Karlsruher Institut für Technologie

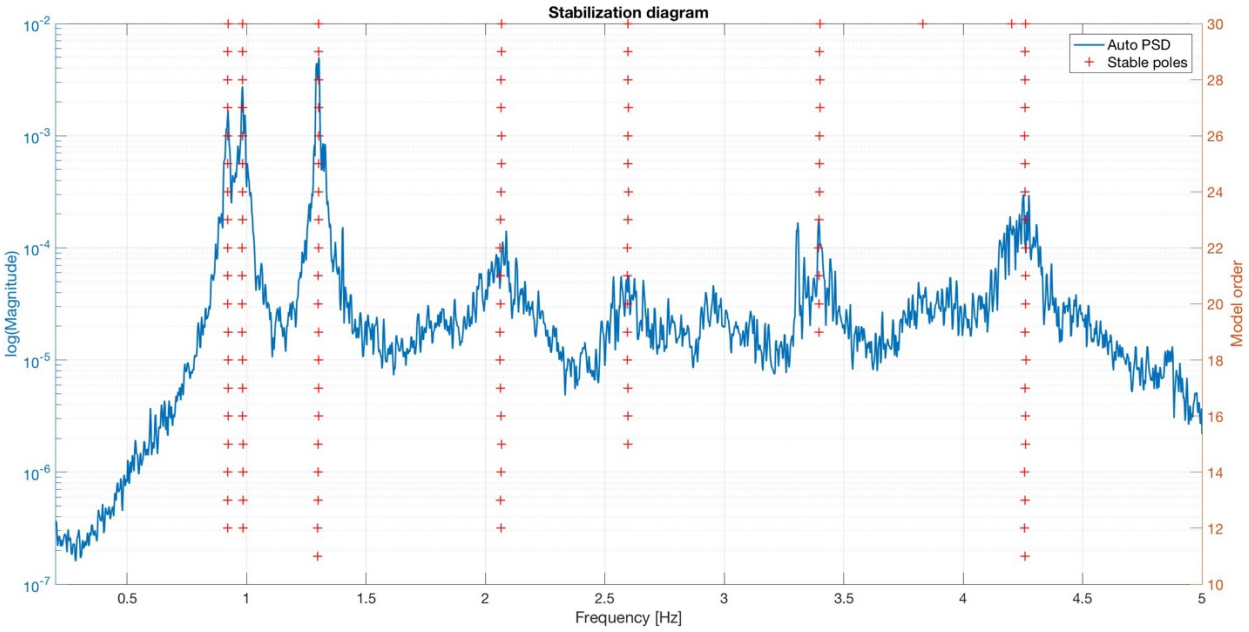
Institut für Fahrzeugsystemtechnik
Teilinstitut Fahrzeugtechnik
Leiter: Prof. Dr. rer. nat. Frank Gauterin

Rintheimer Querallee 2
76131 Karlsruhe

Bachelorarbeit

Implementierung des p-LSCF-Algorithmus zur Operational Modal Analysis

cand. mach. Raphael Frederik Trumpp



Projektleiter: Dipl.-Ing Achim Winandi, Teilinstitut Fahrzeugtechnik

Nr.: 17-F-0062

Karlsruhe, Juli 2017

Bachelorarbeit

Herrn cand. mach. Raphael Frederik Trumpp

Matr.-Nr.: 1808004

Implementierung des p-LSCF-Algorithmus zur Operational Modal Analysis

Aufgabenstellung

Im Rahmen des industriegeförderten Forschungsprojektes „Körperschallsimulation des rollenden Rad-Reifen-Systems mit Fahrwerk-Wechselwirkung“ werden unter anderem die Reifen-Fahrbahn-Interaktion und die Übertragung der Schwingungsenergie von Fahrwerken untersucht. Hierzu werden am Innentrommelprüfstand des Instituts für Fahrzeugsystemtechnik Messungen am rollenden Reifen mit einer Pkw-Vorderradführung durchgeführt.

In dieser Bachelorarbeit soll ein MATLAB-Programm zur Operational Modal Analysis (OMA) erstellt werden. Das Programm soll als Eingangsgrößen die Zeitrohdaten von gemessenen Beschleunigungen erhalten und als Ausgangsgrößen die modalen Parameter des Systems ausgeben.

Die Lösung der Aufgabenstellung erfolgt durch die Bearbeitung folgender Teilaufgaben:

- *Einarbeitung in das Thema durch die Recherche zum Stand der Technik & des Wissens,*
- *Konkretisierung der durchzuführenden Schritte (Zeitplan),*
- *Auswahl eines geeigneten Verfahrens zur OMA,*
- *Implementierung des ausgewählten Verfahrens in MATLAB (Automatische Identifikation der Polstellen und Bestimmung der modalen Parameter),*
- *gegebenenfalls Möglichkeit mehrere Messungen automatisch zusammenzufügen,*
- *Validierung des Programms anhand eines einfachen Beispiels (selbst zu wählen),*
- *gegebenenfalls Validierung des Programms an zur Verfügung gestellten Messdaten und*
- *Dokumentation der gesamten Arbeit.*

Hiermit wird die Vollständigkeit der Aufgabenstellung bestätigt. Alle darüber hinaus gehenden Aufgaben sind nicht Teil der Abschlussarbeit oder werden in der Ausarbeitung als solche kenntlich gemacht. Sollte es der/dem Studierenden, ohne eigenes Verschulden, nicht möglich sein, diese Aufgabenstellung in der vorgesehenen Bearbeitungszeit vollständig zu erfüllen, ist dies in der Ausarbeitung zu begründen.
Mit seiner Unterschrift nimmt der Studierende die Aufgabenstellung an.
Der Ausgabe- sowie Abgabetag ist somit bindend.

Ausgabetag: 18.04.2017

Abgabetag: 13.07.2017

Betreuer:

(Prof. Dr. rer. nat. Frank Gauterin)

Projektleiter:

(Dipl.-Ing Achim Winandi)

Bearbeiter:

(cand. mach. Raphael Frederik Trumpp)

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und nur mit den im Literaturverzeichnis angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Des Weiteren habe ich die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der aktuell gültigen Fassung beachtet.

Karlsruhe, den 13. Juli 2017

Kurzfassung

In der *Operational Modal Analysis* (OMA) werden die modalen Parameter (Eigenfrequenz, Dämpfungsmaß und Eigenform) einer Struktur aus experimentell bestimmten Messdaten identifiziert. Besonderheit der OMA ist, dass die Anregung der Struktur nicht gemessen wird und somit unbekannt ist. Unter der Voraussetzung der Strukturerregung durch weißes Rauschen, enthält die spektrale Leistungsdichte der gemessenen Systemantwort die vollständige Information über die modalen Parameter. Die *Poly-reference Least Squares Complex Frequency* Methode (p-LSCF) ist der aktuelle Industriestandard der OMA-Methoden im Frequenzbereich. In einem ersten Least-Square-Schritt wird auf Basis des parametrischen Modells das Stabilisierungsdiagramm konstruiert, um die stabilen Pole der Struktur zu identifizieren. In einem zweiten Least-Square-Schritt werden die Eigenformen bestimmt. Diese Methode kann auch nah benachbarte Eigenmoden identifizieren und liefert klare, leicht zu interpretierende Stabilisierungsdiagramme. Im Rahmen dieser Bachelorarbeit wurde das p-LSCF-Verfahren als vollautomatisches Programm in MATLAB implementiert und anhand mehrerer Datensätze verifiziert.

Implementation of the p-LSCF Algorithm for Operational Modal Analysis

Abstract

Operational Modal Analysis (OMA) identifies the modal parameters (natural frequency, damping ratio and eigenform) of a structure from experimentally determined measured data. The special feature of the OMA is that the excitation of the structure is not measured and is thus unknown. Given the structure excitation by white noise, the power spectral density of the measured system response contains the complete information on the modal parameters. The *poly-reference least squares complex frequency* method (p-LSCF) is the current industry standard of OMA methods in the frequency domain. In a first least squares step, the stabilization diagram is constructed based on the parametric model to identify the stable poles of the structure. The eigenforms are determined in a second least-square step. This method can also identify closely spaced eigenmodes and provides clear, easily interpretable stabilization diagrams. Within the scope of this Bachelor thesis, the p-LSCF method was implemented as a fully automated program in MATLAB and verified by several data sets.

Inhaltsverzeichnis

A Abbildungsverzeichnis

B Tabellenverzeichnis

C Abkürzungsverzeichnis

CDM	Common Denominator Model
CWN	Continuous-time White Noise
DFT	Discrete Fourier Transform
DOF	Degrees of Freedom
FE	Finite Elemente
FTT	Fast Fourier Tranform
LMFD	Left Matrix Fraction Description
LS	Least Squares
LSCE	Least Squares Complex Exponential
LSCF	Least Squares Complex Frequency
LSCF	Least Squares Complex Frequency
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
NLS	Nonlinear Least Squares
NRMSD	Normalized Root Mean Square Deviation
OMA	Operational Modal Analysis
p-LSCF	Poly-reference Least Squares Complex Frequency
PoGER	Post Global Estimation Re-Scaling
PoSER	Post Separate Estimation Re-Scaling
PreGER	Pre Global Estimation Re-Scaling
PRM	Pole-Residue Model
PSD	Power Spectral Sensity
PSD +	Positiv Power Spectral Densitiy
RMFD	Right Matrix Fraction Desciption

D Symbolverzeichnis

$[\cdot]$	Matrix
$\langle\cdot\rangle$	Reihenvektor
$\{\cdot\}$	Zeilenvektor
i	Imaginäre zahl
$\mathrm{E}[\cdot]$	Erwartungswert

1 Einleitung

1.1 Forschungskontext

Im Konstruktionsprozess werden Anforderungen an das dynamische Verhalten eines mechanischen Produkts gestellt. Meist sind ein niedriges Geräuschlevel und geringe Vibrationen im Betrieb erwünscht. Diese Anforderungen sind in einem wirtschaftlichen Rahmen zu erfüllen. Daher bedarf es Analysemethoden, die in geringer Konstruktions- und Entwicklungszeit zu Konstruktionsverbesserungen führen. Konstruktionsverbesserungen können durch numerische Simulation und dem daraus resultierenden modalen Modell zur Beschreibung des dynamischen Verhalten der Struktur vorhergesagt und optimiert werden. Eine solide dynamische Gestaltung von realen komplexen Strukturen kann nicht nur auf Basis numerischer Untersuchungen gewährt werden. Durch die Modalanalyse wird ausgehend von Messungen des Schwingungsverhaltens einer Struktur das modale Modell ebenfalls bestimmt. Das experimentell bestimmte modale Modell kann genutzt werden, um das durch die Simulation gewonnene modale Modell zu verifizieren und ggf. zu optimieren [Moha05].

Die Modalanalyse umfasst die experimentelle Untersuchung der dynamischen Eigenschaften schwingungsfähiger Systeme auf Basis des modalen Modells. Das modale Modell beschreibt das dynamische Verhalten der Struktur durch die modalen Parameter:

- Eigenfrequenz,
- Eigenform und
- modale Dämpfung.

Das Ziel der Modalanalyse ist es daher, die modalen Parameter einer Struktur zu identifizieren [HeFu01].

Unter der Annahme, dass ein System vollständig durch die modalen Parameter beschrieben werden kann, entstand Mitte des 20. Jahrhunderts die *Experimental Modal Analysis* (EMA, dt. experimentelle Modalanalyse) [RaFa00] . In der EMA werden die modalen Parameter ausgehend von Messungen der Strukturanregung und Schwingungsantwort bestimmt. Durch die Entwicklung der *Finite Elemente Methode* (FEM) hat die EMA weiter an Wert gewonnen, um die theoretisch durch Computersimulation gewonnenen Systemeigenschaften mit real gemessenen Eigenschaften verifizieren zu können. Als wissenschaftliche Grundlage der EMA seien die Werke von [Ewin00], [HeLS07] und [MaMo97] hervorheben. Anwendungsbereiche der EMA sind vielseitig und reichen von der Fahrzeugtechnik, der Raumfahrtechnik über die Industrietechnik hin zum Bauingenieurwesen [RaFa14].

Jedoch gestaltet sich die kontrollierte und harmonische Anregung des Systems häufig als schwierig bzw. diese kann überhaupt nicht gemessen werden. Insbesondere im Bauingenieurwesen sind die anzuregenden Systeme meist groß (z.B. Hochhäuser) und die Anregung ist nicht quantifizierbar (z.B. die Anregung durch Wind).

Aus dieser Ausgangssituation wurde die *Operational Modal Analysis* (OMA, dt. Betriebsanalyse) entwickelt. Weitere gängige Bezeichnungen für OMA sind *Ambient Vibration Analysis* oder *Output-only Analysis*. In [RaFa14] wird OMA als das modale Testverfahren definiert, das die experimentelle Schätzung der modalen Parameter der Struktur nur aus Messungen der Schwingungsantwort ermöglicht. Zur Strukturerregung werden Umgebungskräfte (z.B. Wind, Verkehr, Anregung durch Betriebszustand) genutzt. Daher kann die untersuchte Struktur in realen Betriebszuständen untersucht werden und ist nicht wie EMA an Laborbedingungen gebunden [Caub04].

Die Bestimmung des modalen Modells ist durch folgende in [Moha05] beschriebenen Anwendungen in der Praxis motiviert:

- *Fehlersuche*: Die modalen Parameter werden verwendet, um das dynamische Verhalten einer Struktur zu verstehen und die Ursache ungewollter Probleme im realen Betrieb (z.B. Resonanz, Geräusche) herauszufinden.
- *Modellaktualisierung*: Das dynamische Verhalten einer Struktur kann nicht vollständig korrekt durch ein mathematisches Modell (z.B. FE-Modell) dargestellt werden. Die gewählten Materialparameter, Randbedingungen und getroffene Annahmen (Nicht-Linearitäten, Dämpfung etc.) weichen unweigerlich von der Realität ab. Das experimentell bestimmte modale Modell wird genutzt, um das Modell der Simulation zu verbessern und somit das Verhalten der Struktur besser vorhersagen zu können.
- *Strukturüberwachung und Schadenserkennung*: Der Zustand einer untersuchten Struktur kann durch den Vergleich der modalen Parameter im aktuellen Zustand mit den modalen Parametern im Neuzustand bestimmt werden. Schäden können somit früh identifiziert werden.

1.2 Zielsetzung

Zur Bestimmung des modalen Modells für OMA bedarf es verlässlicher Methoden, die auf Basis der gemessenen Schwingungsantwort die modalen Parameter der untersuchten Struktur identifizieren. Die *Poly-reference Least Squares Complex Frequency* Methode (p-LSCF) ist eine der modernsten OMA-Methoden zur Bestimmung der modalen Parameter und aktueller Industriestandard.

Ziel dieser Bachelorarbeit ist es, den p-LSCF-Algorithmus als Programm in der kommerziell erhältlichen Software MATLAB zu implementieren. Dabei soll das implementierte Programm die modalen Parameter der untersuchten Struktur aus der gemessenen Schwingungsantwort identifizieren. Die zu identifizierenden modalen Parameter sind die Eigenfrequenzen, Dämpfungsmaße und Eigenformen. Der Programmablauf soll vollautomatisch gestaltet werden, damit nur geringfügige Benutzereingaben nötig sind. In diesem Sinne soll der Verarbeitungsprozess auf Grundlage des gemessenen Zeitsignals der Sensoren beginnen. Auch die Verarbeitung von Messdaten, welche in unterschiedlichen Datensätzen (Multi-Sensor-Layout) gemessen wurden, soll mit dem implementierten Programm möglich sein. Hierfür ist ein geeignetes Verfahren zu recherchieren und zu implementieren. Um als Basis weiterer Forschungsarbeiten dienen zu können, soll die Programmstruktur des implementierten Programms vollständig dokumentiert und beschrieben werden.

Die Implementierung der p-LSCF-Methode soll im Anschluss verifiziert werden. Dies soll zuerst an einem aus einem Lehrbuch entnommenen Datensatz geschehen, zu dem Lösungen zur Verfügung stehen. Abschließend soll der Algorithmus zur Analyse eines real gemessenen Datensatzes verwendet werden.

2 Theorie: Allgemeines

2.1 Mathematischer Hintergrund

Die OMA nutzt mathematische Modelle zur Beschreibung von dynamischen Systemen und deren Datenanalyse. Um die in den folgenden Kapiteln dargestellten mathematischen Konzepte verstehen zu können, wird ein breites Grundwissen der Mathematik vorausgesetzt. Hervorzuheben seien hierbei die Matrix-Algebra und ihre gängigen Operationen, das Rechnen mit komplexen Zahlen sowie grundlegende Begriffe der Statistik. Ebenfalls vorausgesetzt sei ein grundlegendes Verständnis der Technischen Mechanik und schwingungsfähiger Strukturen. Der vorausgesetzte Wissensstand orientiert sich hierbei an dem eines Studierende technischer Fachrichtung mit abgeschlossenem Grundstudium. Nichtsdestotrotz seien auch Studierende nichttechnischer Fachrichtung oder mit geringerer mathematischer Vorbildung zum Studium der vorliegenden Arbeit ermutigt. Es wird versucht, mathematische Zusammenhänge so einfach wie möglich darzustellen. Zur Erweiterung des Wissens seien hierbei die Werke von [HeFu01], [Natk88] und [RaFa14] empfohlen.

Im Folgenden werden zusätzlich mathematische Werkzeuge erläutert, die zum Verständnis der vorliegenden Arbeit benötigt werden. Deren mathematischen Herleitungen basieren auf [RaFa14]. Werden weitere Quellen verwendet, wird auf diese explizit verwiesen.

2.1.1 Fouriertransformation (FFT)

Die Grundannahme der Fourier-Analyse ist, dass jedes Signal in eine Linearkombination aus Sinus-Funktionen unterschiedlicher Frequenzen zerlegt werden kann. Diese Zerlegung wurde ursprünglich für periodische Funktionen entwickelt. Sie kann jedoch auch auf nichtperiodische Funktionen, wie z.B. stochastische Signale, angewandt werden. Dazu wird angenommen, dass dies periodische Funktionen mit einer Periode gleich der Dauer T des Signals sind. Die folgende Integrabilitätsbedingung muss für die klassische Fouriertransformation erfüllt sein:

$$\int_{-\infty}^{+\infty} |\mathrm{x(t)}|\,\mathrm{dt} < \infty. \qquad (2.1)$$

Für ein periodisches Signal $x(t)$ ist die Fouriertransformation wie folgt definiert:

$$\mathrm{X(f)} = \int_{-\infty}^{+\infty} \mathrm{x(t)e^{-i2\pi ft}}\,\mathrm{dt}. \qquad (2.2)$$

Die inverse Fouriertransformation wird beschrieben durch:

$$\mathrm{x(t)} = \int_{-\infty}^{+\infty} \mathrm{X(f)e^{i2\pi ft}dt}. \qquad (2.3)$$

In praktischen Anwendungen wird das Signal $x(t)$ durch digitales Equipment aufgezeichnet und analysiert. Es findet eine Diskretisierung des Signals statt, wodurch das Signal durch eine Folge

an Werten zu bestimmten äquidistanten Zeitpunkten dargestellt wird. Hierbei ist die Abtastzeit Δt die Inverse der Abtastfrequenz f_s. Die Abtastfrequenz gibt an, wie oft das Signal pro Sekunde [Hz] abgetastet wird:

$$\Delta \mathrm{t} = \frac{1}{\mathrm{f_s}}. \qquad (2.4)$$

Um für das Signal $x(t)$ eine saubere Signalauflösung zu gewährleisten, muss das Shannonsche Abtasttheorem erfüllt sein. Dieses besagt, dass die Abtastfrequenz f_s mindestens doppelt so hoch sein muss, wie die höchste darzustellende Frequenz f_{max}:

$$\mathrm{f_s} \geq 2\mathrm{f_{max}}. \qquad (2.5)$$

Ein diskretes Signal $\mathrm{x_n}$ ergibt sich zu:

$$\mathrm{x_n} = \mathrm{x(n\Delta t)\ für\ n} = 0{,}1{,}2, \dots, \mathrm{N} - 1, \qquad (2.6)$$

wenn das Signal $x(t)$ zu N gleich beanstandeten Zeitpunkten abgetastet wurde und das Shannonsches Abtasttheorem erfüllt ist.
Die diskreten Frequenzwerte werden durch die „Unschärferelation“ wie folgt definiert:

$$\mathrm{f_k} = \frac{\mathrm{k}}{\mathrm{T}} = \frac{\mathrm{k}}{\mathrm{N}}\Delta \mathrm{t} \quad , \mathrm{k} = 0{,}1{,}2, \dots, \mathrm{N} - 1. \qquad (2.7)$$

Die diskrete Fouriertransformation (DFT) ist gegeben durch:

$$\mathrm{X_k} = \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{x_n} \mathrm{e}^{-\frac{2\mathrm{i}\pi \mathrm{kn}}{\mathrm{N}}} \quad , \mathrm{k} = 0{,}1{,}2, \dots, \mathrm{N} - 1. \qquad (2.8)$$

Die Koeffizienten X_k sind komplexe Zahlen. Zur Berechnung der Fouriertransformation werden N^2 Rechenoperationen benötigt. Um den Rechenaufwand zu reduzieren, wurde die *Fast Fourier* Transform (FFT, dt. schnelle Fouriertransformation) von Cooley und Turkey 1965 entwickelt. Bei erfüllten Voraussetzungen wird die Anzahl der Rechenoperationen hierbei auf $N * log_2(N)$ reduziert. Der Betrag der Fourier-Koeffizienten beschreibt die Amplitude des Signals:

$$|X_k| = \sqrt{[Re(X_k)]^2 + [Im(X_k)]^2}. \qquad (2.9)$$

Dessen Phase wird wie folgt beschrieben:

$$\Theta = \arctan\left(\frac{Im(X_k)}{Re(X_k)}\right). \qquad (2.10)$$

2.1.2 Stochastische Begriffe

Die folgenden Definitionen sind aus [Natk88] übernommen. Für weitere Information zu den stochastischen Begriffen sei ebenfalls auf dieses Werk verwiesen.

- *Ergodisch*: „Der stationäre Zufallsprozess $\{x(t)\}$ heißt ergodisch bezüglich einer Menge G von Funktionen $g[x(t)]$, wenn für jede Funktion $g \in G$ die Beziehung $\overline{g[x(t)]} = E\{g[x(t)]\}$ mit der Wahrscheinlichkeit Eins erfüllt ist. Der Zeitmittelwert fällt mit dem Mittelwert der Grundgesamtheit mit der Wahrscheinlichkeit 1 zusammen. Daher können die Erwartungswerte der Grundgesamtheit aus einer einzigen Musterfunktion des Prozesses gewonnen werden. Für ergodische Prozesse ergibt sich damit, dass die Grundgesamtheit ohne Kenntnis der Verteilungsfunktion aus Zeitmittelwerten erhalten werden und es genügt, *eine* Musterfunktion zu betrachten."
- *Unverzerrt*: „Sei $\hat{a}$ eine Schätzfunktion $\dot{a}$ der wahre Wert. Eine Schätzung wird erwartungstreu (unverzerrt, biasfrei) genannt, wenn $E\{\hat{a}\} = \dot{a}$ ist, andernfalls ist sie nicht erwartungstreu (verzerrt). Der Erwartungswert einer unverzerrten Schätzfunktion entspricht dem wahren Wert."
- *Konsistent*: „Eine Schätzung ist konsistent, wenn sie für $N \to \infty$ mit der Wahrscheinlichkeit 1 gegen den wahren Wert konvergiert. Konsistente Schätzungen können ein Bias für endliches N haben und eine unverzerrte Schätzung muss nicht konsistent sein. Eine konsistente Schätzung ist aber stets asymptotisch unverzerrt."

Im Folgenden werden die statistischen Größen durch die Zusatzbezeichnungen Auto- und Kreuzgekennzeichnet. Beispielsweise wird von Auto-PSD bzw. Kreuz-PSD gesprochen. Auto- bedeutet in diesem Kontext, dass die statistische Größe Signals $x_k(t)$ zu sich selbst bestimmt wird und somit Informationen über die Beziehung des Signals $x_k(t)$ zu sich selbst beschreibt. Dahingegen bedeutet die Ergänzung Kreuz-, dass die statistische Größe auf Basis der zwei Signale $x_k(t)$ und $y_k(t)$ bestimmt wurde. Die durch Kreuz- gekennzeichneten Größen beschreiben also den statistischen Zusammenhang zweier unterschiedlicher Signalen $x_k(t)$ und $y_k(t)$ zueinander.

2.1.3 Korrelationsfunktionen

Für die OMA sind Korrelationsfunktionen wichtig, da sie unter der Annahme eines stationär stochastischen Prozesses alle benötigten physikalischen Informationen beinhalten. Die zwei gegebenen Aufnahmefunktionen $x_k(t)$ und $y_k(t)$ zweier stationär stochastischer Prozesse sind unabhängig von der Zeit t. Die Auto-Korrelationsfunktionen $\mathrm{R_{xx}}$ und $\mathrm{R_{yy}}$ von $x_k(t)$ und $y_k(t)$ sind über den Erwartungswert $\mathrm{E}[\cdot]$ definiert als:

$$\mathrm{R_{xx}} = \mathrm{E}[\mathrm{x_k(t)x_k(t+\tau)}] \qquad (2.11)$$

$$\mathrm{R_{yy}} = \mathrm{E}[\mathrm{y_k(t)y_k(t+\tau)}]. \qquad (2.12)$$

Die Kreuz-Korrelationsfunktion $\mathrm{R_{xy}}$ von $x_k(t)$ und $y_k(t)$ wird berechnet durch:

$$\mathrm{R_{xy}} = \mathrm{E[x_k(t)y_k(t+\tau)]}. \qquad (2.13)$$

In der Praxis liegt jedoch weder die Gesamtheit eines stochastischen Prozesses vor, noch sind die Verteilungsdichtefunktionen bekannt. Ist der Prozess jedoch ergodisch, so entspricht die zeitlich gemittelte Korrelationsfunktion der über die Gesamtheit gemittelten Korrelationsfunktion. Es reicht also eine einzige Abtastfunktion aus, um diesen Kennwert bestimmen zu können. Prozesse in der Praxis sind üblicherweise ergodisch.
Überprüft werden kann das, indem die folgenden Punkte erfüllt sind:

- Der stochastische Prozess ist (schwach) stationär und die zeitlichen Mittelwerte μ_x und die Auto-Kovarianz C_{xx} sind gleich für alle Abtastfunktionen.
- Die Auto-Korrelationsfunktion erfüllt folgende Bedingung: $\frac{1}{T}\int_{-T}^{T}|C_{xx}(\tau)|d\tau \to 0 \; für \; T \to 0.$

In der praktischen Anwendung werden Zeitreihenaufzeichnungen als stationär bezeichnet, wenn ihre Werte in den jeweiligen Aufzeichnungsintervallen nicht groß von einen zum anderen variieren. Da ein aus einem ergodischen Prozess gewonnener Datensatz stets stationär ist, reicht es vereinfachend aus, nur die Stationarität zu überprüfen. Ein stationärer Prozess wird ebenfalls als ergodisch angenommen. Unter der Annahme zweier stationär ergodischer Prozesse, können die Auto- und Kreuz-Korrelationenfunktionen $\hat{R}_{\mathrm{xx}}$, $\hat{R}_{\mathrm{yy}}$ und $\hat{R}_{\mathrm{xy}}$ unverzerrt geschätzt werden [Natk88]:

$$\hat{R}_{\mathrm{xx}} = \frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} \mathrm{x(t)x(t+\tau)dt} \quad ,0 \le \tau \le \mathrm{T} \qquad (2.14)$$

$$\hat{R}_{\mathrm{yy}} = \frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} \mathrm{y(t)y(t+\tau)dt} \quad ,0 \le \tau \le \mathrm{T} \qquad (2.15)$$

$$\hat{R}_{\mathrm{xy}} = \frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} \mathrm{x(t)y(t+\tau)dt} \quad ,0 \le \tau \le \mathrm{T}. \qquad (2.16)$$

Für einen diskreten Datensatz können die Korrelationsfunktionen entweder direkt oder im Sinne einer FFT berechnet werden. Die direkte, unverzerrte Schätzung der Autokorrelation $\widehat{\mathrm{R}}_{\mathrm{xx}}$ (Mittelwert $\mu_x = 0$) ergibt sich zu:

$$\widehat{\mathrm{R}}_{\mathrm{xx}}(\mathrm{r\Delta t}) = \frac{1}{\mathrm{N-r}}\sum_{\mathrm{n=1}}^{\mathrm{N-r}} \mathrm{x_n x_{n+r}} \quad ,\mathrm{r} = 0,1,2,\dots,\mathrm{m}. \qquad (2.17)$$

Die Korrelationsfunktionen dienen als Maß für den statistischen Zusammenhang zweier stochastischer Funktionen und stellen ein Maß für die Energie bzw. Leistung eines Prozesses im Zeitbereich dar [Natk88].

2.1.4 Spektrale Leistungsdichte (PSD)

Um die Information der Korrelationsfunktionen vom Zeitbereich in den Frequenzbereich zu übertragen, werden diese fouriertransformiert. Die erhaltene Größe wird *Power Spectral Density* (PSD, dt. spektrale Leistungsdichte) genannt. Sie stellt ein Maß für die Leistung pro Frequenz eines Prozesses dar [Natk88]. Im Allgemeinen müssen diese jedoch nicht die Dimension einer Leistung pro Frequenz ausweise. Die PSD kann sowohl über das Korrelogramm- als auch über das Periodogramm-Verfahren geschätzt werden. Diese Möglichkeiten werden im Folgenden dargestellt:

Korrelogramm-Verfahren:

Der Ansatz zur Bestimmung der PSD über die Korrelationsfunktionen basiert auf dem Wiener-Khinchin-Theorem. Dieses Theorem besagt, dass die PSD eines stationären Zufallsprozesses die Fouriertransformierte der zugehörigen Auto-Korrelationsfunktion ist. Damit ergeben sich die doppeltseitigen Auto- und Kreuz-PSDs $S_{xx}(f), S_{yy}(f)$ und $S_{xy}(f)$ zu:

$$S_{xx}(f) = \int_{-\infty}^{+\infty} R_{xx}(\tau) e^{-i2\Pi f t} d\tau \qquad (2.18)$$

$$S_{yy}(f) = \int_{-\infty}^{+\infty} R_{yy}(\tau) e^{-i2\Pi f t} d\tau \qquad (2.19)$$

$$S_{xy}(f) = \int_{-\infty}^{+\infty} R_{xy}(\tau) e^{-i2\Pi f t} d\tau. \qquad (2.20)$$

Die doppeltseitige Auto-PSD $S_{xx}(f)$ bzw. $S_{yy}(f)$ ist hierbei eine real-wertige Funktion und die doppeltseitige Kreuz-PSD $S_{xy}(f)$ eine komplexe Funktion ist.

Wendet man statt der doppeltseitigen Fouriertransformation die einseitige an, kann die PSD kann auch als einseitige Funktion dargestellt werden. In praktischen Anwendungen wird meist die einseitige PSD verwendet, da diese das Ergebnis einer Fouriertransformation einer Aufnahme endlicher Länge mit $T < \infty$ ist. Dabei verdoppelt sich die Amplitude, während die Phase erhalten bleibt. $G_{xx}(f), G_{yy}(f)$ und $G_{xy}(f)$ sind die einseitigen Auto- und Kreuz-PSD:

$$G_{xx}(f) = 4 \int_{0}^{+\infty} R_{xx}(\tau) \cos(2\pi f \tau)\, d\tau = 2S_{xx}(f) \qquad (2.21)$$

$$G_{yy}(f) = 4 \int_{0}^{+\infty} R_{yy}(\tau) \cos(2\pi f \tau)\, d\tau = 2S_{yy}(f) \qquad (2.22)$$

$$G_{xy}(f) = 2 \int_{-\infty}^{+\infty} R_{xy}(\tau) e^{-i2\Pi f t} d\tau = 2S_{xy}(f). \qquad (2.23)$$

Die Bedeutung von ein- bzw. doppeltseitiger PSD wird in Abb. 1 dargestellt:

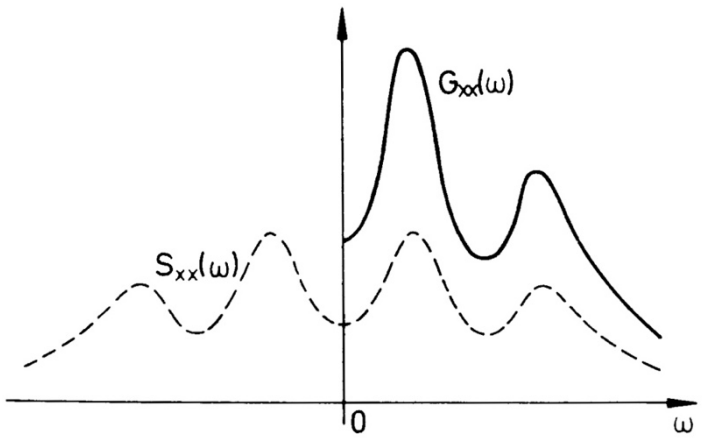


Abb. 1: Prinzipieller Verlauf der einseitigen Auto-PSD $G_{xx}(\omega)$ und doppeltseitigen Auto-PSD $S_{xx}(\omega)$ nach [Natk88]

Eine unverzerrte, konsistente Schätzung der PSD stellt das Korrelogramm-Verfahren dar:

$$\hat{S}_{\mathrm{xx}}(\mathrm{f}) = \int_{-\infty}^{+\infty} \hat{R}_{\mathrm{xx}}(\tau)\mathrm{e}^{-\mathrm{i}2\Pi\mathrm{ft}}\mathrm{d}\tau \qquad (\ 2.24\ )$$

Wird jedoch eine Fensterfunktion $w(\tau)$ zur Schätzung verwendet mit $\tilde{R}_{\mathrm{xx}}(\tau) = w(\tau)\hat{R}_{\mathrm{xx}}(\tau)$, ist die Schätzung lediglich für $w(\tau) = 1$ unverzerrt.

