Anforderungsdefinition (SRS) – Software zur akustischen Zweikanalanalyse

1. Einleitung

1.1 Zweck

Dieses Dokument dient als Leitfaden für die Entwicklung einer Software, die im Rahmen des Projekts *Python und Akustik* von einem vierköpfigen Team konzipiert, implementiert und validiert wird.

Ausgehend von der Aufgabenstellung, die in diesem Kontext als Entwicklungsauftrag dient, wird zunächst ein Lastenheft erstellt. Dieses definiert die übergeordneten Ziele der Software, beschreibt die daraus abgeleiteten Use Cases sowie die wesentlichen Rahmenbedingungen des Projekts, darunter Zeitrahmen, Teamgröße, eingesetzte Werkzeuge und potenzielle Kosten.

Aufbauend auf dem Lastenheft wird ein Pflichtenheft erarbeitet, das die Anforderungen in Form einer detaillierten Anforderungsliste konkretisiert. Diese Liste umfasst sämtliche funktionalen und nicht-funktionalen Anforderungen sowie alle technischen und organisatorischen Randbedingungen. Jede Anforderung wird hinsichtlich ihrer Priorität, Wichtigkeit und Stabilität bewertet. Zur Sicherstellung der Verfolgbarkeit werden alle Anforderungen sowohl mit den zugehörigen Use Cases (rückwärts) als auch mit abhängigen Anforderungen (vorwärts) verknüpft. Eine präzise Beschreibung jeder Anforderung gewährleistet deren Vollständigkeit und Nachvollziehbarkeit. Dieses Dokument bildet damit die Grundlage für die strukturierte Umsetzung und erfolgreiche Durchführung des Projekts.

1.2 Begriffe

- ***GUI***: Graphical User Interface
- ***FFT***: Schnelle Berechnung der Fourier-Transformation zur Frequenzanalyse zeitdiskreter Signale.
- ***HDF5***: Hierarchisches Dateiformat zur strukturierten Speicherung großer numerischer Daten.
- ***Autokorrelation***: Ähnlichkeit eines Signals mit sich selbst bei Zeitverschiebung
- ***Kreuzkorrelation***: Ähnlichkeit zwischen zwei Signalen bei Zeitverschiebung
- ***Frequenzgang***: Frequenzabhängiges Übertragungsverhalten eines Systems
- ***Kohärenz***: Maß für die lineare Beziehung zweier Signale im Frequenzbereich
- ***Impulsantwort***: Systemantwort auf einen idealen Dirac-Impuls
- ***H_1-Schätzer***: Verfahren zur Schätzung des Frequenzgangs bei störungsfreiem Eingangssignal

- ***Segmentierung***: Aufteilung eines Signals in zeitlich begrenzte Abschnitte zur separaten Analyse, z. B. bei der FFT.

2. Allgemeine Beschreibung

2.1 Hauptziele des Projekts

Es soll eine in Python entwickelte Software entstehen, die eine umfassende Zweikanalanalyse akustischer Signale ermöglicht. Gemäß der Aufgabenstellung sollen mit der zu entwickelnden Anwendung die folgenden zentralen Ziele erreicht werden:

- **Z01**: Das System soll akustische Signale aus zwei Kanälen einlesen und verarbeiten können.
- **Z02**: Das System soll eine vollständige Frequenzanalyse durchführen (inkl. FFT, Leistungsspektren, Korrelationsfunktionen, Kohärenz).
- **Z03**: Das System soll den Frequenzgang und die Impulsantwort des Schallübertragungssystems bestimmen.
- **Z04**: Das System soll die Ergebnisse visualisieren und als Grafiken exportierbar machen.
- **Z05**: Das System soll benutzerfreundlich über GUI bedienbar sein.
- **Z06**: Das System soll plattformunabhängig und performant sein.

2.2 Rahmenbedingungen des Projekts

Zeitrahmen:

Das Projekt „Python und Akustik“ an der TU Berlin umfasst ca. 10 bis 12 Wochen und gliedert sich in die Phasen Anforderungsanalyse, Implementierung, Validierung und Dokumentation.

