Adaptive POD-based low-dimensional modeling supported by residual estimates

M.-L. Rapún[1], F. Terragni[2] and J. M. Vega[1,]

[1]*E.T.S.I. Aeronáutica y del Espacio, Universidad Politécnica de Madrid, 28040 Madrid, Spain*
[2]*Gregorio Millán Institute for Fluid Dynamics, Nanoscience and Industrial Mathematics, Universidad Carlos III de Madrid, 28911 Leganés, Spain*

SUMMARY

An adaptive low-dimensional model is considered to simulate time-dependent dynamics in nonlinear dissipative systems governed by PDEs. The method combines an inexpensive POD-based Galerkin system with short runs of a standard numerical solver that provides the snapshots necessary to first construct and then update the POD modes. Switching between the numerical solver and the Galerkin system is decided 'on the fly' by monitoring (i) a truncation error estimate and (ii) a residual estimate. The latter estimate is used to control the mode truncation instability and highly improves former adaptive strategies that detected this instability by monitoring consistency with a second instrumental Galerkin system based on a larger number of POD modes. The most computationally expensive run of the numerical solver occurs at the outset, when the whole set of POD modes is calculated. This step is improved by using mode libraries, which may either be generic or result from former applications of the method. The outcome is a flexible, robust, computationally inexpensive procedure that adapts itself to the local dynamics by using the faster Galerkin system for the majority of the time and few, on demand, short runs of a numerical solver. The method is illustrated considering the complex Ginzburg–Landau equation in one and two space dimensions.

1. INTRODUCTION

Time-dependent problems governed by PDEs and systems are paramount in scientific and engineering applications. Inexpensive computation of time-marching solutions is the goal of a large variety of works in the literature, which either aim at increasing the computational efficiency of classical numerical solvers (NSs) and software or intend new approaches. Among these, reduced-order modeling [1] is a currently fashionable field that takes advantage of the fact that physical systems exhibit a number of *actually relevant DOFs* that is much smaller than the number of *numerical DOFs* (i.e., grid points or elements), which are usually determined by the spatial dimension and purely numerical constraints such as the discretization accuracy and numerical stability. The relevant DOFs, instead, are determined by the actual dynamics and amount to a smaller number because of spatio-temporal redundancies imposed by the physical laws that are implicit in the governing equations. *Reduced-order models* (ROMs) precisely intend to first identify and then use the relevant DOFs to highly improve computational efficiency. Roughly speaking, the number of numerical DOFs increases exponentially with the spatial dimension, but the number of relevant DOFs increases

at a much smaller rate, meaning that the advantage of ROMs is expected to be more evident when the spatial dimension is larger than one.

A large class of ROMs identify the relevant DOFs using Proper Orthogonal Decomposition (*POD*) [2] and substitute the governing equations by their projection (using, e.g., the *Galerkin projection*) onto *modes* associated with these relevant DOFs. POD was invented by Pearson [3] and is closely related to SVD, which had been previously discovered by Beltrami and Jordan [4]. Given a set of N vectors, POD provides a hierarchical set of modes that is an orthonormal basis of the subspace spanned by the vectors. Furthermore, the span of the first $n < N$ POD modes provides the best root mean square (RMS) approximation of the set of vectors. POD modes are defined in terms of the eigenvectors of the so-called covariance matrix, whose elements are the inner products of the vectors, see (12); the square roots of the associated eigenvalues are known as *singular values* and yield an estimate of the truncation error. Because of truncation, these ROMs exhibit the so-called *mode truncation instability*, which represents a major difficulty and may lead to spurious dynamics in a seemingly unpredictable way [5]. In other words, the retained modes are destabilized by the neglected higher-order modes, even when these exhibit a small amplitude. Thus, the instability could be undetectable by estimates based on mode truncation errors. Attempts in the literature to cure the instability include the introduction of either additional terms in the equations [6, 7] or additional specific modes, such as the so-called shift modes ([8] and references therein) and residual modes [9].

POD-based ROMs are constructed to simulate both unsteady [6, 10–15] and steady, multiparameter [16–20] problems, taking advantage of the optimality of the POD modes. The standard method relies on early ideas by Sirovich [21], who suggested applying POD to a set of representative snapshots calculated *offline* by a standard NS along the considered time/parameter span. These ROMs can be called *preprocessed* ROMs because of the preprocess required to calculate the snapshots, which can be quite computationally expensive. The *online* application of the low-dimensional model is more inexpensive than the NS, and thus, these ROMs are quite convenient when the system is to be simulated a large number of times, as in, e.g., optimal design and adaptive control. In unsteady systems, preprocessed ROMs aim at approximating attractors of the system, not transients, and can be seen as a means to interpolate in the time/parameter span for those values of time/parameters that do not correspond to the snapshots.

The computational cost of the preprocess is alleviated in the application of the so-called *proper generalized decomposition* [22] to some linear problems in parallelepipedic domains. The idea is to use a tensorial representation that separates the various spatio-temporal/parameter dimensions (as in standard separation-of-variables methods) and only requires solving some (inexpensive) one-dimensional (1D) problems in the preprocess. The preprocess can be completely avoided for a class of nonlinear dissipative systems by constructing *adaptive* POD-based ROMs [23, 24]. Indeed, as observed and discussed by the authors for a class of dissipative systems [25], the POD subspace may be only weakly dependent on both time and the involved parameters. These ROMs take advantage of such heuristic property by combining 'on the fly' (as time proceeds) a standard NS and a low-dimensional Galerkin system (GS) in interspersed time intervals, I_{NS} and I_{GS}, respectively (Figure 1). The standard NS is applied in the first I_{NS} interval to calculate some snapshots from which the first set of POD modes is calculated and used to construct the first GS that approximates the solution in the first I_{GS}. The initial POD subspace is updated, when needed, in subsequent I_{NS} intervals, which are quite short because of an efficient updating process. The key step is, obviously, deciding 'on the fly' when the POD subspace is no longer valid and updating is required. This is

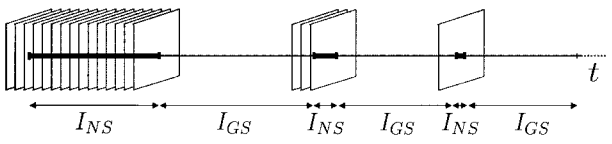

Figure 1. Sketch of the adaptive reduced-order model. Snapshots (the planes) are numerical solver (NS)-computed in the I_{NS} intervals.

performed by identifying the two main reasons for a given set of POD modes to be invalid, namely truncation errors and the mode truncation instability. Truncation errors are accounted for using a standard error estimate based on the amplitude of some high-order modes. The mode truncation instability is detected using a second instrumental GS that retains a larger number of POD modes and requiring consistency between the results provided by both GSs. In order to increase computational efficiency, both POD and the Galerkin projection are based on an inner product that uses only a few points of the computational mesh. The resulting methodology is a flexible, robust means to simulate complex transient dynamics, with the approximation errors continuously monitored. Similar adaptive ROMs have also been constructed to simulate complex bifurcation diagrams [26].

It is to be noted that adaptive preprocessed ROMs have been pursued in a number of papers ([27, 28] and references therein), but adaptation in this context is a means to iteratively enrich the snapshots set and thus improves accuracy in multiparameter steady simulations, while adaptation is made in [23, 24] to avoid the preprocess in unsteady simulations. On the other hand, the combination of a standard NS and a GS in interspersed time intervals is not new; see [9, 29]. But such combination is used in [9, 29] to eliminate the mode truncation instability in preprocessed ROMs (with fixed POD modes), while this idea is used in [23, 24] to adapt the POD subspace to the local dynamics; the mode truncation instability is avoided by appropriately updating the POD subspace.

Because of the efficient updating methodology developed in [23, 24], the NS is used mainly in the first I_{NS} interval, where the whole set of POD modes needs to be calculated. The POD modes only need to be updated in the remaining I_{NS} intervals, which are usually very short. Thus, the main opportunity for increasing the computational efficiency relies in shortening the first I_{NS} interval, which will be carried out in this paper using appropriately selected *mode libraries*. In addition, the need of two GSs will be avoided and the required computational time essentially halved using a more inexpensive means to monitor mode truncation instabilities based on a *residual* of the governing equations.

The present paper improves the adaptive ROM developed in [23, 24] using two new ingredients, namely a residual estimate and mode libraries, proceeding as follows. The basic formulation is presented in Section 2, and the new ingredients are developed in Section 3, with the resulting ROM summarized in Section 3.3. The new method is tested in Section 4, considering the 1D complex Ginzburg–Landau equation (CGLE) and obtaining CPU acceleration factors of the order of 10 for some representative chaotic solutions. In order to illustrate the advantages of the method as the spatial dimension increases, the two-dimensional (2D) CGLE is considered in Section 5, where the obtained CPU acceleration factors are much larger, of the order of 350. The paper ends with some concluding remarks in Section 6.

2. FORMULATION AND BASIC (IMPROVED) INGREDIENTS

For clarity and generality, the modeling reduction method presented in this paper will be described in terms of the following dissipative system, giving the evolution of a state vector $\boldsymbol{q}$:

$$\partial_t \boldsymbol{q} = \mathcal{L}\boldsymbol{q} + \boldsymbol{f}(\boldsymbol{q}, t). \tag{1}$$

Here, $\mathcal{L}$ and $\boldsymbol{f}$ are (generally unbounded) linear and nonlinear operators, respectively, such that the operators $\mathcal{L}^{-1}$ and $\boldsymbol{q} \to \mathcal{L}^{-1}\boldsymbol{f}(\boldsymbol{q}, t)$ are compact [30]. This holds, in particular, if $\mathcal{L}$ is an elliptic operator (with homogenous boundary conditions) depending on the highest-order spatial derivatives and $\boldsymbol{f}$ involves lower-order derivatives only. Compactness facilitates obtaining flexible low-dimensional descriptions of (1). These assumptions have been further discussed in [25] and apply to a variety of dissipative systems of scientific and industrial interest resulting from, e.g., pattern formation, reaction–diffusion–convection, aerodynamics, and microfluidics.

Concerning the NS, time can be discretized using, e.g., the Crank–Nicolson plus Adams–Bashforth scheme [31]

$$\frac{2}{\Delta t}\left(\boldsymbol{q}^{j+1} - \boldsymbol{q}^{j}\right) = \mathcal{L}\left(\boldsymbol{q}^{j+1} + \boldsymbol{q}^{j}\right) + 3\boldsymbol{f}\left(\boldsymbol{q}^{j}, t_j\right) - \boldsymbol{f}\left(\boldsymbol{q}^{j-1}, t_{j-1}\right). \tag{2}$$

The spatial discretization is to be chosen depending on the specific problem that is being considered. For instance, finite volumes are appropriate to treat transonic aerodynamics. In order to avoid spatial interpolation, (i) the same spatial grid of the NS must be used to perform the Galerkin projection. In addition, the discretization errors already present in the NS will be ignored by (ii) comparing the results provided by the low-dimensional model with those provided by the NS itself, which is thus regarded as 'exact'. In other words, the intention is to produce good approximations to the solutions provided by the NS, not to the exact solutions of the governing equations. Items (i) and (ii) are very important to clarify the nature of the approximation, separating the errors produced by the reduced modeling process (the scope of this paper) from the errors already present in the NS, whose computation would require a careful analysis of the NS itself. In other words, comparing GS and NS solutions (as already performed in [23–25]) is the natural way of correlating small singular values to small errors. On the other hand, retaining as many POD modes as possible will mean *increasing the relevant dynamical information that is extracted from the snapshots*, and this is very important because the calculation of the snapshots is usually the most expensive part of the process.

