HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Evaluation of Dynamic Mode Decomposition for Cascaded Electrical Oscillators

Scientific Project

Course of Studies Mechatronics
at Ravensburg-Weingarten University of Applied Sciences

by
Daniel Peters

March 2023

Student Number	35115
Supervisor	Prof. Dr. Lothar Berger
	Stephan Scholz M.Sc.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

DMD	*Dynamic Mode Decomposition*
DMDc	*Dynamic Mode Decomposition with Control*
ECU	*Electronic Control Unit*
RLC	*Resistor Inductor Capacitor*
SISO	*Single Input Single Output*
SVD	*Singular Value Decomposition*

1. Task Definition

Since many years, people have been interested in finding ways to use data in control theory [1]. For classical control approaches, physical dynamic models are necessary for a fairly accurate description of the actual underlaying behavior of the system. To obtain such models for complex dynamical systems is often an extremely hard task to achieve [2]. Therefore, the problem of finding such a model from measured data has become a mature research field [1] because data-driven models are based on observations and measurements of the true system and only require a minimum amount of prior knowledge of the system [2].

One data-driven algorithm developed for system identification and dimensionality reduction is Dynamic Mode Decomposition (DMD).

In this work, the functionality of this algorithm will be demonstrated for n-fold Electrical RLC Oscillators.

First, the pure DMD algorithm is explained. Then, a further development of DMD for input output systems is tested, with which the effects of system control and open loop dynamics can be separated. Finally, the results are used to develop a full state feedback controller that can be used to control a system with a reduced dimension model.

2. System Analysis

The experimental system to which the DMD algorithm is to be applied consists of $\boldsymbol{n}$ RLC elements connected in series, as shown in Figure 1.

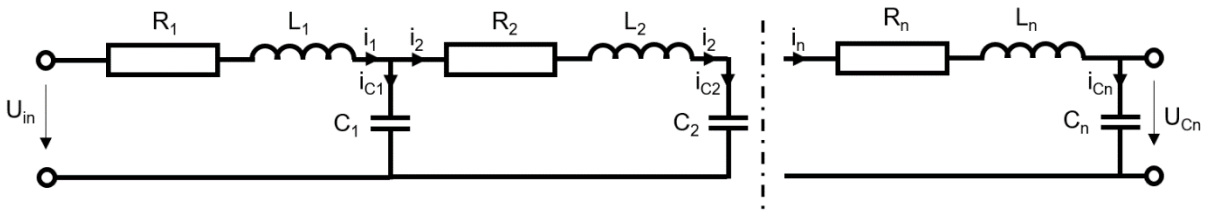


Figure 1: Considered Cascaded RLC Circuit

If one knows the voltage over each component, these systems can be modeled easily using differential equations with Kirchhoff's mesh and node law [3]. Since the state space representation of an n-fold system can be simply derived, this system is taken as a show case because the properties can be easily determined for an arbitrary configuration and compared to the results of DMD.

In the following, the state space representation for an nth-order system according to Cano et al [3, pp. 12-16] is explained shortly.

The mesh equation per mesh is based on the previous voltage and the node equation per node is based on the upcoming current. Therefore, a predictable pattern can be noticed, resulting in following equations

$$\begin{aligned} \mathbf{i_n}(\mathbf{t}) &= \mathbf{C_n}\dot{\boldsymbol{u}}_{\boldsymbol{Cn}}(\boldsymbol{t}), \\ \mathbf{i_{n-1}}(\mathbf{t}) &= \mathbf{C_{n-1}}\dot{\boldsymbol{u}}_{\boldsymbol{Cn-1}}(\boldsymbol{t}) + \mathbf{C_n}\dot{\boldsymbol{u}}_{\boldsymbol{Cn}}(\boldsymbol{t}), \\ &\dots \\ \boldsymbol{i_1}(\boldsymbol{t}) &= \sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} \mathbf{C_n}\dot{\boldsymbol{u}}_{\boldsymbol{Cn}}(\boldsymbol{t}) \end{aligned} \tag{1}$$

and

$$\begin{aligned} \boldsymbol{u}_{\mathbf{in}}(\mathbf{t}) = \boldsymbol{u}_{\mathbf{C0}}(\boldsymbol{t}) &= \boldsymbol{R_1 i_1}(\boldsymbol{t}) + \boldsymbol{L_1}\frac{\boldsymbol{di_1}(\boldsymbol{t})}{\boldsymbol{dt}} + \boldsymbol{u_{c1}}(\boldsymbol{t}), \\ \boldsymbol{u}_{\mathbf{C1}}(\boldsymbol{t}) &= \boldsymbol{R_2 i_2}(\boldsymbol{t}) + \boldsymbol{L_2}\frac{\boldsymbol{di_2}(\boldsymbol{t})}{\boldsymbol{dt}} + \boldsymbol{u_{c2}}(\boldsymbol{t}), \\ &\dots \\ \boldsymbol{u}_{\mathbf{Cn-1}}(\boldsymbol{t}) &= \boldsymbol{R_n i_n}(\boldsymbol{t}) + \boldsymbol{L_n}\frac{\boldsymbol{di_n}(\boldsymbol{t})}{\boldsymbol{dt}} + \boldsymbol{u_{cn}}(\boldsymbol{t}). \end{aligned} \tag{2}$$

Combining equations 1 and 2 results in

$$\begin{aligned} u_{\text{in}}(t) &= R_1\left[\sum_{n=1}^{N} C_n \dot{u}_{Cn}(t)\right] + L_1\left[\sum_{n=1}^{N} C_n \ddot{u}_{Cn}(t)\right] + u_{C1}(t), \\ u_{C1}(t) &= R_2\left[\sum_{n=2}^{N} C_n \dot{u}_{Cn}(t)\right] + L_2\left[\sum_{n=2}^{N} C_n \ddot{u}_{Cn}(t)\right] + u_{C2}(t), \\ &\dots \\ u_{Cn-1}(t) &= R_n C_n \dot{u}_{Cn}(t) + L_n C_n \ddot{u}_{Cn}(t) + u_{Cn}(t), \end{aligned} \tag{3}$$

from which the matrix form

$$M\ddot{z}(t) + D\dot{z}(t) + Sz(t) = Gu(t) \tag{4}$$

can be read. Rewriting this form to

$$\ddot{z}(t) = -M^{-1}D\dot{z}(t) - M^{-1}Sz(t) + M^{-1}Gu(t) \tag{5}$$

and introducing new variable names $x_1(t) = z(t)$, $x_2(t) = \dot{z}(t)$ the final state space notation

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}S & -M^{-1}D \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}G \end{bmatrix} u(t) \tag{6}$$

is created.

3. Dynamic Mode Decomposition

3.1 Fundamentals

Dynamic Mode Decomposition (DMD) is a mathematical technique used for analyzing and modeling complex dynamical systems. It was initially discussed by Peter J. Schmid in 2010 [4]. DMD is particularly useful when dealing with systems that are too complex to be analyzed directly, but whose behavior can be observed or measured [5].

The approach involves decomposing the system's behavior into a set of dynamic modes each of which represents a distinct oscillation or pattern in the measurements. These dynamic modes, together with corresponding eigenvalues, can be used to approximate the systems behavior over time, making it easier to understand and predict [6]. So, the algorithm consists of collecting data over time and constructing a matrix that describes the systems dynamics. The matrix is decomposed using linear algebra techniques (such as Singular Value Decomposition – SVD) to extract the dynamic modes [6].

DMD has been applied to a wide range of systems in various fields, including fluid dynamics, neuroscience, finance and materials science. However complex, many of these systems evolve on a low-dimensional attractor that may be characterized by spatiotemporal coherent structures (dynamic modes) [7].

In the following, the algorithm is explained in detail according to Kutz et al [6] [8].

At basic level, DMD examines the relationships between pairs of data from a dynamical system. The measurements $\boldsymbol{x_k}$ and $\boldsymbol{x_{k+1}}$ are taken to be roughly connected by a linear operator

$$\boldsymbol{x_{k+1}} = \boldsymbol{A}\,\boldsymbol{x_k} \tag{7}$$

where $\boldsymbol{k}$ stands for the temporal iteration from a discrete dynamical system and $\boldsymbol{x_k} \in \mathbb{R}^{\boldsymbol{n}}$ and $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{n}}$. It is assumed that this approximation holds true for all measurement pairs.

