ETH ZÜRICH

# Computing spectral submanifolds using multi-index notation

*Author*
Thomas THURNHER

*Supervisor*
Dr. Shobhit JAIN

April 11, 2020

## Abstract

This thesis develops a method to solve the invariance equation using multi-index notation. Initially after introducing the mathematical tools to do so, the theoretical calculations are performed. In a second step those calculations are implemented in Matlab. The coefficients for expanding the parametrisation **S** of the spectral submanifold as a Taylor series and the coefficients characterising the reduced dynamics **R** are calculated in the developed implementation. This is done for three examples and the results are compared to the results given by the implementation using tensor-notation. Furthermore the performance of the two different implementations is compared. We find that the multi-index version leads to lower memory requirements and speed ups with respect to the tensor-version. It also leads to the correct force response curves, without diagonalising the given system.

# Contents

## 1 Introduction

Recent findings by Haller and Ponsioen proof the existence and uniqueness of spectral submanifolds for systems that fulfill a set of conditions (2016)[1]. We consider such a system. Let $\mathbf{S} : \mathbb{C}^l \mapsto \mathbb{C}^{2n}$ and $\mathbf{R} : \mathbb{C}^l \mapsto \mathbb{C}^l$ be functions, $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{2n\times 2n}$ and let $\mathbf{p} \in \mathbb{C}^l$ be the euclidian parametrised coordinates on the spectral submanifold (SSM). Here $\mathbf{S}$ maps the parametrisation of the SSM to the full phase space, $\mathbf{R}$ represents the reduced dynamics on the SSM and $\mathbf{A}$, $\mathbf{B}$ are matrices determined by the system that is to be analysed. The map $\mathbf{F}$ denotes the external non-linear force acting in phase space and is assumed to be autonomous. The dimension of the phase space of the system is $N = 2n$ and the dimension of the SSM studied is $l$. We wish to solve the invariance equation

$$\mathbf{B}D\mathbf{S}\mathbf{R} = \mathbf{A}\mathbf{S} + \mathbf{F} \circ \mathbf{S} \tag{1.1}$$

to find the coefficients characterising the taylor expansion of the map $\mathbf{S}$. Furthermore the aim is to do this without diagonalising the system. A solution that calculates those coefficients at order $k$ for a diagonalised system is given by Ponsioen, Pedergnana and Haller (2018, p. 274)[2]

$$\begin{aligned}\mathbf{\Lambda}\mathbf{S}_k - \mathbf{S}_k \sum_{|\mathbf{s}|=1} \mathbf{\Lambda}_{\mathcal{E}}^{s_1} \otimes \cdots \otimes \mathbf{\Lambda}_{\mathcal{E}}^{s_k} =& \mathbf{S}_1\mathbf{R}_k + \sum_{u=2}^{k-1} \mathbf{S}_u \sum_{|\mathbf{g}|=1} \mathbf{R}_{k+1-m}^{g_1} \otimes \cdots \otimes \mathbf{R}_{k+1-m}^{g_m} \\ &- \mathbf{F}_k \mathbf{S}_1^{\otimes k} - \sum_{m=2}^{k-1} \mathbf{F}_m \sum_{|\mathbf{r}|=1} \mathbf{S}_{r_1} \otimes \cdots \otimes \mathbf{W}_{r_m}\end{aligned} \tag{1.2}$$

In this equation $\mathbf{s}, \mathbf{g}, \mathbf{r}$ are multi-indices and $\mathbf{\Lambda}$ is obtained by diagonalising $\mathbf{A}$ through a change of coordinates. The system has been set up such that $\mathbf{B} = \mathbf{I}$. $\mathbf{\Lambda}_{\mathcal{E}}$ contains the $l$ master-eigenvalues on its diagonal. $\mathbf{S}_u$ is the tensor that contains the coefficients that describe the Taylor-expansion of $\mathbf{S}$ at order $u$ and likewise for the reduced dynamics $\mathbf{R}$. The tensor $\mathbf{F}_m$ contains the order $m$ nonlinearity-coefficients.
It is desirable to solve Eq. (1.1) using multi-index notation in order to avoid redundancies of coefficients and without diagonalising the system in order to save computational resources. In the tensor-format coefficients corresponding to the same multivariate monomials are stored for all combinations. E.g. for some polynomial $p(x, y)$ the coefficient corresponding to the monomial $xy$ is stored two times, for both ways of arranging the variables $x$ and $y$. This is not the case with coefficients in multi-index format, where the coefficient to every multivariate monomial is stored only once.

# 2 Mathematical foundations

## 2.1 Multi-index calculus and notation

A multi-index vector $\mathbf{u}$ of order $u \in \mathbb{N}$ and size $l \in \mathbb{N}$ is a vector in $\mathbb{N}^l$ where $||\mathbf{u}||_1 = u$ in the canonical one norm, which we will reference to as $|\mathbf{u}|$, that sums all elements of a vector $||\mathbf{u}||_1 = \sum_{i=0}^{l} (\mathbf{u})_i$.
We will use a hat to describe multi-index vectors of order 1, e.g. $\hat{\mathbf{e}}$. More specifically, the vector of order 1 that contains its only non zero entry in row $j$ will be called $\hat{\mathbf{e}}_j$.
The addition and subtraction of multi-indices, as well as (in-) equalities between them are defined element-wise. Taking the power of a vector $\mathbf{p}$ of the same size $l$ as a multi-index $\mathbf{u}$ is defined as the product

$$\mathbf{p}^{\mathbf{u}} = p_1^{u_1} \cdots p_l^{u_l} \tag{2.1}$$

where the $p_i$-s and $u_i$-s are the elements of the vector and the multi-index respectively, i.e. $u_i = (\mathbf{u})_i$.
The spatial derivative associated to a multi-index $\mathbf{k}$ is given by $\boldsymbol{D}^{\mathbf{k}} := D_1^{k_1} \cdots D_l^{k_l}$, for a differential operator $D$. Furthermore upon applying this to the before mentioned term $\mathbf{p}^{\mathbf{u}}$ the result is:

$$\boldsymbol{D}^{\boldsymbol{k}} \frac{\boldsymbol{p}^{\boldsymbol{u}}}{\boldsymbol{u}!} = \frac{\boldsymbol{p}^{\boldsymbol{u}-\boldsymbol{k}}}{(\boldsymbol{u}-\boldsymbol{k})!} \tag{2.2}$$

making use of the definition of the factorial $\mathbf{u}! = u_1! \cdots u_l!$.

## 2.2 Time-derivative of a multi-index polynomial

Take $\mathbf{p} \in \mathbb{C}^l$ and $\mathbf{u} \in \mathbb{N}^l$ for some natural number $l$. Then the time derivative is calculated as follows

$$\frac{d\mathbf{p}^{\mathbf{u}}}{dt} = \sum_{\substack{\mathbf{k} \\ |\mathbf{k}|=1}} \boldsymbol{D}^{\mathbf{k}} \mathbf{p}^{\mathbf{u}} \dot{p}_{j(\mathbf{k})} = \sum_{j=1}^{l} \boldsymbol{D}^{\hat{\mathbf{e}}_j} \mathbf{p}^{\mathbf{u}} \dot{p}_j = \sum_{j=1}^{l} u_j \mathbf{p}^{\mathbf{u}-\hat{\mathbf{e}}_j} \dot{p}_j \tag{2.3}$$

where $p_j/u_j$ is the $j^{th}$element of $\mathbf{p}/\mathbf{u}$ respectively and at the same time the position where $\mathbf{k}$ is nonzero. Here Eq. (2.2) and the chain rule has been used.

## 2.3 Multipication of power series

Suppose we want to take a scalar power $s$ of an arbitrary power Series $\sum_{\mathbf{u}} S_{\mathbf{u}} \mathbf{p}^{\mathbf{u}}$. A convenient method, which offers a recursive definition of coefficients to the multivariate monomials of the multiplicated power series is given by the concept of the radial derivative, which is done in the book of Haro et. al (2016). [3] Ponsioen (2019)[4] uses this to express the coefficients $H_{s,\mathbf{h}}$ in the expansion

$$\Big(\sum_{\mathbf{u}} S_{\mathbf{u}} \mathbf{p}^{\mathbf{u}}\Big)^s = \sum_{\mathbf{h}} H_{s,\mathbf{h}} \mathbf{p}^{\mathbf{h}} \tag{2.4}$$

using the recursion defined by

$$H_{s,\mathbf{h}} = \frac{s}{h_j} \sum_{\mathbf{u} \leq \mathbf{h}} u_j S_{\mathbf{u}} H_{s-1,\mathbf{h}-\mathbf{u}} \tag{2.5}$$

where $j$ is the index of the smallest nonzero element in $\mathbf{h}$, $h_j = (\mathbf{h})_j$. To simplify the calculation of the recursion it is useful to look at some special cases, under assumption that the coefficients $S_{i,\mathbf{u}}$ are the coefficients of the SSM expansion of row $i$. The SSM parametrisation does not contain constant terms which means that all composition-coefficients are zero for $|\mathbf{u}| = 0$. The properties of $H_{i,s,\mathbf{h}}$ are then

- $H_{i,1,\mathbf{0}} = 0$
- $H_{i,1,\mathbf{h}} = S_{\mathbf{h}}$
- $H_{i,0,\mathbf{h}} = 0$ for all $\mathbf{h} \neq \mathbf{0}$,
- $H_{i,0,0} = 1$

## 3 Multi-index treatment

As a first step the functions $\mathbf{S}$ and $\mathbf{R}$ are written as Taylor expansions in every row. Choosing unit vectors $\hat{\mathbf{e}}_i$ as a basis, these expansions are given by

$$\mathbf{S}(\mathbf{p}) = \sum_{i=1}^{2n} \hat{\mathbf{e}}_i \sum_{\mathbf{u}} S_{i,\mathbf{u}} \mathbf{p}^{\mathbf{u}}, \quad \mathbf{R}(\mathbf{p}) = \sum_{i=1}^{l} \hat{\mathbf{e}}_i \sum_{\mathbf{g}} R_{i,\mathbf{g}} \mathbf{p}^{\mathbf{g}} \tag{3.1}$$

The elements $S_{i,\mathbf{u}}$ are the scalar coefficients of the parametrisation of the SSM belonging to the $i^{th}$row and the multi-index $\mathbf{u}$ and analogously for $R_{i,\mathbf{g}}$. Here the multi-indices are of size $l$ which is the dimension of the SSM. The $\hat{\mathbf{e}}_i$ in the expression for the function $\mathbf{S}$ are the standard-basis in the phase space, while the $\hat{\mathbf{e}}_i$ in the expansion of the reduced dynamics are the basis in which the parametrised coordinates $\mathbf{p}$ are expressed, and thus $l$-dimensional unit vectors.

### 3.1 Left hand side

Initially the term $D\mathbf{SR}$ in Eq. (1.1) is derived in multi-index notation by considering single rows $i$. Let $\mathbf{x} = \mathbf{S}(\mathbf{p}) \in \mathbb{C}^{2n}$ and $\mathbf{p} \in \mathbb{C}^{l}$ . Considering the time derivative of an individual row $i$ of the vector $\mathbf{x}$, which we will call $x_i$, one therefore gets

$$\dot{x}_i = \frac{dx_i}{dt} = \frac{d\big(\mathbf{S}(\mathbf{p})\big)_i}{dt} \tag{3.2}$$

We can now plug in the $i^{th}$ row of the polynomial expansion of $\mathbf{S}(\mathbf{p})$ given in Eq. (3.1), which then leads to

$$\dot{x}_i = \frac{d\left(\sum_{\mathbf{u}} S_{i,\mathbf{u}}\mathbf{p}^{\mathbf{u}}\right)}{dt} = \sum_{\mathbf{u}} S_{i,\mathbf{u}} \frac{d\mathbf{p}^{\mathbf{u}}}{dt} \tag{3.3}$$

In order to calculate the time derivative in Eq. (3.3) a single multi-index $\tilde{\mathbf{u}} = [\tilde{u}_1, ..., \tilde{u}_l]^T$ is considered. The time derivative of the term $\mathbf{p}^{\tilde{\mathbf{u}}}$ is then calculated using Eq. (2.3) as

$$\frac{d\mathbf{p}^{\tilde{\mathbf{u}}}}{dt} = \sum_{\substack{j=1\\ \tilde{u}_j>0}}^{l} \tilde{u}_j \mathbf{p}^{\tilde{\mathbf{u}}-\hat{\mathbf{e}}_j} \dot{p}_j = \sum_{\substack{j=1\\ \tilde{u}_j>0}}^{l} \tilde{u}_j \mathbf{p}^{\tilde{\mathbf{u}}-\hat{\mathbf{e}}_j} \big(\mathbf{R}(\mathbf{p})\big)_j \tag{3.4}$$

where $p_j$ is the $j^{th}$element of $\mathbf{p}$, and $m_j > 0$ since the expansion only contains positive power terms. This is possible because the reduced dynamics fulfill $\dot{\mathbf{p}} = \mathbf{R}(\mathbf{p})$, and in particular this also holds for every single row of the reduced dynamics. Now we can use the row-wise expansion of $\mathbf{R}(\mathbf{p})$ to obtain the result

$$\frac{d\mathbf{p}^{\tilde{\mathbf{u}}}}{dt} = \sum_{\substack{j=1\\ \tilde{u}_j>0}}^{l} \tilde{u}_j \mathbf{p}^{\tilde{\mathbf{u}}-\hat{\mathbf{e}}_j} \Big(\sum_{\mathbf{g}} R_{j,\mathbf{g}} \mathbf{p}^{\mathbf{g}}\Big) = \sum_{\substack{j=1\\ \tilde{u}_j>0}}^{l} \sum_{\mathbf{g}} \tilde{u}_j R_{j,\mathbf{g}} \mathbf{p}^{\tilde{\mathbf{u}}-\hat{\mathbf{e}}_j} \mathbf{p}^{\mathbf{g}} \tag{3.5}$$

Using Eq. (2.1) it is clear that the powers of $\mathbf{p}$ can be grouped together which yields:

$$\frac{d\mathbf{p}^{\tilde{\mathbf{u}}}}{dt} = \sum_{\substack{j=1\\ \tilde{u}_j>0}}^{l} \sum_{\mathbf{g}} \tilde{u}_j R_{j,\mathbf{g}} \mathbf{p}^{\tilde{\mathbf{u}}-\hat{\mathbf{e}}_j+\mathbf{g}} \tag{3.6}$$

This result can now be plugged into Eq. (3.3) and we obtain

$$\dot{x}_i = \sum_{\mathbf{u}} S_{i,\mathbf{u}} \sum_{\substack{j=1\\ u_j>0}}^{l} \sum_{\mathbf{g}} u_j R_{j,\mathbf{g}} \mathbf{p}^{\mathbf{u}-\hat{\mathbf{e}}_j+\mathbf{g}} \tag{3.7}$$

Since we wish to construct the parametrisation of the SSM iteratively in increasing orders $u$ of the multi-index $\mathbf{u}$ it is now necessary to extract specific orders from Eq. (3.7).

