École polytechnique fédérale de Lausanne

Chair of Computational Mathematics and Simulation Science

Fall 2021

Semester project

Effect of random parameters on model order reduction of coupled oscillatory systems

Bruno Rodriguez Carrillo
sciper 326180

Professor:
Jan Hesthaven

Assistant:
Niccolò Discacciati

January 7th, 2022

1 Introduction

Mutual interactions in a network of simple, nearly identical, components may cause the system to exhibit synchronization. In the case of large networks, multi-query simulations of the system lead to a large computational cost. Aiming to both reduce it and retain the synchronization properties of the full systems when subject to different coupling strengths, we can make use of model order reduction techniques to construct surrogate models of the underlying systems [2].

Reduced basis (RB) methods represent a very efficient approach for the numerical approximation of problems involving the repeated solution of differential equations arising from engineering and applied sciences like partial differential equations (PDEs) depending on several parameters, optimal control and inverse problems. In all these cases, reducing the severe computational complexity is crucial [5].

For this reason, *model order reduction* (MOR) techniques have been developed aiming at replacing the original large-dimension numerical problem, usually called full-order approximation, by a reduced problem of substantially smaller dimension. The strategy adopted in MOR methods consists of projecting the full-order problem onto a subspace built upon specially selected basis functions. The importance of MOR methods lies in reducing computational complexity while guaranteeing sufficient accuracy [5].

We can state that MOR methods consist of two parts. In the first part, called *offline stage*, a set of vectors is extracted from simulations of the original, full-order model with specific parameter values and then the reduced model is obtained by projecting the full system onto the space spanned by such vectors, usually named the reduced space. During the second part, called *online stage*, only this computationally-efficient reduced model is solved. Such a solution, after a reconstruction procedure, constitutes an approximation of a particular trajectory of the full-order system for a given problem-dependent parameters [2].

In this project, we construct the reduced order models of two networks of globally-coupled oscillatory systems that show synchronization. We then propose a quasi-random sampling of the model parameters to improve the reduced model performance for a fixed distribution. We simulate and discuss the effect of such sampling on the *Kuramoto* and *Circadian* oscillators. Specifically, we focus on synchronization properties of both models. These possess strong reduction properties since their dynamics can be efficiently described by low-dimensional models [2].

2 Models

2.1 Kuramoto model

Let us start assuming we have a network of N limit-cycle oscillators with weak mutual interactions. Moreover, by using its phase variables, which we represent by $\{\phi_i(t)\}_{i=1}^{N}$, $i = 1 \dots N$, we can analyze its dynamics. Since we have N oscillators, such phase variables $\phi_i(t)$ define a system of phase dynamics equations.

Furthermore, if there is no coupling strength – we can think of this as a mutual interaction among all oscillators – denoted by K, each phase variable $\phi_i(t)$ is independent of the others and it grows linearly at a rate given by each oscillator's natural frequency ω_i. On the contrary, when there exist mutual interaction, the dynamics of the coupled N oscillators can be computed by the following system of N ordinary differential equations:

$$\frac{d\phi_i(t)}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N} H\left(\phi_j(t) - \phi_i(t)\right) \tag{1}$$

where we consider that the function H is a periodic function with period 2π that depends on the term $(\phi_j(t) - \phi_i(t))$ only. Choosing H as $\sin(\cdot)$, we obtain the so-called *Kuramoto model*:

$$\frac{d\phi_i(t)}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N} \sin\left(\phi_j(t) - \phi_i(t)\right) \tag{2}$$

We assume that the natural frequency of each oscillator i is an independent random variable distributed according to a unimodal and Ω-symmetric density distribution $g(\Omega)$, for a fixed $\Omega \in \mathbb{R}$.

Now, since we are interested in computing how the system of N oscillators is synchronized, we first define the *complex order parameter* $Z(t)$ as:

$$Z(t) = R(t)e^{i\psi(t)} = \frac{1}{N}\sum_{j=1}^{N} e^{i\phi_j(t)} \tag{3}$$

By computing the absolute value of (3), we can compute the level synchronization $R(t)$, also known as *order parameter* or *coherence parameter*, as follows:

$$R(t) = |Z(t)| = \left|\frac{1}{N}\sum_{j=1}^{N} e^{i\cdot\phi_j(t)}\right| \in [0, 1] \tag{4}$$

We are interested in computing $R(t)$ because such a value can be seen as a measure of how (in)coherent the system dynamics is. This can be interpreted in the following manner:

1. If $R(t) \Rightarrow 0$ in (4). This means that there exists incoherent dynamics and the phase variables are scattered in the interval $[0, 2\pi]$
2. If $R(t) \Rightarrow 1$ in (4). This means that there exists coherent dynamics and the phase variables have a limited variability

It is important to mention that there exists a transition phase where the phase variables $\phi_i(t)$ reach, if that is the case, a common stable value for each oscillator. Consequently, we are usually interested in the asymptotic synchronization properties of the system after such a phase. This is the reason for which we compute (4) for times t larger than a sufficiently large time T_0, which corresponds to the time interval where the transition phase happens. Then we compute (4) at a final simulation time $T > T_0$, i.e., $R = R(t = T)$.

In the limit of an infinite number of oscillators, there exists a critical coupling strength K_c, which is computed as:

$$K_c = \frac{2}{\pi \cdot g(\Omega)} \tag{5}$$

and K_c has the following properties (with K as given in (2):

1. If $K < K_c \Rightarrow R = O(1/\sqrt{N}) \simeq 0$. This means incoherent dynamics
2. $K \gtrsim$ and $g(\omega)$ sufficiently smooth:

$$R \simeq \sqrt{\frac{16}{\pi K_c^3}} \cdot \sqrt{\frac{K - K_c}{-K_c \cdot g''(\Omega)}} \sim \sqrt{K - K_c}$$

   We can view this case as a state between the incoherent and coherent dynamics

3. $k \gg k_c \Rightarrow R \simeq l$. This means coherent dynamics

On the other hand, if we replace $x_i(t)$ and $y_i(t)$ in (2) by $x_i(t) = \cos\phi_i(t)$ and $y_i(t) = \sin\phi_i(t)$, we obtain the the following set of equations for the Kuramoto model:

$$\begin{aligned}
\frac{dx_i(t)}{dt} &= -y_i(t)\left(\omega_i + \tfrac{K}{N}\textstyle\sum_{j=1}^{N}\left(y_j(t)x_i(t) - x_i(t)y_i(t)\right)\right), \\
\frac{dy_i(t)}{dt} &= x_i(t)\left(\omega_i + \frac{K}{N}\textstyle\sum_{j=1}^{N}\left(y_j(t)x_i(t) - x_j(t)y_i(t)\right)\right),
\end{aligned}$$

And to compute (4) using such a change of variables, we compute:

$$R(t) = \left| \frac{1}{N}\sum_{j=1}^{N} e^{i\cdot\tan^{-1}\left(\frac{y_j(t)}{x_j(t)}\right)(t)} \right|$$

2.2 Circadian oscillators

We study a specific model for globally coupled circadian oscillators, which form a network of N neuronal oscillators. In each oscillator i for $i = 1, \ldots N$, the clock gene mRNA $X_i(t)$ produces a clock protein $Y_i(t)$, which activates a transcription inhibitor $Z_i(t)$, and this, in turn, inhibits the transcription of the clock gene. Additionally, the mRNA excites the production of a neurotransmitter $V_i(t)$. The network dynamics is influenced by two additional factors: the external light, modeled as a sinusoidal signal $L(t) = (L_0/2)(1 + \sin(\omega t))$ and the inter-cellular coupling $F(t)$ defined as

$$F(t) = \frac{1}{N}\sum_{i=1}^{N} V_i(t) \tag{6}$$

And from (XX), we have the following set of equations for oscillator i:

$$\begin{aligned}
\tau_i \frac{dX_i(t)}{dt} &= V_1 \frac{K_1^4}{K_1^4 + Z_i(t)^4} - V_2 \frac{X_i(t)}{K_2 + X_i(t)} + V_c \frac{KF(t)}{K_c + KF(t)} + \frac{L_0}{2}(1 + \sin(\omega t)), \\
\tau_i \frac{dY_i(t)}{dt} &= K_3 X_i(t) - V_4 \frac{Y_i(t)}{K_4 + Y_i(t)}, \\
\tau_i \frac{dZ_i(t)}{dt} &= K_5 Y_i(t) - V_6 \frac{Z_i(t)}{K_6 + Z_i(t)}, \\
\tau_i \frac{dV_i(t)}{dt} &= K_7 X_i(t) - V_6 \frac{V_i(t)}{K_8 + V_i(t)}
\end{aligned} \tag{7}$$

As we did with the Kuramoto model, we can measure the synchronization properties of model described in (7) by using four quantities which we define as follows. We define the synchronization variable as

$$Q(t) = \frac{F(t)^2}{\frac{1}{N}\sum_{i=0}^{N} V_i(t)^2} \tag{8}$$

and the parameter synchrony as

$$\rho = \sqrt{\langle Q(t) \rangle} \in [0, 1] \tag{9}$$

where $\langle \cdot \rangle$ denotes the time average once the asymptotic state has been reached. The value of ρ determines whether or not there is a (in)coherent dynamics. When $\rho \Rightarrow 0$ this corresponds to an incoherent dynamics and $\rho \Rightarrow 1$ indicates coherent dynamics.

