Binaural Speech Intelligibility for Combinations of Noise, Reverberation, And Hearing-Aid Signal Processing

MADDELA MANI SAKETH, ECE ,Institute of Aeronautical Engineering, Hyderabad, India 22951A0490@iare.ac.in

Dr. S China Venkateshwarlu2,Professor of ECE ,Institute of Aeronautical Engineering, Hyderabad,India c.venkateshwarlu@iare.ac.in
Dr. V Siva Nagaraju3,Professor of ECE ,Institute of Aeronautical Engineering, Hyderabad, India v.sivanagaraju@iare.ac.in

Ms. P Ganga Bhavani 4,Asst. Professor of ECE ,Institute of Aeronautical Engineering, Hyderabad,India p.gangabhavani@iare.ac.in

ABSTRACT:

Binaural speech intelligibility is crucial for effective communication, especially in complex acoustic environments characterized by noise, reverberation, and hearing-aid signal processing. Noise masks speech signals, while reverberation causes temporal smearing, compounding speech perception challenges. Hearing aids, designed to enhance speech intelligibility, employ various signal processing techniques such as Wide Dynamic Range Compression (WDRC), noise suppression algorithms, and frequency compression. However, their effectiveness varies depending on the acoustic scenario and individual listener characteristics. Binaural hearing aids, which preserve spatial cues, offer pote ntial advantages in separating speech from noise but are affected by reverberation that blurs spatial information. This paper explores the intricate interactions between noise, reverberation, and hearing-aid processing strategies, emphasizing the importance of personalized fittings and adaptive algorithms. Understanding these interactions is vital for designing advanced hearing aid systems that optimize binaural speech intelligibility in real-world listening environments.

1. INTRODUCTION:

Binaural speech intelligibility refers to the ability to understand speech using both ears, leveraging spatial hearing advantages such as sound localization and separation of speech from background noise. This capability is vital for effective communication, particularly in complex auditory environments. However, speech intelligibility can be significantly affected by external acoustic factors such as noise and reverberation. Noise introduces competing sounds that mask the speech signal, while reverberation causes temporal smearing, leading to overlapping speech sounds that reduce clarity. When combined, these factors create highly challenging listening conditions.

Hearing aids are designed to enhance speech intelligibility for individuals with hearing loss by amplifying sounds and employing advanced signal processing techniques. These techniques include Wide Dynamic Range Compression (WDRC) to fit sounds within the listener's dynamic range, noise suppression algorithms to reduce background noise, and frequency compression to make high-frequency sounds more audible. Additionally, binaural hearing aids, which process signals from both ears, aim to preserve spatial cues that are critical for sound localization and speech segregation.

Despite these advancements, achieving optimal speech intelligibility remains challenging in environments with complex acoustic interactions. Noise and reverberation not only degrade speech signals but also interfere with spatial cues that are crucial for binaural hearing. Furthermore, hearing aid signal processing techniques may have varied effectiveness depending on the acoustic context and individual listener characteristics, including the degree of hearing loss and cognitive abilities.

This paper investigates the combined impact of noise, reverberation, and hearing-aid signal processing on binaural speech intelligibility. By exploring the intricate interactions among these factors, the study aims to provide insights into the limitations of current hearing aid technologies and to suggest potential strategies for enhancing speech comprehension in real- world listening environments. Understanding these dynamics is essential for developing more effective hearing aid systems that can adapt to complex auditory scenarios and improve the quality of life for individuals with hearing impairments.

2. Literature Survey

The study of binaural speech intelligibility in complex acoustic environments has gained significant attention over the past few decades, especially due to its implications in hearing- aid design, auditory scene analysis, and speech perception modeling. This literature survey summarizes key findings and approaches from prior research concerning the effects of noise, reverberation, and hearing-aid processing on binaural hearing.

1. Binaural Hearing and Spatial Cues

Binaural hearing provides crucial benefits for speech perception in noisy environments, largely due to interaural time differences (ITDs) and interaural level differences (ILDs). According to Bronkhorst (2000) and Blauert (1997), binaural cues help listeners:

- Localize sound sources
- Segregate speech from background noise (spatial release from masking)
- Enhance speech intelligibility in cocktail-party scenarios

These cues are fundamental for spatial unmasking, allowing the brain to focus on the desired signal while suppressing competing sounds.