### 3.1.1 Separating terms by order

Suppose we want to extract terms of order $k$. We call the number of multi-indices at order $k$ with size $l$ $z_k$. Now we choose a specific multi-index $\mathbf{k}_f$ for $1 \leq f \leq z_k$, $\mathbf{k}_f = [k_{f,1}, ..., k_{f,l}]^T$ such that

$$\sum_{i=1}^{l} k_{f,i} = k \tag{3.8}$$

The multi-index $\mathbf{k}_f$ has been picked out of $\mathbf{K} = (\mathbf{k}_1, ..., \mathbf{k}_{z_k})$ , where the $\mathbf{k}_i \in \mathbb{N}^l$ are all the multi-indices of order $k$ in reverse-lexicographical order. We wish to extract the coefficients of Eq. (3.7) which only contain terms that correspond to the exponent $\mathbf{k}_f$ of $\mathbf{p}$, which we will call $\dot{x}_{i,\mathbf{k}_f}$. This means that $\mathbf{u} - \hat{\mathbf{e}}_j + \mathbf{g} \overset{!}{=} \mathbf{k}_f$ for every individual term that contributes to $\dot{x}_{i,\mathbf{k}_f}$. This couples the two multi-indices $\mathbf{u}$ and $\mathbf{g}$ such that the sum over either of the two indices is now dependent on the other. We can then rewrite Eq. (3.7) as

$$\dot{x}_{i,\mathbf{k}_f} = \sum_{\substack{j=1 \\ u_j>0}}^{l} \sum_{\substack{\mathbf{g},\mathbf{u} \\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f}} u_j S_{i,\mathbf{u}} R_{j,\mathbf{g}} \tag{3.9}$$

In order to eventually write the whole invariance equation in Eq. (1.1) as linear equation that determines the values $S_{i,\mathbf{k}_f}$ we now separate terms of different multi-index orders of the multi-index $\mathbf{u}$ in Eq. (3.9). It is important to keep in mind, that only positive multi-indices are used for the expansion.
First we consider the terms which contain the numbers $S_{i,\mathbf{u}}$ where the order of the multi-index $\mathbf{u}$ is 1, i.e. we take all the terms in Eq. (3.9) where the sum over $\mathbf{u}$ only includes the multi-indices $\hat{\mathbf{e}}_a$, $a = 1, ...l$ . This part of the equation is then:

$$\begin{aligned} \sum_{\substack{j=1 \\ u_j>0}}^{l} \sum_{\substack{\mathbf{g},\mathbf{u} \\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f \\ |\mathbf{u}|=1}} m_j S_{i,\mathbf{u}} R_{j,\mathbf{g}} &= \sum_{j,a=1}^{l} \sum_{\substack{\mathbf{g} \\ \mathbf{g}+\hat{\mathbf{e}}_a-\hat{\mathbf{e}}_j=\mathbf{k}_f}} \delta_{ja} S_{i,\hat{\mathbf{e}}_a} R_{j,\mathbf{g}} \\ &= \sum_{j,a=1}^{l} \delta_{ja} S_{i,\hat{\mathbf{e}}_a} R_{j,(\mathbf{k}_f-\hat{\mathbf{e}}_a+\hat{\mathbf{e}}_j)} \end{aligned} \tag{3.10}$$

The second equality holds, as by taking a combination of multi-indices $\hat{\mathbf{e}}_j, \hat{\mathbf{e}}_a$ and $\mathbf{k}_f$ the multi-index $\mathbf{g}$ is fully determined as $\mathbf{g} = \mathbf{k}_f - \hat{\mathbf{e}}_a + \hat{\mathbf{e}}_j$.

The next group of terms considered takes into account only contributions of the highest order SSM-coefficients, i.e. where $|\mathbf{u}| = k$. Therefore we inspect the following coefficients of Eq. (3.9)

$$\begin{aligned} \sum_{\substack{j=1 \\ u_j>0}}^{l} \sum_{\substack{\mathbf{g},\mathbf{u} \\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f \\ |\mathbf{u}|=k}} u_j S_{i,\mathbf{u}} R_{j,\mathbf{g}} &= \sum_{\substack{j=1 \\ (\mathbf{k}_a)_j>0}}^{l} \sum_{a=1}^{z_k} \sum_{\substack{\mathbf{g} \\ \mathbf{g}+\mathbf{k}_a-\hat{\mathbf{e}}_j=\mathbf{k}_f}} (\mathbf{k}_a)_j S_{i,\mathbf{k}_a} R_{j,\mathbf{g}} \\ &= \sum_{\substack{j=1 \\ (\mathbf{k}_a)_j>0}}^{l} \sum_{a=1}^{z_k} (\mathbf{k}_a)_j S_{i,\mathbf{k}_a} R_{j,(\mathbf{k}_f-\mathbf{k}_a+\hat{\mathbf{e}}_j)} \end{aligned} \tag{3.11}$$

where all multi-indices $\mathbf{k}_a$ are of order $k$. The second equality results from the fact, that the exponent of $\mathbf{p}$ in Eq. (3.9) has to be equal to $\mathbf{k}_f$. This means that by summation over all multi-indices of order $k$, $\mathbf{g}$ is completely determined by a combination $(\hat{\mathbf{e}}_j, \mathbf{k}_f, \mathbf{k}_a)$. Therefore $\mathbf{g} = \mathbf{k}_f - \mathbf{k}_a + \hat{\mathbf{e}}_j$. Again, one has to keep in mind that this equality is only valid for $(\mathbf{g})_i \geq 0,\ \forall i$. Solutions not fulfilling this condition, are not part of the polynomial expansion since they correspond to coefficients $R_{j,\mathbf{g}} = 0$.
Eventually the last part not treated yet, contains all the mixed terms, meaning that it contains the $S_{i,\mathbf{u}}$ where $|\mathbf{u}| \neq 1, |\mathbf{u}| \neq k$. Putting the terms in Eq. (3.11) and Eq. (3.10) together with these mixed terms we can write Eq. (3.9) as

$$\dot{x}_{i,\mathbf{k}_f} = \sum_{\substack{j=1 \\ u_j,(\mathbf{k}_a)_j>0}}^{l} \sum_{\substack{\mathbf{g},\mathbf{u} \\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f}} u_j S_{i,\mathbf{u}} R_{j,\mathbf{g}} \tag{3.12}$$

$$\begin{aligned} &= \sum_{\substack{j=1 \\ u_j>0}}^{l} \Bigg( \sum_{a=1}^{l} \delta_{ja} S_{i,\hat{\mathbf{e}}_a} R_{j,(\mathbf{k}_f - \hat{\mathbf{e}}_a + \hat{\mathbf{e}}_j)} \\ &\quad + \sum_{a=1}^{z_k} (\mathbf{k}_a)_j S_{i,\mathbf{k}_a} R_{j,(\mathbf{k}_f - \mathbf{k}_a + \hat{\mathbf{e}}_j)} \\ &\quad + \sum_{\substack{\mathbf{g},\mathbf{u} \\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f \\ |\mathbf{u}|\neq 1,k}} u_j S_{i,\mathbf{u}} R_{j,\mathbf{g}} \Bigg) \end{aligned} \tag{3.13}$$

Now we wish to find the $b^{th}$row of the Left hand side of Eq. (1.1) corresponding to the multi-index $\mathbf{k}_f$, which we will name $[\mathbf{B}D\mathbf{S}\mathbf{R}]_{b,\mathbf{k}_f}$. This is given by

$$[\mathbf{B}D\mathbf{S}\mathbf{R}]_{b,\mathbf{k}_f} = [\mathbf{B}\dot{\mathbf{x}}]_{b,\mathbf{k}_f} = \sum_{i=1}^{2n} \mathbf{B}_{bi} \dot{x}_{i,\mathbf{k}_f} \tag{3.14}$$

## 3.2 Right hand side

The first term on the Right hand side of the invariance equation (1.1) before inserting the parametrisation is $\mathbf{A}\mathbf{x}$. Using the parametrisation of the SSM it can be written as $\mathbf{A}\mathbf{S}(\mathbf{p})$. We obtain the $b^{th}$ row of this term for the same multi-index $\mathbf{k}_f$ as in section 3.1.1 by summing over the rows of the expansion of $\mathbf{S}$ corresponding to multi-index $\mathbf{k}_f$:

$$[\mathbf{A}\mathbf{S}]_{b,\mathbf{k}_f} = \sum_{i=1}^{2n} \mathbf{A}_{bi} S_{i,\mathbf{k}_f} \mathbf{p}^{\mathbf{k}_f} \tag{3.15}$$

Consider now the external force $\mathbf{F}$. The force can be written as a Taylor-expansion as well:

$$\mathbf{F}(\mathbf{x}) = \sum_{\mathbf{m}} \mathbf{F}_{\mathbf{m}} \mathbf{x}^{\mathbf{m}} \tag{3.16}$$

The multi-indices $\mathbf{m}$ are $N$-dimensional. The vector $\mathbf{F}_{\mathbf{m}}$ is in $\mathbb{R}^{2n}$ and $|\mathbf{m}| > 1$ since all the linear terms are already absorbed in the matrix $\mathbf{A}$. Here we use the results from section 2.3, where we notice, that if we want to find the power $s$ of a row $i$ of $\mathbf{S}$ it can be written as

$$((\mathbf{S}(\mathbf{p}))_i)^s = \big(\sum_{\mathbf{u}} S_{i,\mathbf{u}} \mathbf{p}^{\mathbf{u}}\big)^s = \sum_{\mathbf{h}} H_{i,s,\mathbf{h}} \mathbf{p}^{\mathbf{h}} \tag{3.17}$$

$H_{i,s,\mathbf{h}}$ is recursively defined as

$$H_{i,s,\mathbf{h}} = \frac{s}{h_j} \sum_{\mathbf{u} \leq \mathbf{h}} u_j S_{i,\mathbf{u}} H_{i,s-1,\mathbf{h}-\mathbf{u}} \tag{3.18}$$

Properties of these coefficients are

1. $H_{i,1,\mathbf{h}} = S_{i,\mathbf{h}}$ and $H_{i,0,\mathbf{h}} = 0$ is true for all $\mathbf{h} \neq \mathbf{0}$, as shown in section 2.3.
2. There is no constant term in the nonlinear force, and also none in the parametrisation of the SSM, which gives $H_{i,s,\mathbf{0}} = 0$, $\forall s \neq 0$, $\mathbf{0}$ being the null-vector.
3. $H_{i,0,\mathbf{0}} = 1$.

Due to property (2) the sum in Eq. (3.18) only includes the contributions where $\mathbf{u} < \mathbf{h}$ for $s > 1$, because $\mathbf{u} = \mathbf{h}$ $H_{i,s,\mathbf{0}} = 0$.

For $s = 1$ the composition coefficient multiplied to the SSM coefficient is one and therefore the only nonzero contribution in the sum in Eq. (3.18) results from $\mathbf{u} = \mathbf{h}$ and thus $H_{i,1,\mathbf{u}} = S_{i,\mathbf{u}}$.

Since the lowest order term in the expansion of $\mathbf{S}$ is of order 1, the lowest order term that can appear in the expansion as a function of $H_{i,s,\mathbf{h}}$ necessarily is of order $s$, i.e in Eq. (3.18) $|\mathbf{h}| \geq s$. This means that Eq. (3.17) becomes

$$((\mathbf{S}(\mathbf{p}))_i)^s = \sum_{\substack{\mathbf{h} \\ |\mathbf{h}| \geq s}} H_{i,s,\mathbf{h}} \mathbf{p}^{\mathbf{h}} \tag{3.19}$$

After substituting $\mathbf{x} = \mathbf{S}(\mathbf{p})$ the term $\mathbf{x}^{\mathbf{m}}$ can be evaluated row by row

$$(\mathbf{x}^{\mathbf{m}}) = \prod_{i=1}^{2n} (x_i)^{m_i} \tag{3.20}$$

where $(x_i)^{m_i} = ((\mathbf{S}(\mathbf{p}))_i)^{m_i}$. Here $m_i$ is the $i^{th}$ element in $\mathbf{m}$. As a reminder, in the following treatment, the multi-indices denoted by $\mathbf{m}$ are in $\mathbb{N}^{2n}$, but all the other multi-indices $\mathbf{h}_i$ and also $\mathbf{k}_f$ are in $\mathbb{N}^l$. Upon plugging Eq. (3.20) and Eq. (3.17) into Eq. (3.16) we get an expression looking like

$$
\begin{aligned}
\mathbf{F}(\mathbf{S}(\mathbf{p})) &= \sum_{\mathbf{m}} \mathbf{F}_{\mathbf{m}} \prod_{i=1}^{2n} (\mathbf{S}(\mathbf{p}))_i)^{m_i} \\
&= \sum_{\mathbf{m}} \mathbf{F}_{\mathbf{m}} \prod_{i=1}^{2n} \Big( \sum_{\substack{\mathbf{h}_i \\ |\mathbf{h}_i| \geq m_i}} H_{i,m_i,\mathbf{h}_i} \mathbf{p}^{\mathbf{h}_i} \Big)
\end{aligned}
\tag{3.21}
$$

We wish to get the component corresponding to the multi-index power $\mathbf{k}_f$. This is achieved by imposing the condition, that only the elements where $\sum_{i=0}^{2n} \mathbf{h}_i = \mathbf{k}_f$ contribute.
The restriction yet to be stated is to specify over which $\mathbf{m} \in \mathbb{N}^{2n}$ this sum in Eq. (3.21) has to be taken. This restriction is given by the fact that for every $i$ in the sum $|\mathbf{h}_i| \geq m_i$ has to hold. Therefore any term in Eq. (3.21) that includes multi-indices $\mathbf{m}$ of order higher than $|\mathbf{k}_f| = k$ cannot contribute, since then the $\mathbf{h}_i$s could not sum up to $\mathbf{k}_f$. Eq. (3.21) also does not contain any coefficients $H_{i,1,\tilde{\mathbf{h}}_i}$ where $\tilde{\mathbf{h}}_i = \mathbf{k}_f$ since the force does not contain any linear terms. This is important since, otherwise, the force itself would include the SSM-coefficients at highest order. The term in Eq. (3.21) corresponding to the multi-index power $\mathbf{k}_f$, here called $[\mathbf{F}(\mathbf{S})]_{\mathbf{k}_f}$ then looks like

$$
\begin{aligned}
[\mathbf{F}(\mathbf{S})]_{\mathbf{k}_f} &= \sum_{\substack{\mathbf{m} \\ |\mathbf{m}| \leq k}} \mathbf{F}_{\mathbf{m}} \underbrace{\Big( \sum_{\substack{\mathbf{h}_1 \\ |\mathbf{h}_1| \geq m_1}} \cdots \sum_{\substack{\mathbf{h}_{2n} \\ |\mathbf{h}_{2n}| \geq m_{2n}}} \Big)_{\sum \mathbf{h}_i \overset{!}{=} \mathbf{k}_f} H_{1,m_1,\mathbf{h}_1} \cdots H_{2n,m_{2n},\mathbf{h}_{2n}}}_{=:\pi_{\mathbf{m},\mathbf{k}_f}} \\
&= \sum_{\substack{\mathbf{m} \\ |\mathbf{m}| \leq k}} \mathbf{F}_{\mathbf{m}} \pi_{\mathbf{m},\mathbf{k}_f}
\end{aligned}
\tag{3.22}
$$

This is now a row vector in $\mathbb{C}^{2n}$. The contribution to the $b^{th}$ row of the invariance equation (1.1) is given by

$$
\begin{aligned}
[\mathbf{FS}]_{b,\mathbf{k}_f} &= ([\mathbf{F}(\mathbf{S})]_{\mathbf{k}_f})_b \\
&= \sum_{\substack{\mathbf{m} \\ |\mathbf{m}| \leq k}} \mathbf{F}_{b,\mathbf{m}} \pi_{\mathbf{m},\mathbf{k}_f}
\end{aligned}
\tag{3.23}
$$

with $\mathbf{F}_{b,\mathbf{m}} := (\mathbf{F}_{\mathbf{m}})_b$.

