kl}. We see that for many of the frequency pairs, the total impact at a frequency pair is dominated by just a few mode pairs within the two frequencies. Noting the overlap in the high-value region of (a) and the low-value region of (b), we see that this is particularly true at the more energetic mode pairs. The implication of (a) and (b) together is that the great majority of frequencies may be discarded without having a meaningful effect on the approximation of the nonlinearity.

Figure 11(a) shows the number of interactions at each frequency pair with $r = 5$. These values are only a function of $r_{\mathcal{K}}$ — the number at (ω_k, ω_l) is $r_l r_i$, where $\omega_i = \omega_k - \omega_l$. Figure 11(b) shows the number at each pair that are retained when the threshold in (3.18) is set to $\epsilon = 10^{-1.8}$ (we show this one because it leads to a good balance between accuracy and speed in the results). With this choice of ϵ, 1.7% of the interactions are retained.

We test the accuracy and speed of the method on the 30 trajectories as a function of the number of interactions included. In Figure 12(a), we plot the relative error averaged over time and trajectories against the percentage of the total (1384103) interactions. As one would expect, increasing

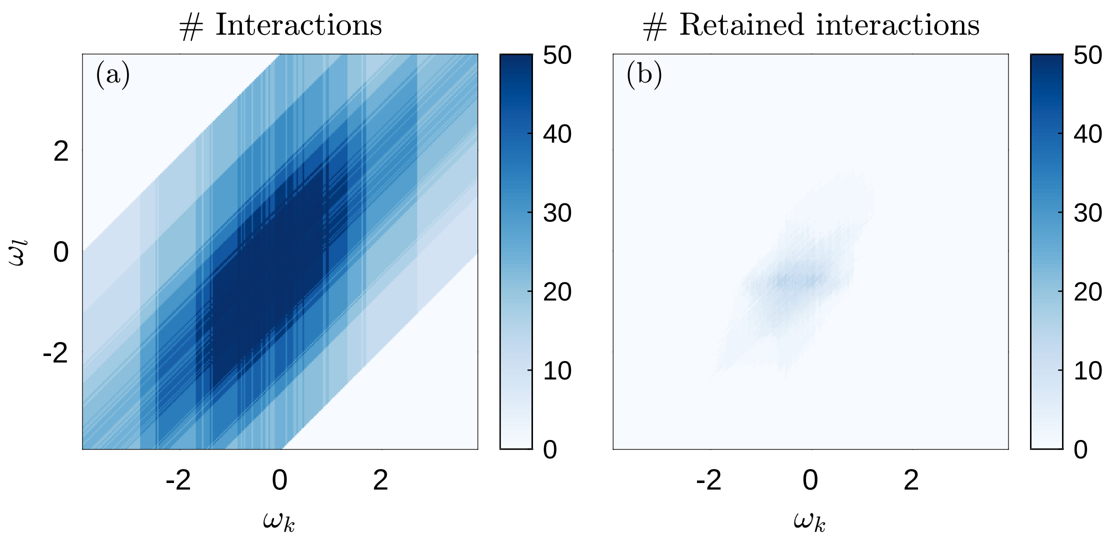

Figure 11: (a) The total number of interactions between SPOD modes at ω_l and $\omega_k - \omega_l$ with $r = 5$. (b) The number of these interactions that are retained when $\epsilon = 10^{-1.8}$. Only 1.7% of the interactions in (a) are retained with this threshold.

the number of retained interactions causes the error to decrease and the CPU time to increase. Figure 12(b) shows that the CPU increase is linear in the number of interactions retained, as is predicted by the scaling analysis in Section 3.5. The error quickly reaches a plateau as the newly retained interactions become less energetic (the plateau is above the SPOD projection error, though it is quite close).

The red ticks in Figure 12 indicate the error and timing results using the DEIM-based approximation of the nonlinearity. For this problem, DEIM is able to achieve lower error at lower CPU time than the triadic-interaction-based approximation, but we do not expect that this is true in general.

5 Conclusions

We have generalized spectral solution operator projection [12] to nonlinear systems. The approach represents the unknown trajectory by representing each temporal frequency with the basis of SPOD modes at that frequency. By projecting an implicit solution to the nonlinear system onto a set of retained SPOD modes, we arrive at a set of nonlinear algebraic equations for the SPOD coefficients. The online phase of the method comprises solving this set of equations given the initial condition, forcing, and system parameters, which we do using a fixed-point iteration.

The method performed well in a series of tests using a 1-dimensional PDE. The key result is that, for the same number of modes, the error is two orders of magnitude lower than that of POD-G and is substantially lower than the POD projection error, which is a lower bound for time-domain Petrov-Galerkin methods. Returning to the two questions outlined in the introduction, our method achieves an affirmative answer to both. Specifically, (1) it does recover the encoding of the trajectory accurately, i.e., the SSOP error is near the SPOD projection error; (2) it does so in a similar (and often shorter) time than POD-G. We found these results to be robust – we