Periodogramm-Verfahren:

Die PSD kann ebenfalls über das Periodogramm-Verfahren geschätzt werden. Für einen stationär stochastischen Prozess $x_k(t)$ existiert die Fouriertransformation nicht, da die Integrabilitätsbedingung nach Formel (2.1) nicht erfüllt ist. Die finiten Fouriertransformationen $\mathrm{X_k}$ und $\mathrm{Y_k}$ für ein gegebenes Datensatzpaar $x_k(t)$ und $y_k(t)$ endlicher Dauer T eines stationär stochastischen Prozesses existieren jedoch zu:

$$\mathrm{X_k}(\mathrm{f},\mathrm{T}) = \int_0^{\mathrm{T}} \mathrm{x_k}(\mathrm{t})\mathrm{e}^{-\mathrm{i}2\Pi\mathrm{ft}}\mathrm{dt} \qquad (\ 2.25\ )$$

$$\mathrm{Y_k}(\mathrm{f},\mathrm{T}) = \int_0^{\mathrm{T}} \mathrm{y_k}(\mathrm{t})\mathrm{e}^{-\mathrm{i}2\Pi\mathrm{ft}}\mathrm{dt}. \qquad (\ 2.26\ )$$

Dann ist die Schätzung der doppeltseitigen Auto- und Kreuz-PSD $\hat{\mathrm{S}}_{\mathrm{xx}}$, $\hat{\mathrm{S}}_{\mathrm{yy}}$ und $\hat{\mathrm{S}}_{\mathrm{xy}}$ für eine ergodischen Prozess für die Frequenz f im Bereich $(-\infty, +\infty)$ definiert als [Natk88]:

$$\hat{\mathrm{S}}_{\mathrm{xx}} = \frac{1}{\mathrm{T}}\mathrm{X_K^*}(\mathrm{f},\mathrm{T})\mathrm{X_K}(\mathrm{f},\mathrm{T}) \qquad (\ 2.27\ )$$

$$\hat{\mathrm{S}}_{\mathrm{yy}} = \frac{1}{\mathrm{T}}\mathrm{Y_K^*}(\mathrm{f},\mathrm{T})\mathrm{Y_K}(\mathrm{f},\mathrm{T}) \qquad (\ 2.28\ )$$

$$\hat{\mathrm{S}}_{\mathrm{xy}} = \frac{1}{\mathrm{T}}\mathrm{X_K^*}(\mathrm{f},\mathrm{T})\mathrm{Y_K}(\mathrm{f},\mathrm{T}). \qquad (\ 2.29\ )$$

Die einseitigen Schätzungen ergeben sich für einen ergodischen Prozess für f im Bereich $(-0, +\infty)$ zu:

$$\widehat{\mathrm{G}}_{\mathrm{xx}} = 2\hat{\mathrm{S}}_{\mathrm{xx}} = \frac{2}{\mathrm{T}}\mathrm{X}_{\mathrm{K}}^{*}(\mathrm{f},\mathrm{T})\mathrm{X}_{\mathrm{K}}(\mathrm{f},\mathrm{T}) \qquad (\ 2.30\ )$$

$$\widehat{\mathrm{G}}_{\mathrm{yy}} = 2\hat{\mathrm{S}}_{\mathrm{yy}} = \frac{2}{\mathrm{T}}\mathrm{Y}_{\mathrm{K}}^{*}(\mathrm{f},\mathrm{T})\mathrm{Y}_{\mathrm{K}}(\mathrm{f},\mathrm{T}) \qquad (\ 2.31\ )$$

$$\widehat{\mathrm{G}}_{\mathrm{xy}} = 2\hat{\mathrm{S}}_{\mathrm{xy}} = \frac{2}{\mathrm{T}}\mathrm{X}_{\mathrm{K}}^{*}(\mathrm{f},\mathrm{T})\mathrm{Y}_{\mathrm{K}}(\mathrm{f},\mathrm{T}). \qquad (\ 2.32\ )$$

Diese Schätzung ist nur asymptotisch unverzerrt und nicht konsistent. Unter asymptotisch unverzerrt versteht man eine Schätzung, die für eine endliche Anzahl an Stichproben nicht erwartungstreu ist. Wird der Stichprobenumfang jedoch größer, wird der Bias kleiner. Für Stichprobenanzahl $N \to \infty$ verschwindet der Bias [Natk88].

Angewandte Verfahren in der Praxis:

In der Praxis gibt es mehrere Möglichkeiten die PSD zu erhalten. Beim Blackman-Tukey-Verfahren wird die PSD-Matrix über das Korrelogramm-Verfahren geschätzt, indem zuerst die Korrelationsfunktion bestimmt und diese anschließend fouriertransformiert wird.

Ein anderer Ansatz ist das Welch-Verfahren. Dieses basiert auf dem Periodogramm-Verfahren, also der direkten Berechnung der FFT der Aufzeichnungen und der Bestimmung der PSDs, wie in (2.30) bis (2.32) beschrieben. Das Welch-Verfahrens ist im Vergleich zum Blackman-Tukey-Verfahren rechnerisch weniger anspruchsvoll. Die Güte der Schätzungen ist jedoch geringer, da die Schätzung nur asymptotisch unverzerrt und nicht konsistent ist. Die Güte der Schätzung kann verbessert werden, indem die einseitige Auto-PSD $\widehat{\mathrm{G}}_{\mathrm{xx}}(\mathrm{f})$ statt wie in (2.30) folgendermaßen bestimmt wird:

$$\widehat{\mathrm{G}}_{\mathrm{xx}}(\mathrm{f}) = \frac{2}{\mathrm{n_d N\Delta t}}\sum_{\mathrm{i}=1}^{\mathrm{n_d}} |\mathrm{X_i(f)}|^2. \qquad (\ 2.33\ )$$

Hierbei wird ein Datensatz in n_d zusammenhängende Segmente der Länge $T = N\Delta t$ aufgeteilt und jedes Segment einzeln fouriertransformiert. Anschließend wird die PSD berechnet, indem über die n_d Segmente gemittelt wird. Die Anzahl der Datenwerte N in jedem Segment wird als Blockgröße bezeichnet und bestimmt die Frequenzauflösung der resultierenden Schätzung. Die Anzahl der Mittelwerte n_d bestimmt stattdessen den zufälligen Fehler der Schätzung.

Aufbau der PSD-Matrix

Werden mehrere Signale (z. B Messdaten mehrerer Sensoren) ausgewertet, werden für die Anwendung von OMA alle zwischen den Signalen möglichen PSDs bestimmt und zur sogenannten PSD-Matrix zusammengefasst. Es werden sowohl die Beziehungen aller Signals $\mathrm{x_k(t)}$ zu sich selbst in Form der einseitigen Auto-PSD $\mathrm{G}_{xx}(\mathrm{f})$, als auch die Beziehungen zwischen allen

Signalen, z. B zwischen x_k und y_k, als Kreuz-PSD $G_{xy}(f)$ bestimmt. Werden l Signale ausgewertet, ist die PSD-Matrix eine $l \times l$-Matrix. Der klassische Aufbau der PSD-Matrix für N_f diskrete Frequenzen des zur PSD-Matrix gehörenden Frequenzbandes wird in Abb. 2 dargestellt:

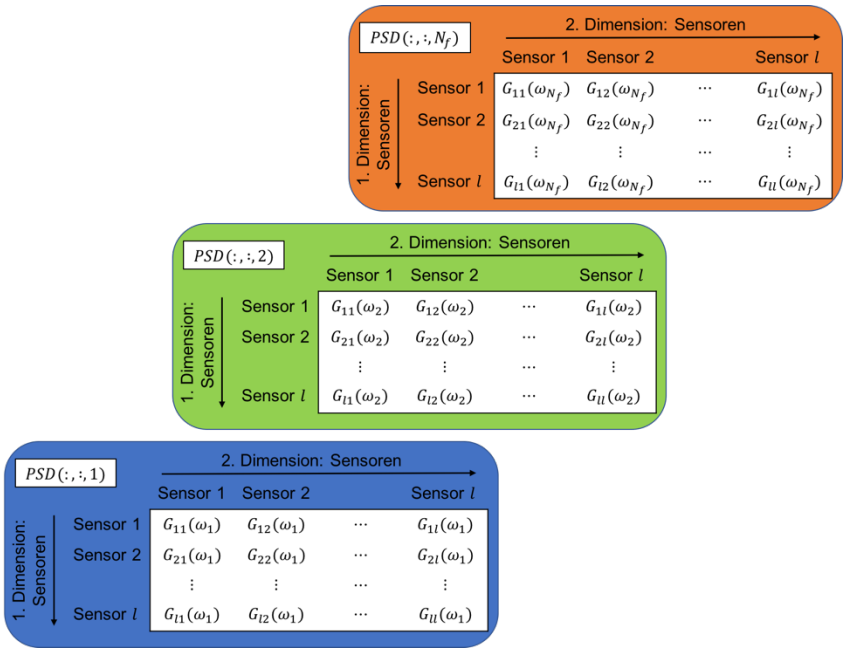


Abb. 2: Klassischer Aufbau der PSD-Matrix aus allen Auto- und Kreuz-PSDs aller ausgewerteten Signale

2.1.5 Z-Transformation

Die z-Transformation entspricht der Laplace-Transformation im diskreten Zeitbereich. Die Laplace-Transformation wandelt einen kontinuierlichen Zeitbereich in einen komplexen Frequenzbereich. Die z-Transformation wandelt also einen diskreten Zeitbereich in einen komplexen Frequenzbereich. Angenommen die reale Funktion $f(t)$ im Zeitbereich wird durch ihre Abtastungen zu den Zeitpunkten $t = 0,1,2, \dots, n$ beschrieben:

$$f(0), f(1), f(2), \dots, f(n).$$

Dann wird das folgende Polynom als z-Transformation bezeichnet:

$$F(z) = f(0) + f(1)z^{-1} + f(2)z^{-2} + \cdots + f(n)z^{-n}. \qquad (2.34)$$

Hierbei ist z eine komplexe Variable mit $z = e^{-st}$ und $s = \sigma + i\omega$ [HeFu01].

2.1.6 Matrix Einzelwertzerlegung

Die *Singular Value Decomposition* (SVD, dt. Einzelwertzerlegung) zerlegt eine reelle Matrix [A] der Dimension $p \times q$ folgendermaßen:

$$[A] = [U][\Sigma][V]^T. \qquad (2.35)$$

Hierbei ist [U] eine $p \times p$ orthogonale Matrix, [V] eine $q \times q$ orthogonale Matrix und [Σ] eine $p \times q$ Diagonalmatrix.
Wenn [A] eine komplexe Matrix der Dimension $p \times q$ ist, dann sieht deren Einzelwertzerlegung folgendermaßen aus:

$$[A] = [U][\Sigma][V]^H. \qquad (2.36)$$

Die Spaltenvektoren von $[U]$ werden Links-Singulärvektoren genannt und die von $[V]$ Rechts-Singulärvektoren. Singulärwerte von $[A]$ werden die positiven Diagonaleinträge von [Σ] genannt. Diese Matrixzerlegung existiert für jede Matrix - auch für nicht-quadratische - und charakterisieren die Eigenschaften einer Matrix (ähnlich den Eigenwerten). Die Singulärwerte und damit auch die Matrix [Σ] sind durch $[A]$ eindeutig bestimmt [HeFu01].

2.2 Sonstige Grundlagen

Die folgende Einführung in die Klassifizierung von Reifen-Eigenformen soll als Grundlage dienen, um die Ergebnisse Kapitel 7.2 interpretieren zu können und ist somit inhaltlich auf wenige Punkte begrenzt. Der interessierte Leser sei auf das weiterführende Werk [WhDK05] verwiesen.

2.2.1 Klassifizierung Reifen-Eigenformen

Die Eigenformen schwingender Reifen weisen charakteristische Merkmale auf, nach denen die Eigenformen benannt werden. Im Folgenden wird die in [Grol13] vorgestellte Konvention nach [Kind09] zur Benennung der Reifen-Eigenschwingungsformen dargestellt. Dies ist eine Erweiterung der Konvention von [WhDK05] durch [Pesc90]. Nach [WhDK05] wird in Abb. 3 zwischen *Clindrical* (Index c) und *Meridional* (Index m) unterschieden:

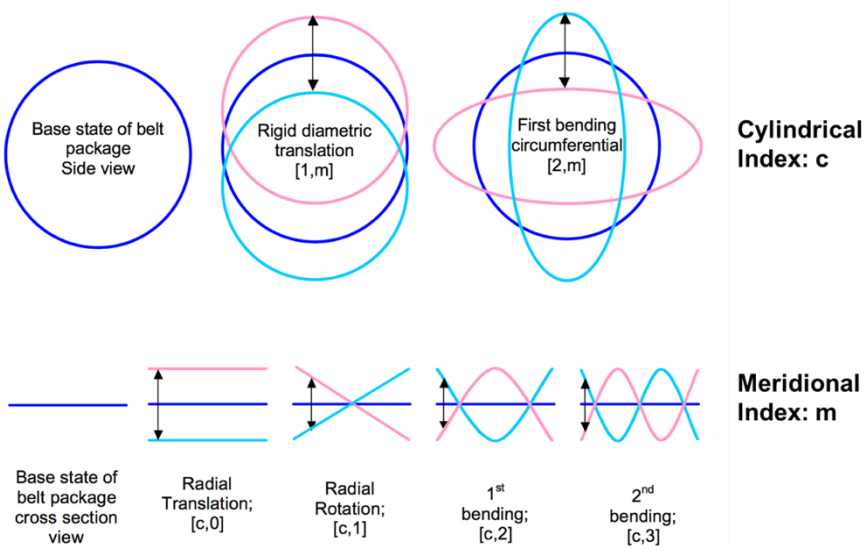


Abb. 3: Benennungskonvention Reifen-Eigenformen nach [WhDK05]

Damit können Reifen-Eigenformen in der Form $[c, m]$ eindeutig beschrieben werden. Eine undeformierte Starrkörperbewegung des Gürtels wird durch $c = 1$ beschrieben. Die Biegemoden des Gürtels werden durch $c = 2,3, \ldots$ beschrieben und gibt die Anzahl der Schwingungsbäuche in radialer Richtung an. Durch den zweiten Index m gibt die Anzahl der Halb-Wellen in Querschnittsrichtung des Reifen an (vereinfacht ist in Abb. 3 nur das Profilband ohne Seitenwand dargestellt).

Aufgrund symmetrischer Eigenformen des unbelasteten, rotationssymmetrischen Reifens, empfiehlt [Grol13], wie in [Kind09] vorgestellt, die Benennungskonvention durch [Pesc90] zu erweitern. Zu den symmetrischen Eigenformen gehört ein doppelter Pol, der durch die Belastung des Reifens und der damit einhergehenden Aufhebung der Reifensymmetrie in zwei einfache Pole zerfällt. Eine dieser resultierenden Eigenformen ist symmetrisch zur Reifenhauptebene durch den Reifenmittelpunkt, währenddessen die andere asymmetrisch zur Reifenhauptebene ist. Zwischen diesen Eigenformen wird unterschieden, indem die Benennungskonvention durch den Zusatz *sym* für symmetrisch und *asym* für asymmetrisch erweitert werden. Damit ebenfalls die $[1, 1]$-Eigenformen eindeutig identifizierbar sind, werden sie um den Zusatz *lateral* bzw. *steering* erweitert.

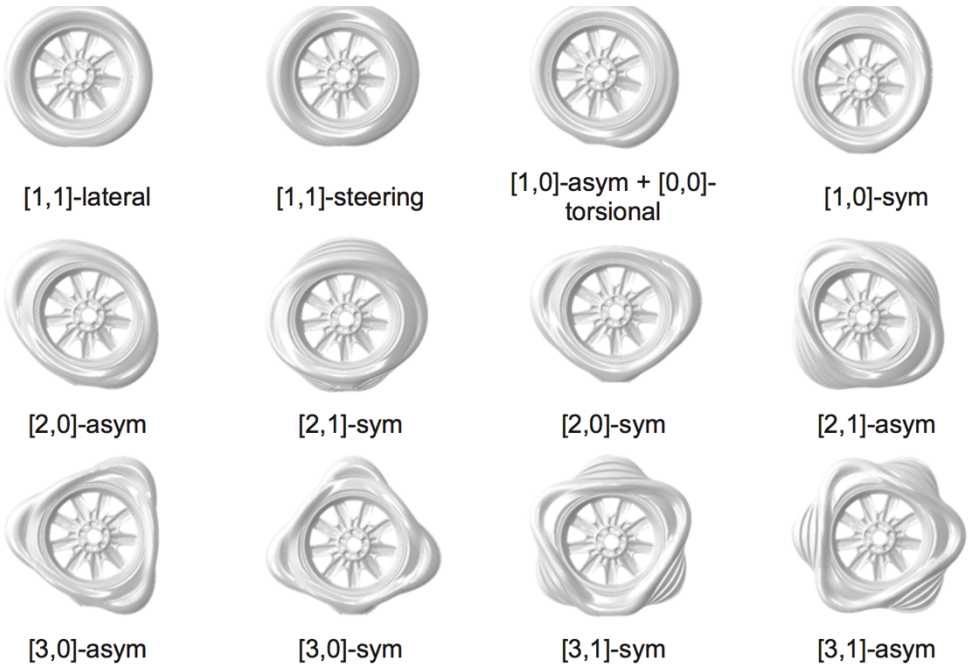


Abb. 4: Reifen-Eigenformen mit Namenskonvention nach [Grol13]

3 Theorie: Multi-Sensor-Layout

In vielen praktischen Anwendungen ist es nicht möglich, alle zu untersuchenden *Degrees of Freedom* (DOF, dt. Freiheitsgrade) mit einem gegebenen Satz an Messsensoren gleichzeitig mit einer einzigen Messung zu erfassen. Entweder ist die Anzahl der zur Verfügung stehenden Sensoren eingeschränkt (z.B. durch hohe Kosten der Sensoren) oder die untersuchte Struktur benötigt aufgrund ihrer komplexen Geometrie eine Vielzahl an darstellbaren DOF. Um trotzdem eine ausreichende räumliche Darstellung der Eigenmodenvektoren auflösen zu können, müssen die aus mehreren Messungen erhaltenen Ergebnisse zu einem einzelnen Ergebnis zusammengeführt werden, das alle untersuchten DOF räumlich abbildet [RaFa14].

Sollen beispielsweise m DOF abgebildet werden, aber nur $N_s < m$ Sensoren zur Verfügung stehen, müssen T Messungen mit unterschiedlich angebrachten Sensoren durchgeführt werden. Sind die das System erregenden (Umgebungs-) Kräfte stationär, können die erhaltenen Messergebnisse und die daraus resultierenden Eigenmodenvektoren direkt, ohne zusätzlichen Aufwand, zusammengeführt werden und somit die Gesamtlösung abbilden. Im Kontext der OMA sind die erregenden Kräfte jedoch häufig nur schwach stationär (vgl. Wind, Verkehr oder Wellen) [Parl03].

Die erhaltenen Ergebnisse können nicht direkt zusammengeführt werden, da die erregenden Kräfte nicht gemessen und die Eigenmodenvektoren nicht auf die modale Masse skaliert werden kann [RaFa14]. Die unskalierten Eigenmodenvektoren können nur zusammengeführt werden, indem N_{ref} Referenzsensoren in den unterschiedlichen Messungen als Referenzen dienen, deren Messpunkte nicht variiert werden. Die übrigen N_{mov} Sensoren werden während jeder Messung bewegt. Nach T Messungen kann somit eine Struktur mit $N_o = N_{ref} + T * N_{mov}$ DOF dargestellt werden. Ein weiterer Grund, warum die einzelnen Ergebnisse der Messungen nicht direkt zusammengeführt werden können ist, dass sich durch das Bewegen der Sensoren zusätzlich die Massenverteilung des Systems, und somit die modalen Parameter, ändern [RDCM09].

Neben dem klassischen *Post Separate Estimation Re-Scaling* Verfahren (PoSER), werden in [Parl03] zwei weitere Verfahren (*Pre Global Estimation Re-Scaling* (*PreGER)* und *Post Global Estimation Re-Scaling* (PoGER)) vorgestellt, die in der praktischen Anwendung gegenüber dem PoSER-Verfahren Vorteile bieten. Neben dem PoSER-Verfahren wird im Folgenden auf das PreGER eingegangen.

3.1 Re-Skalierungsverfahren Multi-Sensor-Layout

3.1.1 PoSER-Verfahren

Der klassische Ansatz, um die Ergebnisse der einzelnen Datensätze zusammenzuführen, wird als *Post Separate Estimation Re-Scaling* Ansatz (PoSER) bezeichnet. Die N_{ref} Referenzsensoren verbleiben während den T Messungen an der gleichen Messstelle, während die übrigen N_{mov} Sensoren bewegt werden. Für jede einzelne Messung werden die modalen Parameter separat von den anderen bestimmt. Daraus resultieren T unterschiedliche Lösungen (Pole und Eigenmodenvektoren), die Eigenformen enthalten jedoch alle DOF der Referenzvektoren. Nach [RaFa14] ist es möglich, eine Beziehung zwischen dem ersten und dem i-ten Datensatz über eine Skalierungskonstante herzustellen [RaFa14]:

$$\{\phi_{ref,1}^k\} = \alpha_{1,i}^k \{\phi_{ref,i}^k\} \qquad (3.1)$$

mit $\{\phi_{ref,1}^k\}$ und $\{\phi_{ref,i}^k\}$ den Teilen des k-ten Eigenmodenvektors an den N_{ref} Referenzsensoren der ersten und i-ten Messung und der Skalierungskonstante $\alpha_{1,i}^k$. Es sollte $N_{ref} \geq 3$ gewählt werden. Die $\alpha_{1,i}^k$ Koeffizienten werden durch ein LS-Verfahren bestimmt. Dieser Zusammenhang zwischen den zu den Referenz-DOF gehörenden Teilen des Eigenmodenvektors der Eigenmode k kann auf die gesamte Schätzung des Eigenmodenvektors übernommen werden. Die Gesamtlösung des Eigenmodenvektors mit N_o DOF zur Eigenmode k ist:

$$\{\phi^k\} = \begin{Bmatrix} \{\phi_{ref,1}^k\} \\ \{\phi_{mov,1}^k\} \\ \alpha_{1,2}^k\{\phi_{mov,2}^k\} \\ \vdots \\ \alpha_{1,i}^k\{\phi_{mov,i}^k\} \\ \vdots \\ \alpha_{1,T-1}^k\{\phi_{mov,T-1}^k\} \end{Bmatrix}. \qquad (3.2)$$

Das PoSER-Verfahren ist aufwendig, da die in den einzelnen Messungen identifizierten Moden eindeutig zu den in den anderen Messungen erhaltenen Moden zugeordnet werden müssen. Dies kann schwierig sein, wenn nicht alle Moden gleich in den Messungen angeregt werden, mehrere Moden nah benachbart sind oder die Referenzsensoren nahe eines Knotens der Eigenmode positioniert sind. In [RDCM09] wird der Einsatz des PoSER-Verfahrens im Kontext der EMA empfohlen, da die hierbei erhaltenen Ergebnisse weniger verrauscht sind und die Modenzuordnung somit einfach ist.

3.1.2 PreGER-Verfahren

Das *Pre Global Estimation Re-scaling* Verfahren (PreGER) führt die PSD-Matrizen der unterschiedlichen Messungen vor der Bestimmung der modalen Parameter zusammen. Die modalen Parameter werden in einem einzigen Prozess identifiziert. Daher ergeben sich keine Probleme hinsichtlich der richtigen Zuordnung der einzelnen Moden wie beim PoSER Verfahren.

Das Zusammenführen der einzelnen Messungen findet direkt bei der Bestimmung der Auto- und Kreuz-PSD statt. In [Parl03] wird beschrieben, dass für jede Messung i $(i = 1, \dots, T)$ die Auto- und Kreuz-PSD-Matrizen zu jeder diskreten (Kreis-) Frequenz in der Matrix $S(\omega)|_i$ folgendermaßen angeordnet werden können:

$$S(\omega)|_i = \begin{bmatrix} S^{mov}(\omega)|_i \\ S^{ref}(\omega)|_i \end{bmatrix}. \qquad (3.3)$$

Hierbei ist $S(\omega)|_i$ eine Matrix der Dimension $(N_{ref} + N_{mov}) \times N_{ref}$ mit den Submatrizen $S^{mov}(\omega)|_i$ und $S^{ref}(\omega)|_i$ der Dimension $N_{mov} \times N_{ref}$ bzw. $N_{ref} \times N_{ref}$. Die Submatrix $S^{mov}(\omega)|_i$ enthält die Kreuz-PSD-Matrix zwischen allen bewegten Sensoren und den Referenzsensoren. Die Submatrix $S^{ref}(\omega)|_i$ enthält die Auto- und Kreuz-PSD-Schätzungen zwischen den Referenzsensoren.

[Parl03] und [RDCM09] beschreiben das Vorgehen des PreGer-Verfahrens folgendermaßen. Die über die Submatrizen $S^{ref}(\omega)|_i$ aller T Messungen gemittelte Matrix ist:

$$S^{ref}(\omega)|_\Sigma = \frac{1}{T} \sum_{k=1}^{T} S^{ref}(\omega)|_k \qquad (3.4)$$

und wird benutzt, um die Submatrizen $S^{mov}(\omega)|_i$ der zugehörigen Messung auf ein mit den anderen Messungen vergleichbares Level zu skalieren:

$$S^{mov}(\omega)|_{i\to\Sigma} = S^{mov}(\omega)|_i \cdot \left(S^{ref}(\omega)|_i\right)^{-1} \cdot S^{ref}(\omega)|_\Sigma. \qquad (3.5)$$

Die erhaltenen Matrizen werden in der gesuchten, vollständig neu skalierten Matrix $S^{res}(\omega)$ angeordnet:

$$S^{res}(\omega) = \begin{bmatrix} S^{ref}(\omega)|_\Sigma \\ S^{mov}(\omega)|_{1\to\Sigma} \\ S^{mov}(\omega)|_{2\to\Sigma} \\ \vdots \\ S^{mov}(\omega)|_{i\to\Sigma} \end{bmatrix} \qquad (3.6)$$

Aus der neu skalierten PSD-Matrix $S^{res}(\omega)$ werden die modalen Parameter in einem einzigen Identifizierungsprozess bestimmt.

Die damit bestimmten Eigenmodenvektoren sind korrekt zusammengesetzt und bilden alle untersuchten DOF ab. Damit ist der r-te Eigenmodenvektor gegeben zu:

$$\phi_r = \begin{bmatrix} \phi_r^{ref}|_{\Sigma} \\ \phi_r^{mov}|_{1\to\Sigma} \\ \phi_r^{mov}|_{2\to\Sigma} \\ \vdots \\ \phi_r^{mov}|_{T\to\Sigma} \end{bmatrix}. \qquad (3.7)$$

Es soll angemerkt werden, dass zur Neuskalierung die gemittelte Matrix $S^{ref}(\omega)|_{\Sigma}$ verwendet wird. Dies ist nötig, da bei der Neuskalierung die PSD-Matrix der jeweiligen Messungen der bewegten Sensoren durch deren PSD-Matrix der Referenzsensoren geteilt wird: $S^{mov}(\omega)|_i \cdot \left(S^{ref}(\omega)|_i\right)^{-1}$. Dadurch geht die Information über die Extremstellen (Pole) des Systems verloren. Durch Multiplikation mit einer einheitlichen PSD-Matrix $S^{ref}(\omega)|_{\mathrm{k}}$ mit $k \in \{1, \dots, T\}$ der Referenzsensoren, wird die Information in der neu skalierten Matrix wiederhergestellt [Parl03]. Diese Referenz-PSD enthält also die Information der Pole des Systems. [Parl03] empfiehlt hierbei jedoch die über die Submatrizen $S^{ref}(\omega)|_i$ aller T Messungen gemittelte Matrix $S^{ref}(\omega)|_{\Sigma}$ (siehe Gleichung (3.5)), anstatt eine einzelne Referenz-PSD $S^{ref}(\omega)|_{\mathrm{k}}$ einer einzelnen Messung, zu verwenden. Damit wird ein besseres Signal-Rausch-Verhältnis erreicht, was die Ergebnisgüte verbessert.

Das PoGER-Verfahren kann auch in angepasster Form (gleiche Operationen, aber anstatt PSD werden Korrelationsfunktionen verwendet) im Zeitbereich verwendet werden. Dieses Äquivalent kann jedoch nicht für nicht-stationäre Anregung verwendet werden, wodurch in [RDCM09] das in [MBBG02] vorgestellt Verfahren für den Zeitbereich empfohlen wird.

3.2 Bewertung Multi-Sensor-Verfahren

Das PoSER-Verfahren liefert die besten Ergebnisse mit den am ehesten vollständigen Datensätzen modaler Parameter. Jedoch ist das Verfahren wesentlich aufwendiger als das PreGER-Verfahren und kann nur schwer automatisiert werden. Da beim PreGER-Verfahren keine vorherige Zuordnung der durch die einzelnen Messungen bestimmten Eigenmoden nötigt ist, kann dieses Verfahren einfach automatisiert werden. Es ist nur ein einzelner globaler Identifikationsprozess der modalen Parameter nötig ist. Allerdings werden hierbei weniger Moden als beim PoSER-Verfahren bestimmt und die Güte der Schätzungen der modalen Parameter ist geringer. Dies ist darauf zurückzuführen, dass das PreGER-Verfahren sensibler auf Nicht-Stationarität (z.B. Temperaturschwankungen zwischen den einzelnen Messungen) zwischen den einzelnen Messungen reagiert und somit größere Abweichungen auftreten bzw. das PoSER-Verfahren robuster gegenüber Nicht-Stationarität ist [RDCM09].

4 Theorie: Operational Modal Analysis (OMA)

Wie bereits in der Einleitung dargestellt, wurde die *Operational Modal Analysis* (OMA, dt. Betriebsmodalanalyse) aus der EMA entwickelt. Motiviert wurde dies, da die kontrollierte und harmonische Anregung des Systems häufig schwierig bzw. überhaupt nicht messbar ist. Klassische EMA ist außerdem meist an Versuchsumgebungen in Laboren gebunden, die keine realen Betriebszustände abbilden [AnBZ00]. Diese Beschränkungen wirken sich insbesondere auf Untersuchungen im Bauingenieurwesen aus. Im Bauingenieurwesen gestaltet sich die Anregung schwierig, da die anzuregenden Strukturen groß sind (z.B. Hochhaus), nicht quantifiziert werden kann (z.B. Anregung durch Wind) und Versuche in Laboren nicht möglich sind [RaFa14].

Allgemein wird unter dem Begriff OMA die Gruppe der modalen Identifikationsverfahren verstanden, die nur auf Messungen der Schwingungsantwort des angeregten Systems basieren. Weitere gängige Bezeichnungen für OMA sind nach [RaFa00]:

- *Output-only Modal Analysis*,
- *Ambient Modal Analysis* und
- *Natural-Exication Modal Analysis*.

Im Vergleich zur EMA wird die Anregung des Systems nicht gemessen, sondern als zufällig angenommen und die natürlichen Anregungen der Umgebung bzw. des Betriebszustandes zur Systemanregung genutzt (z.B. Wind, das ein Hochhaus anregt, Reifen der auf Fahrbahn abrollt). OMA ist somit das stochastisches Gegenstück zur deterministischen EMA [RaFa00].

Neben dem Bauingenieurwesen findet OMA inzwischen auch in weiteren Ingenieursbereichen, wie der Raumfahrt oder dem Maschinenbau, verbreitete Anwendung. Für diese Ingenieursbereiche ergeben sich ebenfalls weitreichende Vorteile. [Caub04] und [AnBZ00] heben besonders hervor, dass OMA unter realen Betriebszuständen gemessen wird, die den späteren Betriebszuständen entsprechen. Realistischere modale Modelle können somit identifiziert werden. Die bestimmten modalen Parameter sind repräsentativ für die Struktur im Betriebszustand [RaFa00]. EMA ist hingegen an die Versuchsumgebung in Laboren gebunden, die sich meist stark von den tatsächlichen Betriebsbedingungen unterscheidet. Weitere in [AnBZ00] erwähnte Vorteile sind:

- OMA ist preiswert und einfach in der Durchführung, da keine aufwendige Ausrüstung zur Anregung benötigt wird. Vorhandene Umgebungskräfte werden zur Anregung genutzt.
- OMA setzt immer MIMO-Methoden voraus (siehe Kapitel 4.3.1), wodurch nah benachbarte Moden besser voneinander getrennt werden.
- Nutzbar für weitere Anwendungen wie die vibrationsbasierte Überwachung und Schadenserkennung von Strukturen.

Die meisten Algorithmen der OMA wurden auf Grundlage der bestehenden und erprobten EMA-Methoden weiterentwickelt. Eine Übersicht der wichtigsten OMA-Methoden ist in Kapitel 4.3 zu finden. Die BFD- und FFD-Methode wird hier zusätzlich zum in Kapitel 5 beschriebenen p-LSCF -Algorithmus vorgestellt. Die Anwendung der OMA bedarf weitreichender Voraussetzungen an die untersuchte Struktur und die Form der Anregung. Diese werden in Kapitel 4.1 eingeführt. Möglichkeiten zur Kontrolle und Bewertung der Ergebnisse werden in Kapitel 4.5 beschrieben. Für einen kurzen Einblick in die geschichtliche Entwicklung der OMA sei der interessierte Leser auf [RaFa14] und [AnBZ00] verwiesen.