## 3.3 Invariance equation in order k

Now all terms of the rows of Eq. (1.1) have been calculated and the expressions of the coefficients can be put together.

$$[\mathbf{B}D\mathbf{S}\mathbf{R}]_{b,\mathbf{k}_f} = [\mathbf{A}\mathbf{S}]_{b,\mathbf{k}_f} + [\mathbf{F}\mathbf{S}]_{b,\mathbf{k}_f} \tag{3.24}$$

At first order, i.e. $\mathbf{k}_f = \hat{\mathbf{e}}_f$:

$$\sum_{i=1}^{2n} \mathbf{B}_{bi} \sum_{j=1}^{l} S_{i,\hat{\mathbf{e}}_j} R_{j,\mathbf{e}_f} = \sum_{i=1}^{2n} \mathbf{A}_{bi} S_{i,\mathbf{e}_f} \tag{3.25}$$

Using the results of Ponsioen, Pedergnana and Haller (2018, p. 272) [2] we know that the linear contribution of the expansion of the reduced dynamics in tensor-format is given by $\mathbf{R}_1 = \mathbf{\Lambda}_M$ which is a matrix containing the master mode eigenvalues on its diagonal. This translates directly to the multi-index format since at linear order the coefficients in tensor- and multi-index-format coincide. Therefore we can write:

$$R_{j,\mathbf{e}_f} = \delta_{jf} \lambda_f \tag{3.26}$$

Inserting Eq. (3.26) into Eq. (3.25) we obtain

$$\sum_{i=1}^{2n} \mathbf{B}_{bi} S_{i,\hat{\mathbf{e}}_f} \lambda_f = \sum_{i=1}^{2n} \mathbf{A}_{bi} S_{i,\mathbf{e}_f} \tag{3.27}$$

This equation is solved by the right eigenvector fulfilling $(\mathbf{B}\lambda_f - \mathbf{A}) \cdot \mathbf{v}_f = 0$. Therefore, these vectors are the first order coefficients of the SSM expansion. Next the expressions found for the individual terms of Eq. (3.24) at higher orders can be inserted, which yields

$$\begin{aligned} &\sum_{i=1}^{2n} \mathbf{B}_{bi} \sum_{\substack{j=1 \\ (\mathbf{k}_a)_j, u_j > 0}}^{l} \left( \sum_{a=1}^{l} \delta_{ja} S_{i,\hat{\mathbf{e}}_a} R_{j,(\mathbf{k}_f - \hat{\mathbf{e}}_a + \hat{\mathbf{e}}_j)} + \sum_{a=1}^{z_k} (\mathbf{k}_a)_j S_{i,\mathbf{k}_a} R_{j,(\mathbf{k}_f - \mathbf{k}_a + \hat{\mathbf{e}}_j)} \right. \\ &\quad \left. + \sum_{\substack{\mathbf{g},\mathbf{u} \\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j = \mathbf{k}_f \\ |\mathbf{u}| \neq 1,k}} u_j S_{i,\mathbf{u}} R_{j,\mathbf{g}} \right) = \sum_{i=1}^{2n} \mathbf{A}_{bi} S_{i,\mathbf{k}_f} + \sum_{\substack{\mathbf{m} \\ |\mathbf{m}| \leq k}} \mathbf{F}_{b,\mathbf{m}} \pi_{\mathbf{m},\mathbf{k}_f} \end{aligned} \tag{3.28}$$

As a final step the whole expression is rewritten, and terms that include $S_{i,\mathbf{u}}$ with $|\mathbf{u}| = k$ are grouped on the left side of the equation.

$$\begin{aligned}
&\sum_{i=1}^{2n}\left(\mathbf{B}_{bi}\sum_{\substack{j=1\\(\mathbf{k}_a)_j>0}}^{l}\sum_{a=1}^{z_k}(\mathbf{k}_a)_j S_{i,\mathbf{k}_a}R_{j,(\mathbf{k}_f-\mathbf{k}_a+\hat{\mathbf{e}}_j)}-\mathbf{A}_{bi}S_{i,\mathbf{k}_f}\right)\\
&=\sum_{\substack{\mathbf{m}\\|\mathbf{m}|\leq k}}\mathbf{F}_{b,\mathbf{m}}\pi_{\mathbf{m},\mathbf{k}_f}-\sum_{i=1}^{2n}\mathbf{B}_{bi}\sum_{\substack{j=1\\u_j>0}}^{l}\left(\sum_{a=1}^{l}\delta_{ja}S_{i,\hat{\mathbf{e}}_a}R_{j,(\mathbf{k}_f-\hat{\mathbf{e}}_a+\hat{\mathbf{e}}_j)}\right.\\
&\quad\left.+\sum_{\substack{\mathbf{g},\mathbf{u}\\\mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f\\|\mathbf{u}|\neq 1,k}}u_jS_{i,\mathbf{u}}R_{j,\mathbf{g}}\right)
\end{aligned}\tag{3.29}$$

In total there are $z_k \cdot 2n$ of these equations to solve. For every multi-index $\mathbf{k}_f, f = 1, ..., z_k$ there are equations for every row $b = 1, ..., 2n$.
It can directly be seen, that the equation for every single $\mathbf{k}_f$ is coupled to the other equations which contain expressions of $S$ of maximal expansion order $k$, explicitly by the terms $S_{i,\mathbf{k}_a}$ on the Left hand side of Eq. (3.29).
We wish to solve this equation for the values $S_{i,\mathbf{k}_j}$ $i = 1, .., 2n$; $j = 1, ..., z_k$ where $|\mathbf{k}_j| = k$, $\forall j$. Furthermore, the values $R_{i,\mathbf{k}_j}$ $i = 1, .., l$; $j = 1, ..., z_k$ are also unknown. They will be determined be removing inner resonances appearing in the coefficient matrix. The details to this are discussed in section 4.3.2. All the other components are already known from earlier iterations, and the coefficients of the non-linearity $\mathbf{F}$ are also known. If the linear order reduced dynamics, the equations decouple, and for every single multi-index at order $k$ there exists a smaller $N \times N$ coefficient matrix, which is the block diagonal part of the coefficient matrix of the coupled system. This smaller matrix and the rows in the RHS corresponding to the multi-index then determine the SSM-coefficients for this multi-index.

# 4 Implementation

## 4.1 Storing the coefficients

At each order $k$ the coefficients are stored in an array. These arrays are stacked into cell arrays with ascending order.

1. At order $k$ the SSM-coefficients are stored in $\mathtt{S}_k \in \mathbb{C}^{2n\times z_k}$. It contains $S_{i,\mathbf{k}_f}$ in row $i$ and column $f$. The $\mathtt{S}_k$ are stored in a cell array $\mathtt{S}$ in position $k$.
2. The coefficients of the expansion of the reduced dynamics at order $k$ are stored in a matrix $\mathtt{R}_k \in \mathbb{C}^{l\times z_k}$. They are stored in the same way as the SSM-coefficients, in a cell array $\mathtt{R}$.
3. For the composition-coefficients of power-series arising in Eq. (3.18) $H_{i,s,\mathbf{h}}$ the following matrix is used: $\mathtt{H}_k$ It has dimension $N \times k \times z_k$. It contains the respective indices of the coefficients in each entry. The first row corresponds to the different directions in phase space, the second to the scalar powers, of which the expansion of the SSM is taken and the third to all the order $k$ multi-indices.

## 4.2 Solving the invariance equation

In order to implement the whole computation of the invariance equation (1.1) at order k we seek for a linear equation of the form

$$\mathtt{C}_k \mathtt{x}_k = \mathtt{b}_k \tag{4.1}$$

for a coefficient matrix $\mathtt{C}_k$ and Right hand side vector $\mathtt{b}_k$, $\mathtt{x}_k$ containing all the $S_{i,\mathtt{m}}$, $|\mathtt{m}| = k, i = 1, .., 2n$. We will now discuss, the size of the system for every order and also rewrite the terms of the equation in a way that is convenient, in order to make calculations as simple and efficient as possible.
We take $z_k, l, n$ as above. Then we wish to solve for the vector

$$\mathtt{x}_k \in \mathbb{C}^{2n\cdot z_k}, \quad \mathtt{x}_k = \left[ S_{(1,\mathbf{k}_1)}, \cdots, S_{(2n,\mathbf{k}_1)}, \cdots, S_{(1,\mathbf{k}_{z_k})}, \cdots, S_{(2n,\mathbf{k}_{z_k})} \right]^T \tag{4.2}$$

Thus $\mathtt{x}_k = \mathrm{vec}(S_k)$ where the vectorisation operator $\mathrm{vec}(\mathtt{M})$ can be understood as reshaping the array $\mathtt{M}$ to dimensions $([\,], 1)$. The coefficient matrix and the Right hand side vector have dimensions

$$\mathtt{C}_k \in \mathbb{C}^{2n\cdot z_k \times 2n\cdot z_k} \tag{4.3}$$

$$\mathtt{b}_k \in \mathbb{C}^{2n\cdot z_k} \tag{4.4}$$

## 4.3 SSMmultiindex

This matlab function calculates the SSM-coefficients. The inputs have to be prepared as follows:

- `H`, cell array containing the coefficients for the expansion of $\mathbf{S}^s$
- $k$, Order of the SSM expansion
- $l$, dimension of the SSM
- `S`, cell array containing coefficients of S at lower orders than $k$
- `R`, cell array containing coefficients of R at lower orders than $k$
- `data`, struct containing the characteristic quantities of the system
  - **A** and **B**, system matrices
  - $N$, Phase space size
  - `Lambda_M`, matrix containing the master eigenvalues on its diagonal
  - `resonance`, contains results of the resonance analysis
  - `F`, nonlinearity coefficients for all orders
  - `Fmulti`, corresponding multi-indices at all orders
  - `option`, which of the options has been chosen for inputing the non-linearity matrices
  - `Inonzero`- cell array with the indices of nonzero force, for all orders

The input `data.option` can be choosen as

- `'one'`: All the columns of `F{k}` that contain only zeros are deleted and also the corresponding columns in `Fmulti{k}` are deleted, this f. This makes sure to minimize calculation efforts.
- `'two'`: An additional array `Inonzero` is passed into the struct `data`: it contains the position of nonzero-columns in `F{k}` for all $k$. In each entry, there is a row array containing these positions.

The inputs **A**, **B**, `F` and `Fmulti` can be sparse/ contain sparse arrays. For orders at which the nonlinearity has no contributions an empty array can be fed into `F` and `Fmulti`.

Listing 1: `SSMmultiindex`

```
function [Sk,Rk, Hk] = SSMmultiindex(data, k, H, S, R, l)

A           = data.A;               % A matrix
B           = data.B;               % B matrix
```

```
N            = data.N;                % Phase space dimension, N = 2n
Lambda_M     = data.Lambda_M;
resonance    = data.resonance;
F            = data.F;                % Nonlinearity coefficients
Fmulti       = data.Fmulti;           % Force Multi−Indices
option       = data.option;           % Force Input Option

switch option
    case 'one'
        % arrays in F only contain the columns with nonzero entries
    case 'two'
        Inonzero = data.Inonzero;
end


%Order k multi−indices in rev−lex. order
K            = flip(sortrows(nsumk(l,k,'nonnegative')).',2);
% number of multi−indices at order k
z_k          = size(K,2);

% **Outer resonance case**
if resonance.outer.occurs
    prompt     = 'Due to (near) outer resonance, the exisitence ...
        of SSM is questionable and the underlying computation ...
        may suffer: Do you wish to continue? (Press 1 for Yes ...
        and 0 for No)';
    user_input = input(prompt);
    if user_input
        disp('Attempting SSM computation')
    else
        disp('Aborting SSM computation')
        return
    end
end

%% Assemble the coefficient matrix of SSM
Ck   = CoefficientMatrix(l,z_k,N,R{1},A,B,K);

%%Assemble RHS%%

% Mixed Terms
RHS = MixedTerms(k,l,z_k,N,S,R,B,K);

% Force Terms
%Compute the coefficients for composition of power series
Hk   = CompositionCoefficients(k,l,z_k,N,S,K,H);

% Force contribution
switch option
    case 'one'
        RHS = RHS + Force(k,z_k,N,K,F,H,Hk,Fmulti,[]);
    case 'two'
        RHS = RHS + Force(k,z_k,N,K,F,H,Hk,Fmulti,Inonzero);
end
RHS = reshape(RHS,N*z_k,1);

% Check for near inner Resonances
```

```
[U,Sigma,¬] = svd(full(Ck));
tol         = 1e−3;
ref         = abs(Lambda_M(1,1));
abstol      = tol*ref;

if Sigma(end,end)/ref < tol
    innerresonance = true;
else
    innerresonance = false;
end

if innerresonance
    % Project equation to kernel of Ck
    V       = U(:,diag(Sigma)/ref < tol);
    Skron   = kron(eye(z_k),B*S{1});
    Rk      = V'*Skron \ V'* RHS;
    Sk      = lsqminnorm(Ck,RHS− Skron*Rk,abstol);
else
    Rk = zeros(l*z_k,1);
    Sk = lsqminnorm(Ck,RHS);
end

Rk          = reshape(Rk,l,z_k);
Sk          = reshape(Sk,N,z_k);
Hk(:,:,1)   = Sk;
```

The functions used here are discussed in the chapters below. `SSMmultiindex` outputs

- `S`, cell array containing coefficients of the taylor expansion of **S** up to order $k$
- `R`, cell array containing coefficients of the taylor expansion of **R** up to order $k$
- `H`, cell array containing all the calculated composition-coefficients

### 4.3.1 Coefficient matrix

The Coefficient Matrix $\mathbf{C}_k$ is determined by the Left hand side of Eq. (3.29). The contribution from Eq. (3.11) will be rewritten as follows. Instead of writing the coefficients as a function of all $|\mathbf{u}| = k$, they will be represented as a function of all $|\mathbf{g}| = 1$. By imposing the condition $\mathbf{g} + \mathbf{u} = \mathbf{k}_f + \hat{\mathbf{e}}_j$, $\forall j$, we can enslave one of the two free multi-indices as a function of the other, which justifies this step. Therefore we can write

$$\sum_{\substack{j=1\\u_j>0}}^{l}\sum_{\substack{\mathbf{g},\mathbf{u}\\\mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f\\|\mathbf{u}|=k}} u_j S_{i,\mathbf{u}} R_{j,\mathbf{g}} = \sum_{\substack{j=1\\u_j>0}}^{l}\sum_{\substack{\mathbf{g},\mathbf{u}\\\mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f\\|\mathbf{g}|=1}} u_j S_{i,\mathbf{u}} R_{j,\mathbf{g}}$$

$$= \sum_{\substack{j,a=1\\(\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a)_j>0}}^{l} (\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a)_j S_{i,\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a} R_{j,\hat{\mathbf{e}}_a}$$

$$= \sum_{\substack{j,a=1\\(\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a)_j>0}}^{l} (k_{f,j}+1-\delta_{aj}) S_{i,\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a} R_{j,\hat{\mathbf{e}}_a} \tag{4.5}$$

since $\mathbf{u} = \mathbf{k}_f + \hat{\mathbf{e}}_j - \hat{\mathbf{e}}_a$ has to hold. Now we can simply calculate $\hat{\mathbf{e}}_a - \hat{\mathbf{e}}_j$ for all combinations of $a$ and $j$. Upon subtracting the resulting terms from $\mathbf{k}_f$ we get all the coupling terms in $|\mathbf{u}| = k$ that will appear in the invariance equation for each $\mathbf{k}_f$.

To progress we split the Left hand side of Eq. (3.29) into a part containing $S_{i,\mathbf{k}_f}$ and then another part containing all the coupling terms with $S_{i,\mathbf{k}_a}$, $a \neq f$.

$$\sum_{i=1}^{2n}\left(\mathbf{B}_{bi}\sum_{\substack{j,a=1\\(\mathbf{k}_f)_j+1-\delta_{aj}>0}}^{l}(k_{f,j}+1-\delta_{aj})S_{i,\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a}R_{j,\hat{\mathbf{e}}_a} - \mathbf{A}_{bi}\mathbf{S}_{i,\mathbf{k}_f}\right)$$

$$= \sum_{i=1}^{2n}\left(\mathbf{B}_{bi}\sum_{\substack{j,a=1\\k_{f,j}>0}}^{l}\delta_{aj}(k_{f,j})R_{j,\hat{\mathbf{e}}_a} - \mathbf{A}_{bi}\right)S_{i,\mathbf{k}_f}$$

$$+ \mathbf{B}_{bi}\sum_{\substack{j,a=1\\j\neq a}}^{l}(k_{f,j}+1-\delta_{aj})S_{i,\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a}R_{j,\hat{\mathbf{e}}_a} \tag{4.6}$$

$$= \underbrace{\sum_{i=1}^{2n}\left(\mathbf{B}_{bi}\sum_{j=1,k_{f,j}>0}^{l}k_{f,j}R_{j,\hat{\mathbf{e}}_j} - \mathbf{A}_{bi}\right)}_{:=C_{b,i,\mathbf{k}_f,\mathbf{k}_f}} S_{i,\mathbf{k}_f}$$

$$+ \sum_{i=1}^{2n}\sum_{\substack{j,a=1\\j\neq a}}^{l}\underbrace{\mathbf{B}_{bi}(k_{f,j}+1)R_{j,\hat{\mathbf{e}}_a}}_{:=C_{b,i,\mathbf{k}_f,(\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a)}} S_{i,\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a} \tag{4.7}$$

Here the coefficients $C_{b,i,\mathbf{k}_f,\mathbf{k}_f+\hat{\mathbf{e}}_j-\hat{\mathbf{e}}_a}$ are numbers corresponding to the given multi-indices, the row $i$ of the coefficient $S_{(i,..)}$ and the row $b$ of the matrices

**A**, **B**. The first part in Eq. (4.7) is the component, that will appear in the block-diagonal element of size $2n \times 2n$ in the coefficient matrix for every single $\mathbf{k}_f$. The coefficient matrix $\mathtt{C}_k$ for the equation at order k then looks like

$$\mathtt{C}_k = \begin{bmatrix} \mathbf{C}_{\mathbf{k}_1,\mathbf{k}_1} & \cdots & \mathbf{C}_{\mathbf{k}_1,\mathbf{k}_{z_k}} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{\mathbf{k}_{z_k},\mathbf{k}_1} & \cdots & \mathbf{C}_{\mathbf{k}_{z_k},\mathbf{k}_{z_k}} \end{bmatrix} \tag{4.8}$$

It is filled with matrix elements $\mathbf{C}_{\mathbf{k}_i,\mathbf{k}_j} \in \mathbb{C}^{2n \times 2n}$, which are defined as

$$\mathbf{C}_{\mathbf{k}_i,\mathbf{k}_j} = \begin{pmatrix} C_{1,1,\mathbf{k}_i,\mathbf{k}_j} & \cdots & C_{1,2n,\mathbf{k}_i,\mathbf{k}_j} \\ \vdots & \ddots & \vdots \\ C_{1,1,\mathbf{k}_i,\mathbf{k}_j} & \cdots & C_{1,2n,\mathbf{k}_i,\mathbf{k}_j} \end{pmatrix} \tag{4.9}$$

**Algorithm**
Input of `CoefficientMatrix`:

- $l$, the SSM dimension
- $z_k$, the amount of multi-indices that exist for order $k$ and size $l$
- $N$, the dimension of the phase space
- `R1`, the array containing the linear order reduced dynamics
- **A**, **B**, the system matrices
- **K**, containing in its columns the set of multi-indices of order $k$ and size $l$ in reverse-lexicographical ordering

It outputs the coefficient matrix used in Eq. (4.1).