By defining the *average gene concentration* $X(t)$ as

$$X(t) = \frac{1}{N} \sum_{i=1}^{N} X_i(t) \tag{10}$$

allows us to introduce the *spectral amplification factor* S

$$S = \frac{4}{L_0^2} \left| \left\langle e^{-i\omega t} X(t) \right\rangle \right|^2 \tag{11}$$

We furthermore define the *average period* $\tilde{T}$ as the average of $X(t)$, (10), that is:

$$\tilde{T} = \langle X(t) \rangle \tag{12}$$

Finally, we define the *order parameter*:

$$R(t) = \left| \frac{1}{N} \sum_{j=1}^{N} e^{i \cdot \phi_j(t)} \right| \in [0, 1] \tag{13}$$

The values of $Q(t)$, ρ, $X(t)$ and S depend substantially on the value of $R(t)$. Specifically,

1. If K in (7) increases, ρ in (9) and $\tilde{T}$ in (12) increase
2. If L_0 is sufficiently large, the whole system is forced to oscillate with a period $2\pi/\omega$, together with a full synchronization, measured by (13)

3 Model reduction

Based on [2], we start by considering a parameterized dynamical system described by

$$\begin{cases} \dfrac{d}{dt}\mathbf{x}(t,\boldsymbol{\mu}) = \mathbf{f}(t,\mathbf{x},\boldsymbol{\mu}), t \in [0,T], \\ \mathbf{x}(0,\boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}) \end{cases} \tag{14}$$

where $T > 0$ is a fixed time value, $\mathbf{x} \in \mathbb{R}^n$ represents the state vector, $\boldsymbol{\mu} \in \Gamma \subset \mathbb{R}^d$ is a vector containing all parameters, and $\mathbf{f} \in \mathbb{R}^n$ is a possibly nonlinear function that characterizes the system dynamics. In addition, we define the solution manifold as the set of solutions to (14):

$$\mathcal{M} := \{\mathbf{x}(t,\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \Gamma, t \in [0,T]\} \subset \mathbb{R}^n \tag{15}$$

Model reduction techniques (MOR) rely on the assumption that the solution manifold $\mathcal{M}$ is low dimensional; that is, it can be approximated by the span of a chosen basis $\{\mathbf{u}_i\}_{i=1}^k$ where k indicates the dimension of the reduced system and $k \leq n$. Refer to [5] for further information on model reduction techniques. With such a set of vectors $\mathbf{u}_i$, we can approximate the solution to (14) as

$$\mathbf{x}(t,\boldsymbol{\mu}) \approx \tilde{\mathbf{x}}(t,\boldsymbol{\mu}) = U\boldsymbol{\alpha}(t,\boldsymbol{\mu}) = \sum_{i=1}^{k} \alpha_i(t,\boldsymbol{\mu})\mathbf{u}_i, \tag{16}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^k$ denotes the vector of latent coordinates $\{\alpha_i\}_{i=1}^k$ – these numbers represent the coefficients of the linear combination of $\mathbf{u}_i$ –, and $U \in \mathbb{R}^{n\times k}$ is an orthogonal matrix whose columns, $\mathbf{u}_i$, form a basis of $\mathbb{R}^n$. Replacing $\mathbf{x}$ by $\tilde{\mathbf{x}}$ from (16) into the system (14), we obtain

$$U\frac{d}{dt}\boldsymbol{\alpha}(t,\boldsymbol{\mu}) = \mathbf{f}\left(t, U\boldsymbol{\alpha}(t,\boldsymbol{\mu}), \boldsymbol{\mu}\right) + \mathbf{r}(t,\boldsymbol{\mu}), \tag{17}$$

where $\mathbf{r}(t,\boldsymbol{\mu})$ represents the residual due to approximation. A system like (17) is over-determined and thus its dynamics can be closed by a Petrov-Galerkin projection. For this reason, we construct a basis $W \in \mathbb{R}^{n\times k}$ that we assume to be orthogonal to the residual vector $\mathbf{r}(t,\boldsymbol{\mu})$, such that the matrix $\left(W^\top U\right)$ is invertible, which yields:

$$\frac{d}{dt}\boldsymbol{\alpha}(t,\boldsymbol{\mu}) = \left(W^\top U\right)^{-1} W^\top \mathbf{f}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{18}$$

However, we restrict ourselves to the Galerkin framework, where the basis U is orthogonal, and $W = U$. See [5]. Then we obtain

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{\alpha}(t,\boldsymbol{\mu}) = U^\top \boldsymbol{f}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}), t \in [0,T], \\ \boldsymbol{\alpha}(0,\boldsymbol{\mu}) = \boldsymbol{\alpha}_0(\boldsymbol{\mu}) = U^\top \boldsymbol{x}_0 \end{cases} \tag{19}$$

Model (19) corresponds to the model we will be working with and now we describe how the basis U is in fact constructed. To that end, we assume that the full solution $\mathbf{x}$ to (14) is available at appropriately chosen times $\{t_i\}_{i=1}^p$ and parameter instances $\{\boldsymbol{\mu}_j\}_{j=1}^q$. We call *snapshot matrix* the matrix containing such vectors as its columns; that is:

$$S = \left[\mathbf{x}\left(t_1,\boldsymbol{\mu}_1\right), \ldots, \mathbf{x}\left(t_p,\boldsymbol{\mu}_1\right), \ldots, \mathbf{x}\left(t_1,\boldsymbol{\mu}_q\right), \ldots, \mathbf{x}\left(t_p,\boldsymbol{\mu}_q\right)\right] \in \mathbb{R}^{n\times pq} \tag{20}$$

To compute the basis $\{\mathbf{u}_i\}_{i=1}^{k}$, which form the columns of U in (19), we need to minimize the projection error of the snapshots onto the reduced space, which leads us to the method called *Proper Orthogonal Decomposition (POD)*. That is, we have to solve the following minimization problem

$$\begin{aligned} &\min_{V\in\mathbb{R}^{n\times k}} \left\|S - VV^\top S\right\|_F, \\ &\text{subject to } V^\top V = \mathbb{I}_k \end{aligned} \tag{21}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbb{I}_k$ is the $k \times k$ identity matrix.

The Eckart-Young theorem states that the best approximation of any matrix by matrices of lower rank can be computed as a sum of 1-rank matrices. In other words, the solution to problem (21) is given by the first k left singular vectors of S. That is, let us consider the SVD of S as

$$S = U_S \Sigma_S V_S^\top, \Sigma_S = \operatorname{diag}(\sigma_i) \in \mathbb{R}^{n\times pq}, \tag{22}$$

Then we can construct our basis U as

$$\begin{aligned} U = U_S(:, 1:k) &= \underset{V\in\mathbb{R}^{n\times k}}{\arg\min} \left\|S - VV^\top S\right\|_F, \\ \left\|S - UU^\top S\right\|_F^2 &= \textstyle\sum_{i=k+1}^{\min(n,pq)} \sigma_i^2 \end{aligned} \tag{23}$$

At each time step, the solution $\hat{\mathbf{x}} = U\boldsymbol{\alpha}$ has to be computed, the function $\mathbf{f}$ as to be evaluated and projected onto the lower dimensional space, which is in fact more computationally expensive than directly solving (14) since $\mathbf{f}$ is in general nonlinear in its arguments. To overcome this bottleneck, we make use of the *Discrete Empirical Interpolation Method* (DEIM) [1], which helps approximate $\mathbf{f}$ as a linear combination of suitable basis vectors $\{\mathbf{v}_i\}_{i=1}^{\tilde{k}}$ that form the columns of V such that

$$\mathbf{f}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}) \approx \hat{\mathbf{f}}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}) = V\mathbf{c}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{24}$$

where the vector $\mathbf{c} \in \mathbb{R}^{\widetilde{k}}$ corresponds to the approximation in the basis V. To compute $\mathbf{c}$, we select $\tilde{k}$ special rows from the over determined (24). Then we consider the matrix:

$$P = \left[e_{p_1}, \ldots, e_{p_{\tilde{k}}}\right], \tag{25}$$

where $\{p_1, \ldots, p_{\tilde{k}}\}$ are $\tilde{k}$ indices selected from $\{1, \ldots, n\}$ and e_{p_i} is the i-th column of the $n \times n$ identity matrix $\mathbb{I}_n$. Assuming the matrix $P^\top V$ is invertible, the vector $\mathbf{c}$ is found by solving the linear system, $P^\top \mathbf{f}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}) = \left(P^\top V\right)\mathbf{c}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu})$, replacing $\mathbf{f}$ with $\hat{\mathbf{f}}$ in (19) leads to:

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{\alpha}(t, \boldsymbol{\mu}) = U^\top V \left(P^\top V\right)^{-1} P^\top f(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}), t \in [0, T], \\ \boldsymbol{\alpha}(0, \boldsymbol{\mu}) = \boldsymbol{\alpha}_0(\boldsymbol{\mu}) = U^\top \mathbf{x}_0 \end{cases} \tag{26}$$

The indices $\{p_i\}_{i=1}^{\tilde{k}}$ corresponds to the components of the largest magnitude of an appropriately defined residual vector. Computing (26) requires evaluating $\tilde{k}$ of the $\mathbf{f}$ with $\tilde{k} \leq n$. Likewise, $U^\top V \left(P^\top V\right)^{-1}$ is constant in time and computed once during the offline stage. To construct the matrix V, we could follow the snapshot matrix approach as before, but now replacing the vector solution $\mathbf{x}$ by evaluations of $\mathbf{f}$ for each sample included in the matrix S. It has been observed in [2] that it is sufficient to choose $\tilde{k} = k$ and $V = U$ to obtain good reduction properties. Thus, unless stated otherwise, we stick to the same approach. Consequently, (26) reduces to

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{\alpha}(t,\boldsymbol{\mu}) = \left(P^\top U\right)^{-1} P^\top \mathbf{f}(t, U\boldsymbol{\alpha}, \boldsymbol{\mu}), \quad t \in [0,T], \\ \boldsymbol{\alpha}(0,\boldsymbol{\mu}) = \boldsymbol{\alpha}_0(\boldsymbol{\mu}) = U^\top x_0 \end{cases} \tag{27}$$

(14) is called the *Full-Order Model*(FOM), and (19) and (27) are the *Reduced-Order Model*(ROM), via POD and DEIM, respectively.

