Discrete Empirical Interpolation for Nonlinear Model Reduction

Saifon Chaturantabut and Danny C. Sorensen
Department of Computational and Applied Mathematics,
MS-134, Rice University, 6100 Main Street, Houston, Texas 77005-1892, USA.
sc3@rice.edu, sorensen@rice.edu

April 8, 2009

Abstract

A dimension reduction method called Discrete Empirical Interpolation (DEIM) is proposed and shown to dramatically reduce the computational complexity of the popular Proper Orthogonal Decomposition (POD) method for constructing reduced-order models for unsteady and/or parametrized nonlinear partial differential equations (PDEs). In the presence of a general nonlinearity, the standard POD-Galerkin technique reduces dimension in the sense that far fewer variables are present, but the complexity of evaluating the nonlinear term remains that of the original problem. Here we describe DEIM as a modification of POD that reduces the complexity as well as the dimension of general nonlinear systems of ordinary differential equations (ODEs). It is, in particular, applicable to ODEs arising from finite difference discretization of unsteady time dependent PDE and/or parametrically dependent steady state problems. Our contribution is a greatly simplified description of Empirical Interpolation in a finite dimensional setting. The method possesses an error bound on the quality of approximation. An application of DEIM to a finite difference discretization of the 1-D FitzHugh-Nagumo equations is shown to reduce the dimension from 1024 to order 5 variables with negligible error over a long-time integration that fully captures non-linear limit cycle behavior. We also demonstrate applicability in higher spatial dimensions with similar state space dimension reduction and accuracy results.

1 Introduction

Model order reduction (MOR) seeks to reduce the computational complexity and computational time of large-scale dynamical systems by approximations of much lower dimension that can produce nearly the same input/output response characteristics. High dimensional ODE systems arising from discretization of partial differential equations (PDEs) are primary examples. Dimension reduction of discretized unsteady and/or parametrized nonlinear PDEs is of great value in reducing computational times in many applications including the steady state flow problem and unsteady neuron modeling examples used here as illustrations. These discrete systems often must become very high dimensional to achieve desired accuracy in the numerical solutions. We introduce a Discrete Empirical Interpolation Method (DEIM) to greatly improve the dimension reduction efficiency of proper orthogonal decomposition (POD), a popular approach to constructing reduced-order models for these discrete systems.

POD has provided reduced-order models of systems in numerous applications such as compressible flow [8], fluid dynamics [5], aerodynamics [2], and optimal control [4]. However, effective dimension reduction for the POD-Galerkin approach is usually limited to the linear or bi-linear terms, as shown explicitly in Section 2.2. When a general nonlinearity is present, the cost to evaluate the projected nonlinear function still depends on the dimension of the original system, resulting in simulation times that hardly improve over the original system. A method for eliminating this inefficiency is therefore essential for MOR of nonlinear systems.

The Discrete Empirical Interpolation Method detailed in Section 3, improves the POD approximation and achieves an honest complexity reduction of the nonlinear term with a complexity proportional to the number of reduced variables. DEIM is a discrete variant of the Empirical Interpolation Method introduced by Barrault, Maday, Nguyen and Patera [1] which was originally posed in an empirically derived finite dimensional function space. The approximation from DEIM is based on projection combined with interpolation. It constructs specially selected interpolation indices that specify an interpolation based projection that gives a nearly optimal subspace approximation to the nonlinear term without the expense of orthogonal projection. A procedure for selecting these indices is provided and shown to be numerically stable and inexpensive. The resulting approximation possesses an error bound indicating it is always nearly as good as the far more expensive orthogonal projection of POD.

Section 2 describes the problem setup arising from nonlinear finite difference (FD) discretized of unsteady time dependent PDE and steady state parameter dependent PDE. Dimension reduction via POD is reviewed in Section 2.1 followed by a discussion of a fundamental complexity issue in Section 2.2. While POD can do an excellent job of reducing the number of variables, the complexity of evaluating the nonlinear terms typically remains to be that of the original system. The DEIM approximation, which provides an excellent remedy to the complexity issue, is introduced in Section 3. The key to complexity reduction is to replace orthogonal projection of POD with the interpolation projection of DEIM in the same POD basis. An algorithm for selecting the interpolation indices used in the DEIM approximation is presented in Section 3.1. Section 3.2 provides an error bound on this interpolatory approximation indicating that it is nearly as good as orthogonal projection. Section 3.3 illustrates the validity of this error bound and the high quality of the DEIM approximations with selected numerical examples. In Section 3.4 we explain how to apply the DEIM approximation to nonlinear terms in POD reduced-order models of FD discretized systems and then we extend this to general nonlinear ODEs in Section 3.5. Finally, in Section 4, we give computational evidence of DEIM effectiveness in two specific problem settings. DEIM applied to a 1024 variable discretization of the FitzHugh-Nagumo equations produced a 5 variable reduced order model which was able to capture limit cycle behavior over a long-time integration. Similar effectiveness is demonstrated for a 2-D steady state parametrically dependent flow problem.

2 Problem Formulation

This section considers FD discretized systems arising from two types of nonlinear PDEs, which are used for our numerical examples in Section 4. One is unsteady time-dependent and the other is steady but parametrized. These settings are simplified for simplicity. We discuss general nonlinearities in Section 3.5.

The FD discretization of an unsteady scalar nonlinear PDE in one spatial variable of the form

$$\frac{\partial y(x,t)}{\partial t} = \mathcal{L}(y(x,t)) + F(y(x,t)) \tag{1}$$

results in a system nonlinear ODEs of the form

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{A}\mathbf{y}(t) + \mathbf{F}(\mathbf{y}(t)), \tag{2}$$

with appropriate initial conditions. Here $t \in [0,T]$ denotes time, $\mathbf{y}(t) = [\mathbf{y}_1(t), \ldots, \mathbf{y}_n(t)]^T \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a constant matrix, $\mathbf{F}$ is a nonlinear function evaluated at $\mathbf{y}(t)$ componentwise, i.e., $\mathbf{F} = [F(\mathbf{y}_1(t)), \ldots, F(\mathbf{y}_n(t))]^T$, $F : \mathcal{I} \mapsto \mathbb{R}$ for $\mathcal{I} \subset \mathbb{R}$. The matrix $\mathbf{A}$ is the discrete approximation of the linear spatial differential operator $\mathcal{L}$, and F is a nonlinear function of a scalar variable.

Steady nonlinear PDEs (in several spatial dimensions) might give rise similarly to a corresponding FD discretized system of the form

$$\mathbf{A}\mathbf{y}(\mu) + \mathbf{F}(\mathbf{y}(\mu)) = 0, \tag{3}$$

with the corresponding Jacobian

$$\mathbf{J}(\mathbf{y}(\mu)) := \mathbf{A} + \mathbf{J}_{\mathbf{F}}(\mathbf{y}(\mu)), \tag{4}$$

where $\mathbf{y}(\mu) = [\mathbf{y}_1(\mu), \ldots, \mathbf{y}_n(\mu)]^T \in \mathbb{R}^n$, $\mathbf{A}$ and $\mathbf{F}$ defined as for (2). Note that from (4), the Jacobian of the nonlinear function is a diagonal matrix given by

$$\mathbf{J}_{\mathbf{F}}(\mathbf{y}(\mu)) = \text{diag}\{F'(\mathbf{y}_1(\mu)), \ldots, F'(\mathbf{y}_n(\mu))\} \in \mathbb{R}^{n\times n}, \tag{5}$$

where F' denotes the first derivative of F. The parameter $\mu \in \mathscr{D} \subset \mathbb{R}^d$, $d = 1, 2, \ldots,$ generally represents the system's configuration in terms of its geometry, material properties, etc.

To simplify exposition, we have considered time dependence and parametric dependence separately. Note, however, that the two may be merged to address time dependent parametrized systems. For example, if one wished to allow for a variety of initial conditions in a time dependent problem, including them as parameters would be a possibility.

The dimension n of (2) and (3) reflects the number of spatial grid points used in the FD discretization. As noted, the dimension n can become extremely large when high accuracy is required. Hence, solving these systems becomes computationally intensive or possibly infeasible. Approximate models with much smaller dimensions are needed to recover the efficiency.

Projection-based techniques are commonly used for constructing a reduced-order system. They construct a reduced-order system of order $k \ll n$ that approximates the original system from a subspace spanned by a *reduced basis* of dimension k in $\mathbb{R}^n$. We use Galerkin projection as the means for dimension reduction. In particular, let $\mathbf{V}_k \in \mathbf{R}^{n\times k}$ be a matrix whose orthonormal columns are the vectors in the reduced basis. Then by replacing $\mathbf{y}(t)$ in (2) by $\mathbf{V}_k\tilde{\mathbf{y}}(t)$, $\tilde{\mathbf{y}}(t) \in \mathbb{R}^k$ and projecting the system (2) onto $\mathbf{V}_k$, the reduced system of (2) is of the form

$$\frac{d}{dt}\tilde{\mathbf{y}}(t) = \underbrace{\mathbf{V}_k^T\mathbf{A}\mathbf{V}_k}_{\tilde{\mathbf{A}}}\tilde{\mathbf{y}}(t) + \mathbf{V}_k^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(t)). \tag{6}$$

Similarly, the reduced-order system of (3) is of the form

$$\underbrace{\mathbf{V}_k^T\mathbf{A}\mathbf{V}_k}_{\tilde{\mathbf{A}}}\tilde{\mathbf{y}}(\mu) + \mathbf{V}_k^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(\mu)) = 0, \tag{7}$$

with corresponding Jacobian

$$\tilde{\mathbf{J}}(\tilde{\mathbf{y}}(\mu)) := \tilde{\mathbf{A}} + \mathbf{V}_k^T\mathbf{J}_{\mathbf{F}}(\mathbf{V}_k\tilde{\mathbf{y}}(\mu))\mathbf{V}_k, \tag{8}$$

where $\tilde{\mathbf{A}} = \mathbf{V}_k^T\mathbf{A}\mathbf{V}_k \in \mathbb{R}^{k\times k}$. The choice of the reduced basis clearly affects the quality of the approximation. The techniques for constructing a set of reduced basis uses a common observation that, for a particular system, the solution space is often attracted to a low dimensional manifold. Therefore, the reduced basis is generally problem dependent. POD constructs these global basis functions from an SVD of *snapshots*, which are discrete samples of trajectories associated with a particular set of boundary conditions and inputs.

Among the various techniques for obtaining a reduced basis, POD constructs a reduced basis that is *optimal* in the sense that a certain approximation error concerning the snapshots is minimized. Thus, the space spanned by the basis from POD often gives a better approximation than other approaches with the same dimension. The POD technique is therefore used here as a starting point.

2.1 Proper Orthogonal Decomposition (POD)

POD is a method for constructing a low-dimensional approximation representation of a subspace in Hilbert space. It is essentially the same as the singular value decomposition (SVD) in a finite dimensional space or in Euclidean space. It efficiently extracts the basis elements that contain characteristics of the space of expected solutions of the PDE. The POD basis in Euclidean space is defined formally as follows.
Given a set of snapshots $\{\mathbf{y}^1, \ldots, \mathbf{y}^{n_s}\} \subset \mathbb{R}^n$ (recall *snapshots* are samples of trajectories), let $\mathscr{Y} = span\{\mathbf{y}^1, \ldots, \mathbf{y}^{n_s}\} \subset \mathbb{R}^n$ and $r = \dim(\mathscr{Y})$. A POD basis of dimension $k < r$ is a set of orthonormal vectors $\{\phi\}_{i=1}^k \subset \mathbb{R}^n$ whose linear span best approximates the space $\mathscr{Y}$. The basis set $\{\phi\}_{i=1}^k$ solves the minimization problem

$$\begin{gathered}\min_{\{\phi_i\}_{i=1}^k} \textstyle\sum_{j=1}^{n_s} \|\mathbf{y}_j - \sum_{i=1}^{k} (\mathbf{y}_j^T \phi_i)\phi_i\|_2^2 \\ \phi_i^T \phi_j = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad i, j = 1, ..., k \end{gathered} \tag{9}$$

It is well known that the solution to (9) is provided by the set of the left singular vectors of the *snapshot matrix* $\mathbf{Y} = [\mathbf{y}^1, \ldots, \mathbf{y}^{n_s}] \in \mathbb{R}^{n \times n_s}$. In particular, suppose that the singular value decomposition (SVD) of $\mathbf{Y}$ is

$$\mathbf{Y} = \mathbf{V}\Sigma\mathbf{W}^T,$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{R}^{n \times r}$ and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r] \in \mathbb{R}^{n_s \times r}$ are orthogonal and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. The rank of $\mathbf{Y}$ is $r \leq \min(n, n_s)$. Then the POD basis or the optimal solution of (9) is $\{\mathbf{v}_i\}_{i=1}^k$. The minimum 2-norm error from approximating the snapshots using the POD basis is then given by

$$\sum_{j=1}^{n_s} \|\mathbf{y}_i - \sum_{i=1}^{k} (\mathbf{y}_j^T \mathbf{v}_i)\mathbf{v}_i\|_2^2 = \sum_{i=k+1}^{r} \sigma_i^2.$$

We refer to [5] for more details on the POD basis in general Hilbert space.

The choice of the snapshot ensemble is a crucial factor in constructing a POD basis. This choice can greatly affect the approximation of the original solution space, but it is a separate issue and will not be discussed here. POD works well in many applications and generally gives better approximations than any other known reduced-basis techniques. However, when POD is used in conjunction with the Galerkin projection, effective dimension reduction is usually limited to the linear terms, as shown next in Section 2.2. Systems with nonlinearities need additional treatment, which will be presented in Section 3.