The measurements can be taken at regular time intervals. Therefore, one sets up the two data matrices

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x_1} & \boldsymbol{x_2} & \cdots & \boldsymbol{x_{m-1}} \\ | & | & & | \end{bmatrix} \tag{8}$$

and

$$\boldsymbol{X'} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x_2} & \boldsymbol{x_3} & \cdots & \boldsymbol{x_m} \\ | & | & & | \end{bmatrix} \tag{9}$$

where $\boldsymbol{m}$ is the total number of measurement points and $\boldsymbol{X'}$ is the time-shifted measurement point matrix of $\boldsymbol{X}$. According to Formula 7, this is needed for the relationship

$$\boldsymbol{X'} = \boldsymbol{AX}. \tag{10}$$

To solve for operator matrix $\boldsymbol{A}$, it is theoretically possible to calculate as the following

$$\boldsymbol{A} = \boldsymbol{X'X}^{\dagger} \tag{11}$$

where $\dagger$ is the Moore-Penrose pseudoinverse. Therefore, $\boldsymbol{A}$ is a least-square regression, minimizing $\sum_{\boldsymbol{k}} \|\boldsymbol{x_{k+1}} - \boldsymbol{A}\,\boldsymbol{x_k}\|_2$. Due to high matrix dimensionality for high-order systems, this computation will be very inefficient. A faster method for determining the pseudoinverse is via the SVD with a truncation value $\boldsymbol{r}$. The matrix $\boldsymbol{X}$ will be thereby reduced in dimension

$$\boldsymbol{X} = \boldsymbol{U\Sigma V}^{*} = \begin{bmatrix} \widetilde{\boldsymbol{U}} & \widetilde{\boldsymbol{U}}_{\boldsymbol{rem}} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{\Sigma}} & \boldsymbol{0} \\ \boldsymbol{0} & \widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{rem}} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{V}}^{*} \\ \widetilde{\boldsymbol{V}}^{*}_{\boldsymbol{rem}} \end{bmatrix} \approx \widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{V}}^{*} \tag{12}$$

with $\boldsymbol{U} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{n}}, \boldsymbol{\Sigma} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{m-1}}, \boldsymbol{V}^{*} \in \mathbb{R}^{\boldsymbol{m-1}\times\boldsymbol{m-1}}$, $\widetilde{\boldsymbol{U}} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{r}}, \widetilde{\boldsymbol{\Sigma}} \in \mathbb{R}^{\boldsymbol{r}\times\boldsymbol{r}}, \widetilde{\boldsymbol{V}}^{*} \in \mathbb{R}^{\boldsymbol{r}\times\boldsymbol{m-1}}$ to compute $\overline{\boldsymbol{A}} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{n}}$ with

$$\overline{\boldsymbol{A}} = \boldsymbol{X'}\widetilde{\boldsymbol{V}}\,\widetilde{\boldsymbol{\Sigma}}^{-1}\widetilde{\boldsymbol{U}}^{*}. \tag{13}$$

Note: * denotes the complex conjugate transpose.

If the number of states is significantly greater than one and if the truncation value is significantly smaller than the number of states, then there is an even computationally more efficient way to calculate the model matrix. This is done by projecting the measurement vectors $\boldsymbol{x_k}$ to a low-rank subspace of dimension $\boldsymbol{r}$ with the transformation

$\tilde{x}_k = \tilde{U}^* x_k$. For the model matrix $\tilde{A}$ this results in the transformation $\tilde{A} = \tilde{U}^* \bar{A} \tilde{U}$. Therefore, the final reduced-order matrix $\tilde{A} \epsilon \mathbb{R}^{r \times r}$ is calculated by

$$\tilde{A} = \tilde{U}^* X' \tilde{V} \tilde{\Sigma}^{-1} \tag{14}$$

which has many of the same eigenvalues as $\bar{A}$.
By the eigendecomposition of $\tilde{A}$ one gets the eigenvalues Λ and eigenvectors W. The dynamic modes of A can be obtained by calculating

$$\Phi = X' \tilde{V} \tilde{\Sigma}^{-1} W. \tag{15}$$

Once the dynamic modes and the reduced-order matrix are obtained, there are various ways to use them. Two use cases are:

- Mode Analysis: One can analyze the spatial and temporal patterns of each dynamic mode to gain insights into the behavior of the system. For example, you can identify the dominant frequencies [9].

- Reduced-Order-Modeling: One can use the dynamic modes to develop reduced-order models of the system. By representing the systems behavior in terms of its dominant dynamic modes, one can reduce the dimensionality of the state space representation (reduced-order state matrix). This can be useful for predicting the systems behavior and designing control strategies that are computationally efficient [8].

Mode Analysis can be applied to the RLC system using the native DMD approach (Chapter 3.3). For prediction and controller design, an extension of the DMD algorithm is needed since the system has an external control input. This will be discussed in Chapters 4 and 5.

3.2 Measurement Data Generation

Based on the derivations in Chapter 2, a Matlab algorithm was written by Gres [3] to represent the state space for an n-fold system. This algorithm is used in this work to produce time-series data as input for DMD algorithm. In this chapter, as a showcase, a system with 50 RLC circuits (= 100 states) and the parameters R=1, L=1 and C=1 is chosen. Here it should be said in advance: The results for variations of the parameters can be derived analogous.

First, the state space of this system must be discretized. The Matlab command *c2d* is used for this. The sampling time is set to 0.1 seconds (here one has to be careful that if the sampling time is too large, then due to the Shannon-Nyquist theorem, the dynamic behavior could be distorted). To get usable measurement data, these must only come from an unforced system, since the native DMD algorithm cannot distinguish between system dynamics and control input. For this purpose, the system response is generated for initial states (all states set to 1) using the Matlab command *initial*.

3.3 Mode Analysis

As already mentioned, the Dynamic Modes represent a pattern, the temporal behavior is expressed with the associated eigenvalues. The general behavior of a mode can be stated depending on the position of the eigenvalues [9]:

- $\mathbf{\Lambda}$ inside unit circle: decaying mode
- $\mathbf{\Lambda}$ outside unit circle: growing mode
- $\mathbf{\Lambda}$ on unit circle: oscillating mode

To analyze the temporal behavior in detail, the DMD eigenvalue $\mathbf{\Lambda}$ (see formula (15)) is converted first to a continuous-time eigenvalue $\mathbf{\Omega}$ with following formula [9]

$$\mathbf{\Omega} = \frac{\mathbf{ln}(\mathbf{\Lambda})}{\mathbf{\Delta t}} = \boldsymbol{\beta} + \mathbf{j}\boldsymbol{\omega}. \tag{16}$$

Then,

$$\boldsymbol{e}^{\mathbf{\Omega t}} = \mathbf{e}^{\boldsymbol{\beta t}}\, \boldsymbol{cos}(\boldsymbol{\omega t}) + \boldsymbol{j}\mathbf{e}^{\boldsymbol{\beta t}}\boldsymbol{sin}(\boldsymbol{\omega t}) \tag{17}$$

can be used to calculate the behavior over time [9].

Figure 2 shows the temporal behavior of the modes for the example system where the truncation value is equal to the dimension of the original system (no dimension reduction to get all modes).

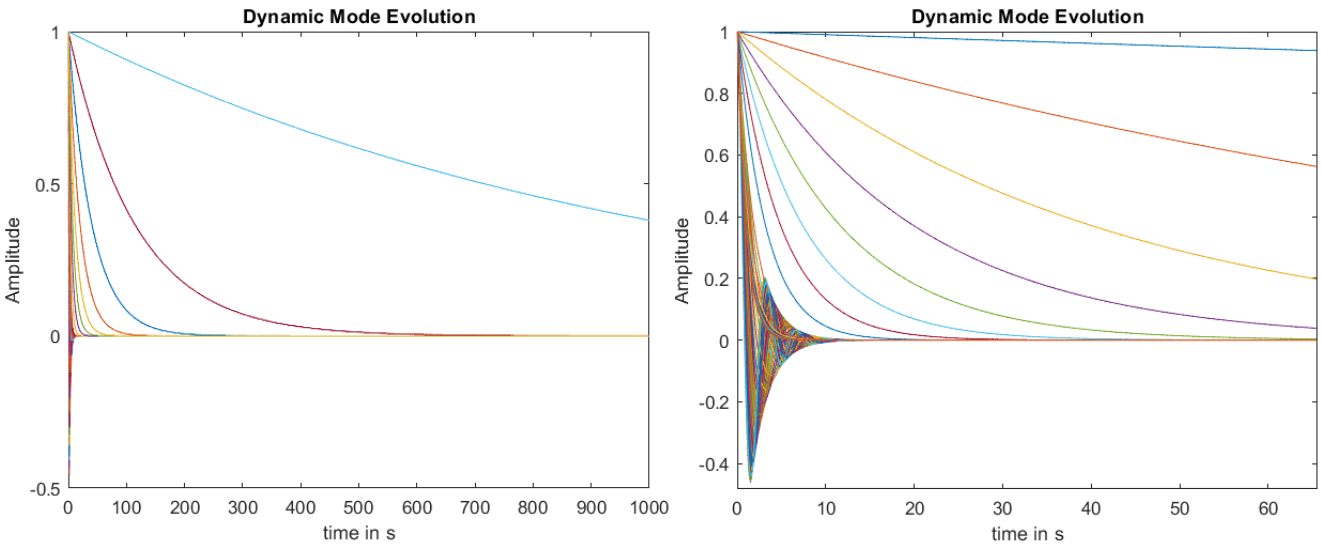


Figure 2: Dynamic Mode Evolution of 50 RLC Circuits

As one can see, most modes fall off directly and converge to 0. So, most modes have an influence only shortly after the impulse and then become vanishing. Only a few modes dominate the behavior over time. The following table shows the 10 in absolute value smallest (corresponding to the slowest) continuous eigenvalues of the system. Here one can see that there is a significant jump between the 8th and 9th eigenvalue.

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$\|\|\boldsymbol{\Omega_i}\|\|$	0.00097	0.0088	0.0248	0.0497	0.085	0.1338	0.2022	0.3097	0.5226	0.5226	0.5824
$\frac{\|\|\boldsymbol{\Omega_i}\|\|}{\min(\|\|\boldsymbol{\Omega_i}\|\|)}$	1	9.0651	25.56	51.31	87.88	138.22	208.77	319.77	538	538	600

Table 1: 10 slowest Eigenvalues of 50 RLC Circuits

This is also visually depicted in Figure 3.