Personelle Ressourcen:

Zur Einhaltung der Rahmenbedingungen wird das Projekt in einer vierköpfigen Gruppe bearbeitet. Neben fristgerechter Abgabe und dem Einsatz der geforderten technischen Mittel erfolgt die arbeitsteilige Umsetzung wie folgt:

- Alexander Nehls (GUI)
- Joel Colin Christ (Klasse zur Spektralanalyse)
- Gwendal Benjamin Jacob Frühauf (Abschlusspräsentation, 50% Software-Dokumentation)
- Omar Ben Salem (Anforderungsdefinition, 50% Software-Dokumentation)

Technische Mittel:

- Python 3.11
- Bibliotheken: Acoular, NumPy, SciPy, Matplotlib, soundfile, Bokeh[1], base64[2]
- Formate: WAV (Input), HDF5 (intern), PNG (Output)

Kosten:

Durch die Verwendung von Open-Source-Software entstehen keine Lizenzkosten.

Benutzergruppen:

Akustik-Forschung, Studierende der technischen Akustik oder Entwickler akustischer Systeme

Einschränkungen:

- Nur Offline-Analyse (kein Live-Modus)
- WAV-Dateien aus genau 2 Kanälen
- Abtastraten beliebig, mit optionalem Downsampling

2.3 Use Cases

Ausgehend von den Systemzielen und der technischen Beschreibung des Zweikanalanalysators gemäß Vorlesung Schallmesstechnik wurden Use Cases formuliert. Diese beschreiben die Interaktion zwischen Benutzer und System und bilden die Grundlage für funktionale Anforderungen sowie deren Rückverfolgbarkeit im Anforderungsmanagement. Im Folgenden sind die Use Cases aufgeführt:

UC01 – Audiosignale einlesen lassen

- **Akteur:** Benutzer

- **Ziel:** Der Benutzer lädt zwei zu analysierende Audiosignale als WAV-Dateien in das System hoch

- **Vorbedingung:** Anwendung ist geladen

[1] Bokeh: Python-Bibliothek zur Erstellung interaktiver Visualisierungen für Webanwendungen
[2] Base64: Verfahren zur Kodierung binärer Daten (z. B. Audiodateien) als ASCII-Text, häufig für Webübertragungen verwendet

- **Ergebnis:** Die gewählten Audiodateien werden als aktuell zu analysierende Signale unter festen Dateinamen auf dem jeweiligen System gespeichert, auf dem die App läuft

- **Ausnahme:** Keine

UC02 – Vorverarbeitungsparameter konfigurieren

- **Akteur:** Benutzer

- **Ziel:** Der Benutzer wählt optional einen Downsampling Faktor zur Reduzierung der Anzahl an Samples und somit automatisch die Auflösung der Plots zu erhöhen

- **Vorbedingung:** Anwendung ist geladen

- **Ergebnis:** Der Faktor ist im System gespeichert und steht bei Analysebeginn zur Verfügung

- **Ausnahme:** Keine

UC03 – Analyse starten

- **Akteur:** Benutzer

- **Ziel:** Der Benutzer startet die Zweikanalanalyse. Das System führt dann automatisch die Vorverarbeitung sowie alle Berechnungen (FFT, Leistungsspektren, Korrelationen, Kohärenz, Frequenzgang, Impulsantwort) durch und zeigt die Ergebnisse im GUI an

- **Vorbedingung:** Zwei Audiosignale sind vorhanden, optional ist ein Downsampling Faktor bekannt

- **Ergebnis:** Die Analyse ist abgeschlossen und Ergebnisse sind interaktiv im Browser sichtbar und exportierbar.

- **Ausnahme:** Ungültige Daten oder Signalprobleme wie z.B. Formatfehler, unterschiedliche Abtastfrequenzen → Fehlermeldung (im Terminal)

Basierend auf den vorher beschriebenen Use Cases folgt nun eine detaillierte Anforderungsliste, die als Grundlage für die Umsetzung und Projektabwicklung dient.