2.2 A Problem with Complexity of the POD-Galerkin Approach

This section illustrates the computational inefficiency that occurs in solving the reduced-order system that is directly obtained from POD-Galerkin approach. Equation (6) has nonlinear term

$$\tilde{\mathbf{N}}(\tilde{\mathbf{y}}) := \underbrace{\mathbf{V}_k^T}_{k \times n} \underbrace{\mathbf{F}(\mathbf{V}_k \tilde{\mathbf{y}}(t))}_{n \times 1}. \tag{10}$$

$\tilde{\mathbf{N}}(\tilde{\mathbf{y}})$ has a computational complexity that depends on n, the dimension of the original full-order system (2). It requires on the order of nk Flops for matrix-vector multiplications and it also requires a full evaluation of the nonlinear function $\mathbf{F}$ at the n-dimensional vector $\mathbf{V}_k\tilde{\mathbf{y}}(t)$. Thus, if the complexity for evaluating the nonlinear function $\mathbf{F}$ with q components is $\mathcal{O}(\alpha(q))$, then the complexity for computing (10) is roughly $\mathcal{O}(\alpha(n) + 4nk)$. As a result, solving this system might still be as costly as solving the original system.

The same inefficiency occurs when solving the reduced-order system (7) for the steady nonlinear PDEs by Newton iteration. At each iteration, both a nonlinear term of the form (10) and Jacobian (8) must be computed with a computational cost that still depends on the full-order dimension n due to the Jacobian of the nonlinear term:

$$\tilde{\mathbf{J}}_{\mathbf{F}}(\tilde{\mathbf{y}}(\mu)) := \underbrace{\mathbf{V}_k^T}_{k\times n}\underbrace{\mathbf{J}_{\mathbf{F}}(\mathbf{V}_k\tilde{\mathbf{y}}(\mu))}_{n\times n}\underbrace{\mathbf{V}_k}_{n\times k}. \tag{11}$$

The computational complexity for computing (11) is roughly $\mathcal{O}(\alpha(n) + 4nk + 2nk^2)$.

3 Discrete Empirical Interpolation Method (DEIM)

An effective way to overcome the difficulty described in Section 2.2 is to approximate the nonlinear function in (6) or (7) by projecting it onto a subspace that approximates space generated by the nonlinear function and that is spanned by a basis of dimension $m \ll n$. This section considers the nonlinear functions $\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(t))$ and $\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(\mu))$ of the reduced-order systems (6) and (7), respectively, in a simplified notation given by $\mathbf{f}(\tau)$, where $\tau = t$ or μ. The approximation from projecting $\mathbf{f}(\tau)$ onto the subspace spanned by the basis $\{\mathbf{u}_1, \dots, \mathbf{u}_m\} \subset \mathbb{R}^n$ is of the form

$$\mathbf{f}(\tau) \approx \mathbf{U}\mathbf{c}(\tau), \tag{12}$$

where $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_m] \in \mathbb{R}^{n\times m}$ and $\mathbf{c}(\tau)$ is the corresponding coefficient vector. To determine $\mathbf{c}(\tau)$, we select m distinguished rows from the overdetermined system $\mathbf{f}(\tau) = \mathbf{U}\mathbf{c}(\tau)$. In particular, consider a matrix

$$\mathbf{P} = [\mathbf{e}_{\wp_1}, \dots, \mathbf{e}_{\wp_m}] \in \mathbb{R}^{n\times m}, \tag{13}$$

where $\mathbf{e}_{\wp_i} = [0, \dots, 0, 1, 0, \dots, 0]^T \in \mathbb{R}^n$ is the $\wp_i$-th column of the identity matrix $\mathbf{I}_n \in \mathbb{R}^{n\times n}$, for $i = 1, \dots, m$. Suppose $\mathbf{P}^T\mathbf{U}$ is nonsingular. Then the coefficient vector $\mathbf{c}(\tau)$ can be determined uniquely from

$$\mathbf{P}^T\mathbf{f}(\tau) = (\mathbf{P}^T\mathbf{U})\mathbf{c}(\tau), \tag{14}$$

and the final approximation from (12) becomes

$$\mathbf{f}(\tau) \approx \mathbf{U}\mathbf{c}(\tau) = \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\tau). \tag{15}$$

To obtain the approximation (18), we must specify

1. Projection basis $\{\mathbf{u}_1, \dots, \mathbf{u}_m\}$;
2. Interpolation indices $\{\wp_1, \dots, \wp_m\}$ used in (13).

The projection basis $\{\mathbf{u}_1, \dots, \mathbf{u}_m\}$ for the nonlinear function $\mathbf{f}$ is constructed by applying the POD on the *nonlinear snapshots* obtained from the original full-order system. These nonlinear snapshots are the sets $\{\mathbf{F}(\mathbf{y}(t_1)), \dots, \mathbf{F}(\mathbf{y}(t_{n_s}))\}$ or $\{\mathbf{F}(\mathbf{y}(\mu_1)), \dots, \mathbf{F}(\mathbf{y}(\mu_{n_s}))\}$ obtained from (10) or (11).

The interpolation indices $\wp_1, \dots, \wp_m$, used for determining the coefficient vector $\mathbf{c}(\tau)$ in the approximation (12), are selected inductively from the basis $\{\mathbf{u}_1, \dots, \mathbf{u}_m\}$ by the DEIM algorithm introduced in the next section.

Algorithm 1 : DEIM

```
INPUT: $\{\mathbf{u}_\ell\}_{\ell=1}^m \subset \mathbb{R}^n$ linearly independent
OUTPUT: $\vec{\wp} = [\wp_1, \dots, \wp_m]^T \in \mathbb{R}^m$
1: $[|\rho| \;\; \wp_1] = \texttt{max}\{|\mathbf{u}_1|\}$
2: $\mathbf{U} = [\mathbf{u}_1]$, $\mathbf{P} = [\mathbf{e}_{\wp_1}]$, $\vec{\wp} = [\wp_1]$
3: for $\ell = 2$ to $m$ do
4:     Solve $(\mathbf{P}^T\mathbf{U})\mathbf{c} = \mathbf{P}^T\mathbf{u}_\ell$ for $\mathbf{c}$
5:     $\mathbf{r} = \mathbf{u}_\ell - \mathbf{U}\mathbf{c}$
6:     $[|\rho| \;\; \wp_\ell] = \texttt{max}\{|\mathbf{r}|\}$
7:     $\mathbf{U} \leftarrow [\mathbf{U} \;\; \mathbf{u}_\ell]$, $\mathbf{P} \leftarrow [\mathbf{P} \;\; \mathbf{e}_{\wp_\ell}]$, $\vec{\wp} \leftarrow \begin{bmatrix} \vec{\wp} \\ \wp_\ell \end{bmatrix}$
8: end for
```

3.1 DEIM: Algorithm for Interpolation Indices

DEIM is a discrete variant of the Empirical Interpolation Method (EIM) proposed in [1] for constructing an approximation of a non-affine parametrized function with spatial variable defined in a continuous bounded domain Ω. The DEIM algorithm treats the *continuous* domain Ω as a *finite set of discrete points* in Ω. The DEIM algorithm selects an index at each iteration to limit growth of an error bound. This provides a derivation of a global error bound as presented in Section 3.2.

The notation `max` in Algorithm 1 is the same as the function `max` in MATLAB. Thus, $[|\rho| \;\; \wp_\ell] = \texttt{max}\{|\mathbf{r}|\}$ implies $|\rho| = |r_{\wp_\ell}| = \max_{i=1,\dots,n}\{|r_i|\}$, with the smallest index taken in case of a tie. Note that we define $\rho = r_{\wp_\ell}$ in each iteration $\ell = 1, \dots, m$.

From Algorithm 1, the DEIM procedure constructs a set of indices inductively on the input basis. The order of the input basis $\{\mathbf{u}_\ell\}_{\ell=1}^m$ according to the dominant singular values is important and an error analysis indicates that the POD basis is a suitable choice for this algorithm. The process starts from selecting the first interpolation index $\wp_1 \in \{1, \dots, n\}$ corresponding to the entry of the first input basis $\mathbf{u}_1$ with largest magnitude. The remaining interpolation indices, $\wp_\ell$ for $\ell = 2, \dots, m$, are selected so that each of them corresponds to the entry with the *largest* magnitude of the *residual* $\mathbf{r} = \mathbf{u}_\ell - \mathbf{U}\mathbf{c}$ from line 5 of Algorithm 1. The term $\mathbf{r}$ can be viewed as the *residual* or the *error* between the input basis $\mathbf{u}_\ell$ and its approximation $\mathbf{U}\mathbf{c}$ from interpolating the basis $\{\mathbf{u}_1, \dots, \mathbf{u}_{\ell-1}\}$ at the indices $\wp_1, \dots, \wp_{\ell-1}$ in line 4 of Algorithm 1. Hence, $\mathbf{r}_{\wp_i} = 0$ for $i = 1, \dots, \ell - 1$. However, the linear independence of the input basis $\{\mathbf{u}_\ell\}_{\ell=1}^m$ guarantees that, in each iteration, $\mathbf{r}$ is a nonzero vector and hence ρ is also nonzero. We shall demonstrate in Lemma 3.2 that $\rho \neq 0$ at each step implies $\mathbf{P}^T\mathbf{U}$ is always nonsingular and hence the DEIM procedure is well-defined. This also implies that the interpolation indices $\{\wp_i\}_{i=1}^m$ are hierarchical and non-repeated. Figure 1 illustrates the selection procedure in Algorithm 1 for DEIM interpolation indices.

To summarize, the DEIM approximation is given formally as follows.

Definition 3.1 *Let $\mathbf{f} : \mathscr{D} \mapsto \mathbb{R}^n$ be a nonlinear vector-valued function with $\mathscr{D} \subset \mathbb{R}^d$, for some positive integer d. Let $\{\mathbf{u}_\ell\}_{\ell=1}^m \subset \mathbb{R}^n$ be a linearly independent set, for $m = 1, \dots, n$. For $\tau \in \mathscr{D}$, the DEIM approximation of order m for $\mathbf{f}(\tau)$ in the space spanned by $\{\mathbf{u}_\ell\}_{\ell=1}^m$ is given by*

$$\hat{\mathbf{f}}(\tau) := \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\tau), \tag{16}$$

where $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_m] \in \mathbb{R}^{n\times m}$ and $\mathbf{P} = [\mathbf{e}_{\wp_1}, \dots, \mathbf{e}_{\wp_m}] \in \mathbb{R}^{n\times m}$ with $\{\wp_1, \dots, \wp_m\}$ being the output from Algorithm 1 with the input basis $\{\mathbf{u}_i\}_{i=1}^m$.

Notice that $\hat{\mathbf{f}}$ in (16) is indeed an interpolation approximation for the original function $\mathbf{f}$, since $\hat{\mathbf{f}}$ is exact at the interpolation indices; i.e, for $\tau \in \mathscr{D}$,

$$\mathbf{P}^T\hat{\mathbf{f}}(\tau) = \mathbf{P}^T\left(\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\tau)\right) = (\mathbf{P}^T\mathbf{U})(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\tau) = \mathbf{P}^T\mathbf{f}(\tau).$$

Notice also that the DEIM approximation is uniquely determined by the projection basis $\{\mathbf{u}_i\}_{i=1}^m$. This basis not only specifies the projection subspace used in the approximation, but also determines the interpolation indices used for computing the coefficient of the approximation. Hence, the choice of projection basis can greatly affect the accuracy of the approximation in (16) as shown also in the error bound of the DEIM approximation (19) in the next section. As noted, proper orthogonal decomposition (POD) introduced in Section 2.1 is an efficient method for constructing this projection basis, since it provides a set of optimal global basis that captures the dynamic of the space of nonlinear function. POD basis is therefore used in this paper for constructing a projection basis.

The selection of the interpolation points is basis dependent. However, once the set of DEIM interpolation indices $\{\wp_\ell\}_{\ell=1}^m$ is determined from $\{\mathbf{u}_i\}_{i=1}^m$, the DEIM approximation is independent of the choice of basis spanning the space Range($\mathbf{U}$). In particular, let $\{\mathbf{q}_\ell\}_{\ell=1}^m$ be any basis for Range($\mathbf{U}$). Then

$$\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\tau) = \mathbf{Q}(\mathbf{P}^T\mathbf{Q})^{-1}\mathbf{P}^T\mathbf{f}(\tau), \tag{17}$$

where $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_m] \in \mathbb{R}^{n\times m}$. To verify (17), note that Range($\mathbf{U}$) = Range($\mathbf{Q}$) so that $\mathbf{U} = \mathbf{QR}$ for some nonsingular matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$. This substitution gives

$$\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\tau) = (\mathbf{QR})((\mathbf{P}^T\mathbf{Q})\mathbf{R})^{-1}\mathbf{P}^T\mathbf{f}(\tau) = \mathbf{Q}(\mathbf{P}^T\mathbf{Q})^{-1}\mathbf{P}^T\mathbf{f}(\tau).$$

3.2 Error Bound for DEIM

This section provides an error bound for the DEIM approximation. The bound is obtained recursively on the magnification factor of the best 2-norm approximation error. This error bound is given formally as follows.