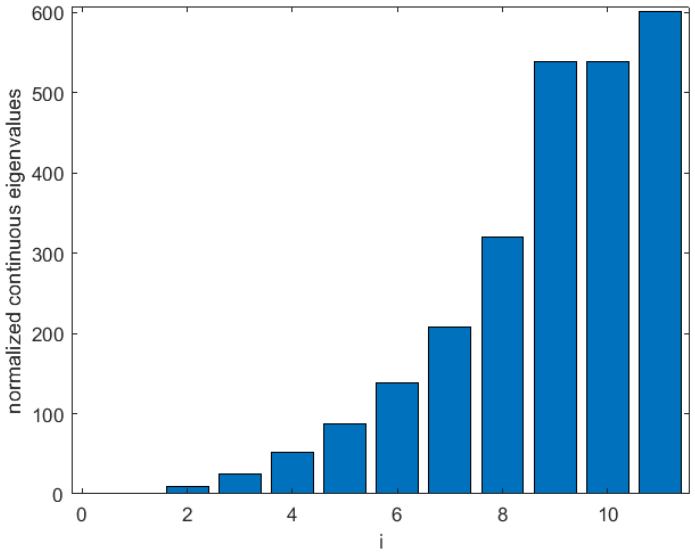
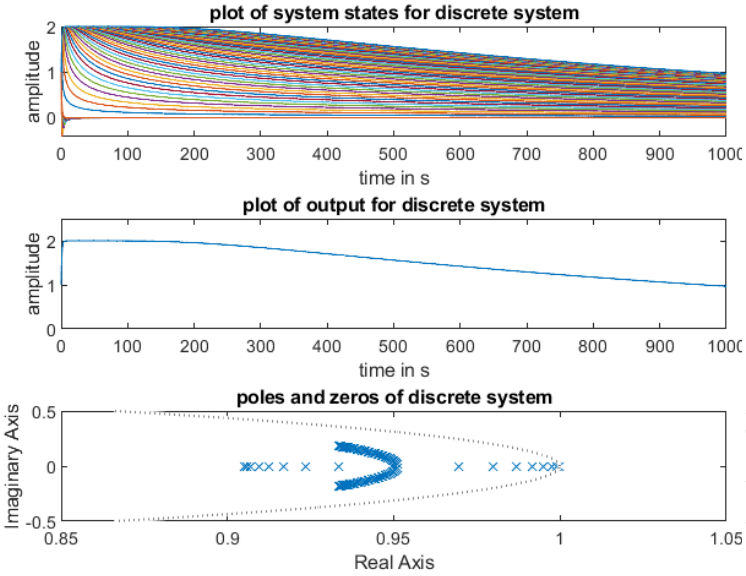


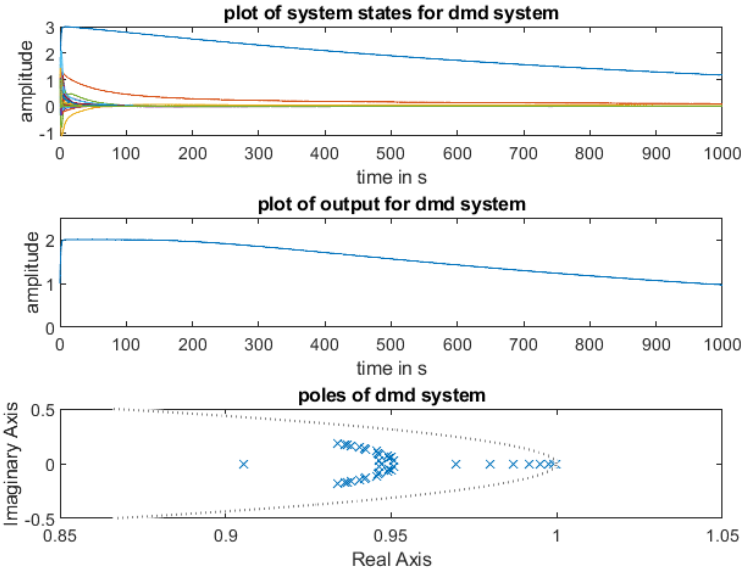
Figure 3: 10 smallest normalized Eigenvalues of 50 RLC Circuits

Therefore, the assumption is close that the system can be represented by a system with 8 modes resp. 8 states to get the best tradeoff between accuracy and dimension reduction. To verify this, the DMD systems are created with 40 states, 8 states, 4 states and 2 states and the output of the systems are reconstructed and compared with the output of the original system. To calculate the output of the reduced systems, the output matrix $\boldsymbol{C}$ for these systems is missing so far. This is calculated artificially in this case, using the expected output (which is the same as the output of the original system) by rearranging the output equation of the state space:

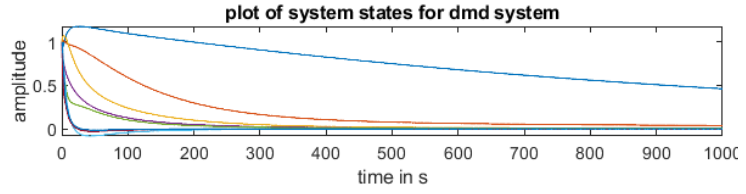
$$\boldsymbol{C_{DMD}} = \boldsymbol{y_{orig}} \cdot \boldsymbol{x}^{\dagger} \tag{18}$$

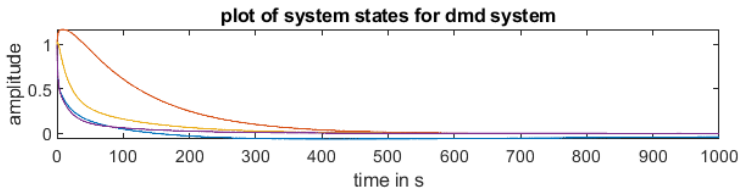
Figure 4a shows the behavior of the original system. The temporal behavior of the output corresponds quite exactly with the output of the system with 40 states (Fig. 4b). This also holds for the system with 8 states in Figure 4c (q.e.d.). In the system with 4 states (Fig. 4d) there is a visible deviation in the first 100 seconds after the stimulus, but subsequently this course also matches. The assumption for the deviation is that here the unrepresented modes are noticeable in the first 100 seconds (see Figure 2).

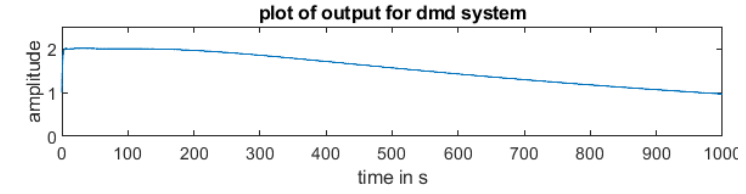


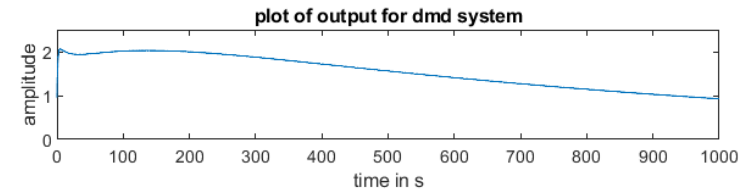


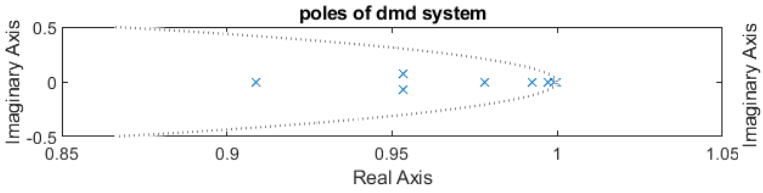
(a) (b)

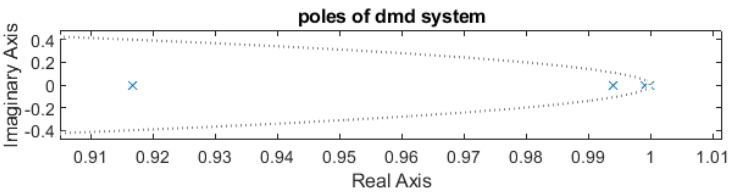




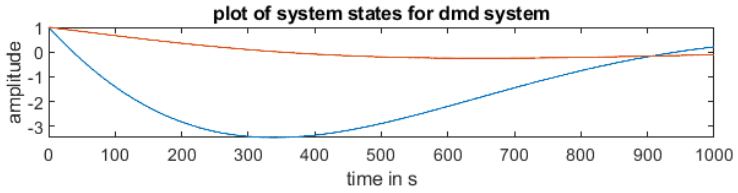


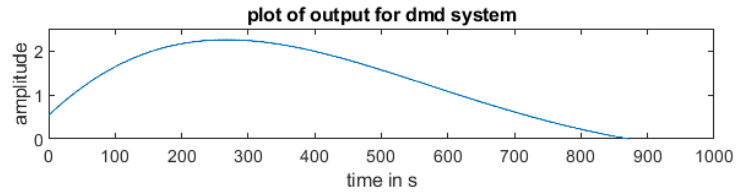


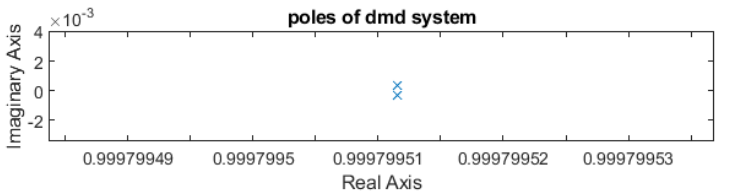




(c) (d)







(e)

Figure 4: Result of DMD, (a) original system, (b) system with 40 states, (c) 8 states, (d) 4 states, (e) 2 states

So, for systems that have only slow changing setpoints the representation with 4 states could also work fine, for systems that have many fast setpoint changes this is likely to be problematic (will be shown in chapter 4). The temporal output of the system with 2 states (figure 4e) does not agree at all with the original system. It confirms the assumption that here too few dominant modes are used to represent the behavior.