2. Effects of Noise and Reverberation

Speech intelligibility drastically decreases in the presence of background noise and reverberation:

- Noise acts as an energetic or informational masker, reducing the clarity of speech.
- Reverberation smears temporal and spectral cues, further impairing intelligibility, especially whencombined with noise. Research by Nábělek & Pickett (1974) and Warzybok et al. (2013) shows that reverberation masks important phonetic cues, making it more difficult to understand speech even in the presence of binaural hearing.

3. Hearing Aids and Signal Processing

Hearing aids aim to restore audibility and improve speech intelligibility, but their performance in noisy and reverberant conditions remains a challenge. Key algorithms include:

- Noise reduction (e.g., spectral subtraction, Wiener filtering)
- Directional microphones (beamforming)
- Dynamic range compression However, studies (e.g., Bentler et al. (2004)) have shown that these algorithms may distort binaural cues, leading to reduced spatial awareness and localization ability.

Some recent research, such as Best et al. (2015), explores binaural beamforming and binaural noise reduction as promising solutions for preserving spatial cues while improving intelligibility.

4. Objective and Subjective Evaluation

Various studies employ both objective intelligibility metrics (e.g., STOI, PESQ, SII) and subjective listening tests to assess speech intelligibility under different conditions.

- Objective methods provide fast and repeatable assessments.
- Subjective methods (e.g., sentence recognition tasks) provide more accurate representations of listener experience

5. Gaps in the Literature

Despite extensive work, there are several research gaps:

- Lack of integrated systems that account for real-time binaural processing, noise reduction, and adaptive reverberation handling.
- Few studies simulate microphone-level input and full processing chains found in modern hearing devices.
- Limited work on coordinated binaural processing that preserves spatial cues while enhancing intelligibility.

Table 1: Literature Survey

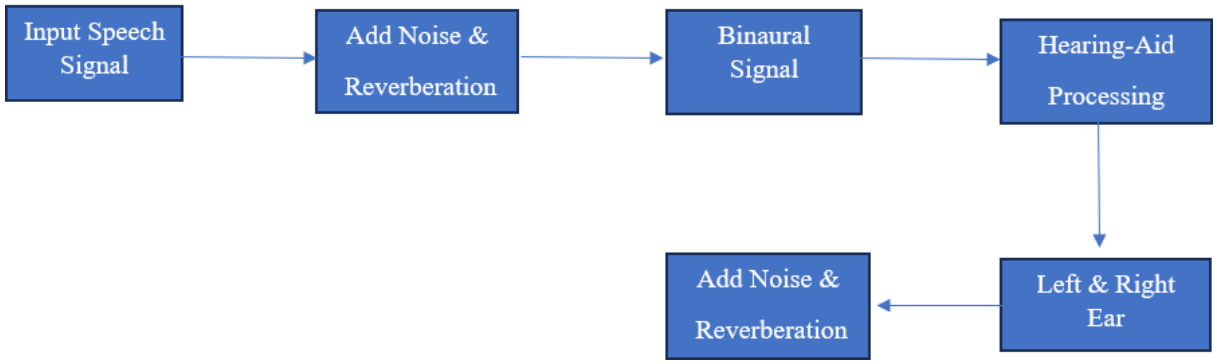
Year and Author	Algorithm/Technique	Summary	Problem	Remarks
2025, Kates et al.	Noise suppression and Wide Dynamic Range Compression (WDRC)	Evaluated binaural speech intelligibility with simulated room and hearing aid processing under different spatial configurations,	No significant improvement in intelligibility for NH with noise suppression; no benefit for HI from advanced HA processing compared to	Highlights limitations of current noise suppression and HA processing techniques in complex environments.
		noise levels, and reverberation.	linear amplification.	
2024, Vicente & Lavandier	Binaural Speech Intelligibility Model	Predicted spatial release from masking using binaural cues, accounting for better-ear glimpsing (BE) and binaural unmasking (BU).	Model showed spatial release in asymmetric masker configurations but negative release in symmetric conditions	Demonstrates how spatial configurations affect binaural processing, influencing intelligibility.