4.1 Voraussetzungen zur OMA

4.1.1 Voraussetzungen an das System

Die OMA identifiziert die modalen Parameter auf Grundlage der messtechnisch erfassten Schwingungsantwort. Die Anregung des Systems wird hingegen nicht gemessen. Die daraus resultierenden Einschränkungen der modalen Parameteridentifikation können bei Erfüllung bestimmter Voraussetzungen überkommen werden. Die nachfolgenden Anforderungen an das zu untersuchende System sind aus [RaFa14] entnommen und müssen erfüllt sein:

- *Linearität*: Die Antwort des Systems auf eine gegebene Kombination von Anregungen ist gleich der Überlagerung aller Systemantworten der einzelnen Anregungen. Superposition des Systems ist möglich.
- *Stationarität*: Die dynamischen Eigenschaften der Struktur ändern sich nicht mit der Zeit, sodass die Koeffizienten der Differentialgleichungen, die das dynamische des Systems beschreiben, zeitunabhängig sind.
- *Beobachtbarkeit*: Das Sensor-Layout zur Datenmessung wurde ordnungsgemäß entworfen, um die interessierenden Moden beobachten zu können (z.B. Messungen in Knotenpunkten werden vermieden).

4.1.2 (Milde) Voraussetzungen an die Anregung

Da die Anregung des Systems nicht kontrolliert verändert werden kann, sondern von ungemessenem, stochastischem Input angeregt wird, müssen zusätzliche Annahmen über die Anregung gemacht werden. Es wird angenommen, dass die Kovarianz zwischen zwei Zeitproben nur von der Zeitdifferenz abhängt und nicht von den Zeitpunkten, zu denen die Messwerte bestimmt wurden. Die Anregung ist also (schwach) stationär. Außerdem wird vorausgesetzt, dass die Anregung im (quadratischen Mittel) ergodisch ist. Bei Nichterfüllung dieser Annahmen wird jedoch lediglich der Schätzungsfehler des untersuchten Systems erhöht, die Methoden zur modalen Parameterbestimmung sind dennoch gültig. Daher werden sie auch als milde Annahmen bezeichnet [Reyn12].

OMA setzt zusätzlich voraus, dass die Struktur an mehreren Anregungspunkten gleichzeitig angeregt wird. Daher basiert OMA nur auf *Multiple-Input-Multiple-Output* Methoden (MIMO). Je mehr unterschiedliche Punkte angeregt werden, desto besser können nah benachbarte Moden unterschieden werden [Trép17] [AnBZ00].

4.1.3 Voraussetzung: Weißes Rauschen

Grundlegende Annahme der OMA ist, dass das System durch *weißes (gaußsches) Rauschen* angeregt wird. Unter *weißem Rauschen* wird ein stochastisches Signal mit gleicher Intensität über das gesamte Frequenzband verstanden. Die Auto-Korrelationsfunktion des *Continuous-time White Noise* (CWN, dt. Zeit-kontinuierlichen weißen Rauschens) R_{yy} ist definiert als:

$$R_{yy}(\tau) = \mathcal{I}\delta(\tau). \qquad (4.1)$$

$\mathcal{I}$ ist die Intensität des CWN und $\delta(\tau)$ die Dirac-Funktion (Stoßfunktion). Daraus resultiert folgende PSD eines durch CWN angeregten System:

$$S_{yy}(s) = \mathcal{I}. \qquad (4.2)$$

Diese PSD ist realwertig und konstant. Bei der Systemanregung durch weißes Rauschen ist das Eingangsspektrum (Eingangs-PSD-Matrix) daher ebenfalls konstant [PeVa00]. Es werden alle Moden gleich angeregt. Die Ausgangs-PSD beinhaltet alle Informationen über das System. Wie in [Reyn12] dargestellt, entspricht die Zeit-kontinuierliche PSD-Matrix der diskreten PSD-Matrix. Die erhaltenen Resultate können somit in den diskreten Zeitbereich übertragen werden. Ist der Mittelwert gleich Null, wird das Rauschen auch als „weißes gaußsches Rauschen“ bezeichnet.

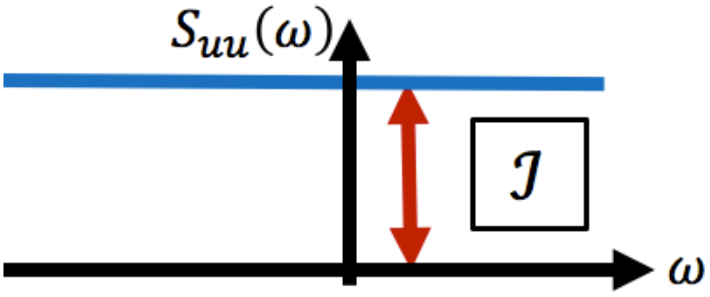


Abb. 5: PSD des weißen Rauschens mit der konstanten Intensität $\mathcal{I}$

In der praktischen Anwendung von OMA kann die Anregung nicht immer als reines weißes Rauschen angenommen werden. Zusätzliche harmonische Anteile treten beispielsweise unweigerlich bei sich rotierenden Komponenten auf [Trép17]. Streng genommen können in diesen Fällen OMA-Methoden nicht angewandt werden, da durch die harmonischen Anteile virtuelle Eigenmoden entstehen, die keine Eigenschaft des Systems widerspiegeln [Moha05].

Nach [Moha05] kann in diesem Fall jedoch angenommen werden, dass die harmonische Schwingungsantwort zu einer virtuellen Eigenmode ohne Dämpfung gehört und die Erregung daher trotz allem reines weißes Rauschen ist. Somit können OMA-Methoden dennoch angewandt werden. Die virtuellen Moden treten dann neben den eigentlichen Eigenmoden der Struktur auf und werden durch die OMA-Methoden ebenfalls identifiziert. Es kann zu Problemen kommen, wenn virtuelle und natürliche Eigenmode nah beieinanderliegen. Die Moden können dann nur schwer voneinander separiert werden und üben gegenseitigen Einfluss aufeinander aus, wodurch die modalen Parameter nicht genau bestimmt werden. [PAVG07] gibt zusätzlich an, dass die Voraussetzung der reinen Erregung durch weißes Rauschen in der Praxis gelockert werden kann, solange das Eingangsspektrum einigermaßen flach ist.

4.1.4 Vorteile der PSD für OMA im Frequenzbereich

Die PSD beschreibt die Energieverteilung eines Signals über das Frequenzband. Die Darstellung der PSD eignet sich insbesondere zur Analyse stochastischer Signale. Im Kontext der OMA wird stets ein (schwach) stationäres, stochastisches Eingangssignal in Form des weißen Rauschens vorausgesetzt.

Da ein stochastisches Eingangssignal aus nichtperiodischen Komponenten bestehen, erweist sich die klassische Fourieranalyse als unpraktisch. Nach [Natk88] ist die Integrabilitätsbedingung Formel (5.1) für einen stationär, stochastischen Prozess $x(t)$ nicht erfüllt. Die Fouriertransformation von $x(t)$ existiert also nicht. Diese wäre auch aufgrund der fehlenden explizierten mathematischen Beschreibung von $x(t)$ sowieso praktisch nicht möglich. Die finite Fouriertransformierte existiert jedoch. Dabei wird allerdings das zeitbegrenzte Signal automatisch außerhalb des Intervalls $[0, T]$ periodisch fortgesetzt. Falls das Intervall hierbei so gewählt wird, dass $x_k(0) \neq x_k(T)$ gilt, entsteht durch die periodische Fortführung an den Stellen nT ein Signalsprung. Damit wird das Frequenzspektrum verfälscht. Dieser auch als Abschneideeffekt (engl. leakage effect) bekannte Fehler wird durch das Verwenden einer Bewertungsfunktion (Fensterung) vermindert [Natk88].Nach [RaFa14] wird durch die Fensterung eine periodische Funktion gebildet, indem die Diskontinuitäten am Anfang und Ende des Datensatzes unterdrückt werden. Für eine detaillierte Beschreibung des Abschneideeffekts und die Auswirkungen der unterschiedlichen Fensterfunktionen sei auf [Natk88], [RaFa14] und [HeLS07] verwiesen. Durch die Bestimmung der PSD über das Korrelogramm-Verfahren (siehe Kapitel 2.1.4) bedarf es keiner Fensterung der FFT und der damit einhergehenden Bias wird vermieden. Das Korrelogramm-Verfahren bestimmt die PSD auf Basis der Korrelationsfunktionen, für die die Integrabilitätsbedingung nach Formel (5.1) stets erfüllt ist [Natk88]. Daher kann die klassische Fouriertransformation für periodische Funktion angewandt werden. [JKPV15] zeigt eine Möglichkeit zur Bestimmung auf Basis der Korrelationsfunktionen. Die Vermeidung einer Fensterung erhöht die numerische Güte der Schätzung der PSD. Daher erweist sich die Analyse der PSD für OMA als praktisch.

Die PSD kann auch mithilfe der finiten Fouriertransformierten des Signals geschätzt werden (siehe Gleichungen (2.27) bis (2.29)). Diese Schätzung nach dem Periodogramm-Verfahren stellt jedoch keine unverzerrte Schätzung dar und ist nicht konsistent. Somit kann aus ihr nicht auf die Gesamtheit der stochastischen inneren Zusammenhänge geschlossen werden, da sie nur auf einer einzigen Realisierung des Prozesses basiert.

4.1.5 Zusammenhang FRF und PSD

Es lässt sich folgender Zusammenhang zwischen der PSD-Matrix und der FRF-Matrix (siehe Kapitel 4.2.1) eines Systems herstellen:

$$\left[S_{yy}(\omega)\right] = [H(\omega)][S_{uu}][H(\omega)]^H. \qquad (4.3)$$

Hierbei ist $\left[S_{yy}(\omega)\right]$ die PSD-Matrix des Signalausgangs (Ausgangs-PSD), $[S_{uu}(\omega)]$ die PSD-Matrix des Signaleingangs (Eingangs-PSD) und $[H(\omega)]$ die FRF-Matrix. Im Kontext von OMA wird angenommen, dass der Input als weißes gaußsches Rauschen beschrieben werden kann. Aus Gleichung (4.2) folgt nun, dass die PSD des weißen gaußsches Rauschens eine Konstante ist. Somit enthält die Ausgangs-PSD-Matrix dieselben Informationen über das System, wie die FRF-Matrix des Systems und enthält ebenfalls die gesamte Information über die modalen Parameter des Systems. Dies bildet die grundlegende Annahme aller OMA-Methoden [RaFa14].

4.2 Modelle zur Beschreibung des modalen Modells

4.2.1 Frequency Response Function (FRF)

Die Bewegungsgleichung eines MDOF wird im Zeitbereich folgendermaßen beschrieben:

$$[M]\{\ddot{y}(t)\} + [C]\{\dot{y}(t)\} + [K]\{y(t)\} = \{f(t)\} \qquad (4.4)$$

Die Vektoren $\{\ddot{y}(t)\}$, $\{\dot{y}(t)\}$ und $\{y(t)\}$ sind die Beschleunigungs-, Geschwindigkeits- und Verschiebungsvektoren. $[M]$, $[C]$ und $[K]$ beschreiben die Massen-, Dämpfungs- und Steifigkeitsmatrizen. $\{f(t)\}$ ist der Kraftvektor. Dies ist eine lineare Differentialgleichung zweiter Ordnung, die das dynamische Verhalten eines schwingungsfähigen Systems vollständig beschreibt. Üblicherweise sind die Bewegungsgleichungen für ein MDOF-System gekoppelt. Sie können unter Annahme proportionaler Dämpfung (z.B. Bequemlichkeitshypothese erfüllt) entkoppelt und durch Lösen des erhaltenen Eigenwertproblems gelöst wird. Anschließend werden die einzelnen Lösungen zur Gesamtlösung superpositioniert [RaFa14]. Gleichung (4.4) kann durch Fouriertransformation im Frequenzbereich dargestellt werden:

$$(-\omega^2[M] + i\omega[C] + [K])\{Y(\omega)\} = \{F(\omega)\}. \qquad (4.5)$$

$\{Y(\omega)\}$ und $\{F(\omega)\}$ sind die Fouriertransformierten der Schwingungsantwort $\{y(t)\}$ bzw. der Anregung $\{f(t)\}$. Das Verhältnis der Fouriertransformierten des Inputs und des Outputs wird als *Frequency Response Function* (FRF, dt. Übertragungsfunktion) bezeichnet:

$$H(\omega) = \frac{\{Y(\omega)\}}{\{F(\omega)\}}. \qquad (4.6)$$

Die FRF beschreibt das dynamische Verhalten vollständig [RaFa14].

4.2.2 Pole-Residue Model (PRM)

Das in Gleichung (4.6) beschrieben Modell auf Basis der FRF-Matrix kann mithilfe einer Partialbruchzerlegung durch seine modalen Parameter beschrieben werden [RaFa00]:

$$[H(\omega)] = \sum_{r=1}^{N_m} \left(\frac{[R_j]}{i\omega_f - \lambda_r} + \frac{[R]_j^*}{i\omega_f - \lambda_r^*} \right) = \sum_{r=1}^{N_m} \left(\frac{Q_r \{\phi_r\}\{\phi_r\}^T}{i\omega_f - \lambda_r} + \frac{Q_r{}^* \{\phi_r\}^* \{\phi_r\}^{*T}}{i\omega_f - \lambda_r^*} \right). \qquad (4.7)$$

Die Anzahl der Moden wird durch N_m beschrieben. Diese Darstellung wird Pole-Residue-Modell (PRM) genannt. Als Residue wird hierbei die komplexe Matrix $[R_j]$ bezeichnet. Sie beinhaltet die Information über die Eigenmoden $\{\phi_r\}$:

$$[R_j] = Q_r \{\phi_r\}\{\phi_r\}^T. \qquad (4.8)$$

Die Pole bestimmen die Eigenfrequenzen und das Dämpfungsmaß des beschriebenen Systems. Gleichung (4.7) beschreibt ein Modell mit zwei komplex-konjugierten Polen je Eigenmode [RaFa14].

Durch den Zusammenhang zwischen FRF und Ausgang-PSD nach Gleichung (4.3), kann das Pole-Residue-Modell auch für die Ausgangs-PSD-Matrix beschrieben werden [RaFa14]:

$$\left[S_{yy}(\omega)\right] = \sum_{r=1}^{N_m} \left(\frac{\{\phi_r\}\{\gamma_r\}^T}{i\omega_f - \lambda_r} + \frac{\{\phi_r\}^*\{\gamma_r\}^H}{i\omega_f - \lambda_r^*} + \frac{\{\gamma_r\}\{\phi_r\}^T}{-i\omega_f - \lambda_r^*} + \frac{\{\gamma_r\}^*\{\phi_r\}^H}{-i\omega_f - \lambda_r^*} \right) \qquad (4.9)$$

mit $\{\gamma_r\}$ als Betriebsreferenzvektor, der zur $r - ten$ Mode gehört. Die Residue-Matrix beschreibt ebenfalls die Eigenformen, während die Eigenfrequenzen und Dämpfungsmaße durch die Pole beschrieben werden. In [RaFa00] wird die Herleitung von (4.9) mithilfe der Heaviside-Partialbruchzerlegung dargestellt. Aufgrund der 4-Quadraten-Symmetrie (vgl. komplexer Einheitskreis) [RaFa14] der doppelseitigen PSD-Matrix $\left[S_{yy}(\omega)\right]$, existieren je Eigenmode vier komplex-konjugierte Pole ($\lambda_r,\ \lambda_r^*, -\lambda_r, -\lambda_r^*$). Es sei angemerkt, dass bei OMA keine Informationen über den Input zur Verfügung stehen, wodurch die erhaltenen Eigenformen nicht auf die modale Masse normiert werden können. Somit können nur unskalierte Eigenmoden bestimmt werden [RaFa14].

4.2.3 Matrix Fraction Description (MFD)

Die *Matrix Fraction Description* (MFD, dt. Matrix-Bruch-Beschreibung), beschreibt das modale Modell als das Verhältnis zweier Matrix-Polynome. Damit kann die FRF-Matrix als *linke MFD* (LMFD) beschrieben werden zu:

$$[H(\omega)] = [A_L(\omega)]^{-1}[B_L(\omega)] \qquad (4.10)$$

bzw. als *rechtes MFD* (RMFD) zu:

$$[H(\omega)] = [B_R(\omega)][A_R(\omega)]^{-1}. \qquad (4.11)$$

mit $A_{L/R}(\omega)$ und $B_{L/R}(\omega)$ als Matrix-Polynome [RaFa14].

Die MFD findet populäre Anwendung als Common-Denominator-Model (CDM) (engl. für Gemeinsamer-Nenner-Modell) bei der LSCF-Methode. Das CDM als RMFD beschreibt eine PSD-Matrix als einen Bruch, bei dem der Zähler ein Polynom ist, während der Nenner ein gemeinsames Polynom ist:

$$\mathrm{G}_k(\omega_\mathrm{f}) = \frac{\sum_{\mathrm{j}=0}^{\mathrm{n}} \mathrm{N}_{\mathrm{k,j}}\, \Omega_\mathrm{f}^{\mathrm{j}}}{\sum_{\mathrm{j}=0}^{\mathrm{n}} \mathrm{d}_\mathrm{j}\, \Omega_\mathrm{f}^{\mathrm{j}}}. \qquad (4.12)$$

Ω_f ist die Polynom Basisfunktion, die frei gewählt werden kann und n die gewählte Modellordnung. Die Beschreibung eines diskreten Modells in der z-Domain empfiehlt sich im Sinne der numerischen Qualität [RaFa14].

4.3 Bestehende OMA-Methoden

4.3.1 Klassifizierung

Es gibt eine Vielzahl an OMA-Methoden, die in der Praxis Anwendung finden. Viele der OMA-Methoden haben sich aus bestehenden Methoden der EMA entwickelt, wobei der mathematische Hintergrund an die Anforderungen der stochastischen Beschreibung der Erregung der OMA angepasst wurden. Die im Folgenden dargestellten Klassifizierungen sind [RaFa14] entnommen.

Gegenüber den EMA Methoden ist erwähnenswert, dass eine Einteilung der OMA-Methoden nach Anzahl der Eingangssignale nicht möglich ist: OMA-Methoden sind immer *Multiple Input*, benötigen also mehrere Eingangssignale an unterschiedlichen Stellen.
Dies ist bedingt durch die Grundannahme der stochastischen Erregung durch weißes Rauschen. In der Praxis sind die erhaltenen Ergebnisse umso besser, je mehr unterschiedliche Stellen erregt werden [Trép17].

Die wichtigste Unterscheidung bzw. Einteilung der OMA-Methoden besteht darin, ob die Lösung über den Frequenzbereich oder direkt über den Zeitbereich erhalten wird. Bei der Lösung über

den Zeitbereich werden direkt die Zeitreihen-Daten bzw. deren Korrelationsfunktionen verarbeitet, während bei der Lösung im Frequenzbereich die Zeitreihen-Daten zuerst in den Frequenzbereich durch FFT bzw. im Falle von OMA in Form der PSD transformiert wird. Es besteht jederzeit die Möglichkeit, vom Zeit- in den Frequenzbereich und umgekehrt zu transformieren.

Außerdem muss zwischen parametrischen und nicht-parametrischen Methoden unterschieden werden. Parametrische Methoden versuchen durch den Aufbau eines parametrischen Modells, die durch die Messdaten repräsentierten physikalischen Eigenschaften auf dieses Modell abzubilden. Da die Ordnung des abzubildenden Modells normalerweise nicht a priori bekannt ist, muss die Modellordnung geschätzt werden. Dafür wird das sogenannte Stabilisierungsdiagramm konstruiert, mit dem die Modellordnung iterativ bestimmt wird (siehe hierzu Kapitel 4.4). Es werden Curve-Fitting-Verfahren eingesetzt, um ein möglichst realitätsnahes Modell aufzubauen. Diese Verfahren sind wesentlich rechenaufwendiger als nicht-parametrische Methoden. Dafür ist die Güte der erhaltenen Resultate höher. Außerdem bestehen vielfältige, meist leicht zu implementierende, Möglichkeiten zur Automatisierung des Prozesses. Die nicht-parametrischen Methoden sind demgegenüber leicht und schnell anwendbar. Sie liefern ohne großen Rechenaufwand schnelle Ergebnisse und sind daher besonders geeignet, um in Feldversuchen erste Ergebnisse zu liefern, die sofort interpretierbar sind. Sie dienen zur ersten Kontrolle von z.B. Messungen oder um weitere Maßnahmen aufgrund der beobachteten Ergebnisse durchzuführen.

Des Weiteren wird zwischen *Single-DOF*- und *Multi-DOF*-Methoden unterschieden. Single-DOF-Methoden setzten die Annahme voraus, dass nur eine Mode in einer vorgegebenen Bandbreite dominant ist. Es wird angenommen, dass die Schwingungsantwort nur durch diese eine dominante Mode bestimmt wird, wodurch deren Parameter separat von den anderen Moden bestimmt werden kann. Quasi eine Einzelbetrachtung des Systems. Die Güte der Ergebnisse hängt hierbei stark davon ab, inwiefern die Moden des Systems tatsächlich getrennt voneinander auftreten. Um nah benachbarte oder sogar übereinstimmende Moden korrekt identifizieren zu können, werden Multi-DOF-Methoden voraussetzt. Diese Überkommen die Nachteile der Single-DOF-Methoden, da sie nicht voraussetzen, dass jeweils nur eine Mode dominant ist. Nichtsdestotrotz ist es selbst für Multi-DOF -Methoden schwierig, nach benachbarte Moden unterscheiden zu können.

Manche Methoden werden außerdem als Zwei-Schritt-Verfahren bezeichnet. Dies lässt sich darauf zurückführen, dass diese Methoden in einem ersten Schritt die Eigenfrequenzen und Dämpfungsmaße bestimmen. In einem anschließenden zweiten Schritt werden auf Grundlage der im ersten Schritt bestimmten Parameter die dazugehörigen Eigenmoden bestimmt. Klassische Ein-Schritt-Verfahren bestimmen dem hingegen alle Parameter in nur einem Schritt.

4.3.2 Übersicht OMA-Methoden

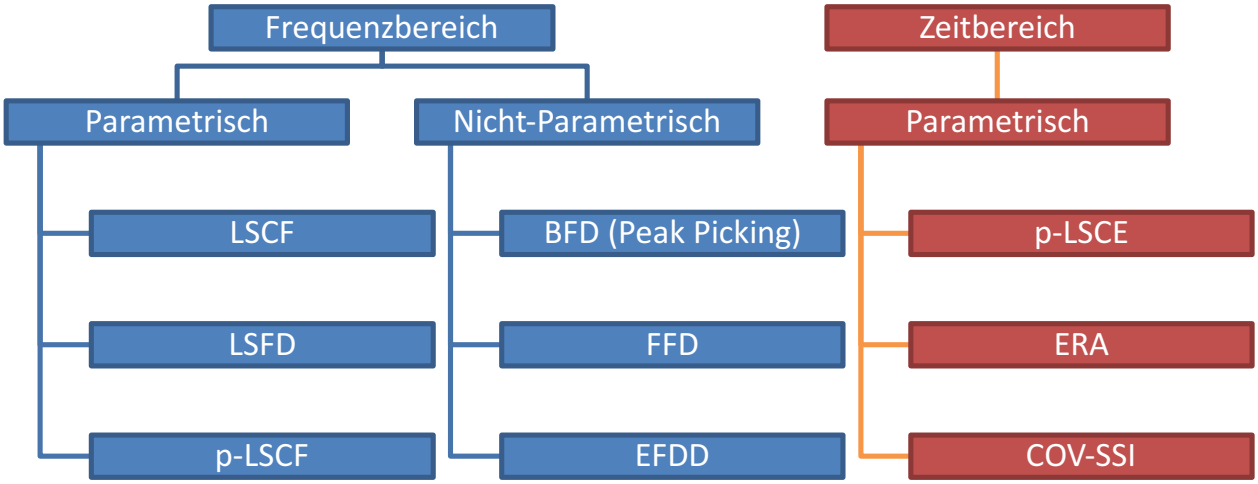


Abb. 6: Übersicht bekannter OMA-Methoden

Im Folgenden werden die nicht-parametrischen Methoden BFD und FFD besprochen. Diese wurden exemplarisch ausgewählt, da sie einen leicht verständlichen Zugang zu den OMA-Methoden bieten. Die p-LSCF-Methode ist die im Rahmen der Bachelorarbeit implementierte OMA-Methode und wird in Kapitel 5 ausführlich besprochen.

4.3.3 Basis Frequency Domain (BFD)

Die *Basis Frequency Domain* Methode (BFD), auch geläufig als *Peak Picking*-Methode, ist die einfachste OMA-Methode im Frequenzbereich. Ausgangssituation ist wie bei den meisten Methoden im Frequenzbereich die PSD-Matrix. Aufgrund der einfachen Implementierung und dem geringen damit verbundenen Rechenaufwand, war die BFD-Methode insbesondere in der Vergangenheit äußert populär. Sie wird als Single-DOF-Methode klassifiziert. Dies bedeutet, dass jeweils nur eine dominante Mode in der Resonanz aktiv ist. Durch diese Annahme ist es möglich, die Pole-Residue-Darstellung der PSD-Matrix folgendermaßen zu vereinfachen [RaFa14]:

$$[G_{YY}(\omega)] = G_{P_rP_r}\{\phi_r\}\{\phi_r\}^H. \qquad (4.13)$$

Hierbei ist $\{\phi_r\}$ der zur dominanten Eigenmode dazugehörige Eigenvektor und $G_{P_rP_r}\{\phi_r\}$ die PSD-Matrix der sogenannten modalen Koordinate. Die modale Koordinate $p_r(t)$ jeder Eigenmode beschreibt hierbei den Zusammenhang zwischen der Systemantwort und dem Eigenmodenvektor:

$$\{y(t)\} \approx \{\phi_r\}p_r(t). \qquad (4.14)$$

Nach [RaFa14] ist jede Zeile der PSD-Matrix im Resonanzfall ($\omega_r = \omega_f$) eine Schätzung des aktuellen Eigenmodenvektors. Dies ist darin begründet, dass für die PSD-Matrix $[G_{YY}(\omega)]$ in Gleichung (4.13) $rang([G_{YY}(\omega)]) = 1$ gilt. Der BFD-Algorithmus ist daher so aufgebaut, dass zuerst

die Resonanzfrequenzen bestimmt werden. Zur Bestimmung dieser Frequenzen wird die Summe aller Auto-PSD-Matrizen aller Ausgangskanäle gebildet:

$$\left[G_{YY,Gesamt}(\omega)\right] = \sum_{y=1}^{l} \left[G_{yy}(\omega)\right] \qquad (4.15)$$

Hierbei müssen nun die Maximalstellen der Auto-PSD $\left[G_{YY,Gesamt}(\omega)\right]$ identifiziert werden. Es ist ratsam, die Matrix $\left[G_{YY,Gesamt}(\omega)\right]$ über das Frequenzband f als Schaubild darzustellen und beispielsweise eine optische Wahl der Resonanzfrequenzen zu treffen. Automatisierte Verfahren zur Detektion der Resonanzstellen sind meist nur nach vorheriger Eingabe des Nutzers hinsichtlich deren Anzahl, ungefähre Resonanzfrequenz etc. möglich. Die r Resonanzstellen entsprechen den r Eigenfrequenzen des untersuchten Systems.

Wurden die r Eigenfrequenzen identifiziert, kann aus der zur r-ten Eigenfrequenzen gehörenden PSD-Matrix $[G_{YY}(\omega = \omega_r)]$ der r-te Eigenmodenvektor geschätzt werden. Dieser entspricht einer der Spalten der PSD-Matrix. Hierfür muss ein Referenzsensor gewählt werden, zu dem die Auto- und Kreuz-PSD-Matrix gebildet wird (siehe hierzu Abb. 2). Die Spalte der PSD-Matrix entspricht also dem Eintrag der Auto-PSD des Referenzsensors sowie den individuellen Kreuz-PSDs zwischen dem Referenzsensor und den anderen Sensoren. Es muss darauf geachtet werden, dass der Referenzsensor so gewählt wurde, dass möglichst alle Moden mit dem Referenzsensor dargestellt werden können. Befindet sich der Referenzsensor nahe eines Knotens der Eigenmoden, kann dieser Sensor nicht als Referenzsensor gewählt werden. Daher kann es nötig sein, für verschiedene Eigenmoden unterschiedliche Referenzsensoren auszuwählen [RaFa14].

Als Ergebnis liefert die BFD-Methode die Eigenfrequenzen und die dazugehörigen Eigenmoden. Bei der Anwendung sollte jedoch darauf geachtet werden, dass das untersuchte System möglichst schwach gedämpft ist und nur ungekoppelte Eigenmoden auftreten.

4.3.4 Frequency Domain Decomposition (FDD)

Wie die BFD-Methode ist auch die *Frequency Domain Decomposition* Methode (FDD) eine nichtparametrische OMA-Methode im Frequenzbereich. Die Grundlage der FDD Methode ist die *Complex Mode Indicator Function* Methode (CMIF) der EMA, die auf PSD-Daten erweitert wurde. Der Vorteil der FDD-Methode gegenüber der BFD-Methode ist, dass die FDD-Methode auch nah benachbarte Moden identifizieren kann und somit auch für stärker gedämpfte Strukturen anwendbar ist. Die theoretische Grundlage der FDD-Methode ist durch Gleichung (4.14) beschrieben, die bei Betrachtung der gesamten Modalmatrix beschrieben wird zu:

$$\{y(t)\} = [\phi]\{p(t)\} \qquad (4.16)$$

mit der Modalmatrix $[\phi]$ und dem Vektor der modalen Koordinaten $\{p(t)\}$. Dadurch ist auch folgender Zusammenhang der Ausgangs-PSD-Matrix $[G_{YY}(\omega)]$ und PSD-Matrix der modalen Koordinaten $\left[G_{pp}(\omega)\right]$ gegeben:

$$[G_{YY}()] = [\phi]\left[G_{pp}(\omega)\right][\phi]^H. \qquad (4.17)$$

Durch eine *Singular Value Decomposition* (SVD) (siehe Kapitel 2.1.6) kann die Ausgangs-PSD-Matrix $[G_{YY}(\omega)]$ bei jeder Frequenz ω zerlegt werden:

$$[G_{YY}(\omega)] = [U][\Sigma][U]^H. \qquad (4.18)$$

Vergleich zwischen Gleichung (4.17) und (4.18) liefert nun den direkten Zusammenhang zwischen den Eigenmoden $[\phi]$ und den singulären Vektoren $[U]$. Unter der Annahme, dass nur eine Eigenmode zur untersuchten Frequenz ω dominant ist und dass die untersuchte Frequenz der Eigenfrequenz der k-ten Eigenmode $\omega \to \omega_{r,k}$ entspricht, kann die PSD-Matrix mit $rang([G_{YY}(\omega)]) = 1$ approximiert werden. Unter dieser Annahme entspricht der erste singuläre Vektor $\{u_1\}$ einer Schätzung der k-ten Eigenmode:

$$\left\{\hat{\phi}_k\right\} = \{u_1(w_k)\}. \qquad (4.19)$$

Die FDD-Methode kann durch die sogenannte *Enhanced Frequency Domain Decomposition* Methode (EFDD) erweitert werden, die zusätzlich die Dämpfungsverhältnisse bestimmen kann [Trép17].

Durch die FDD-Methode können auch nach benachbarte Eigenmoden unterschieden werden. Jedoch kann ein Bias bei der Schätzung der Eigenmoden bestehen. Die durch die SVD erhaltenen singulären Vektoren sind orthogonal. Falls die experimentell bestimmten Eigenvektoren jedoch nicht orthogonal sind, entsteht ein Schätzfehler. Hervorzuheben sei hierbei jedoch, dass die Schätzung der dominanten Mode noch immer gut ist. Lediglich die schwächere Mode wird negativ beeinflusst [RaFa14].

4.3.5 Parametrische OMA-Methoden im Frequenzbereich

Parametrische Methoden bestimmen die modalen Parameter nicht auf Grundlage der Ausgangs-PSD-Matrix, sondern approximieren das durch die Ausgangs-PSD-Matrix beschriebene Systemverhalten mit einem parametrischen Modell. Dieses parametrische Modell wird entweder durch das *Pole-Residue Model* (PRM) (siehe Kapitel 4.2.2) oder das *Matrix Fraction Description* (MFD) (siehe Kapitel 4.2.3) beschrieben. Die durch MFD beschriebenen parametrischen Modelle benötigen Curve-Fitting-Verfahren, um den Fehler zwischen gemessener und vorhergesagter PSD-Matrix zu minimieren. In den meisten Fällen wird dieser Messfehler über ein LS-Verfahren mini-

miert. Iterative Algorithmen wie der „Maximum Likelihood Estimator“ werden in praktischen Anwendung selten verwendet. Ihr Rechenzeitbedarf ist hoch, da zusätzlich in einem vorherigen Schritt optimale Startwerte für die iterativen Verfahren bestimmt werden müssen [RaFa14].