The multiplication $k_{f,j} R_{j,\hat{\mathbf{e}}_j}$ in Eq. (4.7) is executed for all $f = 1, .., z_k$ simultaneously in line 3 of the code. In line 6, a block diagonal matrix is created, that looks like

$$\mathtt{Cff} = \begin{pmatrix} \mathbf{B}(\mathbf{k}_1 \cdot diag(\mathtt{R\{1\}})) - \mathbf{A} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{B}(\mathbf{k}_{z_k} \cdot diag(\mathtt{R\{1\}})) - \mathbf{A} \end{pmatrix} \tag{4.10}$$

Starting in line 16, the other terms corresponding to matrix elements $\mathbf{C}_{\mathbf{k}_i,\mathbf{k}_j}$ for $i \neq j$ are assembled. If the linear order reduced dynamics are diagonal, there will be no further terms, since all non-zero contributions in Eq. (4.7) have been already taken care of. The loop then checks for all combinations of $j$ and $a$, if

$\mathbf{k}_f + \hat{\mathbf{e}}_j - \hat{\mathbf{e}}_a$ is a valid multi-index, if so - then the contribution is stored in an array `M1` in position $(f, i)$ where $i$ corresponds to the position of the resulting multi-index in $\mathbf{K}$. Finally, the contributions corresponding to all cases in Eq. (4.7) where $j \neq a$ are added to the block-diagonal part of the coefficient matrix in line 32.

Listing 2: `CoefficientMatrix`

```
1  function [C] = CoefficientMatrix(l,z_k,N,R1,A,B,K)
2  % This function assembles the coefficient matrix C_k to solve ...
       for the SSM-coefficients.
3  R1c    = sparse(repelem(sum(K.*diag(R1)),N).');
4
5  %Assemble block-diagonal part of Coefficient matrix
6  Cff = sparse(KronProd({B,speye(z_k)},[1,2], [nan,nan])).* R1c - ...
       sparse(KronProd({A,speye(z_k)},[1,2],[nan,nan]));
7
8  unit = eye(l);
9  M1 = sparse(z_k,z_k);
10
11 %if R1 is diagonal, only block diagonal matrix elements
12 if isempty(find(R1-diag(diag(R1)),1))
13     C = Cff;
14 else
15     %loop over all highest order multi-indices
16     for f = 1:z_k
17         kf = K(:,f);
18         %loop over order 1 multi-indices
19         for j = 1:l
20             %loop over all order 1 multi-indices that are not e_j
21             for a = 1:l
22                 if j ≠ a
23                     index = kf + unit(:,j)-unit(:,a);
24                     if isempty(index(index<0))
25                         index       = Multi2Index(index,'antilex');
26                         M1(f,index) = R1(j,a)* ( kf(j) + 1);
27                     end
28                 end
29             end
30         end
31      end
32     C = sparse(Cff +(repmat(B,z_k,z_k)-kron(eye(z_k),B)) .* ...
         repelem(M1.',N,N));
33 end
34 end
```

### 4.3.2 Right hand side vector

The Right hand side of the invariance equation (3.29) can be split to two contributions

$$\sum_{\substack{\mathbf{m}\\|\mathbf{m}|\le k}} \mathbf{F}_{b,\mathbf{m}}\pi_{\mathbf{m},\mathbf{k}_f} - \sum_{i=1}^{2n}\mathbf{B}_{bi}\sum_{\substack{j=1\\u_j>0}}^{l}\left(\sum_{a=1}^{l}\delta_{ja}S_{i,\hat{\mathbf{e}}_a}R_{j,(\mathbf{k}_f-\hat{\mathbf{e}}_a+\hat{\mathbf{e}}_j)}\sum_{\substack{\mathbf{g},\mathbf{u}\\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f\\|\mathbf{u}|\neq 1,k}} u_jS_{i,\mathbf{u}}R_{j,\mathbf{g}}\right) =$$

$$\underbrace{\sum_{\substack{\mathbf{m}\\|\mathbf{m}|\le k}} \mathbf{F}_{b,\mathbf{m}}\pi_{\mathbf{m},\mathbf{k}_f} - \sum_{i=1}^{2n}\mathbf{B}_{bi}\sum_{\substack{j=1\\u_j>0}}^{l}\sum_{\substack{\mathbf{g},\mathbf{u}\\ \mathbf{g}+\mathbf{u}-\hat{\mathbf{e}}_j=\mathbf{k}_f\\|\mathbf{u}|\neq 1,k}} u_jS_{i,\mathbf{u}}R_{j,\mathbf{g}}}_{=:L_{(k,b,\mathbf{k}_f)}}$$

$$\underbrace{-\sum_{i=1}^{2n}\mathbf{B}_{bi}\sum_{\substack{j=1\\m_j>0}}^{l}\delta_{ja}S_{i,\hat{\mathbf{e}}_a}R_{j,(\mathbf{k}_f-\hat{\mathbf{e}}_a+\hat{\mathbf{e}}_j)}}_{=:b_{(b,\mathbf{k}_f,R_k)}} \tag{4.11}$$

The numbers $L_{(k,b,\mathbf{k}_f)}$ and $b_{(b,\mathbf{k}_f,R_k)}$ are stored in arrays $\mathbf{L}_k$ and $\mathtt{b}_{k,R_k}$ as follows:

$$\mathbf{L}_k = \begin{bmatrix} L_{(k,1,\mathbf{k}_1)}\\ \vdots \\ L_{(k,2n,\mathbf{k}_1)}\\ \vdots \\ L_{(k,1,\mathbf{k}_{z_k})}\\ \vdots \\ L_{(k,2n,\mathbf{k}_{z_k})}\end{bmatrix}, \mathtt{b}_{k,R_k} = \begin{bmatrix} b_{(1,\mathbf{k}_1,R_k)}\\ \vdots \\ b_{(2n,\mathbf{k}_1,R_k)}\\ \vdots \\ b_{(1,\mathbf{k}_{z_k},R_k)}\\ \vdots \\ b_{(2n,\mathbf{k}_{z_k},R_k)}\end{bmatrix} \tag{4.12}$$

Then by subtracting these two arrays one obtains the Right hand side vector of the whole system in Eq. (4.1)

$$\mathtt{b}_k = \mathbf{L}_k - \mathtt{b}_{k,R_k} \in \mathbb{C}^{2n\cdot z_k} \tag{4.13}$$

**Algorithm** The Right hand side is implemented in three steps. The first step computes the contribution arising from nonlinear orders of the reduced dynamics, the second step the contribution of the nonlinear forces acting in and on the system, which are stored in $\mathtt{L}_\mathbf{k}$. Then in the third step the order $k$ reduced dynamics are determined.

**Mixed Terms** The first function `MixedTerms` takes as input

- $k$, order at which SSM-coefficients are calculated
- $l$, the SSM dimension
- $z_k$, the amount of multi-indices that exist for order $k$ and size $l$
- $N$, the dimension of the phase space
- `S`, the SSM-coefficients cell array
- `R`, the reduced dynamics cell array
- $\mathbf{B}$, the system matrix
- $\mathbf{K}$, containing in its columns the set of multi-indices of order $k$ and size $l$ in reverse-lexicographical ordering

It returns the contribution to $\mathbf{L}_k$ that corresponds to higher order terms of the reduced dynamics, that are not of order $k$. In line 4 a loop over the orders of the

SSM-coefficients contributing to the mixed terms is excuted. Consequently in line 7 all order $u$ multi-indices are created. For each of those, all the valid multi-indices $\mathbf{g} = \mathbf{k}_f - \mathbf{u}_{\texttt{index}} + \hat{\mathbf{e}}_j$ are calculated for all $j = 1, .., l$ and $f = 1, .., z_k$. This this is done in line 16. Then in line 20, all the combinations of $\mathbf{u}_{\texttt{index}}, j$ and $f$ that lead to a positive nonzero multi-index $\mathbf{g}$ are read out of the SSM-coefficients and the reduced dynamics cell arrays. These are then multiplied to achieve coefficients according to Eq. (3.29). In line 21 these are then subtracted from the multi-index entry of the matrix `RHS`, that they correspond to. Finally the multiplication with $\mathbf{B}$ is carried out. `RHS` has dimensions $N \times z_k$ and has to be reshaped, which is done in the `SSMmultiIndex` function.

Listing 3: `MixedTerms`

```
1  function [RHS] = MixedTerms(k,l,z_k,N,S,R,B,K)
2  % This function computes all the mixed terms of the invariance ...
       equation at order
3  RHS = zeros(N,z_k);
4  for u = 2:k−1
5
6      % Rev.−lex. ordered multi−indices of order u
7      multis = flip(sortrows(nsumk(l,u,'nonnegative')).',2);
8
9      %loops over e_j and order u multi−indices
10     for j  = 1:l
11         ej = (1:l == j).';
12         for index = 1:size(multis,2)
13             if  multis(j,index) ≠ 0
14
```

```
                    % all valid multi—indices that can lead to ...
                        contributions
                    [g,I_k,¬] = gminusk(multis(:,index),K+ej);
                    if ¬isempty (g)
                        Ig        = Multi2Index(g,'antilex');
                        term  = multis(j,index) * S{u}(:,index) .* ...
                            R{(k—u+1)}(j,Ig);
                        for f = unique(I_k)
                            RHS(:,f) = RHS(:,f) — sum(term(:,I_k==f),2);
                        end
                    end
                end

            end
        end
end
RHS = B * RHS;
end
```

**Force** The second function `Force` computes the contribution to $\mathbf{L}_k$ originating in the external force, called `FS` . It takes as input

- $k$, order at which SSM-coefficients are calculated
- $z_k$, the amount of multi-indices that exist for order $k$ and size $l$
- $N$, the dimension of the phase space
- $\mathbf{K}$, containing in its columns the set of multi-indices of order $k$ and size $l$ in reverse-lexicographical ordering
- `H`, the cell array storing the composition-coefficients
- `Hk`, the array containing the composition-coefficients corresponding to order $k$ multi-indices
- `F`, the cell array containing in position $u$ an array with the order $u$ nonlinearity coefficients in multi-index format, each column $f$ corresponding to the nonlinearity with multi-index $\mathbf{u}_f$
- `Fmulti`, the cell array containing in position $u$ the order $u$ multi-indices corresponding to the nonlinearity coefficients in the columns of `F`$\{u\}$
- `Inonzero`, an optional index cell array. It contains in position $u$ an index array with the indices of columns in `F`$\{u\}$ that contain nonzero elements

`F` contains the nonlinearity coefficients at different orders stored in multi-index format, such that every column of `F`$\{u\}$ corresponds to the force contribution

of a multi-index of order $u$. The same column of the array `Fmulti{u}` contains this multi-index.
There are two options on how to prepare the force cell array:

1. If the input for `Inonzero` is empty, then the input of the force cell array only contains those columns of the nonlinear force in multi-index format, which have in them non-zero coefficients. The corresponding multi-index array also contains only the multi-indices corresponding to those columns.

2. If the input for `Inonzero` is not empty, it contains at position $u$ an arrays with the indices of the non-zero columns

**Example 1.** *This is an example for the second option.* $I_2 = \mathtt{Inonzero}\{2\}$ *contains one number, which is* $q$. *Then* $\mathtt{F}\{2\}(:,q)$ *is the only column that contains non-zero entries in* $\mathtt{F}\{2\}$. *The multi-index corresponding to this force contribution is in* `Fmulti`$\{2\}$*(:,q).*

These two options allow for two different ways of preparing the nonlinearities. In case (1.) $\mathtt{F}\{u\}$ only contains those columns that contain non-zero contributions, and $\mathtt{Fmulti}\{u\}$ the multi-indices corresponding to those columns. This allows for explicitly only preparing the force-coefficients that have non-trivial contributions to the system.
In case (2.) the full nonlinearity vector can be put in, and with the use of `Inonzero` only those that contribute are taken into account when calculating the force contribution to the $\mathtt{b}_{\mathbf{k}_f,R}$ terms. This is useful if one is given the force coefficients for all multi-indices at the contributing orders. Of course, one could then delete the columns at every order that are zero and the corresponding multi-indices and use the first input option as well. For orders at which there is no contribution at all, `F` and `Fmulti` can contain empty arrays.

The function `Force` in listing 4 implements the following terms in Eq. (3.22):

$$\sum_{\substack{\mathbf{m} \\ |\mathbf{m}|\leq k}} \mathbf{F}_{b,\mathbf{m}} \pi_{\mathbf{m},\mathbf{k}_f} = \sum_{\substack{\mathbf{m} \\ |\mathbf{m}|\leq k}} \mathbf{F}_{\mathbf{m}} \Big( \sum_{\substack{\mathbf{h}_1 \\ |\mathbf{h}_1|\geq m_1}} \cdots \sum_{\substack{\mathbf{h}_{2n} \\ |\mathbf{h}_{2n}|\geq m_{2n}}} \Big)_{\sum \mathbf{h}_i \overset{!}{=} \mathbf{k}_f} H_{1,m_1,\mathbf{h}_1} \cdots H_{2n,m_{2n},\mathbf{h}_{2n}} \tag{4.14}$$

The loop in row 5 corresponds to the sum over all $\mathbf{m}$, $|\mathbf{m}| \leq k$. In the next lines, the corresponding order non-linearity coefficients which are non zero are read out and looped over. For each multi-index corresponding to a non-vanishing force, the non-zero entries in it `m_i` and the number of them `szm_i` is read out. Then for all multi-indices $\mathbf{k}_f, f = 1, .., z_k$ all combinations of `szm_i` non-zero multi-indices that add up to $\mathbf{k}_f$ are calculated in line 26. They are stored in the 3rd dimension of `hv`. As an example, `hv(:,:,1)` contains in its columns multi-index vectors, and if all columns are summed one gets $\mathbf{k}_f$. The absolute value of each of those multi-indices is stored in habs, where now the second dimension corresponds to the different combinations and the first to vectors of one combination adding up to $\mathbf{k}_f$.

Only combinations consisting of szm_i multi-indices are calculated, since all the zero entries in a multi-index $\mathbf{m}$ corresponding to $\mathbf{F_m}$ do only give contributing composition-coefficients if the multi-index vector $\mathbf{h}_i$ corresponding to $m_i = 0$ is also zero. This is due to the fact, that $H_{i,0,\mathbf{h_i}} \neq 0 \leftrightarrow \mathbf{h}_i = \mathbf{0}$. This is one of the slowest steps in this function.
In line 27 the condition $|\mathbf{h}_i| \geq m_i$ in Eq. (3.22) is checked for all vectors in each combination.
Then a loop over all such combinations is executed. Now for all multi-indices in a combination their position in a reverse-lexicographical set of multi-indices of their order and size is calculated in line 38.

Consequently in line 41 a loop over all nonzero entries in $\mathbf{m}$ is executed. The if condition here is used, since the highest order composition-coefficients have not yet been read into the cell array containing the lower order ones. The multiplied values are stored in a dummy matrix pidum which contains the contributions in Eq. (3.22) corresponding to one combination of multi-index vectors $\mathbf{h}_i, i = 1, .., $szm_i in its columns and the contributions for each combination in its rows. Therefore to obtain the coefficient corresponding to $\mathbf{m}$ and $\mathbf{k}_f$ the rows are multiplied with each other pointwise and then the columns are summed over. This is done in line 50. pi contains in its rows the contributions corresponding to the multi-indices in Fmulti{order} and in its columns correspond to the multi-indices $\mathbf{k}_f$. It has size (szm_i, $z_k$). The multiplication Forder*pi in line 69 carries out the sum over $\mathbf{m}$ in Eq. (3.22) for all multi-indices of order order that correspond to force columns with nonzero entries.
Again, the resulting array FS is a $N \times z_k$ dimensional array.