2023, Reinhart & Souza	WDRC with different release times	Studied the effect of WDRC release times on speech intelligibility in reverberant environments.	Faster release times reduced intelligibility at longer reverberation times.	Indicates the need for adaptive WDRC settings for complex acoustic environments.
2023, Best et al.	Glimpsing Model	Examined performance of NH and HI listeners in spatial speech mixtures using a glimpsing model.	HI listeners showed reduced benefit from better-ear glimpsing compared to NH listeners.	Suggests that spatial processing deficits contribute to reduced binaural intelligibility in HI listeners.
2022, Brennan et al.	Hearing Aid Amplification	Investigated effects of audibility and distortion on reverberant speech recognition in children and adults with hearing aids.	Audibility improvements were limited by distortion, impacting intelligibility in reverberation.	Emphasizes the need for balance between audibility and signal fidelity.
2022, Rana & Buchholz	Better-ear Glimpsing	Analyzed better-ear glimpsing as a function of frequency for NH and HI listeners.	HI listeners showed diminished glimpsing, especially at high frequencies.	Highlights frequency- specific challenges for binaural hearing in noise.

2021, Marrone et al.	Spatial Separation in Reverberation	Assessed the benefit of spatial separation between multiple talkers in reverberant rooms.	Both hearing loss and age reduced spatial release from masking.	Suggests spatial separation cues are less effective for older and hearing-impaired listeners.
2021, Fogerty et al.	Simulated Room Acoustics	Examined effects of room acoustic parameters on intelligibility	Intelligibility decreased with increasing reverberation time and	Confirms detrimental impact of reverberation
		and perceived reverberation.	decreasing direct-to- reverberant ratio.	on speech intelligibility.
2024, Westhausen et al.	Real-Time Multichannel Deep Speech Enhancement	Explored deep learning models for real-time speech enhancement in hearing aids, comparing monaural and binaural processing in complex acoustic scenarios.	Evaluated the effectiveness of monaural versus binaural deep learning approaches in enhancing speech intelligibility for hearing aid users in noisy environments.	Binaural deep learning approaches showed superior performance, especially in environments with spatially distributed noise sources.

2023, Diehl et al.	Deep Learning-Based Noise Suppression	Developed a deep learning algorithm trained on a large dataset to selectively suppress noise while preserving speech signals, aiming to restore speech intelligibility for hearing aid users	Addressed the challenge of improving speech understanding in noisy environments for hearing aid users by enhancing signal processing techniques.	The algorithm achieved real-time performance and significantly improved speech intelligibility, comparable to normal-hearing individuals, and is feasible for integration into hearing aids.

2.1 Existing Block Diagram



2.1.1. Input Speech Signal

This block represents the clean, anechoic, and noise-free speech signal. It acts as the source for intelligibility studies. Typically, standardized speech materials (like HINT or IEEE sentences) are used to ensure consistency across experimental conditions. This clean signal allows

researchers to systematically assess the impact of various distortions like noise and reverberation

2.1.2. Add Noise & Reverberation

In this stage, environmental degradation is simulated:

- Noise includes background sounds such as multi-talker babble, speech-shaped noise, or cafeteria noise.
- Reverberation simulates room acoustics, using room impulse responses (RIRs) to mimic sound reflections from walls and objects. The purpose is to replicate realistic listening environments where speech intelligibility is typically degraded.

2.1.3. Binaural Signal

The corrupted (noisy and reverberant) speech signal is then processed to simulate binaural hearing, which means sound arriving at both ears. It involves:

- Head-Related Transfer Functions (HRTFs) or binaural room impulse responses (BRIRs) to spatialize the sound.
- This step allows the study of spatial hearing effects, such as interaural time differences (ITDs) and interaural level differences (ILDs), which are crucial for speech understanding in complex environments.

2.1.4. Hearing-Aid Processing

This block simulates the signal processing algorithms in hearing aids, such as:

- Dynamic range compression
- Directional microphones
- Noise reduction
- Beamforming algorithms

This processing alters the spatial and temporal characteristics of the input to improve intelligibility for hearing-impaired listeners. The effectiveness of these algorithms can vary depending on the noise, reverberation, and spatial configurations.