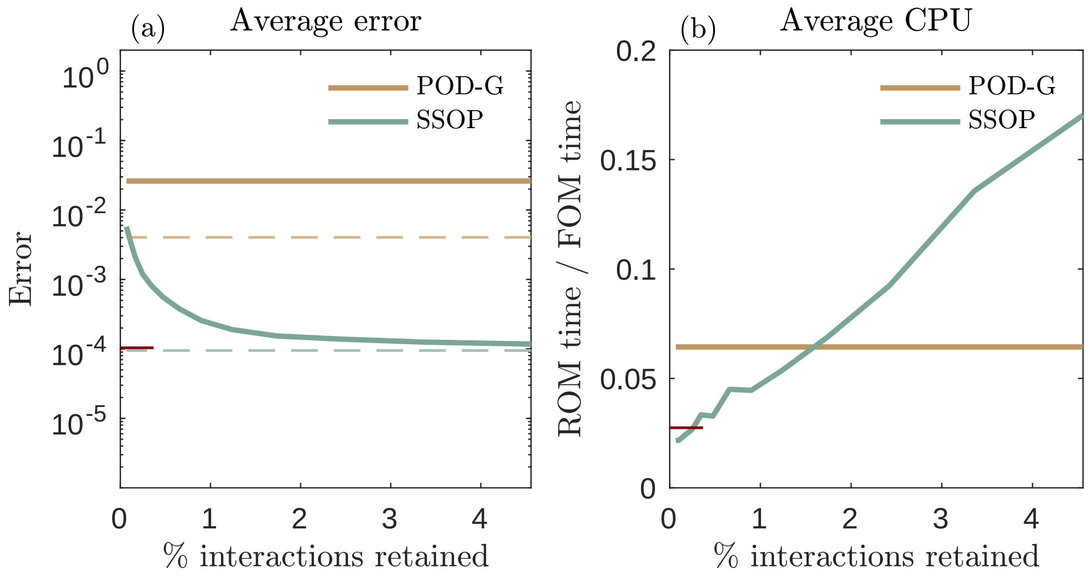

Figure 12: The relative error (a) and CPU time (b) for the method with $r = 5$ using the sparse-triadic-interaction approximation of the nonlinearity as a function of the number of triadic interactions retained. The x-axis shows the percentage of the total triadic interactions used. The red ticks indicate results using the DEIM approximation of the nonlinearity.

tested SSOP against POD-G on out-of-sample forcings, as well as parameter variations, and its error was consistently much lower than that of POD-G. The accuracy improvement does not come at an increased computational cost: the method is faster than POD-G for most numbers of modes.

Three downsides of the method bear mentioning. First, the initial step of the method is to take a Fourier transform of the forcing. This means that if the forcing is not known on the entire interval before starting the method, the method cannot be applied. Second, obtaining the SPOD modes requires more training data than obtaining, e.g., the POD modes that form the basis for many space-only methods. Finally, the method applies to a particular temporal window $[0, T]$, which is set by the block length used in calculating the SPOD modes. If predictions on a longer window are desired, the method may be repeated using the final (or near the final, as the error increases at the end of the interval) state as the new initial condition. This is somewhat cumbersome in comparison to space-only methods, which preserve the Markovian property of the governing equations. On balance, however, we are quite encouraged by the results and believe they give reason for further interest in this method and space-time ROMs more broadly.

Two important questions remain: (1) For which systems do the solution methods converge? In the appendix, we show that when the effect of nonlinearity is small compared to the linear solution, the fixed-point iteration and pseudo-time-stepping method converge, and that in the opposite limit, they do not for a broad class of nonlinearities. We are optimistic about the convergence for systems in which linear methods can be used to predict nonlinear behavior, as is the case in many fluid dynamic problems. (2) How accurate will the operator approximations be for larger systems, and to what extent will a lack of accuracy here impact the error in the solution? In the linear case, we found that we were able to achieve good accuracy in a 2-dimensional advection-diffusion problem. This problem does not have strong modal behavior, which likely makes these approximations less accurate, so we expect they will hold sufficiently in larger problems, especially ones with modal behavior.

6 Acknowledgments

We gratefully acknowledge funding from the National Science Foundation grant No. 2237537.

Appendix A Derivation of full-order equations in the frequency domain

Here, we derive (3.5) from

$$\hat{\boldsymbol{q}}_k = \sum_{j=0}^{N_\omega-1} \left(e^{\mathbf{A} j\Delta t}\boldsymbol{q}_0 + \int_0^{j\Delta t} e^{\mathbf{A}(j\Delta t - t')}\mathbf{B}\boldsymbol{f}(t') + \boldsymbol{n}(\boldsymbol{q}(t'))\ dt' \right) e^{-i\omega_k j\Delta t}. \tag{A.1}$$

We assume *(i)* the forcing is of the form $\boldsymbol{f}(t) = \sum_{l=0}^{N_\omega-1} \hat{\boldsymbol{f}}_l e^{i\omega_l t}$, *(ii)* that $i\omega_l\mathbf{I} - \mathbf{A}$ is invertible for all included l, and *(iii)* that the nonlinearity is of the form $\boldsymbol{n}(\boldsymbol{q}(t)) = \sum_{l=0}^{N_\omega} \hat{\boldsymbol{n}}_l e^{i\omega_l t}$. We refer to the term involving the initial condition as $\hat{\boldsymbol{q}}_{k,ic}$ and the term involving the integral as $\hat{\boldsymbol{q}}_{k,int}$. We start by evaluating

$$\hat{\boldsymbol{q}}_{k,ic} = \sum_{j=0}^{N_\omega-1} e^{(\mathbf{A}-i\omega_k\mathbf{I})j\Delta t}\boldsymbol{q}_0. \tag{A.2}$$

This is a matrix geometric sum, i.e., each matrix in the sum is the previous one multiplied by $e^{(\mathbf{A}-i\omega_k\mathbf{I})\Delta t}$. The solution to the geometric sum is $(\mathbf{I} - e^{(\mathbf{A}-i\omega_k\mathbf{I})\Delta t})^{-1}(\mathbf{I} - e^{(\mathbf{A}-i\omega_k\mathbf{I})N_\omega\Delta t})\boldsymbol{q}_0$. Since $e^{iN_\omega\Delta t} = 1$, this is

$$\hat{\boldsymbol{q}}_{k,ic} = \left(\mathbf{I} - e^{(\mathbf{A}-i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\mathbf{A}T}\right)\boldsymbol{q}_0. \tag{A.3}$$

Next, we evaluate $\hat{\boldsymbol{q}}_{k,int}$. Using assumptions *(i)* and *(iii)*, this may be rewritten as

$$\hat{\boldsymbol{q}}_{k,int} = \frac{1}{N_\omega}\sum_{j=0}^{N_\omega-1} e^{-i\omega_k j\Delta t} e^{\mathbf{A}j\Delta t}\int_0^{j\Delta t}\sum_{l=0}^{N_\omega-1} e^{(i\omega_l\mathbf{I}-\mathbf{A})t'}\left(\mathbf{B}\hat{\boldsymbol{f}}_l + \hat{\boldsymbol{n}}_l\right)\ dt'. \tag{A.4}$$