For all 5 systems, the eigenvalues or poles lie within the right-hand side of the unit circle. This fits the plot of the Dynamic Modes since all modes show a decaying behavior.

The time evolution of the states for the reduced systems is similar to that of the dynamic modes. In Fig. 4b one can see that there are many states that decay quickly and a few that show a very damped behavior.

4. Dynamic Mode Decomposition with Control

4.1 Fundamentals

Dynamic Mode Decomposition with Control (DMDc) was developed by Kutz et al in 2016 [8]. It is an extension of the traditional DMD technique. DMDc incorporates the control input information into the DMD analysis to identify the influence of external control inputs on the system's dynamic [6]. It is therefore particularly useful for analyzing and predicting the behavior of controlled systems that are subject to external input. It also works by decomposing the systems input-output data into a series of spatial modes and corresponding temporal dynamics [6].
The control input is incorporated into the analysis by adding it as an additional input to the DMD algorithm. This allows DMDc to identify the effect of the control input on the systems behavior and predict how the system will respond to different control inputs in the future [8]. In the following, the algorithm according to Kutz et al [6] [8] is explained.
If a system has a control input, Formula 7 for calculating the future state vector, changes as follows:

$$\boldsymbol{x_{k+1}} \approx \boldsymbol{A}\,\boldsymbol{x_k} + \boldsymbol{B}\,\boldsymbol{u_k} \tag{19}$$

In addition to the state measurement data matrices $\boldsymbol{X}$ und $\boldsymbol{X'}$, this results in another one for the control input $\boldsymbol{\Upsilon}$:

$$\boldsymbol{\Upsilon} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u_1} & \boldsymbol{u_2} & \cdots & \boldsymbol{u_{m-1}} \\ | & | & & | \end{bmatrix}, \boldsymbol{u_k} \in \mathbb{R}^{\boldsymbol{l}} \tag{20}$$

Using these three matrices, DMDc now attempts to determine an optimal approximate solution for the system matrix $\boldsymbol{A}$ and the input matrix $\boldsymbol{B}$.

From Formula 10 one gets with $\mathbf{G} = [\boldsymbol{A} \quad \boldsymbol{B}]$ and $\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{\Upsilon} \end{bmatrix}$

$$\boldsymbol{X'} = \boldsymbol{AX} + \boldsymbol{B\Upsilon} = [\boldsymbol{A} \quad \boldsymbol{B}] \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{\Upsilon} \end{bmatrix} = \mathbf{G}\,\boldsymbol{\Omega}. \tag{21}$$

To determine the two unknown matrices in G, it requires a conversion to $\boldsymbol{G}$

$$\boldsymbol{G} = [\boldsymbol{A} \quad \boldsymbol{B}] = \boldsymbol{X}' \, \boldsymbol{\Omega}^{\dagger} = \boldsymbol{X}' \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{\Upsilon} \end{bmatrix}^{\dagger}. \tag{22}$$

The best approximation solution corresponds again to the pseudoinverse of $\boldsymbol{\Omega}$. For this purpose, SVD is applied again, the truncation value is called $\boldsymbol{p}$ here. For $\boldsymbol{G}$ thus results

$$\boldsymbol{G} \approx \overline{\boldsymbol{G}} = \boldsymbol{X}' \widetilde{\boldsymbol{V}} \, \widetilde{\boldsymbol{\Sigma}}^{-1} \widetilde{\boldsymbol{U}}^{*}. \tag{23}$$

To extract matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ from $\boldsymbol{G}$, the matrix $\widetilde{\boldsymbol{U}}$ is split into two components

$$[\boldsymbol{A} \quad \boldsymbol{B}] \approx [\overline{\boldsymbol{A}} \quad \overline{\boldsymbol{B}}] \approx [\boldsymbol{X}' \widetilde{\boldsymbol{V}} \, \widetilde{\boldsymbol{\Sigma}}^{-1} \, \widetilde{\boldsymbol{U}}_1^{*} \qquad \boldsymbol{X}' \widetilde{\boldsymbol{V}} \, \widetilde{\boldsymbol{\Sigma}}^{-1} \widetilde{\boldsymbol{U}}_2^{*}] \tag{24}$$

where $\widetilde{\boldsymbol{U}}_1 \in \mathbb{R}^{n \times p}$ and $\widetilde{\boldsymbol{U}}_2 \in \mathbb{R}^{l \times p}$. However, to make this more computationally efficient, a reduced-order model with a significantly smaller rank is also sought here. For the transformation of $\boldsymbol{x_k}$, $\widetilde{\boldsymbol{U}}$ cannot be used as in DMD, since this matrix defines the input subspace instead, as it was obtained from the augmented matrix $\boldsymbol{\Omega}$, which includes the control inputs. A reduced-order subspace of the output space $\boldsymbol{X}'$ is employed in order to locate a linear transformation for the states alone. This is realized by performing a second SVD for $\boldsymbol{X}'$

$$\boldsymbol{X}' = \widehat{\boldsymbol{U}} \, \widehat{\boldsymbol{\Sigma}} \, \widehat{\boldsymbol{V}}^{*} \tag{25}$$

with truncation value $\boldsymbol{r}$, which is used to transform into a low-rank subspace, resulting in

$$\widetilde{\boldsymbol{A}} = \widehat{\boldsymbol{U}}^{*} \overline{\boldsymbol{A}} \, \widehat{\boldsymbol{U}} = \widehat{\boldsymbol{U}}^{*} \boldsymbol{X}' \, \widetilde{\boldsymbol{V}} \, \widetilde{\boldsymbol{\Sigma}}^{-1} \widetilde{\boldsymbol{U}}_1^{*} \widehat{\boldsymbol{U}}, \tag{26}$$

$$\widetilde{\boldsymbol{B}} = \widehat{\boldsymbol{U}}^{*} \overline{\boldsymbol{B}} = \widehat{\boldsymbol{U}}^{*} \boldsymbol{X}' \, \widetilde{\boldsymbol{V}} \, \widetilde{\boldsymbol{\Sigma}}^{-1} \widetilde{\boldsymbol{U}}_2^{*}$$

where $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{r \times r}$ and $\widetilde{\boldsymbol{B}} \in \mathbb{R}^{r \times l}$.
The dynamic modes of $\boldsymbol{A}$ can be determined similar to DMD:

$$\boldsymbol{\Phi} = \boldsymbol{X}' \widetilde{\boldsymbol{V}} \, \widetilde{\boldsymbol{\Sigma}}^{-1} \, \widetilde{\boldsymbol{U}}_1^{*} \widehat{\boldsymbol{U}} \, \boldsymbol{W} \tag{27}$$

Note: Since the truncation value of $\boldsymbol{\Omega}$ should be larger than that of $\boldsymbol{X'}$ [6], in this work $\boldsymbol{p}$ is set to $\boldsymbol{p = r + \dim\,(u_k)}$.
Note: There is also a variant for DMDc, for the case that B is already known. Since this is rarely the case in practice, this approach is not explained further here.

4.2 Measurement Data Generation and Forecasting

For the application of the DMDc algorithm, the measurement data are obtained similarly to the way of chapter 3.2. But the difference is that now a signal must be given to the control input, so that it can be divided between system dynamics and control input to get the system matrix $\boldsymbol{A}$ input matrix $\boldsymbol{B}$. A sinusoidal oscillation is used as control signal. The same system as in chapter 3 is used here again as an example system.

If the output matrix $\boldsymbol{C}$ is calculated as in chapter 3.3, one has a complete state space representation with which one can simulate or predict the behavior of the system. For testing, two different signals are used as control input. One is a sine wave with low frequency and the other is a square signal with higher frequency.
The results for the sine wave are good. The reduced DMD system with 40 states (Fig. 5e) reflects the course of the output of the original system well. With 8 states (Fig. 5g), the result is still very good, with a maximal deviation in the range 10^{-2}. With the DMD system with 4 states (Fig. 5i), the biggest deviations occur again in the first 100 seconds in range of 10^{-1}, but the subsequent course is very exact.
The results with the square signal look different. Here, the system with 40 states (Fig. 5f) again reproduces the output very accurately. With 8 states (Fig. 5h), the systems output follows the original output, but the signal is noisy with deviations in range 10^{0}. The output of the system with 4 states (Fig. 5j), however, is unusable and shows at most a trend. This is due to the unrepresented modes, which are important when a rapid signal change occurs.
Thus, the assumption from chapter 3.3 is confirmed that the accuracy of the dimension reduction depends on which control signal is applied to the system.