We discuss in the following lines how to apply such model reduction techniques to globally coupled oscillators.

4 Application of model reduction techniques to globally coupled oscillators

Let us consider a network of N oscillators, each represented by $i \in \{1, \dots, N\}$, whose evolution equation is an m-dimensional system of first-order equations with state vector $\mathbf{x}_i = \left[x_i^1, \dots, x_i^m\right]^\top \in \mathbb{R}^m$.

The oscillators' uncoupled dynamics is governed by a function $\mathbf{h}_i(\mathbf{x}_i) \in \mathbb{R}^m$ and the global system formed by N oscillators is constructed by stacking each local state vector as follows:

$$\mathbf{x} = \left[\mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top\right]^\top = \left[x_1^1, \dots, x_1^m, \dots, x_N^1, \dots, x_N^m\right]^\top \in \mathbb{R}^{Nm} \tag{28}$$

We assume that the coupling terms contribute to the dynamics in an additive way, that is, they do not directly interact $\mathbf{h}_i(\mathbf{x}_i)$. We also assume that their effect is identical for each oscillator. As a result, interactions can be modeled by a function $\mathcal{H} = \mathcal{H}(\mathbf{x}) \in \mathbb{R}^m$, which results in a global system described by

$$\mathbf{f} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} \mathbf{h}_1(\mathbf{x}_1) + \mathcal{H}(\mathbf{x}) \\ \vdots \\ \mathbf{h}_N(\mathbf{x}_N) + \mathcal{H}(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^{Nm} \tag{29}$$

Afterwards, we collect the components of the state vector associated to the j-th, $j = 1, \dots m$ variable in a vector

$$\mathbf{x}^j = \left[x_1^j, \dots, x_N^j\right] \in \mathbb{R}^N.$$

By doing this, we construct m separate snapshot matrix for each coordinate j as follows. This is done to preserve the block structure of the problem.

$$S_j = \left[\mathbf{x}^j(t_1, \boldsymbol{\mu}_1), \dots \mathbf{x}^j(t_p, \boldsymbol{\mu}_1), \dots, \mathbf{x}^j(t_1, \boldsymbol{\mu}_q), \dots, \mathbf{x}^j(t_p, \boldsymbol{\mu}_q)\right] \tag{30}$$

where $S_j \in \mathbb{R}^{N \times pq}$.

We can construct a reduced model for each S_j by computing its SVD and choosing the first r singular vectors; then we have m different approximations

$$\mathbf{x}^j \approx \hat{\mathbf{x}}^j = U_j \boldsymbol{\alpha}_j \tag{31}$$

where $U_j \in \mathbb{R}^{N \times r}$ is an orthogonal matrix and $\boldsymbol{\alpha}_j \in \mathbb{R}^r$ is the latent coordinates. Globally we have:

$$\mathbf{x} \approx \hat{\mathbf{x}} = U\boldsymbol{\alpha} = \mathcal{P}\text{blkdiag}\left(\{U_j\}_{j=1}^m\right)\left[\alpha_1^\top, \dots, \alpha_m^\top\right]^\top \tag{32}$$

where $U \in \mathbb{R}^{Nm \times r \cdot m} = \mathbb{R}^{n \times k}$ and $\boldsymbol{\alpha} \in \mathbb{R}^{rm} = \mathbb{R}^k$.

The matrix $\mathcal{P}$ is defined as the permutation matrix such that

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} = \mathcal{P} \begin{pmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^m \end{pmatrix},$$

We do not apply directly the DEIM method to U in (32) since this will lead to a loose of the block structure. Instead, we apply DEIM to the U_i matrix associated to the j coordinate.

On the other hand, computing the function $\mathcal{H}(\mathbf{x})$ requires evaluating each state $\mathbf{x}$ of the N oscillators, which represents a total cost proportional to $n = Nm$ at each time step. To overcome this, we rely on a *mean-field coupling*, which assumes that $\mathcal{H}(\mathbf{x})$ depends on an average over N oscillators for each coordinate $j = 1, \ldots, m$. This gives

$$\begin{aligned}
&\mathcal{H}(\mathbf{x}) = \mathcal{H}\left(\frac{1}{N}\sum_{i=1}^{N} x_i^1, \ldots, \frac{1}{N}\sum_{i=1}^{N} x_i^m\right) \\
&\approx \mathcal{H}\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{r} (U_1)_{ij}\,(\alpha_1)_j \ldots, \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{r} (U_m)_{ij}\,(\alpha_m)_j\right) \\
&= \mathcal{H}\left(M_1 \cdot \alpha_1, \ldots, M_m \cdot \alpha_m\right)
\end{aligned}$$

where: $(M_j)_\ell = \dfrac{1}{N}\sum_{i=1}^{N} (U_j)_{i\ell}\,, j = 1, \ldots, m, \ell = 1, \ldots, r$ is computed once during the offline phase.

For explicit expression for both models described before, refer to [2].

5 Random number generators

Let us consider a random variable Z that is the output quantity of a stochastic model. We set the goal of computing its expectation or mean value $\mu = \mathbb{E}[Z]$. We assume that the probability distribution of Z is not known analytically, but Z can be simulated. The Monte Carlo method consists simply of generating N *i.i.d* replicas $Z^{(1)}, \ldots, Z^{(N)}$ of Z and estimating $\mu = \mathbb{E}[Z]$ by a *sample mean estimator* $\hat{\mu}$

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} Z^{(i)}, \tag{33}$$

and we assume that $\mathbb{V}\text{ar}[Z] = \sigma^2 < +\infty$. This method is also called standard Monte Carlo method.

Furthermore, based on [3], the purpose of a uniform random number generator is to produce an unlimited stream of numbers U_1, U_2, ... that behave statistically as independent and uniformly distributed random variables on the interval $(0, 1)$. From such a stream, it is easy to construct an infinite sequence of independent and uniformly distributed random vectors (points) in $(0, 1)^d$, by defining $\mathbf{U}_1 = (U_1, \ldots, U_d)$, $\mathbf{U}_2 = (U_{d+1}, \ldots, U_{2d})$, ... For any real-valued function h on $(0, 1)^d$ these random vectors can then be used for numerical evaluation of multidimensional integrals that are complicated or impossible to solve analytically –we also denote the exact value of such an integral by μ–

$$\mu = \int h(\mathbf{u})\, d\mathbf{u} \tag{34}$$

via the sample average (33). In particular, it can be shown [3] that the standard error of Monte Carlo sampling, denoted by $\mathrm{E}\left[(\hat{\mu} - \mu)^2\right]$, decreases at a rate $\mathcal{O}(N^{-1/2})$. Hence, asymptotically, to decrease the error by a factor 2, one needs 4 times as many samples. This convergence rate can often be improved by constructing quasirandom points $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ that fill the unit cube in a much more regular way than is achieved via *iid* random points.

Quasirandom points are no longer independent, but do have a high degree of uniformity, which is often expressed in terms of their discrepancy. See [3].

According to [3], Quasi Monte Carlo techniques provide a powerful way to estimate d-dimensional integrals of the form (34) by means of sample average

$$\frac{1}{N} \sum_{\mathbf{u} \in \mathcal{P}_N} h(\mathbf{u}),$$

but now we compute the average over a set of N quasi-random points $\mathcal{P}_N$. To use such a points and compute error estimates, we can shift these points randomly and then use them to compute the sample average (33). There exist several manner of creating such points; herein we use *Latin-Hypercube* and *Sobol's* sequences although there are others such as Halton or Faure sequences. Quasi Monte Carlo methods can lead to significant variance reduction, which is the their main application. For further details on such sequences, refer to [3].

As a manner of illustration, we include the following picture:

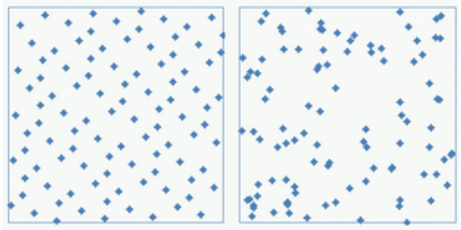


Figure 1: Quasi-random and uniform random numbers, from Wikipedia [7].