Integrating brings out a resolvent operator and (A.4) becomes

$$\hat{\boldsymbol{q}}_{k,int} = \frac{1}{N_\omega}\sum_{j=0}^{N_\omega-1} e^{-i\omega_k j\Delta t}\sum_{l=0}^{N_\omega-1}\mathbf{R}_l\left(e^{i\omega_l j\Delta t} - e^{\mathbf{A}j\Delta t}\right)\left(\mathbf{B}\hat{\boldsymbol{f}}_l + \hat{\boldsymbol{n}}_l\right). \tag{A.5}$$

Simplifying, we have

$$\hat{\boldsymbol{q}}_{k,int} = \frac{1}{N_\omega}\sum_{j=0}^{N_\omega-1}\sum_{l=0}^{N_\omega-1}\mathbf{R}_l\left(e^{i(\omega_l-\omega_k)j\Delta t} - e^{(\mathbf{A}-i\omega_k\mathbf{I})j\Delta t}\right)\left(\mathbf{B}\hat{\boldsymbol{f}}_l + \hat{\boldsymbol{n}}_l\right). \tag{A.6}$$

The frequency difference term evaluates to zero for $\omega_l \neq \omega_k$, and the other term may be evaluated by noting that it is a geometric sum. With this, we have

$$\hat{\boldsymbol{q}}_{k,int} = \mathbf{R}_k\left(\mathbf{B}\hat{\boldsymbol{f}}_k + \hat{\boldsymbol{n}}_k\right) + \frac{1}{N_\omega}\left(\mathbf{I} - e^{(\mathbf{A}-i\omega_k\mathbf{I})\Delta t}\right)^{-1}\left(\mathbf{I} - e^{\mathbf{A}T}\right)\sum_{l=0}^{N_\omega-1}\mathbf{R}_l\left(\mathbf{B}\hat{\boldsymbol{f}}_l + \hat{\boldsymbol{n}}_l\right). \tag{A.7}$$

Adding $\hat{\boldsymbol{q}}_{k,ic}$ and $\hat{\boldsymbol{q}}_{k,int}$, we recover (3.5).

Appendix B Analysis of the solution procedure

Here, we analyze two limiting cases of the fixed-point iteration used to find the solution of the nonlinear system of equations. While we do not have practical necessary and sufficient conditions for the convergence of the iteration, these limiting cases are consistent with our observation that the cases where the iteration did not converge were ones in which the nonlinearity was strong compared to the other terms.

The nonlinear system of equations to be solved is given in (3.27), and, in order to avoid the notational burden in this section, we rewrite it as

$$\boldsymbol{x} = \boldsymbol{c} + \boldsymbol{f}(\boldsymbol{x}), \tag{B.1}$$

where $\boldsymbol{c} \in \mathbb{C}^N$ is a known constant vector, $\boldsymbol{f} : \mathbb{C}^N \to \mathbb{C}^N$ is a known nonlinear function, and $\boldsymbol{x} \in \mathbb{C}^N$ is the unknown vector to be obtained.

B.1 Small nonlinearity

Here, we analyze the case where the nonlinear term is small, and show a correspondence between the i-th fixed-point iterate and the i-th term of a perturbation series about $\boldsymbol{c}$. We require that the nonlinear term be analytic, so the equations to solve are written

$$\boldsymbol{x} = \boldsymbol{c} + \epsilon \sum_{k=2}^{\infty} \boldsymbol{f}_k \overbrace{(\boldsymbol{x}, \dots, \boldsymbol{x})}^{k \text{ arguments}}, \tag{B.2}$$

where $\epsilon \in \mathbb{R}$, and $\boldsymbol{f}_k$ is linear in each of its k arguments. We have excluded the $k = 0$ and $k = 1$ terms of the Taylor series for the nonlinearity since we have assumed that constant and linear terms of the right-hand side in the governing equations are accounted for by $\boldsymbol{c}$, though the following holds if these terms are included in the sum as well.

B.1.1 Perturbation series

If $\epsilon = 0$, then $\boldsymbol{x} = \boldsymbol{c}$. We use the regular perturbation ansatz $\boldsymbol{x} = \boldsymbol{x}_0 + \epsilon \boldsymbol{x}_1 + \epsilon^2 \boldsymbol{x}_2 + \dots$ to search for solutions near $\boldsymbol{c}$ when ϵ is small. To find a recurrence relation for $\boldsymbol{x}_i$, we insert this perturbation ansatz into (B.2), giving

$$\sum_{i=0}^{\infty} \epsilon^i \boldsymbol{x}_i = \boldsymbol{c} + \epsilon \sum_{k=2} \boldsymbol{f}_k \left(\sum_{i_1=0}^{\infty} \epsilon^{i_1} \boldsymbol{x}_{i_1}, \dots, \sum_{i_k=0}^{\infty} \epsilon^{i_k} \boldsymbol{x}_{i_k} \right). \tag{B.3}$$

Using the multilinearity of $\boldsymbol{f}_k$ and grouping powers of ϵ, this may be rewritten as

$$\sum_{i=0}^{\infty} \epsilon^i \boldsymbol{x}_i = \boldsymbol{c} + \sum_{i=1}^{\infty} \epsilon^i \sum_{k=2}^{\infty} \sum_{i_1+\dots+i_k=i-1} \boldsymbol{f}(\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k}). \tag{B.4}$$

Here, the indices in the inner sum are non-negative integers. Finally, equating like powers of ϵ on the right and left, we have the following recurrence relation

$$\boldsymbol{x}_0 = \boldsymbol{c}, \tag{B.5a}$$

$$\boldsymbol{x}_i = \sum_{k=2}^{\infty} \sum_{i_1+\dots+i_k=i-1} \boldsymbol{f}_k(\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k}). \tag{B.5b}$$

(B.5b) is a recurrence relation since the highest order term on the right is $\boldsymbol{x}_{i-1}$.