The GS will be required to approximate the NS (namely, the difference between them be smaller than a given small ε) in time intervals of suitable length T_0, which should be not too small to avoid penalizing the computation speed because of the need of updating the POD subspace too many times. Also, a too small T_0 would yield a poor temporal approximation. This is because the difference between any two consistent numerical approximations in the interval $[t, t + T_0]$ converges to zero as $T_0 \to 0$, meaning that allowing a too small T_0 would not impose anything, unless ε is scaled with an appropriate power of T_0 (depending on the temporal discretization scheme), which will not be carried out. On the other hand, T_0 should neither be too large, especially when the dynamics of the system is chaotic, which involves intrinsic divergence of nearby trajectories in the phase space, as illustrated in Figure 3. This point was more extensively discussed in Section 2.6 of [26]. The time span T_0 could be chosen beforehand when a sufficient a priori information on the dynamics is available, but a better strategy consists in automatically selecting T_0 to adapt to the local dynamics. To this end, note that imposing consistency between the NS and the GS in time intervals of appropriate length T_0 can also be seen as imposing that the dynamical systems defined by the NS and the GS be appropriately close to each other, which means that a value of T_0 that is equal to several times the characteristic timescale of the system, δ_0, is appropriate. In other words, T_0 is selected beforehand in the first I_{GS} interval only, while in subsequent I_{GS} intervals, T_0 is calculated using the timescale δ_0, which is estimated as

$$\delta_0 = 2\pi \sqrt{\int_{I_{\mathrm{GS}}} |\boldsymbol{q}^n_{\mathrm{GS}}|^2 \, dt \Big/ \int_{I_{\mathrm{GS}}} |\partial_t \boldsymbol{q}^n_{\mathrm{GS}}|^2 \, dt}\,, \tag{3}$$

where I_{GS} is the previous I_{GS} interval and $\boldsymbol{q}^n_{\mathrm{GS}}$ is as defined in Equation (5). The timescale δ_0 can be estimated by other means. For instance, δ_0 was chosen in [26] using a Poincaré map and an estimate of the most unstable Lyapunov exponent, which were readily calculated in [26], where a bifurcation problem was considered. Here, instead, Equation (3) is less computationally expensive and produces values of δ_0 that are similar to those estimated in [26]. Also, this strategy is seen to estimate well the characteristic timescale. For instance, in the CGLEs that will be used in the following to illustrate the method, (25) and (30), the characteristic timescale is of the order of $1/\mu$; for generic runs of these equations with $\mu \sim 100$, Equation (3) yields typical values of δ_0 of the order of 0.02. Using Equation (3), we calculate T_0 as

$$T_0 = \min\left\{5\delta_0, 1.5\, T_0^{\mathrm{old}}\right\}, \tag{4}$$

which means that the time span T_0 includes several oscillations of the solution. The factors 5 and 1.5 are tunable.

The various ingredients of the adaptive reduced-order modeling methodology are now considered.

2.1. *Computationally efficient Galerkin projection*

Let $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$ be a set of POD modes that are orthonormal with respect to an inner product $\langle \cdot, \cdot \rangle$ (discussed in the following). The Galerkin projection of the evolution problem (1) onto the POD subspace relies on the expansion

$$\boldsymbol{q}(\boldsymbol{x}, t) \simeq \boldsymbol{q}^n_{\mathrm{GS}}(\boldsymbol{x}, t) = \sum_{j=1}^{n} A_j(t) \boldsymbol{Q}_j(\boldsymbol{x}), \tag{5}$$

where the mode amplitudes A_j obey the following GS:

$$\frac{dA_i}{dt} = \sum_{j=1}^{n} \mathcal{L}^{\mathrm{GS}}_{ij} A_j + f^{\mathrm{GS}}_i(A_1, \ldots, A_n, t), \tag{6}$$

which is obtained by substituting (5) into (1) and by projecting (with the same inner product used to apply POD) the resulting equations onto the retained POD modes. It follows that

$$\mathcal{L}^{\mathrm{GS}}_{ij} = \langle \boldsymbol{Q}_i, \boldsymbol{\mathcal{L}} \boldsymbol{Q}_j \rangle, \quad f^{\mathrm{GS}}_i = \left\langle \boldsymbol{Q}_i, \boldsymbol{f} \left(\sum_{k=1}^{n} A_k \boldsymbol{Q}_k, t \right) \right\rangle. \tag{7}$$

For consistency, the system of ordinary differential equations (6) is integrated using the same temporal scheme used in the NS to integrate (1). The counterpart of the scheme (2) is

$$\frac{2}{\Delta t} \left(\mathbf{A}^{j+1} - \mathbf{A}^j \right) = \boldsymbol{\mathcal{L}}^{\mathrm{GS}} \left(\mathbf{A}^{j+1} + \mathbf{A}^j \right) + 3 \boldsymbol{f}^{\mathrm{GS}} \left(\mathbf{A}^j, t_j \right) - \boldsymbol{f}^{\mathrm{GS}} \left(\mathbf{A}^{j-1}, t_{j-1} \right), \tag{8}$$

where $\mathbf{A} = (A_1, \ldots, A_n)^\top$ is the mode amplitudes vector. This scheme could have also been obtained by projecting onto the POD modes the temporal scheme (2).

The $n \times n$ matrix $\boldsymbol{\mathcal{L}}^{\mathrm{GS}}$ needs to be computed only once, but the vector $\boldsymbol{f}^{\mathrm{GS}}$ depends nonlinearly on $A_1, \ldots, A_n$ and t, and must be calculated in (8) at each time step. This can be quite computationally expensive if the inner product $\langle \cdot, \cdot \rangle$ (e.g., the discrete L_2 inner product) involves the whole computational mesh used in the spatial discretization of the problem. Instead, the evaluation of $\boldsymbol{f}^{\mathrm{GS}}$ can be performed using a limited number of mesh points, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$, considering

$$\langle \boldsymbol{q}_1, \boldsymbol{q}_2 \rangle = \sum_{k=1}^{M} \overline{\boldsymbol{q}}_1(\boldsymbol{x}_k) \cdot \boldsymbol{q}_2(\boldsymbol{x}_k), \tag{9}$$

where the overbar stands for the complex conjugate and $\cdot$ is a natural inner product to multiply two state vectors. Generally, (9) is not a true inner product in the space of all possible spatial distributions of $\boldsymbol{q}$ on the whole computational grid but is generically an inner product in the POD subspace, provided that the selected number of mesh points, M, be somewhat larger than the number of retained POD modes. Furthermore, selection of the set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ should satisfy mild requirements only, as further explained and repeatedly checked by the authors both in the context of the Galerkin projection [23–25] and in related applications [16–19]. Namely, these points should be located in representative (with a rich spatial structure) regions of the expected solutions. If no a priori information is available or the spatial complexity is uniform over the domain (which occurs in the CGLE), an equispaced distribution is usually the simplest choice. Nonuniform spatial complexity is frequent, because of, e.g., viscous boundary layers, which require a nonuniform selection [18, 24]. More sophisticated, somewhat optimal selections are possible via, e.g., the so-called missing point estimation [32] and hyper-reduction [33] methods, which are quite appropriate for preprocessed ROMs. In these, sampling is performed offline and hence does not affect the online

computational cost. In our adaptive ROM, instead, optimal sampling should be performed online at each updating event to adapt the recalculated POD modes. Thus, care must be taken to avoid that the optimization process offsets (at least partially) the advantage of using (9).

2.2. *Computation of POD modes and singular values*

A POD basis can be obtained by the *method of snapshots* [21], which consists in applying POD to a set of numerically calculated (using the NS) spatial portraits of $\boldsymbol{q}$ at N (possibly not uniformly distributed) values of time, namely

$$\boldsymbol{q}_1 = \boldsymbol{q}(t_1),\ \boldsymbol{q}_2 = \boldsymbol{q}(t_2),\ \ldots,\ \boldsymbol{q}_N = \boldsymbol{q}(t_N). \tag{10}$$

In principle, the snapshots (10) should be representative of the actual dynamics that is to be simulated, but POD will also be applied in Section 2.3 replacing the snapshots (10) by sets of weighted modes. The POD modes, $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots$, are calculated in terms of the snapshots as

$$\boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{S}}\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{D}}^{-1}, \quad \text{with } \boldsymbol{\mathcal{Q}} = [\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_m] \text{ and } \boldsymbol{\mathcal{S}} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N]. \tag{11}$$

Here, $\boldsymbol{\mathcal{D}}$ is the diagonal matrix formed by the strictly positive singular values, $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_m > 0$. These and the $N \times m$ matrix $\boldsymbol{\mathcal{A}}$ are given by

$$\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{D}}^2, \quad \text{with } \boldsymbol{\mathcal{R}} = (R_{ij}) \text{ and } R_{ij} = \langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle. \tag{12}$$

The (Hermitian, positive semidefinite) matrix $\boldsymbol{\mathcal{R}}$ is known as the covariance matrix and is calculated here using the inner product (9), meaning that the POD modes are orthonormal with this inner product. The appropriate number of POD modes (say, n) to be retained for a prescribed accuracy is determined by means of the relative RMS mode truncation error (RRMSE) estimate

$$\mathrm{RRMSE}_n^N = \sqrt{\sum_{j=n+1}^{N} (\sigma_j)^2 \Big/ \sum_{j=1}^{N} (\sigma_j)^2}. \tag{13}$$

This means that the truncated expansion (5) provides a good approximation of the snapshots retaining a small number of POD modes if the associated singular values decay fast, which depends on the intrinsic dynamics of the system and the selection of the snapshots.

Using (13), the number of retained POD modes n to ensure an approximation within accuracy ε could be in principle selected such that $\mathrm{RRMSE}_n^N < \varepsilon$. Nevertheless, in order to anticipate drifts in the system, a larger number of modes will be retained. In addition, an even larger number of modes, $n_1 > n$, will be retained in the GS (8) to monitor the mode truncation error. In fact, n and n_1 will be selected such that

$$\mathrm{RRMSE}_n^N < \varepsilon_1 \equiv \varepsilon/k, \quad \mathrm{RRMSE}_{n_1}^N < \varepsilon_1/k, \tag{14}$$

where the RRMSE is as defined in (13) and k (equal to, say, 100) is a flexibly tunable parameter [23, 24]. Tuning the parameter k is facilitated by the use of the scaled relative error (13), which makes the selection of k somewhat independent of the order of magnitude of the snapshots. Thus, the set of n_1 modes that are retained in the GS is divided into the first n *primary modes* that are used to construct the low-dimensional approximation and the remaining $n_1 - n$ *secondary modes* that permit monitoring the mode truncation error.

Concerning the calculation of POD modes and singular values, the usual method quoted in the literature consists in solving the eigenvalue problem (12) using a standard routine (e.g., the MATLAB command *eig*). Unfortunately, POD modes are calculated with an accuracy that scales as

$$(\sigma_1)^2\ 10^{-16}/(\sigma_j)^2, \tag{15}$$

where σ_j are the associated singular values, meaning that those POD modes associated with singular values such that $\sigma_j/\sigma_1 < 10^{-8}$ are useless. This dramatic loss of accuracy is due to the amplification of round-off errors in the product of snapshots that is involved in the definition of $\mathcal{R}$ in (12). This is well known in the numerical linear algebra literature but is generally not mentioned in papers dealing with POD-based reduced-order modeling, perhaps because POD modes associated with very small singular values are not retained in these works. The problematic product of snapshots is avoided noting that $\mathcal{R} = \overline{\mathcal{S}}_{\text{red}}^{\top}\mathcal{S}_{\text{red}}$, where the reduced snapshots matrix $\mathcal{S}_{\text{red}}$ contains in its columns the values of the snapshots at the mesh points selected in (9), and rewriting the eigenvalue problem (12) as $\mathcal{S}_{\text{red}}\mathcal{A} = \mathcal{B}\mathcal{D}$, $\overline{\mathcal{S}}_{\text{red}}^{\top}\mathcal{B} = \mathcal{A}\mathcal{D}$. This enlarged eigenvalue problem is precisely what the MATLAB command *svd* solves (quite efficiently) to perform SVD and essentially doubles the accuracy of the standard computation. Namely, the accuracy of the resulting POD modes scales as (cf. (15))