4.4 Stabilisierungsdiagramm

Parametrische Methoden benötigen a priori Wissen über die Modellordnung des zu approximierenden Systems [RaFa14]. Dieses Vorwissen kann beispielsweise durch die Analyse des Auto-PSD-Plots erhalten werden, indem die Anzahl der Maximalstellen abgezählt wird.

In der Praxis wird dieses Wissen erhalten, indem das erhaltene Modell der Struktur für verschiedene Modellordnungen nach vordefinierten Kriterien verglichen wird. Die Übermodellierung der Struktur führt jedoch dazu, dass „falsche“ Pole auftreten und Struktureigenschaften beschrieben werden, die keine reale Struktureigenschaft abbilden. Die „falschen“ Pole können verursacht werden durch:

- *Störungs-Moden*: Sie werden beispielsweise durch Pole des Anregungssystems verursacht und bilden eine physikalische Eigenschaft des Anregungssystems ab.
- *Mathematische Pole*: Resultieren aus der mathematischen Übermodellierung des Systems und entstehen, um zusätzlich zu den physiklaschen Polen die mathematischen Anforderungen der Modellbeschreibung zu erfüllen. Sie bilden zusätzliche Eigenschaften wie Messungsrauschen etc. ab [RaFa14].

Zur Bestimmung der tatsächlichen Modellordnung des Systems muss zwischen den „falschen“ und den physikalischen Polen unterschieden werden. Die physikalischen Moden bilden reale Eigenschaften der Struktur ab. Ziel ist es diese physiklaschen Moden zu bestimmen.

Hierfür wird das sogenannte *Stabilisierungsdiagramm* verwendet. Für jede Modellordnung werden die Pole des CDM-Modells bestimmt. Das Stabilisierungsdiagramm zeigt die unterschiedlichen, durch die verschiedenen gewählten Modellordnungen erhaltenen, Pole in Abhängigkeit ihrer Frequenzen. Die vertikale Achse gibt die Modellordnung an. Abb. 7 zeigt exemplarisch ein Stabilisierungsdiagramm, das für Modellordnungen von 20 bis 55 gebildet wurde.
Das Stabilisierungsdiagramm in Abb. 7 kann interpretiert werden, indem verfolgt wird, wie sich die einzelnen Pole bei Erhöhung der Modellordnung verhalten. Die physikalischen Moden werden durch stabile Pole dargestellt. Mit stabilen Polen ist gemeint, dass diese Pole stets bei gleicher Frequenz auftreten und eine gemeinsame Orientierung bei Erhöhung der Modellordnung aufweisen. Im Stabilisierungsdiagramm sind diese daher als senkrecht nach oben gerichtete Linien sichtbar. Die mathematischen Moden stabilisieren sich hingegen nicht, sondern treten unregelmäßig und ohne erkennbare Struktur auf [GAVP00]. Ziel es über die Bestimmung der

stabilen Pole die mathematischen Pole auszusortieren. Die stabilen Pole können daher sowohl durch die gesuchten physikalischen Moden als auch durch Störungs-Moden verursacht werden.

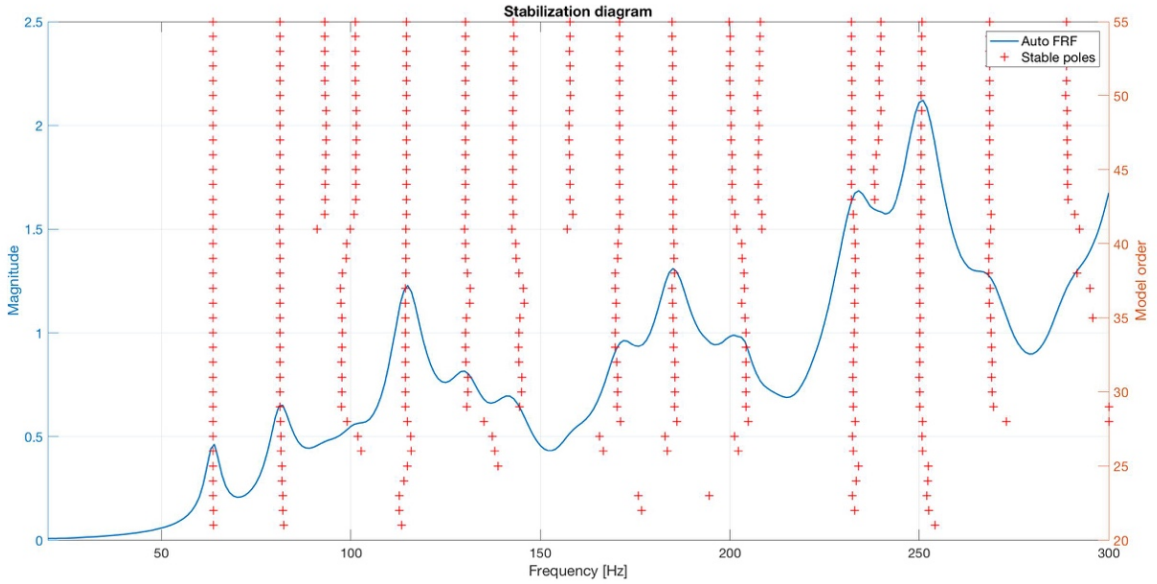


Abb. 7: Stabilisierungsdiagramm, ausgewertet von Modellordnung 20 bis 55

Das Stabilisierungsdiagramm wird erhalten, indem die zu jeder Modellordnung gehörenden Pole mit denen des Modells, der um eine geringere Modellordnung, verglichen werden. Falls die vorher definierten Kriterien erfüllt sind, werden die dazugehörigen Pole als stabil deklariert. Die Stabilisierung bzw. gleichmäßige Ausrichtung der stabilen Pole kann sowohl bei hohen als auch niedrigen Modellordbungen auftreten. Dies ist davon abhängig davon, wie stark die jeweilige Mode während der Messung angeregt wurde [RaFa00]. Typischerweise beschreiben diese Kriterien die zulässige Abweichung zwischen den zu den jeweiligen Polen gehörende Eigenfrequenzen und Dämpfungsmaßen.

In [RaFa14] und [GAVP00] werden folgende Kriterien empfohlen:

$$\left(\frac{|f(n) - f(n-1)|}{f(n)}\right) < 0.01 \qquad (4.20)$$

$$\left(\frac{|\xi(n) - \xi(n-1)|}{\xi(n)}\right) < 0.05. \qquad (4.21)$$

Hierbei wird stets eine relative Abweichung zwischen den Polen der Modellordnung n und der Modellordnung $n-1$ beschreiben. Erfüllt nun ein Pol diese Kriterien, wird er als stabil deklariert. Das hier dargestellte Kriterium soll nur als grundlegende Orientierung dienen und muss je nach Struktureigenschaften angepasst werden.

Durch zusätzliche Untersuchung der abgebildeten physikalischen Eigenschaften (z.B. erwartetes Dämpfungsmaß oder erwartete Schwingungsformen) können die gesuchten physikalischen

Pole von den Störungs-Polen unterschieden werden, die ebenfalls als stabil deklariert wurden. Bei bestimmten OMA-Methoden weisen die stabilen Pole besondere Eigenschaften auf, die die Identifizierung der physikalischen Pole vereinfacht. Exemplarisch sei hier die p-LSCF-Methode genannt. Bei geeigneter Wahl der gesuchten Koeffizienten weisen die physikalischen Pole einen negativen Realteil (positives Dämpfungsmaß) auf. Diese Besonderheit des p-LSCF-Algorithmus wird in Kapitel 5.3.5 dargestellt.

Wurde die höchste Modellordnung zu niedrig gewählt, kommt es zu einem Bias der Schätzungen der Eigenmoden. Dies ist darauf zurückzuführen, dass bei einem unterbestimmten Modell unterschiedliche Moden zu einem einzigen Pol zusammengefasst werden. Bei ausreichend hoher Modellordnung ist dieses Phänomen im Stabilisierungsdiagramm dadurch sichtbar, dass ein stabiler Pol sich ab einer bestimmten Modellordnung in zwei Pole aufteilt (zu sehen in Abb. 7 bei ca. 230 Hz. Daher ist stets eine ausreichend große Übermodellierung der Struktur zu empfehlen. Zu hohe Übermodellierung für allerdings zu vermehrtem Auftreten mathematischer Pole, was ebenfalls vermieden werden sollte [RaFa14].

4.5 Qualitätskontrolle der erhaltenen Ergebnisse

4.5.1 Eigenfrequenzen

Es wird empfohlen, die durch OMA-Methoden erhaltenen Eigenfrequenzen der Struktur zu überprüfen. Dabei werden die durch unterschiedliche OMA-Methoden erhaltenen Lösungen miteinander verglichen. Wenn die Konsistenz der experimentell bestimmten Eigenfrequenzen verifiziert wurde, können diese zum Vergleich mit theoretisch bestimmten Ergebnissen verwendet werden. Die Eigenfrequenzen können über die relative Abweichung in [%] bewertet werden:

$$\Delta f_n = \frac{f_{2,n} - f_{1,n}}{f_{1,n}} \cdot 100 \qquad (4.22)$$

$f_{1,n}$ und $f_{2,n}$ sind die Eigenfrequenzen der n-ten Eigenmoden, die durch unterschiedliche Verfahren (sowohl experimentell als auch theoretisch) bestimmt wurden und verglichen werden sollen [RaFa14]. Die Abweichung sollte hierbei möglichst gering sein, Abweichungen von 1% sind jedoch durchaus üblich. Die Ergebnisse können auch über graphische Methoden verglichen werden, hierbei sei jedoch auf die Literatur [RaFa14] verwiesen.

4.5.2 Eigenformen

Die erhaltenen Eigenformen können auf vielfältige Art verglichen werden. Der klassische Ansatz basiert auf dem optischen Vergleich der durch die unterschiedlichen Eigenmoden beschriebenen Schwingungsformen. Reale Eigenformen können hierbei direkt über die durch den Eigenformenvektor beschriebene Auslenkung visualisiert werden. Bei komplexen Eigenformen ist es sinnvoll,

die Schwingungsform zu animieren, indem die Auslenkung als sinusförmige Schwingung beschrieben wird:

$$s(t) = a_r * \sin(2\pi f * t + \psi_r) \qquad (4.23)$$

mit der Amplitude a_r:

$$a_r = |\phi_r| \qquad (4.24)$$

und der Phasenverschiebung ψ_r:

$$\psi_r = \tan^{-1}\frac{Im(\phi)}{Re(\phi)}. \qquad (4.25)$$

Durch die optische Analyse kann bewertet werden, ob zwei durch unterschiedliche OMA-Methoden erhaltenen Eigenformen übereinstimmen, sprich die gleiche Schwingungsform beschreiben. Die Bewertung der Güte ist hierbei subjektiv und es besteht keine Möglichkeit, die Güte zu quantifizieren. Außerdem tritt im Falle der OMA das Problem auf, dass die Eigenformenvektoren nicht auf die modale Masse normiert werden können und somit keine eindeutig skalierten Eigenmoden erhalten werden. Damit kann es zu Skalierungsfehlern zwischen den zu vergleichenden Eigenformen kommen [RaFa14].

Der bekannteste formale Ansatz zur Bewertung der Eigenformen ist das *Model Assurance Criterion* (MAC):

$$MAC\left(\{\phi_n^a\}\{\phi_n^b\}\right) = \frac{\left|\{\phi_n^a\}^H\{\phi_n^b\}\right|^2}{\left(\{\phi_n^a\}^H\{\phi_n^a\}\right)\left(\{\phi_n^b\}^H\{\phi_n^b\}\right)} \qquad (4.26)$$

mit zwei untersuchten Eigenmodenvektoren $\{\phi_n^a\}$ und $\{\phi_n^b\}$. Dieses Kriterium misst und bewertet die Korrelation zwischen zwei Eigenformenvektoren und kann Werte zwischen $[0, 1]$ annehmen. Wenn die untersuchten Eigenformenvektoren orthogonal zueinander sind, ist der MAC-Wert gleich 0. Bei kollinearen Vektoren hingegen ist der MAC-Wert gleich 1 und die miteinander verglichenen Schwingformen beschreiben den selben Vektor. MAC stellt damit ein Kriterium dar, das die Inkonsistenz bzw. Konsistenz zweier Vektoren bewertet. Das MAC ist anwendbar, um die Übereinstimmung sowohl zwischen zwei durch unterschiedliche OMA-Methoden erhaltene Eigenvektoren als auch Eigenformenpaare zwischen FE-Modell und OMA-Methoden zu bewerten.

Bei großen Unterschieden zwischen den dazugehörenden Komponenten der untersuchten Eigenmodenvektoren reagiert das MAC sensibel und nimmt einen sehr niedrigen Wert an. Demgegenüber werden kleinere Abweichungen wenig beachtet und die MAC-Werte sind dennoch fast nahe 1.

Bei nah benachbarten Eigenformen ist es häufig schwierig, zusammengehörende Modenpaare zwischen den durch OMA-Methoden erhaltenen experimentellen Eigenformen und den theoretischen Eigenformen der FE-Simulation zu identifizieren. Daher muss zusätzlich zur Eigenfrequenz die Eigenformen zum Vergleich hinzugezogen werden, wobei das MAC benutzt wird, um die zusammengehörenden Moden zu identifizieren. Eigenformen, die zur selben Mode gehören, weisen einen MAC-Werte nahe 1 auf [RaFa14].

5 p-LSCF-Algorithmus zur Modalanalyse

EMA- und OMA-Methoden stellen einen wichtigen Teil der bei der Untersuchung von dynamischen Strukturen dar. Die experimentell bestimmten modalen Parameter dienen als Grundlage, um bestehende theoretische Modellbeschreibungen zu verbessern (vgl. Model Updating eines FE-Modells in Kapitel 1.1). Daher besteht stetig der Bedarf an verbesserten OMA- bzw. EMA-Methoden, um eine höhere Genauigkeit der modalen Parameter gewähren zu können. Problematisch gestaltet sich hierbei die genaue Identifikation der modalen Parameter bei stark gedämpften Strukturen, die nah benachbarte Eigenmoden aufweisen. Die p-LSCF Methode wurde mit der Motivation entwickelt, auch bei solchen Systemen verlässliche Ergebnisse zu liefern [PAGL04].

5.1 Entwicklung p-LSCF

Die klassische *Least Squares Complex Frequency* Methode (LSCF) ist die Implementierung der weit verbreiteten LSCE-Methode im Frequenzbereich. Die p-LSCF-Methode wurde aus der LSCF-Methode entwickelt und ist dessen *Poly-reference* Implementierung [GAVP00]. Die LSCF-Methode wurde zuerst im EMA-Kontext entwickelt, um Anfangswerte für das iterative Maximum-Likelihood-Verfahren zu finden. Dabei wurde festgestellt, dass die gesuchten Anfangswerte bereits sehr genaue Schätzungen der modalen Parameter darstellen und das aufgebaute *Common Denominator Model* (CDM) das durch die FRF beschriebene Modell äußerst gut beschreibt [PAGL04]. Als größter Vorteil der LSCF-Methode wird in [PAGL04],[PeVa00] und [GAVP00] hervorgehoben, dass ausgesprochen klare Stabilisierungsdiagramme erhalten werden (siehe Kapitel 5.3.5). Unterschiedliche Variationen der klassischen LSCF-Methode sind in [Verb02] zu finden. Eine Möglichkeit zur Implementierung des LSCF wird in [RaFa14] vorgestellt.

Nach [PAGL04] verschlechtert sich die Güte der Systembeschreibung des LSCF sich jedoch, wenn das CDM zu einem *Pole-Residue Model* (PRM) transformiert wird. Hierbei wird die Residue-Matrix per *Singular Value Decomposition* (SVD) zu einer Rang-Eins-Matrix reduziert. Durch das CDM wird das Stabilisierungsdiagramm außerdem nur auf Grundlage der Eigenfrequenzen und Dämpfungsmaße gebildet. Die Teilnahmefaktoren (engl. Participation Factors) oder Eigenmodenvektoren sind nicht bekannt. Daraus resultiert, dass nah benachbarte Pole fälschlicherweise als ein einzelner Pol identifiziert werden.

Aus dieser Ausgangssituation wurde die *poly-reference* Version des LSCF entwickelt, die nah benachbarte Pole unterscheiden kann und zuerst in [GAVP00] als EMA-Methode beschrieben wurde. In [PeVa00] wird die p-LSCF-Methode als OMA-Methode erweitert und in [RaFa14] die

praktische Implementierung für OMA beschrieben. Die Entwicklung des p-LSCF geht auf das Unternehmen LMS International zurück, das die p-LSCF-Methode kommerziell unter dem Namen *PolyMAX* vertreibt.

5.2 Begriffserklärung p-LSCF

Die Abkürzung p-LSCF steht für *Poly-reference Least Squares Complex Frequency*. Diese Bezeichnung soll im Folgenden erläutert werden:

- *Poly-reference* (engl. für Poly-Referenz): Alle Messsensoren werden als Referenzsensoren benutzt, um die gesuchten Koeffizienten zu bestimmen. Es wird der globale Charakter aller Sensoren betrachtet und nicht nur der direkte Zusammenhang zwischen zwei Sensorpaaren, wie es bei LSCF-Methode [RaFa14].
- *Least Squares* (engl. für Methode der kleinsten Fehlerquadrate): Die p-LSCF-Methode ist eine parametrische OMA-Methode. Hierbei wird das *Least Squares* Verfahren (LS) als Curve-Fitting-Verfahren genutzt, um das durch die Messwerte beschriebene Modell mit der parametrischen Modellschreibung möglichst gut zu beschreiben. Das LS-Verfahren minimiert den Messfehler im quadratischen Mittel.
- Complex Frequency (engl. komplexe Frequenz): Das p-LSCF-Verfahren bestimmt die gesuchten Parameter (Eigenfrequenz, Dämpfungsmaß und Eigenform) im Frequenzbereich. Die Lösung des ersten LS-Schritts des p-LSCF-Verfahrens liefert komplexe Pole mit der Information über Eigenfrequenzen und Dämpfungsmaß.

5.3 Mathematische Herleitung

Die folgende mathematische Herleitung des p-LSCF-Algorithmus basiert auf der in [RaFa14] empfohlenen Implementation des Algorithmus für OMA. Ergänzungen, basierend auf anderen Quellen, wurden explizit gekennzeichnet. Die hierfür benötigten mathematischen Grundlagen werden in Kapitel 2 beschrieben. Die folgende Beschreibung der p-LSCF-Methode setzt als Ausgangsdaten *einseitige* PSD-Matrizen $[G_{YY}(\omega_f)]$ voraus. Die Implementierung auf Basis von *Positiv PSD* (PSD+) ist ebenfalls möglich, hierbei müssen lediglich die im Folgenden verwendeten Formulierungen $[G_{yy}(\omega_f)]$ durch $[S^{+}_{YY}(\omega_f)]$ und $\langle \hat{G}_o(\omega_f \rangle$ durch $\langle \hat{S}^{+}_{YY}(\omega_f \rangle$ nach [RaFa14] ersetzt werden. Die daraus resultierenden Besonderheiten werden in [PeVa00], [RaFa14] vorgestellt und in Kapitel 8 besprochen.

5.3.1 CDM der PSD -Matrix

Basierend auf den Annahmen der *Right Matrix Fraction Description* (RMFD), kann die PSD-Matrix zu jedem diskreten Eintrag des Frequenzbandes f $(f = 1, 2, \dots, N_f)$ beschrieben werden als:

$$\left[\mathrm{G_{yy}}(\omega_\mathrm{f})\right] = [\mathrm{B}(\Omega_\mathrm{f}, [\theta])][\mathrm{A}(\Omega_\mathrm{f}, [\theta])]^{-1}. \qquad (5.1)$$

Für jeden Ausgangskanal o $(o = 1, \dots, l)$, an dem die Systemantwort gemessen wird, wird das Zähler-Polynom folgendermaßen beschrieben:

$$\langle B_o(\Omega_\mathrm{f}, [\theta])\rangle = \sum_{\mathrm{j}=0}^{\mathrm{n}} \langle B_{o,j}\rangle \, \Omega_\mathrm{f}^\mathrm{j}. \qquad (5.2)$$

Das dazugehörige Nenner-Polynom ist definiert zu:

$$[\mathrm{A}(\Omega_\mathrm{f}, [\theta])] = \sum_{\mathrm{j}=0}^{\mathrm{n}} \left[\mathrm{A}_\mathrm{j}\right] \Omega_\mathrm{f}^\mathrm{j}. \qquad (5.3)$$

Die Polynom-Basisfunktion Ω_f^j kann frei gewählt werden. Die Modellordnung der Basisfunktion wird durch n beschrieben. Im Fall des CDM empfiehlt sich die Beschreibung der Basisfunktion in der diskreten z-Domain. Dies ist sowohl möglich mit

$$\Omega_\mathrm{f}^\mathrm{j} = \mathrm{e}^{(\mathrm{i}\omega_\mathrm{f}\Delta\mathrm{t})\mathrm{j}} = \mathrm{z}_\mathrm{f}^\mathrm{j}, \qquad (5.4)$$

als auch mit

$$\Omega_\mathrm{f}^\mathrm{j} = \mathrm{e}^{-(\mathrm{i}\omega_\mathrm{f}\Delta\mathrm{t})\mathrm{j}} = \mathrm{z}_\mathrm{f}^{-\mathrm{j}}. \qquad (5.5)$$

Im Folgenden wird (5.4) als Polynom-Basisfunktion angenommen. Bei der Wahl von (5.5) als Basisfunktion müssten die Formel (5.25) bis (5.27), wie in [RaFa14] beschrieben, durch eine andere Wahl der Konstanten angepasst werden.

Die $1\times l$ Matritzen $\langle B_{o,j}\rangle$ und die $l\times l$ Matrizen $\left[A_j\right]$ sind die zu bestimmenden unbekannten Parameter. Deren einzelne Matrizen lassen sich zusammenfassen zu:

$$[\beta_0] = \begin{bmatrix} \langle \mathrm{B}_{\mathrm{o},0}\rangle \\ \vdots \\ \langle \mathrm{B}_{\mathrm{o,n}}\rangle \end{bmatrix} \qquad (5.6)$$

$$[\alpha] = \begin{bmatrix} [\mathrm{A}_0] \\ \vdots \\ [\mathrm{A}_\mathrm{n}] \end{bmatrix}. \qquad (5.7)$$

Die Polynom-Koeffizienten werden in der Matrix $[\theta]$ zusammengefasst:

$$[\theta] = \begin{bmatrix} [\beta_1] \\ \vdots \\ [\beta_\mathrm{l}] \\ [\alpha] \end{bmatrix}. \qquad (5.8)$$

Ziel ist es nun, eine Fehler-Funktion der Schätzung zu bestimmen, diese im Sinne LS-Methode zu minimieren und somit eine Schätzung der zu bestimmenden Polynom-Koeffizienten zu erhalten.

5.3.2 Formulierung des LS-Problems

Die zu minimierende Kostenfunktion des *nichtlinearen Least Squares* Problems (NLS) ist:

$$\mathrm{l_{NLS}}([\theta]) = \sum_{0=1}^{\mathrm{l}} \sum_{\mathrm{f}=1}^{\mathrm{N_f}} \mathrm{tr}\left(\langle \epsilon_{\mathrm{o}}^{\mathrm{NLS}}(\omega_{\mathrm{f}}, [\theta])\rangle^{\mathrm{H}} \langle \epsilon_{\mathrm{o}}^{\mathrm{NLS}}(\omega_{\mathrm{f}}, [\theta])\rangle\right) \qquad (5.9)$$

mit dem ungewichteten NLS-Gleichungsfehler:

$$\epsilon_{\mathrm{o}}^{\mathrm{NLS}}(\omega_{\mathrm{f}}, [\theta]) = \langle \mathrm{B_o}(\Omega_{\mathrm{f}}, [\theta]\rangle [\mathrm{A}(\Omega_{\mathrm{f}}, [\theta])]^{-1} - \langle \widehat{\mathrm{G}}_{\mathrm{o}}(\omega_{\mathrm{f}})\rangle. \qquad (5.10)$$

Dieses NLS-Problem kann durch ein lineares LS-Problem angenähert werden, indem das NLS-Problem $\epsilon_o^{NLS}(\omega_f, [\theta])$ mit dem Nenner-Polynom $A(\Omega_f, [\theta])$ rechtsseitig multipliziert wird. Daraus resultiert eine Fehlerbeschreibung, die linear in den Parametern ist:

$$\begin{aligned} \epsilon_{\mathrm{o}}^{\mathrm{LS}}(\omega_{\mathrm{f}}, [\theta]) &= \epsilon_{\mathrm{o}}^{\mathrm{NLS}}(\omega_{\mathrm{f}}, [\theta])\, \mathrm{A}(\Omega_{\mathrm{f}}, [\theta]) \\ &= \langle \mathrm{B_o}(\Omega_{\mathrm{f}}, [\theta]\rangle - \langle \widehat{\mathrm{G}}_{\mathrm{o}}(\omega_{\mathrm{f}})\rangle [\mathrm{A}(\Omega_{\mathrm{f}}, [\theta])]. \end{aligned} \qquad (5.11)$$

Durch diese Formulierung ist Formel (5.9) ebenfalls linear in den Parametern und kann neu als lineares LS-Problem formuliert werden [GAVP00]. Die zu minimierende lineare Kostenfunktion lautet:

$$\mathrm{l_{LS}}([\theta]) = \sum_{0=1}^{\mathrm{l}} \sum_{\mathrm{f}=1}^{\mathrm{N_f}} \mathrm{tr}\left(\langle \epsilon_{\mathrm{o}}^{\mathrm{LS}}(\omega_{\mathrm{f}}, [\theta])\rangle^{\mathrm{H}} \langle \epsilon_{\mathrm{o}}^{\mathrm{LS}}(\omega_{\mathrm{f}}, [\theta])\rangle\right) = \mathrm{tr}\left([\theta]^{\mathrm{H}} [\mathrm{J}]^{\mathrm{H}} [\mathrm{J}] [\theta]\right). \qquad (5.12)$$

Eine Möglichkeit zur Formulierung des LS-Problems als gewichtetes LS wird in [GAVP00] dargestellt[1]. Die Minimierung der Kostenfunkton (5.12) wird durch die Lösung folgender Matrixgleichung beschrieben:

$$[\mathrm{J}][\theta] = [0]. \qquad (5.13)$$

Hierbi ist $[0]$ die Nullmatrix und $[J]$ die Jacobi-Matrix, die bestimmt wird zu:

$$[\mathrm{J}] = \begin{bmatrix} [\Gamma_1] & [0] & \dots & [0] & [\Upsilon_1] \\ [0] & [\Gamma_2] & \dots & [0] & [\Upsilon_1] \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ [0] & [0] & \dots & [\Gamma_1] & [\Upsilon_1] \end{bmatrix} \qquad (5.14)$$

mit:

$$[\Gamma_{\mathrm{o}}] = \begin{bmatrix} \langle 1 & \mathrm{z}_1 & \dots & \mathrm{z}_1^{\mathrm{n}}\rangle \\ \langle 1 & \mathrm{z}_2 & \dots & \mathrm{z}_1^{\mathrm{n}}\rangle \\ \vdots & \vdots & \dots & \vdots \\ \langle 1 & \mathrm{z}_{\mathrm{N_f}} & \dots & \mathrm{z}_{\mathrm{N_f}}^{\mathrm{n}}\rangle \end{bmatrix} \qquad (5.15)$$

$$[\Upsilon_{\mathrm{o}}] = \begin{bmatrix} \langle 1 & \mathrm{z}_1 & \dots & \mathrm{z}_1^{\mathrm{n}}\rangle & \otimes & \langle \widehat{\mathrm{G}}_{\mathrm{o}}(\omega_1)\rangle \\ \langle 1 & \mathrm{z}_2 & \dots & \mathrm{z}_2^{\mathrm{n}}\rangle & \otimes & \langle \widehat{\mathrm{G}}_{\mathrm{o}}(\omega_2)\rangle \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ \langle 1 & \mathrm{z}_{\mathrm{N_f}} & \dots & \mathrm{z}_{\mathrm{N_f}}^{\mathrm{n}}\rangle & \otimes & \langle \widehat{\mathrm{G}}_{\mathrm{o}}(\omega_{\mathrm{N_f}})\rangle \end{bmatrix}. \qquad (5.16)$$

Die Matrizen $[\Gamma_o]$ und $[\Upsilon_o]$und haben Dimension $N_f \times (n+1)$ bzw. $N_f \times (n+1)l$.

[1] Durch die Gewichtungsfunktion kann die Güte der Schätzung weiter erhöht werden, wird im Kontext dieser Arbeit aus Gründen der aufwendigeren Implementierung vernachlässigt.

5.3.3 Aufstellen der reduzierten Normalengleichung

Durch die Einführung von Normalengleichungen und dem rechten Teil der Gleichung (5.12) kann die Dimension der Gleichung reduziert werden:

$$[\mathrm{J}]^{\mathrm{H}}[\mathrm{J}][\theta] = \begin{bmatrix} [\mathrm{R}_1] & \cdots & [0] & [\mathrm{S}_1] \\ \vdots & \ddots & \vdots & \vdots \\ [0] & \cdots & [\mathrm{R}_\mathrm{l}] & [\mathrm{S}_\mathrm{l}] \\ [\mathrm{S}_1]^{\mathrm{H}} & \dots & [\mathrm{S}_\mathrm{l}]^{\mathrm{H}} & \sum_{\mathrm{o}=1}^{\mathrm{l}} [\mathrm{T}_\mathrm{o}] \end{bmatrix} \begin{bmatrix} [\beta_1] \\ [\beta_1] \\ \vdots \\ [\alpha] \end{bmatrix} = [0]. \qquad (5.17)$$

Die Herleitung hierzu ist in [GAVP00] zu finden. Die darin enthaltenen Matrizen $[R_o]$, $[S_0]$ und $[T_o]$ weisen die Dimensionen $(n+1)\times(n+1)$, $(n+1)\times(n+1)l$ bzw. $(n+1)l\times(n+1)l$ auf und werden in [RaFa14] beschrieben durch:

$$\left[\mathrm{R}_{\mathrm{o,komplex}}\right] = [\Gamma_\mathrm{o}]^{\mathrm{H}}[\Gamma_\mathrm{o}] \qquad (5.18)$$

$$\left[\mathrm{S}_{\mathrm{o,komplex}}\right] = [\Gamma_\mathrm{o}]^{\mathrm{H}}[\Upsilon_\mathrm{o}] \qquad (5.19)$$

$$\left[\mathrm{T}_{\mathrm{o,komplex}}\right] = [\Upsilon_\mathrm{o}]^{\mathrm{H}}[\Upsilon_\mathrm{o}]. \qquad (5.20)$$

Die durch die Formeln (5.18) bis (5.20) erhalten Matrizen $\left[R_{o,komplex}\right]$, $[S_{o,lomplex}]$ und $\left[T_{o,komplex}\right]$ sind komplexwertig. In [PAGL04] und [PeVa00] werden entgegen der Implementierung in [RaFa14] nur die Realteile der Matrizen $\left[R_{o,reak}\right]$, $[S_{o,real}]$ und $\left[T_{o,real}\right]$ betrachtet:

$$\left[\mathrm{R}_{\mathrm{o,real}}\right] = \mathrm{real}\left([\Gamma_\mathrm{o}]^{\mathrm{H}}[\Gamma_\mathrm{o}]\right) \qquad (5.21)$$

$$\left[\mathrm{S}_{\mathrm{o,real}}\right] = \mathrm{real}\left([\Gamma_\mathrm{o}]^{\mathrm{H}}[\Upsilon_\mathrm{o}]\right) \qquad (5.22)$$

$$\left[\mathrm{T}_{\mathrm{o,real}}\right] = \mathrm{real}\left([\Upsilon_\mathrm{o}]^{\mathrm{H}}[\Upsilon_\mathrm{o}]\right). \qquad (5.23)$$

Somit sind die zu bestimmenden Koeffizienten $[\beta_0]$ und $[\alpha]$ ebenfalls realwertig.

Nach [Caub04] resultiert das Zulassen von komplexen Koeffizienten $[\beta_0]$ und $[\alpha]$ nach den Formeln (5.18) bis (5.20) in erhöhter numerischer Qualität, da die benötigte höchste Modellordnung zur Abbildung der Struktur um den Faktor zwei reduziert wird. Jedoch kann das Zulassen komplexer Koeffizienten zu numerischen Problemen in der praktischen Implementierung des Algorithmus führen.

Das im Kontext dieser Bachelorarbeit implementierte Programm basiert daher auf realen Koeffizienten $\left[\mathrm{R}_{\mathrm{o,real}}\right]$, $\left[\mathrm{S}_{\mathrm{o,real}}\right]$ und $\left[\mathrm{T}_{\mathrm{o,real}}\right]$ nach den Formeln (5.21) bis (5.23).