On the right hand side, we observe a uniform distribution in $(0,1)^2$ without using quasi-random number and on the left hand side, we observe its counterpart for quasi-random numbers. In general, when we use such quasi random numbers, we sample the hyper-cube $(0,1)^d$ in such a manner that we have points covering more uniformly such a hyper-cube.

As observed in Fig.1, since quasi-random numbers sample uniformly the square $(0,1)^2$, we would need fewer points and then fewer evaluations to compute the sample average. This means that we could have savings in terms of computational resources.

In our numerical experiments, we use the Sobol and Latin-hypercube sequences that are implemented in Matlab and compared their performance against the standard random generator in Matlab. More details on how to construct these quasi-random numbers, review the Matlab website [4].

On the other hand, after having the sequences generated, we shift them randomly. Then to use these sequences, we compute the inverse of the cumulative distribution function according to what distribution we want to generate; that is, we follow the steps below, which is called the *inverse transform sampling method*:

1. Generate a random number u from the standard uniform distribution (in our case these are the quasi-random numbers) in the interval $[0,1]$ e.g. from $U \sim \text{Unif}\,[0,1]$
2. Find the inverse of the desired CDF, e.g.$F_X^{-1}(x)$
3. Compute $X = F_X^{-1}(u)$. The computed random variable X has distribution $F_X(x)$

One of the main drawbacks of this sampling method is that we need the inverse $F_X^{-1}(x)$ of the distribution we want to sample from and unfortunately such inverse is available for a few distributions only. One alternative to this is use importance sampling, which we do not discuss further.

Regarding the distributions of the natural frequencies ω_i, we use the normal, the beta and the gamma distributions to sample the natural frequencies ω_i for our experiments. The inverses of each function are the ones already implemented in Matlab:*erfinv*, *betaincinv* and *gammaincinv*.

6 Results and simulations

We report a number of numerical results to show the strengths and weaknesses of applications of MOR techniques to our models of interest.

To this end, we run several numerical experiments to test how the reduced-dimension model generalizes when we sample the natural frequencies ω_i according to a given distribution. We test for various values of ω's because we are interested in testing the performance of the MOR approach for general values of the model's parameters to make sure that the model is not overfitting over the training set. Moreover, our focus is on the random parameters and its effect over the performance of the reduced model. Generalization for different values of the strength parameter K has already been investigated in [2].

First, we fix the simulations parameters as follows. We use a explicit Runge-Kutta method of third order, a final time $T = 100$, $N = 100$ oscillators, time step $\Delta t = 0.20$, total number of steps per simulation $N_{steps} = 500$, number of Monte Carlo samples $N_{MC} = 50$. We sample the strength coupling by dividing the interval $[0, K_{max}]$ into 15 equally-spaced sub-intervals with $K_{max} = 1.5$ and we assume that the initial phases ϕ_0 are uniformly spaced in the interval $[0, \pi]$ at initial time. For all experiments, we work with zero-mean natural frequencies.

During the offline(training) phase, we construct the reduced basis by using a different set of $i.d.d$ natural frequencies ω_i for each Monte Carlo sample. Once in the online(testing) phase, we simulate the reduced model (27) using an independent set of ω_i for each N_{MC}. This means that we train our model with a different set of ω_i than the one we use to test it. Furthermore, because of comparison reasons, we fix K_{max} and the same number of sub-intervals of the interval $[0, K_{max}]$ during offline and online phases.

We clarify that we are interested in computing the asymptotic order parameter R as a function of the coupling strength K and time for the given simulation parameters. Let us call these quantities $R(K)$ and $R(t)$. This is important to be clear about since we essentially solve the dynamics of the system, which means we are computing the phase dynamics ϕ_i for each oscillator. However, we are not explicitly interested in knowing the final phases but how the transition from uncoupled to coupled dynamics looks like and how the synchronization properties of the oscillators are at final time.

We will be showing how the decay of the singular values behaves as we vary the simulation parameters. We plot $\sigma_j/\sigma_{\max}$, the first k singular values σ_j, $j = 2, \ldots, k$ and $\sigma_{\max} = \sigma_1$ as the largest singular value. Intuitively, the magnitude of the singular values σ_i of the SVD of the snapshot matrix indicates how reducible the dynamics of the Kuramoto model can be. In other words, if the quantity $\sigma_j/\sigma_{\max}$ decays rapidly, this is a good indicator of the capacity of the model to be described by a lower dimensional model. This decay, together with $R(K)$ and $K(t)$, are our quantities of interest. As a result, we could say the reduced model generalizes more efficiently, in terms of computational consumption, when the whole dynamics can be explained with a few left singular vectors $\mathbf{u}_i$, which form the matrix U in (27). This could be observed as a fast decay of the singular values in the plots.

As we will observe, the variance of $\omega_i$and the coupling strength K play a crucial role for our model to generalized properly. Consequently, we may say that our goal with this project is to improve the reduced model's capacity to capture the dynamics of the full system using a few left singular vectors of the SVD of the snapshot matrix S.

Regarding the Kuramoto and Circadian models, we use three different distributions with their corresponding parameters, each is given below. It is worth noticing that the value of the variance

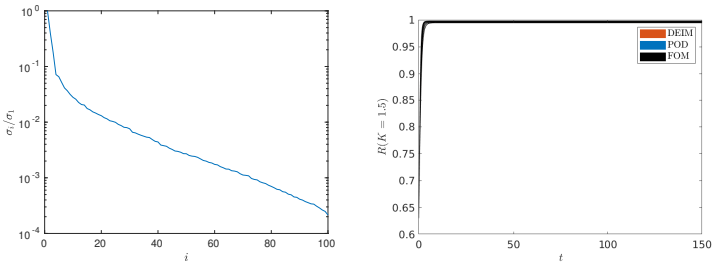
plays an crucial role in terms of how the reduced model captures the dynamics of the full model.

Kuramoto model

We first simulate the Kuramoto model for the natural frequencies ω_i distributed as $\mathcal{N}(0, \sigma^2)$. We then order ω_i in descending order and compute the quantities of interests as previously discussed. We sort the natural frequencies because there is a global coupling among oscillators, and there are not additional oscillator-dependent parameters; thus, sorting the natural frequencies reduces the variability of the $i.i.d$ sampling, which might be critical. In [6], it has been shown that the Kuramoto model is invariant under shifts of the natural frequencies, thus we assume that the mean of the ω's is zero; this is also typically the case when the Kuramoto model is analyzed.

We now show the plots for the quantities of interests for the normal distribution. The dimension of the reduced model is indicated in each plot as k.

For $\sigma = 10^{-2}$, $\mu = 0$, sorting ω_i in descending order, using the same ω_i during offline and online phases, we obtain Fig.2.

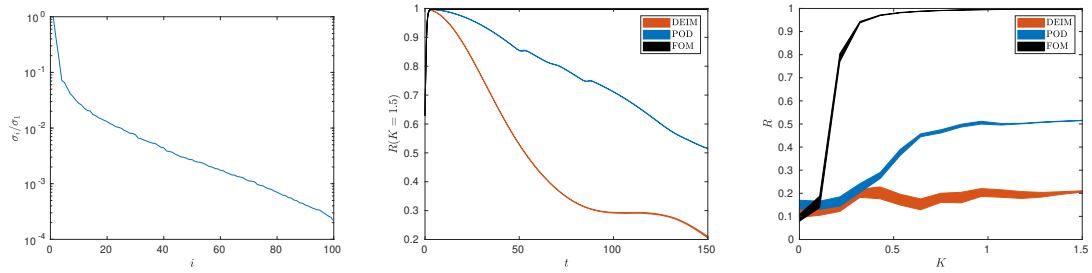


(a) Singular value decay. (b) R as a function of t.

Figure 2: Quantities of interest, $k = 20$

In Fig.2.a, the decay of the singular values is reported. Despite the high degree of heterogeneity, the singular values decay at a sufficiently fast rate to be able to capture the synchronization pattern. Moreover, the confidence interval for the order parameter R is shown as a function of time for $K_{\max}$ in Fig.2.b and Fig.2.c shows the confidence interval for R as a function of K at final time, which demonstrates that the synchronization properties are captured by the reduced model.

Afterwards, we construct our reduced basis using two different sets of ω_i for training and testing, and keeping exactly the same parameters as Fig.2; we notice the results in Fig.3.

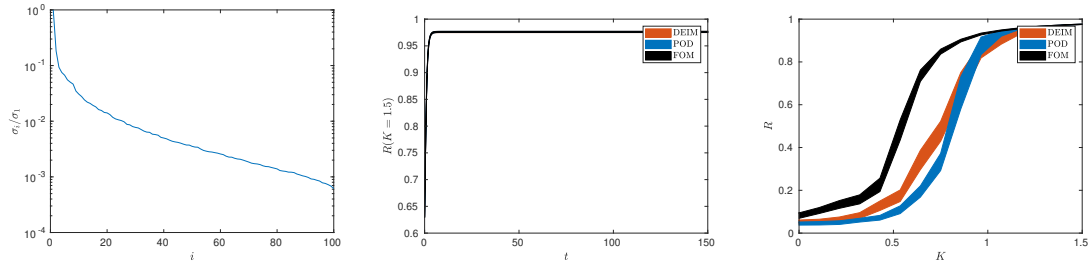


(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

Figure 3: Quantities of interest when changing testing ω, $k = 20$

From Fig.3, we clearly observe that the model cannot capture the synchronization properties of the model when using different natural frequencies for training and testing although the decay of the singular values seems to indicate a high potential of the model to be described by a lower-dimensional model. We may say in that the reduced model under-fits in this case.