Lemma 3.2 *Let $\mathbf{f} \in \mathbb{R}^n$ be an arbitrary vector. Let $\{\mathbf{u}_\ell\}_{\ell=1}^m \subset \mathbb{R}^n$ be a given orthonormal set of vectors. From Definition 3.1, the DEIM approximation of order m for $\mathbf{f}$ in the space spanned by $\{\mathbf{u}_\ell\}_{\ell=1}^m$ is*

$$\hat{\mathbf{f}} = \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}, \tag{18}$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m] \in \mathbb{R}^{n\times m}$ and $\mathbf{P} = [\mathbf{e}_{\wp_1}, \ldots, \mathbf{e}_{\wp_m}] \in \mathbb{R}^{n\times m}$ with $\{\wp_1, \ldots, \wp_m\}$ being the output from Algorithm 1 with the input basis $\{\mathbf{u}_i\}_{i=1}^m$. An error bound for $\hat{\mathbf{f}}$ is then given by

$$\boxed{\|\mathbf{f} - \hat{\mathbf{f}}\|_2 \le \|(\mathbf{P}^T\mathbf{U})^{-1}\|_2\, \mathscr{E}_*(\mathbf{f}) \le \mathcal{C}\, \mathscr{E}_*(\mathbf{f})} \tag{19}$$

where

$$\mathscr{E}_*(\mathbf{f}) = \|(\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{f}\|_2 \tag{20}$$

is the error of the best 2-norm approximation for $\mathbf{f}$ from the space Range($\mathbf{U}$)*, and $\mathcal{C}$ is a constant obtained recursively from the DEIM iteration as follows:*
For $m = 1$,

$$\mathcal{C}_1 = \frac{1}{|\mathbf{e}_{\wp_1}^T\mathbf{u}_1|} = \|\mathbf{u}_1\|_\infty^{-1}. \tag{21}$$

For $\ell = 2, \ldots, m$,

$$\mathcal{C}_\ell = \left[1 + \sqrt{2n}\right] \mathcal{C}_{\ell-1}. \tag{22}$$

Finally, $\mathcal{C} = \mathcal{C}_m$.

Proof:

This proof provides motivation for DEIM selection process in terms of minimizing the local error growth for the approximation.

Consider the DEIM approximation $\hat{\mathbf{f}}$ given by (18). We wish to determine a bound for the error $\|\mathbf{f}-\hat{\mathbf{f}}\|_2$ in terms of the *optimal* 2-norm approximation for $\mathbf{f}$ from Range($\mathbf{U}$). This best approximation is given by

$$\mathbf{f}_* = \mathbf{U}\mathbf{U}^T\mathbf{f}, \tag{23}$$

which minimizes the error $\|\mathbf{f} - \hat{\mathbf{f}}\|_2$ over Range($\mathbf{U}$). Consider

$$\mathbf{f} = (\mathbf{f} - \mathbf{f}_*) + \mathbf{f}_* = \mathbf{w} + \mathbf{f}_*, \tag{24}$$

where $\mathbf{w} = \mathbf{f} - \mathbf{f}_* = (\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{f}$. Define the projector $\mathbb{P} = \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T$. From (16) and (24),

$$\hat{\mathbf{f}} = \mathbb{P}\mathbf{f} = \mathbb{P}(\mathbf{w} + \mathbf{f}_*) = \mathbb{P}\mathbf{w} + \mathbb{P}\mathbf{f}_* = \mathbb{P}\mathbf{w} + \mathbf{f}_*. \tag{25}$$

Equations (24) and (25) imply $\mathbf{f} - \hat{\mathbf{f}} = (\mathbf{I} - \mathbb{P})\mathbf{w}$ and

$$\|\mathbf{f} - \hat{\mathbf{f}}\|_2 = \|(\mathbf{I} - \mathbb{P})\mathbf{w}\|_2 \le \|\mathbf{I} - \mathbb{P}\|_2\|\mathbf{w}\|_2 \tag{26}$$

Note that

$$\|\mathbf{I} - \mathbb{P}\|_2 = \|\mathbb{P}\|_2 = \|\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\|_2 \le \|\mathbf{U}\|_2\|(\mathbf{P}^T\mathbf{U})^{-1}\|_2\|\mathbf{P}^T\|_2 = \|(\mathbf{P}^T\mathbf{U})^{-1}\|_2. \tag{27}$$

The first equality in (27) follows from the fact that $\|\mathbf{I} - \mathbb{P}\|_2 = \|\mathbb{P}\|_2$, for any projector $\mathbb{P} \neq 0$ or $\mathbf{I}$ (see [9]). The last equality in (27) follows from the fact that $\|\mathbf{U}\|_2 = \|\mathbf{P}^T\|_2 = 1$, since each of the matrices $\mathbf{U}$ and $\mathbf{P}$ has orthonormal columns.

Note that $\mathscr{E}_*(\mathbf{f}) := \|\mathbf{w}\|_2$ is the minimum 2-norm error for $\mathbf{f}_*$ defined in (23). From (27), the bound for the error in (26) becomes

$$\|\mathbf{f} - \hat{\mathbf{f}}\|_2 \le \|(\mathbf{P}^T\mathbf{U})^{-1}\|_2 \;\; \mathscr{E}_*(\mathbf{f}), \tag{28}$$

and $\|(\mathbf{P}^T\mathbf{U})^{-1}\|_2$ is the magnification factor needed to express the DEIM error in terms of the optimal approximation error. Hence determination of the bound for $\|\mathbf{f}-\hat{\mathbf{f}}\|_2$ is reduced to a problem of giving a bound for the matrix norm

The matrix norm $\|(\mathbf{P}^T\mathbf{U})^{-1}\|_2$ depends on the DEIM selection of indices $\wp_1, \ldots, \wp_m$ through the permutation matrix $\mathbf{P}$. We now show that each iteration of the DEIM algorithm aims to select an index to limit stepwise growth of $\|(\mathbf{P}^T\mathbf{U})^{-1}\|_2$ and hence to limit size of the bound for the error $\|\mathbf{f} - \hat{\mathbf{f}}\|_2$.

To simplify notation, for $\ell = 2, \ldots, m$, we denote the relevant quantities at iteration ℓ by

$$\begin{array}{rclrcl} \bar{\mathbf{U}} & = & [\mathbf{u}_1, \ldots, \mathbf{u}_{\ell-1}] \in \mathbb{R}^{n\times(\ell-1)}, & \bar{\mathbf{P}} & = & [\mathbf{e}_{\wp_1}, \ldots, \mathbf{e}_{\wp_{\ell-1}}] \in \mathbb{R}^{n\times(\ell-1)}, \\ \mathbf{u} & = & \mathbf{u}_\ell \in \mathbb{R}^n, & \mathbf{p} & = & \mathbf{e}_{\wp_\ell} \in \mathbb{R}^n, \\ \mathbf{U} & = & [\bar{\mathbf{U}} \;\; \mathbf{u}] \in \mathbb{R}^{n\times\ell}, & \mathbf{P} & = & [\bar{\mathbf{P}} \;\; \mathbf{p}] \in \mathbb{R}^{n\times\ell}, \end{array} \tag{29}$$

Put $\mathbf{M} = \mathbf{P}^T\mathbf{U}$ and consider the matrix norm $\|\mathbf{M}^{-1}\|_2$ from Algorithm 1.
At the initial step of Algorithm 1, $\mathbf{P} = \mathbf{e}_{\wp_1}$ and $\mathbf{U} = \mathbf{u}_1$. Thus,

$$\mathbf{M} = \mathbf{P}^T\mathbf{U} = \mathbf{e}_{\wp_1}^T\mathbf{u}_1 \qquad \|\mathbf{M}^{-1}\|_2 = \frac{1}{|\mathbf{e}_{\wp_1}^T\mathbf{u}_1|} = \|\mathbf{u}_1\|_\infty^{-1} \geq 1, \tag{30}$$

which establishes condition (21). Note that the choice of the first interpolation index $\wp_1$ minimizes the matrix norm $\|\mathbf{M}^{-1}\|_2$ and hence minimize the error bound in (28).

Now consider a general step $\ell \geq 2$ with matrices defined by Equations (29). With $\mathbf{M} = \mathbf{P}^T\mathbf{U}$, we can write $\mathbf{M} = \begin{bmatrix} \bar{\mathbf{M}} & \bar{\mathbf{P}}^T\mathbf{u} \\ \mathbf{p}^T\bar{\mathbf{U}} & \mathbf{p}^T\mathbf{u} \end{bmatrix}$, where $\bar{\mathbf{M}} = \bar{\mathbf{P}}^T\bar{\mathbf{U}}$ and $\mathbf{M}$ can be factored in the form of

$$\mathbf{M} = \begin{bmatrix} \bar{\mathbf{M}} & \bar{\mathbf{P}}^T\mathbf{u} \\ \mathbf{p}^T\bar{\mathbf{U}} & \mathbf{p}^T\mathbf{u} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{M}} & \mathbf{0} \\ \mathbf{a}^T & \rho \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{c} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{31}$$

where $\mathbf{a}^T = \mathbf{p}^T\bar{\mathbf{U}}$, $\mathbf{c} = \bar{\mathbf{M}}^{-1}\bar{\mathbf{P}}^T\mathbf{u}$, and $\rho = \mathbf{p}^T\mathbf{u} - \mathbf{a}^T\mathbf{c} = \mathbf{p}^T(\mathbf{u} - \bar{\mathbf{U}}\bar{\mathbf{M}}^{-1}\bar{\mathbf{P}}^T\mathbf{u})$. Note $|\rho| = \|\mathbf{r}\|_\infty$ where $\mathbf{r}$ is defined at Step 5 of Algorithm 1. Now, the inverse of $\mathbf{M}$ is

$$\begin{aligned}
\mathbf{M}^{-1} &= \begin{bmatrix} \mathbf{I} & -\mathbf{c} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{M}}^{-1} & \mathbf{0} \\ -\rho^{-1}\mathbf{a}^T\bar{\mathbf{M}}^{-1} & \rho^{-1} \end{bmatrix} && (32)\\
&= \begin{bmatrix} \mathbf{I} & -\mathbf{c} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\rho^{-1}\mathbf{a}^T & \rho^{-1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{M}}^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} && (33)\\
&= \left\{ \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{0} & 0 \end{bmatrix} + \rho^{-1} \begin{bmatrix} \mathbf{c} \\ -1 \end{bmatrix} [\mathbf{a}^T, -1] \right\} \begin{bmatrix} \bar{\mathbf{M}}^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} && (34)
\end{aligned}$$

A bound for the 2-norm of $\mathbf{M}^{-1}$ is then given by

$$\|\mathbf{M}^{-1}\|_2 \;\leq\; \left\{ \left\| \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{0} & 0 \end{bmatrix} \right\|_2 + |\rho|^{-1} \left\| \begin{bmatrix} \mathbf{c} \\ -1 \end{bmatrix} [\mathbf{a}^T, -1] \right\|_2 \right\} \left\| \begin{bmatrix} \bar{\mathbf{M}}^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \right\|_2 \tag{35}$$

Now, observe that

$$\begin{aligned}
\left\| \begin{bmatrix} \mathbf{c} \\ -1 \end{bmatrix} [\mathbf{a}^T, -1] \right\|_2 &= \left\| [\bar{\mathbf{U}}, \mathbf{u}] \begin{bmatrix} \mathbf{c} \\ -1 \end{bmatrix} [\mathbf{a}^T, -1] \right\|_2 && (36)\\
&= \left\| \bar{\mathbf{U}}\mathbf{c} - \mathbf{u} \right\|_2 \left\| [\mathbf{a}^T, -1] \right\|_2 && (37)\\
&\leq \sqrt{1 + \|\mathbf{a}\|_2^2}\sqrt{n} \left\| \bar{\mathbf{U}}\mathbf{c} - \mathbf{u} \right\|_\infty \leq \sqrt{2n}|\rho|. && (38)
\end{aligned}$$

Substituting this into (35) gives

$$\|\mathbf{M}^{-1}\|_2 \;\leq\; [1 + \sqrt{2n}]\|\bar{\mathbf{M}}^{-1}\|_2 \tag{39}$$

to establish the recursive Formula (22) and conclude the proof. $\square$

Since the DEIM procedure selects the index $\wp_\ell$ that *maximizes* $|\rho|$, it *minimizes* the reciprocal $\frac{1}{|\rho|}$, which controls the increment in the bound of $\|\mathbf{M}^{-1}\|_2$ at iteration ℓ. Therefore, the selection process for the interpolation index in each iteration of DEIM (line 6 of Algorithm 1) can be explained in terms of limiting growth of the error bound of the approximation $\hat{\mathbf{f}}$. This error bound applies to any nonlinear vector-valued function $\mathbf{f}(\tau) = \mathbf{F}(\mathbf{y}(\tau))$ approximated by DEIM. However, the bound is mainly of theoretical interest since it is very pessimistic and grows far more rapidly than the actual values of $\|(\mathbf{P}^T\mathbf{U})^{-1}\|_2$ that we have observed in practice.