5.3.4 Lösen der reduzierten Normalengleichung

Dies führt auf die reduzierte Normalengleichung, durch deren Lösung die gesuchten Koeffizienten $[\alpha]$ erhalten werden:

$$\sum_{o=1}^{l}\left([T_o]-[S_o]^H[R_o]^{-1}[S_o]\right)[\alpha]=[M][\alpha]=[0]. \qquad (5.24)$$

$[M]$ ist eine quadratische Matrix der Dimension $(n+1)l\times(n+1)l$. In der Lösung zu Gleichung (5.24) wird ein Koeffizient als $l\times l$ Einheitsmatrix festgesetzt. Dadurch wird die Parameter-Redundanz des CDM verhindert. Die Parameter-Redundanz entsteht, da Zähler und Nenner des Bruchs mit einem Skalar multipliziert werden können, wodurch sich zwar die Polynome verändern, die Pol-Information hingegen nicht [PAGL04]. Bei der Wahl des höchsten Koeffizienten des Nenner-Matrix-Polynoms als Einheitsmatrix, ergibt sich die Lösung des homogenen Gleichungssystems zu:

$$[\alpha]=\begin{bmatrix}-\left[M_{(1:n\cdot l,1:n\cdot l)}\right]^{-1}\left[M_{(1:n\cdot l,(n\cdot l+1):(n+1)n\cdot l)}\right]\\ [I_l]\end{bmatrix}. \qquad (5.25)$$

Aus Gleichung (5.25) können nun die gesuchten Koeffizienten des Nenner-Polynoms bestimmt werden. Der Nenner des CDM kann somit vollständig beschrieben werden. Dieser enthält die Information über die Pole des Systems. Da der Nenner jedoch aus Matrix-Polynomen besteht, ist es nicht möglich, dessen Nullstellen direkt zu bestimmen, die den Polen entsprechen.
Die Nullstellen werden erhalten, indem die Begleitmatrix (engl. Companion Matrix) $[A_c]$ des Nenner-Polynoms $[A(\Omega_f,[\theta])]$ folgendermaßen aufgebaut wird [GuEl00]:

$$[A_c]=\begin{bmatrix}[A'_{n-1}] & \cdots & [A'_1] & [A'_0]\\ [I] & [0] & [0] & [0]\\ \vdots & \ddots & \vdots & \vdots\\ [0] & \cdots & [I] & \vdots\end{bmatrix} \qquad (5.26)$$

mit den Koeffizienten:

$$\left[A'_j\right]=-[A_n]^{-1}\left[A_j\right]. \qquad (5.27)$$

Die Begleitmatrix ist eine quadratische $n\cdot l\times n\cdot l$ Matrix. Die Eigenwerte $z_r$der Begleitmatrix $[A_c]$ stellen die gesuchten Pole des untersuchten Systems dar. Die erhaltenen Pole sind komplexwertig und durch die Transformation (5.4) in der z-Domain beschrieben. Die Rücktransformation in den Frequenzbereich ist gegeben durch:

$$\lambda_r=\frac{\ln(z_r)}{\Delta t}. \qquad (5.28)$$

Aus den r Polen λ_r können die gesuchten modalen Parameter

- natürliche Eigenfrequenz f_r in [Hz],
- gedämpfte Eigenfrequenz $f_{d,r}$ in [Hz] und
- Dämpfungsmaß ξ_r in $[-]$

des Systems bestimmt werden, die zu der r-ten Eigenmoden gehören:

$$f_r = \frac{|\lambda_r|}{2\pi} \qquad (5.29)$$

$$f_{d,r} = \frac{\mathrm{Im}\,(\lambda_r)}{2\pi}. \qquad (5.30)$$

$$\xi_r = -\frac{\mathrm{Re}\,(\lambda_r)}{|\lambda_r|}. \qquad (5.31)$$

Damit ist der erste LS-Schritt der p-LSCF-Methode abgeschlossen.

5.3.5 Stabilisierungsdiagramm

Die identifizierten Pole haben folgende Struktur [RDCM09]:

$$\lambda_r = -\xi_r \omega_r \pm i\left(\omega_r \sqrt{1-\xi_r^2}\right). \qquad (5.32)$$

Wie in [PAGL04] und [RaFa14] beschrieben, weisen die durch den p-LSCF-Algorithmus bestimmte Pole die besondere Eigenschaft auf, dass die stabilen Pole der untersuchten Struktur einen negativen Realteil, also ein positiv Dämpfungsmaß, aufweisen. Dadurch können die stabilen von den unstabilen Polen unterschieden werden, was sich in äußerst klaren Stabilisierungsdiagrammen der p-LSCF-Methode zeigt. In Abb. 8 wird das Stabilisierungsdiagramm der p-LSCF-Methode im Vergleich zur LSCE-Methode dargestellt. Das Stabilisierungsdiagramm der p-SCF-Methode ist wesentlicher klarer und somit eindeutiger zu interpretieren.

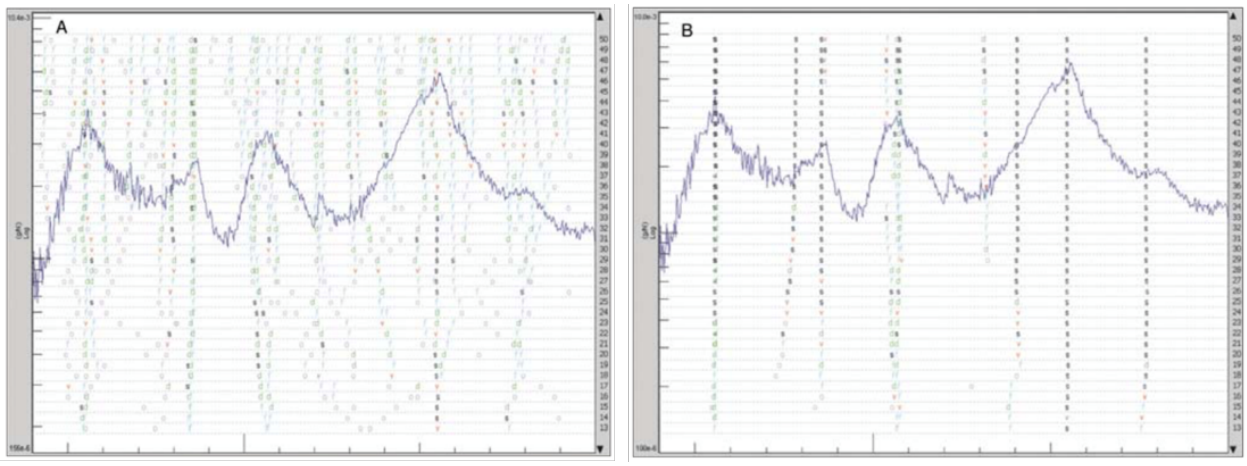


Abb. 8: Stabilisierungsdiagramme, erhalten durch die LSCE Methode (links) und p-LSCF Methode (rechts) nach [BGHJ04]

5.3.6 LSFD-Gleichungssystem

Theoretisch wäre es möglich, aus dem Nenner-Polynomen $[\theta]$ die Zähler-Polynome zu bestimmen und anschließend aus ihnen die Eigenmoden zu schätzen. Nach [RaFa14] findet dieses Vorgehen in der Praxis keine Anwendung, da das CDM keine Residue-Matrizen $[R_j]$ mit

$Rang([R_j]) = 1$ erzeugt. Der Theorie zufolge müssen diese jedoch $Rang([R_j]) = 1$ aufweisen. Dadurch würde sich die Qualität des Curve-Fittings verschlechtern.

Um dies zu umgehen, wird stattdessen in einem zweiten LS-Schritt mit der *Least Squares Frequency Domain* Methode (LSFD) ein neues ein parametrisches Modell aufgebaut. Auf Basis dessen werden die Eigenformen bestimmt. Nachdem auf Grundlage des Stabilisierungsdiagramms die physikalisch stabilen Pole identifiziert wurden, können die dazugehörigen Eigenmoden über ein weiteres LS-Verfahren bestimmt werden. Ziel ist es dabei, den Error zwischen der auf Messdaten aufgebauten PSD-Matrix $[\hat{G}_{YY}(\omega_f)]$ und dem modalen Modell auf Basis des Pole-Residue-Modells, hinsichtlich der gesuchten Residue-Matrizen $[R_j]$, zu minimieren. Im Folgenden wird nicht die in [RaFa14] empfohlene Implementierung dargestellt, sondern die in [HeLS07] dargestellte Implementierung des LSFD für EMA auf OMA sinngemäß übertragen:

Die aus dem LS-Problem erhaltende zu minimierende Kostenfunktion lässt sich zu jeder diskreten Frequenz ω_f mit $f = 1, \dots, N_f$ beschreiben:

$$[l_{LS}([R_j])] = \left([\hat{G}_{YY}(\omega_f)] - \sum_{j=1}^{N_m} \left(\frac{[R_j]}{i\omega_f - \lambda_j} + \frac{[R_j]^*}{i\omega_f - \lambda_j^*} + \frac{[B_j]}{i\omega_f - \lambda_j} + \frac{[B_j]^*}{i\omega_f - \lambda_j^*} \right) \right)^2 \qquad (5.33)$$

Vergleiche hierzu Formel (4.9). Die Kostenfunktion $[l_{LS}([R_j])]$ ist linear in ihren Parametern, da die PSD-Matrix $[\hat{G}_{YY}(\omega_f)]$, die aktuelle Kreisfrequenz ω_f, sowie die Pole des Systems λ_r bekannt sind. Die Kostenfunktion $[l_{LS}([R_j])]$ wird nach der Residue-Matrix $[R_j]$ abgeleitet und zu Null gesetzt, um das Minimum zu bestimmen:

$$\frac{d[l_{LS}([R_j])]}{d[R_j]} = 2 \left([\hat{G}_{YY}(\omega_f)] - \sum_{j=1}^{N_m} \left(\frac{[R_j]}{i\omega_f - \lambda_j} + \frac{[R_j]^*}{i\omega_f - \lambda_j^*} + \frac{[B_j]}{i\omega_f - \lambda_j} + \frac{[B_j]^*}{i\omega_f - \lambda_j^*} \right) \right) * \frac{[l_l]}{i\omega_f - \lambda_j} = 0 \qquad (5.34)$$

und vereinfacht zu:

$$\sum_{j=1}^{N_m} \left(\frac{[R_j]}{i\omega_f - \lambda_j} + \frac{[R_j]^*}{i\omega_f - \lambda_j^*} + \frac{[B_j]}{i\omega_f - \lambda_j} + \frac{[B_j]^*}{i\omega_f - \lambda_j^*} \right) = [\hat{G}_{YY}(\omega_f)]. \qquad (5.35)$$

Formel (5.36) beschreibt das zu lösende Gleichungssystem des LS-Problem zu jeder diskreten Frequenz ω_f mit $f = 1, \dots, N_f$. Auswertung zu jeder diskreten Frequenz führt zu folgender Beschreibung in Matrixform:

$$[\Lambda_\Lambda][R] = [G_\Lambda] \qquad (5.36)$$

mit:

$$[\Lambda_\Lambda] = \begin{bmatrix} \frac{[l_l]}{(\omega_1 - \lambda_1)} & \frac{[l_l]}{(\omega_1 - \lambda_1^*)} & \frac{[l_l]}{(-\omega_1 - \lambda_1)} & \frac{[l_l]}{(-\omega_1 - \lambda_1^*)} & \cdots & \frac{[l_l]}{(\omega_{N_f} - \lambda_{N_f})} & \frac{[l_l]}{(\omega_{N_f} - \lambda_{N_f}^*)} & \cdots \\ \frac{[l_l]}{(\omega_2 - \lambda_1)} & \frac{[l_l]}{(\omega_2 - \lambda_1^*)} & \frac{[l_l]}{(-\omega_2 - \lambda_1)} & \frac{[l_l]}{(-\omega_2 - \lambda_1^*)} & \cdots & \frac{[l_l]}{(\omega_{N_f} - \lambda_{N_f})} & \frac{[l_l]}{(\omega_{N_f} - \lambda_{N_f})} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{[l_l]}{(\omega_{N_f} - \lambda_1)} & \frac{[l_l]}{(\omega_{N_f} - \lambda_1^*)} & \frac{[l_l]}{(-\omega_{N_f} - \lambda_1)} & \frac{[l_l]}{(-\omega_{N_f} - \lambda_1^*)} & \cdots & \frac{[l_l]}{(\omega_{N_f} - \lambda_{N_f})} & \frac{[l_l]}{(\omega_{N_f} - \lambda_{N_f})} & \cdots \end{bmatrix} \qquad (5.37)$$

sowie:

$$[G_\Lambda] = \begin{bmatrix} [\hat{G}_{YY}(\omega_1)] \\ [\hat{G}_{YY}(\omega_2)] \\ \vdots \\ [\hat{G}_{YY}(\omega_{N_f})] \end{bmatrix}. \qquad (5.38)$$

Die gesuchten Residue-Matrizen lassen sich zusammenfassen zu:

$$[R] = \begin{bmatrix} [R_1] \\ [R_1]^* \\ [B_1] \\ [B_1]^* \\ \vdots \\ [R_{N_m}]^* \\ [R_{N_m}] \\ [B_{N_m}] \\ [B_{N_m}]^* \end{bmatrix} \qquad (5.39)$$

Die Lösung des Gleichungssystems (5.36) ist gegeben durch:

$$[R] = [\Lambda_\Lambda]^{-1}[G_\Lambda] \qquad (5.40)$$

Sobald die Residue-Matrizen $[R_j]$ für $j = 1, \dots, N_m$ bestimmt wurden, können aus ihnen die Eigenmoden geschätzt werden. Die in [RaFa14] vorgeschlagene Methode nutzt die SVD, um die Eigenmodenvektoren zu bestimmen. Der SVD zufolge können die Residue-Matrizen $[R_j]$ folgendermaßen zerlegt werden:

$$[R_j] = [U][\Sigma][V]^T. \qquad (5.41)$$

Der in der ersten Zeile der singulären Matrix $[U]$ enthaltene Vektor enthält nun eine Schätzung des gesuchten Eigenmodenvektors $\{\phi_r\}$, der zu den vier komplex-konjugierte Polen (λ_r, $\lambda_r^*, -\lambda_r, -\lambda_r^*$) gehört. Hierbei wird angenommen, dass gilt $Rang([R_j]) = 1$ und somit die Zeilenvektoren der singulären Matrix $[U]$ linear abhängig sind und somit der erste Zeilenvektor den Eigenmodenvektor bestimmt. Durch die Bestimmung des Eigenmodenvektors $\{\phi_r\}$ sind alle gesuchten modalen Parameter bestimmt und das p-LSCF-Verfahren abgeschlossen.

5.4 Bewertung p-LSCF

- Als wichtigster Vorteil der p-LSCF-Methode wird in [PAGL04],[PeVa00] und [GAVP00] hervorgehoben, dass ausgesprochen klare Stabilisierungsdiagramme erhalten werden (siehe Kapitel 5.3.5).

- Während andere Methoden im Frequenzbereich nur zur Untersuchung von stärker gedämpften Strukturen empfohlen werden und Methoden im Zeitbereich für schwächer gedämpfte Strukturen, kann die p-LSCF-Methode sowohl für schwach, als auch stark gedämpfte Strukturen angewandt werden [BGHJ04]. Die p-LSCF-Methode ist somit universell einsetzbar.
- Viele Methoden im Frequenzbereich beinhalten die Inversion einer Matrix, die Potenzen der Frequenzachse der Daten enthält. Dies kann zu numerischen Problemen führen und das nutzbare Frequenzband bzw. die höchste untersuchbare Modellordnung beschränken. Durch die Formulierung der Basisfunktion Ω_f in der z-Domain (siehe Formel (5.4)) können diese Probleme beim p-LSCF verhindert werden. Dies ist darin begründet, dass Potenzen der z-Variable nicht den Wertebereich vergrößern, sondern nur zu einer Rotation in der komplexen Ebene führt. Mit dem p-LSCF ist es daher möglich, weite Frequenzbänder und hohe maximale Modellordnungen zu untersuchen [BGHJ04]. Dies beschleunigt den Auswerteprozess erheblich, da weite Frequenzbänder analysiert werden können, anstatt diese in mehrere kleine Intervalle aufgeteilt zu untersuchen.
- In [BGHJ04] und [PAGL04] wird zusätzlich hervorgehoben, dass durch effiziente mathematische Formulierung und Implementierung der Rechenaufwand der p-LSCF-Methode gering ist. Der benötigte Rechenaufwand entspricht dem der LSCE-Methode, die aufgrund ihrer hohen Rechengeschwindigkeit vor der p-LSCF-Methode der Industriestandard war.

6 Programmstruktur des implementierten Programms

Im Kontext dieser Bachelorarbeit wurde ein Programm zur Modalanalyse in der kommerziell erhältlichen Programmier-Software MATLAB implementiert. Das erstellte Programm basiert auf der theoretischen Grundlage des p-LSCF-Algorithmus (siehe Kapitel 5) und bestimmt aus experimentell bestimmten Messdaten die modalen Parameter der untersuchten Struktur. Der Fokus während der Implementierung lag auf OMA. Aufgrund er großen Ähnlichkeit zwischen der Implementierung für OMA und EMA (vergleiche hierzu [PAGL04], [GAVP00] mit [RaFa14]), wurde das implementierte Programm um die Funktionalität erweitert, EMA Messungen zu verarbeiten.

Die .m-Datei „modalAnalysis_pLSCF_main.m“ ist das auszuführende Skipt des implementierten Programms. Als Eingangsgrößen werden für die OMA Zeitrohdaten und für die EMA die FRF-Matrix benötigt (siehe Kapitel 6.2) Ausgehend von diesen Daten werden folgende Werte bestimmt:

- Eigenfrequenzen [Hz],
- physikalisch stabile Pole des Systems [komplexwertig],
- Dämpfungsmaße der Moden [%],
- komplexe Eigenformen (normalisiert) [-] und
- reale Eigenformen (normalisiert) [-].

Zusätzlich wird das Stabilisierungsdiagramm des Berechnungsprozesses dargestellt. Dieses visualisiert die identifizierten stabilen Pole der Struktur entsprechend des gewählten Modellordnungs-Intervalls. Außerdem wird die auf Basis der identifizierten Eigenmoden rekonstruierte und gemittelte PSD- bzw. FRF-Matrix graphisch dargestellt. Der User kann hiermit die identifizierten Eigenmoden graphisch auf Plausibilität prüfen. Um die Ergebnisse numerisch zu verifizieren, können zusätzlich die Eigenformen zu den gefundenen Eigenfrequenzen über die BFD-Methode (siehe Kapitel 4.3.3) bestimmt werden. Diese werden über das MAC-Verfahren (siehe Kapitel 4.5.2) mit den Eigenformen der p-LSCF Methode verglichen.

Der vollständige Quellcode des implementierten Programms in Form des auszuführenden Skripts „modalAnalysis_pLSCF_main.m sowie dessen Funktionen sind sowohl im Anhang dieser Bachelorarbeit enthalten, als auch online über die File-Sharing Plattform GitHub (Link zum Programm https://github.com/raphajaner/modalAnalysis), verfügbar. Das Programm wird hierbei unter der Open-Source-Lizenz GNU GPLv3 frei zur Nutzung gestellt. Der dazugehörige Lizenzvertrag, ebenso wie weitere Informationen zur Nutzung eines Programms mit GNU GPLv3 sind auf: „https://www.gnu.org/licenses/“ zu finden. Der interessierte Leser sei hiermit zur Nutzung und insbesondere auch zur Weiterentwicklung des veröffentlichten Quellcodes ermutigt.

Im folgenden Kapitel werden die Programmstruktur und der -ablauf des implementierten Programms anhand der wesentlichen Teilschritte dargestellt und soll somit auch als „Benutzerhandbuch“ angesehen werden. Dieses soll sowohl als Hilfestellung bei Problemen, als auch als theoretische Grundlage für Weiterentwicklungen des Programms dienen.

6.1 Funktionsübersicht und Ordnerstruktur

Das implementierte Programm besteht aus dem folgenden Skript und dessen Funktionen:

Skript	Funktionen
• modalAnalysis_pLSCF_main.m	• OMA_PSD_calc.m • OMA_PSDmultiSensor_calc.m • pLSCF_findPoles_calc.m • OMA_pLSCF_modeShapes_calc.m • EMA_pLSCF_modeShapes_calc.m • modeNormalized_calc.m • BFD_modeShapes_calc.m • MAC_calc.m

Tabelle 1: Skript und Funktionen implementiertes Programm

Kurze Erläuterungen zum Skript und den Funktionen:

- ***modalAnalysis_pLSCF_main.m***: Start-Skript des Programms. Muss ausgeführt werden, um das Programm zu starten. Je nach Benutzereingaben werden entsprechende Funktionen aufgerufen, um die gesuchten modalen Parameter zu bestimmen.
- ***OMA_PSD_calc*.m**: Berechnet die PSD-Matrix aus den Messdaten (Zeitrohdaten). Die PSD-Matrix wird für OMA als Eingangsmatrix benötigt.
- ***OMA_PSDmultiSensor_calc.m***: Berechnung der PSD-Matrix, wenn diese aus mehreren Sätzen an Messdaten (= Multi Sensor Layout) zusammengeführt werden muss.
- ***pLSCF_findPoles_calc.m***: Identifiziert die Pole der Struktur auf Grundlage des p-LSCF-Algorithmus. Die Pole sind komplex-wertig und stellen durch Umrechnung in den Frequenzbereich die Eigenfrequenzen dar.
- ***OMA_pLSCF_modeShapes_calc.m*:** Bestimmung der Eigenformen über die LSFD-Methode (zweiter Schritt des p-LSCF Algorithmus) zu den identifizierten Eigenfrequenzen der Struktur für OMA.
- ***EMA_pLSCF_modeShapes_calc.m***: Bestimmung der Eigenformen über die LSFD-Methode (zweiter Schritt des p-LSCF-Algorithmus) zu den identifizierten Eigenfrequenzen der Struktur für EMA.

- ***modeNormalized_calc.m***: Normalisierung der Eigenformen.
- ***BFD_modeShapes_calc.m***: Bestimmung der Eigenformen über die BFD-Methode. Benötigt, um Eigenformen über MAC zu vergleichen.
- ***MAC_calc.m***: Berechnung der MAC-Werte, um Kollinearität der identifizierten Eigenformen zu vergleichen.

Das Start-Skript und die Funktionen sind im Ordner modalAnalysis angeordnet, dessen Ordnerstruktur in Abb. 9 dargestellt wird. Die Funktionen sind im Ordner f*unctions* enthalten. Die zu untersuchenden Messdaten müssen sich im Ordner *records* befinden. Die Ergebnisse des implementierten Programms werden im Ordner *results* gespeichert

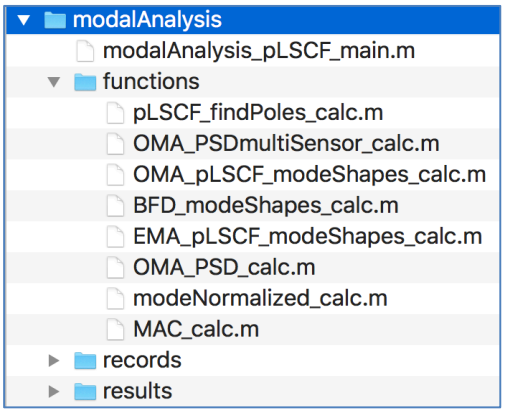


Abb. 9: Ordnerstruktur des implementierten Programms

6.2 Dateistruktur Messdaten

Die auszuwertenden Messdaten müssen in Matrixform als Variablen in den MATLAB-Workspace geladen werden. Dazu müssen die als Programm-Input benötigten Variablen zusammen in *einer* .mat-Datei gespeichert werden. Deren Dateiname ist frei wählbar, muss jedoch als Benutzereingabe (siehe Kapitel 6.3.3) übermittelt werden. Die benötigten Variablen unterscheiden sich zwischen OMA und EMA. Sie müssen nach den nachfolgenden Variablennamen benannt sein und einen speziellen Matrix-Aufbau aufweisen, der im Folgenden dargestellt wird. Hierbei darf es keine Abweichungen zu den geforderten Konventionen geben.

6.2.1 Dateistruktur OMA

Für OMA eines einzelnen zu untersuchenden Messdatensatzes wird als Programm-Input die Variable *records* benötigt.

Einzelmessung

Wurden die Messdaten in einem einzigen Datensatz gemessen, ist die Variable *records* ist eine 2-D-Matrix der Dimension $N_t \times l$. Die benötigte Variable beinhaltet das auszuwertende Zeitsignal

(Auslenkung, Kraft etc.) $S_{i=1,2,\dots,l}$ zu N_t abgetasteten diskreten Zeitpunkten aller l Sensoren. Deren benötigter Aufbau in Abb. 10 dargestellt wird.

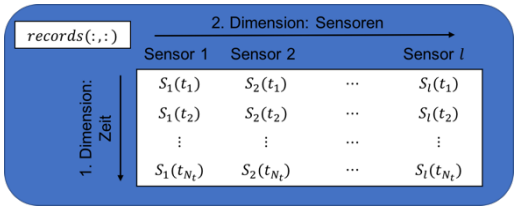


Abb. 10: Aufbau der Matrix records für einen Datensatz (OMA)

Mehrere Messungen (Multi Sensor Layout)

Wurden die Messdaten für OMA in p Datensätzen gemessen, werden diese einzelnen Datensätze über die Funktion *PSDmultiSensor*() zu einer einzigen PSD-Matrix skaliert. Die als Programm-Eingang benötigte Variable *records* wird hierzu in die 3. Dimension zu einer $N_t \times l \times p$-Matrix erweitert. Die Zeitsignal der r Referenzsensoren $RS_{i=1,2,\dots,r}$, die in jedem Datensatz enthalten sind, belegen die ersten r Spalten jedes Datensatzes. Die Zeitsignale der b bewegten Sensoren $BS_{i=1,2,\dots,b}$ werden in den nachfolgenden Spalten angeordnet. Die 3. Dimension entspricht hierbei den p Datensätzen. Der daraus resultierende Aufbau der benötigten Variable wird in Abb. 11 visualisiert.

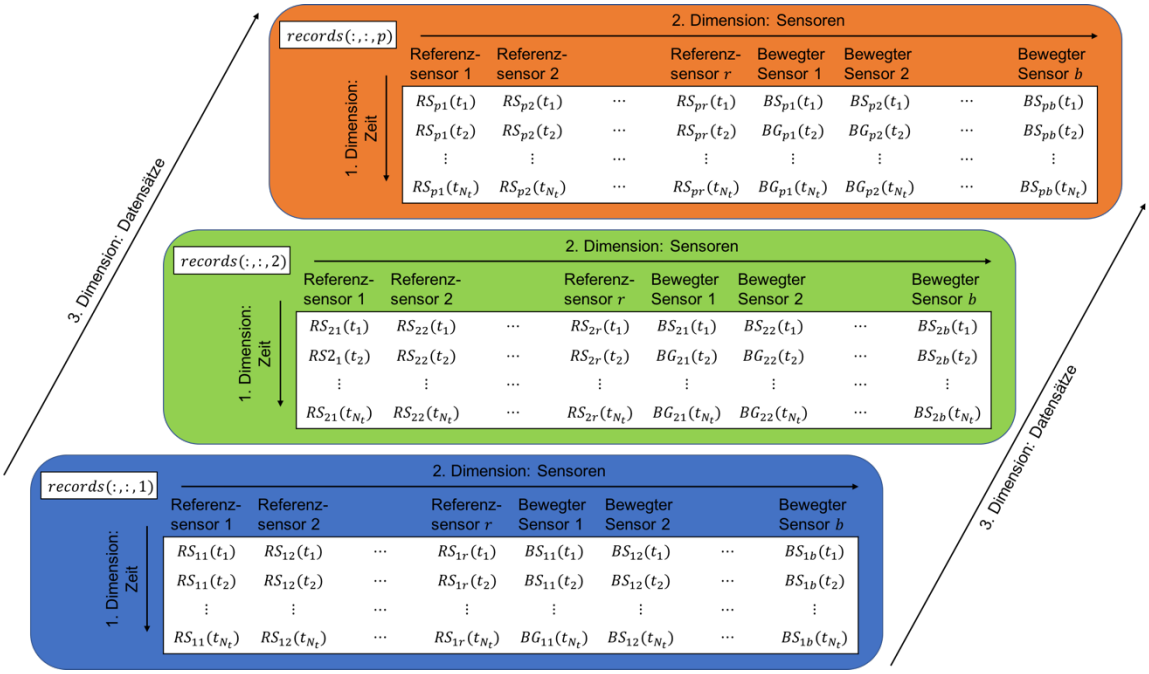


Abb. 11: Aufbau der Matrix records für mehrere Datensätze (OMA)

6.2.2 Dateistruktur EMA

Für EMA werden die Variablen *FRF* und *frequencyBand* als Programmeingang benötigt. Die Variable *FRF* beinhaltet die zuvor (unabhängig von dem im Rahmen dieser Arbeit implementierten Programm) bestimmte FRF-Matrix aller l Sensoren und einer Krafteinleitungsstelle. Die Variable *FRF* ist eine 2-D-Matrix der Dimension $N_f \times l$. Die Variable *frequencyBand* ist ein Spaltenvektor, der alle N_f diskreten Frequenzen der FRF-Matrix beinhaltet. Der Aufbau der Variablen *FRF* und *frequencyBand* wird in Abb. 12 dargestellt.

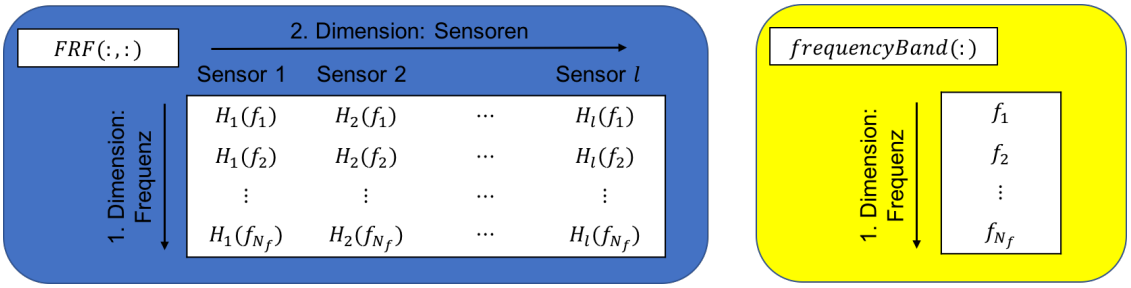


Abb. 12: Aufbau der Matrix FRF (links) und dem Spaltenvektor frequencyBand (rechts)

6.3 Skript: modalAnalysis_pLSCF_main.m

6.3.1 Allgemeine Funktion

Die .m-Datei „modalAnalysis_pLSCF_main.m“ ist das Skript des implementierten Programms und muss ausgeführt werden, um das Programm zu starten. Ausgehend von den Messdaten (Input) werden die modalen Parameter der untersuchten Struktur (Output) bestimmt:

Input	Outputs
• Messdaten	• Eigenfrequenzen [Hz] • Physikalisch stabile Pole des Systems [komplexwertig] • Dämpfungsmaße der Moden [%] • Komplexe Eigenformen (normalisiert) [-] • Reale Eigenformen (normalisiert) [-].

Tabelle 2: Input/Output modalAnalysis_pLSCF_main.m

Das implementierte Programm basiert auf dem p-LSCF-Algorithmus (siehe Kapitel 5). Der Programmablauf ist in zwölf Teilschritte gegliedert. Diese werden in den nachfolgenden Kapiteln

6.3.1 bis 6.3.13 detailliert erläutert. Zusätzlich sei hierbei erwähnt, dass der Quellcode des Programms vollständig kommentiert ist (in englischer Sprache) und somit ebenfalls leichten Zugang zum Verständnis der Funktionsweise des Programms bietet.

6.3.2 1. Schritt: Clean-Up

Um Probleme mit ggf. ungewollt im Workspace befindlichen Variablen aus vorherigen Programmdurchläufen zu verhindern, wird zu Beginn jedes Programmdurchlaufs der Workspace geleert. Zusätzlich wird das Command-Window geleert und eventuell geöffnete Figures geschlossen. Dadurch kann der reibungslose Programmablauf gewährleistet werden.

6.3.3 2. Schritt: User Input

Vor Programmstart müssen folgende Benutzereingaben im Programmcode in der Sektion „%% 2. User input“ festgelegt werden:

Variable	**Mögliche Eingaben**
• *fileNameRecords*	⇒ Dateiname der .mat-Datei der Messdaten
• *dataType*	⇒ ‚OMA’ / ‚EMA’
• *isCompareWithBFD*	⇒ true / false
• *isMultiSensorLayout*	⇒ true / false
• *n_outputsRef*	⇒ Positiv integer
• *modelOrder_min*	⇒ Positiv integer
• *modelOrder_max*	⇒ Positiv integer
• *frequencyBand_min*	⇒ Positiv double
• *frequencyBand_max*	⇒ Positiv double
• *fs*	⇒ Positiv double
• *window_opt*	⇒ Integer / vector / []
• *noverlap_opt*	⇒ Positiv integer / []
• *nfft_opt*	⇒ Positiv integer / []

Tabelle 3: Benötigte Benutzereingaben des implementierten Programms

Erläuterungen zu den geforderten Benutzereingaben:

- ***fileNameRecords***:
  Vollständiger Dateiname der .mat-Datei, die die benötigten Messdaten (Variablen) beinhaltet.
  Syntax: *fileNameRecords = ’EMA_records_tire.mat’;*

- ***dataType***:

  Mithilfe des implementierte Programms können sowohl Messdaten für OMA und EMA ausgewertet werden.

  Syntax: *dataType = 'OMA';*

- ***isCompareWithBFD:***

  Um die Ergebnisse der p-LSCF-Methode einfach vergleichen und verifizieren zu können, wurde das implementierte Programm um die BFD-Methode erweitert. Ist *true* ausgewählt, werden zusätzlich die Eigenformen zu den identifizierten Eigenfrequenzen über die BFD-Methode bestimmt. Über das MAC werden die beiden Schätzungen der Eigenformen miteinander verglichen.

  Syntax: *isCompareWithBFD = true;*

- ***isMultiSensorLayout:***

  Über diese Option können mehrere unterschiedliche Messdatensätze mit unterschiedlichen Sensoren (Referenzsensoren benötigt) der untersuchten Struktur zusammengefügt und gemeinsam analysiert werden.