$$\sigma_1\ 10^{-16}/\sigma_j. \tag{16}$$

The modified method is analytically equivalent to the standard method but much more computationally efficient, and proceeds as follows. The MATLAB command *svd* is applied to $\mathcal{S}_{\text{red}}$, retaining only the strictly positive singular values, which provides two (generally rectangular) matrices U and V such that $\overline{U}^{\top}U =$ unit matrix and $\overline{V}^{\top}V =$ unit matrix, and a (square) diagonal matrix Σ such that $\mathcal{S}_{\text{red}} = U\Sigma\overline{V}^{\top}$. Substituting these into the definition of $\mathcal{R}$ in (12) shows that the matrices $\mathcal{A} = V$ and $\mathcal{D} = \Sigma$ solve the indicated eigenvalue problem. A further substitution of $\mathcal{A}$ into (11) yields the POD modes (at all mesh points), which completes the computation. However, multiplying by $\mathcal{D}^{-1}$ in (11) involves (small but) non-negligible round-off errors due to the small magnitude of the high-order singular values. Thus, the resulting POD modes could be not strictly orthogonal but slightly oblique. This is improved by applying the Gram–Schmidt method (with the inner product (9)) to the POD modes obtained as explained earlier. This is carried out considering the counterpart of (11) for the mesh points selected in the inner product (9), namely $\mathcal{Q}_{\text{red}} = \mathcal{S}_{\text{red}}\mathcal{A}\mathcal{D}^{-1}$. Applying a QR decomposition (with the MATLAB command *qr*) to this matrix yields $\mathcal{Q}_{\text{red}} = \mathcal{Q}_{\text{red}}^{\text{corr}}\mathcal{R}$, where $\mathcal{R}$ is a square, upper triangular, nonsingular matrix and the (orthonormal) columns of $\mathcal{Q}_{\text{red}}^{\text{corr}}$ are the corrected POD modes at the mesh points selected in (9). It follows that $\mathcal{Q}_{\text{red}}^{\text{corr}} = \mathcal{S}_{\text{red}}\mathcal{A}\mathcal{D}^{-1}\mathcal{R}^{-1}$, which readily implies that the re-orthonormalized modes at all mesh points are the columns of the matrix $\mathcal{Q}_{\text{corr}} = \mathcal{S}\mathcal{A}\mathcal{D}^{-1}\mathcal{R}^{-1}$. This re-orthonormalization of the POD modes corrects them only slightly and does not modify the scaling (16).

The advantages of the new formulation are now illustrated with an example that is easily checked by the reader using MATLAB. In order to emphasize that the loss of accuracy of the standard method has nothing to do with the non-standard inner product (9), the POD modes are calculated in the example using the standard inner product. Let the matrix $\mathcal{D}$ be the 20×20 diagonal matrix whose diagonal elements are $\sigma_j = 10^{-j}$ for $j = 1, \ldots, 20$, and let the snapshots matrix be defined as

$$\mathcal{S} = \mathcal{S}_0 + 10^{-16}\mathcal{C}, \quad \text{with } \mathcal{S}_0 = \mathcal{D}\mathcal{B}, \tag{17}$$

where $\mathcal{B}$ is unitary and $\mathcal{C}$ is a random matrix whose elements are of the order of one. Thus, $10^{-16}\mathcal{C}$ mimics round-off errors in the snapshots matrix. If these are ignored, then the (exact) singular values of the columns of $\mathcal{S}_0$ are $\sigma_1, \sigma_2, \ldots$, with the associated POD modes given by $\mathcal{Q}_1 = (1, 0, \ldots, 0)^{\top}$, $\mathcal{Q}_2 = (0, 1, \ldots, 0)^{\top}, \ldots$. Applying both the standard method and the modified method described earlier to the snapshots matrix $\mathcal{S}$ defined in (17) (and repeating calculations 10 times by randomly choosing the matrix $\mathcal{C}$) provides the singular values σ_j plotted in Figure 2 (left). The standard method does not yield (real) singular values with $j > 16$ because, owing to round-off errors, the MATLAB command *eig* applied to $\mathcal{R}$ produces negative values of $(\sigma_j)^2$. As can be seen in Figure 2, the singular values computed by the standard and modified methods saturate at $\sigma_j/\sigma_1 \sim 10^{-8}$ and $\sigma_j/\sigma_1 \sim 10^{-16}$, respectively, and the POD modes accuracy scales as anticipated in (15) and (16), respectively.

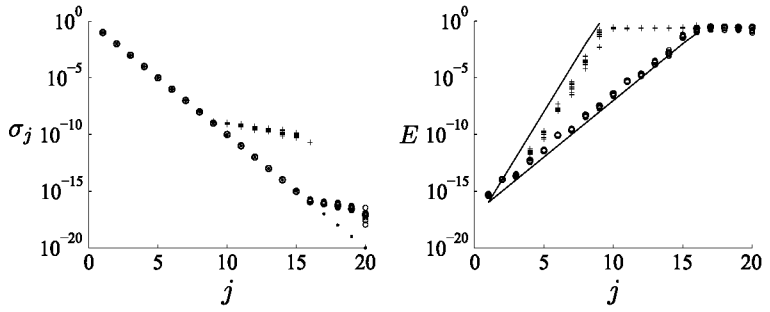

Figure 2. Left: exact (dots) singular values of the columns of the matrix $\mathcal{S}_0$ in (17) and their counterparts for $\mathcal{S}$ computed by the standard (crosses) and modified (plain circles) methods. Right: root mean square errors in the calculation of the POD modes of $\mathcal{S}$ with the standard (crosses) and modified (plain circles) methods; the estimates appearing in (15) and (16) are also plotted with solid lines.

2.3. Updating the POD subspace

As further explained in Section 3.1, the POD modes need to be updated at some particular instants as time proceeds. Updating is performed as follows. First, some *new* modes $\boldsymbol{Q}_j$ are calculated by applying POD to a set of few snapshots that are obtained by running the NS in a new I_{NS} interval, starting from the last state approximated by the GS. Then, these new modes are mixed with the *old* (i.e., previously used) modes $\widehat{\boldsymbol{Q}}_j$ by applying POD to the set of vectors

$$\widehat{\nu}_1 \widehat{\boldsymbol{Q}}_1, \ldots, \widehat{\nu}_{N_{\mathrm{old}}} \widehat{\boldsymbol{Q}}_{N_{\mathrm{old}}}, \nu_1 \boldsymbol{Q}_1, \ldots, \nu_{N_{\mathrm{new}}} \boldsymbol{Q}_{N_{\mathrm{new}}}, \tag{18}$$

where the number N_{new} of new modes depends on the new snapshots (which are automatically selected by the method as explained at the end of Section 3.1) and the weights $\widehat{\nu}_j$ and ν_j are defined as

$$\widehat{\nu}_j = \min \left\{ \frac{\widehat{\sigma}_j}{\sqrt{\sum_{k=1}^{N_{\mathrm{old}}} (\widehat{\sigma}_k)^2}}, \frac{\langle |A_j| \rangle}{\sqrt{\sum_{k=1}^{N_{\mathrm{old}}} \langle |A_k| \rangle^2}} \right\}, \quad \nu_j = \frac{\sigma_j}{\sqrt{\sum_{k=1}^{N_{\mathrm{new}}} (\sigma_k)^2}}. \tag{19}$$

Here, $\widehat{\sigma}_j$ and σ_j are the singular values associated with $\widehat{\boldsymbol{Q}}_j$ and $\boldsymbol{Q}_j$, respectively, and $\langle |A_j| \rangle$ is the temporal mean value of $|A_j|$ provided by the GS over the last I_{GS} interval. Thus, the weights $\widehat{\nu}_j$ eliminate those (no longer necessary) modes whose average energy considerably decreased in the last I_{GS} interval, but do not enhance the modes that excessively increased, which could be due to possible mode truncation instabilities (Section 3.1).

3. NEW INGREDIENTS AND THE NEW ADAPTIVE LOW-DIMENSIONAL MODELING

3.1. Residual-assisted time integration

Monitoring the ROM time integration is necessary to decide when the approximation is no longer good enough, and updating the POD subspace is necessary. In principle, updating must be performed when either (i) the system dynamics has (slightly) departed from the subspace of the phase space spanned by the current POD modes, which should be slightly rotated/enlarged, or (ii) the approximation exhibits the mode truncation instability.

Departure from the actual dynamics is detected when the following condition is no longer satisfied:

$$E_n^{n_1} = \sqrt{\sum_{j=n+1}^{n_1} |A_j|^2 \Big/ \sum_{j=1}^{n_1} |A_j|^2} < \varepsilon, \tag{20}$$

where $A_1, \ldots, A_{n_1}$ are the amplitudes of the modes retained in the GS, while ε and the mode numbers n and n_1 are as defined in Equation (14). As further explained in [23, 24], this is a relative truncation error estimate, namely $E_n^{n_1} \simeq ||\boldsymbol{q}_{\mathrm{GS}}^{n_1} - \boldsymbol{q}_{\mathrm{GS}}^{n}|| \,/\, ||\boldsymbol{q}_{\mathrm{GS}}^{n_1}||$, where $\boldsymbol{q}_{\mathrm{GS}}^{n_1}$ and $\boldsymbol{q}_{\mathrm{GS}}^{n}$ are as given in (5); similar error estimates are used in the context of spectral methods for de-aliasing [34].

Mode truncation instabilities are monitored in [23, 24] by testing consistency between the current GS and a second instrumental GS (retaining $n_2 > n_1$ POD modes), which is simultaneously integrated. The method provided very good results but required to simultaneously integrate two GSs, which increases the required computational cost. Here, instead, truncation instabilities are monitored using a residual of the governing equations. Residuals have been seen to play a role in constructing dissipative models for PDEs [35], stabilizing POD-based ROMs [9], and providing error bounds for ROMs of parametrized parabolic problems [36]. In the spirit of the present paper, we consider the normalized residual

$$E_{\mathrm{res}}^{n_1} = \frac{\| \frac{2}{\Delta t} \left(\tilde{\boldsymbol{q}}_{\mathrm{GS}}^{n_1,j+1} - \tilde{\boldsymbol{q}}_{\mathrm{GS}}^{n_1,j} \right) - \boldsymbol{\mathcal{L}} \left(\tilde{\boldsymbol{q}}_{\mathrm{GS}}^{n_1,j+1} + \tilde{\boldsymbol{q}}_{\mathrm{GS}}^{n_1,j} \right) - 3\boldsymbol{f}_j + \boldsymbol{f}_{j-1} \|}{2\| \boldsymbol{\mathcal{L}} \tilde{\boldsymbol{q}}_{\mathrm{GS}}^{n_1,j} \| + 2\| \boldsymbol{f}_{j-1} \|} \tag{21}$$

where $\tilde{\boldsymbol{q}}_{\mathrm{GS}}^{n_1,j}$ is the spatially discretized GS state in (5), retaining n_1 POD modes at the temporal step j, $\boldsymbol{f}_j = \boldsymbol{f}(\tilde{\boldsymbol{q}}_{\mathrm{GS}}^{n_1,j}, t_j)$, and the norm $\| \cdot \|$ is associated with the inner product (9) (thus based on a limited number of mesh points), which makes the calculation computationally inexpensive. This normalized residual is required to be such that

$$E_{\mathrm{res}}^{n_1} < \varepsilon / k_1 \tag{22}$$

where ε is the prescribed error bound as in Equation (14) and the factor k_1 is tunable. As carried out in (14), the selection of k_1 is facilitated by the scaling of the normalized residual (21), which makes this estimate somewhat independent of the order of magnitude of the various terms in the right-hand side of Equation (2).

Using these arguments, we perform an updating of the POD subspace when either of the two conditions (20) and (22) fails. In this case, a new POD subspace is calculated as explained in Section 2.3, which requires computing new snapshots in a new I_{NS} interval. The length of this I_{NS}, δ_{NS}, is chosen dynamically by requiring that the length δ_{GS} of the subsequent I_{GS} interval be at least equal to T_0, defined as in Equation (4); otherwise, δ_{NS} is iteratively enlarged according to the formula

$$\delta_{\mathrm{NS,new}} = \delta_{\mathrm{NS,old}} + \max \{ \delta_{\mathrm{NS,min}}, (T_0 - \delta_{\mathrm{GS}}) \delta_{\mathrm{NS,old}} / T_0 \} \tag{23}$$

until the length of the subsequent I_{GS} interval is appropriate, where the minimum length $\delta_{\mathrm{NS,min}}$ is tunable. This iterative formula is used to both update the POD subspace and select the length of the initial I_{NS} interval. The iterative process is initiated with $\delta_{\mathrm{NS}} = \tau_0$ in the first I_{NS} interval and $\delta_{\mathrm{NS}} = \tau_1$ in the remaining I_{NS} intervals, where τ_0 and τ_1 are tunable parameters.