For a given dimension m of the DEIM approximation, the term $\mathcal{C}$ is constant for any vector $\mathbf{f}$ and hence it applies to the approximation to any sample $\mathbf{f}(\tau) = \mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(\tau))$, where τ represents time t in Equation (2) or parameter vector μ in Equation(3). However, the best approximation error

$$\mathcal{E}_* = \mathcal{E}_*(\mathbf{f}(\tau))$$

is dependent upon $\mathbf{f}(\tau)$ and changes with each new value of τ. This would be quite expensive to compute, so an easily computable estimate is highly desirable. A reasonable estimate is available with the SVD of the nonlinear snapshot matrix

$$\widehat{\mathbf{F}} = [\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^{n_s}].$$

Let $\mathcal{F} = \mathrm{Range}(\widehat{\mathbf{F}})$ and let $\widehat{\mathbf{F}} = \widehat{\mathbf{U}}\widehat{\boldsymbol{\Sigma}}\widehat{\mathbf{W}}^T$ its SVD. where $\widehat{\mathbf{U}} = [\mathbf{U}, \widetilde{\mathbf{U}}]$ where $\mathbf{U}$ represents the leading m columns of the orthogonal matrix $\widehat{\mathbf{U}}$. Partition $\widehat{\Sigma} = \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0} & \widetilde{\Sigma} \end{bmatrix}$ to conform with the partitioning of $\widehat{\mathbf{U}}$. The singular values are ordered as usual with $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_m \geq \sigma_{m+1} \geq \cdots \geq \sigma_n \geq 0$. The diagonal matrix Σ has the leading m singular values on its diagonal. The orthogonal matrix $\widehat{\mathbf{W}} = [\mathbf{W}, \widetilde{\mathbf{W}}]$ is partitioned accordingly. Any vector $\mathbf{f} \in \mathcal{F}$ may be written in the form

$$\mathbf{f} = \widehat{\mathbf{F}}\hat{\mathbf{g}} = \mathbf{U}\Sigma\mathbf{g} + \widetilde{\mathbf{U}}\widetilde{\Sigma}\tilde{\mathbf{g}},$$

where $\mathbf{g} = \mathbf{W}^T\hat{\mathbf{g}}$ and $\tilde{\mathbf{g}} = \widetilde{\mathbf{W}}^T\hat{\mathbf{g}}$. Thus

$$\|\mathbf{f} - \mathbf{f}_*\|_2 = \|(\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{f}\|_2 = \|\widetilde{\mathbf{U}}\widetilde{\Sigma}\tilde{\mathbf{g}}\|_2 \leq \sigma_{m+1}\|\tilde{\mathbf{g}}\|_2.$$

For vectors $\mathbf{f}$ nearly in $\mathcal{F}$, we have $\mathbf{f} = \widehat{\mathbf{F}}\hat{\mathbf{g}} + \mathbf{w}$ with $\mathbf{w}^T\widehat{\mathbf{F}}\hat{\mathbf{g}} = 0$, and thus

$$\mathcal{E}_* = \mathcal{E}_*(\mathbf{f}) \approx \sigma_{m+1}. \tag{40}$$

is reasonable so long as $\|\mathbf{w}\|_2$ is small ($\|\mathbf{w}\|_2 = \mathcal{O}(\sigma_{m+1})$ ideally). The POD approach (and hence the resulting DEIM approach) is most successful when the trajectories are attracted to a low dimensional subspace (or manifold). Hence, the vectors $\mathbf{f}(\tau)$ should nearly lie in $\mathcal{F}$ and this approximation will then serve for all of them.

To illustrate the error bound for DEIM approximation, we next present numerical examples on some examples of nonlinear parametrized functions defined on 1D and 2D discrete spatial points. These experiments show that the approximate error bound using σ_{m+1} in place of $\mathcal{E}_*$ is quite reasonable in practice.

3.3 Numerical Examples of the DEIM Error Bound

This section demonstrates the accuracy and efficiency of the approximation from DEIM as well as its error bound given in Section 3.2. The examples here use the POD basis in the DEIM approximation. The POD basis is constructed from a set of *snapshots* corresponding to a selected set of elements in $\mathscr{D}$. In particular, we define

$$\mathscr{D}^s = \{\mu_1^s, \ldots, \mu_{n_s}^s\} \subset \mathscr{D} \tag{41}$$

used for constructing a set of snapshots given by

$$\mathcal{S} = \{\mathbf{f}(\mu_1^s), \ldots, \mathbf{f}(\mu_{n_s}^s)\}, \tag{42}$$

which is used for computing the POD basis $\{\mathbf{u}_\ell\}_{\ell=1}^m$ for the DEIM approximation.
To evaluate the accuracy, we apply the DEIM approximation $\hat{\mathbf{f}}$ in (16) to the elements in the set

$$\bar{\mathscr{D}} = \{\bar{\mu}_1, \ldots, \bar{\mu}_{\bar{n}}\} \subset \mathscr{D}, \tag{43}$$

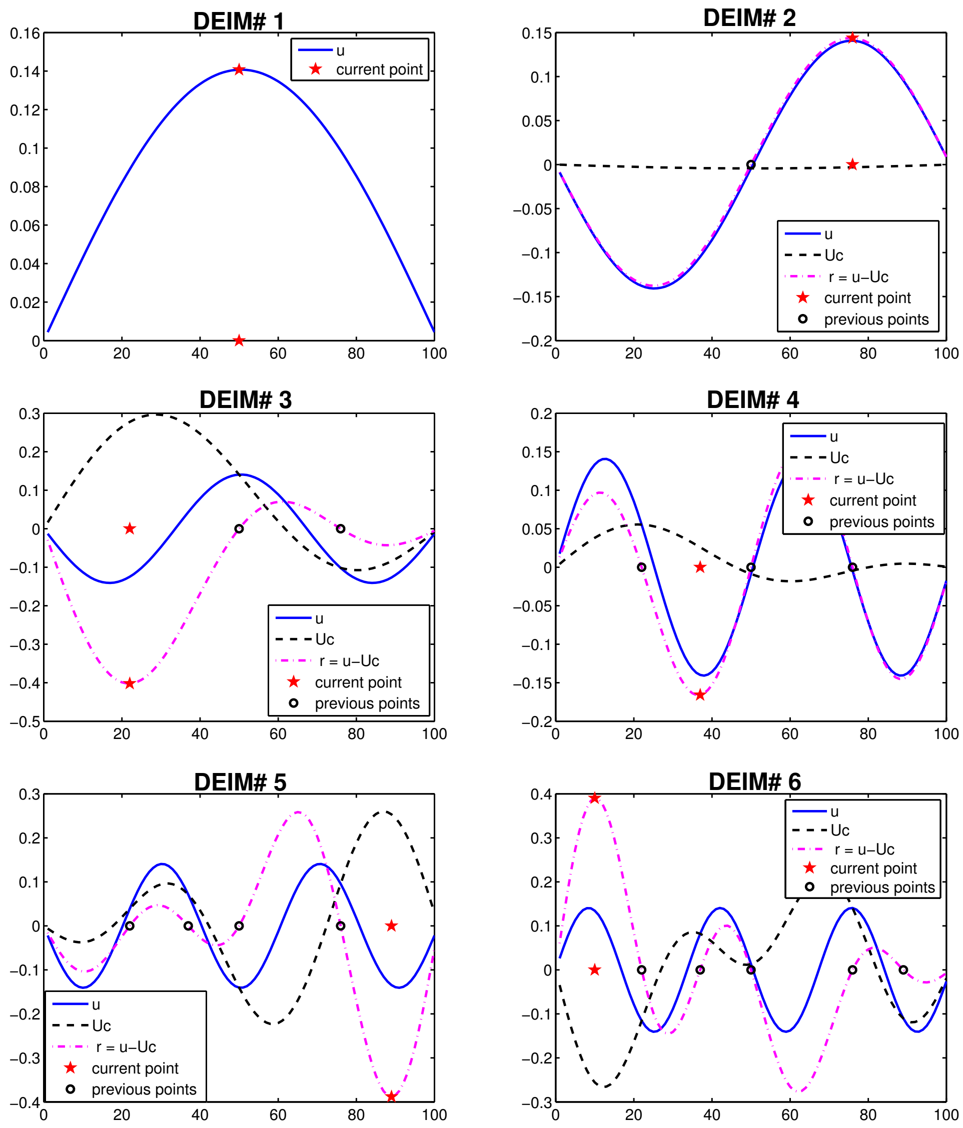


Figure 1: Illustration of the selection process of indices in Algorithm 1 for the DEIM approximation. The input basis vectors are the first 6 eigenvectors of the discrete Laplacian. From the plots, $\mathbf{u} = \mathbf{u}_\ell$, $\mathbf{Uc}$ and $\mathbf{r} = \mathbf{u}_\ell - \mathbf{Uc}$ are defined as in the iteration ℓ of Algorithm 1.

which is different from and larger than the set $\mathscr{D}^s$ used for the snapshots. Then we consider the average error for *DEIM approximation* $\hat{\mathbf{f}}$ over the elements in $\bar{\mathscr{D}}$ given by

$$\bar{\mathcal{E}}(\mathbf{f}) = \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \|\mathbf{f}(\bar{\mu}_i) - \hat{\mathbf{f}}(\bar{\mu}_i)\|_2, \tag{44}$$

the average POD error in (20) for *POD approximation* $\hat{\mathbf{f}}_*$ from (23) over the elements in $\bar{\mathscr{D}}$ given by

$$\bar{\mathcal{E}}_*(\mathbf{f}) = \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \|\mathbf{f}(\bar{\mu}_i) - \hat{\mathbf{f}}_*(\bar{\mu}_i)\|_2 = \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \mathscr{E}_*(\mathbf{f}(\bar{\mu}_i)). \tag{45}$$

The error bound given in (19) guarantees the existence $(\mathbf{P}^T\mathbf{U})^{-1}$ and therefore, in this section, instead of using $\mathcal{C}$ given in Lemma 3.2, we use $\bar{\mathcal{C}} := \|(\mathbf{P}^T\mathbf{U})^{-1}\|_2$ directly to compute the *average error bound* which is then given by

$$\frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \|\mathbf{f}(\bar{\mu}_i) - \hat{\mathbf{f}}(\bar{\mu}_i)\|_2 \leq \bar{\mathcal{C}} \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \mathscr{E}_*(\mathbf{f}(\bar{\mu}_i)), \tag{46}$$

with the corresponding approximation using (40):

$$\frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \|\mathbf{f}(\bar{\mu}_i) - \hat{\mathbf{f}}(\bar{\mu}_i)\|_2 \lesssim \bar{\mathcal{C}} \sigma_{m+1}. \tag{47}$$

3.3.1 Nonlinear parametrized function with spatial points in 1D

Consider a nonlinear parametrized function $s : \Omega \times \mathscr{D} \mapsto \mathbb{R}$ defined by

$$s(x; \mu) = (1 - x) cos(3\pi\mu(x + 1)) e^{-(1+x)\mu}, \tag{48}$$

where $x \in \Omega = [-1, 1]$ and $\mu \in \mathscr{D} = [1, \pi]$. This nonlinear function is from an example in [7]. Let $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$, with x_i equidistantly spaced points in Ω, for $i = 1, \ldots, n$, $n = 100$. Define $\mathbf{f} : \mathscr{D} \mapsto \mathbb{R}^n$ by

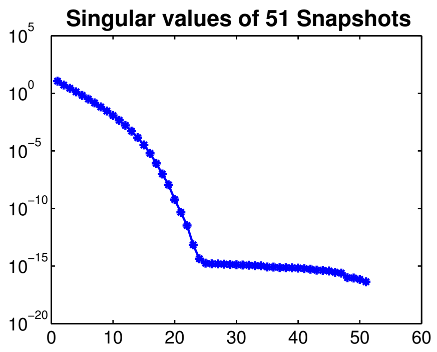
$$\mathbf{f}(\mu) = [s(x_1; \mu), \ldots, s(x_n; \mu)]^T \in \mathbb{R}^n, \tag{49}$$

for $\mu \in \mathscr{D}$. This example uses 51 snapshots $\mathbf{f}(\mu_j^s)$ to construct POD basis $\{\mathbf{u}_\ell\}_{\ell=1}^m$ with $\mu_1^s, \ldots, \mu_{51}^s$ are selected from equally spaced points in $[1, \pi]$. Figure 2 shows the singular values of these snapshots and the corresponding first 6 POD basis vectors with the first 6 spatial points selected from the DEIM algorithm using this POD basis as an input. Figure 3 compares the approximate functions from DEIM of dimension 10 with the original function of dimension 100 at different values of $\mu \in \mathscr{D}$. This demonstrates that DEIM gives a good approximation at arbitrary values $\mu \in \mathscr{D}$. Figure 4 illustrates the average errors defined in (44) and (45), the average error bound and its approximation computed from the right hand side of (46) and (47), respectively, with $\bar{\mu}_1, \ldots, \bar{\mu}_{\bar{n}} \in \bar{\mathscr{D}}$ selected uniformly over $\mathscr{D}$ and $\bar{n} = 101$.

3.3.2 Nonlinear parametrized function with spatial points in 2D

Consider a nonlinear parametrized function $s : \Omega \times \mathscr{D} \mapsto \mathbb{R}$ defined by

$$s(x, y; \mu) = \frac{1}{\sqrt{(x - \mu_1)^2 + (y - \mu_2)^2 + 0.1^2}}, \tag{50}$$



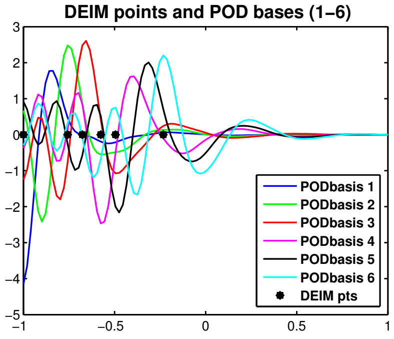


Figure 2: Singular values and the corresponding first 6 POD bases with DEIM points of snapshots from (49)

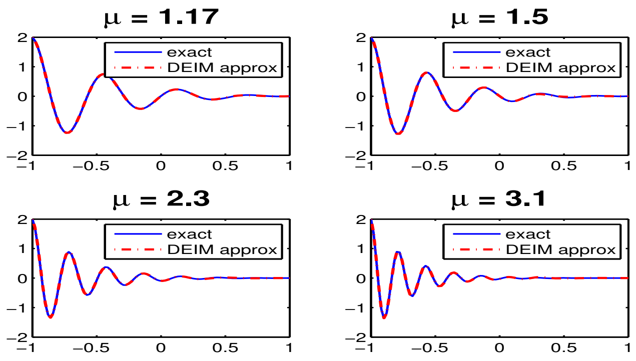


Figure 3: The approximate functions from DEIM of dimension 10 compared with the original functions (49) of dimension $n = 100$ at $\mu = 1.17, 1.5, 2.3, 3.1$.