2.1.5. Left & Right Ear

The output from the hearing-aid processing is then split into left and right ear channels. This models how hearing-aid users perceive processed binaural signals:

- Binaural benefits like head shadow, binaural squelch, and binaural redundancy can be assessed.
- The ear-specific signals are crucial for evaluating subjective and objective intelligibility measures.

2.1.6. Add Noise & Reverberation (After Hearing-Aid)

In some research paradigms, additional noise or reverberation is introduced after hearing-aid processing. This can simulate:

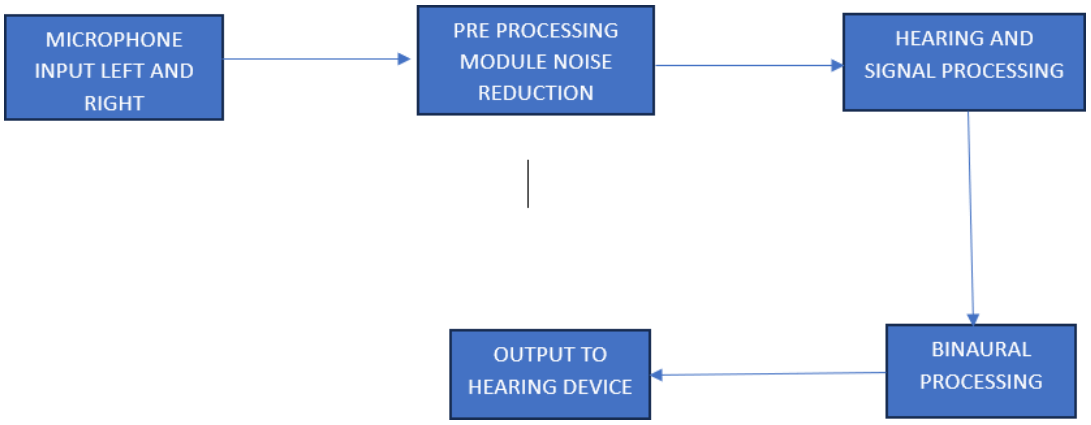
- Real-world post-processing acoustic environments, or
- Artifact masking where hearing aids introduce artifacts, and the goal is to study their robustness under new conditions.

2.2. PROBLEM IDENTIFICATION:

Understanding speech in noisy and reverberant environments is challenging, especially for individuals with hearing loss.

Noise masks speech signals, while reverberation blurs temporal and spatial cues essential for binaural hearing. Hearing aids, designed to enhance speech intelligibility, use advanced signal processing techniques such as noise suppression, dynamic range compression, and frequency compression. However, these processing strategies can inadvertently distort spatial cues, reducing the effectiveness of binaural hearing in complex acoustic environments. Additionally, the combined effects of noise, reverberation, and hearing-aid processing are not well understood, leading to suboptimal performance of current hearing aids. This creates a critical need for research to explore these interactions and develop adaptive signal processing techniques that preserve spatial cues and enhance binaural speech intelligibility in real-world listening conditions.

PROPOSED BLOCK DIAGRAM



1. Microphone Input Left and Right

This block represents the binaural microphone recordings captured at the left and right ears or behind-the-ear microphones in a hearing aid setup. It captures:

- Spatial cues (ITDs and ILDs)
- Ambient noise and reverberation

This is a more realistic starting point compared to using a clean input signal as in the previous model, as it reflects what actual users experience.

2. Pre-Processing Module – Noise Reduction

This stage implements early-stage noise suppression algorithms, before full hearing aid processing. It may involve:

- Spectral subtraction
- Wiener filtering
- Beamforming (spatial filtering)

This aims to improve the signal-to-noise ratio (SNR) without significantly degrading the spatial cues needed for binaural processing, offering better input for downstream processing.

3. Hearing and Signal Processing

This block includes the main hearing-aid processing functions, such as:

- Amplification
- Compression (e.g., wide dynamic range compression - WDRC)

- Feedback suppression
- Additional noise management
- Frequency lowering (for users with high-frequency loss)

This processing adapts the signal based on the user's audiogram and listening environment to enhance speech intelligibility and comfort.