  Syntax: *isMultiSensorLayout = true;*

- ***outputsRef:***

  Anzahl der unbewegten Referenzsensoren, die in jedem Messdatensatz enthalten sind. Nur benötigt bei aktiviertem Multi-Sensor-Layout.

  Syntax: *outputsRef = 2;*

- ***modelOrder_min:***

  Niedrigste Modellordnung, für die das parametrischen Modells mittels Curve-Fitting-Verfahren aufgebaut wird.

  Synatx: *modelOrder_min = 20;*

- ***modelOrder_max:***

  Höchste Modellordnung, für die das parametrischen Modells mittels Curve-Fitting-Verfahren aufgebaut wird. Dadurch entsteht das Intervall *modelOrder_min : modelOrder_max*, das alle untersuchten Modellordnungen enthält.

  Synatx: *modelOrder_max = 20;*

- ***frequencyBand_min:***

  Untere Intervallgrenze des untersuchten Frequenzbands in Hz. Schränkt das untersuchte Frequenzband nach unten ein.

  Syntax: *frequencyBand_min = 20;*

- ***frequencyBand_max:***

  Obere Intervallgrenze des untersuchten Frequenzbands in[Hz. Schränkt das untersuchte Frequenzband nach oben ein. Gesetzter Wert muss die die folgende Ungleichung

$frequencBand\_\max < \frac{fs}{2}$ mit der Abtastfrequenz fs der Messung erfüllen. Empfohlen seien Werte mit $frequencBand\_\max < \frac{fs}{2{,}56}$.

Syntax: *frequencyBand_max = 300;*

- ***fs:***

  Abtastfrequenz der Messung in [Hz].

  Syntax: *fs = 1000;*

- ***window_opt:***

  Anpassen der Fensterfunktion für die Welch-Methode. Wird zur Bestimmung der PSD-Matrix bei OMA benötigt. Standardmäßig als *[]* festgelegt, wodurch ein Hamming-Fenster mit 8 überlappenden Segmenten genutzt wird. Nur bei Bedarf anzupassen. Für weitere Informationen siehe in MATLAB-Dokumentation „cpsd/window“ (https://de.mathworks.com/help/signal/ref/cpsd.html#inputarg_window).

  Syntax: *window_opt = rectwin(10);*

- ***noverlap_opt:***

  Anpassen der Anzahl der sich überlappenden Segmente für die Welch-Methode. Wird zur Bestimmung der PSD-Matrix bei OMA benötigt. Standardmäßig als *[]* festgelegt, wodurch sich alle Segmente um 50 % überlappen. Nur bei Bedarf anzupassen. Für weitere Informationen siehe in MATLAB-Dokumentation „cpsd/noverlap“ (https://de.mathworks.com/help/signal/ref/cpsd.html#inputarg_noverlap).

  Syntax: *noverlap_opt = 0;*

- ***nfft_opt***

  Anpassen der DFT-Punkte für die Welch-Methode. Wird zur Bestimmung der PSD-Matrix bei OMA benötigt. Standardmäßig als *[]* festgelegt, wodurch die Anzahl der DFTPunkte zu $\max(256, 2^p)$ mit $p = log_2(N)$ festgelegt wird (N ist Länge der Eingangssignale). Nur bei Bedarf anzupassen. Für weitere Informationen siehe in MATLAB-Dokumentation „cpsd/nfft“ (https://de.mathworks.com/help/signal/ref/cpsd.html#inputarg_nfft).

  Syntax: *nfft_opt = 1024;*

6.3.4 3. Schritt: Initialization

Während der Initialisierung wird die .mat-Datei mit den benötigten Variablen der Messdaten in den Workspace geladen. Außerdem werden die zuvor gesetzten Benutzereingaben in der Befehlszeile ausgegeben. Damit kann der Benutzer seine getätigten Eingaben kontrollieren.

6.3.5 4. Schritt: Estimation of the PSD matrix or loading of the FRF matrix

Wird das Programm im Kontext der OMA verwendet, wird in diesem Schritt aus den vorher geladenen Zeitrohdaten der verschiedenen Sensoren die PSD Matrix berechnet. Wurden alle Sensoren als gemeinsame Messung ausgewertet, wird zur PSD-Berechnung die Funktion *OMA_PSD_calc.m* aufgerufen:

```
[PSD, frequencyBand] = OMA_PSD_calc(records, window_opt, noverlap_opt, nfft_opt, fs);
```

An diese Funktion werden die zuvor geladenen Messdaten *records* sowie die zuvor festgelegten Optionen zur cpsd-Funktion übergeben. Aus diesen wird die PSD-Matrix *PSD* sowie das dazugehörige Frequenzband *frequencyBand* berechnet.
Falls die Messdaten durch Multi-Sensor-Layout gemessen wurden, wird hingegen die Funktion *OMA_PSDmultiSensor_calc.m* (siehe Kapitel 6.4.2) aufgerufen:

```
[PSD_multiSensor, frequencyBand] = OMA_PSDmultiSensor_calc(records, n_outputsRef, win-
dow_opt, noverlap_opt, nfft_opt, fs);
```

Diese bestimmt ebenfalls die PSD-Matrix *PSD_multiSensor* sowie das dazugehörige Frequenzband *frequencyBand.*

Das Äquivalent zur PSD-Matrix im EMA-Kontext ist die FRF-Matrix und wird als Programm-Input geladen, wodurch diese nicht berechnet werden muss[2]. Die PSD- bzw. FRF-Matrix sind die in den folgenden Schritten zu untersuchten Variablen, aus denen die modalen Parameter bestimmt werden. Im weiteren Programmverlauf wird die PSD- bzw. FRF-Matrix unter dem gemeinsamen Namen *PSDoFRF* geführt:

```
PSDoFRF = PSD;
PSDoFRF = FRF;
```

Des Weiteren wird in diesem Schritt die berechnete PSD- bzw. FRF-Matrix und das dazugehörige Frequenzband auf das zuvor gesetzte Frequenzband-Intervall verkleinert.

```
frequencyBand = frequencyBand(frequencyBand_minIndex : frequencyBand_maxIndex);
PSDoFRF = PSDoFRF(frequencyBand_minIndex : frequencyBand_maxIndex, :, :);
```

6.3.6 5. Schritt: Calculation of the poles for different model orders

Auf Grundlage der zuvor berechneten PSD-Matrix (OMA) bzw. der FRF-Matrix (EMA) werden die Pole der untersuchten Struktur bestimmt.

[2] Eine Weiterentwicklung des Programms, damit auch für den EMA-Fall Zeitrohdaten als Input verwendet werden können, ist möglich. Dies war im Rahmen dieser Arbeit nicht mehr umzusetzen

Hierzu wird die Funktion *pLSCF_findPoles_calc.m* für alle Modellordnungen im zuvor festgelegten Intervall *modelOrder_min : modelOrder_max* aufgerufen:

```
for i_modelOrder = modelOrder_min : modelOrder_max

[lambdaR(:, i_modelOrder)] = pLSCF_findPoles_calc(i_modelOrder, modelOrder_max, ...
   n_outputs, n_outputsRed, deltaTime , matrixGamma_max, matrixYpsilon_max);

end
```

Die identifizierten Pole werden für alle unterschiedlichen Modellordnungen in der Variable *lambdaR* gespeichert. Die Pole sind komplexe Zahlen der in Kapitel 5.3.5 beschriebenen Struktur.

Der Funktionsaufruf *pLSCF_findPoles_calc()* weist die Besonderheit auf, dass die Variablen *matrixGamma_max* (siehe Formel (5.15)) und *matrixYpsilon_max* (siehe Formel(5.16)) vor dem Funktionsaufruf nur für die maximal gewählte Modellordnung bestimmt und übergeben werden. Die Variable *matrixGamma_max* entspricht:

```
matrixGamma_max = exp(1i * 2 * pi * frequencyBand * (0:modelOrder_max) * deltaTime);
```

und die Variable *matrixYpsilon_max* entspricht:

```
for i_outputs = 1 : n_outputs
  for i_frequencyBand = 1 : n_frequencyBand

   matrixYpsilon_max(i_frequencyBand, :, i_outputs) = ...
   -kron(matrixGamma_max(i_frequencyBand, :), PSDoFRF(i_frequencyBand, :, i_outputs));

  end
end
```

Innerhalb der Funktion *pLSCF_findPoles_calc()* wird aus diesen Matrizen die benötigten Submatrizen, die den Matrizen *[matrixGamma]* und *[matrixYipsilion]* für die untersuchte Modellordnung entsprechen, extrahiert. Durch diese Besonderheit konnte der Rechenbedarf stark reduziert werden. Genaueres ist hierzu im Kapitel 6.4.3 zur Funktion *pLSCF_findPoles_calc.m* zu lesen.

6.3.7 6. Schritt: Identification of the stable poles

Im nächsten Schritt werden die stabilen Pole der Struktur aus den zuvor identifizierten Polen bestimmt. Wie in Kapitel 5.3.5 beschrieben, weisen die stabilen Pole, wenn diese über den p-LSCF-Algorithmus bestimmt wurden, die Eigenschaft auf, dass ihr Realteil kleiner 0 ist ist:

```
for i_modelOrder = modelOrder_max : -1 : modelOrder_min
stablePoles_index = (real(lambdaR(:, i_modelOrder)) < 0) ...
& (imag(lambdaR(:, i_modelOrder)) > 0);

stablePoles_unsorted(1:size(find(stablePoles_index)), i_modelOrder) = ...
lambdaR(stablePoles_index, i_modelOrder);

end
```

Die hiermit identifizierten stabilen Pole der Struktur werden in der Variable *stablePoles* in aufsteigender Reihenfolge (nach ihren Absolutwerten) gespeichert:

```
stablePoles = sort(stablePoles_unsorted, 'ComparisonMethod', 'abs');
```

6.3.8 7. Schritt: Construction of the stabilization diagram

Auf Basis der zuvor identifizierten stabilen Pole *stablePoles* wird das Stabilisierungsdiagramm (siehe Kapitel 4.4) aufgebaut. Das Stabilisierungsdiagramm wird über den *plot*-Befehl als Figure graphisch dargestellt. Die rot dargestellten Kreuze symbolisieren hierbei die bei der jeweiligen Modellordnung identifizierten stabilen Pole der untersuchten Struktur. Ein beispielhaft durch das implementierte Programm bestimmtes Stabilisierungsdiagramm wird Kapitel 4.4 als Abb. 7 dargestellt.

6.3.9 8. Schritt: Identification of the physical stable poles

Die physikalischen Pole können durch Betrachtung der physikalischen Eigenschaften der stabilen Pole (diese können neben physikalischen Polen auch Störungs-Pole enthalten) identifiziert werden (siehe hierzu Kapitel 4.4). Als Kriterium wurde im Kontext dieser Bachelorarbeit lediglich das Stabilisierungsverhalten betrachtet. Es wurde festgelegt, dass sich ein stabiler Pol bei mindestens 20% der untersuchten Modellordnungen in einem Intervall von +-1% um den Absolutwert der Pole der höchsten Modellordnung befinden muss, um als physikalischer Pol deklariert zu werden:

```
for i_stablePoles = 1 : n_stablePoles

counterStablePoles =...
   0.99 * stablePoles_abs(i_stablePoles, 1) < stablePoles_abs(:) & ...
   stablePoles_abs(:) < 1.01 * stablePoles_abs(i_stablePoles, 1);

stablePoles_maxOrder(2, i_stablePoles) = size(find(counterStablePoles),1);

end
```

Hierbei wird theoretisch angenommen, dass bei der höchsten untersuchten Modellordnung alle identifizierten stabilen Pole auftreten. Sollten an dieser Stelle Probleme auftreten, muss lediglich die maximale untersuchte Modellordnung erhöht werden.

```
n_StablePolesRequired = round((modelOrder_max - modelOrder_min) * 0.2);

physicalStablePoles_pLSCF = stablePoles_maxOrder(1, stablePoles_maxOrder(2, :) >=
n_StablePolesRequired).';
```

Die identifizierten physikalisch stabilen Pole der untersuchten Struktur werden in der nach Absolutwerten sortierten Variable *physicalStablePoles_pLSCF* gespeichert:

```
physicalStablePoles_pLSCF = sort(physicalStablePoles_pLSCF, 'ComparisonMethod',
'abs');
```

Aus den physikalisch stabilen Polen werden die gesuchten Eigenfrequenzen und dazugehörige Dämpfungsmaße bestimmt:

```
naturalFrequencies_pLSCF = abs(physicalStablePoles_pLSCF(:, 1)) / (2 * pi);

dampingRatio_pLSCF = - real(physicalStablePoles_pLSCF(:, 1)) ./ ...
   abs(physicalStablePoles_pLSCF(:, 1));
```

Die Variable *naturalFrequencies_pLSCF* enthält die über den p-LSCF-Algorithmus bestimmten Eigenfrequenzen [Hz] der untersuchten Struktur. Die dazugehörigen Dämpfungsmaße werden in der Variable *dampingRatio_pLSCF* gespeichert. Bei erfolgreicher Identifizierung werden beide Variablen im Befehlsfenster ausgegeben.

Hierbei sei darauf hingewiesen, dass die durch das implementierte Programm identifizierten physiklaschen Pole keiner direkten physiklaschen Interpretation zur Bestimmung unterzogen wurden. Da sich die physikalischen Eigenschaften je nach untersuchter Struktur stark unterscheiden können, wurde lediglich das Stabilisierungsverhalten als Kriterium eines physiklaschen Poles verwendet. Die identifizierten Pole müssen daher abschließend über den Benutzer hinsichtlich physiklascher Eigenschaften interpretiert werden. Möglichkeiten zur Überprüfung sind beispielsweise charakteristische Werte für die Dämpfungsmaße oder erwartete Eigenformen.

Der p-LSCF Algorithmus ist ein 2-Schritte Verfahren. In einem ersten LS-Schritt werden die Eigenfrequenzen und dazugehörige Dämpfungsmaße bestimmt. Der erste LS-Schritt ist mit der Bestimmung der Variablen *naturalFrequencies_pLSCF* und *damingRatio_pLSCF* in diesem Schritt des implementierten Programms abgeschlossen.

6.3.10 9. Schritt: Calculation of the mode shapes

Nach Bestimmung der Eigenfrequenzen und Dämpfungsmaße werden in einem zweiten LS-Schritt die dazugehörigen Eigenformen bestimmt. Im Falle von OMA wir hierzu die Funktion *OMA_pLSCF_modeShapes_calc.m* aufgerufen, die die Eigenformen *complexModes_pLSCF* bestimmt:

```
[complexModes_pLSCF, residue, capitalLambda] = ...
OMA_pLSCF_modeShapes_calc(physicalStablePoles_pLSCF, PSDoFRF, frequencyBand);
```

Für EMA wird die Funktion EMA_pLSCF_modeShapes_calc.m aufgerufen:

```
[complexModes_pLSCF, residue, capitalLambda] = ...
EMA_pLSCF_modeShapes_calc(physicalStablePoles_pLSCF, PSDoFRF, frequencyBand);
```

Die Funktionsweisen dieser Funktionen werden in den Kapiteln 6.4.4 und 6.4.5 ausführlich dargestellt.

Die durch den zweiten LS-Schritt nach dem LSFD-Verfahren geschätzten Eigenformen werden durch die Funktion *modeNormalized_calc()* normalisiert und auf die maximale Amplitude eins skaliert:

```
[complexModes_normalized_pLSCF, modes_normalized_pLSCF] = modeNormalized_calc(complex-
Modes_pLSCF);
```

Die Variable *complexModes_normalized_pLSCF* enthält die normalisierten komplexen Eigenformen. Die Phaseninformation ist enthalten. Dahingegen enthält die Variable modes*_normalized_pLSCF* die realwertigen Eigenformen. Hierbei wird die Phaseninformation vereinfacht zu 0° bzw. 180°, also in Phase bzw. in Gegenphase, über Plus bzw. Minus festgelegt. Die Variable *complexModes_normalized_pLSCF* wird über das Befehlsfenster ausgegeben.

6.3.11 10. Schritt: PSD/FRF reconstruction

Auf Basis der im vorherigen Schritt bestimmten Eigenformen, kann die PSD- bzw. FRF-Matrix durch das gebildete parametrische Modell rekonstruiert werden.

```
PSD_rec = capitalLambda * residue;
FRF_rec = capitalLambda * residue;
```

Dazu werden die Variablen *residue* und *capitalLambda* benötigt. Zu deren Bedeutung im LSFD-Schritt sei auf Kapitel 5.3.6 verwiesen. Die rekonstruierte PSD- bzw. FRF-Matrix wird als Graph in einer Figure zusammen mit der ursprünglichen, aus den Messdaten gewonnenen Matrix, dargestellt. Somit kann der Benutzer die erhaltenen Ergebnisse graphisch bewerten. Ein beispielhaft durch das implementierte Programm rekonstruierte PSD- bzw. FRF-Matrix wird in Kapitel 7.1.2 als Abb. 18 dargestellt.

6.3.12 11. Schritt: Comparison of mode shapes with BFD

Bei aktivierter Option *isCompareWithBFD*, werden zusätzlich die Eigenformen durch die BFD-Methode bestimmt:

```
[complexMode_normalized_BFD] = BFD_modeShapes_calc(PSDoFRF, frequencyBand, ...
   naturalFrequencies_pLSCF);
```

Die durch die BFD-Methode bestimmten Eigenformen werden ebenfalls normalisiert und anschließend über das MAC mit den Ergebnissen des p-LSCF-Algorithmus verglichen:

```
[MACvalue] = MAC_calc(complexModes_normalized_pLSCF,complexMode_normalized_BFD);
```

Die in der Variablen *MACvalue* gespeicherten Werte werden über das Befehlsfenster ausgegeben. Wird das MAC entlang der zweiten Dimension gelesen, entspricht das den MAC-Werten der

Eigenformen der p-LSCF-Methode zu den Eigenformen der BFD-Methode. Entlang der ersten Dimension hingegen den MAC-Werten der Eigenformen der BFD-Methode zu den Eigenformen der p-LSCF-Methode.

6.3.13 12. Schritt: Saving results

Als Programmabschluss werden die bestimmten Ergebnisse und die wichtigsten Benutzereingaben unter ihrem dazugehörigen Variablennamen in einer Ergebnisdatei gespeichert:

Ergebnisse	Name der dazugehörigen Variable
• Eigenfrequenzen [Hz	⇒ *physicalStablePoles_pLSCF*
• Physikalisch stabile Pole des Systems [komplexwertig]	⇒ *naturalFrequencies_pLSCF*
• Dämpfungsmaße der Moden [%]	⇒ *dampingRatio_pLSCF*
• Komplexe Eigenformen (normalisiert) [-]	⇒ *complexModes_normalized_pLSCF*
• Reale Eigenformen (normalisiert) [-]	⇒ *modes_normalized_pLSCF*
• Minimal gewählte Modellordnung	⇒ *modelOrder_min*
• Maximal gewählte Modellordnung	⇒ *modelOrder_max*
• Untere Grenze des Frequenzbands	⇒ *frequencyBand_min*
• Obere Grenze des Frequenzbandes	⇒ *frequencyBand_max*

Tabelle 4: Ergebnisse des Programms und Namen der dazugehörigen Variablen

Der Ergebnissdatei wird im Ordner *results* (siehe Kapitel 6.1) als .mat-Datei gespeichert. Dem Dateinamen wird hinzugefügt, ob die Ergebnisse über OMA oder EMA bestimmt wurden. Zusätzlich wird der Zeitpunkt der Programmausführung in der Form *JahrMonatTagTStundeMinuteSekunde* (nach ISO 8601) im Dateinamen festgehalten. Damit ergibt sich folgende Benennungskonvention für OMA bzw. EMA:

- OMA: *OMA_pLSCF_results_JahrMonatTagTStundeMinuteSekunde.mat*
- EMA: *EMA_pLSCF_results_JahrMonatTagTStundeMinuteSekunde.mat*
  Beispielhabt sei hier eine mögliche Benennung der Ergebnissdatei angeführt: *OMA_pLSCF_results_20170708T181128.mat.* Die Messdaten wurden über OMA am 08.07.2017 um 18:11:28 Uhr ausgewertet.

```
currentDate = datestr(datetime('now'), 30);
pathFolderResults = [pwd filesep 'results' filesep];

save([pathFolderResults, 'OMA_pLSCF_results_' currentDate], ...
'physicalStablePoles_pLSCF', 'naturalFrequencies_pLSCF', 'dampingRatio_pLSCF', ...
'complexModes_normalized_pLSCF','modes_normalized_pLSCF')
save([pathFolderResults, 'EMA_pLSCF_results_' currentDate], ...
'physicalStablePoles_pLSCF', 'naturalFrequencies_pLSCF', 'dampingRatio_pLSCF', ...
'complexModes_normalized_pLSCF','modes_normalized_pLSCF')
```

Nach erfolgreicher Speicherung der Ergebnisse, ist der Programmablauf beendet, was im Befehlsfenster ausgegeben wird:

```
disp('Program completed successfully.')
```

6.4 Programmstruktur: Funktionen

6.4.1 OMA_PSD_calc.m

Die Funktion *OMA_PSD_CALC.m* wird nur für OMA verwendet. Ausgehend von den Messdaten (Input) wird die einseitige PSD-Matrix bestimmt:

Die einseitigePSD-Matrix wird über die MATLAB-Funktion *cpsd()* bestimmt:

```
for i_outputs = 1:n_outputs
    for j_outputs = 1:n_outputs

    [PSD(:, j_outputs, i_outputs), ~] = ...
        cpsd(records(:, i_outputs), records(:, j_outputs), window_opt, ...
        noverlap_opt, .nfft_opt, fs, 'onesided');

    end
end
```

Die *cpsd()* Funktion bestimmt die Kreuz-PSD-Matrix zweier diskreter Zeitsignale. Durch die Option *onesided* werden die einseitigen PSDs bestimmt. Dazu wird die Welch-Methode verwendet. Für weitere Informationen sei auf die MATLAB-Dokumentation verwiesen (als Quelle in Anhang: https://de.mathworks.com/help/signal/ref/cpsd.html).

Das Ergebnis ist die PSD-Matrix *PSD*, welche die Auto- und Kreuz-PSD zwischen den untersuchten Zeitsignalen der Sensoren beinhaltet. Die Auto-PSD kann über die *cpsd()* bestimmt werden, indem statt zweier unterschiedlicher Zeitsignale zweimal dasselbe Zeitsignal als Funktionsinput übergeben wird [3].

[3] Die Berechnung der Auto-PSD kann ebenfalls mit der MATLAB-Funktion *pwelch()* erfolgen, die Ergebnisse sind identisch

Die aus l Sensoren resultierende PSD-Matrix *PSD* hat den in Abb. 13 dargestellten Aufbau.

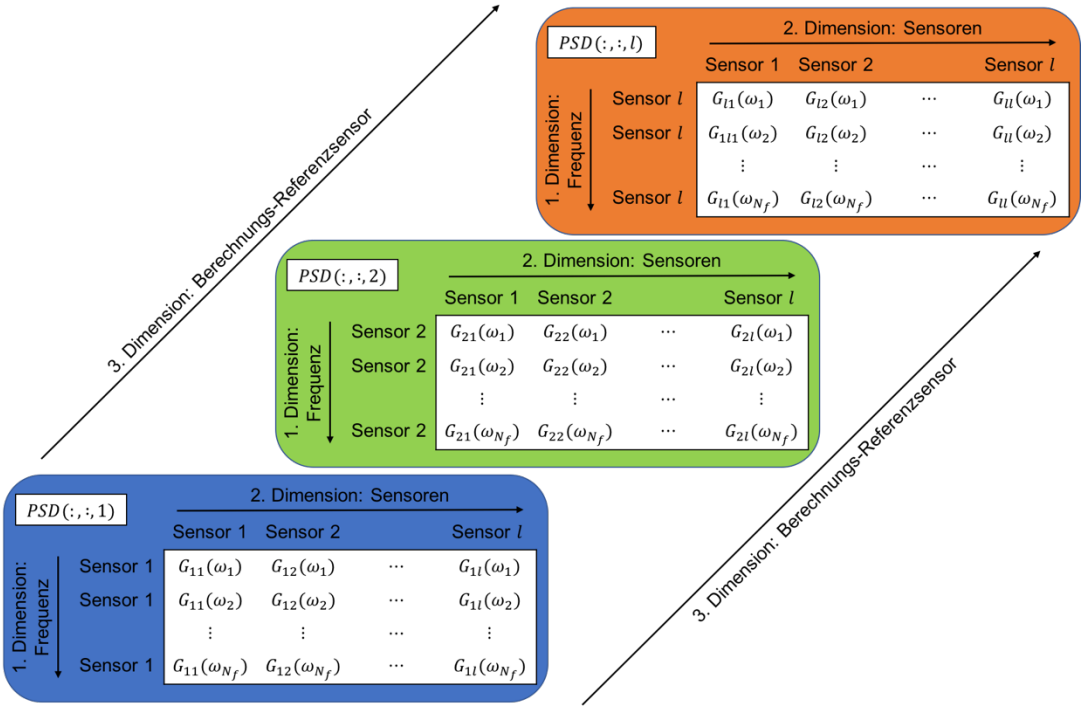


Abb. 13: Darstellung der Struktur der PSD-Matrix, die durch OMA_PSD_calc() bestimmt wird

6.4.2 OMA_PSDmultiSensor_calc.m

Funktionsaufruf:

```
[PSD_multiSensor, frequencyBand] = ...
   OMA_PSDmultiSensor_calc(records, n_outputsRef, window_opt, ...
   noverlap_opt, nfft_opt, fs)
```

Falls die Messdaten nicht in einer einzelnen Messung gemessen wurden, muss über die Funktion *OMA_PSDmultiSensor_calc()* die PSD-Matrix bestimmt werden. Hierbei wird das PreGer-Verfahren für Multi-Sensor-Layouts (siehe Kapitel 3.1.2) verwendet, um die einzelnen Messdatensätze zu einer einzigen PSD-Matrix zusammenzuführen. Die Variable *n_outputsRef* gibt die Anzahl der Referenzdatensensoren an, die in jedem Messdatensatz enthalten sind. Zur benötigten Matrix-Struktur der Messdaten *records,* siehe Kapitel 6.2.

Die Variable *PSDref* ist die PSD-Matrix zwischen allen Referenzsensoren, während die Variable *PSDmov* die PSD-Matrix der bewegten Sensoren mit den Referenzsensoren ist. Diese werden für alle Messdatensätze erstellt. Siehe hierzu Formel (3.3).

Nach Formel (3.5) werden die Sub-Matrizen der einzelnen Messdatensätze einheitlich skaliert:

```
PSDrescal_zero(:, :, n_frequencyBand) = sum(PSDref(:, :, n_frequencyBand, :), 4) /
n_patches;

for i_patches = 1 : n_patches
PSDrescal_coeff(:, :, n_frequencyBand, i_patches) = ...
   PSDmov(:, :, n_frequencyBand, i_patches) ...
   / PSDref(:, :, n_frequencyBand, i_patches) * PSDrescal_zero(:, :, n_frequencyBand);
end
```

Die resultierenden Koeffizienten werden zur PSD-Matrix nach Formel (3.6) zusammengeführt:

```
for i_patches = 1 : n_patches
PSDrescal_res((i_patches - 1) * n_outputsMov + n_outputsRef + 1 : ...
   i_patches * n_outputsMov + n_outputsRef, :, n_frequencyBand) = ...
   PSDrescal_coeff(:, :, n_frequencyBand, i_patches);
end
PSD_multiSensor = permute(PSDrescal_res, [3, 2, 1]);
```

Die Variable *PSD_multiSensor* ist die zu bestimmende PSD-Matrix, die aus unterschiedlichen Messdatensätzen zusammengefügt wurde. Der resultierende Aufbau der PSD-Matrix *PSD_multiSensor* wird in Abb. 14 dargestellt.

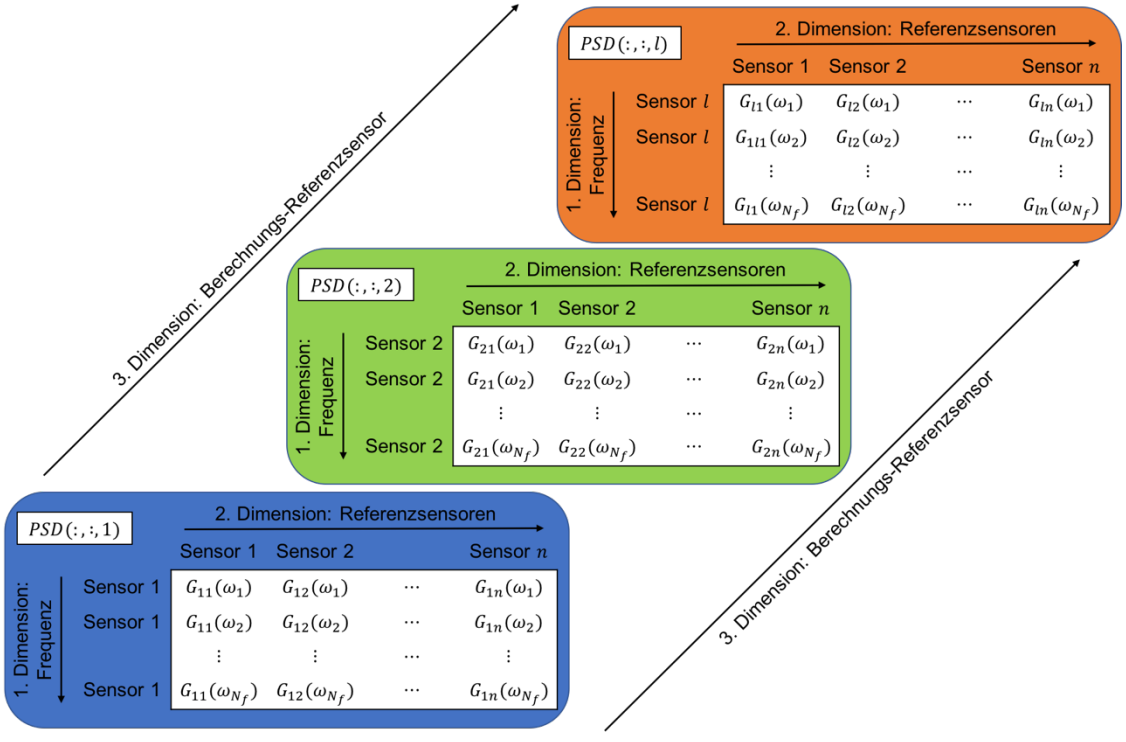


Abb. 14: Darstellung der Struktur der PSD-Matrix PSD_multiSensor*, die durch OMA_PSDmultiSensor_calc() bestimmt wird*

6.4.3 pLSCF_findPoles_calc.m

Funktionsaufruf:

```
[poles_fDomain] = pLSCF_findPoles_calc(n_modelOrder, modelOrder_max, n_outputs, ...
   n_outputsRed, deltaTime, matrixGamma_max, matrixYpsilon_max)
```

Auf Basis des p-LSCF-Algorithmus (mathematische Herleitung siehe Kapitel 5.3) werden die Pole des Systems identifiziert. Ausgang ist hierbei die PSD-Matrix für OMA bzw. die FRF-Matrix für EMA. Für die mathematische Beschreibung des ersten LS-Schritts zur Bestimmung der Pole, ist es unabhängig, ob dieses auf Basis der PSD- oder FRF-Matrix formuliert wurde. Daher kann die Funktion *sub_findPoles_calc()* gleichermaßen für OMA und EMA verwendet werden

Da die Funktion *sub_findPoles_calc()* durch das aufrufende Skript mehrmals in einer for-Schleife für unterschiedliche Modellordnungen *n_modelOrder* aufgerufen wird, wurde hierbei gezielt auf die mathematisch effiziente Formulierung und Implementierung hinsichtlich des Rechenbedarfs geachtet. Die Variablen *matrixGamma* und *matrixYpsilion* entsprechen den Formel (5.16) und (5.17) für die zu untersuchende Modellordnung *n_modelOrder*. Insbesondere *matrixYipsilion* bedarf aufgrund der Kronecker-Produkte hohen Rechenbedarfs. Indem diese Variablen im Skript vorab für die höchste zu untersuchende Modellordnung berechnet und an den Funktionsaufruf von *sub_findPoles_calc()* übergeben werden, kann der benötigte Rechenbedarf verringert werden. Aus den übergebenen Variablen *matrixGamma_max* und *matrixYipsilion_max* müssen lediglich die benötigten Submatrizen für die niedrigeren Modellordnungen ausgewählt werden, anstatt diese bei jeder Iteration und unterschiedlicher Modellordnung vollständig neu zu berechnen.

```
matrixGamma = matrixGamma_max(:, 1 : n_modelOrder + 1);
matrixYpsilon = matrixYpsilon_max(:, 1 : (n_modelOrder + 1) * n_outputsRed, :);
```

Die Variablen *matrixR_o*, *matrixS_o* und *matrixT_o* werden im erstem Schritt aus *matrixGamma* und *matrixYpsilon* nach den Formeln (5.18) bis (5.20) berechnet. Diese werden nach Formel (5.24) zur Variable *matrixM_o* zusammengefasst:

```
for i_outputs = 1 : n_outputs

 matrixM(:, :, i_outputs) =...
   matrixT_o(:, :, i_outputs) - (matrixS_o(:, :, i_outputs)') / ...
   (matrixR_o(:, :)) * matrixS_o(:, :, i_outputs);

end
```

Mit *matrixM* kann die reduzierte Normalengleichung im zweiten Schritt gelöst werden. Die Lösung ist die Variable *matrixAlpha* (siehe Formel (5.25)):

```
matrixAlpha = - (matrixM(n_outputsRed + 1 : (n_modelOrder + 1) * n_outputsRed, ...
   n_outputsRed + 1 : (n_modelOrder + 1) * n_outputsRed)) \ ...
   (matrixM(n_outputsRed + 1 : (n_modelOrder + 1) * n_outputsRed, 1 : n_outputsRed));
```

Die Parameterredundanz wird entfernt, indem der Koeffizient der niedrigsten Ordnung als Einheitsmatrix der Dimension $n\_outputsRed \times \_outputsRed$ festgesetzt wird:

```
matrixAlpha = cat(1,eye(n_outputsRed), matrixAlpha);
```

Im 3. Schritt werden die gesuchten Koeffizienten *matrixA_o* des Teiler-Polynoms aus *matrixAlpha* extrahiert (Formel (5.7)).

```
for i_modelOrder = 0 : n_modelOrder
matrixA_o(:, :, i_modelOrder + 1)= ...
   matrixAlpha(i_modelOrder * n_outputsRed + 1 : ...
   (i_modelOrder + 1) * n_outputsRed,1 : n_outputsRed);
end
```

Anschließend werden im 4. Schritt nach Formel (5.27) die Koeffizienten der Begleitmatrix bestimmt.

```
for i_modelOrder = 1 : n_modelOrder
 companionMatrix_coeff(:, :, i_modelOrder) =...
   - (matrixA_o(:, :, n_modelOrder + 1)) \ ...
   matrixA_o(:, :, n_modelOrder + 1 - i_modelOrder);
end
```

Die Begleitmatrix wird auf Basis von Formel (5.26) folgendermaßen konstruiert:

```
for i_modelOrder = 1 : n_modelOrder
companionMatrix(1 : n_outputsRed, (i_modelOrder - 1) * n_outputsRed + 1 : ...
   i_modelOrder * n_outputsRed) = companionMatrix_coeff(:, :, i_modelOrder);
end
```

Die gesuchten Pole *poles_zDomain* sind die Eigenwerte der Begleitmatrix *companionMatrix:*

```
poles_zDomain = eig(companionMatrix);
```

Diese werden aus dem z-Bereich in den Frequenzbereich transformiert.