Prof. Dr.-Ing. Ennes Sarradj Projektarbeit Python & Akustik **Technische Universität Berlin**					**Anforderungsliste** für **Zweikanalanalysator - App**	**Auftrag:** Projekt Python & Akustik SS25 **Auftragsnummer:** **PA-SS25-ZKA**
Organisations-daten		**Prozessdaten**			**Anforderungen**	**Beschreibung - Quantifizierung**
Nr.	Name	Art	Wichtig-keit	Stabilität		
					Funktionale Anforderungen	
F01	JO	M	5	5	Audiodateien importieren und aufbereiten UC01-02	**Ein**: Zwei **WAV**-Dateien, sowie ein optionaler Downsampling-Faktor. Speichert Kopien von den Audiosignalen im Arbeitsverzeichnis. Importiert die zwei Signale und bereitet diese ggf. auf die Analyse vor (Downsampling mit dem gewählten Faktor). **Aus**: Signale (roh, nicht aufbereitet) gespeichert im Arbeitsverzeichnis als WAV-Dateien (gekennzeichnet als aktuelle Audiosignale) und **aufbereitete** Signale (ggf. downsampled) intern in der Klasseninstanz gespeichert: - als zwei **(np.ndarray (1D, float))** **-** zusätzlich als ein **acoular.TimeSamples-Objekt** mit zugehöriger Abtastrate **(int)** für weitere Berechnungen mit Acoular.
F02	JO	M	5	5	Finite Fourier-Transformation vorbereiten Segmentweise Real-FFT UC03	**Ein**: Acoular-Objekt **(acoular.TimeSamples),** Ergebnis aus **F01**. Instanziert einen Acoular-FFT-Generator mit den zu analysierenden Signalen als Quelldaten. **Aus**: **(Intern)** Acoular-Objekt **(acoular.spectra.RFFT)**, ein **Acoular-Generator** für Frequenzdaten, enthält die Einstellungen der **FFT** (**Quelldaten** und **Segmentlänge** u.a.), bereitet auf die Durchführung der **FFT** vor.
F03	JO	M	5	5	Auto- und Kreuzleistungsspektrum berechnen $C_{xx}(f)$, $C_{yy}(f)$, $C_{xy}(f)$ UC03	**Ein**: **Acoular-Objekt (acoular.spectra.RFFT)** Generator aus **F02**. Instanziert ein Acoular-Generator für die Berechnung der Leistungsspektren **(acoular.spectra.CrossPowerSpectra)**, der auf den RFFT-Generator aus **F02** aufbaut. Ebenso wird einen Acoular-Generator **(acoular.process.Average)** instanziert, um die Scharmittelung über die Segmente zu gewährleisten. Berechnet eine Kreuzspektralmatrix, die sowohl die Auto- und Kreuzleistungsspektren enthält. **Aus**: **(Intern)** Auto- und Kreuzleistungsspektren als jeweils **(np.ndarray (1D, Complex))**, Einheit **[Amplitude^2] (nicht kalibriert).**
Ausgabe vom: 20.07.2025 Ers. Ausgabe vom: 11.07.2025 Bearbeiter: BEN SALEM					Art: M = Must, O = Optional, E = Extra Wichtigkeit: 1–5 (1 = unwichtig, 5 = kritisch) Stabilität: 1–5 (1 = instabil, 5 = stabil) Bearbeiter: JO = Joel, AL = Alex, BE = Benjamin, OM = Omar	Version: 3.0 (final)

Prof. Dr.-Ing. Ennes Sarradj Projektarbeit Python & Akustik **Technische Universität Berlin**					**Anforderungsliste** für **Zweikanalanalysator - App**	**Auftrag:** Projekt Python & Akustik SS25 **Auftragsnummer:** **PA-SS25-ZKA**
Organisations-daten		**Prozessdaten**			**Anforderungen**	**Beschreibung - Quantifizierung**
Nr.	Name	Art	Wichtig-keit	Stabilität		
					Funktionale Anforderungen	
F04	JO	M	5	5	Kohärenz berechnen $\gamma^2(f) = \lvert C_{xy}(f)\rvert^2 / (C_{xx}(f) * C_{yy}(f))$ UC03	**Ein**: Auto- und Kreuzleistungsspektren als **(np.ndarray (1D, Complex))** aus F03. Berechnet die Kohärenz nach der links angegebenen Formel **Aus**: **(intern)** Kohärenzfunktion als **(np.ndarray (1D, float))**, **einheitenlos**, Werte in [0, 1].
F05	JO	M	5	5	Auto- und Kreuzkorrelationsfunktion berechnen $\Psi_{xx}(\tau), \Psi_{yy}(\tau), \Psi_{xy}(\tau)$ UC03	**Ein**: intern gespeicherte Signale als zwei **(np.ndarray (1D, float))** in der Klasseninstanz aus **F01**. Berechnet die Auto- und Kreuzkorrelationfunktionen von den zwei Signalen mit scipy.signal.correlate() . **Aus**: **(intern)** Auto- und Kreuzkorrelationsdaten insgesamt als 3 **(np.ndarray (1D, float)**, **[einheitenlos] (nicht normiert).** **(intern)** Die Verzögerungsachse als **np.ndarray (1D, float)** (Einheit: **Samples**)
F06	JO	M	5	5	Frequenzgang berechnen $H_1(f) = C_{xy}(f) / C_{xx}(f)$	**Ein**: Auto- und Kreuzleistungsspektren als **(np.ndarray (1D, Complex))** aus F03. Schätzt den Frequenzgang vom analysierten Schallübertragungssystem (stets als Black-Box angenommen, da nur die Audiosignale vorliegen) mit dem H_1-Schätzer ab. **Aus**: **(intern)** Der Frequenzgang als **(np.ndarray (1D, Complex))**, **einheitenlos**
Ausgabe vom: 20.07.2025 Ers. Ausgabe vom: 11.07.2025 Bearbeiter: BEN SALEM					Art: M = Must, O = Optional, E = Extra Wichtigkeit: 1–5 (1 = unwichtig, 5 = kritisch) Stabilität: 1–5 (1 = instabil, 5 = stabil) Bearbeiter: JO = Joel, AL = Alex, BE = Benjamin, OM = Omar	Version: 3.0 (final)