Listing 4: Force

```
1  function [FS] = Force(k,z_k,N,K,F,H,Hk,Fmulti,Inonzero)
2  %Computes the contribution arising due to the nonlinear force.
3  FS = zeros(N,z_k);
4  %loop over all multi-index orders of the size 2n multi-indices
5  for order = 2:k
6      if isempty(Inonzero)
7          %option 1, all zero columns have been deleted
8          Forder  = F{order};
9          Fm      = Fmulti{order};
10         sz      = size(Forder,2);
11         columns = 1:sz;
12     else
13         %option 2, Inonzero contains positions of nonzero columns
14         columns = Inonzero{order};
15         Forder  = F{order}(:,columns);
16         Fm      = Fmulti{order}(:,columns);
17     end
```

```
    pi = zeros(size(columns,2),z_k);
    %loop over nonzero columns
    if ¬isempty(columns)
        for col = columns
            m     = Fm(:,col);
            m_i   = find(m);    % all nonzero entries of m
            szm_i = size(m_i,1);
            for f = 1:z_k
                [hv, habs] = nsumkvector(szm_i,K(:,f), 'abs');
                cond2      = habs ≥ m(m_i);
                hv(:,:,sum(cond2,1)≠size(m_i,1)) =[];
                if isempty(hv)
                    %no action since no contribution
                else
                    %number of combinations of h_is
                    combno  = size(hv,3);
                    h_iabs  = reshape(sum(hv,1),size(m_i,1),combno);
                    pidum = ones(szm_i,combno);

                    for j   = 1:combno
                        Indices = Multi2Index(hv(:,:,j),'antilex');

                        %looping over the individual h_is in ...
                            each combination
                        for ij  = 1:szm_i
                            i = m_i(ij);
                            if h_iabs(ij,j) ==k
                                pidum(ij,j) = ...
                                    Hk(i,Indices(ij),m(i));
                            else
                                pidum(ij,j) = H{h_iabs(ij,j)} ...
                                    (i,Indices(ij),m(i));
                            end
                        end
                    end
                    pi(col,f) = sum(prod(pidum),2);
                end
            end
        end
        FS = FS + Forder*pi;
    end
end
end
```

**Reduced Dynamics** If the coefficient matrix $\mathrm{C}_k$ is singular, then the SSM-coefficients are not determined uniquely. The freedom we have for choosing the reduced dynamics allows to choose it, such that the projection of the vector $\mathrm{x}_k$ onto the kernel of the unperturbed coefficient matrix is zero. This is used if $\mathrm{C}_k$ is singular to remove inner resonances, as explained by Ponsioen (2019, p 18-19)[4]. As a consequence $\mathrm{x}_k$ is in the kernel of the coefficient matrix. Let $\mathbf{V}$ be the matrix, whose columns span the kernel of $\mathrm{C}_k$. To get an equation for the coefficients of the reduced dynamics $\mathbf{R}$ at order $k$, the term $\mathtt{b}_{k,R_k}$ is rewritten as

$$\mathtt{b}_{k,R_k} = (\mathbf{I} \otimes \mathbf{B}\mathbf{S}_1)\mathrm{vec}(\mathtt{R}_k) \tag{4.15}$$

The condition to have $\mathbf{x}_k \in \mathrm{Ker}(\mathbf{C}_k)$ can be achieved by choosing the reduced dynamics at highest order, such that they solve

$$\mathbf{V}^{\mathbf{T}}\left(\mathbf{L}_k + (\mathbf{I} \otimes \mathbf{S}_1)\mathrm{vec}(\mathtt{R}_k)\right) = 0 \tag{4.16}$$

This can be solved for $\mathtt{R}_k$ as a linear system of equations.

**Algorithm** If the smallest singular value, divided by the eigenvalue of $\mathbf{A}$ with the largest real part is smaller than `tol` we consider the coefficient matrix as singular and near inner resonances appear. In line 13 the vectors spanning the kernel of `Ck` are extracted and then in line 15 the reduced dynamics is calculated according to Eq. (4.16). The highest order SSM-coefficients are then found using the `lsqminnorm` function, which is good at dealing with sparse coefficient matrices. If there are no near inner resonances then the reduced dynamics are zero at this order.

Listing 5: `ReducedDynamics`

```
1  [U,Sigma,¬] = svd(full(Ck));
2  tol         = 1e-3;
3  ref         = abs(Lambda_M(1,1));
4  abstol      = tol*ref;
5
6  if Sigma(end,end)/ref < tol
7      innerresonance = true;
8  else
9      innerresonance = false;
10 end
11
12 if innerresonance
13     V       = U(:,diag(Sigma)/ref < tol);
14     Skron   = kron(eye(z_k),B*S{1});
15     Rk      = V'*Skron \ V'* RHS;
16     Sk      = lsqminnorm(Ck,RHS- Skron*Rk,abstol);
17 else
18     Rk = zeros(l*z_k,1);
19     Sk = lsqminnorm(Ck,RHS);
20 end
```

### 4.3.3 Coefficients of composition

`CompositionCoefficients` takes as input

- $k$, order at which SSM-coefficients are calculated

- $l$, the SSM dimension
- $z_k$, the amount of multi-indices that exist for order $k$ and size $l$
- $N$, the dimension of the phase space
- `S`, the SSM-coefficients cell array, or cell array containing in its entries the coefficients of another power series that is to be multiplied with itself, in the same assembly as the SSM-coefficients are put in
- $\mathbf{K}$, containing in its columns the set of multi-indices of order $k$ and size $l$ in reverse-lexicographical ordering
- `H`, the cell array for composition-coefficients, containing $\mathtt{H}_u$ in position $u$

The composition-coefficients of a power series are recursively defined as mentioned in chapter 2.3.
In line 5 all minimal nonzero entries of the order $k$ multi-indices and their positions are computed. For all lower orders $u < k$ the set of multi-indices of order $k - u$ is created in line 8. Then in line 9 all valid combinations of vectors in $\mathbf{K}$ minus all vectors in the columns of `g` are calculated, and the index arrays `I_k`, `I_g` correspond to the respective index of the multi-index in `K` and `g` that give a multi-index in `u_I`.

For example: `u_I(:,3) = K(:,I_k(3))-g(:,I_g(3))`.

In line 17 a linear index array is constructed, to read out the entries $u_j$ in Eq. (3.18) for all multi-indices at order $u$. For every multi-index in `K(:,I_k)` the value of the corresponding multi-index in `gmink` is read out at the position where the multi-index in `K(:,I_kf)` has its minimal non zero value. We name the total number of columns in `gmink` $u_\alpha$. In line 18 the contributions are put together. Here `S{u}(:,u_I)` contains $N$ rows and $u_\alpha$ columns, and `u_j` contains `sz` entries. `H{k-u}` is a matrix containing the coefficients $H_{i,s,\mathbf{g}}$ for all $\mathbf{g}$, $|\mathbf{g}| = k - u$, $s = 1, .., k - u$ and $i = 1, .., N$.

The resulting array `coeff` in line 20 contains $N \times$ `sz` $\times\, k - 1$ entries. In every slice of the third dimension given by `coeff(:,:,s-1)` it contains $u_j S_{i,\mathbf{u}} H_{i,s-1,\mathbf{k}-\mathbf{u}}$, for all the order $k - u$ multi-indices in `gmink`. The first dimension corresponds to the indices $i$, the second dimension to all the multi-indices in `gmink`. In order to get the sum over all terms in Eq. (3.18) for the order `u` multi-indices that correspond to a multi-index $\mathbf{k}_f$ the loop in line 20 is executed over unique indices $f$ in $\mathtt{I}_k$, so all contributions corresponding to a the $f$th multi-index of order $k$ are added up. Finally in line 26 the minimal nonzero values of the order $k$ multi-indices and the scalar exponents $s$ are multiplied with the matrix containing the calculated coefficients, such that the result gives all composition-coefficients $H_{i,s,\mathbf{k}_f} \forall i = 1, .., N; s = 2, .., k; f = 1, .., z_k$ The coefficients for $s = 1$

are precisely the highest order coefficients of the power series, which in the case of the SSM still have to be computed.

Listing 6: `CompositionCoefficients`

```
function[Hk] = CompositionCoefficients(k,l,z_k,N,S,K,H)
% This function computes the coefficients for the composition of ...
    a multivariate polynomial power series.

Hk = zeros(N,z_k,k);
[kj,Ikj] = min(K +(k+1)*(K ==0)); %find nonzero minima of all ...
    multi-indices of order k
for u = 1:k-1
    %order k-u multi-indices in rev.-lex. order
    g       = flip(sortrows(nsumk(l,k-u,'nonnegative')).',2);
    [gmink,I_k,I_g] = gminusk(g,K);
    u_I     = Multi2Index(gmink,'antilex');
    % positions of minimal entries
    pos     = Ikj(I_k);
    sz      = size(I_k,2);
    poslin  = sub2ind([l,sz],pos,1:sz); %create linear index array
    % read out minimal entries
    u_j     = gmink(poslin);

    coeff   = (S{u}(:,u_I).* u_j) .* H{k-u}(:,I_g,:);

    for f = unique(I_k)
        Hk(:,f,2:(k-u+1)) = Hk(:,f,2:(k-u+1)) + ...
            sum(coeff(:,I_k==f,:),2);
    end
end
s  = 1:k;
%match dimensions for pointwse multiplication
Hk = permute(Hk,[2,3,1]) .* ((1./kj).' * s);
Hk = permute(Hk,[3,1,2]);
end
```

## 4.4 Additional functions

The functions treated in this chapter offer a set of functionalities that are necessary to solve the invariance equation in the multi-index form.

### 4.4.1 Multi2Index

Let $\mathbf{K}$ be a matrix in $\mathbb{N}^{l \times m_k}$ containing multi-index vectors $\mathbf{k}_f$ in every column $f$. The individual $\mathbf{k}_f$ satisfy $||\mathbf{k}_f||_1 = k$ and are ordered in lexicographic order, meaning they are ordered with increasing value of their first entry. If two entries are the same, the second entry decides the order and so forth.

To find $\mathbf{k}_f$ knowing $f$ and vice versa a bijective map is needed. The idea of the map is based on finding the amount of combinations possible to build a vector of length $l$ and order $k$. Firstly let us see an example, which will help formalise the idea properly.

**Example 2.** *Suppose now we have a vector $\mathbf{k}_f = [k_{f,1}, \ldots, k_{f,l}]^T$ and want to find its position in $\mathbf{K}$. The first index $k_{f,1}$ already gives a lower bound $I_1$ for the position $f$, namely the following: The first column that has the same entry $k_{f,1}$ appears after all columns containing first entries lower than $k_{f,1}$. The amount of columns that correspond to a certain value $i$ in its first position is given by the amount of possibilities to distribute $k-i$ identical elements to $l-1$ identical slots. This number is given by $\binom{(k-i)+(l-1)-1}{(l-1)-1}$. Therefore*

$$I_1 = \sum_{i=0}^{\mathbf{k}_f[1]-1>0} \binom{(k-i)+(l-1)-1}{(l-1)-1} \tag{4.17}$$

*This can now be used for the second entry of $\mathbf{k}_f$ as well, obtaining another lower bound $I_2$. Now we distribute the elements to $l-2$ slots and the role of $k$ is taken by $k - k_{f,1}$. Recursively applied for all rows in $\mathbf{k}_f$ except for the last one and adding up the $I_j$'s for $j = 1, .., l-1$ this gives exactly the position of the vector in the matrix $\mathbf{K}$.*

**Index** Let $k, l \in \mathbb{N}$ be fixed numbers and $\mathbf{k}_f \in \mathbb{N}^l$, such that $||\mathbf{k}_f||_1 = k$ . Then $f$ is explicitly given by

$$f = 1 + \sum_{j=1}^{l-1} I_j \tag{4.18}$$

where $I_1$ is the same as in Eq. (4.17) and for $j \neq 1$ it is

$$I_j = \sum_{i=0}^{(\mathbf{k}_f)_j-1>0} \binom{((k-(\mathbf{k}_f)_{j-1})-i)+(l-j)-1}{(l-j)-1} \tag{4.19}$$

Note that if for some $j$ we have $(\mathbf{k}_f)_j = 0$, then $b_j = 0$. Using this, the following function is implemented:

**Function** `Multi2Index(m,ordering)` takes as an input a multi-index array `m` and calculates the position of all multi-indices in `m` in an ordered set of multi-indices of same order and size. The option `ordering` can be set to `'lex'` or `'antilex'` specifying the ordering of that set.

The loop in line 11 loops over all multi-indices in the input-array. In lines 8-12 the calculation of Eq. (4.17) is carried out. Then in lines 14-22 the same calculation is done for all the other components of the multi-index, the mathematical formulation of the calculated terms is given in Eq. (4.19). This is done for all multi-indices in the input array. Then in lines 32-37 the index is mirrored with respect to the centre of the array containing the whole ordered set, if the ordering is reverse-lexicographical. If the multi-indices in `m` do not all have the same order, they have to be mirrored about the centre of the set corresponding to their order individually, which is done in line 36.

Listing 7: `Multi2Index`

```
1  function [I] = Multi2Index(m,ordering)
2  % Given a multi-index array containing multi-indices in its ...
       columns this function computes the position of the ...
       multi-indices in a ordered set of all multi-index vectors ...
       that have the same order as each of the multi-indices. The ...
       ordering can be 'lex' for lexicographical or 'antilex for ...
       reverse-lexicographical
3  if isempty(m)
4      I = [];
5  else
6      l    = size(m,1);
7      sz   = size(m,2);
8      mabs = sum(m,1);
9      I    = ones(1,sz);
10     for ij  = 1:sz
11         bij = 0;
12         if m(1,ij) ≠ 0
13             for i = 0:m(1,ij)-1
14                 bij = bij+ nchoosek(mabs(ij)-i+l-1-1,l-1-1);
15             end
16         end
17         for j = 2:(l-1)
18             if m(j,ij) ≠ 0
19                 for i  = 0:(m(j,ij)-1)
20                     bij = bij+ ...
                           nchoosek(mabs(ij)-sum(m(1:j-1,ij)) ...
21                         -i+l-j-1,l-j-1);
22                 end
23             end
24         end
25         I(ij) = bij+1;
26     end
27 
28     %if order is reverse-lexicographical, then the position is
29     % the mirrored value about the centre of the array.
30     switch ordering
31         case 'antilex'
32             if all(mabs == mabs(1))
33                 I = nchoosek(mabs(1)+l-1,l-1)+1-I;
34             else
35                 for i = 1:sz
36                     I(i) = nchoosek(mabs(i)+l-1,l-1)+1-I(i);
37                 end
38             end
39     end
40 end
41 end
```

### 4.4.2 nsumkvector

`nsumkvector(n,kf,option)` Given a number $n$ and a positive integer vector $\mathbf{k}_f$ this function returns all combinations of $n$ positive integer vectors adding up to $\mathbf{k}_f$.

Let $\mathbf{k}_f \in \mathbb{N}^l$. this function finds all combinations consisting of $n$ positive integer vectors such that they sum up to $\mathbf{v}$: $\sum_{i=1}^n \mathbf{g}_i = \mathbf{v}$ All of those combinations are stored as 2D arrays, where the individual columns are filled with the vectors, in column $i$ the vector $\mathbf{g}_i$ is stored. The various combinations are stacked upon each other in a 3D array, which contains a third dimension with size equal to the number of combinations summing up to $\mathbf{v}$. The option `'abs'` allows to also get an array that contains the absolute values of all the $\mathbf{g}_i$'s, each combination of them being in one column. So for example for the first combination, the absolute value of $\mathbf{g}_2$ would be stored in the output `giabs(2,1)`.

The idea is to use the function `nsumk` to find all combinations of $n$ numbers summing up to every entry of the input array $\mathbf{v}$. In lines 5-14 exactly this is done. `S` now contains `l` arrays. In each array contained in `S{i}`, each column corresponds to a combination of `n` numbers adding up to `v(i)`. Then in line 15 all combinations of columns of all arrays in `S` are stacked onto each other.
For example if `S` contains two arrays, i.e. $l = 2$, and $S_1 :=$ `S{1}` contains two columns, $S_2 :=$ `S{2}` contains three columns. Then `gi` contains six columns, and $2n$ rows. $\mathtt{S}_1(:,1)$ contains a combination of $n$ numbers adding up to the first element of $\mathbf{v}$. `gi` looks like

$$\mathtt{gi} = \begin{pmatrix} \mathtt{S}_1(:,1) & \mathtt{S}_1(:,1) & \mathtt{S}_1(:,1) & \mathtt{S}_1(:,2) & \mathtt{S}_1(:,2) & \mathtt{S}_1(:,2) \\ \mathtt{S}_2(:,1) & \mathtt{S}_2(:,2) & \mathtt{S}_2(:,3) & \mathtt{S}_2(:,1) & \mathtt{S}_2(:,2) & \mathtt{S}_2(:,3) \end{pmatrix} \tag{4.20}$$

Now all the elements of a combination of vectors are stored in the columns of `gi` in the following order: the first `n` rows contain the first element of each of the vectors of this combination. The second `n` rows then contain the second elements of all the vectors of one combination. So each column has to be reshaped to size `(n,l)` and then transposed to obtain $n$ vectors summing up to $\mathbf{v}$. The resulting matrices then are stacked up onto each other. This is done in lines 16-17.