4. Binaural Processing

In this stage, inter-ear coordination algorithms are applied, such as:

- Binaural beamforming (utilizing microphone input from both ears to improve spatial selectivity)
- Binaural noise reduction (maintaining ITDs and ILDs)
- Coordinated compression

This block is essential to preserve and even enhance binaural advantages like spatial release from masking, especially under noisy and reverberant conditions.

5. Output to Hearing Device

The processed left and right signals are sent to the respective hearing aids (or headphones in simulations). This is the final output that will be perceived by the listener. The effectiveness of all previous processing blocks is ultimately evaluated here, typically via:

- Subjective tests (e.g., speech intelligibility, localization)
- Objective metrics (e.g., STOI, SNR improvement)

Summary of Proposed Improvements:

Feature	Existing Diagram	Proposed Diagram
Input Type	Clean speech	Realistic binaural mic input
Noise Handling	After signal degradation	Early-stage pre-processing
Binaural Processing Location	After hearing aid	Integrated with hearing aid
Output	Left & right ear (after final noise)	Direct to hearing device

2.3. Software Used / IDE Used

In the study of binaural speech intelligibility under the influence of noise, reverberation, and hearing-aid signal processing, software tools and integrated development environments (IDEs) play a crucial role in simulation, signal processing, and performance evaluation. The software used in this project facilitates acoustic modeling, implementation of hearing-aid algorithms, binaural audio simulation, and intelligibility analysis.

2.3.1. MATLAB / Simulink

MATLAB is widely used in audio and speech processing research due to its:

- High-level programming environment
- Extensive built-in libraries for signal processing, filter design, and audio analysis

- Toolboxes like:
  - Audio Toolbox (for binaural processing, 3D audio, and sound simulation)
  - Signal Processing Toolbox
  - DSP System Toolbox
- Capability to simulate:
  - Room Impulse Responses (RIRs)
  - Additive noise environments
  - Hearing-aid algorithms (compression, noise reduction)
  - Objective intelligibility metrics (STOI, PESQ, etc.)

Simulink may also be used for model-based design and simulation of hearing aid pipelines and real-time systems.

2.3.2. Python (with Scientific Libraries)

Python is another powerful tool used for signal processing and machine learning tasks. Popular libraries include:

- NumPy, SciPy – for numerical and signal processing operations
- Librosa – for audio analysis and feature extraction
- Pyroomacoustics – for simulating reverberant environments and RIRs
- Scikit-learn / TensorFlow / PyTorch – for implementing and evaluating machine learning models (if used for noise reduction or intelligibility prediction) Python is ideal for flexible, open-source development and is often used in conjunction with MATLAB for specific tasks or validation.

2.3.3. Audacity / Praat (for Audio Pre/Post-Processing)

- Audacity: Used for basic audio editing tasks such as trimming, mixing, and normalizing audio files.
- Praat: Useful for analyzing phonetic and speech characteristics. Can help verify intelligibility cues like formant structure, pitch, and timing.

2.3.4. IDEs and Development Tools

- MATLAB IDE: The primary environment for code development, testing, and visualization of results in MATLAB.
- Jupyter Notebook / VS Code (for Python): For running Python scripts interactively with real-time visualization.
- Git / GitHub: For version control and collaboration.

2.3.5. Optional: Hearing Aid Simulation Software

If real-time hearing aid simulations are used, tools like:

- OpenMHA (Open Master Hearing Aid) – an open-source platform for real-time hearing-aid algorithm development
- Cochlear implant simulators (for extended applications)
- may be included for more advanced processing.

2.4 Practical Setup

2.4.1. Input Stage – Microphone Array (Left and Right Channels)

- Devices Used: Two calibrated omnidirectional microphones or dummy head microphones (e.g., Brüel & Kjær HATS or custom binaural mic rigs).
- Purpose: To capture real-world binaural recordings from two spatially separated inputs (left and right ears).
- Alternative: If real recording is not possible, use Head-Related Impulse Responses (HRIRs) to spatialize clean speech for simulation.