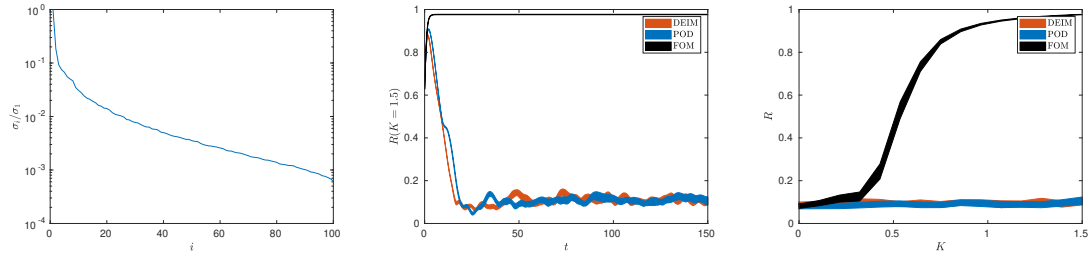
On the other hand, if we increase the value of the variance from the one used in Fig.2 to $\sigma^2 = 10^{-1}$ and use the same ω_i for training and testing –these are sorted too–, we obtain Fig.4



(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

Figure 4: Quantities of interest when changing σ^2, $k = 20$

If we set $\sigma^2 = 10^{-1}$ and we generate a new set of ω_i for testing, we have Fig.5



(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

Figure 5: Quantities of interest when changing σ^2 and test set, $k = 20$

In Fig.4, we can see that increasing the variance by an factor of 10 leads to a model that is still able to describe the synchronization characteristics of the model but this happens for values of K close to $K_{\max}$, Fig.4.c. Furthermore, the synchronization behavior is visually approaching the one from the full model.

However by looking at Fig.5, both the POD and DEIM solutions fail to capture the dynamics of the system and describe the coupled properties of the system of oscillators. This behavior is observed even when we sort the values of ω_i. This case supports our claim that even thought the model can be described by a reduced model – seen as a relatively fast decay of the singular values– the same model is incapable of capturing the synchronization dynamics of the system. This may happen since the variability of the natural frequencies make it impossible for the model to learn the synchronization behavior.

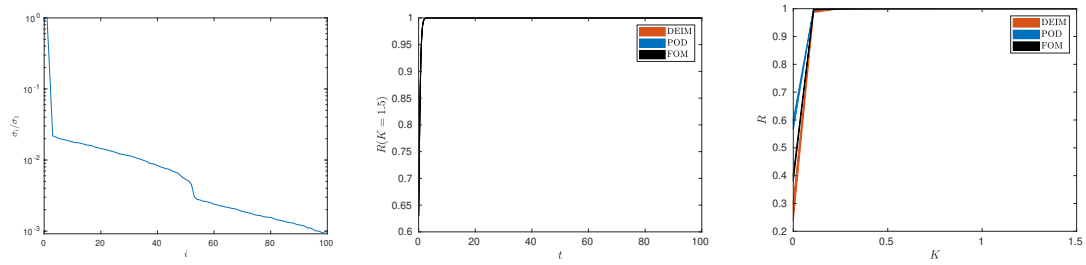
From Fig.3 and 5, we can clearly observe that for simple changes in the values of the natural frequencies ω_i, the reduced model is incapable of capturing and describing properly the synchronization dynamics of the Kuramoto model. Before moving on to the next section, we shall mention that having a model that is sensible to small changes in the input parameters can be disadvantageous in several manners, one of which is its restriction to *learn* the intrinsic dynamics for a larger set of parameters. We may say that two of the reasons why this happens is that the training set, when we construct the snapshot matrix S, does not have enough information about the sample space of the inputs when the variance is large and we use two different sets of natural frequencies during testing and training. Then when we change either the variance of the distribution of the ω_i or the ω_i for testing, the model finds it challenging to capture and describe the dynamics of the system since it has not *seen* such input values.

As a consequence, our idea is to help the model improve its performance, which is measured by how close the approximate solutions to the quantities of interest are to the full solution ones. We achieve this by sampling more uniformly the natural frequency space; that is, to generate the training set first we use the Sobol sequences technique (we also tested with Latin Hypercube sequence and it works practically similarly to Sobol's) to sample the hypercube $(0,1)^d$ more uniformly than when using the built-in Matlab random generator.

As we mentioned before, we observe that both σ^2 and sorting ω_i play an important role in how well the model is able to capture the dynamics; specifically, for those values of K in the transition phase from uncoupled to coupled dynamics. Similarly, the value of K is relevant as expected and even though using quasi-random numbers ameliorate the impact of changing the training and testing natural frequencies, its effect on the coupled dynamics of the system is evident, in particular, for small values of K, when the model struggles to explain the system dynamics for a fixed reduced model dimension.

We create random samples in the hypercube $(0,1)^d$, and then we make use of the inverse of the CDF of the normal distribution to generate $\omega_i \sim \mathcal{N}(0,\sigma^2)$.

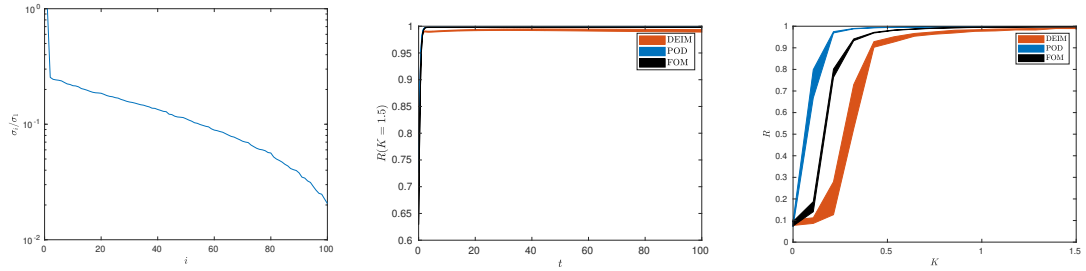
As a result, we have Fig.6, which is created using exactly the same parameters as Fig.3.



(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

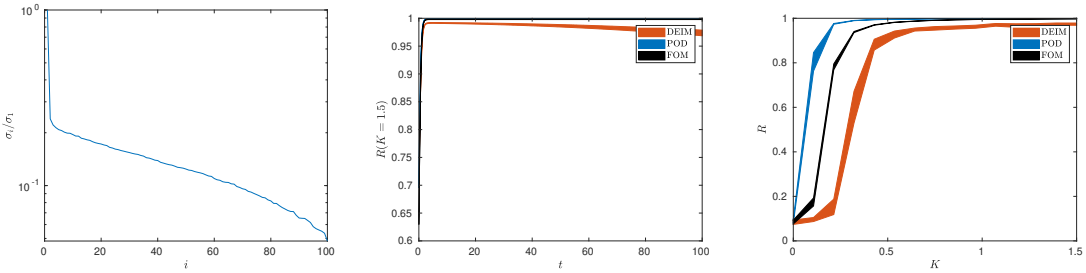
Figure 6: Quantities of interest for Sobol sequences, parameters of Fig.3, $k = 20$

Setting $\sigma^2 = 0.10$ and using a different set of ω_i for training and testing, we obtain Fig.7.



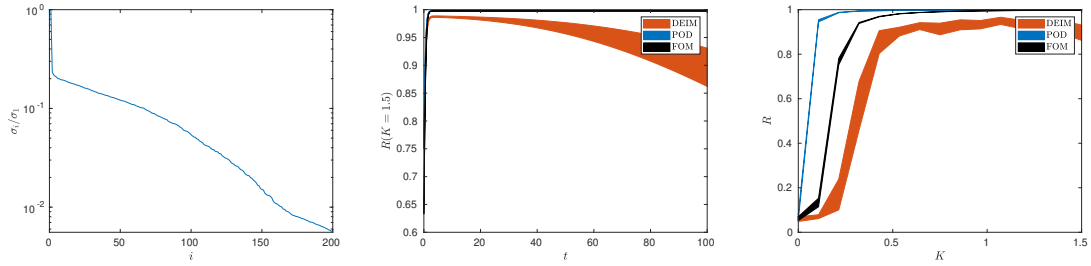
(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

Figure 7: Quantities of interest for Sobol sequences when changing σ^2 and testing ω_i, $k = 20$



(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

Figure 8: Quantities of interest for Sobol sequences and increasing N_{MC}, $k = 20$



(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

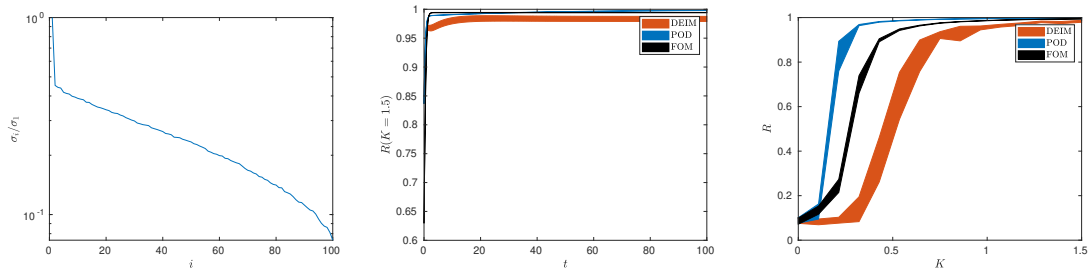
Figure 9: Quantities of interest when increasing N to 100, $k = 20$

In Fig.8, we keep the same parameters of Fig.7 but increase the number of Monte Carlo samples N_{MC} to 100. We comment on the fact that the singular values in Fig.8.a is interestingly slower than in Fig.Fig.7. a but in both cases the models can reproduce the synchronization properties of the full model. We may think about this as a consequence of the intrinsic heterogeneity of the uncoupled model, which is hardly reducible for a larger number of Monte Carlo samples. This means what computing more N_{MC} could lead to a reduced model whose dimension is close to the original one, then we would lose the computational advantage of MOR techniques.