3.2. Mode libraries

The described strategy to update on demand the POD basis turns out to be quite efficient, because very few additional snapshots are generally sufficient. Thus, the major computational effort (associated with the snapshots calculation by the NS) is concentrated in the first I_{NS} interval, which, in principle, must be fairly large to capture representative information on the initial dynamics.

A key observation to decrease the initial computational effort relies on the generic nature of POD modes for a class of dissipative systems. For these, the POD subspace depends only weakly on the particular values of the parameters, which has been extensively discussed and exploited in [25, 26] in the context of bifurcation problems. This idea suggests that the low-dimensional POD subspace

computed for some fixed values of the parameters contains the attractors and (a significant part of) the most relevant transients for other parameter values. The latter statement is the core of a strategy to reduce the length of the first I_{NS} interval. The method assumes that one has a *library* of modes defined as explained in the following. These modes may (possibly) not contain a sufficiently good approximation of the actual initial dynamics of the system. Nevertheless, they can be (i) initially selected to span some of the dynamically relevant directions and (ii) then enriched, with a small computational cost, by adding few new modes. This is carried out by the process used in Section 2.3 to update the POD subspace, applying POD to the vectors in (18) but substituting the old modes, $\widehat{v}_j \widehat{\boldsymbol{Q}}_j$, by the modes in the library, weighted as detailed in the following. Iterating as defined in (23), a few additional snapshots generally succeed in rotating as needed the modes in the library, which results in a flexible and efficient way of starting up the GS time integration and shortening the first I_{NS} interval.

Suitable mode libraries are as follows:

- The set of modes resulting from applying POD to a set of generic functions, such as trigonometric functions or orthogonal polynomials in 1D problems and the eigenvectors of an appropriate operator (such as the Laplacian) with appropriate boundary conditions for complex geometries in higher dimensions. A more customized generic library consists in the eigenvectors of the linear operator $\boldsymbol{\mathcal{L}}$ appearing in the governing equation (1) if this operator is self-adjoint or the right SVD modes of this operator otherwise. In generic libraries, the weights $\widehat{v}_j$ appearing in (19) to enrich the modes must be selected 'ad hoc'.
- The POD subspace used in the last I_{GS} interval in a previous simulation (for different parameter values) with the adaptive ROM described in Section 3.3. These modes are enriched using (19) with the weights $\widehat{v}_j$ chosen as

$$\widehat{v}_j = \frac{\sigma_j}{\sqrt{\sum_k (\sigma_k)^2}}, \tag{24}$$

  where σ_j are the singular values in the POD that provided the modes in the library. The selected library is then used for generic parameter values; see Table III.
- The result of mixing two libraries by applying POD to the joint set of modes, weighted as explained in the previous items. Enrichment is performed as in the last item, selecting the weights $\widehat{v}_j$ according to (24).

It is important to stress that more sophisticated mode libraries [37, 38] could be necessary in some specific problems. The aforementioned mode libraries pretend to be neither optimal nor general but are considered precisely because (i) they are well suited to the adaptive ROM that is being developed and (ii) their simplicity makes the construction computationally inexpensive.

3.3. Summary of the new method

The ideas outlined in Section 2 and the new ingredients developed in Sections 3.1 and 3.2 can be suitably set together in a robust, adaptive method to approximate a solution of (1) in a time span $0 < t \leqslant T$. This goal is achieved by combining a standard NS and a GS, defined in Equations (2) and (8), respectively, in interspersed time intervals, I_{NS} and I_{GS}, respectively, as illustrated in Figure 1. The *new adaptive ROM* that is proposed in this paper proceeds in the following steps:

(i) A preliminary selection is made of the various tunable parameters and a suitable mode library, as explained in Sections 2, 3.1, and 3.2.
(ii) The simulation begins by choosing the initial set of POD modes either by applying POD to a set of snapshots calculated by the NS and iteratively proceeding using (23) or enriching a mode library, as explained in Section 3.2. An initial GS is constructed as explained in Section 2.1 and is used in the first I_{GS} interval.
(iii) Integration of the GS proceeds provided that conditions (20) and (22) are satisfied. Otherwise, the POD subspace is updated as explained in Section 2.3, and a new GS is constructed and integrated in the subsequent I_{GS} interval.

(iv) Step (iii) is applied again until the final value $t = T$ is reached.

The Galerkin projection and POD are always performed using the inner product (9), and POD is applied using the modified method described in Section 2.2. Selection of the tunable parameters is quite flexible, attending to natural, mild requirements only. For instance, the time interval between NS-computed snapshots should be such that they convey non-redundant and significant information on the dynamics (say, few time integration steps with the NS).

3.4. A heuristic justification of the method and some additional comments

The adaptive method developed earlier aims at strongly decreasing the computational cost of unsteady simulations while preserving accuracy. The method relies on several ingredients already used by us in former versions of the method (the selection of the snapshots and the retained number of POD modes from the past dynamics, updating the POD modes when needed and performing POD and the Galerkin projection using a limited number of mesh points), two ingredients that have been improved (the selection of the timescale T_0 in which the approximation is controlled and the strategy to ensure the accuracy of the approximation based on a residual estimate), and a new ingredient (the use of mode libraries). None of these ingredients is new in the field. What is new in the method is the robust and synergistic combination of the ingredients. A rigorous justification of the whole method would be highly nontrivial and well beyond the scope of the paper. But a heuristic justification is possible based on the following arguments, with some further comments to complete a synthesis of the main ideas:

- Selection of the POD modes in the whole process is made using information (i.e., snapshots calculated by the standard NS) from the past to simulate the near future. The strategy relies on the expectation that the dynamics of the system is approximately contained in a low-dimensional linear subspace of the phase space that depends continuously on time. In other words, POD modes that will be necessary to keep a given accuracy in the near future are already present in the past dynamics as higher-order modes. Some of these are already retained in the current POD subspace because of the scaling factor $k \gg 1$ in (14). But, as time proceeds, some new dimensions of the phase space may be visited by the system that either (i) were not retained using (14) or (ii) were not visited at all in the NS time spans used to calculate the POD subspace. The scenarios (i) and (ii) are detected by the truncation error estimate (20) and the residual estimate (22), respectively. The efficiency of (20) is obvious and that of the residual estimate is justified, noting that, as is well known in numerical analysis, keeping the residual small *over time spans of appropriate finite length* T_0 (to adapt to possible residual amplification) ensures an accuracy consistent with the nature of the simulated trajectories (see the next item). In the present context, T_0 is dynamically selected according to (4).
- Concerning the timescale T_0, it must be kept in mind that, except for some trivial cases, ensuring accuracy over large periods is just not possible. Even in simple periodic dynamics, small errors in the frequency promote deviations from the exact solution that grow algebraically in time. Chaotic dynamics, by their own nature, produce exponential divergence (illustrated in the next section, Figure 3). Therefore, accuracy can only be maintained over a limited time span, which is selected here dynamically, as several times the characteristic timescale of the system defined in (3). This ensures, in particular, that the dynamical systems associated with the GS and the NS are close to each other. By a good approximation of a dynamical system, we mean a good approximation of the associated discrete Poincaré map [39] defined by subsequent intersections with an appropriately defined Poincaré hypersurface. In other words, even though the time-dependent solutions of the NS and the GS will generally diverge as time increases, the periodic, quasi-periodic, or chaotic attractors will be close to each other in the phase space; see [26, Figure 8] for an illustration of this property (using a different, more expensive computation of T_0), showing that even fairly specific details of chaotic attractors are well approximated.

DOI: 10.1002/nme

- The strategy to update the POD modes works so efficiently because (i) it is the old and new POD modes (not the old and new snapshots) that are collected to apply POD, and (ii) these modes are weighted according to their importance in the last I_{GS} interval and the new I_{NS} interval.
- Mode libraries facilitate the construction of the first set of POD modes by appropriately weighting and mixing the modes in the library and the new modes calculated by the NS. If the quality of the library is not good, the process itself will adaptively discard these modes, and no significant gain will result. But the examples in the next two sections show that the adaptive strategy is able to take great advantage of both generic libraries and libraries obtained from other generic (namely, not selected on purpose) runs of the method simulating dynamics that are completely different from the dynamics that are being simulated.
- It might be argued that the use of mode libraries is some kind of preprocess, while we claim that the present method is a non-preprocessed ROM. But there are fundamental differences between our ROM and both preprocessed ROMs and other methods in the literature that also use mode libraries, namely

– While the preprocess is fairly expensive in standard preprocessed ROMs, the computational cost of generating the mode libraries is negligible in the present paper. As shown in the applications in the next two sections, our libraries either (i) are generic libraries (such as Fourier modes); (ii) result from former runs of the method for different parameter values that do not need to be selected 'ad hoc'; or (iii) are obtained by mixing libraries of the two former categories. Concerning libraries of type (ii), it must be kept in mind that ROMs are usually required to simulate the system for several parameter values, meaning that libraries of this type are usually available for free.
– Concerning comparison with other methods using mode libraries, the modes in our libraries do not need any sampling method to minimize their number. On the contrary, as shown in the next two sections, a large number of modes (in both generic libraries and libraries resulting from former runs of the method) are selected at the outset; this number will be drastically decreased by the adaptive strategy of our method as time proceeds.

- Using a limited number of points to perform POD and especially to perform the Galerkin projection of the equations, is an essential step to improve the computational efficiency. This is because the computational cost of the GS scales with the product of the number of modes and the number of mesh points used to perform the projection. In the applications in Section 4, the limited number of mesh points is taken as several times the number of modes, and such points are selected equispacedly. Of course, a better selection using appropriate sampling could further increase the computational efficiency, but the next application to the 2D CGLE shows that this generic selection of mesh points already makes the computational cost of the GS negligible compared with that of the standard NS.

4. APPLICATION TO THE COMPLEX GINZBURG–LANDAU EQUATION IN 1D

Let us consider the 1D CGLE

$$\partial_t q = (1 + \mathrm{i}\alpha)\partial_{xx}^2 q + \mu q - (1 + \mathrm{i}\beta)|q|^2 q, \quad \text{with } q = 0 \text{ at } x = 0, 1, \tag{25}$$

which is discretized using centered second-order finite differences in an equispaced grid of 1000 points, using as NS the temporal scheme (2) with a time step $\Delta t = 5 \cdot 10^{-5}$. The number of mesh points has been calibrated by using the NS in a finer mesh with 2000 points and checking that both trajectories remain close to each other in intervals of length $T_0 \sim 0.1$ (see succeeding discussion). This equation is a well-known paradigm of a simple nonlinear equation that exhibits intrinsically complex dynamics [40]. The state variable q is complex, which means that the equation could also be considered as a system of two real equations, while the parameters μ, α, and β are real. The problem (25) is invariant under the $D_1 \times SO(2)$ group, namely under transformations $x \to 1 - x$ and $q \to q\,\mathrm{e}^{\mathrm{i}c}$. In addition, the modulational instability for $\alpha\beta < -1$ (the Newell condition) and

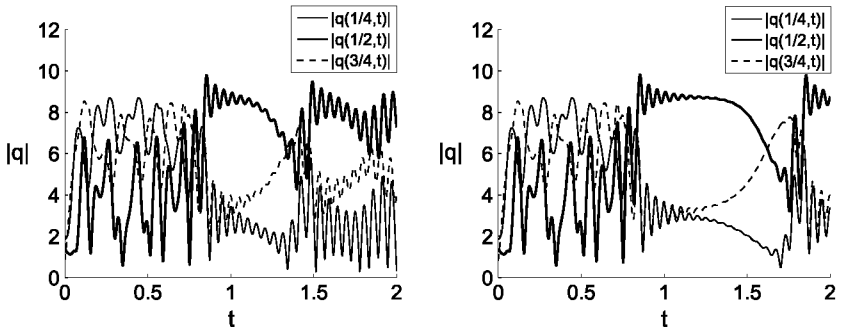

Figure 3. Temporal evolution of $|q|$ in test case 1 using the numerical solver with 1000 (left) and 2000 (right) grid points.