Listing 8: `nsumkvector`

```
1  function [gi,giabs] = nsumkvector(n,v,option)
2
3  l = size(v,1);
4  S = {};
5  for i = 1:l
6      %special case
7      if v(i) == 1 & n == 1
8          S{i} = 1;
9      else
```

```
        S{i} = nsumk(n,v(i),'nonnegative').' ;
        %finds all combinations for each index in v
    end
end

gi = combvec(S{:});
gi = reshape(gi,n,1,[]);
gi = permute(gi,[2 1 3]);

switch option
    case 'abs'
    giabs = reshape(sum(gi,1),n,[]);
end
end
```

### 4.4.3 gminusk

This function takes two arrays with the same amount of rows as input and then subtracts all columns of the second array `k` from all the columns of the first array `g`. It then deletes all columns that contain negative numbers. The output is an array `S` with all these columns, `I_k` which contains in position $i$ the column index of the column in `k` that the $i^{th}$column of `S` corresponds to and analogously for `I_g` and `g`.

Listing 9: `gminusk`

```
function [S,I_k,I_g] = gminusk(g,k)
k_l = size(k,2);
g_l = size(g,2);
S   = cell(1,g_l);
I_k = cell(1,g_l);
I_g = cell(1,g_l);

for i = 1:g_l
    S{i}   = k - g(:,i);
    I_k{i} = 1:k_l;
    I_g{i} = ones(1,k_l)*i;
end
S   = cell2mat(S);
I_k = cell2mat(I_k);
I_g = cell2mat(I_g);

[¬,neg]     = find(S<0);
S(:,neg)    = []; %delete columns with negative entries
I_g(:,neg)  = [];
I_k(:,neg)  = [];
end
```

# 5 Results

To carry out simulations a Macbook Air 6.2 is used. It is equipped with a `1.4 GHz Intel Core i5` processor and the memory specifications are `4 GB 1600 MHz DDR3`.

## 5.1 Comparing SSM-coefficients

### 5.1.1 Modified Shaw-Pierre example

Similarly to Ponsioen Pedergnana and Haller (2019)[5] we treat a modified example of Shaw and Pierre (1994) [6], by adding a cubic nonlinear damper as in Habib, Cirillo and Kerschen (2017)[7]. The mathematical setting of the equations of motion is as follows: $\mathbf{B}\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{F}(\mathbf{z})$ is the equation of motion, where we define $\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}$. This requires

$$\mathbf{B} = \mathbf{I}_4, \quad \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_2 \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix}, \quad \mathbf{F}(\mathbf{z}) = \begin{bmatrix} \mathbf{0} \\ -\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) \end{bmatrix} \tag{5.1}$$

The following definitions are used.

- $\mathbf{M} = \begin{pmatrix} m, & 0 \\ 0, & m \end{pmatrix}$ is the mass matrix, both particles have mass $m$ and the mass matrix is diagonal in this example
- $\mathbf{K} = \begin{pmatrix} 2k, & -k \\ -k, & 2k \end{pmatrix}$ is the linear stiffness matrix with stiffness coefficient $k$
- $\mathbf{C} = \begin{pmatrix} c, & -c \\ -c, & 2c \end{pmatrix}$ is the linear damping matrix, with linear damping coefficient $c$
- as nonlinear force we choose $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) = \begin{pmatrix} \gamma x_1^3 \\ 0 \end{pmatrix}$, which is autonomous in contrast to the example in Ponsioen's work

The values of the parameters used to define those matrices can be seen in Table 1.

| Parameter | Value | Meaning |
|---|---|---|
| m | 1 kg | Mass of oscillators |
| $c_a$,$c_b$ | $10^{-3}$,1 Nm/s | Linear damping coefficients |
| k | 1 N/m | Linear spring constant |
| $\gamma$ | 0.5 N/m$^3$ | Cubic spring constant |

Table 1: Parameter Values for the modified Shaw-Pierre example with low and high linear damping.

The comparison is made for two different values of $c$. In order to compare the coefficients obtained for both implementations a norm is introduced. Let $\mathtt{S}_{k,m}$ and $\mathtt{S}_{k,t}$ be the SSM-coefficients obtained at order $k$ using multi-index and the tensor-implementation respectively, both given in multi-index format. The arrays containgin these indices at order $k$ then have dimensions ($\mathtt{N}$,$\mathtt{z}_k$). We define a distance function between the calculated coefficients as follows:

$$D_S(k) := \frac{1}{z_k}\sum_{i=1}^{N}\frac{\sum_{j=1}^{z_k}|(\mathtt{S}_{k,m})_{ij} - (\mathtt{S}_{k,t})_{ij}|}{\max_j(|(\mathtt{S}_{k,m})_{ij}|, |(\mathtt{S}_{k,t})_{ij})|)} \tag{5.2}$$

And analogously $D_R(k)$ is defined for the reduced dynamics coefficients. This is a sensible definition, since for every basis-vector direction in the phase space, it norms the total difference of the SSM-coefficients in this direction with the biggest coefficient of the SSM parametrisation that contributes to this direction. So the bigger the difference in the coefficients is, the more they contribute relative to the biggest coefficient in the direction they correspond to. If the maximal absolute value of coefficients at an order in a row is 0 then the contribution of this row to the norm is set to zero. The idea behind this distance function is to compare the differences in the coefficients with the impact that they would have on the parametrisation of the SSM and on the reduced dynamics.

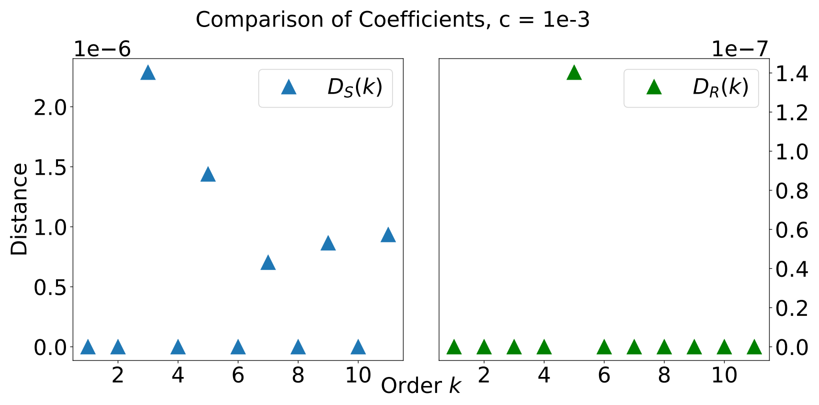

Figure 1: On the left side the distance function $D_S(k)$ for the SSM-coefficients is plotted, in the case where the linear damping is given by $c_a$. The maximal distance is of order $10^{-6}$. On the right side the same can be seen for the reduced dynamics where the maximal distance is one order of magnitude smaller.

In order to make the comparison meaningful, an implementation of the tensor-version using the same method to do resonance analysis as the one described in section 4.3.2 has been used.

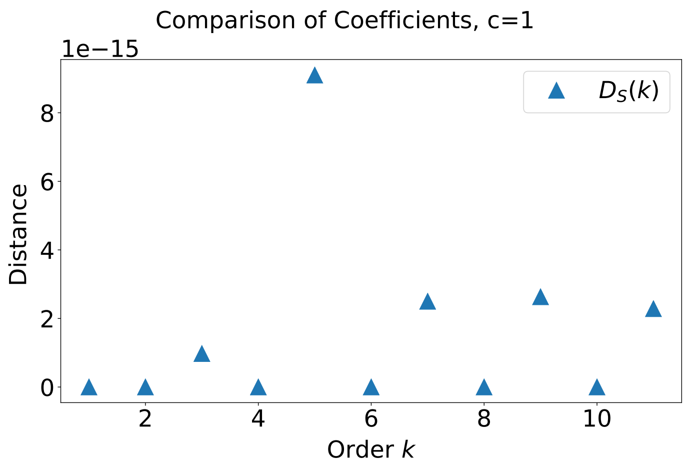

Figure 2: The distance function $D_S(k)$ for the SSM-coefficients is plotted, in the case where the linear damping is given by $c_b = 1$. This is of the order of machine accuracy for doubles in matlab and it can be assumed, that the errors in this scenario arise from computational errors [8]. The reduced dynamics coefficients are zero in this case, since high damping does not allow for near inner resonances.

Figure 1 shows that for the case of near inner resonances the distance of the coefficients is of $O(10^{-6})$. For the case where the reduced dynamics is linear the errors are substantially smaller, namely by 10 orders of magnitude as can be seen in Figure 2. Therefore it can be assumed, that the errors occurring for the case with damping coefficient $c_a$ result from the method that is used to determine the reduced dynamics. Presumably the singular value decomposition in line 1 of the listing `ReducedDynamics` introduces those errors.

### 5.1.2 Nonlinear Bernoulli beam

In this section we seek to verify the results obtained by Ponsioen, Pedergnana and Haller (2019)[5] for a discretized Bernoulli beam with a cubic spring attached to its last node that has cubic damping. We only consider the autonomous SSM-coefficients. For more details about the system setup, the reader is referred to [4] and [5].

**Setup**
For the calculations to compare the coefficients we shall follow the same path as in the reference example an treat the system in diagonalised form. The initial equations of motion read $\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{X}\mathbf{F}(\mathbf{z})$ with $\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}$. $\mathbf{x}$ is a vector in

$\mathbb{R}^n$. Every discrete element has a translatory and rotatroy degree of freedom. Therefore the phase space will be $4m = 2n$ dimensional for $m$ discrete elements. The system matrices and the force are given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_2 \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix}, \quad \mathbf{F}(\mathbf{z}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) \end{bmatrix}, \mathbf{X}(\mathbf{z}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{M}^{-1} \end{bmatrix} \tag{5.3}$$

The system is then diagonalised by defining diagonal coordinates $\mathbf{q}$ such that $\mathbf{z} = \mathbf{Tq}$. The matrix $\mathbf{T}$ diagonalises $\mathbf{A}$ such that $\mathbf{T^{-1}AT} = \mathbf{\Lambda}$ which contains the eigenvalues of $\mathbf{A}$ on its diagonal, ordered such that their real parts decrease. The equations of motion then transform to $\dot{\mathbf{q}} = \mathbf{\Lambda z} + \mathbf{T^{-1}F}(\mathbf{Tq})$. The chosen parameters can be seen in Table 2.

| Parameter | Value | Meaning |
|---|---|---|
| $L$ | 2700 mm | Length of beam |
| $h$ | 10 mm | Height of beam |
| $b$ | 10 mm | Width of beam |
| $\rho$ | $1780 \times 10^{-9}$ kg/mm$^3$ | Density |
| $E$ | $45 \times 10^6$ kPa | Young's Modulus |
| $I$ | 833.3 mm$^4$ | Area moment of inertia |
| $\kappa$ | 6 mN/mm$^3$ | Coefficient cubic spring |
| $\gamma$ | $-0.02$ mN s/mm$^3$ | Coefficient cubic damper |
| $A$ | 100 mm$^2$ | Cross section of beam |
| $\alpha$ | $1.25 \times 10^{-4}$ s$^{-1}$ | Structural damping parameter |
| $\beta$ | $2.5 \times 10^{-4}$ s | Structural damping parameter |
| $m$ | 25 | Number of discrete elements |

Table 2: Parameter Values for the Bernoulli beam. The system is only damped linearly.

**Transformation of the Force**

The force due to the coupling between discrete elements of the non diagonalised system only contains one nonlinear component:

$$\mathbf{f}_s(\mathbf{x}, \dot{\mathbf{x}}) = \begin{pmatrix} 0 \\ \vdots \\ \kappa\, x_{n-1}^3 \\ 0 \end{pmatrix} \tag{5.4}$$

Writing this part of the force as a Taylor expansion in multi-index notation we therefore obtain $\mathbf{F}_s(\mathbf{z}) = \mathbf{F}_{\mathbf{m}_1,s}\mathbf{z}^{\mathbf{m}_1}$, where $\mathbf{m}_1 \in \mathbf{N}^{2n} = (0,..,0, \underbrace{3}_{n-1^{th}\text{ entry}}, 0,..,0)^T$

and $\mathbf{F}_{\mathbf{m}_1,s} = (0,..,0,\underbrace{\kappa}_{2n-1^{th}\text{ entry}},0)^T$. Transforming this into diagonalised coordinates we have to calculate $x_{n-1}^3$ given in the new coordinates. Using the multinomial coefficient this can be written as

$$(\sum_{j=1}^{2n}(\mathbf{T})_{n-1,j}(\mathbf{q})_j)^3 = \sum_{\substack{\mathbf{v}\in\mathbb{N}^{2n}\\|\mathbf{v}|=3}} \binom{3}{\mathbf{v}} \prod_{j=1}^{2n} \left((\mathbf{T})_{n-1,j}(\mathbf{q})_j\right)^{(\mathbf{v})_j} \tag{5.5}$$

We define vectors $\tilde{\mathbf{F}}_{\mathbf{v},s} = \mathbf{F}_{\mathbf{m}_1,s}\binom{3}{\mathbf{v}}\prod_{j=1}^{2n}\left((\mathbf{T})_{n-1,j}(\mathbf{q})_j\right)^{(\mathbf{v})_j}$. They are exactly the vectors that would be obtained by Taylor-expanding the force in the diagonalised coordinates. They correspond to the multi-index $\mathbf{v}$. Thus the force in the new coordinates is looks like

$$\mathbf{F}_s(\mathbf{Tq}) = \sum_{\substack{\mathbf{v}\in\mathbb{N}^{2n}\\|\mathbf{v}|=3}} \tilde{\mathbf{F}}_{\mathbf{v},s}\mathbf{q}^{\mathbf{v}} \tag{5.6}$$

The same can be done for the cubic damping, with $\mathbf{F}_d(\mathbf{z}) = \mathbf{F}_{\mathbf{m}_2,d}\mathbf{z}^{\mathbf{m}_2}$, where $\mathbf{m}_2 \in \mathbf{N}^{2n} = (0,..,0,\underbrace{3}_{2n-1^{th}\text{ entry}},0)^T$ and $\mathbf{F}_{\mathbf{m}_2,d} = (0,..,0,\underbrace{\gamma}_{2n-1^{th}\text{ entry}},0)^T$.

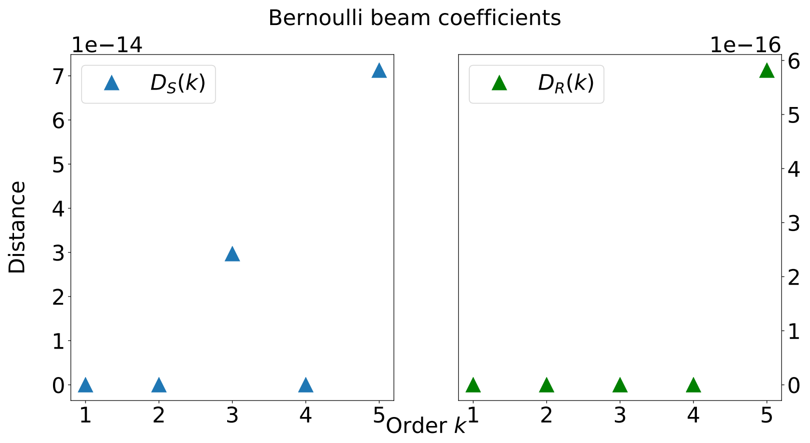

Figure 3: This figure compares the SSM-coefficients (left side) and the reduced dynamics coefficients (right side) for the nonlinear beam example. The phase space size for this simulations is $N = 100$. The coefficients coincide up to the order of working precision as in the previous case of the modified Shaw-Pierre example without inner resonances. Here however there appear inner-resonances for the third and fifth order computations.

**Comparison**
For the resonance analysis in listing 5 the parameter `tol` is adjusted to $10^{-2}$. Using the same definition of distance between SSM-coefficients as in section 5.1.1 the results obtained by Ponsioen and the coefficients obtained with our implementation are compared. The resulting differences are displayed in Figure 3.

**Force response curve**
In their work Ponsioen, Pedergnana and Haller show that the force response curve of a system can be obtained using the coefficients of the reduced dynamics [5]. By changing the parametrised euclidean coordinates to polar coordinates, the force response curve of externally forced systems can be obtained as a zero level set of a function $G$, dependent on the external forcing frequency. This corresponds to a coordinate change $\mathbf{p} = (p_1, p_2)^T \mapsto (\rho, \theta)$. It is explicitly given in section 6.2.