On the other hand, in Fig.9, we keep the same parameters of Fig.7 but increase the number of oscillators N to 200. It can be observed that in this case, the DEIM struggles capturing the synchronization behavior of the model and that we have a proper decay in the singular values.

We now show similar plots but for ω_i following different distributions given a fixed set of parameters depending on each distribution.

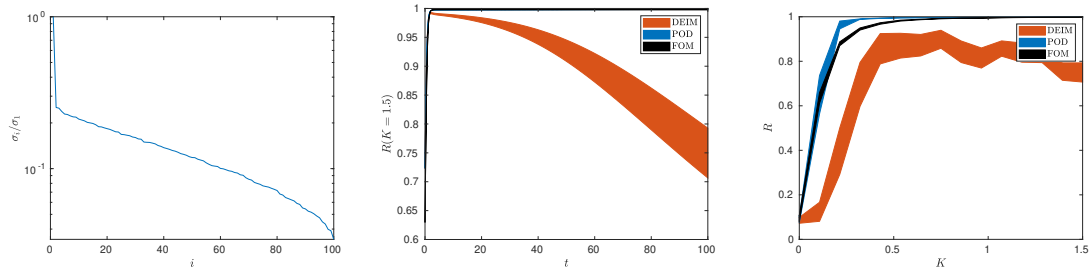
Beta distribution $\alpha = 2$ and $\beta = 5$, $\sigma^2 = 0.025$



(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

Figure 10: Quantities of interest for Sobol sequences and beta distribution, $k = 20$

Gamma distribution $\alpha = 0.90$ and $\beta = 0.10$, $\sigma^2 = 0.0064$. We use the inverse incomplete gamma distribution to compute from the Sobol sequence.



(a) Singular value decay. (b) R as a function of t. (c) R as a function of K.

Figure 11: Quantities of interest for Sobol sequences and gamma distribution, $k = 20$

From Fig.10 and Fig.11, we observe that Sobol sequences and the *simplified* version of DEIM can struggle to capture the synchronization characteristics of the full model. In addition, from the same figures, we observe that the Sobol sequence method work properly even for more complex distribution such as the beta and the gamma distributions.

6.1 Circadian oscillators

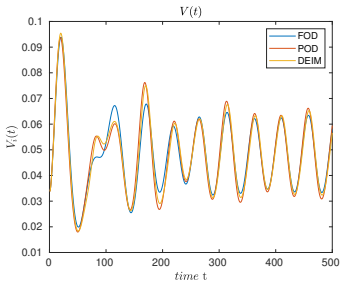
We use the model parameters shown in [2]; that is for each oscillator $\upsilon_1 = 0.7$, $\upsilon_2 = \upsilon_4 = \upsilon_6 0.35$, $\upsilon_8 = 1$, $K_1 = K_2 = K_4 = K_6 = K_8 = 1$, $k_3 = k_5 = 0.7$, and $k_7 = 0.35$. We also fix $\upsilon_c = 0.40$, $K_c = 1$, and $\omega = 2\pi 24$. Furthermore, we have $N = 100$ oscillators in the network and use again a Runge-Kutta scheme with time step $\Delta = 0.5$ and final time $T = 500$. The parameters τ_i follow a uniform distributed in the interval $[a, b]$. Regarding the initial condition for each oscillators –this is different from the Kuramoto's case – we first identify a state that belongs to the limit cycle of a single oscillator. This is performed by simulating a single oscillator for a sufficiently large time T and then selecting a state in the limit cycle. Afterwards, we assume that all the oscillators start at this synchronized state. As a result, we fix exactly the same initial state for each oscillator. That is, the initial $\mathbf{x}_0$ is

$$\mathbf{x}_0 = [0.0998, 0.2468, 2.0151, 0.0339]^T.$$

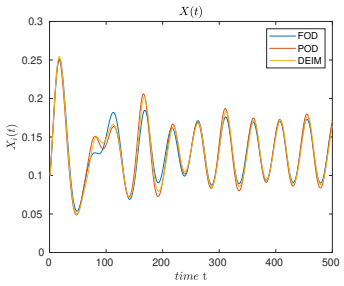
Regarding the coupling strengths $K_{\max}$ $L_{0,\max}$, we sample them in the intervals $[0, K_{\max}]$ and $[0, L_{0,\max}]$. As in [2], we initially fix $K_{\max} = 0.6$ and $L_{0,\max} = 0.02$ and we select 5 and 3 subintervals, respectively. Thus we have $q = 15$ different coupling strengths. Recall that we have four coordinates in this model; in Kuramoto's, it was two.

On the other hand, DEIM is applied to the fourth coordinate or variable V_i by selecting $r = 10$ oscillators. To study the synchronization properties, we focus on the evolution of the the quantities of interest previously introduced as a function of the coupling strengths and τ_i, which corresponds to what we plot below. We restrict our simulations to sampling the τ_i parameter using Sobol sequences for the uniform distribution in the interval $[a, b]$.

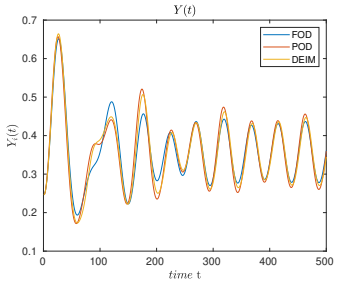
Regarding our numerical results, we simulate the quantities of interest for the Circadian oscillators using $a = 0.80$ and $b = 1.2$ as proposed in [2], which gives Fig.12, 15a and 13. These figures use the default random number generator in Matlab.



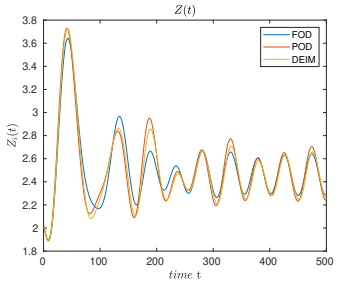
(a) Solution to $V(t)$.



(b) Solution to $X(t)$.



(c) Solution to $Y(t)$.



(d) Solution to $Z(t)$.

Figure 12: Approximated solution to our quantities of interests, parameters according to [2].