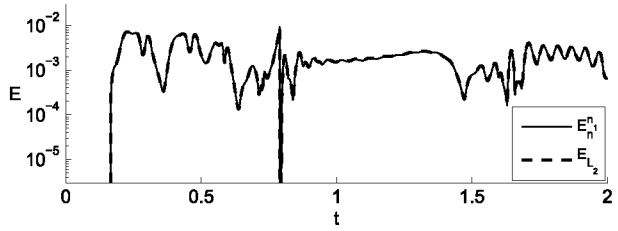

Figure 4. Error estimate $E_n^{n_1}$ and actual error E_{L_2} defined in Equation (29) resulting from the application of the new adaptive reduced-order model without mode libraries to test case 1.

μ larger than a critical value usually leads to chaotic-like behaviors at large time, which makes the CGLE a fairly demanding test problem for the method. The chaotic nature of the solutions is illustrated in Figure 3, which shows that applying the NS to moderately chaotic dynamics with two computational grids produces different results for $t \geqslant 0.75$. Likewise, the adaptive ROM provides solutions that also diverge from the NS solutions for $t \geqslant 0.75$ (compare Figures 3 (left) and 9 (top)), but it approximates well the NS trajectories in time intervals of length T_0, which is selected to adapt to the local dynamics as explained in Section 2.

Three representative test cases will be used to evaluate the performance of the new adaptive ROM:

- *Test case 1* (moderate complexity) for $(\alpha, \mu, \beta) = (1, 80, -2.5)$, considered in Figures 3, 4, and 9.
- *Test case 2* (full chaos) for $(\alpha, \mu, \beta) = (1, 125, -20)$, considered in Figures 5, 6, and 9.
- *Test case 3* (intermittency) for $(\alpha, \mu, \beta) = (0.75, 100, -13)$, considered in Figures 7 and 9.

In order to avoid restriction to an invariant subspace in these three examples, the following non-reflection symmetric initial condition will be used:

$$q(x, 0) = \mathrm{i}\sin(2\pi x) + (1 + \mathrm{i})\sin(3\pi x). \tag{26}$$

The tunable parameters of the adaptive ROM are selected as follows. The error bound is $\varepsilon = 10^{-2}$; the denominators appearing in Equations (14) and (22) are $k = 100$ and $k_1 = 2$, respectively; the initial value of the parameter T_0 defined for subsequent I_{GS} intervals as in Equation (4) is $T_{0,\mathrm{init}} = 0.06$; and the initial values for the iterations mentioned after Equation (23) are $\tau_0 = 0.1$ and $\tau_1 = 0.004$. The inner product (9) is based on 100 equispaced grid points. The performance of the new adaptive ROM will be measured by the *acceleration factor*, which is the ratio of the CPU times (using a desktop PC, with a Intel i7 3.5-GHz microprocessor and 8-GB RAM) required by the NS and the ROM, namely

$$C = \frac{\text{CPU time (NS)}}{\text{CPU time (new adaptive ROM)}}. \tag{27}$$

On the other hand, we may define the *theoretical acceleration factor* as the ratio of the total time span to the total length of the I_{NS} intervals, namely

$$C_{\text{theor}} = \frac{T}{\sum \delta_{\text{NS}}}. \tag{28}$$

Note that this is the asymptotic value of the CPU acceleration factor (27) as the computational cost of the ROM (associated with POD, the Galerkin projection, and the GS integration) becomes negligible compared with the integration by the standard NS.

In order to subsequently check the advantages of the two new ingredients, namely the residual and the mode libraries, the new adaptive ROM without mode libraries is considered first.

4.1. The new adaptive ROM using the residual estimate without mode libraries

Test case 1 is considered in Figure 4, where both the truncation error estimate defined in (20) and the actual (calculated comparing with the NS) error, defined as

$$E_{L_2} = ||\boldsymbol{q}^n_{\text{GS}} - \boldsymbol{q}|| \,/\, ||\boldsymbol{q}^n_{\text{GS}}||, \tag{29}$$

are jointly plotted. For consistency with the definition of $E_n^{n_1}$, the norm $\|\cdot\|$ in (29) is that associated with the inner product (9). Also, consistently with the definition of T_0 in Equation (4), the actual error E_{L_2} is set to zero in the I_{GS} intervals (setting the NS solution equal to the GS solution) at time instants t_j, where t_0 is the beginning of the I_{GS} interval and $t_{j+1} = t_j + T_0$. Recall that, as explained in Section 2, restricting the comparison between E_{L_2} and $E_n^{n_1}$ to time intervals of length T_0 is necessary because of the highly unstable dynamics (illustrated in Figure 3). The ROM requires two

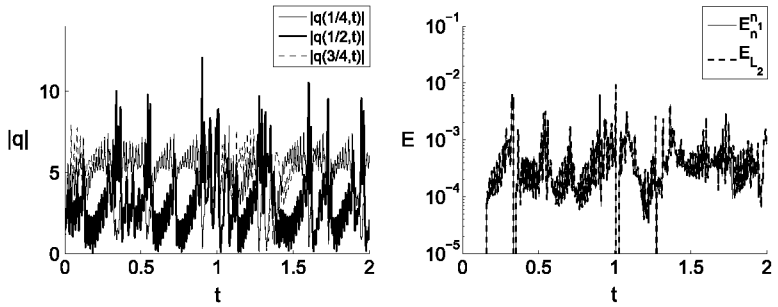

Figure 5. Test case 2: time evolution of $|q|$ (left) and counterpart of Figure 4 (right).

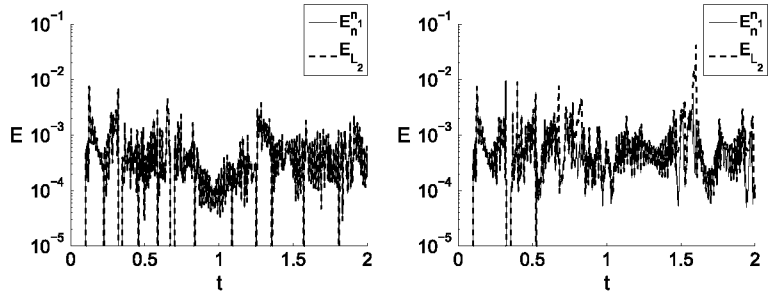

Figure 6. Counterpart of Figure 5 (right) using the former adaptive method developed in [23, 24] (left) and ignoring the residual control (22) (right).

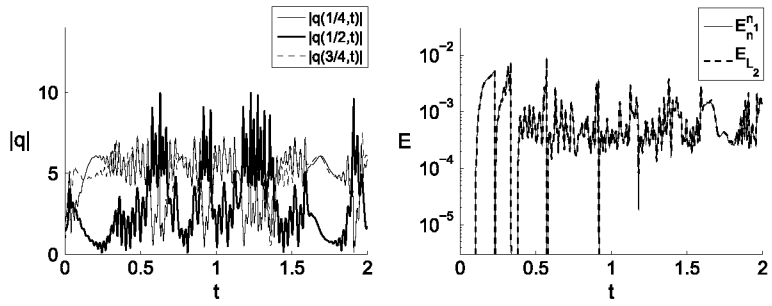

Figure 7. Counterpart of Figure 5 for test case 3.

$I_{\rm NS}$ intervals (identified in Figure 4 because $E_n^{n_1} = E_{L_2} = 0$ in these intervals), an initial interval of length 0.164 to fully calculate the POD subspace, and a 30-time smaller interval near $t = 0.8$ to update the POD modes. In the associated $I_{\rm GS}$ intervals, $0.164 \leqslant t \leqslant 0.791$ and $0.796 \leqslant t \leqslant 2$, the numbers of retained modes defined in Equation (14) and the time span T_0 defined in Equation (4) are $(n, n_1, T_0) = (9, 14, 0.06)$ and $(n, n_1, T_0) = (10, 15, 0.09)$, respectively. As it can be seen in Figure 4, the truncation error estimate and the actual error are plot-indistinguishable in both $I_{\rm GS}$ intervals. The acceleration factor defined in Equation (27) is fairly large, namely $C = 7.06$, but smaller than the theoretical acceleration factor defined in (28), which is $C_{\rm theor} = 11.83$ in the present case. This is because the computational cost of the ROM is not negligible compared with that of the NS in spite of the reduced number of retained modes. The difference between the CPU and the theoretical acceleration factors will be more dramatic when using libraries, as will be shown in the next subsection. Comparison with Figure 3 (left) shows that the POD subspace is updated precisely when the dynamics exhibit a qualitative change, approximately at $t \sim 0.8$. It is remarkable that the POD subspace computed at $t = 0.164$ approximates well the dynamics until $t \sim 0.8$. A closer look at the performance of the method shows that updating is required because the error control (20) fails, meaning that the second control (22) is not necessary in this particular simulation; but this is not always the case, as illustrated in the next example. On the other hand, the previous adaptive ROM with two GSs developed in [23, 24] produces similar results in this example, but the acceleration factor, $C = 3.03$, is smaller than with the present ROM.

Test case 2 is considered in Figure 5, which shows four $I_{\rm NS}$ intervals of lengths 0.158, 0.017, 0.021, and 0.007, and four associated $I_{\rm GS}$ intervals, where the mode numbers $(n, n_1) \sim (23, 33)$ remain almost constant; the time span defined in Equation (4) takes the values $T_0 = 0.06$, 0.074, 0.075, and 0.072. Because of the higher complexity of the dynamics, the numbers of modes n and n_1 are larger, and the acceleration factor ($C = 3.63$) is smaller than in test case 1. Again, the updating occurs precisely when the dynamics exhibit qualitative changes. Comparison with the previous adaptive method in [23, 24] that uses two GSs (Figure 6 (left)) shows that the former method updates the POD subspace more times but requires a similar total length of the $I_{\rm NS}$ intervals and gives a smaller acceleration factor, $C = 1.78$. In order to illustrate the necessity of controlling the mode truncation instability, the case in which the residual control (22) is ignored is considered in Figure 6 (right). Note that for the resulting method, the actual error E_{L_2} exceeds the required accuracy $\varepsilon = 10^{-2}$ at $t \sim 1.6$, while the error estimate is $E_n^{n_1} < \varepsilon$, meaning that the approximation cannot be regarded as acceptable.

Test case 3 is considered in Figure 7, which shows five $I_{\rm NS}$ intervals whose lengths are 0.1, 0.004, 0.046, 0.006, and 0.004. Note that the second and the last two $I_{\rm NS}$ intervals are rather short, meaning that the POD subspace is only slightly rotated at these events. The acceleration factor in this example, $C = 3.70$, is again larger than its counterpart ($C = 1.58$) with the previous adaptive method that uses two GSs.

In summary, the new ingredient consisting in monitoring mode truncation instabilities using a residual is fairly robust and yields accurate results, similar to those provided by the previous adaptive method with two GSs. As expected, not using the second instrumental GS more than doubles the acceleration factor in all cases considered earlier; see Table I.

Table I. Acceleration factors obtained by the former adaptive ROM with two GSs and the present ROMs.

	Test case 1	Test case 2	Test case 3
Former ROM with two GSs	3.03	1.78	1.58
Present ROM with residual but without libraries	7.06	3.63	3.70
Present ROM with residual and library $L_F + L_1$	11.28	13.63	14.60

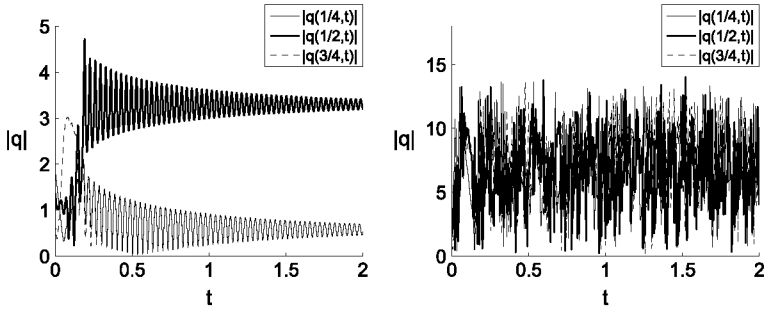

Figure 8. Solution of the complex Ginzburg–Landau equation (25) with initial condition (26) for L_1 : $(\alpha, \mu, \beta) = (-1, 50, 30)$ (left) and L_2 : $(\alpha, \mu, \beta) = (-2, 170, 12)$ (right).