B.1.2 Fixed point iteration

The fixed-point iteration with the analytic right-hand side is

$$\boldsymbol{x}^0 = \boldsymbol{c} \tag{B.6a}$$

$$\boldsymbol{x}^{i+1} = \boldsymbol{c} + \epsilon \sum_{k=2}^{\infty} \boldsymbol{f}_k(\boldsymbol{x}^i, \dots, \boldsymbol{x}^i). \tag{B.6b}$$

B.1.3 Relation between the two

Statement

The i-th fixed point iterate is equal to the perturbation series truncated at the i-th order *to* $\mathcal{O}(\epsilon^i)$. That is,

$$\boldsymbol{x}^i = \sum_{j=0}^{i} \epsilon^j \boldsymbol{x}_j + \epsilon^{i+1} \boldsymbol{r}_i \quad \text{for} \quad i \in \{0, 1, \dots\}, \tag{B.7}$$

where $\boldsymbol{r}_i$ is an error term, and the prefactor of ϵ^{i+1} indicates that this error term is of order ϵ^{i+1} or higher.

Proof

We prove (B.7) inductively. The base case is trivial: the zeroth iterate is exactly equal to the zeroth term in the perturbation series, since both are equal to $\boldsymbol{c}$ (see (B.5a) and (B.6a)). We assume the statement is true for $i = n$, and show that this implies it for $i = n + 1$. Namely, we assume

$$\boldsymbol{x}^n = \sum_{j=0}^{n} \epsilon^j \boldsymbol{x}_j + \epsilon^{n+1} \boldsymbol{r}_n \tag{B.8}$$

With this assumption, the $n + 1$-st fixed point iterate is

$$\boldsymbol{x}^{n+1} = \boldsymbol{c} + \epsilon \sum_{k=2}^{\infty} \boldsymbol{f}_k \left(\sum_{i_1=0}^{n} \epsilon^{i_1} \boldsymbol{x}_{i_1} + \epsilon^{n+1} \boldsymbol{r}_n, \dots, \sum_{i_k=0}^{n} \epsilon^{i_k} \boldsymbol{x}_{i_k} + \epsilon^{n+1} \boldsymbol{r}_n \right). \tag{B.9}$$

All terms involving $\boldsymbol{r}_n$ are of order ϵ^{n+2} or higher, so they may be grouped into the $n + 1$-th error term. By grouping terms at the same order in ϵ, and combining all terms of order higher than ϵ^{n+1} into the error term, (B.9) may be rewritten as

$$\boldsymbol{x}^{n+1} = \boldsymbol{c} + \sum_{j=1}^{n+1} \epsilon^j \sum_{k=2}^{\infty} \sum_{i_1 + \dots + i_k = j-1} \boldsymbol{f}_k \left(\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k} \right) + \epsilon^{n+2} \boldsymbol{r}_{n+1}. \tag{B.10}$$

Since $\boldsymbol{x}_0 = \boldsymbol{c}$ and since the inner two sums are $\boldsymbol{x}_j$ by (B.5b), the above equation may be rewritten as

$$\boldsymbol{x}^{n+1} = \sum_{j=0}^{n+1} \epsilon^j \boldsymbol{x}_j + \epsilon^{n+2} \boldsymbol{r}_{n+1}. \tag{B.11}$$

This completes the proof.

B.2 Bilinear right-hand side

Here we analyze the iteration in the case where $\boldsymbol{c} = \mathbf{0}$ and where $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x}, \boldsymbol{x})$ where $\boldsymbol{b} : \mathbb{C}^N \times \mathbb{C}^N \to \mathbb{C}^N$ is bilinear. This case is relevant where the nonlinear term is quadratic and dominant, and this case is sometimes discussed in the context of resolvent analysis (see Fig. 1 of Ref. [24], a review of the subject). We show that neither the fixed-point iteration nor the pseudo-time-stepping method can converge to nonzero fixed points due to a linear instability about these points. We first analyze the fixed-point iteration

$$\boldsymbol{x}^{i+1} = \boldsymbol{b}(\boldsymbol{x}^i, \boldsymbol{x}^i) \tag{B.12}$$

near a solution $\overline{\boldsymbol{x}}$ satisfying $\overline{\boldsymbol{x}} = \boldsymbol{b}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}})$. Writing $\boldsymbol{x}^i = \overline{\boldsymbol{x}} + \epsilon \boldsymbol{y}^i$ and inserting this into the iteration, we have

$$\boldsymbol{y}^{i+1} = \epsilon \left(\boldsymbol{b}(\overline{\boldsymbol{x}}, \boldsymbol{y}^i) + \boldsymbol{b}(\boldsymbol{y}^i, \overline{\boldsymbol{x}}) \right) + \epsilon^2 \boldsymbol{b}(\boldsymbol{y}^i, \boldsymbol{y}^i). \tag{B.13}$$

Sufficiently close to the fixed point (i.e., with ϵ sufficiently small), the dynamics are then given by

$$\boldsymbol{y}^{i+1} = \mathbf{L}\boldsymbol{y}^i, \tag{B.14}$$

where $\mathbf{L}\boldsymbol{y}^i = \boldsymbol{b}(\overline{\boldsymbol{x}}, \boldsymbol{y}^i) + \boldsymbol{b}(\boldsymbol{y}^i, \overline{\boldsymbol{x}})$ is the Jabobian about the fixed point $\overline{\boldsymbol{x}}$. If all the eigenvalues of $\mathbf{L}$ are within the unit circle, $\|\boldsymbol{y}\|$ decays as the iteration proceeds, and the fixed point $\overline{\boldsymbol{x}}$ is stable; otherwise, it is unstable.

If $\overline{\boldsymbol{x}} = \mathbf{0}$, then the fixed point is stable since $\mathbf{L}$ is the zero matrix. However, if $\overline{\boldsymbol{x}} \neq \mathbf{0}$, the fixed point is unstable because there is an eigenvalue of 2 associated with the eigenvector $\boldsymbol{y} = \overline{\boldsymbol{x}}$. This is true because $\mathbf{L}\overline{\boldsymbol{x}} = \boldsymbol{b}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}) + \boldsymbol{b}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}) = 2\overline{\boldsymbol{x}}$. This argument can be generalized to the case when the constant is zero and the nonlinearity is any k-linear function. In this case, the origin is again always a stable fixed point (for $k > 1$), and the linear operator about any other fixed point admits an eigenvalue of k associated with the eigenvector $\overline{\boldsymbol{x}}$.