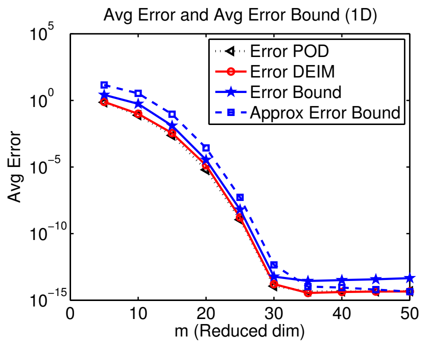
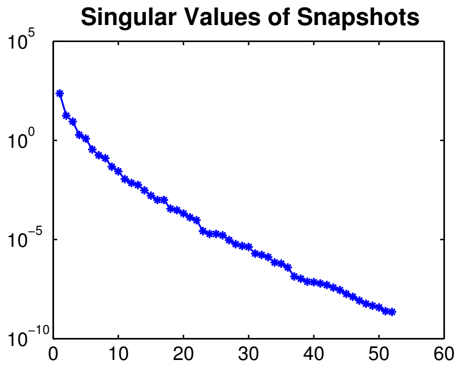


Figure 4: Compare average errors of POD and DEIM approximations for (49) with the average error bounds and their approximations given in (46) and (47), respectively.

where $(x, y) \in \Omega = [0.1, 0.9]^2 \subset \mathbb{R}^2$ and $\mu = (\mu_1, \mu_2) \in \mathscr{D} = [-1, -0.01]^2 \subset \mathbb{R}^2$. This example is modified from one given in [3]. Let (x_i, y_j) be uniform grid points in Ω, for $i = 1, \ldots, n_x$ and $j = 1, \ldots, n_y$. Define $\mathbf{s} : \mathscr{D} \mapsto \mathbb{R}^{n_x \times n_y}$ by

$$\mathbf{s}(\mu) = [s(x_i, y_j; \mu)] \in \mathbb{R}^{n_x \times n_y} \tag{51}$$

for $\mu \in \mathscr{D}$ and $i = 1, \ldots, n_x$, and $j = 1, \ldots, n_y$. In this example, the full dimension is $n = n_x n_y = 400$ ($n_x = n_y = 20$). Note that we can define a corresponding vector-valued function $\mathbf{f} : \mathscr{D} \mapsto \mathbb{R}^n$ for this problem, e.g. by reshaping the matrix $\mathbf{s}(\mu)$ to a vector of lenght $n = n_x n_y$. The 225 snapshots constructed from uniformly selected parameters $\mu^s = (\mu_1^s, \mu_2^s)$ in parameter domain $\mathscr{D}$ are used for constructing the POD basis. The different 625 pairs of parameters μ are used for testing (error and cpu time). Figure 5 shows the singular values of these snapshots and the corresponding first 6 POD basis vectors. Figure 6 illustrates the distribution of the first 20 spatial points selected from the DEIM algorithm using this POD basis as an input. Notice that most of the selected points cluster close to the origin, which the singularity of the function s. Figure 7 shows that the approximate functions from DEIM of dimension 6 can reproduce the original function of dimension 400 very well at different values of $\mu \in \mathscr{D}$. Figure 8 gives the average errors with the bounds from the last section and the corresponding average CPU times for different dimensions of POD and DEIM approximations. The average errors of POD and DEIM approximations are computed from (44) and (45), respectively. The average error bounds and their approximations are computed from the right hand side of (46) and (47), respectively. This example uses $\bar{\mu}_1, \ldots, \bar{\mu}_{\bar{n}} \in \bar{\mathscr{D}}$ selected uniformly over $\mathscr{D}$ and $\bar{n} = 625$. The CPU times are averaged over the same set $\bar{\mathscr{D}}$.



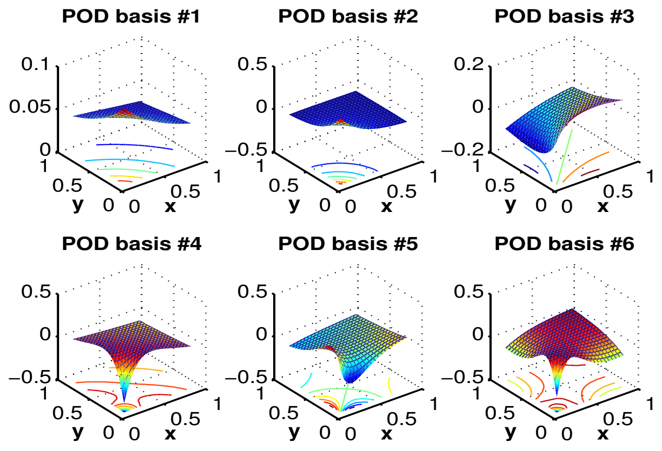


Figure 5: Singular values and the first 6 corresponding POD basis vectors of the snapshots of the nonlinear function (51).

3.4 Application of DEIM to MOR for Nonlinear FD Discretized Systems

The DEIM approximation (18) developed in the previous section may now be used to approximate the nonlinear term in (10) and the Jacobian in (11) with nonlinear approximations having computational complexity proportional to the number of reduced variables obtained with POD.

In the case of unsteady nonlinear PDEs, from (18), set $\tau = t$ and $\mathbf{f}(t) = \mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(t))$, then the nonlinear function in (6) approximated by DEIM can be written as

$$\begin{aligned} \mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(t)) &\approx \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(t)) && (52)\\ &= \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{F}(\mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}}(t)). && (53) \end{aligned}$$

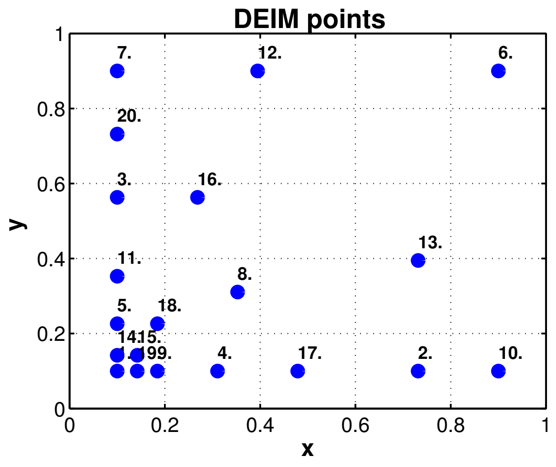


Figure 6: First 20 points selected by DEIM for the nonlinear function (51).

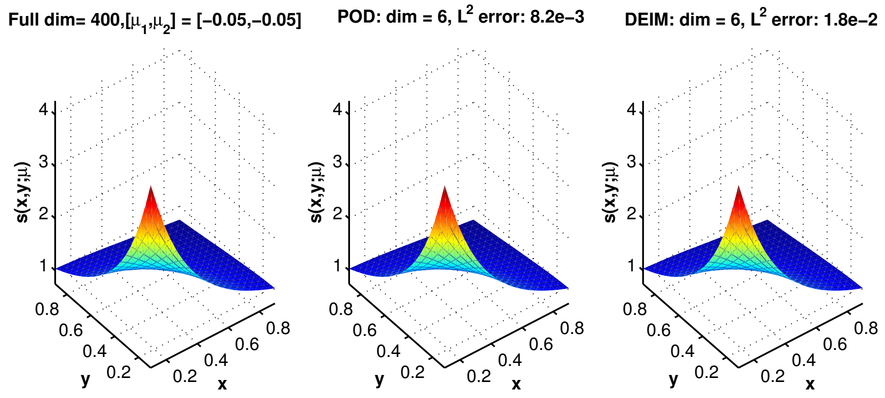


Figure 7: Compare the original nonlinear function (51) of dimension 400 with the POD and DEIM approximations of dimension 6 at parameter $\mu = [-0.05, -0.05]$.

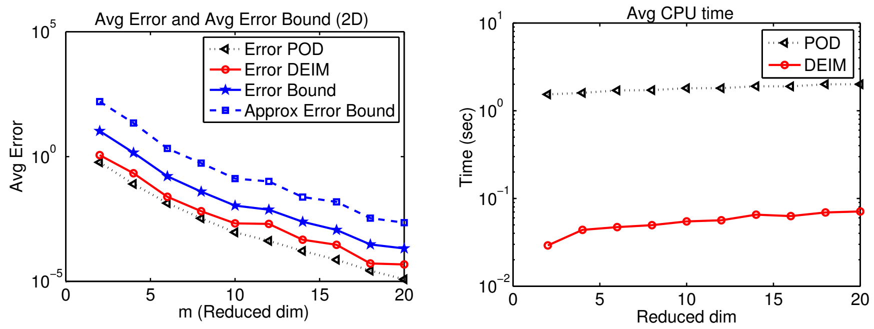


Figure 8: Left: Average errors of POD and DEIM approximations for (51) with the average error bounds given in (46) and their approximations given in (47). Right: Average cpu time for evaluating the POD and DEIM approximations.

The last equality in (53) follows from the fact that the function $\mathbf{F}$ evaluates componentwise at its input vector. The nonlinear term in (10) then can be approximated by

$$\tilde{\mathbf{N}}(\tilde{\mathbf{y}}) \approx \underbrace{\mathbf{V}_k^T\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}}_{\text{precomputed}:k\times m}\underbrace{\mathbf{F}(\mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}}(t))}_{m\times 1}. \tag{54}$$

Note that the term $\mathbf{V}_k^T\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}$ in (54) does not depend on t and therefore it can be precomputed before solving the system of ODEs. Note also that $\mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}}(t) \in \mathbb{R}^m$ in (54) can be obtained by extracting the rows $\wp_1, \ldots, \wp_m$ of $\mathbf{V}_k$ and then multiplying to $\tilde{\mathbf{y}}$, which requires $2mk$ operations. Therefore the complexity for computing this approximation of the nonlinear term roughly becomes $\mathcal{O}(\alpha(m) + 4km)$, which is independent of dimension n of the full-order system (2).

Similarly, in the case of steady parametrized nonlinear PDEs, from (18), set $\tau = \mu$ and $\mathbf{f}(\mu) = \mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(\mu))$. Then the nonlinear function in (7) approximated by DEIM can be written as

$$\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}(\mu)) \quad\approx\quad \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{F}(\mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}}(\mu)), \tag{55}$$

and the approximation for Jacobian of the nonlinear term (11) is of the form

$$\tilde{\mathbf{J}}_{\mathbf{F}}(\tilde{\mathbf{y}}(\mu)) \approx \underbrace{\mathbf{V}_k^T\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}}_{\text{precomputed}:k\times m}\underbrace{\mathbf{J}_{\mathbf{F}}(\mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}}(\mu))}_{m\times m}\underbrace{\mathbf{P}^T\mathbf{V}_k}_{m\times k}, \tag{56}$$

where

$$\mathbf{J}_{\mathbf{F}}(\mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}}(\mu)) = \operatorname{diag}\{F'(\mathbf{y}_1^r(\mu)), \ldots, F'(\mathbf{y}_m^r(\mu))\},$$

and $\mathbf{y}^r(\mu) = \mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}}(\mu)$, which can be computed with complexity independent of n as noted earlier. Therefore, the computational complexity for the approximation in (56) is roughly $\mathcal{O}(\alpha(m) + 4mk + 2mk^2)$.

The approximations from DEIM are now in the form of (54) and (56) that recover the computational efficiency of (10) and (11), respectively.

Note that the nonlinear approximation from DEIM in (53) and (55) are obtained by exploiting the special structure of the nonlinear function $\mathbf{F}$ being evaluated componentwise at $\mathbf{y}$. The next section provides a completely general scheme.

3.5 Interpolation of General Nonlinear Functions

The very simple case of $\mathbf{F}(\mathbf{y}) = [F(\mathbf{y}_1), \ldots, F(\mathbf{y}_n)]^T$, has been discussed for purposes of illustration and is indeed important in its own right. However, DEIM extends easily to general nonlinear functions. MATLAB notation is used here to explain this generalization.

$$[\mathbf{F}(\mathbf{y})]_i = \mathbf{F}_i(\mathbf{y}) = F_i(\mathbf{y}_{j_1^i}, \mathbf{y}_{j_2^i}, \mathbf{y}_{j_3^i}, \ldots, \mathbf{y}_{j_{n_i}^i}) = F_i(\mathbf{y}(\mathbf{p}_i)), \tag{57}$$

where $F_i : \mathcal{Y}_i \to \mathbb{R}$, $\mathcal{Y}_i \subset \mathbb{R}^{n_i}$. The integer vector $\mathbf{p}_i = [j_1^i, j_2^i, j_3^i, \ldots, j_{n_i}^i]^T$ denotes the indices of the subset of components of $\mathbf{y}$ required to evaluate the i-th component of $\mathbf{F}(\mathbf{y})$) for $i = 1, \ldots, n$. Suppose the complexity of evaluating the nonlinear scalar valued function F_i of the n_i variables indexed by $\mathbf{p}_i$ is β_i Flops. Then the total computational complexity for (57) is $\sum_{i=1}^n \beta_i$ Flops.