2.4.2. Pre-processing Module – Noise and Reverberation Simulation

- Software: MATLAB or Python (with Pyroomacoustics)
- Noise Types: Multi-talker babble, cafeteria noise, traffic noise
- Reverberation: Simulated using Room Impulse Responses (RIRs)
  - Tools: rir_generator in MATLAB or pyroomacoustics in Python
- Goal: Replicate real-world noisy and reverberant environments for testing intelligibility

2.4.3. Hearing Aid and Signal Processing Unit

- Software Tools:
  - MATLAB for implementing hearing-aid algorithms
  - Python for testing alternative ML-based denoising algorithms
- Key Algorithms:
  - Noise reduction (Wiener filter, spectral subtraction)
  - Dynamic Range Compression (DRC)
  - Directional microphone simulation (beamforming)
- Optional: Use OpenMHA (Open Master Hearing Aid) for real-time hearing-aid signal chain simulation

2.4.4. Binaural Processing Unit

- Goal: Maintain and utilize interaural cues (ITDs, ILDs) for spatial awareness and intelligibility
- Processing Steps:
  - Binaural noise reduction
  - Cross-ear coordination (coordinated gain and filtering)
- Tools: MATLAB with Audio Toolbox or Python using spatial audio libraries

2.4.5. Output Stage – Delivery to Listener

- Devices: High-quality headphones (e.g., Sennheiser HD 650) or simulated hearing aids
- Calibration: Adjust for equal loudness and level matching between left and right channels
- Optional: Include headphone compensation filter if using a dummy head

2.4.6. Evaluation Setup

- Objective Measures:
  - STOI (Short-Time Objective Intelligibility)
  - SNR Improvement
  - PESQ (Perceptual Evaluation of Speech Quality)
- Subjective Measures:
  - Listening tests with normal-hearing or hearing-impaired participants
  - Sentence-in-noise recognition tasks (e.g., HINT, QuickSIN)
- Environment:
  - Quiet lab room or anechoic chamber for controlled testing
  - Use of GUI (MATLAB App Designer or Python GUI) for test management and user input

3. Implementation

Inputs:

1. Inputs

A. Speech Signal

- **Clean speech from a speech corpus (e.g., TIMIT, IEEE sentences).**
- **Sample rate: 16 kHz or 44.1 kHz.**

B. Noise Signals

- **Types: White noise, babble noise, cafeteria noise, traffic noise, etc.**
- **Spatial configuration: Same direction as speech or different azimuths.**

C. Room Impulse Responses (RIRs)

- **For simulating reverberation (use tools like Roomsim or pyroomacoustics).**
- **Binaural RIRs for both ears (HRIRs can be used for spatialization).**
- **Reverberation times (RT60): e.g., 0.3s, 0.6s, 0.9s.**

D. Hearing-Aid Signal Processing Algorithms

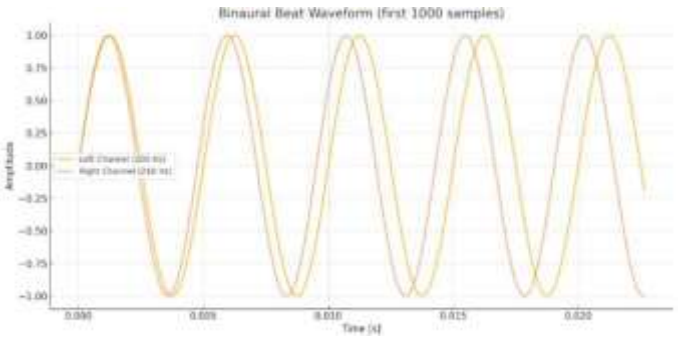
- **Algorithms:**
  - **Dynamic range compression**
  - **Directional microphones (beamforming)**
  - **Noise reduction (spectral subtraction, Wiener filtering)**
  - **Binaural coherence-based post-filtering**
- **Optional: Include real hearing-aid simulations (e.g., openMHA)**