The above analysis implies that the pseudo-time-stepping method also cannot converge to non-zero fixed points in the case of $\boldsymbol{c} = \mathbf{0}$ and a k-linear right-hand side since the stability in that case requires that the eigenvalues of the same Jacobian have a real part less than 1.

Appendix C Simulation-based approximation of operators

In the main text, we approximated the operator $\boldsymbol{\Psi}_k^* \mathbf{W} \left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k \mathbf{I})\Delta t} \right)^{-1} \left(\mathbf{I} - e^{\mathbf{A}T} \right) \boldsymbol{\Phi} \in \mathbb{C}^{r_k \times p_1}$ by using the SPOD modes to form a Galerkin-type approximation of $\mathbf{A}$. Here, we detail a means of forming this operator exactly (up to FOM errors) by performing p_1 runs of length T of the linearized FOM. The benefit of the approach is that this accounts more accurately for the transient term. The drawbacks are that this requires additional simulations and that the affine parameter dependence cannot be accounted for, i.e., the matrix cannot be modified quickly to new parameters.

Equation (A.3) implies

$$\left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k \mathbf{I})\Delta t} \right)^{-1} \left(\mathbf{I} - e^{\mathbf{A}T} \right) \boldsymbol{v}_0 = \mathrm{DFT}_k \left[e^{\mathbf{A} t_{\mathcal{J}}} \boldsymbol{v}_0 \right]. \tag{C.1}$$

That is, the effect of the operator on the left-hand side above applied to a vector $\boldsymbol{v}_0 \in \mathbb{C}^{N_x}$ is to take the k-th frequency of the DFT of the time series generated by using $\boldsymbol{v}_0$ as the initial condition to the linear system $\dot{\boldsymbol{v}} = \mathbf{A}\mathbf{v}$.

Using this equivalence, the l-th column of the matrix $\boldsymbol{\Psi}_k^* \mathbf{W} \left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k \mathbf{I})\Delta t}\right)^{-1} \left(\mathbf{I} - e^{\mathbf{A}T}\right) \boldsymbol{\Phi}$ is $\boldsymbol{\Psi}_k^* \mathbf{W} \hat{\boldsymbol{\phi}}_k^l$, where $\hat{\boldsymbol{\phi}}_k^l = \mathrm{DFT}_k[e^{\mathbf{A}t_{\mathcal{J}}} \boldsymbol{\phi}^l]$ is the k-th component of the DFT of the time series created by using the l-th column of $\boldsymbol{\Phi}$ as an initial condition to the unforced linear system. Thus, by initializing this linear system with each columns of $\boldsymbol{\Phi}$, and taking the DFT of each result, all matrices $\boldsymbol{\Psi}_k^* \mathbf{W} \left(\mathbf{I} - e^{(\mathbf{A} - i\omega_k \mathbf{I})\Delta t}\right)^{-1} \left(\mathbf{I} - e^{\mathbf{A}T}\right) \boldsymbol{\Phi}$ can be computed up to the time-stepping errors in integrating the linear system.

In principle, linear runs could also be used to compute the action of $\boldsymbol{\Psi}_k^* \mathbf{W} \mathbf{R}_k$, and in a different context, such a strategy has been employed by Ref. [11]. In the present context, such an approximation would involve performing linear runs with harmonic forcings with spatial modes corresponding to the vectors to which $\boldsymbol{\Psi}_k^* \mathbf{W} \mathbf{R}_k$ is applied. However, unlike the runs described above, these linear runs would need to be long enough for transients to decay [11]. Moreover, with the triadic-interaction handling of the nonlinearity, there are prohibitively many vectors that would need to be used as the forcing in separate runs. Thus, we do not recommend this approach in this context.

Appendix D Algorithms

The offline phase of the method is given in Algorithm 1. The first step in the online phase of the method is computing the constant term using Algorithm 2. Then, either the fixed point iteration or the pseudo-time-stepping method where the nonlinear term $\boldsymbol{w}_{\mathcal{K}}(\boldsymbol{a}_{\mathcal{K}})$ is calculated using either Algorithm 3 or Algorithm 4, depending on the nature of the nonlinearity. For readability, we have written Algorithm 4 with for loops, but we note that in our numerical implementation, we used sparse matrices, which will run faster on most systems.