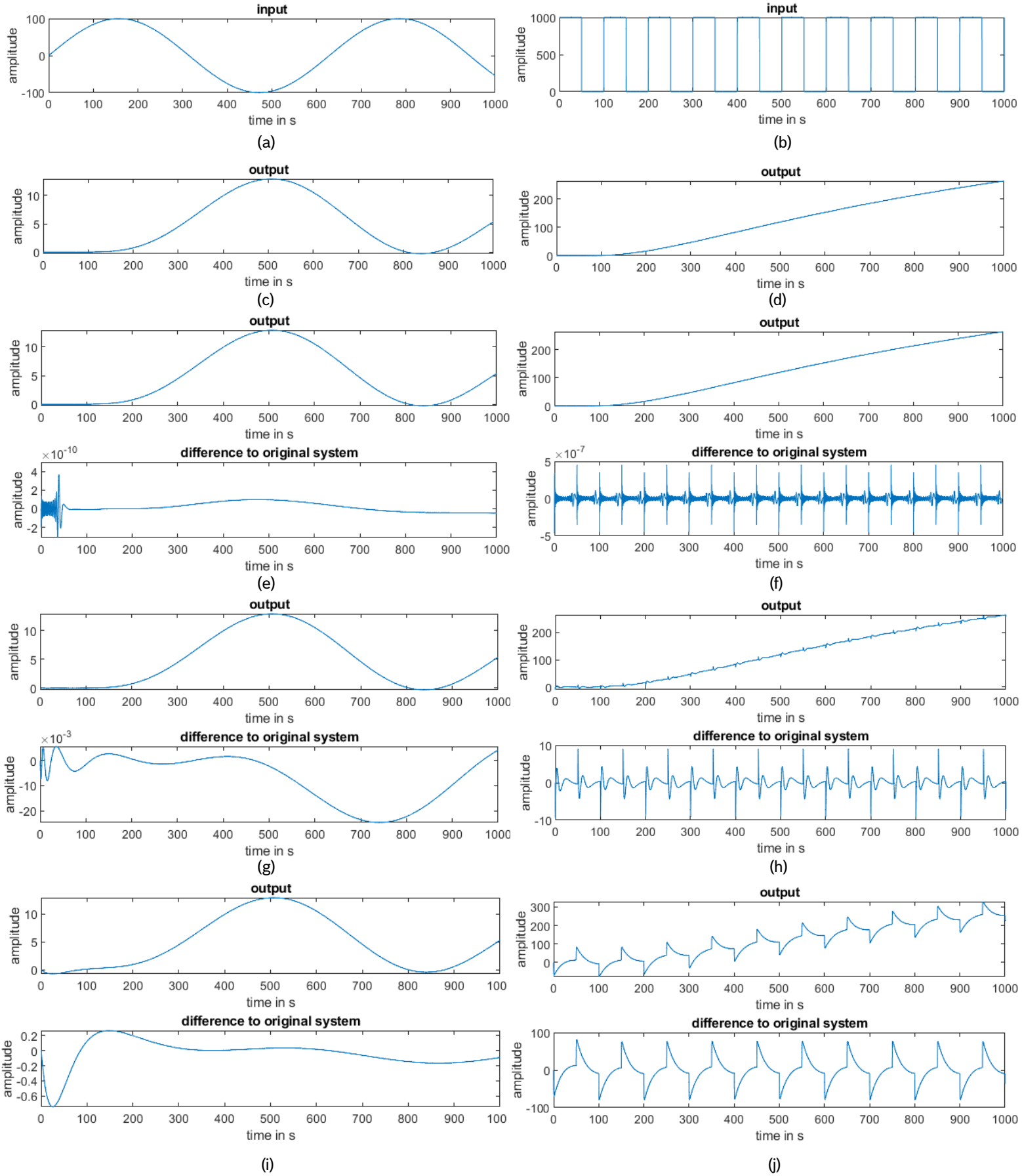


Figure 5: Forecasting results for sine and square signal, (c) and (d) output original system, (e) and (f) 40 states, (g) and (h) 8 states, (i) and (j) 4 states

5. State Variable Feedback Controller

As mentioned in chapter 3.1, one use case of the reduced order state space representation is for designing computationally efficient control strategies. This is especially interesting for real-time simulations, since the computational effort of a state space has the complexity $\boldsymbol{\Theta(n^2 r)}$ for $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times 1}$ [10] and computation effort should be held low to be real-time capable. But also for embedded systems the goal is to lower the effort.
Therefore, an example of a state variable feedback controller with pole placement based on a reduced state space is given below. The detailed theory behind state feedback control will be not explained in detail in this work, as it is not the scope. A detailed explanation can be found in [11], [12], [13].

5.1 Fundamentals

There are two goals to follow when dealing with state variable feedback systems:

- Determine the controller matrix $\boldsymbol{K}$ so that any initial states can be reduced to zero (for $t \rightarrow \infty$) [11].
- The output should converge towards the reference input $\boldsymbol{r}$. This will be achieved by calculating a filter matrix $\boldsymbol{W}$ [11].

This changes the open-loop system to the closed-loop system depicted in Figure 7.

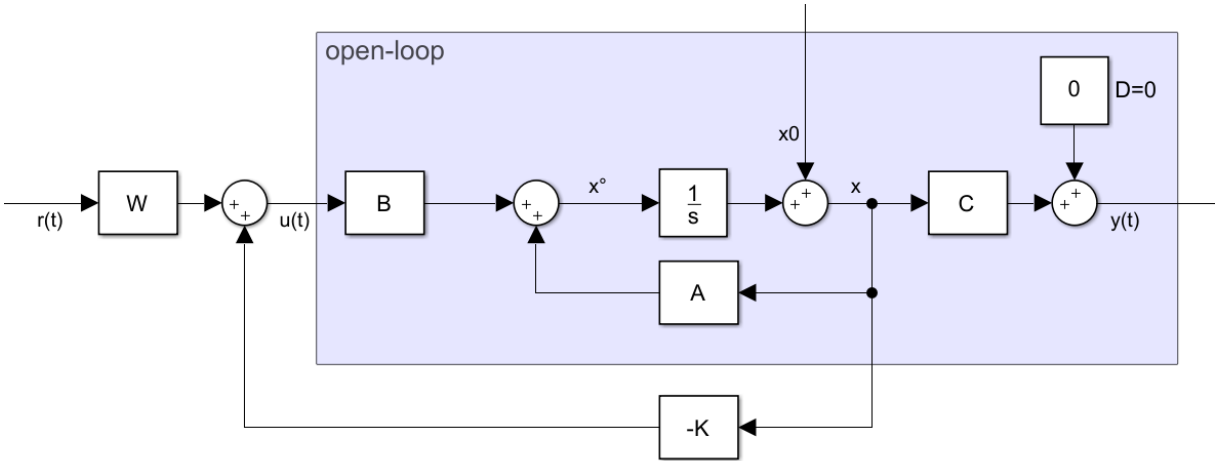


Figure 6: Closed-loop state space

The state space model of the closed control loop results therefore in

$$\begin{aligned} \dot{\boldsymbol{x}} &= (\underbrace{\boldsymbol{A}-\boldsymbol{BK}}_{\boldsymbol{A_{CL}}})\boldsymbol{x}+\boldsymbol{BWr}, \\ \boldsymbol{y} &= \boldsymbol{Cx}. \end{aligned} \tag{28}$$

The closed loop state matrix $\boldsymbol{A_{CL}}$ is depending on the controller matrix $\boldsymbol{K}$. Since the systems behavior is specified by the eigenvalues of the state matrix, the goal is to choose the controller matrix $\boldsymbol{K}$ in a way, that the system has desired eigenvalues [11]. One way to do so for Single Input Single Output (SISO) systems is via Ackermann Theorem (see [11], [13]) with controllability matrix $\boldsymbol{P_c}$ and the characteristic equation of the desired eigenvalues $\boldsymbol{P_d}$ for the matrix $\boldsymbol{A}$

$$\boldsymbol{k^T} = \boldsymbol{P_{c,n}^{-1}}\,\boldsymbol{P_d}(\boldsymbol{\lambda}\rightarrow\boldsymbol{A}). \tag{29}$$

The filter matrix $\boldsymbol{W}$ for the discrete system gets calculated by [11]

$$\boldsymbol{W} = \left[\boldsymbol{C}\left(\boldsymbol{I}-(\boldsymbol{A}-\boldsymbol{BK})\right)^{-1}\boldsymbol{B}\right]^{-1}. \tag{30}$$

5.2 Application

As an example, a circuit of 13 coupled RLC circuits is used. This means, the system has 26 states. This system shall be reduced to a system of 4 states and based on the reduced state space a controller will be designed. The sizes of R, L and C are assumed to be 1 again. For the sake of shortness, the process of reducing the system via DMDc is here not shown again (see previous chapters).

Before attempting the method, one has to prove that the system is controllable. To prove that the system is controllable, the controllability matrix must have full rank [11]. This is the case for this system.

Next, the step response of the reduced open-loop system and the location of the poles will be analyzed. As can be seen in Figure 7, the system converges to 1 in approximately 300 seconds without any oscillating or unstable behavior. Looking at the location of the poles, one can see, that this is because all poles are located on the real axis (high damping). Three of four poles are located close to the unit circle, which can be a hint that the system reacts slow.

When placing the poles, the focus is on ensuring that the system is robust, i.e. that overshoots do almost not occur and that the system becomes faster. This is attempted with a heuristic strategy, since the goal is not to design an optimal controller, but to improve the system response in general.