We now consider external periodic forcing of the form $\epsilon P \cos \Omega t$ that acts on the last element of the beam. The system equation then looks like

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{X}\mathbf{F}(\mathbf{z}) + \epsilon \mathbf{X}\mathbf{P} \cos \Omega t \tag{5.7}$$

with $\mathbf{P} = (0, .., P, 0)^T \in \mathbb{R}^{2n}$. We choose $P = 0.1$ [mN] and define $\phi$ via $\dot{\phi} := \Omega$. This system will not be diagonalised, for the purpose of demonstrating that our method also leads to the force response curve (FRC) without performing a change of coordinates and thereby destroying the sparsity of the nonlinearity. We can then rewrite this equation in terms of the SSM-parametrisation and the reduced dynamics. However now the coefficients will not be autonomous anymore. We consider a Taylor-expansion of the two functions $\mathbf{R}$ and $\mathbf{S}$ in $\epsilon$:

$$\mathbf{S}(\mathbf{p},\phi) = \mathbf{S}_0(\mathbf{p},\phi) + \epsilon \mathbf{S}_1(\mathbf{p},\phi) + \mathrm{O}(\epsilon^2) \tag{5.8}$$

$$\mathbf{R}(\mathbf{p},\phi) = \mathbf{R}_0(\mathbf{p},\phi) + \epsilon \mathbf{R}_1(\mathbf{p},\phi) + \mathrm{O}(\epsilon^2) \tag{5.9}$$

Each of those functions can be expanded as a power series using multi-indices, where the coefficients now show time dependence. At $O(0)$ this leads to Eq. 3.29. Therefore the time dependence for $\mathbf{S}_0$ and $\mathbf{R}_0$ drops. The reduced dynamics at first order in epsilon and zeroth multi-index order are then given as [5]

$$R_{i,\mathbf{0}} = c_{i,\mathbf{0}} e^{i\Omega t} + d_{i,\mathbf{0}} e^{-i\Omega t} \tag{5.10}$$

The following result that we introduce is current work by Jain Shobhit, and will be useful when solving for the zero level sets of $G$. Let $\mathbf{w}, \overline{\mathbf{w}}$ be the vectors that satisfy $\mathbf{w}^H \cdot (\mathbf{A} - \lambda_1 \mathbf{1}) = 0$ for the master mode eigenvalue $\lambda_1$ and the same with $\overline{\mathbf{w}}$ and $\lambda_2$. At first order in $\epsilon$ and zeroth multi-index order the reduced dynamics coefficients of the non diagonalised system can be chosen as

$$R_{1,\mathbf{0}} = \mathbf{w}^H \cdot \frac{\mathbf{XP}}{2} e^{i\Omega t} \tag{5.11}$$

$$R_{2,\mathbf{0}} = \overline{\mathbf{w}}^H \cdot \frac{\overline{\mathbf{XP}}}{2} e^{-i\Omega t} \tag{5.12}$$

This implies $c_{1,\mathbf{0}} = \mathbf{w}^H \cdot \frac{\mathbf{XP}}{2}$. We also know the autonomous reduced dynamics coefficients which are used to define the functions $a(\rho)$ and $b(\rho)$. Now we can use the definition of $G$ to obtain the force response curve. The obtained force response curve is displayed for two different values of $\epsilon$ in Figures 4 and 5. The force response curve can have isolated regions that do not connect to the main branch of it, called isolas. Under certain conditions on the reduced dynamics it can be guaranteed that such isolas exist, and perturb smoothly from the main branch of the force response curve at approximately the position of the backbone curve. The approximate value of $\epsilon$ when they merge with the main branch is given by [5]

$$\epsilon_m = \frac{1}{||c_{1,\mathbf{0}}||} \sqrt{\frac{4|\Re(\lambda_1)|^3}{27\Re(\gamma_1)}} \tag{5.13}$$

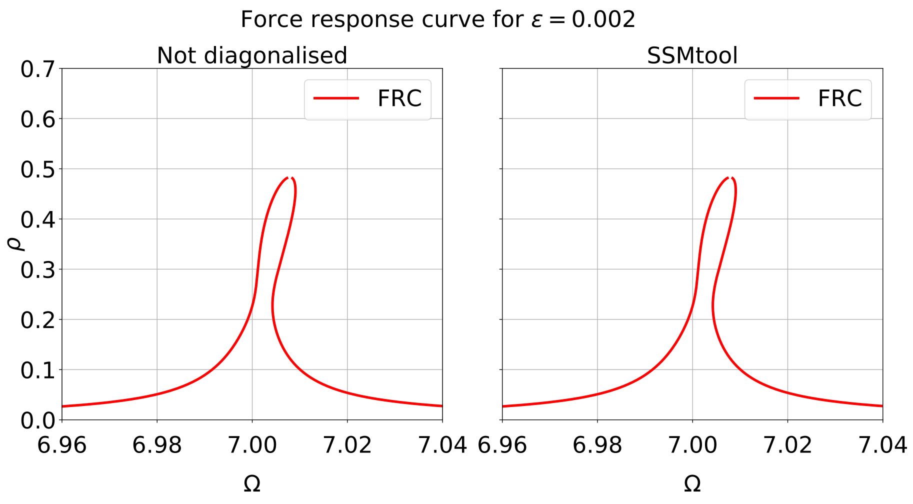

Figure 4: This figure shows the force response curve for $\epsilon = 0.002$. It compares the force response curve that we obtain without diagonalising the system and the FRC obtained by using the public library `SSMtool` which works with the diagonalised system. They can be seen to be identical [9].

The parameter $\gamma_1$ is the third order autonomous coefficient of the reduced dynamics. For the Bernoulli beam where $\epsilon_m \approx 0.0018$ we try to confirm this behaviour and plot the FRC for $\epsilon = 0.002$ and $\epsilon = 0.0016$. The resulting curves are compared to the ones obtained by using the library `SSMtool` [9].

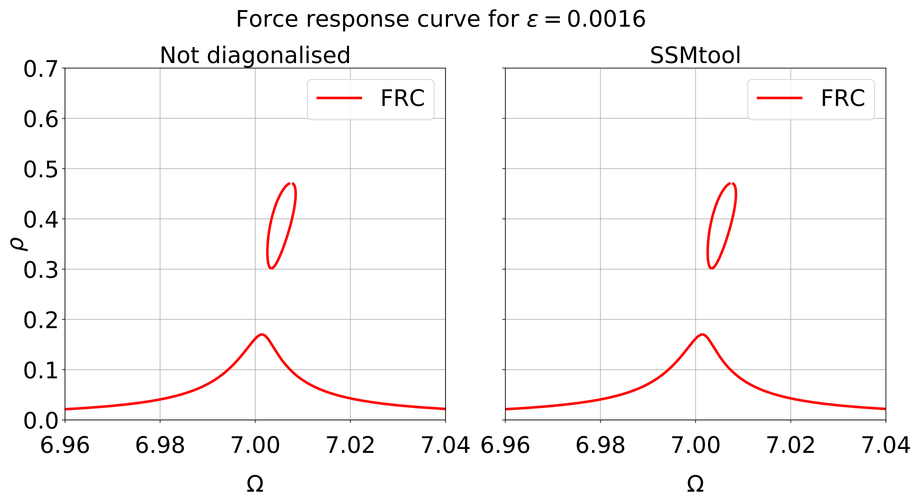

Figure 5: This figure shows the force response curves obtained for $\epsilon = 0.0016$. This value is below the theoretically determined threshold for merging isolas which lies at $\epsilon_m \approx 0.0018$ [5]. Therefore it is expected, that there exist isolated areas of the force response curve. It can be seen that our implementation also correctly predicts those.

## 5.2 Theoretical predictions for storing the SSM-coefficients

For a given size of the Phase space $N$, dimension of the SSM $l$ and order $k$ the number of coefficients to all the multivariate monomials describing the SSM at this order is given by

$$f(k,l,N) = \texttt{nchoosek}(k+l-1, l-1) * N \tag{5.14}$$

Since each coefficient is a complex number, which require 16 bytes of storage each, the total storage required to store all coefficients describing the multinomial expansion of $\mathbf{S}(\mathbf{p})$ at order $k$ is given by $f(k,l,N) * 16$.
For simplicity we define $\texttt{C(S\{k\})}_m$, the actual amount of memory used by Storing the coefficients of $\mathbf{S}(\mathbf{z})$ at order $k$. The subindex can be either $_m$ or $_t$ which denotes that the coefficients were computed and are stored using the multi-index and tensor-implementation respectively.

## 5.3 Comparison of performance

### 5.3.1 The modified Shaw-Pierre example

The setting for this example is the same as in section 5.1.1. This example shall serve to see how the Multi-index treatment manages to deal with higher order expansions of the SSM in comparison with the tensor-implementation. For the analysis of performance related aspects, the newer version of the tensor implementation that directly determines the vectors spanning the kernel is taken as reference.

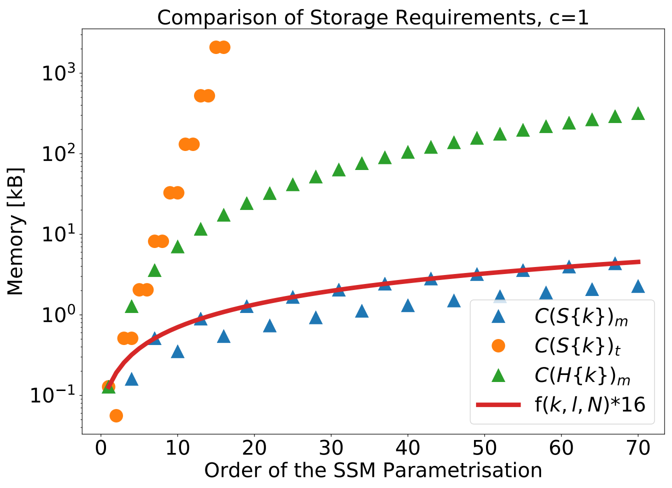

Figure 6: While the memory requirements to store coefficients using the tensor-implementation grows exponentially with increasing order of the expansion, In contrast, the total size of the SSM-coefficients plus the composition-coefficients only grow linearly. Since there are no inner resonances the reduced dynamics are zero for all orders. Also, only every third value is plotted for the quantities calculated with the multi-index routine, his serves to make the graph more clear.

**High Damping** We choose $c = 1$ as a damping parameter, which leads to the absence of near inner resonances. All other parameters are chosen as in Table 1. The storage, that is used in the multi-index-version of the calculation differs in one qualitative aspect. Not only does one need to store the coefficients for the reduced dynamics and the SSM parametrisation, but also the coefficients for the composition of the power-series of $\mathbf{S}(\mathbf{z})$, stored in $\mathtt{H}_k$ at order $k$.
In Figure 6 the memory requirements of the SSM-coefficients stored in tensor-format grow in pairs. This is due to the fact, that for even orders the SSM-coefficients are zero in this example, which means they only need half the space of conplex numbers. The behaviour of the multi-index coefficients that require less memory than the theoretical bound do so for the same reason.

The linear growth of the memory requirements for the coefficients in multi-index format is a property inherent only to two-dimensional SSM's. The amount of multi-indices at order $k$ is then given by

$$f(k,2,N)/N = \frac{(k+1)!}{1!(k)!} = k+1 \tag{5.15}$$

which grows linearly with increasing order. For bigger $l$ the growth will be polynomial with increasing order.

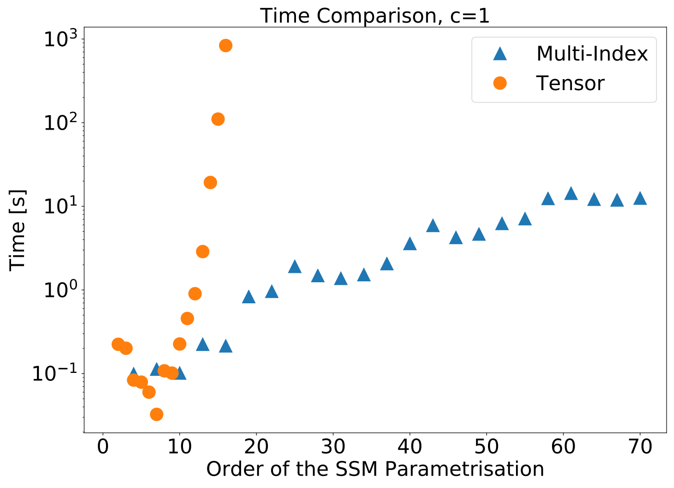

Figure 7: This figure shows how long the calculation of given orders take, and compares the tensor and multi-index calculations. While the tensor-version takes almost $10^3$ seconds for the coefficients at 16th order, the multi-index-version takes less than $10^2$ seconds to compute the coefficients at order 70. Again for the multi-index calculations only the data for every third order is plotted in order to make the plot look more clear.

**Low Damping** In this case $c$ is choosen to be small, i.e. $c = 10^{-3}$ Low damping in the modified Shaw-Pierre example leads to near inner resonances. They are dealt with by choosing the reduced dynamics according to section 4.3.2. In Figure 8 the memory requirements of the reduced dynamics and SSM-coefficients are compared. Once again an exponential increase in memory for the tensor-version is observed, while with the multi-index way of Storing the coefficients the theoretical curve is reached for the SSM-coefficients. The datapoints that lie below this curve, come from the fact that for even orders the SSM-coefficients are all zero and only need half the storage space of a complex number.

Figures 7 and 9 show one of the big advantages of the multi-index-version over the tensor-version. Apart from achieving the theoretical limit for storage requirements of the SSM-coefficients, the multi-index-version also is sig-

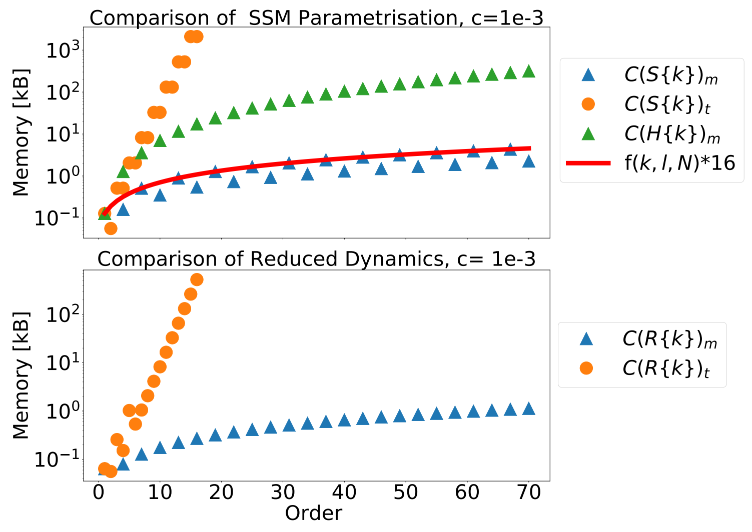

Figure 8: In this figure the amount of memory used for the reduced dynamics, and the SSM-coefficients including the composition-coefficients $\mathtt{H}_k$ is compared for tensor and multi-index-implementation in the case of near inner resonances in the modified Shaw-Pierre example. While the used memory for the tensor-version grows exponentially with increasing order of the SSM expansion, the multi-index-version is seen to grow linearly.

nificantly faster when computing coefficients at higher orders. Even the computation of SSM-coefficients at order $k = 70$ take two orders of magnitudes less time than the computation of the order 16 coefficients calculated with the tensor-implementation.