4.2. The new adaptive ROM using the residual estimate and mode libraries

Four representative libraries will be selected in the three ways suggested in Section 3.2, namely a generic library, a fairly simple library and a quite complex library resulting from previous runs of the ROM, and a library resulting from mixing two of these libraries. Specifically,

- L_F is the POD subspace resulting from applying POD to the Fourier modes $a_k \sin(k\pi x)$, with $a_k = 1/k$, for $k = 1, \ldots, 25$; note that the weights a_k are selected to decrease to zero as $k \to \infty$. The resulting modes are weighted as in (24).
- L_1 consists in the 16 POD modes (weighted as explained in Section 3.2) that are used in the last I_{GS} interval when applying the adaptive method in this paper (without libraries) to the case considered in Figure 8 (left). Note that the dynamics are fairly simple after a short transient.
- L_2 is constructed as L_1 but considers the case in Figure 8 (right). Now, the dynamics are chaotic, and the mode library contains 36 modes.
- $L_F + L_1$ is the 27-mode library obtained by mixing up the libraries L_F and L_1 as explained in Section 3.2. Note that a generic library is mixed up with a simple library.

Other combinations are of course possible that are not considered for brevity. In particular, anticipating that the best results are obtained using the library $L_F + L_1$ (Figure 9), we can conclude that combining L_F with a richer mode library produces better results. The acceleration factors produced by the present ROM with the $L_F + L_1$ library for the three test cases 1, 2, and 3 defined earlier are given in Table I, where they are compared with their counterparts using the former ROM with two GSs [23, 24] and the present ROM without mode libraries.

A more detailed description of the effect of the four libraries in the present ROM is given in Figure 9 where, for clarity, the evolution of $|q|$ versus t for the three test cases is recalled in the left plots. The right plots indicate the various I_{NS} intervals that are required in $0 < t \leqslant 2$ by the present adaptive ROM without libraries (labeled 'ROM') and with each one of the four libraries; the number of snapshots calculated by the NS in each I_{NS} interval is also indicated, as is the acceleration factor C. The following remarks are in order in connection with Figure 9:

- The four libraries improve the performance of the adaptive ROM in the three test cases, except for the library L_1 in the more complex test cases 2 and 3. In fact, L_1 generally produces the worst results, even worse than the generic library. This is because the modes in this library have

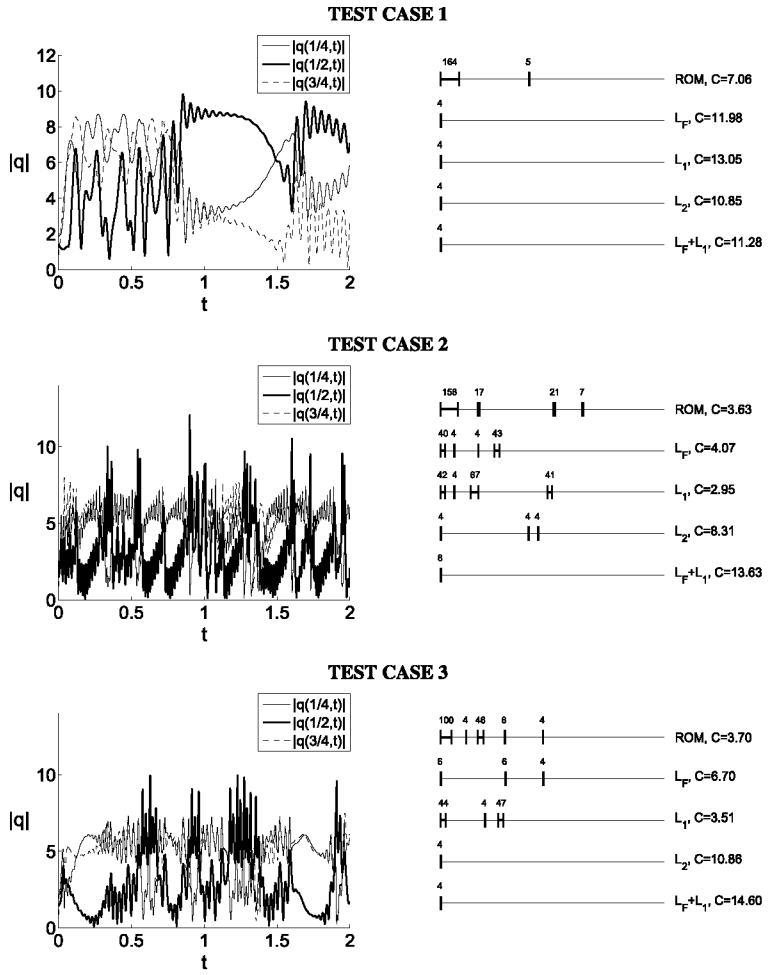

Figure 9. Performance of the adaptive reduced-order model with libraries in test cases 1, 2, and 3.

been calculated in fairly simple transient dynamics that cover a small region of the phase space that is only partially visited in the dynamics of the test cases.

- The library L_2 generally produces fairly good results. This is because this library contains dynamics that are as complex as the dynamics that are being simulated (or even more complex). Such good results are obtained in spite of the fact that both the parameters α, β, and μ, and the resulting dynamics for L_2 are rather different from the ones in the selected test cases.
- The largest acceleration factor is generally produced by the $L_F + L_1$ library, although each of the libraries L_F and L_1 alone does not produce so good results. This suggests that mixing libraries is a good strategy to cover a significant part of the phase space.

The theoretical (28) and CPU (27) acceleration factors for the three selected cases are condensed in Table II, where it can be seen that they are far from each other in most of the cases. We will see in the next section that both acceleration factors become comparable when considering the CGLE in two space dimensions.

Finally, in order to further check the robustness of the method, the parameters α, μ, and β are randomly generated 30 times, obtaining the results displayed in Table III for the number of I_{NS} intervals, the number of calculated snapshots, and the acceleration factor. Note that the main conclusions for the three test problems are roughly confirmed. In particular, the libraries L_2 and $L_F + L_1$ clearly outperform both the 'basic' ROM and the remaining libraries. The library L_F slightly improves the 'basic' ROM; L_1, instead, does not statistically improve it. This is because in spite of having selected

Table II. Theoretical and CPU acceleration factors when applying the new adaptive ROM without libraries (ROM) and the method with the indicated libraries for the three selected cases.

		ROM	L_F	L_1	L_2	$L_F + F_1$
Test case 1	C_{theor}	11.83	500	500	500	500
	C	7.06	11.98	13.05	10.85	11.28
Test case 2	C_{theor}	9.85	21.05	12.98	166.66	500
	C	3.63	4.07	2.95	8.31	13.63
Test case 3	C_{theor}	12.50	125	21.05	500	500
	C	3.70	6.70	3.51	10.86	14.16

Table III. Minimum, maximum, and mean values of the required number of I_{NS} intervals (NSI), snapshots (S), and the acceleration factor (C), resulting from integrating the CGLE in the time interval $0 < t \leqslant 2$ for 30 randomly generated parameter values in the ranges $1 < \alpha < 3$, $75 < \mu < 125$, and $-15 < \beta < -5$.

	NSI			S			C		
Library	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ROM	1	6	2.10	100	324	136.53	2.75	9.84	6.18
L_F	1	4	1.66	4	90	22.40	2.77	11.66	7.85
L_1	1	6	2.26	40	312	80.76	2.38	7.48	4.88
L_2	1	2	1.03	4	8	4.10	6.30	11.67	10.57
$L_F + L_1$	1	2	1.03	4	8	4.26	10.45	13.49	11.40

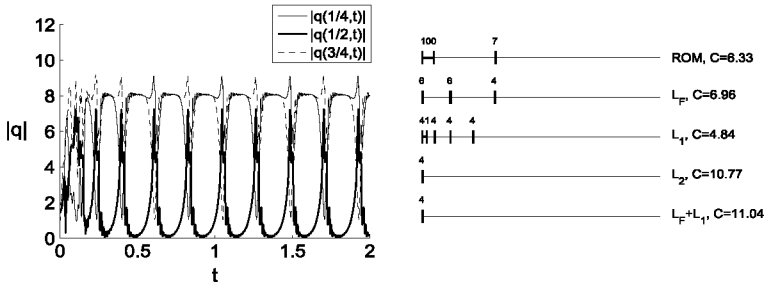

Figure 10. Counterpart of Figure 9 for a representative case (among those considered in Table III) exhibiting simple dynamics: $(\alpha, \mu, \beta) = (2.23, 104.3, -11.34)$.

pretty large values of μ and modulationally unstable cases ($\alpha\beta < -1$), the CGLE shows very simple dynamics for a significant part of this parameter range. It is precisely in these simple cases that the libraries L_F and L_1 do not generally improve the 'basic' ROM. A representative example is shown in Figure 10, in which the trajectory converges very quickly to a quasi-periodic attractor.

In summary, mode libraries generally improve the computational efficiency of the adaptive ROM, especially in complex dynamics when such libraries are constructed either from previous runs of the adaptive ROM for complex solutions or by mixing up simple mode libraries.

5. APPLICATION TO THE COMPLEX GINZBURG–LANDAU EQUATION IN 2D

Let us now consider the 2D CGLE in the unit square,

$$\partial_t q = (1 + \mathrm{i}\alpha)\left(\partial^2_{xx} q + \partial^2_{yy} q\right) + \mu q - (1 + \mathrm{i}\beta)|q|^2 q \quad \text{in } 0 < x < 1, \quad 0 < y < 1, \tag{30}$$

with the homogeneous Dirichlet boundary conditions,

$$q(0,y,t) = q(1,y,t) = q(x,0,t) = q(x,1,t) = 0. \tag{31}$$

This problem is invariant under the $D_4 \times SO(2)$ group generated by the actions $x \to -x$, $y \to -y$, $x \leftrightarrow y$, and $q \to q\mathrm{e}^{\mathrm{i}c}$. As in the 1D case, we avoid restriction to an invariant subspace by imposing a non-symmetric initial condition, namely (Figure 11 (left))

$$q(x,y,0) = (1+7\mathrm{i})(x-3y)\sin(2\pi x)\sin(\pi y) + (2+\mathrm{i})(2x+y)\sin(\pi x)\cos(\pi(1+2y)/2). \tag{32}$$

The spatial discretization is performed using centered second-order finite differences in an equispaced 250×250 grid, which is somewhat coarser than in the 1D case to somewhat limit the computational cost. As in 1D, the grid was calibrated by using the NS in a finer mesh with 1000×1000 points and checking that both trajectories remain close to each other in intervals of length T_0 for the test case considered in the following. For other test cases resulting from different sets of parameters exhibiting more complex dynamics (for instance, for the 2D counterparts of test cases 2 and 3 in 1D), the spatial grid should be finer. Temporal discretization is made using the Crank–Nicolson plus Adams–Bashforth scheme (2), with the same time step as in 1D.

In order to illustrate the robustness of the method, the tunable parameters are the same as in the 1D case except for $T_{0,\mathrm{init}}$ that after some calibration, is taken now three times smaller; the inner product (9) is now based on 16×16 equispaced mesh points.