The nonlinear function of the reduced-order system obtained from POD-Galerkin method by projecting on the space spanned by columns of $\mathbf{V}_k \in \mathbb{R}^{n\times k}$ is in the form of $\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}})$, where the components of $\tilde{\mathbf{y}} \in \mathbb{R}^k$ are the reduced variables. Recall that the DEIM approximation of order m for $\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}})$ is given by

$$\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}) \approx \underbrace{\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}}_{k\times m}\underbrace{\mathbf{P}^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}})}_{m\times 1}, \tag{58}$$

where $\mathbf{U} \in \mathbb{R}^{n\times m}$ is the projection matrix for the nonlinear function $\mathbf{F}$, $\mathbf{P} = [\mathbf{e}_{\wp_1}, \dots, \mathbf{e}_{\wp_m}] \in \mathbb{R}^{n\times m}$, and $\wp_1, \dots, \wp_m$ are interpolation indices from the DEIM Algorithm. In the simple case when $\mathbf{F}$ is evaluated componentwise at $\mathbf{y}$, we have $\mathbf{P}^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}) = \mathbf{F}(\mathbf{P}^T\mathbf{V}_k\tilde{\mathbf{y}})$ where $\mathbf{P}^T\mathbf{V}_k$ can be obtained by extracting rows of $\mathbf{V}_k$ corresponding to $\wp_1, \dots, \wp_m$ and hence its computational complexity is independence of n. However, this is clearly not applicable to the general nonlinear vector-valued function. This section provides an efficient method for computing $\mathbf{P}^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}})$ in the DEIM approximation (58) for a general nonlinear function.

Notice that, since $\mathbf{y}_j \approx \mathbf{V}_k(j,:)\tilde{\mathbf{y}}$, an approximation to $\mathbf{F}(\mathbf{y})$ is provided by

$$\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}) = [F_1(\mathbf{V}_k(\mathbf{p}_1,:)\tilde{\mathbf{y}}), \dots, F_n(\mathbf{V}_k(\mathbf{p}_n,:)\tilde{\mathbf{y}})]^T \in \mathbb{R}^n, \tag{59}$$

and thus

$$\mathbf{P}^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}) = [F_{\wp_1}(\mathbf{V}_k(\mathbf{p}_{\wp_1},:)\tilde{\mathbf{y}}), \dots, F_{\wp_m}(\mathbf{V}_k(\mathbf{p}_{\wp_m},:)\tilde{\mathbf{y}})]^T \in \mathbb{R}^m. \tag{60}$$

The complexity for evaluating each component $\wp_i$, $i = 1, \dots, m$, of (60):

$$\tilde{F}_{\wp_i}(\tilde{\mathbf{y}}) := F_{\wp_i}(\mathbf{V}_k(\mathbf{p}_{\wp_i},:)\tilde{\mathbf{y}}) \tag{61}$$

is $n_{\wp_i} \times k$ Flops plus the complexity of evaluating the nonlinear scalar valued function $F_{\wp_i}$ of the $n_{\wp_i}$ variables indexed by $\mathbf{p}_{\wp_i}$. That is, the total complexity for (60) is $(k \times \sum_{i=1}^m n_{\wp_i}) + \sum_{i=1}^m \beta_{\wp_i}$ Flops. Therefore, $n_{\wp_i} \ll n$ is required in all components in order for (60) to be an effective reduction of complexity. In particular, $\sum_{i=1}^m n_{\wp_i} < n$ must hold. Otherwise, it will be more efficient to just evaluate $\hat{\mathbf{y}} = \mathbf{V}_k\tilde{\mathbf{y}} \in \mathbb{R}^n$ and just compute the required m components $F_{\wp_i}(\hat{\mathbf{y}})$ directly.

The sparse evaluation procedure may be implemented using a *compressed sparse row* data structure as used in sparse matrix factorizations. Two linear integer arrays are needed, $irstart$ is a vector of length $m+1$ containing pointers to locations in the vector $jrow$ which is of length $n_{\vec{\wp}} = \sum_{i=1}^m n_{\wp_i}$. The successive n_i entries of $jrow(irstart(i)$ indicate the dependence of the i component of $\mathbf{F}(\mathbf{y})$ on the selected variables from $\mathbf{y}$. In particular,

- $irstart(i)$ contains location of start of i-th row with $irstart(m+1) = n_{\vec{\wp}} + 1$.
  I.e., $irstart(1) = 1$, and $irstart(i) = 1 + \sum_{j=1}^{i-1} n_{\wp_j}$ for $i = 2, \dots, m+1$.
- $jrow$ contains the indices of the $\mathbf{y}$ variables required to compute the $\wp_i$-th function $F_{\wp_i}$ in locations $irstart(i)$ to $irstart(i+1) - 1$, for $i = 1, \dots, m$. I.e.,

$$jrow \quad = \quad [\underbrace{\overset{\overset{irstart(1)}{\downarrow}}{\boxed{j_1^{\wp_1}}}, \dots, j_{n_{\wp_1}}^{\wp_1}}_{\mathbf{p}_{\wp_1}},\ \underbrace{\overset{\overset{irstart(2)}{\downarrow}}{\boxed{j_1^{\wp_2}}}, \dots, j_{n_{\wp_2}}^{\wp_2}}_{\mathbf{p}_{\wp_2}},\ \dots,\ \underbrace{\overset{\overset{irstart(m)}{\downarrow}}{\boxed{j_1^{\wp_m}}}, \dots, j_{n_{\wp_m}}^{\wp_m}}_{\mathbf{p}_{\wp_m}}]^T \quad \in \mathbb{Z}_+^{n_{\vec{\wp}}}.$$

Given $\mathbf{V}_k$ and $\tilde{\mathbf{y}}$, the following demonstrates how to compute the approximation $\tilde{F}_{\wp_i}(\tilde{\mathbf{y}})$ in (61), for $i = 1, \dots, m$, from the vectors $irstart$ and $jrow$.

```
for $i = 1 : m$
    $\mathbf{p}_{\wp_i} = jrow(irstart(i) : irstart(i+1) - 1)$
    $\tilde{F}_{\wp_i}(\tilde{\mathbf{y}}) = F_{\wp_i}(\mathbf{V}_k(\mathbf{p}_{\wp_i},:)\tilde{\mathbf{y}})$
end
```

The next section will provide a detailed analysis for the computational complexity of the systems given earlier in (2) and (3) with their corresponding reduced systems obtained from POD-Galerkin method and POD with DEIM approximation.

3.6 Computational Complexity

Recall that the POD-DEIM reduced system for the unsteady nonlinear problem (2) is

$$\frac{d}{dt}\tilde{\mathbf{y}}(t) = \tilde{\mathbf{A}}\tilde{\mathbf{y}}(t) + \mathbf{B}\ \mathbf{F}(\mathbf{V}_{\vec{\wp}}\tilde{\mathbf{y}}(t)), \tag{62}$$

and the approximation for the steady state problem (3) is given by

$$\tilde{\mathbf{A}}\tilde{\mathbf{y}}(t) + \mathbf{B}\ \mathbf{F}(\mathbf{V}_{\vec{\wp}}\tilde{\mathbf{y}}(t)) = 0, \tag{63}$$

where $\tilde{\mathbf{A}} = \mathbf{V}_k^T\mathbf{A}\mathbf{V}_k \in \mathbb{R}^{k\times k}$, and $\mathbf{B} = \mathbf{V}_k^T\mathbf{U}\mathbf{U}_{\vec{\wp}}^{-1} \in \mathbb{R}^{k\times m}$ with $\mathbf{U}_{\vec{\wp}} = \mathbf{P}^T\mathbf{U}$ and $\mathbf{V}_{\vec{\wp}} = \mathbf{P}^T\mathbf{V}_k$. This section summarizes the computational complexity for constructing and solving these POD-DEIM reduced systems compared to the both the original full-order systems and the *POD reduced systems*, obtained directly from POD-Galerkin projection. A POD-DEIM reduced system is constructed by first collecting a set of n_s solution snapshots, which can be used for generating a corresponding set of n_s nonlinear snapshots. Next, POD bases are constructed for each of the matrices of solution snapshots and nonlinear snapshots via the SVD. The first k dominant POD basis vectors of the snapshot solutions are used to form a projection matrix $\mathbf{V}_k$ and the first m dominant POD basis vectors of the nonlinear snapshots are used to form a projection matrix $\mathbf{U}$ for the nonlinear approximation. A set of m interpolation indices are obtained from $\mathbf{U}$ through the DEIM algorithm. Finally, the constant coefficient matrices $\tilde{\mathbf{A}}$ and $\mathbf{B}$ are precomputed for the linear and nonlinear terms in the reduced system. The following table gives the computational complexity for these steps.

Procedure	Complexity
Snapshots	Problem dependence
SVD: POD basis	$\mathcal{O}(nn_s^2)$
DEIM Algorithm: m interpolation indices	$\mathcal{O}(m^4 + mn)$
Pre-compute: $\tilde{\mathbf{A}} = \mathbf{V}_k^T\mathbf{A}\mathbf{V}_k$	$\begin{cases} \mathcal{O}(n^2k + nk^2), & \text{for dense } \mathbf{A} \\ \mathcal{O}(nk + nk^2), & \text{for sparse } \mathbf{A} \end{cases}$
Pre-compute: $\mathbf{B} = \mathbf{V}_k^T\mathbf{U}\mathbf{U}_{\vec{\wp}}^{-1}$	$\mathcal{O}(nkm + m^2n + m^3)$

Table 1: Computational complexity for constructing a POD-DEIM reduced-order system.

Note that for large snapshot sets, it is far more efficient to compute the dominant singular values and vectors iteratively via ARPACK (or `eigs` in MATLAB) [6]. The steps in Table 1 have to be done only once before solving the POD-DEIM reduced systems. The constant coefficient matrices $\tilde{\mathbf{A}}$ and $\mathbf{B}$ are pre-computed, stored, and reused during solving the reduced systems. The systems (62) and (63) are generally solved by iterative scheme.

Tables 2 and Table 3 give the computational complexity for each iteration when solving (62) by forward Euler method and (63) by Newton's method, respectively. The corresponding plots of these tables are shown in Figures 9 and 11. Note that each plot in Figures 9 to 12 is scaled so that the value of the Flops or the CPU time for the *sparse* full-order system (sparse in coefficient matrix $\mathbf{A}$) is equal to 1. Note also that $\alpha(p)$ is the Flops for evaluating the nonlinear function $\mathbf{F}$ at p components and $\alpha^d(p)$, used in Table 3, is the Flops for evaluating derivative of the nonlinear function $\mathbf{F}$ at p components. When $\mathbf{F}$ evaluates at its input vector componentwise, $\alpha(p)$ and $\alpha^d(p)$ are linear in p. In this case, the computational complexities for evaluating one forward Euler time step and performing one Newton iteration of full-order system, POD reduced system, and POD-DEIM reduced system are shown in the last columns of Table 2 and Table 3, respectively.

Although the forward (explicit) Euler method may not be the best approach due to the step limiting stability issue, its cost per iteration is typical of other explicit methods and hence it is suitable for illustration purposes. An implicit scheme would require solution of a nonlinear system at each time step. The computational complexity for each Newton iteration is shown in Table 3. In practice, the CPU time may not be directly proportional to these predicted Flops since there are many other factors that might affect the CPU timings. However, this analysis does reflect the relative computational requirements and may be useful for predicting expected relative computational times and performance enhancements possible with DEIM.

When $\mathbf{A} \in \mathbb{R}^{n \times n}$ represents the discretization of a linear differential operator, it is usually sparse. Then, from Table 2, we can employ the sparsity of $\mathbf{A}$ so that the total complexity for each iteration of full-order system becomes $\mathcal{O}(n)$ instead of $\mathcal{O}(n^2)$. Similarly, from Table 3, the total complexity becomes $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^3)$. In this case, the total complexity of the POD reduced system can be higher than the complexity of the full-order system as shown in Figures 9 and 11. E.g., the results in Figure 11 for the steady state problem with dimension of (sparse) full-order system $n = 2500$, indicate that roughly when $k = 50$ or $nk^2 = n^2$, the computational time of POD reduced system starts to exceed the computational time of the full-order system. This follows from Table 3 that the complexity $\mathcal{O}(k^3 + nk^2)$ for POD reduced system is equivalent to the complexity $\mathcal{O}(n^2)$ for the sparse full-order system when $k^2 \approx n$.

This inefficiency of POD reduced system indeed occurs in the actual computation as shown in Figures 10 and 12. From Figure 10 for the unsteady nonlinear system, the CPU time of the POD reduced system used for computing each time step exceeds the CPU time for the original system as soon as its dimension reaches 30. The same phenomenon happens for the POD reduced system of the steady-state problem as shown in Figure 12, which illustrates the (scaled) CPU time of the highly nonlinear 2-D steady state problem introduced later in Section 4. The corresponding POD-DEIM reduced system with both POD and DEIM having dimension 15 is order $\mathcal{O}(100)$ faster than the original system with $\mathcal{O}(10^{-4})$ accuracy. On the other hand, the POD reduced system of dimension 15 gives only $\mathcal{O}(10)$ reduction in CPU time from the original system with roughly the same order of accuracy as the POD-DEIM reduced system. These demonstrate the inefficiency of the POD reduced system that has been remedied by the introduction of DEIM.