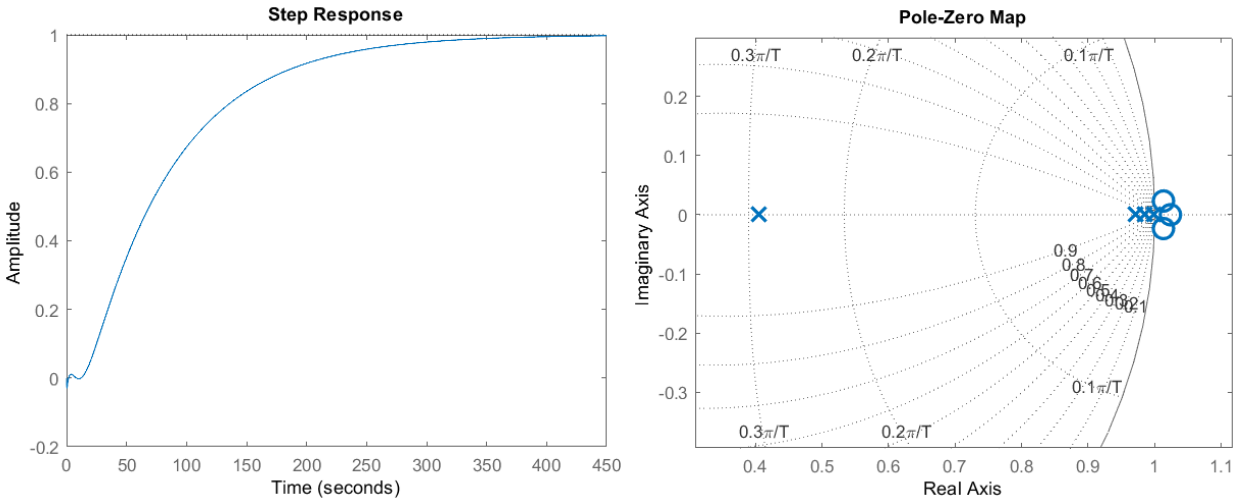


Figure 7: Step response and location of poles for open-loop

Therefore, the most right pole was shifted a bit to the left (see Table 2) as this pole has the biggest impact on the speed.

open-loop poles	0.4054	0.9986	0.9716	0.9844
closed-loop poles	0.4054	0.99	0.9716	0.9844

Table 2: Open-loop and closed-loop poles

Figure 8 shows the step response of the closed-loop system compared to the open-loop. The setpoint is reached within ~70 seconds without any overshoots, so this is a good improvement in speed while maintaining robustness.
Moving the poles further would have resulted in undesirable overshoots (not shown here).
To test the controller, an experiment is created within Simulink (Figure 9). The setpoint is applied to the original open-loop system and to the reduced open-loop system in order to detect potential deviations due to the dimensional reduction. In addition, the setpoint is then applied to the original closed-loop system and to the reduced closed-loop system whose inputs are based on the reduced closed-loop system states.

Figure 11 and Figure 12 show the results for two different setpoint signals.