Algorithm 1 SSOP (offline)

```
1: $\mathbf{\Phi} = \texttt{POD}(\mathbf{Q}^t, \mathbf{W}, p_1)$    ▷ Intermediary basis
2: $\hat{\mathbf{Q}}_{\mathcal{K}} = \texttt{WelchBlocks}(\mathbf{Q}^t, N_\omega)$    ▷ Obtain data matrices of each frequency from snapshot matrix
3: if $\boldsymbol{n}$ is non-quadratic then
4:     $[\mathbf{U}^n, \mathbf{P}^{nT}] = \texttt{DEIM}(\boldsymbol{n}(\mathbf{Q}^t), p_2)$    ▷ Obtain sample points and basis for nonlinearity
5: end if
6: $\left[\mathbf{\Psi}_k^{N_d}, \mathbf{\Lambda}_k^{N_d}, \mathbf{\Psi}_k, \mathbf{\Lambda}_k\right] = \texttt{SPOD}(\hat{\mathbf{Q}}_{\mathcal{K}}, r)$    ▷ Get SPOD modes and energies
7: for $k \in \mathcal{K}$ do
8:     $\mathbf{L}_k \leftarrow (i\omega_k \mathbf{I} - \mathbf{A})$
9:     $\mathbf{G}_k \leftarrow \mathbf{L}_k \hat{\mathbf{Q}}_k$    ▷ Used for approximating resolvent
10:    $\mathbf{E}_k \leftarrow \mathbf{\Psi}_k^* \mathbf{W} \hat{\mathbf{Q}}_k (\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \mathbf{G}_k^* \mathbf{W} \mathbf{B}$    ▷ Output variable
11:    $\mathbf{J}_k \leftarrow \mathbf{\Phi}^* \mathbf{W} \hat{\mathbf{Q}}_k (\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \mathbf{G}_k^* \mathbf{W} \mathbf{B}$    ▷ Output variable
12:    $\tilde{\mathbf{A}}_k \leftarrow \mathbf{\Psi}_k^{N_d *} \mathbf{W} \mathbf{A} \mathbf{\Psi}_k^{N_d}$    ▷ Used for calculating $\mathbf{H}_k$
13:    $\mathbf{P}_k \leftarrow \left[\mathbf{I}_{r_k \times r_k} \ \mathbf{0}_{r_k \times (N_d - r_k)}\right]$    ▷ Used for calculating $\mathbf{H}_k$
14:    $\mathbf{H}_k \leftarrow \mathbf{P}_k \left(\mathbf{I} - \exp\left[\left(\tilde{\mathbf{A}} - i\omega_k \mathbf{I}\right)\Delta t\right]\right)^{-1} \left(\mathbf{I} - \exp\left[\tilde{\mathbf{A}} T\right]\right) \left(\mathbf{\Psi}_k^{N_d *} \mathbf{W} \mathbf{\Phi}\right)$    ▷ Output variable
15:    if $\boldsymbol{n}$ is non-quadratic then
16:        $\mathbf{N}_k \leftarrow \mathbf{\Psi}_k^* \mathbf{W} \hat{\mathbf{Q}}_k (\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \mathbf{G}_k^* \mathbf{W} \mathbf{U}^n \left(\mathbf{P}^{nT} \mathbf{U}^n\right)^{-1}$    ▷ Output variable
17:        $\mathbf{M}_k \leftarrow \mathbf{\Phi}^* \mathbf{W} \hat{\mathbf{Q}}_k (\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \mathbf{G}_k^* \mathbf{W} \mathbf{U}^n \left(\mathbf{P}^{nT} \mathbf{U}^n\right)^{-1}$    ▷ Output variable
18:        $\mathbf{S}_k \leftarrow \mathbf{P}^{nT} \mathbf{\Psi}_k$    ▷ Output variable
19:    else
20:        $\boldsymbol{b}(\boldsymbol{q}_1, \boldsymbol{q}_2) := \frac{1}{2}\left[\boldsymbol{n}(\boldsymbol{q}_1 + \boldsymbol{q}_2) - \boldsymbol{n}(\boldsymbol{q}_1) - \boldsymbol{n}(\boldsymbol{q}_2)\right]$    ▷ Symmetric bilinear form
21:        $\mathcal{N}_k \leftarrow \{\}$
22:        for $(l, i)$ such that $\omega_l + \omega_i = \omega_k$ do    ▷ Looping over triadic interactions
23:            for $m \in \{1, \ldots, r_l\}$ do
24:                for $n \in \{1, \ldots, r_i\}$ do
25:                    $\tilde{\boldsymbol{n}} \leftarrow \mathbf{\Psi}_k^* \mathbf{W} \hat{\mathbf{Q}}_k (\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \mathbf{G}^* \mathbf{W} \boldsymbol{b}(\boldsymbol{\psi}_{lm}, \boldsymbol{\psi}_{in})$
26:                    if $\|\tilde{\boldsymbol{n}}\|_x^2 \lambda_{lm} \lambda_{in} > \epsilon$ then    ▷ Checking if interaction meets threshold
27:                        $\boldsymbol{n}_{klmn} \leftarrow \tilde{\boldsymbol{n}}$    ▷ Output variable
28:                        $\boldsymbol{m}_{klmn} \leftarrow \mathbf{\Phi}^* \mathbf{W} \hat{\mathbf{Q}}_k (\mathbf{G}_k^* \mathbf{W} \mathbf{G}_k)^{-1} \mathbf{G}^* \mathbf{W} \boldsymbol{b}(\boldsymbol{\psi}_{lm}, \boldsymbol{\psi}_{in})$    ▷ Output variable
29:                        $\mathcal{N}_k \leftarrow \mathcal{N}_k \cup (l, m, n)$    ▷ Updating list of interactions
30:                    end if
31:                end for
32:            end for
33:        end for
34:    end if
35: end for
```

Inputs: $\mathbf{Q}^t$, time series data; Δt, time step used in data; N_ω, number of time steps desired for solution blocks; $\boldsymbol{n}$, nonlinear function; p_1, number of modes to use in intermediary basis; p_2 (if DEIM is used), number of sample points to use in DEIM approximation; r, average number of modes per frequency for ROM; $\mathbf{A}$, $\mathbf{B}$, system matrices; $\mathbf{W}$, weight matrix; ϵ (if triadic interactions are used), tolerance for inclusion.
Outputs: $\mathbf{\Phi}$, intermediary basis; $\mathbf{J}_{\mathcal{K}}$, $\mathbf{J}$ operators at all frequencies; $\mathbf{E}_{\mathcal{K}}$, $\mathbf{E}$ operators at all frequencies ; $\mathbf{H}_{\mathcal{K}}$, $\mathbf{H}$ operators at all frequencies. If using DEIM, $\mathbf{S}_{\mathcal{K}}$, the sampling operators at all frequencies; $\mathbf{N}_{\mathcal{K}}$, the first set of hyper-reduction matrices; $\mathbf{M}_{\mathcal{K}}$ the second set of hyper-reduction matrices. Otherwise, $\mathcal{N}_{\mathcal{K}}$, the set $\mathcal{N}$ at all frequencies; $\boldsymbol{n}_{klmn}$ for all $k \in \mathcal{K}$, $(l, m, n) \in \mathcal{N}_k$; $\boldsymbol{m}_{klmn}$ for all $k \in \mathcal{K}$, $(l, m, n) \in \mathcal{N}_k$.