System	Computation in forward Euler	Complexity (1 time step)	Total Complexity For linear $\alpha(\cdot), \alpha^d(\cdot)$
Full	$\mathbf{y} \leftarrow \mathbf{y} + dt(\mathbf{A}\mathbf{y} + \mathbf{F}(\mathbf{y}))$	$2n^2 + 2n + \alpha(n)$ or $cn + \alpha(n)$ $c \sim$ sparsity of $\mathbf{A}$	► $\mathcal{O}(n^2)$ ► Sparse $\mathbf{A}$: $\mathcal{O}(n)$ ► Sparse $\mathbf{A}$ (MATLAB): $\mathcal{O}(n\log(n))$
POD	$\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} + dt(\tilde{\mathbf{A}}\tilde{\mathbf{y}} + \mathbf{V}_k^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}}))$	$2k^2 + 2k + \alpha(n) + 4nk$	$\mathcal{O}(k^2 + nk)$
POD-DEIM	$\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} + dt(\tilde{\mathbf{A}}\tilde{\mathbf{y}} + \mathbf{B}\mathbf{F}(\mathbf{V}_{\vec{\wp}}\tilde{\mathbf{y}}))$	$2k^2 + 2k + \alpha(m) + 4mk$	$\mathcal{O}(k^2 + mk)$

Table 2: Compare the computational complexity for each time step of forward Euler

4 Numerical Results

The efficiency and accuracy of the approximation from DEIM will be demonstrated through two problems. The first is a 1D unsteady nonlinear PDE arising in neuron modeling. The second is a highly nonlinear 2-D steady state problem whose solution is obtained by solving its FD discretized

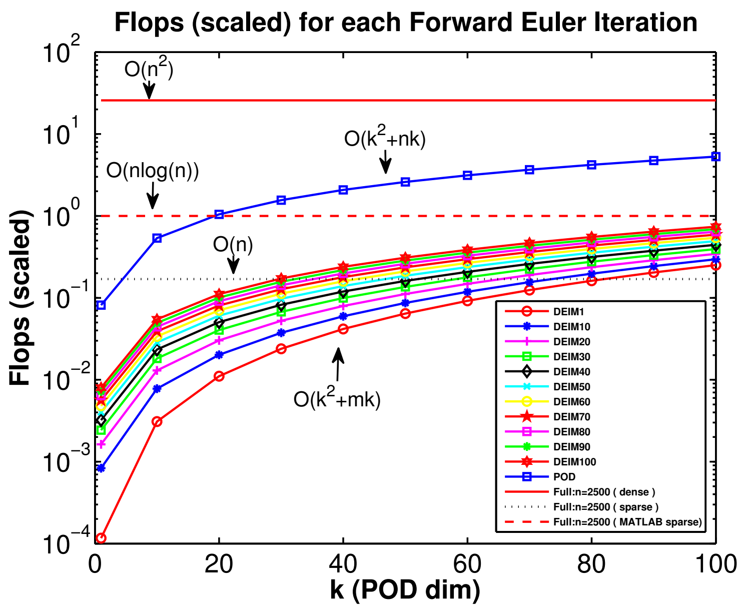


Figure 9: Approximate Flops (scaled with the Flops for full-sparse system) in each time step of forward Euler.

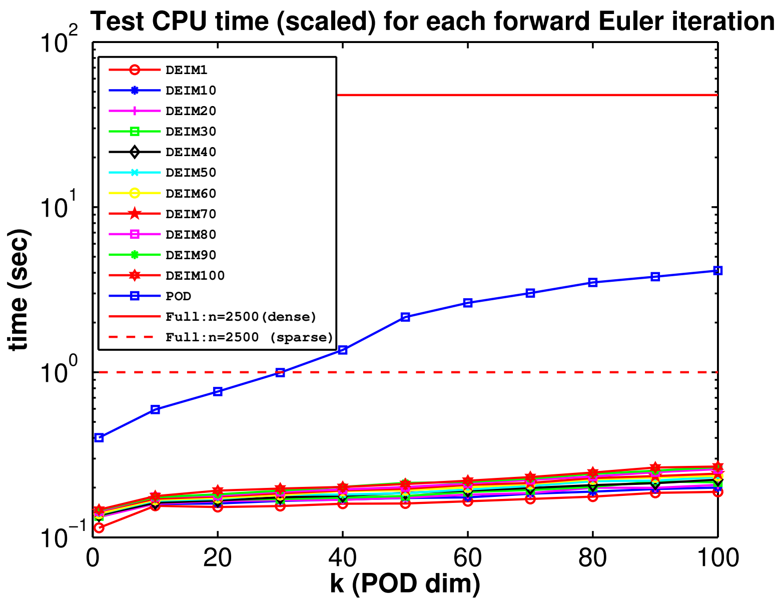


Figure 10: Average CPU time (scaled with the CPU time for full-sparse system) in each time step of forward Euler.

System	Computation in Newton iteration	Complexity (1 iteration)	Total Complexity For linear $\alpha(\cdot), \alpha^d(\cdot)$
Full	$\mathbf{G}(\mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{F}(\mathbf{y})$ $\mathbf{J}(\mathbf{y}) = \mathbf{A} + \text{diag}\{\mathbf{F}'(\mathbf{y})\}$ $\mathbf{y} \leftarrow \mathbf{y} - \mathbf{J}(\mathbf{y})^{-1}\mathbf{G}(\mathbf{y})$	$2n^2 + \alpha(n) + n$ or $cn + \alpha(n)$ $n^2 + \alpha^d(n)$ or $n + \alpha^d(n)$ $\mathcal{O}(n^3)$ or $\mathcal{O}(n^2)$ $c \sim$ sparsity of $\mathbf{A}$	$\mathcal{O}(n^3)$ Sparse: $\mathcal{O}(n^2)$
POD	$\tilde{\mathbf{G}}(\mathbf{y}) = \tilde{\mathbf{A}}\tilde{\mathbf{y}} + \mathbf{V}_k^T\mathbf{F}(\mathbf{V}_k\tilde{\mathbf{y}})$ $\tilde{\mathbf{J}}(\mathbf{y}) = \tilde{\mathbf{A}} + \mathbf{V}_k^T\text{diag}\{\mathbf{F}'(\mathbf{V}_k\tilde{\mathbf{y}})\}\mathbf{V}_k$ $\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} - \tilde{\mathbf{J}}(\mathbf{y})^{-1}\tilde{\mathbf{G}}(\mathbf{y})$	$2k^2 + \alpha(n) + k + 4nk$ $k^2 + \alpha^d(n) + 4nk + 2nk^2$ $\mathcal{O}(k^3)$	$\mathcal{O}(k^3 + nk^2)$
POD-DEIM	$\tilde{\mathbf{G}}(\mathbf{y}) = \tilde{\mathbf{A}}\tilde{\mathbf{y}} + \mathbf{B}\mathbf{F}(\mathbf{V}_{\vec{\wp}}\tilde{\mathbf{y}})$ $\tilde{\mathbf{J}}(\mathbf{y}) = \tilde{\mathbf{A}} + \mathbf{B}\ \text{diag}\{\mathbf{F}'(\mathbf{V}_{\vec{\wp}}\tilde{\mathbf{y}})\}\mathbf{V}_{\vec{\wp}}$ $\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} - \tilde{\mathbf{J}}(\mathbf{y})^{-1}\tilde{\mathbf{G}}(\mathbf{y})$ where $\mathbf{B} = \mathbf{V}_k^T\mathbf{U}\mathbf{U}_{\vec{\wp}}^{-1}$, $\mathbf{U}_{\vec{\wp}} = \mathbf{P}^T\mathbf{U}, \mathbf{V}_{\vec{\wp}} = \mathbf{P}^T\mathbf{V}_k$	$2k^2 + \alpha(m) + k + 4mk$ $k^2 + \alpha^d(m) + 4mk + 2mk^2$ $\mathcal{O}(k^3)$	$\mathcal{O}(k^3 + mk^2)$

Table 3: Compare the computational complexity for each Newton iteration.

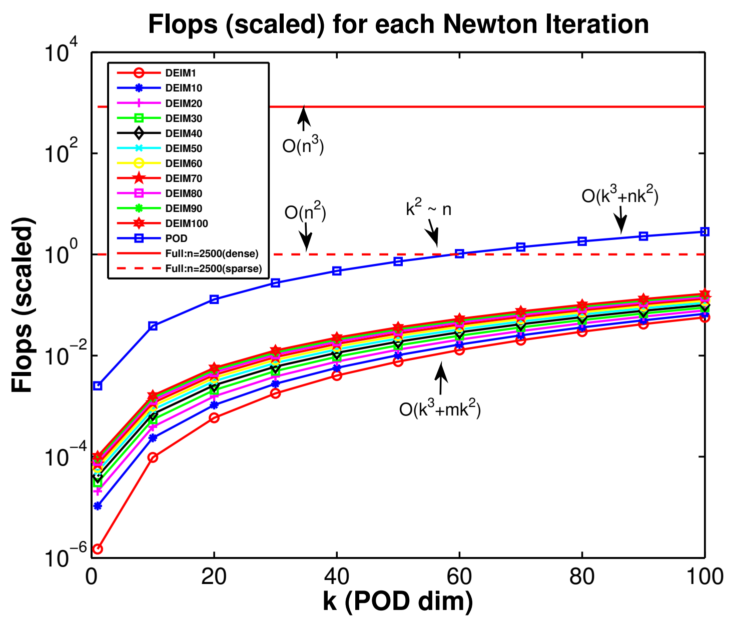


Figure 11: Approximate Flops (scaled with the Flops for full-sparse system) in each Newton iteration from Table 3.

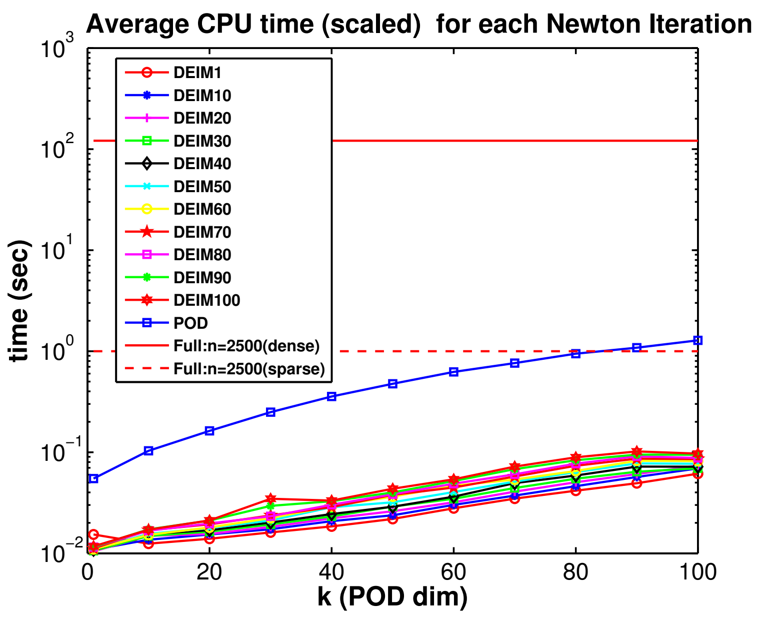


Figure 12: Average CPU time (scaled with the CPU time for full-sparse system) in each Newton iteration for solving the steady-state 2D problem.

system by using Newton's method. We shall refer to the method combining POD-Galerkin approach with the DEIM approximation as *POD-DEIM approach*.

4.1 The FitzHugh-Nagumo(FN) System

The FitzHugh-Nagumo system is used in neuron modeling. It is a simplified version of the Hodgkin-Huxley model, which models in a detailed manner activation and deactivation dynamics of a spiking neuron. This system is given by (64)-(67): Let $x \in [0, L], t \geq 0$,

$$\varepsilon v_t(x,t) = \varepsilon^2 v_{xx}(x,t) + f(v(x,t)) - w(x,t) + c \tag{64}$$

$$w_t(x,t) = bv(x,t) - \gamma w(x,t) + c, \tag{65}$$

with nonlinear function $f(v) = v(v - 0.1)(1 - v)$ and initial conditions and boundary conditions:

$$v(x,0) = \quad 0, \qquad w(x,0) = 0, \qquad x \in [0, L], \tag{66}$$

$$v_x(0,t) = \quad -i_0(t), \quad v_x(L,t) = 0, \qquad t \geq 0, \tag{67}$$

where the parameters $L = 1$, $\varepsilon = 0.015$, $b = 0.5$, $\gamma = 2$, $c = 0.05$. The stimulus $i_0(t) = 50000t^3 \exp(-15t)$. The variables v and w are voltage and recovery of voltage, respectively. The dimension of the full-order system (finite difference) is 1024. The POD basis vectors are constructed from 100 snapshot solutions obtained from the solutions of the full-order FD system at equally-spaced time steps in the interval $[0, 8]$. This is not a scalar equation and requires a slight generalization of the problem setting discussed earlier. However, the FD discretization does indeed yield a system of ODEs of the same form as (2).

Fig. 13 shows the fast decay around the first 40 singular values of the snapshot solutions for v, w, and the nonlinear snapshots $f(v)$. Note that the solution of this system has limit cycle for each spatial variable x. We therefore illustrate the solutions v and w through the plots of phase-space diagram as shown in Fig. 14 for the solutions of full-order system and the POD-DEIM reduced system using both POD and DEIM of dimension 5. We see that this reduced-order system captures the limit cycle of the original full-order system very well. The average relative errors of the solutions of the reduced systems and the average CPU time (scaled with the CPU time from sparse full-order system) for each time step from different dimensions of POD and DEIM are presented in Fig. 15.

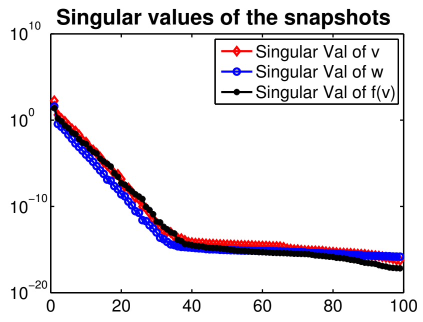


Figure 13: The singular values of the 100 snapshot solutions for v, w, and $f(v)$ from the full-order FD system of F-N system.