Prof. Dr.-Ing. Ennes Sarradj Projektarbeit Python & Akustik **Technische Universität Berlin**					**Anforderungsliste** für **Zweikanalanalysator - App**	**Auftrag:** Projekt Python & Akustik SS25 **Auftragsnummer:** **PA-SS25-ZKA**
Organisations-daten		**Prozessdaten**			**Anforderungen**	**Beschreibung - Quantifizierung**
Nr.	Name	Art	Wichtig-keit	Stabilität		
					Funktionale Anforderungen	
F07	JO	M	5	5	Impulsantwort berechnen Inverse - Real FFT $h(\tau)$ UC03	**Ein**: Der Frequenzgang als **(np.ndarray (1D, Complex))** aus **F06**. Berechnet die Impulsantwort durch Anwendung der inversen Real-FFT auf den komplexen Frequenzgang H(f). **Aus**: **(intern)** Die rekonstruierte Impulsantwort im Zeitbereich als **np.ndarray (1D, float), einheitenlos**
F08	AL	M	5	5	Analyseergebnisse visualisieren UC03	**Ein**: **Analyseergebnisse aus F02 bis F07** (Spektren, Korrelationsfunktionen, Kohärenz etc.), jeweils als reale oder komplexe Arrays **(np.ndarray (1D, float)) oder (np.ndarray (1D, komplex))** Erlaubt die Visualisierung der Analyseergebnisse als interaktive Bokeh-Grafiken **Aus**: Darstellung als interaktive **Bokeh-Grafik** (Linienplot, ggf. mit Achsenbeschriftung, Legende, Zoom)
F09	AL	M	5	5	Analyseergebnisse exportieren UC03	**Ein**: **interaktive Bokeh-Grafiken** aus **F08.** Erlaubt das Exportieren der Grafiken als .png **Aus**: Exportierte Bokeh-Grafiken als **.png**
F10	AL	M	5	5	Benutzeroberfläche (GUI) bereitstellen UC01-03	Software ermöglicht Benutzerinteraktionen via ein **Bokeh-GUI.** Das **GUI** steuert die Auswahl der Signale, den Slider (Für Downsampling Faktor) und stellt ein Start-Button zur Ausführung bereit. Das GUI ermöglicht die Visualisierung der Ergebnisse als interaktive Bokeh-Grafiken, die auf Anweisung des Benutzers exportierbar sind.
Ausgabe vom: 20.07.2025 Ers. Ausgabe vom: 11.07.2025 Bearbeiter: BEN SALEM					Art: M = Must, O = Optional, E = Extra Wichtigkeit: 1–5 (1 = unwichtig, 5 = kritisch) Stabilität: 1–5 (1 = instabil, 5 = stabil) Bearbeiter: JO = Joel, AL = Alex, BE = Benjamin, OM = Omar	Version: 3.0 (final)