Algorithm 2 SSOP (online, constant term)

```
1: $\hat{\boldsymbol{f}}_{\mathcal{K}} = \texttt{FFT}_{\mathcal{K}}[\boldsymbol{f}_{\mathcal{J}}]$    ▷ FFT of forcing
2: $\boldsymbol{q}_0^{\boldsymbol{\Phi}} \leftarrow \boldsymbol{\Phi}^*\mathbf{W}\boldsymbol{q}_0$    ▷ The initial condition in the intermediary basis
3: $\tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}} \leftarrow \mathbf{0}_{p_1\times 1}$    ▷ Initializing the forcing sum term
4: for $k \in \mathcal{K}$ do
5:     $\tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}} \leftarrow \tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}} + \frac{1}{N_\omega}\mathbf{J}_k\hat{\boldsymbol{f}}_k$    ▷ Influence of each frequency on forcing sum
6: end for
7: for $k \in \mathcal{K}$ do
8:     $\boldsymbol{c}_k \leftarrow \mathbf{E}_k\hat{\boldsymbol{f}}_k + \mathbf{H}_k\left(\boldsymbol{q}_0^{\boldsymbol{\Phi}} - \tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}}\right)$    ▷ Constructing each frequency of the constant term
9: end for
```

Inputs: $\boldsymbol{q}_0$, the initial condition; $\boldsymbol{\Phi}$, intermediary basis (POD modes); $\boldsymbol{W}$, weight matrix; $\boldsymbol{f}_{\mathcal{J}}$, the forcing on the temporal grid; $\mathbf{J}_{\mathcal{K}}$, $\mathbf{J}$ operators at all frequencies (defined in 2.27); $\mathbf{E}_{\mathcal{K}}$, $\mathbf{E}$ operators at all frequencies (defined in (2.24)); $\mathbf{H}_{\mathcal{K}}$, $\mathbf{H}$ operators at all frequencies (defined below 2.29).
Outputs: $\boldsymbol{c}_{\mathcal{K}}$, all frequency components of the constant term.

Algorithm 3 SSOP (online, non-quadratic nonlinear term)

```
1: for $k \in \mathcal{K}$ do
2:     $\tilde{\boldsymbol{q}}_k^s \leftarrow \mathbf{S}_k\tilde{\boldsymbol{a}}_k$    ▷ Getting sample points in frequency domain
3: end for
4: $\boldsymbol{q}_{\mathcal{J}}^s = \texttt{IFFT}[\tilde{\boldsymbol{q}}_{\mathcal{K}}^s]$    ▷ Sample points in time domain
5: $\boldsymbol{n}_{\mathcal{J}}^s = \boldsymbol{n}(\boldsymbol{q}_{\mathcal{J}}^s)$    ▷ Nonlinearity at sample points in time domain
6: $\hat{\boldsymbol{n}}_{\mathcal{K}}^s = \texttt{FFT}[\boldsymbol{n}_{\mathcal{J}}^s]$    ▷ Nonlinearity at sample points in frequency domain
7: $\tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}} \leftarrow \mathbf{0}_{p_1\times 1}$
8: for $k \in \mathcal{K}$ do
9:     $\tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}} \leftarrow \tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}} + \frac{1}{N_\omega}\mathbf{M}_k\hat{\boldsymbol{n}}_k^s$
10:    $\boldsymbol{w}_k \leftarrow \mathbf{N}_k\hat{\boldsymbol{n}}_k^s$
11: end for
12: for $k \in \mathcal{K}$ do
13:    $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_k - \mathbf{H}_k\tilde{\boldsymbol{q}}_0^{\boldsymbol{\Phi}}$
14: end for
```

Inputs: $\boldsymbol{a}_{\mathcal{K}}$, the SPOD coefficients at the current iteration; $\mathbf{S}_{\mathcal{K}}$, the sampling operators at all frequencies; $\boldsymbol{n}$, the nonlinear function; $\mathbf{N}_{\mathcal{K}}$, the first set of matrices hyper-reduction matrices (see (3.11)); $\mathbf{M}_{\mathcal{K}}$ the second set of hyper-reduction matrices (see (3.12)); $\mathbf{H}_{\mathcal{K}}$, the set of matrices defined under (2.29).
Output: $\boldsymbol{w}_{\mathcal{K}}$, the approximate nonlinear term formed by concatenating $\boldsymbol{w}$ at all frequencies.

Algorithm 4 SSOP (online, quadratic nonlinear term)

```
1: $\boldsymbol{w}^{\boldsymbol{\Phi}} \leftarrow \mathbf{0}_{p_1 \times 1}$
2: for $k \in \mathcal{K}$ do
3:     $\boldsymbol{w}_k \leftarrow \mathbf{0}_{r_k \times 1}$
4:     for $(l, m, n) \in \mathcal{N}_k$ do
5:         $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_k + \boldsymbol{n}_{klmn} \tilde{a}_{lm} \tilde{a}_{in}$
6:         $\boldsymbol{w}^{\boldsymbol{\Phi}} \leftarrow \boldsymbol{w}^{\boldsymbol{\Phi}} + \boldsymbol{m}_{klmn} \tilde{a}_{lm} \tilde{a}_{in}$
7:     end for
8: end for
9: for $k \in \mathcal{K}$ do
10:     $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_k - \mathbf{H}_k \boldsymbol{w}^{\boldsymbol{\Phi}}$
11: end for
```

Inputs: $\boldsymbol{a}_{\mathcal{K}}$, the SPOD coefficients at the current iteration; $\mathcal{N}_{\mathcal{K}}$, the set $\mathcal{N}$ at all frequencies (see (3.19)); $\boldsymbol{n}_{klmn}$ for all $k \in \mathcal{K}$, $(l, m, n) \in \mathcal{N}_k$ (see (3.17a)); $\boldsymbol{m}_{klmn}$ for all $k \in \mathcal{K}$, $(l, m, n) \in \mathcal{N}_k$ (see (3.17b)); $\mathbf{H}_{\mathcal{K}}$, the set of matrices defined under (2.29).
Output: $\boldsymbol{w}_{\mathcal{K}}$, the approximate nonlinear term formed by concatenating $\boldsymbol{w}$ at all frequencies.

References

[1] D. G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, Oct. 1965.

[2] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *Journal of Fluid Mechanics*, 192:115–173, July 1988.

[3] S. Bagheri, D. S. Henningson, J. Hœpffner, and P. J. Schmid. Input-output analysis and control design applied to a linear model of spatially developing flows. *Applied Mechanics Reviews*, 62(2), Feb. 2009.