Damit der Ergebnisvektor *poles_zDomain* stets dieselbe Länge aufweist, wird er zuerst mit der maximalen Größe deklariert und anschließend mit den identifizierten Polen befüllt:

```
poles_fDomain = zeros(modelOrder_max * n_outputsRed, 1);
poles_fDomain(1 : size(poles_zDomain)) = log(poles_zDomain) / deltaTime;
```

Die Variable *poles_fDomain* beinhaltet die gesuchten Pole des Systems bei der Modellordnung *n_modelOrder.*

6.4.4 OMA_pLSCF_modeShapes_calc.m

Die Eigenformen zu den zuvor identifizierten physikalisch stabilen Polen *physicalStablePoles_pLSCF* werden über einen weiteren LS-Schritt nach dem LSFD-Verfahren bestimmt. Hierbei wird das parametrische Modell als Pole-Residue-Modell (PRM) beschrieben. Da sich diese Beschreibung zwischen OMA auf Basis von einseitiger PSD und EMA auf Basis von FRFs unterscheidet (siehe Kapitel 4.2), wurden die Funktionen zur Schätzung der Eigenformen sowohl für OMA als auch EMA separat implementiert

Aufgrund der 4-Quadranten Symmetrie der einseitigen PSD, wird das PRM pro Mode durch 4 Koeffizienten (*capitalLambda_coeff1, capitalLambda_coeff2*, *capitalLambda_coeff3* und *capitalLambda_coeff4*) beschrieben:

```
for i_physicalStablePoles = 1 : n_physicalStablePoles
   for i_frequencyBand = 1 : n_frequencyBand

      capitalLambda_coeff1((i_frequencyBand - 1) * n_outputs + 1 : ...
         i_frequencyBand * n_outputs, :, i_physicalStablePoles) = ...
         eye(n_outputs) ./ (1i * 2 * pi * frequencyBand(i_frequencyBand) - ...
         physicalStablePoles_sort(i_physicalStablePoles));

      capitalLambda_coeff2((i_frequencyBand - 1) * n_outputs + 1 : ...
         i_frequencyBand * n_outputs, :, i_physicalStablePoles) = ...
         eye(n_outputs) ./ (1i * 2 * pi * frequencyBand(i_frequencyBand) ...
         - conj(physicalStablePoles_sort(i_physicalStablePoles)));
   end
end
```

Die Variablen *capitalLambda_coeff3* und *capitalLambda_coeff4* können recheneffizient bestimmt werden, da sie das komplex konjugierte der Variablen *capitalLambda_coeff1* und *capitalLambda_coeff2* sind:

```
capitalLambda_coeff3 = conj(capitalLambda_coeff2);
capitalLambda_coeff4 = conj(capitalLambda_coeff1);
```

Korrekt angeordnet ergeben diese Koeffizienten die Variable *capitalLambda*, die Formel (5.37) entspricht.

Die Variable *gLambda* nach Formel (5.38) wird bestimmt zu:

```
for i_frequencyBand = 1 : n_frequencyBand
gLambda((i_frequencyBand - 1) * n_outputs + 1 : i_frequencyBand * n_outputs, :) ...
   = permute(PSDoFRF(i_frequencyBand, :, :), [3, 2, 1]);
end
```

Die Residue-Matrix wird durch Lösung der Formel (5.40) gewonnen:

```
residue = capitalLambda \ gLambda;
```

An dieser Stelle kann eine Warnung von MATLAB hinsichtlich zu niedrigen Rangs der Variable *capitalLambda* entstehen: „Warning: Rank deficient“. Um dies zu umgehen, wird in der Praxis häufig die Pseudoinverse von *capitalLambda*, anstatt der korrekten Inversen, gebildet. Wird die Pseudoinverse gebildet, wird keine Fehlermeldung ausgegeben, da die Pseudoinverse in diesem Kontext immer gebildet werden kann. Durch das Bilden der Pseudoinverse kann sich jedoch die numerische Güte verschlechtern. Dabei wird keine Warnung ausgegeben, da die Pseudoinverse korrekt gebildet wurde. Im Kontext dieser Bachelorarbeit wurde daher nicht die Pseudoinverse im implementierten Programm verwendet, sondern die Inverse. Durch die Ausgabe der Warnung kann der Benutzer die inneren Vorgänge des Programms bei auftretenden Problemen interpretieren und ggf. auf einen zu geringen Rang der Variable *capitalLambda* zurückführen.

Durch SVD der Residue-Matrix *residue* werden die Eigenformen der untersuchten Struktur identifiziert. Jede erste Spalte der singulären Vektoren *uVector* entspricht einer Schätzung der dazugehörigen Eigenform *complexModes*

```
for i_physicalStablePoles = 1 : n_physicalStablePoles
[uVector, ~, ~] = ...
   svd(residue((i_physicalStablePoles - 1) * n_outputs + 1 : ...
   n_outputs * i_physicalStablePoles, :));
complexModes(:, i_physicalStablePoles) = uVector(:, 1);
end
```

Die Eigenformen *complexModes* werden als Resultat der Funktion zurück übergeben. Zusätzlich werden auch die berechneten Variablen *capitalLambda* und *gLambda* übergeben, da diese im Haupt-Programm *modalAnalysis_pLSCF_main.m* zur Rekonstruktion der PSD-Matrix benötigt werden.

6.4.5 EMA_pLSCF_modeShapes_calc.m

Funktionsaufruf:

```
[complexModes, residue, capitalLambda] = ...
   EMA_pLSCF_modeShapes_calc(physicalStablePoles_pLSCF,PSDoFRF, frequencyBand)
```

Die Eigenformen zu den zuvor identifizierten physikalisch stabilen Polen *physicalStablePoles_pLSCF* werden über einen weiteren LS-Schritt nach dem LSFD-Verfahren bestimmt. Hierbei wird das parametrische Modell als Pole-Residue Modell beschrieben (siehe Kapitel 4.2.2)
In der Variable *physicalStablePoles* werden alle identifizierten Pole und deren komplex konjugiertes gespeichert:

```
physicalStablePoles_pLSCF_all = cat(1, physicalStablePoles_pLSCF, conj(physi-
calStablePoles_pLSCF));
```

Damit wird der Nenner des Pole-Residue-Modells folgendermaßen bestimmt:

```
for i_physicalStablePoles = 1 : 2 * n_physicalStablePoles
   capitalLambda(:,i_physicalStablePoles) = 1 ./ (1i * 2 * pi * frequencyBand(:)...
      - physicalStablePoles_pLSCF_all(i_physicalStablePoles));
end
```

Zusätzlich kann die *Upper-* und *Lower-Residue*, die den Einfluss der Moden, die nicht im untersuchten Frequenzband *frequencyBand* enthalten sind, berechnet werden:

```
capitalLambda(:, i_physicalStablePoles + 1) = 1;
capitalLambda(:, i_physicalStablePoles + 2) = -1 ./ (1i*2*pi*frequencyBand(:)) .^ 2;
```

Die gesuchten Residuen werden nach Formel (5.40) berechnet:

```
residue = capitalLambda \ FRF_res;
```

Die ersten $n\_physicalStabelPoles$ Zeilen der Residue Matrix *residue* sind Schätzungen der Eigenformen der identifizierten physikalischen Pole:

```
complexModes = residue(1 : n_physicalStablePoles, :).';
```

Die Variable *complexModes* beinhaltet die gesuchten Eigenformen und wird zusammen mit den Variablen *capitalLambda* und *gLambda* übergeben, da diese im Skript *modalAnalysis_pLSCF_main.m* zur Rekonstruktion der FRF-Matrix benötigt werden.

6.4.6 modeNormalized_calc.m

Funktionsaufruf:

```
[complexModes_normalized, modes_normalized] = modeNormalized_calc(complexModes)
```

Um die erhaltenen Eigenformen mit anderen Schätzungen zu vergleichen, können die Eigenformen über die Funktion *modeNormalized_calc* normalisiert werden. Dabei wird jeweils der größte Absolutwert der einzelnen Eigenformen bestimmt:

```
[amplitudeModes_max] = max(complexModes);
```

Die einzelnen Eigenformen werden auf diese gefundenen Absolutwerte *amplitudeModes_max* skaliert, wodurch der größte Eintrag zu $1 + 0i$ skaliert wird:

```
complexModes_normalized = complexModes(:, :) ./ amplitudeModes_max;
```

Die skalierten Eigenformen sind weiterhin komplexwertig und werden in der Variable *complexModes_normalized* gespeichert. Zusätzlich werden die realwertigen Eigenformen aus den komplexwertigen bestimmt. Hierbei wir vereinfachend die der Phasenwinkel zu 0° oder 180° festgesetzt. In Phase wird durch ‚+' gekennzeichnet und Gegenphase durch ‚-':

```
for i_physicalStablePoles = 1 : size(complexModes, 2)

isInPhase = (phase_complex(:,i_physicalStablePoles) <= pi/2) & ...
   (phase_complex(:,i_physicalStablePoles) >= -pi/2);

modes_normalized(isInPhase == 0, i_physicalStablePoles) = ...
   - modes_normalized(isInPhase == 0, i_physicalStablePoles);

end
```

6.4.7 BFD_modeShapes_calc.m

Funktionsaufruf:

```
[complexMode_normalized_BFD] = ...
   BFD_modeShapes_calc(PSDoFRF, frequencyBand, naturalFrequencies)
```

Die Funktion *BFD_modeShapes_calc* bestimmt ausgehend von vorher identifizierten Eigenfrequenzen *naturalFrequencies* die Eigenformen. Der Algorithmus basierend auf der BFD-Methode und kann sowohl für OMA und EMA gleichermaßen verwendet werden.

Dazu werden im ersten Schritt die diskreten Frequenzen im Frequenzband *frequencyBand*, die den übergebenen Eigenfrequenzen am nächsten sind, lokalisiert und deren Indizes bestimmt:

```
for i_naturalFrequencies_pLSCF = 1 : n_naturalFrequencies_pLSCF
    [~, autoPSDoFRF_peaksIndex(i_naturalFrequencies_pLSCF)] = ...
        min(abs(frequencyBand-naturalFrequencies(i_naturalFrequencies_pLSCF)));
end
```

Die PSD- bzw. FRF-Matrix zu den diskreten Frequenzen, entsprechend den Indizes *autoPSDoFRF_peaksIndex*, sind Schätzungen der dazugehörigen Eigenformen:

```
for i_naturalFrequencies_pLSCF = 1 : n_naturalFrequencies_pLSCF
complexMode(:, i_naturalFrequencies_pLSCF) = ...
   PSDoFRF(autoPSDoFRF_peaksIndex(i_naturalFrequencies_pLSCF), refPoint, :);
end
```

Über die Variable *refPoint* kann bestimmt werden, welcher Sensor als Referenzsensor bei der Bestimmung der Eigenformen fungiert.

6.4.8 MAC_calc.m

Funktionsaufruf:

```
[MACvalue] = MAC_calc(complexMode_normalized_pLSCF, complexMode_normalized_BFD)
```

Über das MAC können unterschiedliche Schätzungen der selben Eigenformen miteinander verglichen werden (siehe Kapitel 4.5.2). Werden zwischen allen Eigenformen der unterschiedlichen Schätzmethoden (z.B. p-LSCF und BFD) die MAC-Werte bestimmt, können diese zu einer $n \times l$ Matrix angeordnet, die alle n Eigenformen einer Methode mit den l Eigenformen der anderen vergleicht. Die Variable *MACvalue* enthält all diese MAC-Werte und wird folgendermaßen berechnet:

```
for i_modeShapeA = 1 : n_modeShapeA
   for i_modeShapeB = 1 : n_modeShapeB
      MACvalue(i_modeShapeA, i_modeShapeB)=...
         (abs(modeShapeA(:, i_modeShapeA)' * modeShapeB(:, i_modeShapeB))) ^ 2 /...
         ((modeShapeA(:, i_modeShapeA)' * modeShapeA(:, i_modeShapeA))...
         * (modeShapeB(:, i_modeShapeB)' * modeShapeB(:, i_modeShapeB)));
    end
end
```

8 Ausblick

Das im Rahmen dieser Bachelorarbeit implementierte Programm zur Modalanalyse (OMA und EMA) soll als Grundlage für weitere Forschungsarbeiten dienen, in denen der implementierte Algorithmus weiterentwickelt wird.

Die im folgenden dargestellten Möglichkeiten zur Weiterentwicklung der implementierten p-LSCF-Methode betreffen die Anwendung des implementierten Programms für OMA.

In [JKPV15], [PeVa00] und [Caub04] wird empfohlen statt der PSD der Zeitsignale die positive Power Spectral Density PSD+ zu bilden. Die positive PSD-Matrix $\left[S_{yy}^{+}\right]$ wird nach [RaFa14] als die DFT der Korrelationsfunktionen mit positiver Zeitverzögerung $\left[S_{yy}^{+}\right] = DFT\left(R_{yy}(\tau)|_{\tau \geq 0}\right)$ definiert. In [RaFa14] und [Caub04] werden Möglichkeiten dargestellt, wie die PSD+ bestimmt werden kann. [Caub04] führt als wichtigsten Vorteil der PSD+ an, dass diese wie die FRF eine 2-Quadranten-Symmetrie aufweist (PSD weist 4-Quadranten-Symmetrie auf). Dadurch wird nur die Hälfte der maximalen Modellordnung benötigt, um das gleiche Modell, wie auf Basis der PSD, zu modellieren. Durch eine geringere benötigte Modellordnung wird der Rechenbedarf minimiert und die numerische Güte erhöht.

Eine Besonderheit der p-LSCF-Methode ist, dass die modalen Teilnahmefaktoren (engl. modal participation factors) als die Eigenvektoren der Begleitmatrix bestimmt werden können. Durch die Bestimmung der modalen Teilnahmefaktoren kann die SVD in (5.41) umgangen werden, wodurch nah benachbarte Moden besser separiert werden können [PAGL04].

Der Rechenbedarf kann weiter reduziert werden, indem der in Kapitel 6.4.3 vorgestellte Ansatz weiter fortgeführt wird. Es kann Formel (5.24) nur für die maximale Modellordnung bestimmt werden. Aus der Matrix *M* der höchsten Modellordnung können nun die gesuchten Matrizen *M* der niedrigen Modellordnungen durch Submatrizen der Matrix *M* der höchsten Modellordnung bestimmt werden.

In [JKPV15] wird eine Möglichkeit dargestellt, wie die sogenannten „Out-of-Order“-Peaks bei rotierenden Systemen umgangen werden können. Diese Peaks treten durch die harmonischen Anteile bei rotierenden Strukturen auf. Durch die Auswertung der Messdaten entlang einer der auftretenden Ordnungen, kann dieses Problem umgangen werden. Insbesondere für Anwendungen im Maschinenbau, wäre die Erweiterung des implementierten Programms um diese Funktionalität sinnvoll, da hier häufig rotierende Komponenten ausgewertet werden müssen. Hierbei kann die in MATLAB vorimplementierten Funktionen *ordertrack()* genutzt werden. Die

OMA von rotierenden Strukturen ist aktueller Bestandteil der Forschung, wodurch hierzu in Zukunft noch weitere Möglichkeiten zur Optimierung des implementieren Programms zu erwarten sind.

Die Funktionalität des implementierten Programms für EMA kann erweitert werden, indem nicht die FRF-Matrix, sondern das Zeitsignal der Sensoren als Programmeingang benötigt wird. Auf Basis der Zeitsignale kann die FRF-Matrix bestimmt werden. Hierzu kann die in MATLAB implementierte Funktion *modalfrf()* genutzt werden. Durch die Verarbeitung der Zeitsignale für EMA würde sowohl OMA als auch EMA die Zeitsignale als Programmeingang benötigen, wodurch eine eindeutige Schnittstelle als Programmeingang definiert werden könnte.

Um die allgemeine Benutzerfreundlichkeit des implementierten Programms zu erhöhen, sollte eine graphische Oberfläche (GUI) zusätzlich implementiert werden. Empfohlen sei hierbei der Aufbau einer App in MATLAB, die eine graphische Oberfläche bietet. Über den MATLAB-App-Designer wird eine Entwicklungsumgebung zur Erstellung von MATLAB-Apps angeboten. Über diese kann sowohl das Layout der App als auch die Programmierung des App-Verhaltens bestimmt werden.

9 Zusammenfassung

Die Modalanalyse beschreibt auf Grundlage der modalen Parameter die dynamischen Eigenschaften eines schwingungsfähigen Systems. Die wichtigsten modalen Parameter sind Eigenfrequenz, Dämpfungsmaß und Eigenform.

Rahmen dieser Bachelorarbeit wurde ein Programm zur Modalanalyse in der Programmier-Software MATLAB implementiert. Es kann sowohl zur Auswertung von OMA als auch EMA verwendet werden. Das implementierte Programm basiert auf dem p-LSCF-Algorithmus. Bei der p-LSCF-Methode wird durch das RFMD-Modell die PSD- bzw. FRF-Matrix durch ein parametirisches Modell approximiert. Die p-LSCF-Methode minimiert hierbei in einem ersten LS-Schritt den Fehler zwischen der PSD- bzw. FRF-Matrix, die durch die Messdaten beschrieben wird, und dem parametrischen Modell. Über den ersten LS-Schritt werden die Pole der untersuchten Struktur bestimmt. Durch die Konstruktion des Stabilisierungsdiagramms werden die physikalisch stabilen Pole der Struktur bestimmt. Diese Pole beschreiben die Eigenfrequenzen und die dazugehörigen Dämpfungsmaße der Struktur. Über eine zweiten LS-Schritt, der LSFD genannt wird, werden die zugehörigen Eigenformen geschätzt. Die wichtigste Eigenschaft der p-LSCF-Methode ist, dass die durch sie bestimmten Pole die Eigenschaft haben, dass die stabilen Pole einen negativen Realteil aufweisen. Aufgrund dieser Besonderheit können die stabilen Pole einfach bestimmt werden, wodurch die p-LSCF-Methode klare und einfache Stabilisierungsdiagramme beschreibt.

Das implementierte Programm benötigt als Programmeingang bei OMA die Zeitsignale und bei EMA die vorher bestimmte FRF-Matrix. Durch Benutzereingabe kann das auszuwertende Frequenzband und die zu untersuchenden Modellordnungen vorgeben werden. Auf Basis der Benutzereingaben werden die gesuchten modalen Parameter bestimmt. Diese werden als Programmausgang gespeichert. Um die Güte der Schätzung der Eigenformen bestimmen zu können, werden die Eigenformen der p-LSCF-Methode durch die Option *isCompareWithBFD()* über das MAC mit den Eigenformen verglichen, die durch die BFD-Methode bestimmt wurden. Das implementierte Programm kann für OMA auch Messdaten auswerten, die in unterschiedlichen Datensätzen gemessen wurden. Durch die Option *isMultiSensor()* wird auf Grundlage des Pre-Ger-Verfahrens eine skalierte PSD-Matrix für alle ausgewerteten Messtellen bestimmt.

Das implementierte Programm wurde sowohl für OMA als auch EMA verifiziert. Zur Verifizierung für die Verwendung im Kontext von OMA wurden die in einem Lehrbuch bereitgestellten Messdaten eines zu Schwingungen angeregten Hochhauses ausgewertet und mit den angegebenen Ergebnissen im Lehrbuch verglichen. Zur Verifizierung des implementierten Programms

für EMA wurden von Herrn Winandi bereitgestellte Messdaten eines liegenden Reifens ausgewertet. Diese wurden mit den Ergebnissen der kommerziell erhältlichen Software ME'scope verglichen. Anhand dieser beiden Datensätze wurde das implementierte Programm verglichen.

Das implementiere Programm soll im Sinne der Wissenschaft über die freie Lizenz GPLv3 veröffentlicht werden. Interessierte Leser seien daher zur Verwendung und Weitereinwicklung des implementierten Programms ermutigt.

10 Literaturverzeichnis

[AnBZ00] ANDERSEN, PALLE ; BRINCKER, RUNE ; ZHANG, LINGMI: *An Overview of Operational Modal Analysis: Major Development and Issues*

[BGHJ04] BART, PEETERS ; GEERT, LOWET ; HERMAN, VAN DER AUWERAER ; JAN, LEURIDAN: A New Procedure for Modal Parameter Estimation. In: *Sound and Vibration* (2004)

[Caub04] CAUBERGHE, BART: *Applied Frequency-Domain System Identification in the Field of Experimental and Operational Modal Analysis*, Vrije Universiteit Brussel, 2004

[Ewin00] EWINS, JOHN DAVID: *Modal testing: theory, practice and application* : Research Studies Press, 2000

[GAVP00] GUILLAUME, PATRICK ; AUWERAER, HERMAN VAN DER ; VANLANDUIT, S. ; PEETERS, BART: *A poly-reference implementation of the least-squares complex frequency-domain estimator*

[Grol13] GROLLIUS, STEFANIE: *Analyse des gekoppelten Systems Reifen- Hohlraum-Rad-Radführung im Rollzustand und Entwicklung eines Rollgeräuschmodells*. Karlsruhe, Karlsruher Institut für Technologie (KIT), 2013

[GuEl00] GUILLAUME, PATRICK ; EL-KAFAFY, MAHMOUD: *Model Parameter Estimation of Structures with Over- damped Poles* : Vrije Universiteit Brussels

[HeFu01] HE, JIMIN ; FU, ZHI-FANG: *Modal analysis* : Butterworth-Heinemann, 2001

[HeLS07] HEYLEN, WARD ; LAMMENS, STEFAN ; SAS, PAUL: *Modal analysis theory and testing* : Katholieke Univ. Leuven, Departement Werktuigkunde, 2007

[JKPV15] JANSSENS, KARL ; KOLLAR, ZSOLT ; PEETERS, BART ; VAN DER AUWERAER, HERMAN ; PAUWELS, STEVEN: Order-based resonance identification usingoperational PolyMAX (2015)

[Kind09] KINDT, P: *Structure-Borne Tyre/Road Noise due to Road Surface Discon- tinuities*. Leuven, Belgien, Katholieke Universiteit Leuven, 2009

[MaMo97] MAIA, N. M. M. ; MONTALVÃO E SILVA, J. M. (Hrsg.): *Theoretical and experimental modal analysis*, *Mechanical engineering research studies*. Taunton, Somerset, England : New York : Research Studies Press ; Wiley, 1997 — ISBN 978-0-86380-208-9

[MBBG02] MEVEL, L ; BASSEVILLE, M ; BENVENISTE, A ; GOURSAT, M: MERGING SENSOR DATA FROM MULTIPLE MEASUREMENT SET-UPS FOR NON-STATIONARY SUBSPACE-BASED MODAL ANALYSIS. In: *Journal of Sound and Vibration* Bd. 249 (2002), Nr. 4, S. 719–741

[Moha05] MOHANTY, PRASENJIT: *Operational Modal Analysis in the Presence of Harmonic Excitations*, Universiteit Delft, 2005

[Natk88] NATKE, GÜNTHER HANS: *Einführung in Theorie und Praxis der Zeitreihen- und Modalanalyse: Identifikation schwingungsfähiger elastomechanischer Systeme* : Vieweg, 1988

[PAGL04] PEETERS, BART ; AUWERAER, HERMAN VAN DER ; GUILLAUME, PATRICK ; LEURIDAN, JAN: The PolyMAX frequency-domain method: a new standard for modal parameter estimation? In: *Shock and Vibration* Bd. 11 (2004)

[Parl03] PARLOO, ELI: *Application of frequency-domain system identification techniques in the field of operational modal analysis*, Vrije Universiteit Brussel, 2003

[PAVG07] PEETERS, BART ; AUWERAER, DER HERMAN VAN ; VANHOLLEBEKE, FREDERIK ; GUILLAUME, PATRICK: Operational Modal Analysis for Estimating the Dynamic Properties of a Stadium Structure during a Football Game. In: *Shock and Vibration* Bd. 14 (2007), S. 283–303

[Pesc90] PESCHEL, W: *Modalanalyse im Reifenbau*. Österreich : Technische Universität Wien, 1990

[PeVa00] PEETERS, BART ; VAN DER AUWERAER, HERMAN: *PolyMAX: A Revolution in Operational Modal Analysis* : LMS International

[RaFa00] RAINIERI, CARLO ; FABBROCINO, GIOVANNI: Operational Modal Analysis: overview and applications. In: , *MEETING- Mitigation of the Earthquake Effects in Towns and in INdustrial reGionaldistricts*.

[RaFa14] RAINIERI, CARLO ; FABBROCINO, GIOVANNI: *Operational Modal Analysis of Civil Engineering Structures* : Springer New York, 2014

[RDCM09] REYNDERS, EDWIN ; DE ROECK, GUIDO ; CUNHA, ÁLVARO ; MAGALHÃES, FILIPE: Merging Strategies for Multi-Setup Operational Modal Analysis: Application to the Luiz I steel Arch Bridge. In: . Florida USA, 2009

[Reyn12] REYNDERS, EDWIN: System Identification Methods for (Operational) Modal Analysis: Review and Comparison. In: *Archives of Computational Methods in Engineering* Bd. 19 (2012), Nr. 1, S. 51–124

[Trép17] TRÉPANIER, ROBERT: *Introduction to OMA Operational Modal Analysis* : Brüel & Kjær, 2017

[Verb02] VERBOVEN, PETER: *Frequency-domain system identification for modal analysis*, Vrije Universiteit Brussel, 2002

[WhDK05] WHEELER, ROBERT L. ; DORFI, HANS R. ; KEUM, BRIAN B.: Vibration Modes of Radial Tires: Measurement, Prediction, and Categorization Under Different Boundary and Operating Conditions. In: , 2005

[Wina17] WINANDI, ACHIM: *Messdaten des liegenden Reifens im Rahmen der Bachelorarbeit bereitgestellt*. Karlsruhe, Karlsruher Institut für Technologie (KIT), FAST, 2017

11 Anhang

11.1 Programmcode: modalAnalysis_pLSCF_main.m

```
%% 1. Clean-Up
clc, clf, clearvars

%% 2. User input

% Name of the file containing the measurement data/records
fileNameRecords = 'EMA_records_tire.mat';

% OMA or EMA data?
dataType = 'EMA';

% Comparison function with BFD method true or false?
isCompareWithBFD = true;

% Multi sensor layout function true or false?
isMultiSensorLayout = false;
n_outputsRef = 2;

% Setting of the minimum and maxium model order
modelOrder_min = 20;
modelOrder_max = 55;

% Specify the examined frequency band. frequencyMax must be smaller than
% fs/2.

frequencyBand_min = 20;
frequencyBand_max = 300;

% Sampling frequency [Hz] of the sampeling records
fs = 600;

% Settings for the FFT of the cpsd (welch method) function
window_opt = [];
noverlap_opt = [];
nfft_opt = [];

3. Initialization

disp('Initialization...')
disp('-------------------------------------------------------------------------')
disp('User inputs:')

switch dataType

    case 'OMA'

        disp('--> Data typ: Operational Modal Analysis (OMA).')

        if isMultiSensorLayout == true
            disp('--> Multi sensor layout function "on".')
        else
            disp('--> Multi sensor layout function "off".')
        end

    case 'EMA'

        disp('--> Data typ: Eyperimental Modal Analysis (EMA).')

    otherwise

        error(['Unknown data type: ', dataType, ' --> either ''OMA'' or ''EMA'' possible'])
end
```

```
if isCompareWithBFD == true
    disp('--> Comparison Function is "on".')
else
    disp('--> Comparison Function is "off".')
end

disp(['--> Minimum model order is set to: "', num2str(modelOrder_min),...
    '" and maximum model order is set to: "' num2str(modelOrder_max),'".'])
disp(['--> Smallest examined frequency is set to: "', num2str(frequencyBand_min),...
    ' Hz" and highest examined frequency is set to: "' num2str(frequencyBand_max),' Hz".'])

disp('--------------------------------------------------------------------------')

% Adding path to sub folders
addpath('./functions');
addpath('./records');

disp('Importing the sampling records.')

% Importing the sample records. Imported Data must be named 'records'
load(fileNameRecords);

disp('--> The recordings were successfully imported.')
disp('--------------------------------------------------------------------------')
```

%% 4. Estimation of the PSD matrix or loading of the FRF matrix

```
% Different procedures, depending on user input

switch dataType

    case 'OMA'

        disp('Calculating the PSD matrix')

        if isMultiSensorLayout == true

            % Calling the function PSD_calc which calculates the PSD of the records
            % (auto and cross spectra between all output channels).
            [PSD_multiSensor, frequencyBand] = ...
            OMA_PSDmultiSensor_calc(records, n_outputsRef, window_opt, noverlap_opt, nfft_opt, fs);

            % Renaming the PSD matrix to an common name (used by OMA and EMA
            PSDoFRF = PSD_multiSensor;

        else

            % Importing the PSD matrix calculated by the function for multi sensor
            % layout.
            [PSD, frequencyBand] = OMA_PSD_calc(records, window_opt, noverlap_opt, nfft_opt, fs);

            % Renaming the PSD matrix to an common name (used by OMA and EMA
            PSDoFRF = PSD;

        end

        disp('--> The PSD matrix was successfully calculated.')
        disp('--------------------------------------------------------------------------')

    case 'EMA'

        disp('Loading the FRF matrix')

        % Renaming the FRF matrix to an common name (used by OMA and EMA)
        % and permuting the dimension to the required structure
        PSDoFRF = permute(FRF, [1 3 2]);

        % Sampling frequency [Hz] of the sampeling records
        fs = 2 * frequencyBand_max;
        disp('--> The FRF matrix was successfully loaded.')
        disp('--------------------------------------------------------------------------')

end
```

```
% Basic values:

% Number of output channels
n_outputs = size(PSDoFRF, 3);
% For multi sensor layout: Reduced dimension (only reference sensors)
n_outputsRed = size(PSDoFRF, 2);



% Sampling intervall [s]
deltaTime = 1 / fs;

% Identifying the frequencies of the frequency band which are
% nearest to the set frequency intervall.
[~, frequencyBand_minIndex] = min(abs(frequencyBand - frequencyBand_min));
[~, frequencyBand_maxIndex] = min(abs(frequencyBand - frequencyBand_max));

% Reduced frequency band and PSDoFRF matrix
frequencyBand = frequencyBand(frequencyBand_minIndex : frequencyBand_maxIndex);
PSDoFRF = PSDoFRF(frequencyBand_minIndex : frequencyBand_maxIndex, :, :);
```

%% 5. Calculation of the poles (lambdaR) for different model orders

```
% Index of the highest frequency
n_frequencyBand = length(frequencyBand);

disp('Calculating the poles and the natural frequencies.')

% The chosen polynomial basis function is exp(i*2*Pi*frequency*delta_t)=Omega_f=z_f
% in the z-domain. This complex polynomial basis function ensures good numerical
% conditioning. The matrix matrix_gamma has dimension n_frequencyBand x (n_modelOrder + 1).
matrixGamma_max = exp(1i * 2 * pi * frequencyBand * (0:modelOrder_max) * deltaTime);

% Preallocation
matrixYpsilon_max = zeros(n_frequencyBand - 1, (modelOrder_max + 1) * n_outputsRed, n_outputs);

% The operator "kron" is the Kronecker product.The matrix matrix_yipsilon has
% dimension n_frequencyBand x (n_modelOrder + 1) * n_outputs.
for i_outputs = 1 : n_outputs
    for i_frequencyBand = 1 : n_frequencyBand

        matrixYpsilon_max(i_frequencyBand, :, i_outputs) =...
            -kron(matrixGamma_max(i_frequencyBand, :), PSDoFRF(i_frequencyBand, :, i_outputs));

    end
end

% Preallocation
lambdaR = zeros(modelOrder_max * n_outputsRed, modelOrder_max);

% Estimation of lambda_r for different modal orders
for i_modelOrder = modelOrder_min : modelOrder_max

    [lambdaR(:, i_modelOrder)]...
        = pLSCF_findPoles_calc(i_modelOrder, modelOrder_max, n_outputs, n_outputsRed, deltaTime,...
        matrixGamma_max, matrixYpsilon_max);

end

disp('--> The poles and the natural frequencies were successfully calculated.')
disp('------------------------------------------------------------------------')
```