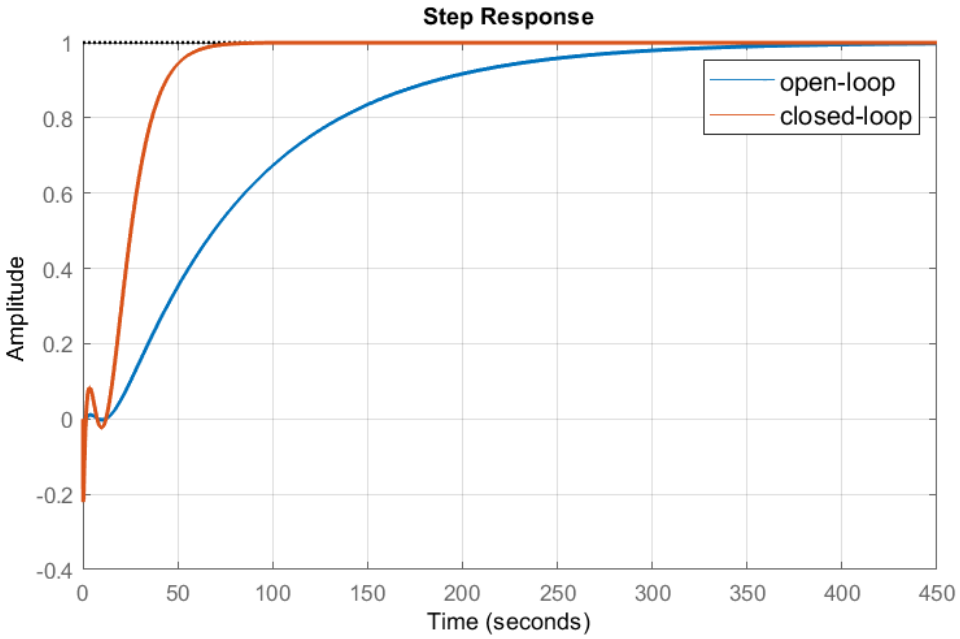


Figure 8: Closed-loop step response

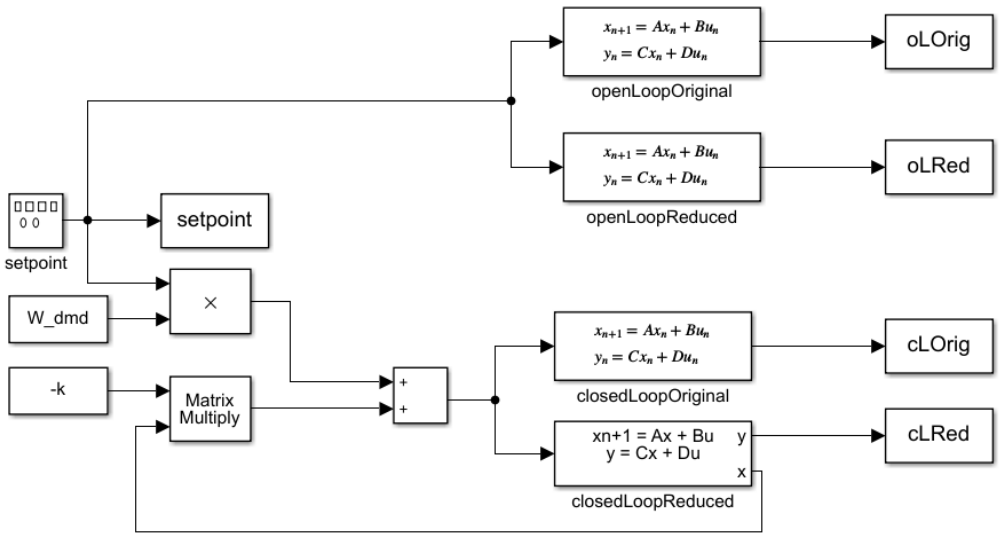


Figure 9: Simulink experiment for controller testing

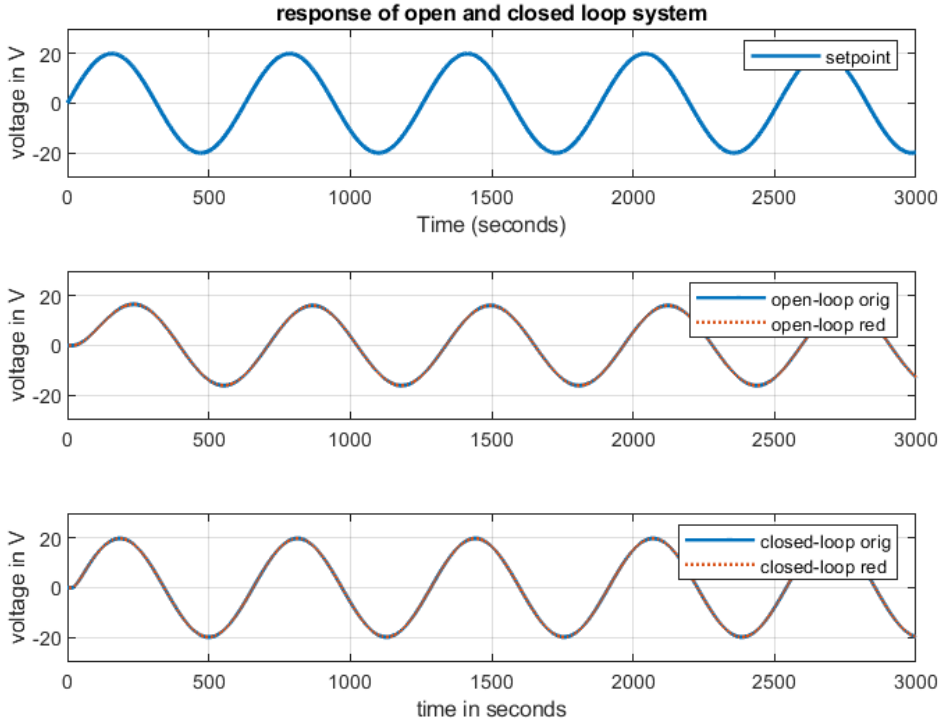


Figure 10: System response for sine wave

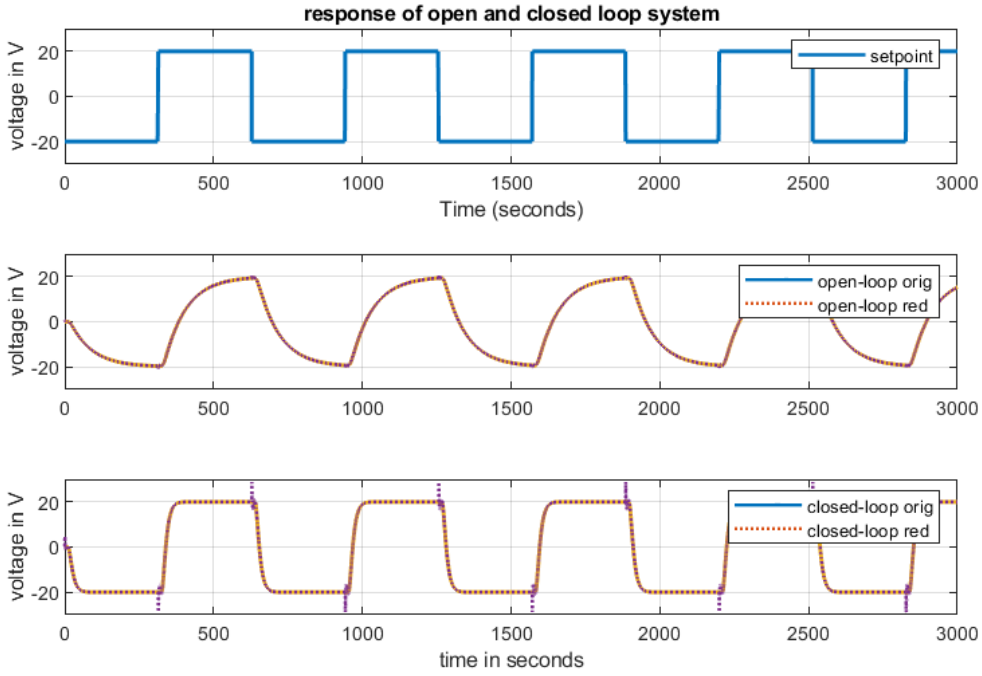


Figure 11: System response for square

In both cases it can be seen that the closed-loop controller follows the setpoint much better compared to the open-loop system. This is the proof that the controller works. Only when testing with the square signal it is noticeable that the reduced system shows an undesired deflection during the setpoint change, but this does not appear in the original system. Again, the assumption is that the deviation during the jumping setpoint change is due to an underrepresentation of the modes during the jump.

It should be noted that the simulation of the controller is based on the states of the model and not on measured states (which is ok for a simulation, but not for a real controller). If such a controller would be implemented in an ECU, either the states would have to be measured, if possible, or, if the system is too complex or sensors too expensive, an observer would have to be implemented additionally to estimate the states (system is observable).
Another theoretical possibility would be to use this controller model as a feedforward control for system stabilization and to control remaining deviations with e.g. a PID controller.

6. Conclusion

In this work, a system identification and dimensionality reduction method called Dynamic Mode Decomposition (DMD) was applied to cascaded RLC circuits. Initially, the native algorithm was used. Here it is shown how one can analyze the Dynamic Modes and their influence on the temporal behavior. Since the system under consideration is a system with a control input, an extension of DMD, namely DMDc was applied to distinguish the difference between control signal and system dynamics. The results of this can be used to predict the output of the system with a reduced state space representation. Here it was shown that the success of the dimensionality reduction depends on the type of control signal used to simulate the behavior.

Last but not least, DMDc was used to design a state controller that was designed on a reduced state space.

7. REFERENCES

[1] Institut für Systemtheorie und Regelungstechnik Universität Stuttgart, "Data-Driven System Analysis and Control," [Online]. Available: https://www.ist.uni-stuttgart.de/research/group-of-frank-allgoewer/data-driven-system-analysis-and-control/. [Accessed 27 February 2023].

[2] S. Hirche, "Data-driven Control," [Online]. Available: https://www.ce.cit.tum.de/en/itr/research/data-driven-control/. [Accessed 28 February 2023].

[3] O. Cano, A. Gres, K. Thaqi, M. Schichta and D. Schlumberger, High-order electrical RLC oscillators - PID and LQR control, Weingarten: University of Applied Sciences Weingarten, 2022, pp. 12-14.

[4] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics,* vol. 656, pp. 5-28, 2010.

[5] Z. Wu, S. L. Brunton and S. Revzen, "Challenges in Dynamic Mode Decomposition," 7 September 2021. [Online]. Available: https://arxiv.org/pdf/2109.01710.pdf. [Accessed 10 March 2023].

[6] Z. Bai, E. Kaiser, J. Proctor, N. Kutz and S. Brunton, "Dynamic Mode Decomposition for Compressive System Identification," February 2020. [Online]. Available: https://arc.aiaa.org/doi/pdf/10.2514/1.J057870. [Accessed 5 March 2023].

[7] N. Kutz, S. Brunton, B. Brunton and J. Proctor, Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, New York: Society for Industrial & Applied Mathematics,U.S., 2016.

[8] J. Proctor, S. Brunton and N. Kutz, "Dynamic Mode Decomposition with Control," 2016. [Online]. Available: https://epubs.siam.org/doi/epdf/10.1137/15M1013857. [Accessed 2 March 2023].

[9] A. Muniraju, "Analysis of Dynamic Mode Decomposition," May 2018. [Online]. Available: https://dc.uwm.edu/cgi/viewcontent.cgi?article=2884&context=etd. [Accessed 15 March 2023].

[10] F. L. Gall, "Complexity of Matrix Multiplication and Bilinear Problems," 24 August 2017. [Online]. Available: https://conferences.mpi-inf.mpg.de/adfocs-17/material/FLG_L3.pdf. [Accessed 11 March 2023].

[11] L. Berger, Advanced Control Systems Digital Control, Weingarten: Hochschule Ravensburg-Weingarten, 2022.

[12] "Control Systems/State Feedback," 8 April 2021. [Online]. Available: https://en.wikibooks.org/wiki/Control_Systems/State_Feedback. [Accessed 15 March 2023].

[13] O. Nelles, "Zustandsraum und Digitale Regelung," 2018. [Online]. Available: https://www.mb.uni-siegen.de/mrt/lehre/dr/skript_dr.pdf. [Accessed 15 March 2023].

8. APPENDIX

8.1 Script for DMD Analysis

```
clear
% main program

%% define variables
% parameters for RLC system
rlcDim=50;
R = ones(1,rlcDim)*1;
L = R;
C = R;
Ts = 0.1;
x0 = ones(1,2*rlcDim);

%dimension for DMD
dim=20;
states=2*dim;
% simulation time for continuous system
simEnd=1000;

%% calculate system
[M,N,A,B] = RLC_osc(R, L, C);
[Ac,Bc,Cc,Dc] = RLC_ss(M, N, A, B);
sys = ss(Ac,Bc,Cc,Dc);
sys_d = c2d(sys,Ts);

%% plot of unforced system for continuous system
[y,t,x] = initial(sys, x0, simEnd); %plots the unforced
system response
figure(1);
subplot(3,1,1);
plot(t,x);
legend('x1', 'x2', 'x3', 'x4', 'x5', 'x6');
title('plot of system states for continuous system');
xlabel('time in s');
ylabel('amplitude');
xlim([0 1000]);

subplot(3,1,2);
plot(t,y);
```

```
title('plot of output for continuous system');
xlabel('time in s');
ylabel('amplitude');
xlim([0 1000]);
ylim([1 2.5]);

subplot(3,1,3);
pzmap(sys);
title('poles and zeros of continuous system');

%% plot of unforced system for discrete system
[y,t,x] = initial(sys_d, x0, simEnd); %plots the unforced
system response
y_rec=y;
figure(2);
subplot(3,1,1);
plot(t,x);
title('plot of system states for discrete system');
xlabel('time in s');
ylabel('amplitude');
xlim([0 1000]);

subplot(3,1,2);
plot(t,y);
title('plot of output for discrete system');
xlabel('time in s');
ylabel('amplitude');
xlim([0 1000]);
ylim([1 2.5]);

subplot(3,1,3);
pzmap(sys_d);
title('poles and zeros of discrete system');
xlim([0.5 1.5]);

%% DMD
% create X1 and X2
X1 = x(1:length(t)-1,:);
X1 = X1';
X2 = x(2:length(t),:);
X2 = X2';
x0 = ones(1,states);
% calculate reduced A matrix (DMD)
[A_dmd, dynModes, U] = Dmd(X1,X2, states);
```

```
sys_dmd = ss(A_dmd,[],[],[],Ts);

[y,t,x] = initial(sys_dmd, x0, 1000); %plots the unforced
system response

figure(3);
subplot(3,1,1);
plot(t,x);
title('plot of system states for dmd system');
xlabel('time in s');
ylabel('amplitude');
xlim([0 1000]);

% calculate C matrix for reduced dmd system
C_dmd = y_rec' * pinv(x');
% calculate output with C matrix
for i = 1:length(x)
    y(i) = C_dmd * x(i,:)';
end

subplot(3,1,2);
plot(t,y);
title('plot of output for dmd system');
xlabel('time in s');
ylabel('amplitude');
xlim([0 1000]);
ylim([1 2.5]);

subplot(3,1,3);
eigW = eig(sys_dmd.A);
plot(real(eigW),imag(eigW), 'x');
title('poles of dmd system');
ylabel('Imaginary Axis');
xlabel('Real Axis');
viscircles([0 0], 1, 'color', [.5 .5 .5], 'LineStyle',
':', 'LineWidth', 1);
ylim([-1 1]);
xlim([0.5 1.5]);

figure(4);

% calculate behavior of dynamic modes

for i=1:length(eigW)
   cont = log(complex(eigW(i)))/Ts;
```

```
    modeBhvor = exp(cont*t);
    plot(t,modeBhvor);
    hold on;
end

title('Dynamic Mode Evolution');
xlabel('time in s');
ylabel('Amplitude');

%% helper functions

function [M,N,A,B] = RLC_osc(R, L, C)

    % get size of array(s)
    order = length(R);

    % create base matrices
    M=zeros(order);
    N=zeros(order);
    A=-1*eye(order);
    B=zeros(order,1);

    for row = 1:order
        for column = row:order
            %fill matrices with corresponding value pairs
            M(row, column) = L(row)*C(column);
            N(row, column) = R(row)*C(column);

            if row > 1
          %fill matrix A with ones under the main diagonal
                A(row, row-1) = 1;
            end
        end
    end

    B(1) = 1;
end

function [Ac,Bc,Cc,Dc] = RLC_ss(M, N, A, B)
    %get order of resulting matrices
    order = length(B);

    %build matrices
    S = [eye(order), zeros(order);
        zeros(order), M];
```

```
    T = [zeros(order), eye(order);
        A, -N];

    %calculate Ac and Bc

    Ac=S\T;
    Bc=S\[zeros(order,1); B];
    Cc=zeros(1,order);
    Cc(1,order)=1;
    Cc=[Cc zeros(1, order)];
    Dc=0;
end

function [A_dmd, psi, U] = Dmd(X1, X2, dim)
    % SVD for X1
    [U,S,V] = svds(X1,dim);

    % koopman matrix (reduced A)
    A_dmd = U'*X2*V*inv(S);

    % calculate dynamic modes
    [eigV, eigW]=eig(A_dmd);
    psi = X2*V*inv(S)*eigV;
end
```