For brevity, we only consider one test case, for the parameters $(\alpha, \mu, \beta) = (1, 80, -2.5)$ (the same as in test case 1 in Section 4), considered in Figure 12. The left plot shows the temporal evolution at five representative points, and the right plot is the counterpart of Figure 4. As can be seen, the ROM requires six I_{NS} intervals. The numbers of modes n and n_1 vary in the ranges $21 \leqslant n \leqslant 34$ and $33 \leqslant n_1 \leqslant 64$; a comparison with their counterparts in 1D for test case 1 ($9 \leqslant n \leqslant 10$ and $14 \leqslant n_1 \leqslant 15$) shows that these numbers grow algebraically as the spatial dimension is increased,

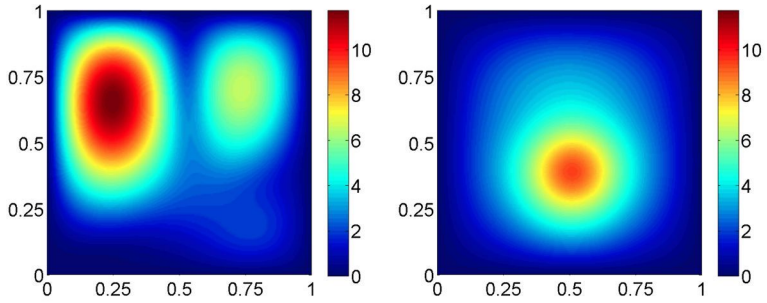

Figure 11. Color maps of $|q|$ for the initial condition (32) (left) and the snapshot at the end of the first I_{NS} interval ($t = 0.34$) in Figure 12 (right).

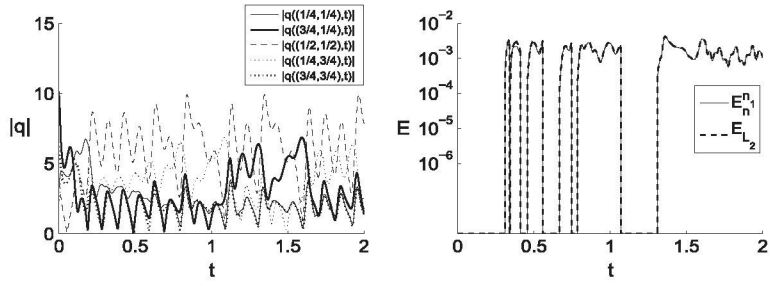

Figure 12. Left: temporal evolution of $|q|$. Right: error estimate $E_n^{n_1}$ and actual error E_{L_2} defined in (29) resulting from the application of the new adaptive reduced-order model without mode libraries to the test case.

as anticipated. The CPU and theoretical acceleration factors defined in Equations (27) and (28), respectively, are fairly close to each other, namely $C = 2.61$ and $C_{\text{theor}} = 2.66$, respectively. This will occur in all cases considered in this section and is because the computational cost of integrating the GS is much smaller than its counterpart for the full NS. The slight difference between C and C_{theor} is due to the computational effort of the updating steps.

In order to illustrate how the method adapts 'on the fly' the POD modes and the aforementioned continuity of the POD subspace, the evolution of the first four POD modes is given in Figure 13. As can be seen, the qualitative shape of the first mode remains unchanged, but the remaining modes show a more drastic topological evolution. On the other hand, Figure 12 shows that the second I_{NS} interval covers a very small time span (0.004), and thus, it does not produce drastic modifications in the first four modes; of course, higher-order modes (not plotted here) do change more drastically. As expected, corrections are more and more drastic for higher and higher modes. As time proceeds, the method produces strong modifications even in the first few POD modes, as comparison of the first and last rows shows.

Let us test now the new adaptive ROM using the residual estimate and mode libraries. The following libraries will be considered:

- ℓ_{F1} is the library resulting from applying POD to the Fourier modes

$$\frac{1}{\sqrt{n^2+m^2}} \sin(\pi n x) \sin(\pi m y), \tag{33}$$

for $\sqrt{n^2+m^2} < 11$. The resulting 83 modes are weighted as in (24).

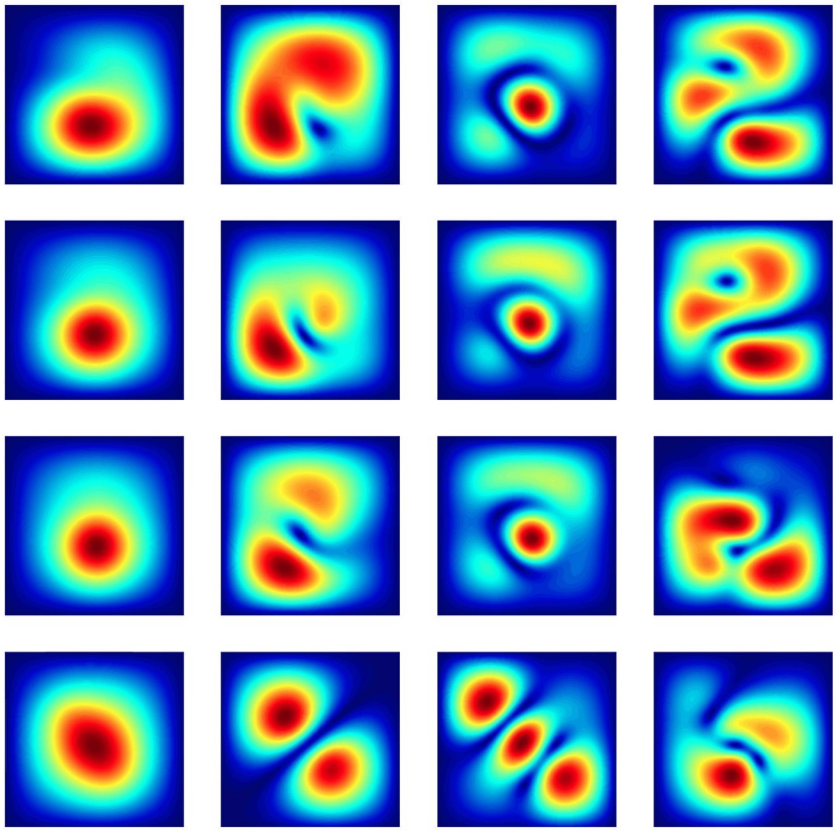

Figure 13. Color maps of the first four POD modes selected by the method in the first, second, third, and sixth I_{GS} intervals (first, second, third, and fourth rows, respectively).

- ℓ_{F2} is the counterpart of ℓ_{F1}, considering the Fourier modes (33) for $\sqrt{n^2+m^2} < 13$. The number of modes in ℓ_{F2} is 117.
- ℓ_1 is the counterpart of L_1 for the 2D case, which contains the 40 POD modes that were used in the last $I_{\rm GS}$ interval when the adaptive method (without libraries) was applied to the CGLE for the parameters $(\alpha, \mu, \beta) = (-1, 50, 30)$. In Figure 14, we represent the modulus of the solution at five representative points (counterpart of Figure 8 (left)).
- $\ell_{F1} + \ell_1$ is a library with 100 modes, the result of mixing up the libraries ℓ_{F1} and ℓ_1 as in Section 3.2.
- $\ell_{F2} + \ell_1$ is obtained by mixing up the libraries ℓ_{F2} and ℓ_1. It contains 130 modes.

The performance of the method when using libraries is summarized in Table IV, which shows the theoretical and CPU acceleration factors defined in (28) and (27), respectively. Note that both acceleration factors are now comparable (a main overall difference with the 1D case, compare with Table II) and that all libraries improve the performance of the method to an impressive extent in all cases except for the simplest library ℓ_1. This is consistent with the similar result in 1D and owing to the same reason. A more detailed account of the performance of the method is given in Figure 15 (the counterpart of Figure 9), which confirms the main conclusions extracted in 1D. In particular, it is remarkable that both libraries ℓ_{F1} and ℓ_{F2}, which are generic libraries, highly outperform

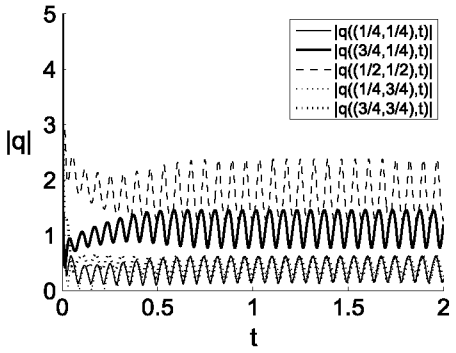

Figure 14. Temporal evolution of $|q|$ for the parameter set $(\alpha, \mu, \beta) = (-1, 50, 30)$ (library ℓ_1).

Table IV. Theoretical and CPU acceleration factors when applying the new adaptive ROM without libraries (ROM) and the method with the indicated libraries.

	ROM	ℓ_{F1}	ℓ_{F2}	ℓ_1	$\ell_{F1}+\ell_1$	$\ell_{F2}+\ell_1$
$C_{\rm theor}$	2.66	31.74	100	3.13	40.81	500
C	2.61	26.91	90.96	3.04	35.42	357.60

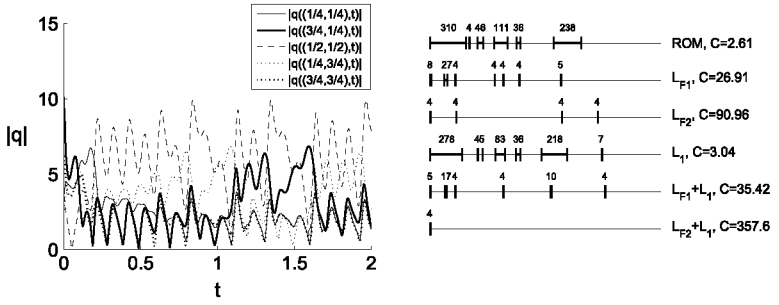

Figure 15. Performance of the adaptive reduced-order model with libraries in the test case $(\alpha, \mu, \beta) = (1, 80, -2.5)$.

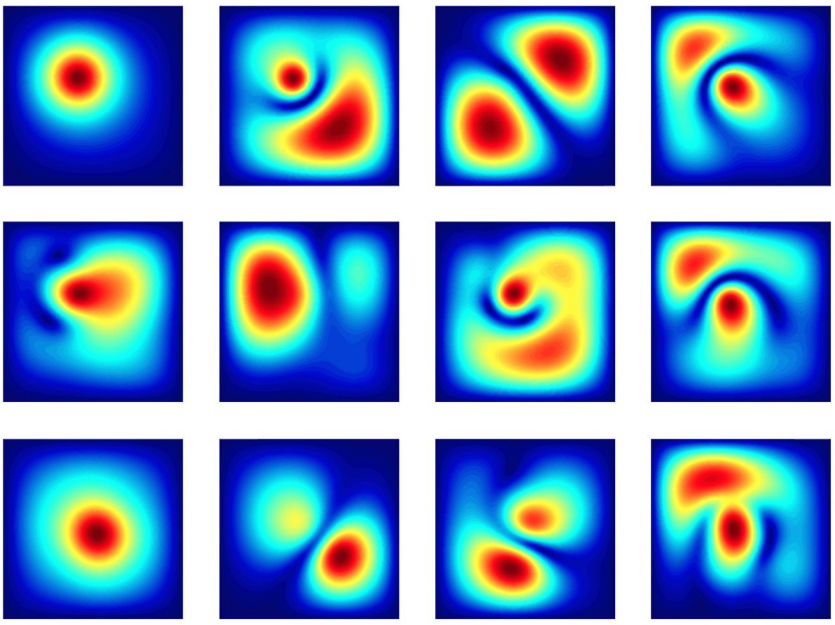

Figure 16. First four modes in the library $\ell_{F1} + \ell_1$ (first row), the POD subspace in the first I_{GS} interval (second row), and the POD subspace in the last I_{GS} interval (third row).

the method without libraries, providing CPU acceleration factors of $C = 26.91$ and $C = 90.96$, respectively. Furthermore, combining such generic libraries with the 'simple' library ℓ_1 produces outstanding CPU acceleration factors, namely $C = 35.42$ and $C = 357.60$ for $\ell_{F1} + \ell_1$ and $\ell_{F2} + \ell_1$, respectively.