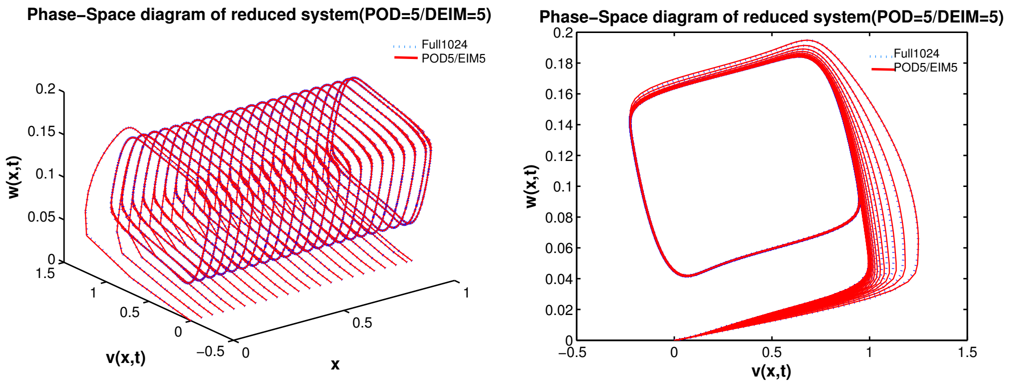


Figure 14: The phase-space diagram of v and w at different spatial points x (left) and its projection onto the v-w plane (right) from the original FD system (dim 1024) and from *POD-DEIM* reduced systems (POD dim 5,DEIM dim 5).

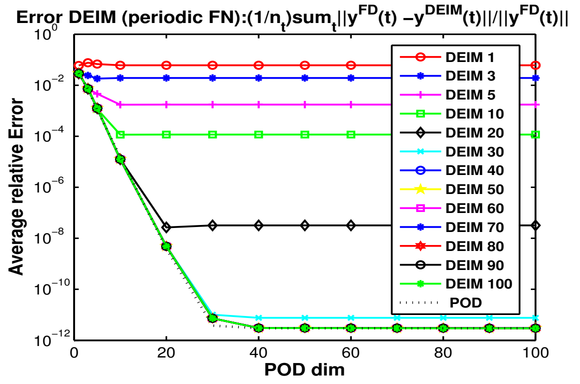
4.2 A Highly Nonlinear 2-D Steady State Problem

In this subsection, we apply the POD-DEIM method to nonlinear PDEs in a 2D spatial domain. The equations are

$$-\nabla^2 u(x,y) + s(u(x,y);\mu) = 100\sin(2\pi x)\sin(2\pi y), \tag{68}$$

$$s(u;\mu) = \frac{\mu_1}{\mu_2}(e^{\mu_2 u} - 1), \tag{69}$$

where spatial variable $(x,y) \in \Omega = (0,1)^2$, parameter $\mu = (\mu_1,\mu_2) \in \mathscr{D} = [0.01, 10]^2 \subset \mathbb{R}^2$, with a homogeneous Dirichlet boundary condition. We numerically solve this system by applying Newton's



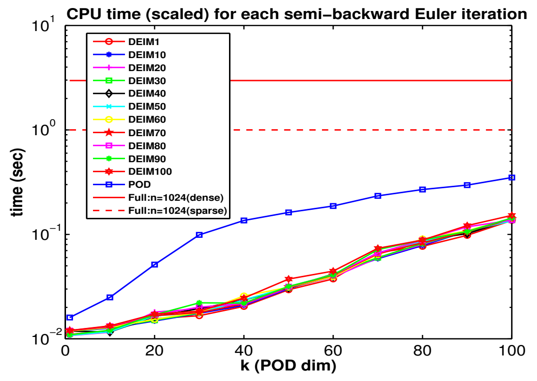


Figure 15: Left: Average relative errors from the POD-DEIM reduced system (solid lines) and from POD reduced systems (dashed line) for FN system. Right: Average CPU time (scaled with the CPU time for full-sparse system) in each time step of semi-implicit(backward) Euler method.

method to the nonlinear equations resulting from a FD discretization. The spatial grid points (x_i, y_j) are equally spaced in Ω for $i, j = 1, \ldots, 50$. The full dimension is then $n = 2500$. Figs. 16 and 17 show the singular values and the first 6 corresponding POD bases of the uniformly selected 144 sampled snapshot solutions for (68) and of the uniformly selected 144 nonlinear snapshots for (69). Fig. 18 shows the distribution of the first 30 points in Ω selected from the DEIM algorithm. Fig. 19 shows that the POD-DEIM reduced system (with POD and DEIM having dimension 6) can accurately reproduced the solution of the full-order system of dimension 2500 with error $O(10^{-3})$. The average errors and the average CPU time (scaled with the CPU time from sparse full-order system) for each Newton iteration of the reduced systems with different dimensions of POD and DEIM are presented in Fig. 20. The average CPU times for higher dimensions are shown earlier in Section 3.6. These errors are averaged over a set of 225 parameters μ that are not used in sampled snapshots. This suggests that the DEIM-POD reduced-order system can give a good approximation to the original system with any value of parameter $\mu \in \mathscr{D}$. This example is from [3].

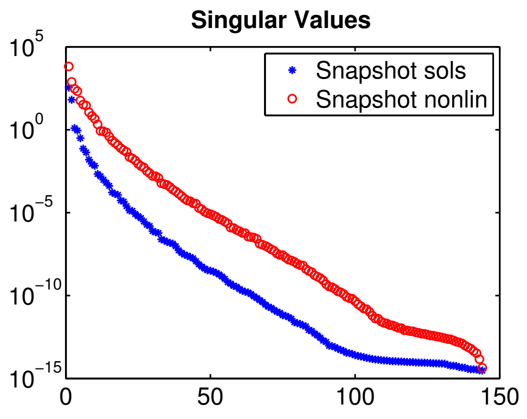


Figure 16: Singular values of the snapshot solutions u from (68) and the nonlinear snapshots $s(u;\mu)$ from (69).

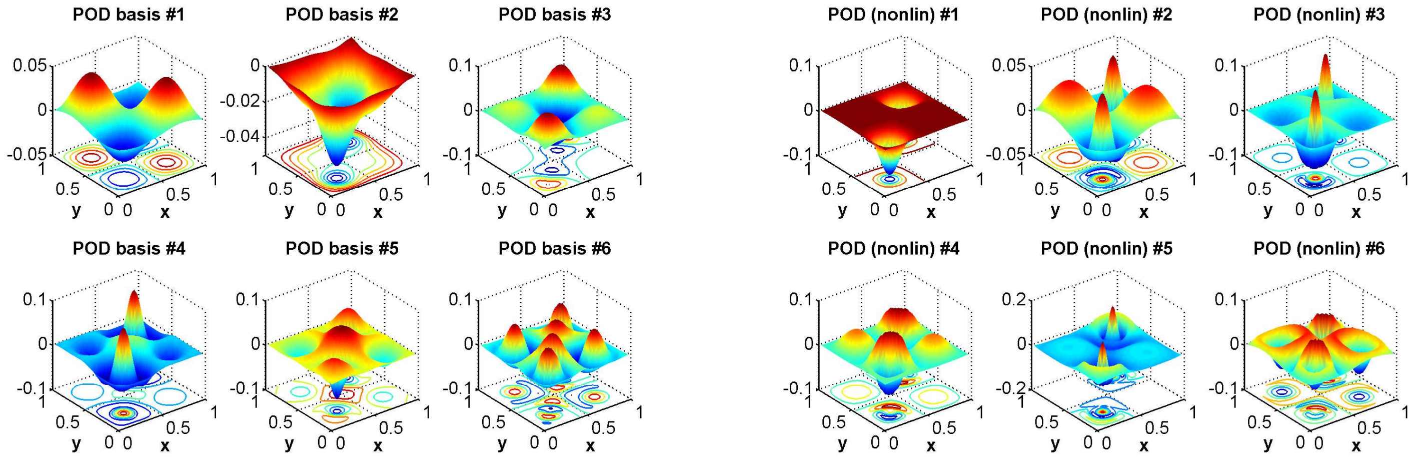


Figure 17: The first 6 dominant POD basis vectors of the snapshot solutions u from (68) and of the nonlinear snapshots $s(u;\mu)$ from (69).

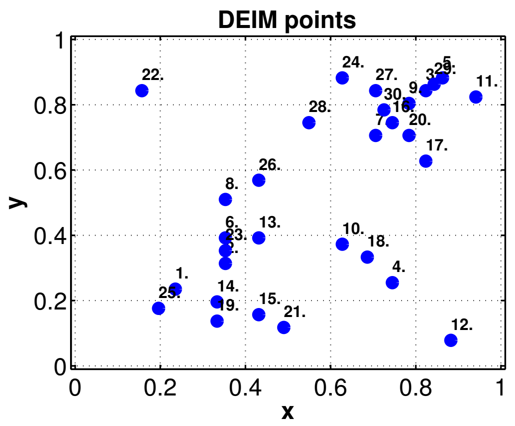


Figure 18: First 30 points selected by DEIM

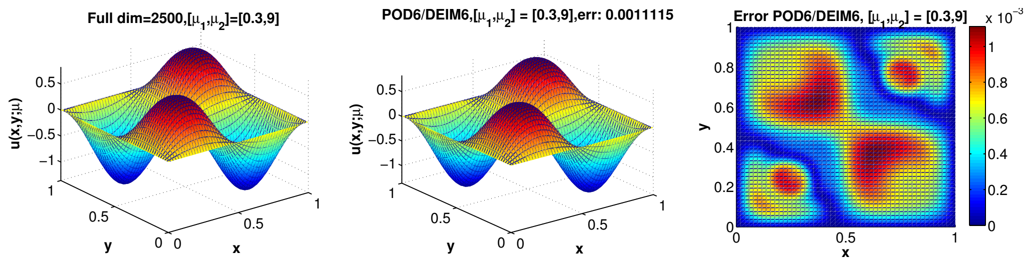


Figure 19: Numerical solution from the full-order system (dim= 2500) with the solution from POD-DEIM reduced system (POD dim = 6, DEIM dim = 6) for $\mu = (\mu_1, \mu_2) = (0.3, 9)$. The last plot shows the corresponding error at the grid points.

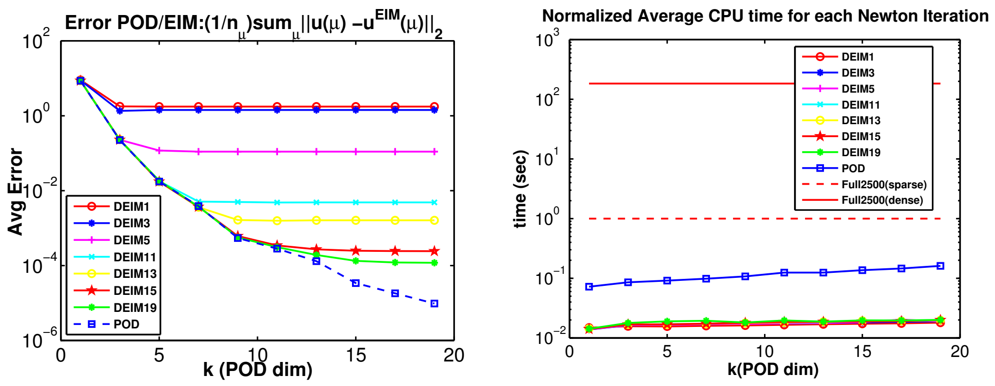


Figure 20: Average error from POD-DEIM reduced systems and average CPU time (scaled) in each Newton iteration for solving the steady-state 2D problem.

5 Conclusions

We have demonstrated that DEIM is very effective in overcoming the deficiencies of POD with respect to *general* nonlinearities. The method has an error bound showing the obtained approximation to be nearly optimal. The average errors for POD-DEIM approach in Fig. 15 and Fig. 20 show that the accuracy of the approximation depends on the dimensions of both POD and DEIM. The numerical results demonstrate that the POD-DEIM approach not only gives an accurate reduced system that is substantially smaller than the original system with general nonlinearity, but it also preserves the steady-state behavior (e.g. the limit cycle) of the original system. The POD-Galerkin approach combined with DEIM approximation is therefore a promising dimension reduction technique for FD discretized systems of unsteady or parametrized nonlinear PDEs.

6 Acknowledgments

The authors wish to thank Professor Mark Embree for several enlightening discussions and for pointing out reference [9]. DCS would also like to acknowledge Professor Jacob White for a lively discussion about DEIM for general nonlinearities at the SIAM CSE_09 meeting. This discussion led directly to the material in Section 3.5.

References

[1] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An 'Empirical Interpolation' Method: Application to Efficient Reduced-Basis Discretization Of Partial Differential Equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[2] T. Bui-Thanh, M. Damodaran, and K. Willcox. Aerodynamic Data Reconstruction and Inverse Design using Proper Orthogonal Decomposition. *AIAA Journal*, 42(8):1505–1516, August 2004.

[3] M. A. Grepl, Y. Maday, N. C. Nguyen, and A. T. Patera. Efficient Reduced-Basis Treatment of Nonaffine and Nonlinear Partial Differential Equations. *Mathematical Modelling and Numerical Analysis*, 41(3):575–605, 2007.

[4] K. Kunisch and S. Volkwein. Control of the Burgers Equation by a Reduced-Order Approach Using Proper Orthogonal Decomposition. *J. Optim. Theory Appl.*, 102(2):345–371, 1999.

[5] K. Kunisch and S. Volkwein. Galerkin Proper Orthogonal Decomposition Methods for a General Equation in Fluid Dynamics. *SIAM J. Numer. Anal.*, 40(2):492–515, 2002.

[6] Richard B. Lehoucq, Danny C. Sorensen, and Chao Yang. *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, 1998.

[7] N. C. Nguyen, A. T. Patera, and J. Peraire. A "Best Points" Interpolation Method for Efficient Approximation of Parametrized Functions. *Int. J. Numer. Meth. Engng*, 73:521–?543, 2007.

[8] C. W. Rowley, T. Colonius, and R. M. Murray. Model Reduction for Compressible Flows using POD and Galerkin Projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115– 129, 2004.

[9] Daniel B. Szyld. The Many Proofs of an Identity on the Norm of Oblique Projections. *Numerical Algorithms*, 42:309–323, 2006.