E. Listener Model or Evaluation Metric

- **Objective intelligibility models:**
  - **BSIM (Binaural Speech Intelligibility Model)**
  - **SII (Speech Intelligibility Index)**
  - **STOI (Short-Time Objective Intelligibility)**
  - **MBSTOI (Modified Binaural STOI)**
  - **HASPI (Hearing Aid Speech Perception Index)**
- **Subjective listening tests (optional)**

Generate a Simple Binaural Beat

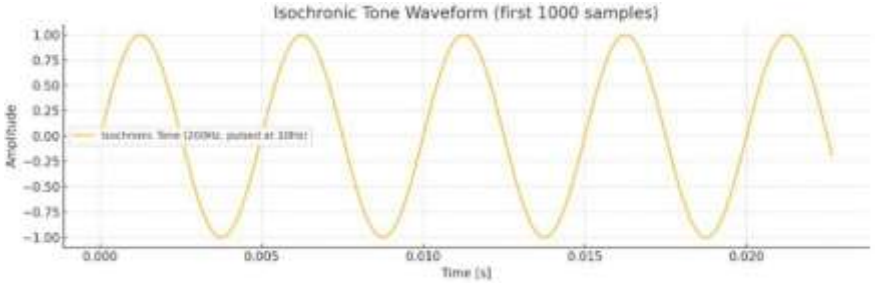
```python
import numpy as np
import scipy.io.wavfile as wavfile

def generate_binaural_beat(base_frequency, beat_frequency, duration, sample_rate=44100
    t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)
    left = np.sin(2 * np.pi * base_frequency * t)
    right = np.sin(2 * np.pi * (base_frequency + beat_frequency) * t)
    return left, right

# Generate a 10Hz binaural beat at 200Hz base frequency for 5 seconds
left, right = generate_binaural_beat(
    base_frequency=200,
    beat_frequency=10,
    duration=5
)

# Stack and normalize
stereo = np.vstack((left, right)).T.astype(np.float32)

# Save as WAV file
wavfile.write('binaural_beat.wav', 44100, stereo)
```



IsochronicTone:

```
[10] # Import necessary libraries
     import numpy as np
     import scipy.io.wavfile as wavfile
     from IPython.display import Audio

     # Function to generate isochronic tone
     def generate_isochronic_tone(frequency, pulse_rate, duration, sample_rate=44100):
         t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)

         # Sine wave at the given frequency
         tone = np.sin(2 * np.pi * frequency * t)

         # Square wave to modulate the tone at the pulse rate (on/off)
         pulse = 0.5 * (1 + np.sign(np.sin(2 * np.pi * pulse_rate * t)))  # values: 0 or 1

         # Multiply the tone with the pulse to turn it on/off
         modulated_tone = tone * pulse

         # Normalize to float32 range
         modulated_tone = modulated_tone.astype(np.float32)

         return modulated_tone

     # Parameters
     frequency = 200      # Hz
     pulse_rate = 10      # Hz (isochronic pulse rate)
     duration = 5         # seconds
     sample_rate = 44100  # Hz

     # Generate isochronic tone
```

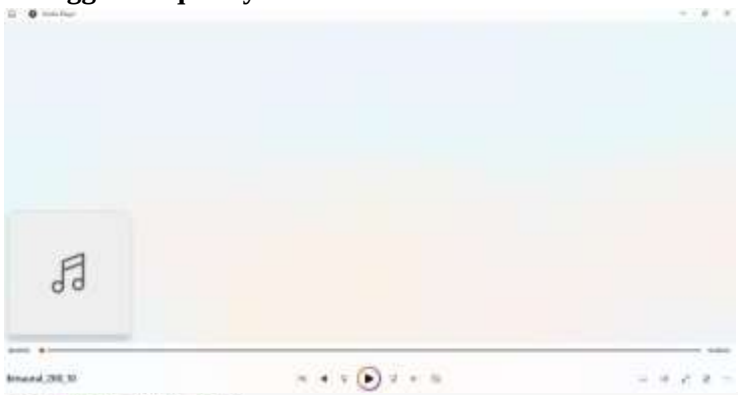
```
         # Normalize to float32 range
         modulated_tone = modulated_tone.astype(np.float32)

         return modulated_tone

     # Parameters
     frequency = 200      # Hz
     pulse_rate = 10      # Hz (isochronic pulse rate)
     duration = 5         # seconds
     sample_rate = 44100  # Hz

     # Generate isochronic tone
     tone = generate_isochronic_tone(frequency, pulse_rate, duration, sample_rate)

     # Save as WAV file
     wavfile.write('isochronic_tone.wav', sample_rate, tone)

     # Play the audio in Colab
     Audio('isochronic_tone.wav')
```