### 5.3.2 Oscillator chain

A chain of nonlinear oscillators is used to investigate the performance of the multi-index-implementation with respect to the size of a given system. This is compared with the tensor notation as well. Since for the multi-index-version the method of directly extracting the kernel of the coefficient matrices has not been implemented yet, only the case with high damping and thus without near inner resonances is treated. The setup used here to solve $\mathbf{B}\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{F}(\mathbf{z})$ is given by setting $\mathbf{z} = \begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{x} \end{bmatrix}$ which leads to

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} & \mathbf{M} \\ \mathbf{M} & \mathbf{C} \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & -\mathbf{K} \end{bmatrix}, \quad \mathbf{F}(\mathbf{z}) = \begin{bmatrix} \mathbf{0} \\ -\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) \end{bmatrix}. \tag{5.16}$$

Again, the mass matrix is diagonal, and all masses are the same. All springs have the same linear spring constant $k$ and the same linear damping coeffi-

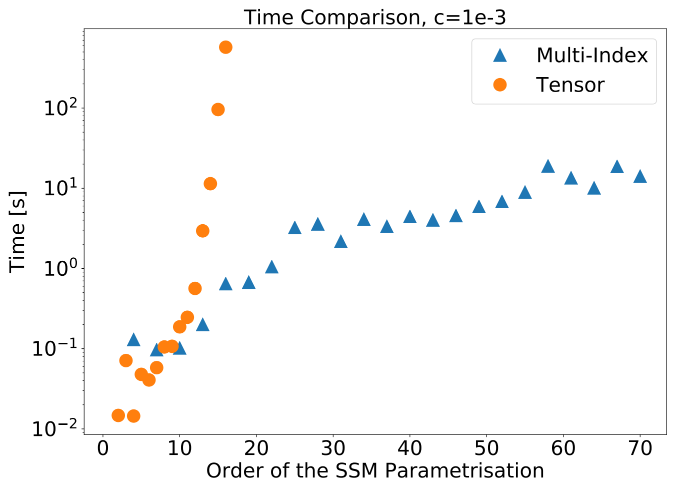

Figure 9: This figure analogously to Figure 7 shows how long the calculation of given orders take, and compares the tensor and multi-index calculations. The two versions of calculating the coefficients in the weakly damped example respectively show the same behaviour as they do in the strongly damped example, where the multi-index-version is faster.

cient is used for every degree of freedom. All springs in the system are also equipped with a quadratic spring constant $\kappa_2$ and a cubic spring constant $\kappa_3$. These contributions amount to the nonlinear force. The first and last spring are assumed to be fixed at a wall. As an example, for a 4 dimensional system, the nonzero multi-index force coefficients at second order then look like

$$\mathbf{f}_2(\mathbf{x}, \dot{\mathbf{x}}) = -\begin{pmatrix} 0 & \kappa_2 & 0 & 0 & -2\kappa_2 & 0 & 0 \\ -\kappa_2 & 0 & \kappa_2 & 0 & 2\kappa_2 & -2\kappa_2 & 0 \\ 0 & -\kappa_2 & 0 & \kappa_2 & 0 & 2\kappa_2 & -2\kappa_2 \\ 0 & 0 & -\kappa_2 & 0 & 0 & 0 & 2\kappa_2 \end{pmatrix} \tag{5.17}$$

which gives $\mathbf{F}_2(\mathbf{z}) = \begin{bmatrix} \mathbf{0} \\ -\mathbf{f}_2(\mathbf{x}, \dot{\mathbf{x}}) \end{bmatrix}$ corresponding to a multi-index array

$$\texttt{Fmulti}\{2\} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 & 1 \end{pmatrix} \tag{5.18}$$

The multi-index-version only needs the columns of $\mathbf{F}(\mathbf{z})$ that contain nonzero elements and their corresponding multi-indices as input. Alternatively an index array can be passed into the function, containing the positions of all those columns in the entire multi-index nonlinearity matrix which contribute, this is explained in detail in section 4.3.2. The advantage of this can be directly seen here, since only very few of all order two multi-indices have contributions in the force, i.e. not all degrees of freedom are coupled.
The chosen values for the parameters can seen in Table 3.

| **Parameter** | **Value** | **Meaning** |
|---|---|---|
| m | 1 kg | Mass of oscillators |
| c | 1 Nm/s | Linear damping coefficient |
| k | 1 N/m | Linear spring constant |
| $\kappa_2$ | 0.5 N/m | Quadratic spring constant |
| $\kappa_3$ | 0.5 N/m | Cubic spring constant |

Table 3: Parameter values for the oscillator chain example.

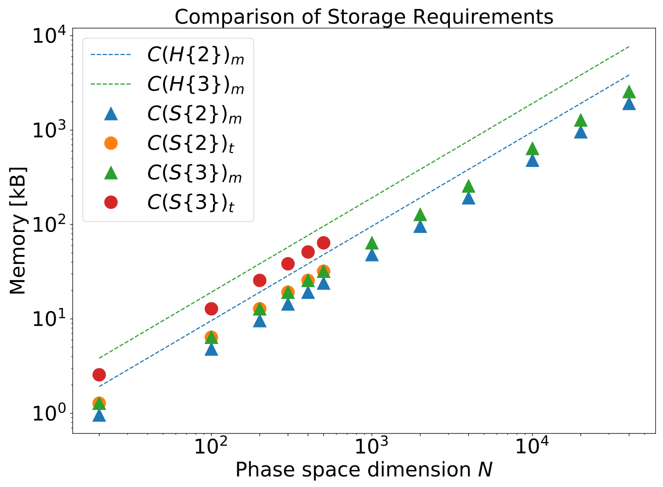

Figure 10: This figure compares the memory requirements for storing the SSM-coefficients at second and third order for different sizes of the phase space. Furthermore, the memory required to store the composition-coefficients is portrayed with dashed lines. One can see, that the memory used for the coefficients using the multi-index notation is lower than storing them in the tensor-format. Since this is a log-log plot, the displacements on the y axis correspond to the logarithms of the slope of the linear growth of the memory requirements. Therefore the requirements also grow slower for the coefficients stored in multi-index format.

Another point which needs to be mentioned is that there appear outer resonances for this example, therefore the existence of the SSM is not guaranteed, as shown by Haller and Ponsioen (2016, p. 1503-1504)[1].

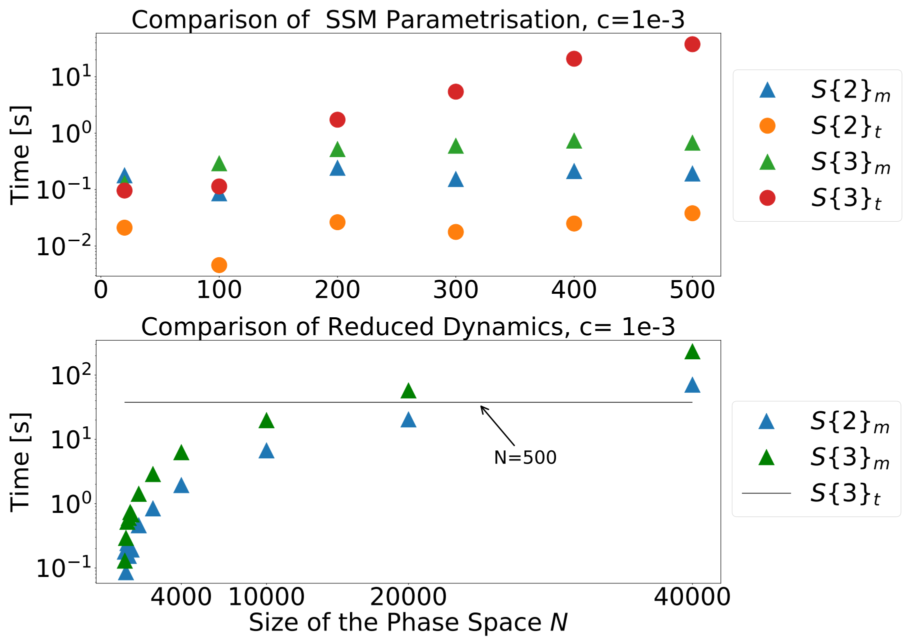

Figure 11: The first plot portrays the time needed to compute the coefficients at a given order, and compares the two different implementations. It can be seen, that the second order SSM-coefficients take less time using the tensor-version while at the third order the opposite is true. The plot on the bottom shows the evolution of the time consumption of the multi-index calculations for bigger system sizes. As a reference, the time needed to compute the third order coefficients with the tensor-version for $n = 250$ is shown.

In Figure 10 the memory requirements are compared as a function of phase space dimension. One notices, that although the SSM-coefficients need less memory in the multi-index format, the storing of the composition-coefficients needs more memory, than storing the SSM-coefficients in tensor-format. However to now simply add the memory of the composition-coefficients to the SSM-coefficients and therefore conclude that more storage is required in the multi-index-version would be a faulty comparison. In the tensor computation the memory usage is hidden in computing Kronecker products, which need big amounts of RAM. This can be clearly seen by the fact, that the laptop was not able to compute the SSM-coefficients for systems with more than $n = 250$ with the tensor-version, however with the multi-index-version even simluating $n = 2 * 10^4$ did not pose any problem.

While in the first part of Figure 11 it seems like there is no advantage with respect to time performance over the tensor-version, the second plot shows more clear results. Here computing the third order coefficients with multi-index notation for $n = 10^4$ takes roughly as much time as the third order coefficients in the tensor-format for $n = 250$. It becomes clear from this plot that for bigger system sizes we expect that the multi-index-version will also outperform the tensor-version in terms of time consumption.

**Profile** This section analyses which parts of the code take most of the time and therefore give insight into which bottlenecks should be tackled next. For the analysis the oscillator chain has been used, taking $n = 5 * 10^4$ and the same parameters as for the preceding analysis, they can be found in Table 3. The second and third order SSM-coefficients have been calculated and the resonance analysis has been left out of the calculation. As can be seen in Table 4, the most

| **Function** | **Calls** | **Time [s]** | **Self Time [s]** |
|---|---|---|---|
| `OscillatorChainMulti` | 1 | 1837.442 | 41.248 |
| `SSMmultiIndex` | 2 | 1790.889 | 0.129 |
| `CoefficientMatrix` | 2 | 1503.178 | 1503.048 |
| `Force` | 2 | 285.627 | 76.211 |
| `nsumkvector` | 1299985 | 146.136 | 32.193 |
| `nchoosek` | 10499923 | 67.631 | 67.631 |
| `nsumk` | 2299990 | 63.674 | 32.027 |
| `Multi2Index` | 1949979 | 63.283 | 27.305 |
| `combvec` | 1299985 | 50.295 | 14.153 |

Table 4: This table shows the most time consuming parts of calculating a 2 dimensional SSM for a $10^5$ dimensional phase space. It shows how many times a function has been called, what time it totally needed, and the amount of time actually spent in that function and not in other nested functions, here denoted by **Self Time**.

time consuming part is the assembly of the coefficient matrix. The part consuming all that time is located in line 6 of `CoefficientMatrix`.
The next function that consumes a large amount of time is the assembly of the force-terms. Here however most of the time spent is not self-time but nested in other functions. The two main factors here are `nsumkvector` and `Multi2Index`. In `nsumkvector`, the function `nsumk` is called iteratively which in turn uses `nchoosek`. Furthermore the function `combvec` is also called by `nsumkvector`. `Multi2Index` is also based on using `nchoosek`.

# 6 Discussion

## 6.1 Discussion of results

The two implementations are observed to produce the same multi-index coefficients with respect to the defined distance in Eq. (5.2) up to numerical errors. The errors bigger than that seem to arise from the resonance analysis, that determines the reduced dynamics. This is no surprise, since the newest version of the tensor-implementation uses a different method than a singular value decomposition to determine the near inner resonances. Additionally the multi-index version is tested against the published results of Ponsioen, Pedergnana and Haller (2019) [5], and it can be seen in Figure 3 that the obtained coefficients for the SSM and the reduced dynamics are the same up to working precision.

In section 5 it is portrayed clearly, that using the multi-index formalism one observes better performance than for the current implementation in tensor-format. Figures 6 and 8 show, that the theoretical limit for storage requirements for the SSM-coefficients is reached, which cannot be achieved by Storing the coefficients in tensor-format.

The next aspect to consider is the phase space size of which SSM-coefficients can be computed. Chapter 5.3.2 shows that while using the tensor-format a simulation of sizes bigger than $n = 250$ is not feasible on the mentioned computer, the multi-index-implementation can do $n = 5 * 10^4$ in a reasonable amount of time. Trials for bigger systems have not been run. While the multi-index-versions shows clear advantages for computing higher orders of SSM-coefficients, concerning memory requirements (Figure 6 and 8) as well as the time needed to compute the coefficients (Figure 7 and 9), the advantage for bigger phase space sizes mainly mainfests itself in aspects of computation time (Figure 11). The memory requirements of coefficients are also seen to be less than in the tensor-version (Figure 10).
As a conclusion, the multi-index-implementation is seen to work with less RAM, which is mainly due to not using Kronecker-products as intensively as the tensor-implementation, and performs better than the tensor-implementation. However, as soon as the degrees of freedom couple to more than only their direct neighbour it can be expected that the force assembly starts taking longer, since it only takes into account the force-values corresponding to non-zero coefficients. As soon as the relative amount of those non-zero columns with respect to the amount of degrees of freedom changes, calculations will take more time.

## 6.2 Possible improvements

As just mentioned, the force-assembly is expected to perform worse for more coupling between the degrees of freedom of a system. Therefore vectorising the assembly is expected to significantly improve performance. In order to do this, vectorised versions of the functions `nsumkvector` and `Multi2Index` are required. A possible implementation for the later could be achieved using the matlab function `sub2ind`.

Another part in the code that could be improved, of which the overall performance would profit immensely is line 6 of `CoefficientMatrix`. It is seen to consume most of the time for SSM computations of systems with large sizes, as can be seen in Table 4. For example, since for diagonal linear reduced dynamics, the coefficient-matrix always remains blockdiagonal. So instead of creating the big coefficient matrix, the block diagonal elements could be stored in a cell array and then with the part of the Right hand side corresponding to each block-diagonal matrix element, $z_k$ smaller linear problems could be solved.

Furthermore the current implementation can only deal with autonomous dynamical systems. A version which deals with time dependence is yet to be implemented.For bigger systems, the resonance analysis using `svd` gets very time and memory consuming. It is desirable to also implement a method similar to that currently used for the tensor-implementation.

# Appendix A

The reduced dynamics up to the order that we have calculated it for the example of the Bernoulli beam can be written as

$$\dot{p}_1 = \lambda_1 p_1 + \gamma_1 p_1^2 p_2^1 + \epsilon c_{1,\mathbf{0}} e^{i\phi} \tag{6.1}$$

By construction $\dot{p}_2$ is given by the complex conjugate of Eq. (6.1). Changing to polar coordinates $(\rho, \theta)$ and assigning $\psi = \theta - \phi$ one obtains

$$\dot{\rho} = a(\rho) + \epsilon(f_1(\rho, \Omega)) \cos\psi + f_2(\rho, \Omega) \sin\psi) \tag{6.2}$$

$$\dot{\psi} = b(\rho) - \Omega + \frac{\epsilon}{\rho}(g_1(\rho, \Omega) \cos\psi - g_2(\rho, \Omega) \sin\psi) \tag{6.3}$$

and the functions used are given as

$$a(\rho) = \Re(\lambda_1)\rho + \Re(\gamma_1)\rho^3$$
$$b(\rho) = \Im(\lambda_1) + \Im(\gamma_1)\rho^2$$
$$f_1 = \Re(c_{1,\mathbf{0}})$$
$$f_2 = \Im(c_{1,\mathbf{0}})$$
$$g_1 = \Im(c_{1,\mathbf{0}})$$
$$g_2 = \Re(c_{1,\mathbf{0}})$$

The force response curve is then given by solutions to

$$G(\rho, \Omega) := (b(\rho) - \Omega)\rho + \epsilon\left(g_1 \frac{1 - K^{\pm}(\rho)^2}{1 + K^{\pm}(\rho)^2} - g_2 \frac{2K^{\pm}(\rho)}{1 + K^{\pm}(\rho)^2}\right) = 0 \tag{6.4}$$

$$\text{with } K^{\pm}(\rho) = \frac{-\epsilon f_2 \pm \sqrt{\epsilon^2(f_1^2 + f_2^2) - a(\rho)^2}}{a(\rho) - \epsilon f_1}$$

These results were obtained by Ponsioen, Pedergnana and Haller [5].

## List of Figures

## List of Tables

## Listings

## References

[1] G. Haller, S. Ponsioen, *Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction*, Nonlinear Dynam. 86 (3) (2016) 1493–1534, https://doi.org/10.1007/s11071-016-2974-z.

[2] S. Ponsioen, T. Pedergnana, G. Halle *Automated computation of autonomous spectral sub-manifolds for nonlinear modal analysis,* . ournal of Sound and Vibration 420 (2018) 269-295, https://doi.org/10.1016/j.jsv.2018.01.048.

[3] A. Haro, M. Canadell, J.-L. Figueras, A. Luque, J.-M. Mondelo, *The Parameterization Method for Invariant Manifolds*, Springer (2016) 40-41, https://doi.org/10.1007/978-3-319-29662-3.

[4] S. Ponsioen, *Exact nonlinear model reduction in structural dynamics*, ETH Zürich (2019) 85-86, https://doi.org/10.3929/ethz-b-000401859.

[5] S. Ponsioen, T. Pedergnana, G. Haller, *Analytic prediction of isolated forced response curves from spectral submanifolds* Nonlinear Dyn 98, 2755–2773 (2019), https://doi.org/10.1007/s11071-019-05023-4.

[6] Shaw,S.W.,Pierre,C., *Normal modes of vibration for non- linear continuous systems* , J. Sound Vib. 163, 319–347 (1994).

[7] G. Habib, G. Cirillo, G. Kerschen, *Uncovering detached resonance curves in single-degree-of-freedom systems*, Procedia Eng. 199 (2017) 649 – 656. https://doi.org/10.1016/j.proeng.2017.09.116.

[8] *eps, Floating-point relative accuracy*, Matlab, viewed 22 March 2020, https://ch.mathworks.com/help/matlab/ref/eps.html.

[9] S. Ponsioen, G. Haller, *SSMtool*, viewed 1 April 2020, https://github.com/sponsioen/SSMtool.