[4] S. L. Brunton, J. H. Tu, I. Bright, and J. N. Kutz. Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. *SIAM Journal on Applied Dynamical Systems*, 13(4):1716–1732, Jan. 2014.

[5] K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, Oct. 2010.

[6] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, Jan. 2010.

[7] K. K. Chen and C. W. Rowley. Optimal actuator and sensor placement in the linearised complex Ginzburg–Landau system. *Journal of Fluid Mechanics*, 681:241–260, June 2011.

[8] Y. Chen and C. Liu. Data driven robust control of complex ginzburg-landau equations for spatial developing flow. In *2021 IEEE 4th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, page 490–500. IEEE, Nov. 2021.

[9] Y. Choi, P. Brown, W. Arrighi, R. Anderson, and K. Huynh. Space–time reduced order model for large-scale linear dynamical systems with application to Boltzmann transport problems. *Journal of Computational Physics*, 424:109845, 2021.

[10] Y. Choi and K. Carlberg. Space–time least-squares petrov–galerkin projection for nonlinear model reduction. *SIAM Journal on Scientific Computing*, 41(1):A26–A58, 2019.

[11] A. Farghadan, E. Martini, and A. Towne. Scalable resolvent analysis for three-dimensional flows. *Journal of Computational Physics*, 524:113695, Mar. 2025.

[12] P. Frame, C. Lin, O. T. Schmidt, and A. Towne. Linear model reduction using spectral proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 447:196–215, Dec. 2025.

[13] P. Frame and A. Towne. Space-time POD and the Hankel matrix. *PLOS ONE*, 18(8):e0289637, Aug. 2023.

[14] K. C. Hall, K. Ekici, J. P. Thomas, and E. H. Dowell. Harmonic balance methods applied to computational fluid dynamics problems. *International Journal of Computational Fluid Dynamics*, 27(2):52–67, Jan. 2013.

[15] K. C. Hall, J. P. Thomas, and W. S. Clark. Computation of unsteady nonlinear flows in cascades using a harmonic balance technique. *AIAA Journal*, 40(5):879–886, May 2002.

[16] C. Hoang, K. Chowdhary, K. Lee, and J. Ray. Projection-based model reduction of dynamical systems using space–time subspace and machine learning. *Computer Methods in Applied Mechanics and Engineering*, 389:114341, Feb. 2022.

[17] M. Ilak, S. Bagheri, L. Brandt, C. W. Rowley, and D. S. Henningson. Model reduction of the nonlinear complex ginzburg–landau equation. *SIAM Journal on Applied Dynamical Systems*, 9(4):1284–1302, Jan. 2010.

[18] Y. Kim, K. Wang, and Y. Choi. Efficient space–time reduced order model for linear dynamical systems in python using less than 120 lines of code. *Mathematics*, 9(14), 2021.

[19] B. Kramer, B. Peherstorfer, and K. E. Willcox. Learning nonlinear reduced models from data with operator inference. *Annual Review of Fluid Mechanics*, 56(1):521–548, Jan. 2024.

[20] X. Li and D. Lasagna. Space-time nonlinear reduced-order modelling for unsteady flows, 2025.

[21] C. Lin. Model order reduction in the frequency domain via spectral proper orthogonal decomposition. *Master's thesis, University of Illinois at Urbana-Champaign*, 2019.

[22] J. L. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*, 1967.

[23] J. L. Lumley. Stochastic tools in turbulence. 1970.

[24] B. J. McKeon. The engine behind (wall) turbulence: perspectives on scale interactions. *Journal of Fluid Mechanics*, 817:P1, 2017.

[25] B. J. McKeon and A. S. Sharma. A critical-layer framework for turbulent pipe flow. *Journal of Fluid Mechanics*, 658:336–382, July 2010.

[26] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, Feb. 1981.

[27] B. R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, Dec. 2003.

[28] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, Nov. 1982.

[29] S. A. Orszag. On the elimination of aliasing in finite-difference schemes by filtering high-wavenumber components. *Journal of the Atmospheric Sciences*, 28(6):1074–1074, Sept. 1971.

[30] A. Padovan, B. Vollmer, and D. J. Bodony. Data-driven model reduction via non-intrusive optimization of projection operators and reduced-order dynamics, 2024.

[31] E. J. Parish and K. T. Carlberg. Windowed least-squares model reduction for dynamical systems. *Journal of Computational Physics*, 426:109939, 2021.

[32] B. Peherstorfer and K. Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, July 2016.

[33] G. Rigas, D. Sipp, and T. Colonius. Nonlinear input/output analysis: application to boundary layer transition. *Journal of Fluid Mechanics*, 911, 2021.

[34] C. W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03):997–1013, Mar. 2005.

[35] P. J. Schmid. Nonmodal stability theory. *Annual Review of Fluid Mechanics*, 39(1):129–162, Jan. 2007.

[36] O. T. Schmidt. Bispectral mode decomposition of nonlinear flows. *Nonlinear Dynamics*, 102(4):2479–2501, Nov. 2020.

[37] D. Sipp, M. Fosas de Pando, and P. J. Schmid. Nonlinear model reduction: A comparison between pod-galerkin and pod-deim methods. *Computers & Fluids*, 208:104628, Aug. 2020.

[38] L. Sirovich. Turbulence and the dynamics of coherent structures. ii. symmetries and transformations. *Quarterly of Applied Mathematics*, 45(3):573–582, 1987.

[39] A. Towne. Space-time Galerkin projection via spectral proper orthogonal decomposition and resolvent modes. In *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2021.

[40] A. Towne, O. T. Schmidt, and T. Colonius. Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. *Journal of Fluid Mechanics*, 847:821–867, 2018.

[41] L. N. Trefethen, A. E. Trefethen, S. C. Reddy, and T. A. Driscoll. Hydrodynamic stability without eigenvalues. *Science*, 261(5121):578–584, 1993.

[42] H. F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, Jan. 2011.

[43] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.

[44] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, Nov. 2002.

[45] M. Yano, A. T. Patera, and K. Urban. A space-time hp-interpolation-based certified reduced basis method for Burgers' equation. *Mathematical Models and Methods in Applied Sciences*, 24(09):1903–1935, May 2014.