%% 6. Identification of the stable poles

```
disp('Identifying the physical poles.')

% Following statement finds all stable poles from every modal order n. Criterium for
% stable poles: real(lambda_r) < 0. Result is the vector stablePoles_all
% which contains all stable poles (redundancy of the poles is removed by
% just picking the poles of the complex conjugate pairs with positiv imag part).
for i_modelOrder = modelOrder_max : -1 : modelOrder_min

    stablePoles_index = (real(lambdaR(:, i_modelOrder)) < 0) & (imag(lambdaR(:, i_modelOrder)) > 0);
```

```
    stablePoles_unsorted(1:size(find(stablePoles_index)), i_modelOrder) = ...
        lambdaR(stablePoles_index, i_modelOrder);

end

stablePoles_unsorted = fliplr(stablePoles_unsorted);

% Sorting the list in ascending order (compares the abs values)
stablePoles = sort(stablePoles_unsorted, 'ComparisonMethod', 'abs');
```

%% 7. Construction of the stabilization diagram

```
% Different procedures, depending on user input
switch dataType

    case 'OMA'
        % Preallocation
        PSDoFRF_plot = zeros(n_frequencyBand, 1);

        % Calculating PSD_auto_all, which is the sum of all auto-PSDs
        for i_outputs = 1 : n_outputsRed

            PSDoFRF_plot = PSDoFRF_plot + PSDoFRF(:, i_outputs, i_outputs);

        end

        % Averaging to a single representative sensor
        PSDoFRF_plot = PSDoFRF_plot / n_outputsRed;

    case 'EMA'

        % Calculating FRF_auto_all, which is the sum of all FRFs and averaging to a
        % single representative sensor.
        PSDoFRF_plot = sum(abs(PSDoFRF(:, 1, :)), 3) / n_outputs;

end

% Plot of the auto-PSD or FRF
subplot(1, 2, 1)
yyaxis left
plot(frequencyBand, PSDoFRF_plot, 'lineWidth', 1.5);

hold on

for i_modelOrder = 1 : modelOrder_max - modelOrder_min

    % Conversion to frequenzy domain
    frequenciesPlot = abs(stablePoles(:, i_modelOrder)) ./ (2 * pi);

    % No displaying of the zero entries
    frequenciesPlot = frequenciesPlot(frequenciesPlot > 0);

    if ~isempty(frequenciesPlot)

        % Displaying points for each pole at different model orders.
        yyaxis right
        plot (frequenciesPlot,modelOrder_max + 1 - i_modelOrder,...
            '+', 'MarkerEdgeColor', 'r', 'MarkerFaceColor', 'r', 'MarkerSize', 8);

    end
end

% Title and axis labels of the stabilization diagramm
% Caption of the left y axis
yyaxis left
title('Stabilization diagram', 'FontSize', 26, 'Color', 'k')
xlabel({'Frequency [Hz]'}, 'FontSize', 20);
ylabel({'Magnitude'}, 'FontSize', 20);

% Caption of the right y axis
yyaxis right
ylabel({'Model order'}, 'FontSize', 20)
```

```
% Different procedures, depending on user input
switch dataType

    case 'OMA'

        % Legend
        legend({'Auto PSD', 'Stable poles'}, 'FontSize', 14, 'TextColor', 'k')

    case 'EMA'

        % Legend
        legend({'FRF', 'Stable poles'}, 'FontSize', 14, 'TextColor', 'k')

end
% Specific settings
grid on
hold off
```

%% 8. Identification of the physical stable poles

```
% Conversion to frequency domain
stablePoles_abs = abs(stablePoles) ./ (2*pi);

% Preallocation
stablePoles_maxOrder = zeros(2, size(stablePoles, 1));

% Assuming that all physical stable poles occures at the maximum model order,
% the identification of the physical stable poles starts from the poles,
% which occures at maximum model order.
stablePoles_maxOrder(1, :) = stablePoles(:, 1);

% Number of stable poles
n_stablePoles = size(stablePoles, 1);

% Criterum for "major stable pole":
% A major stable pole occuries at 20% of the examined model orders or more. Also its abs value changes
% less than +-1%.
for i_stablePoles = 1 : n_stablePoles

    counterStablePoles =...
        0.99 * stablePoles_abs(i_stablePoles, 1) < stablePoles_abs(:) & ...
        stablePoles_abs(:) < 1.01 * stablePoles_abs(i_stablePoles, 1);

    % Number of times a stable pole is occurring over all model orders.
    stablePoles_maxOrder(2, i_stablePoles) = size(find(counterStablePoles),1);

end

% Number of times a stable pole must occur until it is declared as a physical stable pole.
n_StablePolesRequired = round((modelOrder_max - modelOrder_min) * 0.2);

% Results:

% List with all physical stable poles
physicalStablePoles_pLSCF = stablePoles_maxOrder(1, stablePoles_maxOrder(2, :) >= n_StablePolesRequired).'; % Checken
physicalStablePoles_pLSCF = sort(physicalStablePoles_pLSCF, 'ComparisonMethod', 'abs');

% Number of major stable poles
n_physicalStablePoles_pLSCF = size(physicalStablePoles_pLSCF, 1);

% List with the corresponding natural frequencies to the physical stable poles
naturalFrequencies_pLSCF = abs(physicalStablePoles_pLSCF(:, 1)) / (2 * pi);

% List with the corresponding damping ratios to the physical stable poles
dampingRatio_pLSCF = - real(physicalStablePoles_pLSCF(:, 1)) ./ abs(physicalStablePoles_pLSCF(:, 1));

disp('--> The physical stable poles were succesfully found.')
disp('------------------------------------------------------------------------')

disp('The corresponding natural frequencies [Hz] are: ')
disp(naturalFrequencies_pLSCF.')
disp('------------------------------------------------------------------------')

disp('The corresponding damping ratios [%] are: ')
disp(dampingRatio_pLSCF.')
disp('------------------------------------------------------------------------')
```

```
%% 9. Calculation of the mode shapes

disp('Determining the associated mode shapes to the identified physical poles.')

switch dataType

    case 'OMA'

        disp('The following warning doesn´t affect the results much. Rank deficient can´t be avoided.')

        [complexModes_pLSCF, residue, capitalLambda] = ...
            OMA_pLSCF_modeShapes_calc(physicalStablePoles_pLSCF, PSDoFRF, frequencyBand);

    case 'EMA'
        disp('The following warning doesn´t affect the results much. Rank deficient can´t be avoided.')

        [complexModes_pLSCF, residue, capitalLambda] = ...
            EMA_pLSCF_modeShapes_calc(physicalStablePoles_pLSCF, PSDoFRF, frequencyBand);

end

[complexModes_normalized_pLSCF, modes_normalized_pLSCF] = modeNormalized_calc(complexModes_pLSCF);

disp('--> The mode shapes of the system were succesfully identified.')
disp('------------------------------------------------------------------------')

%disp('The complex mode shapes are:')
%disp(complexModes_normalized_pLSCF)
%disp('------------------------------------------------------------------------')

%% 10. PSD/FRF reconstruction

switch dataType

    case 'OMA'

        disp('Reconstruction of the PSD matrix.')

        % Reconstruction of the PSD matrix by the parametric model of the LSFD step
        PSD_rec = capitalLambda * residue;

        PSDoFRF_plot_rec = zeros(n_frequencyBand, 1);

        % The trace of the PSD matrix are the auto PSD
        for i_frequencyBand = 1 : n_frequencyBand
            for i_outputs_red = 1 : n_outputsRed

                PSDoFRF_plot_rec(i_frequencyBand, 1) = PSDoFRF_plot_rec(i_frequencyBand, 1)...
                    + abs(PSD_rec(n_outputs * (i_frequencyBand - 1) + i_outputs_red, ...
                    i_outputs_red));


            end
        end

        % Averaging to a single representative sensor
        PSDoFRF_plot_rec = PSDoFRF_plot_rec / n_outputsRed;

        disp('--> Succesfully reconstructed the PSD matrix.')
        disp('------------------------------------------------------------------------')

    case 'EMA'

        disp('Reconstruction of the FRF matrix.')

        % Reconstruction of the FRF matrix by the paramteric model of the LSFD step
        FRF_rec = capitalLambda * residue;

        % Sum of all sensors
        PSDoFRF_plot_rec = sum(abs(FRF_rec), 2) / n_outputs;

        disp('--> Succesfully reconstructed the FRF matrix.')
        disp('------------------------------------------------------------------------')

end
```

```
% Plot of the measured PSD and reconstructed PSD to compare them with each
% other.

subplot(1, 2, 2)
yyaxis left
plot(frequencyBand, PSDoFRF_plot, 'lineWidth', 1.5);

hold on
yyaxis right
plot(frequencyBand, PSDoFRF_plot_rec, 'lineWidth', 1.5);
ylabel({'Magnitude'}, 'FontSize', 20);

% Title and axis labels of the reconstructed FRF curve
yyaxis left

xlabel({'Frequency [Hz]'}, 'FontSize', 20);
ylabel({'Magnitude'}, 'FontSize', 20);

if isequal(dataType, 'OMA')
    title('PSD Curve', 'FontSize', 26, 'Color', 'k')
else
    title('FRF Curve', 'FontSize', 26, 'Color', 'k')
end

% Legend
legend({'Measured data', 'Parametric model'}, 'FontSize', 14, 'TextColor', 'k')

% Specific settings
grid on

hold off
```

%% 11. Comparison of mode shapes with BFD

```
if isCompareWithBFD == true

    disp('Comparison of the mode shape estimates with the BFD method.')

    % Mode shape estimation of the BFD method
    [complexMode_normalized_BFD] = BFD_modeShapes_calc(PSDoFRF, frequencyBand, naturalFrequencies_pLSCF);

    % MAC (modal assurance criterion) compares and assesses the various
    % estimates of the mode shape vectors. MAC values over 0.9 indicates good accuracy of the estimates. MAC values
    % under 0.1 should be avoided and their corresponding mode shape vectors
    % used with caution.
    [MACvalue] = MAC_calc(complexModes_normalized_pLSCF,complexMode_normalized_BFD);

    % Columns are the estimates of the pLSCF Method, rows from the BFD method
    disp('Columns are the estimates of the pLSCF Method, rows from the BFD method.')

    disp(MACvalue)

    disp('------------------------------------------------------------------------')

end
```

%% 12. Saving results

```
% Current date and time is added to the file name.
% Format:YearMonthDayTHourMinuteSecond (ISO 8601)
currentDate = datestr(datetime('now'), 30);

% Path of the folder "results"
pathFolderResults = [pwd filesep 'results' filesep];

switch dataType

    case 'OMA'

        save([pathFolderResults, 'OMA_pLSCF_results_' currentDate], 'physicalStablePoles_pLSCF', ...

            'naturalFrequencies_pLSCF', 'dampingRatio_pLSCF', ...
            'complexModes_normalized_pLSCF','modes_normalized_pLSCF', ...
            'modelOrder_min', 'modelOrder_max', 'frequencyBand_min', 'frequencyBand_max')
```

```
        disp(['The results are saved as: "ResultsOMA_plscf_', currentDate, '.mat"'])
        disp('--------------------------------------------------------------------')

    case 'EMA'

        save([pathFolderResults, 'EMA_pLSCF_results_' currentDate], 'physicalStablePoles_pLSCF', ...
            'naturalFrequencies_pLSCF', 'dampingRatio_pLSCF', ...
            'complexModes_normalized_pLSCF','modes_normalized_pLSCF', ...
            'modelOrder_min', 'modelOrder_max', 'frequencyBand_min', 'frequencyBand_max')

        disp(['The results are saved as: "ResultsEMA_plscf_', currentDate, '.mat"'])
        disp('--------------------------------------------------------------------')

end

disp('Program completed successfully.')
disp('--------------------------------------------------------------------')
```

11.2 Programmcode: OMA_PSD_calc.m

```
function [PSD, frequencyBand] = OMA_PSD_calc(records, window_opt, noverlap_opt, nfft_opt, fs)

% Number of output channels
n_outputs = size(records, 2);

[~, frequencyBand] = ...
        cpsd(records(:, 1), records(:, 1), window_opt, noverlap_opt, nfft_opt, fs, 'onesided');

n_frequencyBand = size(frequencyBand,1);

% Preallocation
PSD = zeros(n_frequencyBand, n_outputs, n_outputs);

% Calculation of the PSD matrix
for i_outputs = 1:n_outputs
    for j_outputs = 1:n_outputs

    [PSD(:, j_outputs, i_outputs), ~] = ...
        cpsd(records(:, i_outputs), records(:, j_outputs), window_opt, noverlap_opt, nfft_opt, fs, 'onesided');

    end
end

end
```

11.3 Programmcode: OMA_PSDmuktiSensor_calc.m

```
function [PSD_multiSensor, frequencyBand] =...
    OMA_PSDmultiSensor_calc(records, n_outputsRef, window_opt, noverlap_opt, nfft_opt, fs)

%% Basic values

% Number of patches
n_patches = size(records, 3);

% Number of all sensors (reference and moved) per patch
n_outputsPatch = size(records, 2);

% Number of moved sensors per patch
n_outputsMov = n_outputsPatch - n_outputsRef;
```

```
%% 1. Calculation of the PSD matrices for each patch

% Calculation of the frequency band of the PSD matrix
[~, frequencyBand] = cpsd(records(:, 1, 1), records(:, 1, 1), window_opt, noverlap_opt, nfft_opt, fs, 'onesided');

% Number of discrete frequency entries
n_frequencyBand = size(frequencyBand, 1);

% Preallocation
PSDref = zeros(n_frequencyBand, n_outputsRef, n_outputsRef, n_patches);

% Calculation of PSDref. The matrix PSDref contains all auto- and
% cross PSD estimates between the reference sensors.
for i_patches =1 : n_patches
    for i_outputsRef = 1 : n_outputsRef
        for j_outputsRef = 1 : n_outputsRef

            [PSDref(:, j_outputsRef, i_outputsRef, i_patches), ~] = ...
                cpsd(records(:, i_outputsRef, i_patches), records(:, j_outputsRef, i_patches), window_opt, noverlap_opt, nfft_opt, fs, 'one-
sided');

        end
    end
end

% Preallocation
PSDmov = zeros(n_frequencyBand,n_outputsRef,n_outputsMov,n_patches);

% Calculation of PSDmov. The matrix mov_PSD contains all cross-PSD
% estimates between the moved sensors and the reference sensors.
for i_patches = 1 : n_patches
    for i_outputsRef = 1 : n_outputsMov
        for j_outputsRef = 1 : n_outputsRef

            [PSDmov(:, j_outputsRef, i_outputsRef, i_patches), ~] =...
                cpsd(records(:, n_outputsRef + i_outputsRef, i_patches), records(:, j_outputsRef, i_patches), window_opt, noverlap_opt,
nfft_opt, fs, 'onesided');

        end
    end
end

%% 2. Re-scaling of the PSD matrices to a single PSD matrix

PSDrescal_zero = zeros(n_outputsRef, n_outputsRef, n_frequencyBand);
PSDrescal_coeff = zeros(n_outputsMov, n_outputsRef, n_frequencyBand, n_patches);
PSDrescal_res = zeros(n_outputsPatch, n_outputsRef, n_frequencyBand);


% Rearrange of the matrices
PSDmov = permute(PSDmov, [3, 2, 1, 4]);
PSDref = permute(PSDref, [3, 2, 1, 4]);

% Re-scaling for every discrete frequency entry
for n_frequencyBand = 1 : size(frequencyBand)

    % Calculation of coefficients for the re-scaled data set

    % Re-scaling factor for every patch. Averaged over all ref_PSD from all
    % patches.
    PSDrescal_zero(:, :, n_frequencyBand) = sum(PSDref(:, :, n_frequencyBand, :), 4) / n_patches;

    % Calculation of the re-scaled coefficients
    for i_patches = 1 : n_patches

        PSDrescal_coeff(:, :, n_frequencyBand, i_patches) = ...
            PSDmov(:, :, n_frequencyBand, i_patches) / PSDref(:, :, n_frequencyBand, i_patches) * PSDrescal_zero(:, :, n_frequencyBand);

    end

    % First entry of the re-scaled PSD
    PSDrescal_res(1 : n_outputsRef, :, n_frequencyBand) = PSDrescal_zero(:, :, n_frequencyBand);
```

```
    % Construction of the re-scaled PSD with the previous calculated
    % coefficients.
    for i_patches = 1 : n_patches

    PSDrescal_res((i_patches - 1) * n_outputsMov + n_outputsRef + 1 : i_patches * n_outputsMov + n_outputsRef, :, n_frequencyBand)...
        = PSDrescal_coeff(:, :, n_frequencyBand, i_patches);

    end

end

%% 3. Rearrangement of the PSD matrix

PSD_multiSensor = permute(PSDrescal_res, [3, 2, 1]);
end
```

11.4 Programmcode: pLSCF_findPoles

```
function [poles_fDomain] =...
    pLSCF_findPoles_calc(n_modelOrder, modelOrder_max, n_outputs, n_outputsRed, deltaTime, matrixGamma_max, matrixYpsilon_max)

%% 1. Calculation of matrices [matrixR_o], [matrixS_o] and [matrixT_o]
matrixGamma = matrixGamma_max(:, 1 : n_modelOrder + 1);

matrixYpsilon = matrixYpsilon_max(:, 1 : (n_modelOrder + 1) * n_outputsRed, :);
% Computation of matrixR_o
matrixR_o(:,:) = matrixGamma' * matrixGamma;

% Preallocation
matrixS_o = zeros(n_modelOrder + 1, (n_modelOrder + 1) * n_outputsRed, n_outputs);
matrixT_o = zeros((n_modelOrder + 1) * n_outputsRed, (n_modelOrder + 1) * n_outputsRed, n_outputs);

% Computation of matrixS_o and matrixT_o
for i_outputs = 1 : n_outputs

    matrixS_o(:, :, i_outputs) = matrixGamma' * matrixYpsilon(:, :, i_outputs);

    matrixT_o(:, :, i_outputs) = matrixYpsilon(:, :, i_outputs)' * matrixYpsilon(:, :, i_outputs);

end

% The searched coefficients of the common denominator model are assumpted
% to be real valued. Through that, only the real parts of the matrices
% matrix_R_o, matrix_R_o and matrix_R_ has to be considered.
matrixR_o = real(matrixR_o);
matrixT_o = real(matrixT_o);
matrixS_o = real(matrixS_o);

%% 2. Calculation and solving the reduced normal equation [matrixAlpha]

% Preallocation
matrixM = zeros((n_modelOrder + 1) * n_outputsRed, (n_modelOrder + 1) * n_outputsRed, n_outputs);

% The matrix_M describes the reduced normal equation as follows:
% [matrix_M][matrix_alpha] = [0]. This is the system of equations which
% has to be solved. The result for the matrix [matrix_alpha] contains the
% searched coefficents of the common denominator model.
for i_outputs = 1 : n_outputs

    matrixM(:, :, i_outputs) =...
        matrixT_o(:, :, i_outputs) - (matrixS_o(:, :, i_outputs)') / (matrixR_o(:, :)) * matrixS_o(:, :, i_outputs);

end

% Sum over all output sensors
matrixM = sum(matrixM, 3);

matrixAlpha =...
    - (matrixM(n_outputsRed + 1 : (n_modelOrder + 1) * n_outputsRed, n_outputsRed + 1 : (n_modelOrder + 1) * n_outputsRed))...
    \ (matrixM(n_outputsRed + 1 : (n_modelOrder + 1) * n_outputsRed, 1 : n_outputsRed));
```

```
% The lowest order coefficent is constrained to be equal to the identity
% matrix of dimension n_outputs x n_outputs. Through that, the parameter
% redundancy is removed.
matrixAlpha = cat(1,eye(n_outputsRed), matrixAlpha);
```

%% 3. Calculation of the denominator coefficients

```
% Preallocation
matrixA_o = zeros(n_outputsRed, n_outputsRed);

% Extracting of the searched coefficients of the common denominator model
% from the matrix [matrix_A_o]. The coefficients are described through
% matrices of dimension n_outputs x n_outputs.

for i_modelOrder = 0 : n_modelOrder

    matrixA_o(:, :, i_modelOrder + 1)= ...
        matrixAlpha(i_modelOrder * n_outputsRed + 1 : (i_modelOrder + 1) * n_outputsRed, 1 : n_outputsRed);

end
```

%% 4. Calculation of the companion matrix

```
% The roots of the denominator polynomial are the eigenvalues of the
% corresponding companion matrix. The companion matrix is a square
% n_outputs * n_modelOrder x n_outputs * n_modelOrder matrix and models a
% dynamic system with n_outputs * n_modelOrder / 2 modes.

%Preallocation
companionMatrix_coeff = zeros(n_outputsRed, n_outputsRed, n_modelOrder);

for i_modelOrder = 1 : n_modelOrder

    companionMatrix_coeff(:, :, i_modelOrder) =...
        - (matrixA_o(:, :, n_modelOrder + 1)) \ matrixA_o(:, :, n_modelOrder + 1 - i_modelOrder);

end

%Preallocation
companionMatrix = zeros(n_outputsRed * n_modelOrder, n_outputsRed * n_modelOrder);

% Construction of the companion matrix of the denominator polynomial
for i_modelOrder = 1 : n_modelOrder

    companionMatrix(1 : n_outputsRed, (i_modelOrder - 1) * n_outputsRed + 1: i_modelOrder * n_outputsRed) =...
        companionMatrix_coeff(:, :, i_modelOrder);

end

% Filling the matrix with identity matrices
companionMatrix(n_outputsRed + 1 : n_modelOrder * n_outputsRed, 1 : n_outputsRed * n_modelOrder - n_outputsRed) =...
    eye(n_modelOrder * n_outputsRed - n_outputsRed);
```

%% 5. Estimation of the eigenfrequencies

```
% Poles in z-domain
poles_zDomain = eig(companionMatrix);

% Filling with zeros so that returned vectors have always the same length
poles_fDomain = zeros(modelOrder_max * n_outputsRed, 1);

% Transformation of the poles into frequency-domain
poles_fDomain(1 : size(poles_zDomain)) = log(poles_zDomain) / deltaTime;
```

11.5 Programmcode OMA_pLSCF_modeShapes.m

```
function [complexModes, residue, capitalLambda] = ...
    OMA_pLSCF_modeShapes_calc(physicalStablePoles_sort,PSDoFRF, frequencyBand)
% Number of output channels
n_outputs = size(PSDoFRF, 3);

% For multi sensor layout: Reduced dimension (only reference sensors)
n_outputsRed = size(PSDoFRF, 2);
% Number of major stable poles
```

```
n_physicalStablePoles = size(physicalStablePoles_sort, 1);

% Number of discrete frequencies
n_frequencyBand = size(frequencyBand,1);
```

%% 1. Calculation of capitalLambda

```
% Preallocation
capitalLambda_coeff1 = zeros(n_frequencyBand * n_outputs, n_outputs, n_physicalStablePoles);
capitalLambda_coeff2 = zeros(n_frequencyBand * n_outputs, n_outputs, n_physicalStablePoles);

% Computation of capitalLambda
for i_physicalStablePoles = 1 : n_physicalStablePoles
    for i_frequencyBand = 1 : n_frequencyBand

        capitalLambda_coeff1((i_frequencyBand - 1) * n_outputs + 1 : i_frequencyBand * n_outputs, :, i_physicalStablePoles) = ...
            eye(n_outputs) ./ (1i * 2 * pi * frequencyBand(i_frequencyBand) - physicalStablePoles_sort(i_physicalStablePoles));

        capitalLambda_coeff2((i_frequencyBand - 1) * n_outputs + 1 : i_frequencyBand * n_outputs, :, i_physicalStablePoles) = ...
            eye(n_outputs) ./ (1i * 2 * pi * frequencyBand(i_frequencyBand) - conj(physicalStablePoles_sort(i_physicalStablePoles)));

    end
end

% The matrices capitalLambda_coeff3 and capitalLambda_coeff4 are complex conjugates of
% the previously calculated coefficients.
capitalLambda_coeff3 = conj(capitalLambda_coeff2);
capitalLambda_coeff4 = conj(capitalLambda_coeff1);

% Preallocation
capitalLambda = zeros(n_outputs * n_frequencyBand, 4 * n_outputs * n_physicalStablePoles);

for i_physicalStablePoles = 1 : n_physicalStablePoles

    capitalLambda(:, (i_physicalStablePoles - 1) * n_outputs + 1 : i_physicalStablePoles * n_outputs) =...
        capitalLambda_coeff1(:, :, i_physicalStablePoles);

    capitalLambda(:, (i_physicalStablePoles - 1) * n_outputs + 1 + n_outputs * n_physicalStablePoles : i_physicalStablePoles * n_outputs +
n_outputs * n_physicalStablePoles) =...
        capitalLambda_coeff2(:, :, i_physicalStablePoles);

    capitalLambda(:, (i_physicalStablePoles - 1) * n_outputs + 1 + 2 * n_outputs * n_physicalStablePoles : i_physicalStablePoles * n_outputs
+ 2 * n_outputs * n_physicalStablePoles) =...
        capitalLambda_coeff3(:,:,i_physicalStablePoles);

    capitalLambda(:, (i_physicalStablePoles - 1) * n_outputs + 1 + 3 * n_outputs * n_physicalStablePoles : i_physicalStablePoles*n_outputs +
3 * n_outputs * n_physicalStablePoles) =...
        capitalLambda_coeff4(:, :, i_physicalStablePoles);
end
```

%% 2. Calculation of gLambda

```
% Preallocation
gLambda = zeros(i_frequencyBand * n_outputs, n_outputsRed);

% Calculation of gLambda
for i_frequencyBand = 1 : n_frequencyBand

    gLambda((i_frequencyBand - 1) * n_outputs + 1 : i_frequencyBand * n_outputs, :) =...
        permute(PSDoFRF(i_frequencyBand, :, :), [3, 2, 1]);

end
```

%% 3. Estimation of the mode shapes

```
% The calculation of the residue can suffer from a rank deficient matrix.
% But the warning is off, because this doesn't affect the results much.
% warning('off','MATLAB:rankDeficientMatrix')

% Calculation of the residues
residue = capitalLambda \ gLambda;

% Preallocation
complexModes = zeros(n_outputs, n_physicalStablePoles);
```

```
% Following loop calculates the SVD of all the coefficients the residue
% matrix contains  Every first column of the corresponding U Vectors is an
% estimate of the corresponding modeshape vector.

for i_physicalStablePoles = 1 : n_physicalStablePoles

    [uVector, ~, ~] = svd(residue((i_physicalStablePoles - 1) * n_outputs + 1 : n_outputs * i_physicalStablePoles, :));

    complexModes(:, i_physicalStablePoles) = uVector(:, 1);

end
end
```

11.6 Programmcode: EMA_pLSCF_modeShapes.m

```
function [complexModes, residue, capitalLambda] = ...
    EMA_pLSCF_modeShapes_calc(physicalStablePoles_pLSCF,PSDoFRF, frequencyBand)

% Number of major stable poles
n_physicalStablePoles = size(physicalStablePoles_pLSCF, 1);

n_frequencyBand = size(frequencyBand,1);

% The vector physicalStablePoles_all_con contains in the first
% n_physicalStablePoles entries the complex poles and in second
% n_physicalStablePoles entries the complex conjugate ot the complex poles.
physicalStablePoles_pLSCF_all = cat(1, physicalStablePoles_pLSCF, conj(physicalStablePoles_pLSCF));

% Preallocation
capitalLambda = zeros(n_frequencyBand, 2 * n_physicalStablePoles);

% Computation of gLambda
for i_physicalStablePoles = 1 : 2 * n_physicalStablePoles
    capitalLambda(:,i_physicalStablePoles) =...
        1 ./ (1i * 2 * pi * frequencyBand(:) - physicalStablePoles_pLSCF_all(i_physicalStablePoles));
end

% Calculation of the lower and upper residue
capitalLambda(:, i_physicalStablePoles + 1) = 1;
capitalLambda(:, i_physicalStablePoles + 2) = -1 ./ (1i*2*pi*frequencyBand(:)) .^ 2;

% Avoiding dividing with zero
capitalLambda(1, i_physicalStablePoles + 2) = capitalLambda(2, i_physicalStablePoles + 2);

% Rearrangement of the FRF matrix to an 2-D matrix. First dim is frequency,
% second dim sensors.
FRF_res = squeeze(PSDoFRF);

% Computation of the residues
residue = capitalLambda \ FRF_res;

% Computation of the complex modeshapes
complexModes = residue(1 : n_physicalStablePoles, :).';

end
```

11.7 Programmcode: BFD_modeShapes_calc.m

```
function [complexMode_normalized_BFD] = BFD_modeShapes_calc(PSDoFRF, frequencyBand, naturalFrequencies)
% Basic values:

% Number of output channels
n_outputs = size(PSDoFRF, 3);

% Reference sensor for mode shape estimation
refPoint = 1;
% Numver of natural frequencies
n_naturalFrequencies_pLSCF = size(naturalFrequencies, 1);
```

```
%% 1. Finding the indices of the natural frequencies

%Preallocation
autoPSDoFRF_peaksIndex = zeros(n_naturalFrequencies_pLSCF, 1);

% Identification of the peaks in the set intervals.
for i_naturalFrequencies_pLSCF = 1 : n_naturalFrequencies_pLSCF

    % Identifying the frequencies of the frequency band which are
    % nearest to the set intervall.
    [~, autoPSDoFRF_peaksIndex(i_naturalFrequencies_pLSCF)] = ...
        min(abs(frequencyBand-naturalFrequencies(i_naturalFrequencies_pLSCF)));

end

%% 2. Estimation of the mode shapes

% Preallocation
complexMode = zeros(n_outputs, n_naturalFrequencies_pLSCF);

% Identifying the mode comples modeshapes corresponding to the identified
% poles respectively natural frequencies
for i_naturalFrequencies_pLSCF = 1 : n_naturalFrequencies_pLSCF

    complexMode(:, i_naturalFrequencies_pLSCF) = PSDoFRF(autoPSDoFRF_peaksIndex(i_naturalFrequencies_pLSCF), refPoint, :);

end

%As FDD Method:
for i_naturalFrequencies_pLSCF = 1 : n_naturalFrequencies_pLSCF

    PDS_fequLine = zeros(n_outputs);

    for i_outputs = 1 : n_outputs
        PDSoFRF_fequLine(i_outputs,:) = PSDoFRF(autoPSDoFRF_peaksIndex(i_naturalFrequencies_pLSCF),:,i_outputs); %#ok<SAGROW>
    end

    [uVector, ~, ~] = svd(PDSoFRF_fequLine);

    complexMode(:,i_naturalFrequencies_pLSCF) = uVector(:,1);
end

% Calling the function modeNormalized_calc to obtain the normalized mode
% shapes complexMode_normalized and the real valued modeshape
% mode_normalized.
[complexMode_normalized_BFD, ~] = modeNormalized_calc(complexMode);
end
```

11.8 Programmcode: modeNormalized_calc.m

```
function [complexModes_normalized, modes_normalized] = modeNormalized_calc(complexModes)

% Returns maximum value (= maximum magnitude) of each complexMode estimation.
[amplitudeModes_max] = max(complexModes);

% Normalization of the complex modeshape vetors by dividing each modeshape by its maximum entry
complexModes_normalized = complexModes ./ amplitudeModes_max;

% Normalized modes are the real part of complexMode_normalized
modes_normalized = abs(real(complexModes_normalized));

 % Phase
phase_complex = angle(complexModes_normalized);

for i_physicalStablePoles = 1 : size(complexModes, 2)

    isInPhase = (phase_complex(:,i_physicalStablePoles) <= pi/2) & (phase_complex(:,i_physicalStablePoles) >= -pi/2);

    modes_normalized(isInPhase == 0, i_physicalStablePoles) = - modes_normalized(isInPhase == 0, i_physicalStablePoles);

end

end
```

11.9 Programmcode: MAC_calc.m

```
function [MACvalue] = MAC_calc(complexMode_normalized_pLSCF,complexMode_normalized_BFD)

% Shorter names for the mode shape estiamtes
modeShapeA = complexMode_normalized_pLSCF;
modeShapeB = complexMode_normalized_BFD;

% Number of different poles / mode shape
n_modeShapeA = size(complexMode_normalized_pLSCF, 2);
n_modeShapeB = size(complexMode_normalized_BFD, 2);

% Preallocation
MACvalue = zeros(n_modeShapeA,n_modeShapeB);

for i_modeShapeA = 1 : n_modeShapeA
    for i_modeShapeB = 1 : n_modeShapeB

        MACvalue(i_modeShapeA, i_modeShapeB)=...
            (abs(modeShapeA(:, i_modeShapeA)' * modeShapeB(:, i_modeShapeB))) ^ 2 /...
            ((modeShapeA(:, i_modeShapeA)' * modeShapeA(:, i_modeShapeA))...
            * (modeShapeB(:, i_modeShapeB)' * modeShapeB(:, i_modeShapeB)));

    end
end

end
```