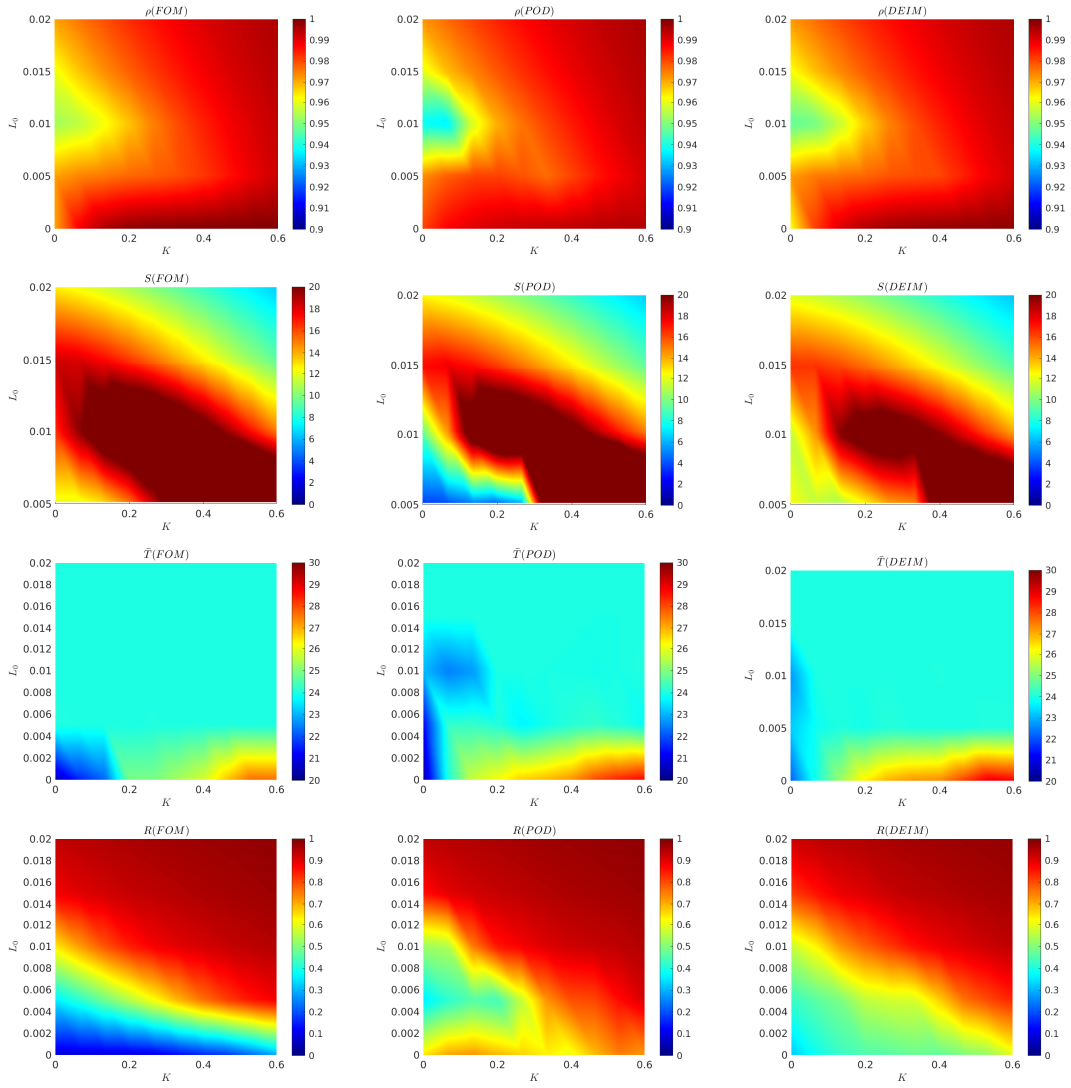
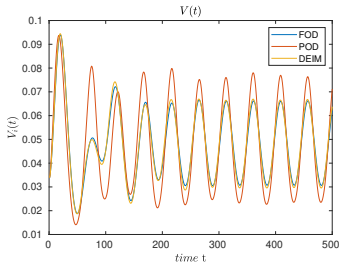
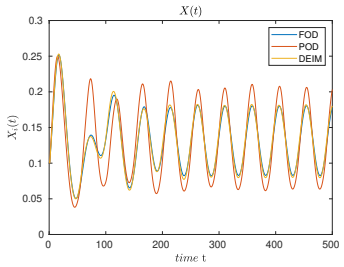


Figure 13: Evolution of quantities of interests as function of strength parameters, parameters according to [2].

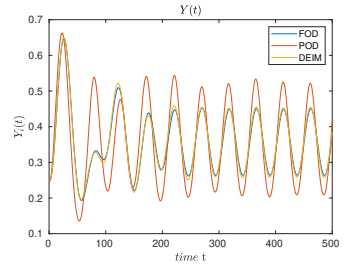
When we keep the same parameters of Fig.13 but change the values of the interval where τ_i is distributed to $a = 0.5$ and $b = 1.2$, we have Fig.14, 15b and 16. These figures also use the default random number generator in Matlab.



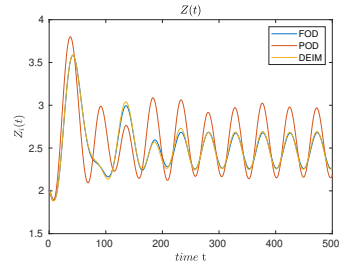
(a) Solution to $V(t)$.



(b) Solution to $X(t)$.

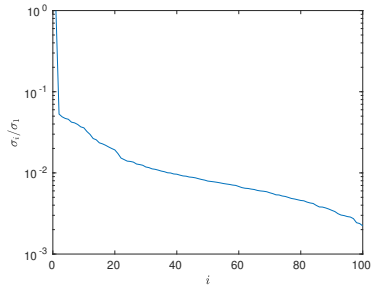


(c) Solution to $Y(t)$.

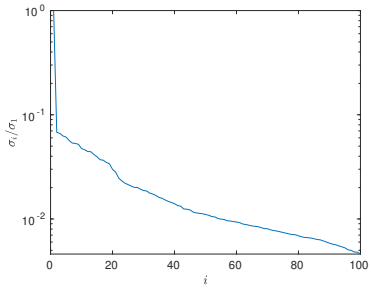


(d) Solution to $Z(t)$.

Figure 14: Approximated solution to our quantities of interests changing $[a, b]$.



(a) Singular value decay changing $[a, b]$.



(b) Singular value decay changing a and b.

Figure 15: Singular value decay.

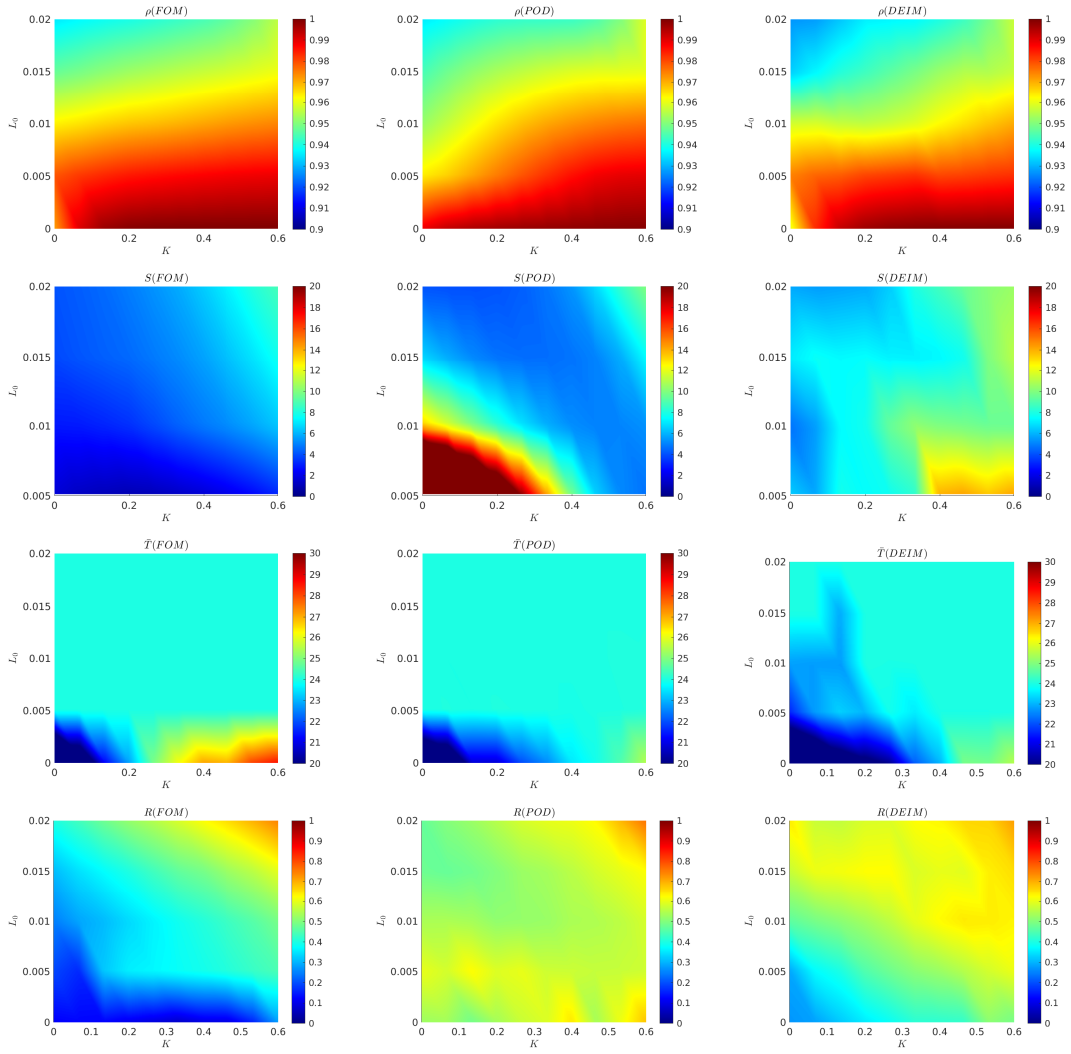
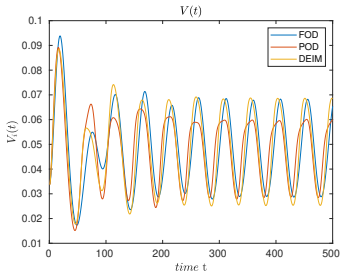
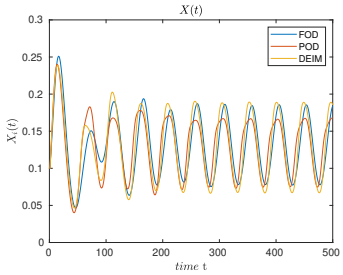


Figure 16: Evolution of quantities of interests as function of strength parameters changing $[a, b]$.

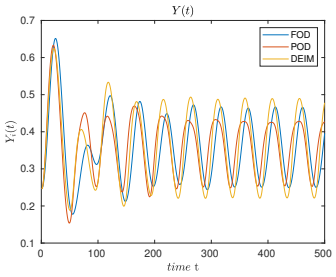
We now reproduce Fig.14, 15b and 16 but using Sobol sequences to generate τ_i with $a = 0.5$ and $b = 1.2$, which gives Fig.17, 18 and 19.