8.2 Script for DMDc Analysis

```
clear
% main program

%% define variables
% parameters for RLC system
rlcDim=50;
nrStates= rlcDim*2;
R = ones(1,rlcDim)*1;
L = ones(1,rlcDim)*1;
C = ones(1,rlcDim)*1;
Ts = 0.1;

%dimension for DMD
dmdDim=20;
dmdStates=dmdDim*2;
% simulation time for continuous system
simEnd=1000;
%% calculate system
[M,N,A,B] = RLC_osc(R, L, C);
[Ac,Bc,Cc,Dc] = RLC_ss(M, N, A, B);
sys = ss(Ac,Bc,Cc,Dc);
sys_d = c2d(sys,Ts);
%% plot for discrete system

t = 0:Ts:simEnd;
u = 100*sin(0.01*t); % signal for DMD measurement data
u2 = u;
%u2=1000*0.5*(square(2*pi*0.01*t)+1); % test signal

[y,t,x] = lsim(sys_d, u, t);
[y2,t2,x2] = lsim(sys_d, u2, t);
figure(2);
subplot(3,1,1);
plot(t2,u2);
title('input');
subplot(3,1,2);
plot(t2,x2);
title('system states');
subplot(3,1,3);
plot(t,y2);
title('output');
xlabel('time in s');
ylabel('amplitude');
```

```
%% DMDc
% create X1 and X2 and Y
X1 = x(1:length(t)/2-1,:);
X1 = X1';
X2 = x(2:length(t)/2,:);
X2 = X2';
Y = u(1:length(t)/2-1);
% calculate reduced A matrix (DMD)
[A_dmd, B_dmd, U_hat] = dmdwc(X1,X2,Y,dmdStates,
nrStates);

sys_dmd = ss(A_dmd,B_dmd,[],[],Ts);

% simulate system with dmd matrices
[y,t,x] = lsim(sys_dmd, u2, t);

% reconstruct original states out of dmd results to
compare accuracy
x_rec = [];
for i = 1:length(x)
    x_rec(i,:) = x(i,:) * U_hat';
    y_rec(i) = Cc * x_rec(i,:)';
end

% calculate C matrix for reduced dmd system
C_dmd = y_rec * pinv(x');

% calculate output with C matrix
for i = 1:length(x)
    y(i) = C_dmd * x(i,:)';
end

figure(3);
subplot(3,1,1);
plot(t2,u2);
title('input');
subplot(3,1,2);
plot(t,x);
title('system states');
subplot(3,1,3);
plot(t,y);
title('output');
xlabel('time in s');
ylabel('amplitude');
```

```
%% helper functions

function [A_dmd, B_dmd, U_hat] = dmdwc(X1, X2, Y, dim,
nrStates)
    % create Omega
    OM = [X1;Y];

    % SVD for input space omega, truncation value p must
be larger than
    % that of X -> more returned singular values
    [U_tilde,S_tilde,V_tilde] = svds(OM, min(dim+1,
nrStates));

    % U_tilde1 corresponds to X1 (dim=nrStates), U_tilde2
to Y (dim=1)
    U_tilde1 = U_tilde(1:nrStates,:);
    U_tilde2 = U_tilde(nrStates+1,:);

    % SVD for output space X2
    [U_hat, S_hat, V_hat] = svds(X2, dim);

    % compute A_dmd and B_dmd
    A_dmd =
U_hat'*X2*V_tilde*pinv(S_tilde)*U_tilde1'*U_hat;
    B_dmd = U_hat'*X2*V_tilde*pinv(S_tilde)*U_tilde2';
end
```

8.3 Script for Controller Design

```
clear
% main program

%% define variables
% parameters for RLC system
rlcDim=13;
nrStates= rlcDim*2;
R = ones(1,rlcDim)*1;
L = ones(1,rlcDim)*1;
C = ones(1,rlcDim)*1;
Ts = 0.1;

%dimension for DMD
dmdDim=2;
dmdStates=dmdDim*2;
% simulation time for continuous system
simEnd=1000;
%% calculate system
[M,N,A,B] = RLC_osc(R, L, C);
[Ac,Bc,Cc,Dc] = RLC_ss(M, N, A, B);
sys = ss(Ac,Bc,Cc,Dc);
sys_d = c2d(sys,Ts);
Ad = sys_d.A;
Bd = sys_d.B;
Cd = sys_d.C;
Dd = sys_d.D;

t = 0:Ts:simEnd;
u = 300*sin(0.01*t);

[y,t,x] = lsim(sys_d, u, t);

%% DMDc
% create X1 and X2 and Y
X1 = x(1:length(t)/2-1,:);
X1 = X1';
X2 = x(2:length(t)/2,:);
X2 = X2';
Y = u(1:length(t)/2-1);
% calculate reduced A matrix (DMD)
[A_dmd, B_dmd, U_hat] = dmdwc(X1,X2,Y,dmdStates,
nrStates);
```

```
sys_dmd = ss(A_dmd,B_dmd,[],[],Ts);

% simulate system with dmd matrices
[y,t,x] = lsim(sys_dmd, u, t);

% reconstruct original states out of dmd results to
compare accuracy
x_rec = [];
for i = 1:length(x)
    x_rec(i,:) = x(i,:) * U_hat';
    y_rec(i) = Cc * x_rec(i,:)';
end

% calculate C matrix for reduced dmd system
C_dmd = y_rec * pinv(x');

% calculate output with C matrix
for i = 1:length(x)
    y(i) = C_dmd * x(i,:)';
end

% setup whole dmd system
sys_dmd = ss(A_dmd,B_dmd,C_dmd,[],Ts);

ctrlblty = rank(ctrb(sys_dmd)); % Prüfe dessen
Steuerbarkeit
EW = [0.4054 0.99 0.9716 0.9844]; % Gewünschte Eigenwerte
k = place(A_dmd,B_dmd,EW); % Reglerentwurf mittels
Polvorgabe
W_dmd=inv(C_dmd*inv(eye(4)- (A_dmd-B_dmd*k))*B_dmd);

sys_dmdCl = ss(A_dmd - B_dmd*k,B_dmd*W_dmd,C_dmd,[],Ts);
figure(2);
step(sys_dmd);
hold on;
step(sys_dmdCl);
grid on;
legend('open-loop','closed-loop');

figure(3);
subplot(3,1,1);
plot(t,u);
title('input');
subplot(3,1,2);
plot(t,x);
```

```
title('system states');
subplot(3,1,3);
plot(t,y);
title('output');
xlabel('time in s');
ylabel('amplitude');

sim("dmdc_controllersim.slx");

figure(4);
subplot(3,1,1);
plot(setpoint,'LineWidth',1.5);
grid on;
ylim([-30 30]);
ylabel('voltage in V');
title('response of open and closed loop system');
legend('setpoint');
subplot(3,1,2);
plot(oLOrig,'LineWidth',1.5);
hold on;
plot(oLRed,':','LineWidth',1.5);
grid on;
ylim([-30 30]);
ylabel('voltage in V');
legend('open-loop orig', 'open-loop red');
hold on;
subplot(3,1,3);
plot(cLOrig,'LineWidth',1.5);
hold on;
plot(cLRed,':', 'LineWidth',1.5);
xlabel('time in seconds');
grid on;
ylim([-30 30]);
ylabel('voltage in V');
legend('closed-loop orig', 'closed-loop red');
```