Prof. Dr.-Ing. Ennes Sarradj Projektarbeit Python & Akustik **Technische Universität Berlin**					**Anforderungsliste** für **Zweikanalanalysator - App**	**Auftrag:** Projekt Python & Akustik SS25 **Auftragsnummer:** **PA-SS25-ZKA**
Organisations-daten		**Prozessdaten**			**Anforderungen**	**Beschreibung - Quantifizierung**
Nr.	Name	Art	Wichtig-keit	Stabilität		
					Optionale funktionale Anforderungen	
F11	JO	O	3	3	Anpassung der Abtastrate UC01	*Die ursprünglich geplante „anpassbare Abtastrate“ wurde durch einen Downsampling-Mechanismus beim Import (F01) umgesetzt.*
F12	AL	O	4	4	Selektive Darstellung von Ergebnissen UC03	Der Benutzer kann bestimmte Ergebnisse ausblenden lassen
F13	JO	E	1	1	Vorfilterung (Hoch-/Tief-/Bandpass) UC02	Der Benutzer kann bestimmte Frequenzen in den Audiosignalen filtern (Hoch-, Tief und Bandpassfilter)
F14	JO	E	2	1	Störgrößenfilterung UC02	Der Benutzer kann eine bekannte Störfrequenz filtern
F15	JO	E	2	1	Auswahl alternativer Spektralanalyseverfahren UC03	Der Benutzer kann von einem FFT- zu einem STFT-Algorithmus wechseln um zeitabhängige Analyse zu ermöglichen
F16	AL	E	2	1	Spektrogramme erzeugen UC03	Das System soll eine Option bieten, aus einem Signal interaktiv ein zeitfrequenzaufgelöstes Spektrogramm zu erzeugen
F17	AL	E	2	1	Anpassung des Diagrammlayouts UC03	Layout-Elemente der Visualisierung individuell anzupassbar – darunter Achsenskalierung, Farbschemata, Legenden, Diagrammgröße und Auflösung.
Ausgabe vom: 20.07.2025 Ers. Ausgabe vom: 11.07.2025 Bearbeiter: BEN SALEM					Art: M = Must, O = Optional, E = Extra Wichtigkeit: 1–5 (1 = unwichtig, 5 = kritisch) Stabilität: 1–5 (1 = instabil, 5 = stabil) Bearbeiter: JO = Joel, AL = Alex, BE = Benjamin, OM = Omar	Version: 3.0 (final)

Prof. Dr.-Ing. Ennes Sarradj Projektarbeit Python & Akustik Technische Universität Berlin					**Anforderungsliste** für **Zweikanalanalysator - App**	**Auftrag:** Projekt Python & Akustik SS25 **Auftragsnummer:** **PA-SS25-ZKA**
Organisations-daten		**Prozessdaten**			**Anforderungen**	**Beschreibung - Quantifizierung**
Nr.	Name	Art	Wichtig-keit	Stabilität		
					Qualitätsanforderungen	
Q01	JO / AL	M	5	5	Modularer Aufbau des Codes	Code in Klassen gegliedert
Q02	JO / AL	M	5	5	Plattformunabhängig	Python code
Q03	OM/BE	M	5	5	Dokumentierte Software	Sphinx-Dokumentation
Q04	JO/AL	M	4	4	Leistung / Rechenzeit	Startet in weniger als 7s. Ergebnissewerden in weniger als 3s angezeigt.
Q05	JO/AL	M	4	4	Reproduzierbarkeit	Ergebnisse sind deterministisch bei gleichen Eingangsdaten
Q06	AL/OM	M	4	4	Datenintegrität	Eingabedateien werden dupliziert, originale bleiben unangetastet
Q07	AL	O	2	3	Benutzerfreundlichkeit	Intuitives GUI, klarer Aufbau
Q08	JO/AL	E	2	1	Nachvollziehbarkeit	Zwischenschritte im code speichern
Q09	AL	E	2	3	Lesbarkeit der Ergebnisse	Darstellung der Analyseergebnisse soll durch eine interne Glättung visuell optimiert werden
					Randbedingungen	
R01		M	5	5	Programmiersprache	Python 3.11
R02		M	5	5	Bibliotheken	Acoular, NumPy, SciPy, Bokeh, Sphinx u.a.
Ausgabe vom: 20.07.2025 Ers. Ausgabe vom: 11.07.2025 Bearbeiter: BEN SALEM					Art: M = Must, O = Optional, E = Extra Wichtigkeit: 1–5 (1 = unwichtig, 5 = kritisch) Stabilität: 1–5 (1 = instabil, 5 = stabil) Bearbeiter: JO = Joel, AL = Alex, BE = Benjamin, OM = Omar	Version: 3.0 (final)