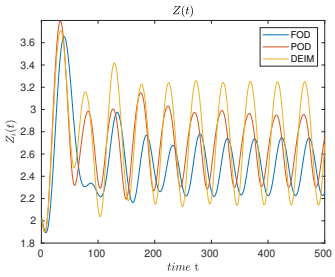
(a) Solution to $V(t)$, parameters according to [2].



(b) Solution to $X(t)$.



(c) Solution to $Y(t)$.



(d) Solution to $Z(t)$.

Figure 17: Approximated solution to our quantities of interests using Sobol sequences.

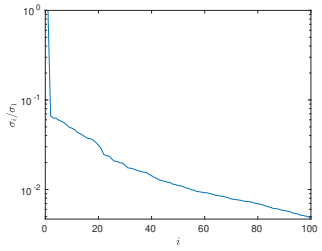


Figure 18: Singular value decay using Sobol sequences.

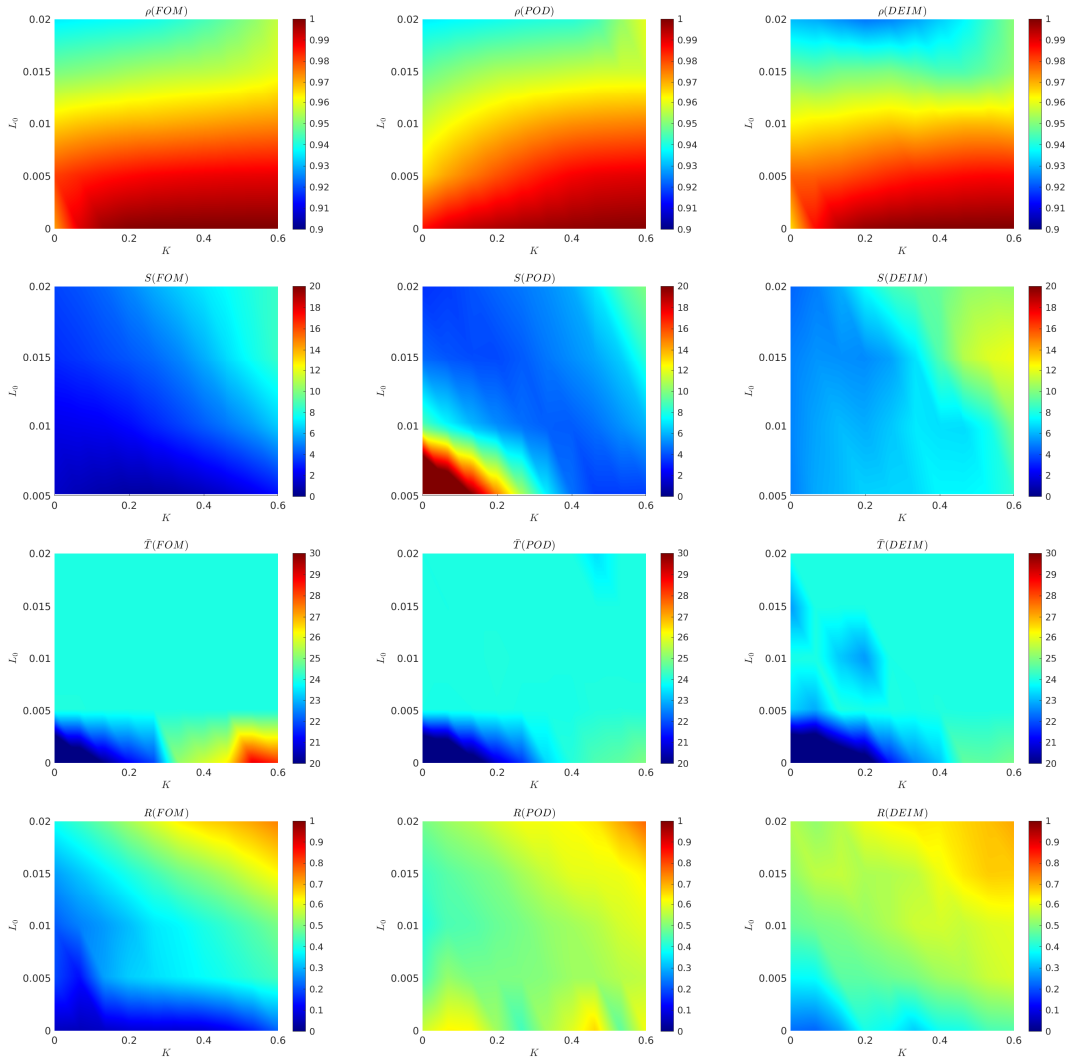


Figure 19: Evolution of quantities of interests as function of strength parameters using Sobol sequences.

We decide to enlarge the interval $[a, b]$ since by doing this, the probability of a specific value τ is smaller and then we have more options from which we select the τ_i. This means that we have more variability in such values, which is in fact, better for our reduced model since it knows about such a variability.

We observe that sampling τ_i by either in-built random generator or Sobol sequences does not have a significant impact on the quantities of interest as function of the strength coupling at least from a visual perspective. Nevertheless, we shall mention that there is a difference between the time evolution of the transcriptional inhibitor $Z(t)$ when using the random generator in Matlab and Sobol sequences, Fig.14d and Fig.17d. It seems that the Sobol method makes $Z(t)$ to stay a

little distant from FOM and POD solutions.

On the other hand, the singular value decay behave quite alike in Fig.15d and Fig.18.

We may then state that in terms of the parameter τ_i, using Sobol sequences does not have an important impact on the performance of the reduced model. One of the reasons can be that we are actually the full DEIM in the Circadian oscillator plots and thus the sampling method does not play a crucial role for model accuracy.

7 Future work

Related to future work and improvements, if we require the natural frequencies to follow a more complex, application-specific distribution, we may need more sophisticated methods like importance sampling. Under such circumstances, we could still make use of quasi-random sequences as long as some condition are fulfilled but this process becomes much more cumbersome.

Moreover, we could dtudy whether using the original form of DEIM, computing both matrices U and V improves the results shown here by quasi-random numbers; however, based on the results, this might not be the case.

In addition, we can also think about analyzing the impact of generating quasi-random numbers on the total computational cost when having more complex distribution. However, according to the results and in case Kuramoto model, we could state that such a cost plus the cost of computing the inverse of the CDF could be balance to the savings we incur due to using an incomplete version of DEIM, which involves the computation of an extra matrix, i.e. V.

In case of the Circadian oscillators, we might study the performance of using different distributions for the τ_i values.

8 Conclusions

Our results show that even when using a *simplified* version of DEIM, the results seem to capture and describe better the synchronization behaviour of the model because of quasi-random numbers.

As discussed, we may say that the variance is a crucial parameter for our reduced model to work properly. This is important to point out since there can be cases where the variance is too large that even using quasi-random numbers, the reduced model does not capture the coupled properties of the model. Under these circumstances, we could use the full DEIM or select more left-singular vectors from the SVD of the snapshot matrix; both of these increase the total computational cost. This also makes sense since the variance, intuitively, measures how a data set is spread out. We comment on the fact that under such circumstances each left-singular vector contains an *equal* quantity of information about the system dynamics and then discarding some of them could lead to model that poorly behaves.

Equally important is to mention that the negative impact of a large variance of the natural frequencies on the reduced model performance can be mitigated by a stronger strength coupling term, which was observed in the previous figures. Nevertheless, mathematically this is can be carried out, but there must exist a limit on such quantity such that the system is physically realizable. Trivially, if the strength parameter goes to infinity, all oscillators will be coupled no matter how large the variance is. Unless the mean value, the variance cannot be simply set to zero and thus its value can be seen as an intrinsic parameter once a set of natural frequencies is fixed.

Furthermore, it was shown that the number of oscillators can impact the reduce model performance since the more oscillators we add to the network of oscillators, the more information about the natural frequency space the model has and the more variability there is.

We believe that the reduced model shown here works. This also applies to the case when we do not use any quasi-random-number strategy. It is also relevant to mention that when the variance is sufficiently small, we could just sort the natural frequencies and the model may still learn the synchronization properties of the system. However, if the variance is large, using quasi-random number can lead to improvements on the model performance.

Particularly interesting is to notice that even thought we did not use the full DEIM, the results seem to capture and describe better the synchronization behaviour of the model when we used quasi-random numbers over standard sampling.

Bibliography

[1] Saifon Chaturantabut and Danny C Sorensen. "Nonlinear model reduction via discrete empirical interpolation". In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2737–2764.

[2] Niccolò Discacciati and Jan S Hesthaven. "Modeling synchronization in globally coupled oscillatory systems using model order reduction". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.5 (2021), p. 053127.

[3] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of monte carlo methods*. Vol. 706. John Wiley & Sons, 2013.

[4] MathWorks. *Generating Quasi-Random Numbers*. 2022. URL: `https://ch.mathworks.com/help/stats/generating-quasi-random-numbers.html` (visited on 12/20/2021).

[5] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*. Vol. 92. Springer, 2015.

[6] Steven H Strogatz. "From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators". In: *Physica D: Nonlinear Phenomena* 143.1-4 (2000), pp. 1–20.

[7] Wikipedia. *Low-discrepancy sequence*. 2022. URL: `https://en.wikipedia.org/wiki/Low-discrepancy_sequence` (visited on 01/08/2022).