The ability of the ROM developed in the paper to take advantage of mode libraries adapting their modes to the system dynamics is illustrated in Figure 16. Now, the mode libraries are not extracted from the dynamics of the system itself and thus are not well suited for the first I_{GS} interval. As a consequence, the POD modes constructed by the method by mixing mode libraries and modes resulting from snapshots calculated by the NS in the first I_{NS} interval (second row) are qualitatively different from those in the library (first row). This is a completely different situation to what happened in the application of the ROM without libraries, as comparison of Figures 13 and 16 shows. On the other hand, the sets of modes in these two figures are quite different from each other in spite of the fact that they are appropriate to simulate the same dynamics at comparable values of time, which means that the POD subspaces spanned by the two sets of POD modes must not be quite far from each other. Nevertheless, even if the two POD subspaces were exactly the same, the two sets of POD modes are just two orthonormal bases of the subspace and by no means need to coincide.

In summary of the earlier discussion, the performance of the method in 2D is much better than in 1D. Thanks to the more favorable ratio of the numerical DOFs to the relevant DOFs identified by the low-dimensional system, the CPU acceleration factors are now much closer to their asymptotic value, namely the theoretical acceleration factor. On the other hand, the same adaptive method used in 1D, with essentially the same tunable parameters, is appropriate for the 2D test problem, which emphasizes the robustness of the method.

6. FINAL REMARKS

An adaptive ROM has been developed that first calculates and then updates 'on the fly' (using a standard NS) the POD subspace that is used to construct a GS. The method is quite effective in both ensuring accuracy and avoiding the mode truncation instability, which is performed monitoring the amplitudes of some additional higher-order POD modes and monitoring a conveniently defined

residual, respectively. The resulting method adapts well the POD subspace to the local dynamics in complex behaviors. Using mode libraries produces an additional improvement of the computational efficiency, especially for solutions exhibiting chaotic dynamics.

Illustration has been made considering the 1D CGLE, which is a well-known benchmark for chaotic dynamics, and the 2D version of this equation. Both applications have been performed using essentially the same values of the tunable parameters of the method, without any intention to optimize the implementation of the method and the MATLAB software that has been constructed. In these applications, CPU acceleration factors as large as 15 and 350 have been obtained in simulations of fairly complex chaotic dynamics in one and two space dimensions, respectively. In the 1D application, the improvement in terms of CPU time is generally smaller than what could be expected from the observed ratio between the total time span and the total length of the I_{NS} intervals (which would give the acceleration factor if the GS cost were negligible). The reason is that the finite difference NS scheme involves sparse matrices, while the counterparts in the GS are full matrices. In spite of that, the adaptive ROM yields remarkably large acceleration factors. In the 2D application, instead, the CPU acceleration factors are comparable with their asymptotic values because now the CPU cost of the ROM is negligible. This is quite promising envisaging industrial applications in higher spatial dimensions.

Application to other problems exhibiting complex behaviors is expected to work equally well if the number of relevant modes is not too large. This includes many multi-dimensional pattern-forming systems. Note that, in principle (for a given temporal complexity), the ratio of the physically relevant DOFs to the numerical DOFs decreases as the spatial dimension increases, which suggests that the efficiency of the adaptive ROM will be even better in higher-dimensional problems.

The adaptive ROM in this paper is expected to be much more computationally efficient than current preprocessed ROMs, especially when the system is to be simulated a limited number of times. In this case, a fair comparison between adaptive and preprocessed ROMs should consist in comparing the total offline/online CPU time involved in the calculation. In many query scenarios, it is only the online time that matters, and preprocessed ROMs (which are designed precisely for this case) are appropriate. Nevertheless, even in this case, the adaptive ROM could be advantageously used (instead of the NS) in the offline calculation of the snapshots.

Concerning fluid flows, the adaptive ROM in this paper is expected to work well for laminar and incipiently transitional flows, exhibiting only temporal complexity, such as those relevant in typical pattern-forming systems. Fully transitional and turbulent flows exhibiting a high degree of spatio-temporal complexity, instead, may require a huge number of POD modes. In fact, the computationally efficient reduced-order modeling of these very complex flows remains unperformed, in spite of the many efforts in the literature in this direction, after the pioneering work by Sirovich [21] and Aubry *et al.* [41]. Some recent ideas, combining reduced modeling with classical steady [18] and nonsteady [42] turbulence modeling, might be useful in this direction.

Unsteady laminar/incipiently transitional flows are of interest in industrial environments, to which the appropriate extensions of the results in this paper could apply. It may be argued that very complex behaviors exhibiting temporal chaos, such as those considered in this paper, are to be avoided in many engineering systems. But this is only partially true, because temporal complexity involving quasi-periodic/chaotic dynamics is sometimes unavoidable. Two examples, among others, are some thermal convection systems [43] (of industrial interest in, e.g., microcooling devices [44]) and some aeroelastic systems (relevant in, e.g., aeronautics [11]), which very easily exhibit fairly complex dynamics even in formulations based on a few DOFs only.

In any event, the authors expect that the results in this paper contribute to current efforts to increase the computational efficiency of current simulation tools.

ACKNOWLEDGEMENTS

This work was partially supported by the Spanish Ministry of Education, under grants TRA2013–45808-R and FIS2011–28838-C02-02. The authors are also indebted to two anonymous referees for a careful reading of a previous version of the manuscript and for some useful suggestions that helped improve the paper.

REFERENCES

1. Lucia DJ, Beran PS, Silva WA. Reduced-order modelling: new approaches for computational physics. *Progress in Aerospace Sciences* 2004; **40**:51–117.
2. Chatterjee A. An introduction to the proper orthogonal decomposition. *Current Science* 2000; **78**:808–817.
3. Pearson K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine* 1901; **2**:559–572.
4. Stewart GW. On the early history of the singular value decomposition. *SIAM Review* 1993; **35**:551–566.
5. Rempfer D. On low-dimensional Galerkin models for fluid flow. *Theoretical and Computational Fluid Dynamics* 2000; **14**:75–88.
6. Couplet M, Basdevant C, Sagaut P. Calibrated reduced-order POD-Galerkin system for fluid flow modelling. *Journal of Computational Physics* 2005; **207**:192–220.
7. Sirisup S, Karniadakis GE. A spectral viscosity method for correcting the long-term behavior of POD models. *Journal of Computational Physics* 2004; **194**:92–116.
8. Tadmor G, Lehmann O, Noack BR, Morzynski M. Mean field representation of the natural and actuated cylinder wake. *Physics of Fluids* 2010; **22**:034102.
9. Bergmann M, Bruneau CH, Iollo A. Enablers for robust POD models. *Journal of Computational Physics* 2009; **228**:516–538.
10. Berkooz G, Holmes P, Lumley JL. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics* 1993; **25**:539–575.
11. Dowell EH, Hall KC. Modeling of fluid-structure interaction. *Annual Review of Fluid Mechanics* 2001; **33**:445–490.
12. Lieu T, Farhat C, Lesoinne M. Reduced-order fluid/structure modeling of a complete aircraft configuration. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:5730–5742.
13. Rempfer D. Low-dimensional modeling and numerical simulation of transition in simple shear flows. *Annual Review of Fluid Mechanics* 2003; **35**:229–265.
14. Siegel SG, Seidel J, Fagley C, Luchtenburg DM, Cohen K, McLaughlin T. Low-dimensional modelling of a transient cylinder wake using double proper orthogonal decomposition. *Journal of Fluid Mechanics* 2008; **610**:1–42.
15. Thomas JP, Dowell EH, Hall KC. Using automatic differentiation to create a nonlinear reduced-order-model aerodynamic solver. *AIAA Journal* 2010; **48**:19–24.
16. Alonso D, Velazquez A, Vega JM. A method to generate computationally efficient reduced order models. *Computer Methods in Applied Mechanics and Engineering* 2009; **198**:2683–2691.
17. Alonso D, Vega JM, Velazquez A. Reduced order model for viscous aerodynamic flow past an airfoil. *AIAA Journal* 2010; **48**:1946–1958.
18. Alonso D, Vega JM, Velazquez A, de Pablo V. Reduced-order modeling of three-dimensional external aerodynamic flows. *Journal of Aerospace Engineering* 2012; **25**:588–599.
19. Bache E, Vega JM, Velazquez A. Model reduction in the back step fluid-thermal problem with variable geometry. *International Journal of Thermal Sciences* 2010; **49**:2376–2384.
20. LeGresley PA, Alonso JJ. Investigation of non-linear projection for POD based reduced order models for aerodynamics. *AIAA Paper 2001-0926,* 39th AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV, 2001.
21. Sirovich L. Turbulence and the dynamics of coherent structures. *Quarterly of Applied Mathematics* 1987; **XLV**: 561–590.
22. Chinesta F, Keunings R, Leygue A. *The Proper Generalized Decomposition for Advanced Numerical Simulations*, SpringerBriefs in Applied Sciences and Technology. Springer-Verlag: Berlin, 2014.
23. Rapún ML, Vega JM. Reduced order models based on local POD plus Galerkin projection. *Journal of Computational Physics* 2010; **229**:3046–3063.
24. Terragni F, Valero E, Vega JM. Local POD plus Galerkin projection in the unsteady lid-driven cavity problem. *SIAM Journal of Scientific Computing* 2011; **33**:3538–3561.
25. Terragni F, Vega JM. On the use of POD-based ROMs to analyze bifurcations in some dissipative systems. *Physica D* 2012; **241**:1393–1405.
26. Terragni F, Vega JM. Construction of bifurcation diagrams using POD on the fly. *SIAM Journal on Applied Dynamical Systems* 2014; **13**:339–365.
27. Braconnier T, Ferrier M, Jouhaud JC, Montagnac M, Sagaut P. Towards an adaptive POD/SVD surrogate model for aeronautic design. *Computers & Fluids* 2011; **40**:195–209.
28. Lorente LS, Vega JM, Velazquez A. Efficient computation of the POD manifold containing the information required to generate a multi-parameter aerodynamic database. *Aerospace Science and Technology* 2013; **25**:152–160.
29. Sirisup S, Karniadakis GE, Xiu D, Kevrekidis IG. Equations-free/Galerkin-free POD assisted computation of incompressible flows. *Journal of Computational Physics* 2005; **207**:568–587.
30. Renardy M, Rogers RC. *An Introduction to Partial Differential Equations*, Texts Appl. Math, vol. 13. Springer-Verlag: New York, 2004.
31. Cebeci T. *Convective Heat Transfer*. Springer: Berlin, 2002.
32. Astrid P, Weiland S, Willcox K, Backx T. Missing point estimation methods in models described by proper orthogonal decomposition. *IEEE Transactions on Automatic Control* 2008; **53**:2237–2250.
33. Ryckelynck D. Hyper-reduction of mechanical models involving internal variables. *International Journal for Numerical Methods in Engineering* 2009; **77**:75–89.
34. Gottlieb D, Orszag SA. *Numerical Analysis of Spectral Methods: Theory and Applications*. SIAM: Philadelphia, 1977.

35. Wang Z, Oberai AA. Spectral analysis of the dissipation of the residual-based variational multiscale method. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**:810–818.
36. Grepl MA, Patera AT. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: M2AN* 2005; **39**:157–181.
37. Amsallem D, Farhat C. An interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA Journal* 2008; **46**:1803–1813.
38. Haasdonk B, Ohlberger M. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: M2AN* 2008; **42**:277–302.
39. Ott E. *Chaos in Dynamical Systems* (Second Edition). Cambridge University Press: Cambridge, 2002.
40. Aranson IS, Kramer L. The world of the complex Ginzburg–Landau equation. *Reviews of Modern Physics* 2002; **74**:100–142.
41. Aubry N, Holmes P, Lumley JL, Stone E. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *Journal of Fluid Mechanics* 1988; **192**:115–173.
42. Balajewicz MJ, Dowell EH, Noack BR. Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier–Stokes equation. *Journal of Fluid Mechanics* 2013; **729**:285–308.
43. Cross M, Greenside H. *Pattern Formation and Dynamics in Nonequilibrium Systems*. Cambridge University Press: Cambridge, UK, 2009.
44. Meis M, Varas F, Velazquez A, Vega JM. Heat transfer enhancement in micro-channels caused by vortex promoters. *International Journal of Heat and Mass Transfer* 2010; **53**:29–40.