Monaural Beat:

```python
from IPython.display import Audio

# Function to generate a monaural beat
def generate_monaural_beat(frequency1, frequency2, duration, sample_rate=44100):
    t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)

    # Two sine waves at different frequencies
    tone1 = np.sin(2 * np.pi * frequency1 * t)
    tone2 = np.sin(2 * np.pi * frequency2 * t)

    # Monaural beat: mix them into one channel (not stereo)
    beat = (tone1 + tone2) / 2

    # Normalize
    beat = beat.astype(np.float32)
    return beat

# Parameters
frequency1 = 200    # Hz
frequency2 = 210    # Hz
duration = 5        # seconds
sample_rate = 44100 # Hz

# Generate the monaural beat
beat = generate_monaural_beat(frequency1, frequency2, duration, sample_rate)

# Save to WAV
wavfile.write('monaural_beat.wav', sample_rate, beat)

# Play the audio in Colab
Audio('monaural_beat.wav')
```

Solfeggio Frequency:

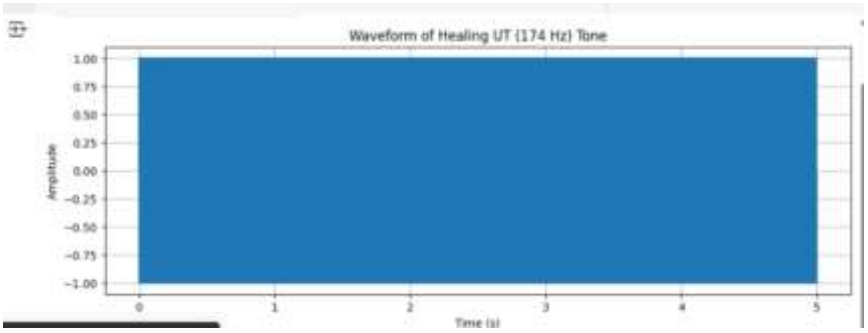


```
import numpy as np
import scipy.io.wavfile as wavfile
from IPython.display import Audio

# Define Solfeggio frequencies (in Hz)
SolfeggioFrequency = {
    "UT": 174.0,
    "RE": 285.0,
    "MI": 396.0,
    "FA": 417.0,
    "SOL": 528.0,
    "LA": 639.0,
    "SI": 741.0,
    "DO": 852.0,
    "TI": 963.0
}

def generate_solfeggio_tone(frequency, duration, sample_rate=44100, amplitude=1.0):
    t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)
    tone = amplitude * np.sin(2 * np.pi * frequency * t)
    return tone.astype(np.float32)

def mix_solfeggio_frequencies(frequencies, amplitudes, duration, sample_rate=44100):
    mixed = np.zeros(int(sample_rate * duration), dtype=np.float32)
    for freq, amp in zip(frequencies, amplitudes):
        mixed += generate_solfeggio_tone(freq, duration, sample_rate, amp)
    # Normalize to avoid clipping
```

```
    # Normalize to avoid clipping
    mixed = mixed / np.max(np.abs(mixed))
    return mixed

def generate_solfeggio_binaural(base_freq, beat_frequency, duration, sample_rate=4410€
    t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)
    left = np.sin(2 * np.pi * base_freq * t)
    right = np.sin(2 * np.pi * (base_freq + beat_frequency) * t)
    stereo = np.vstack((left, right)).T.astype(np.float32)
    return stereo

# 1. Generate a single UT tone (174 Hz)
healing_tone = generate_solfeggio_tone(SolfeggioFrequency["UT"], duration=5)

# Save and play healing tone
wavfile.write("healing_ut_174hz.wav", 44100, healing_tone)
print("✅ Saved: healing_ut_174hz.wav")
display(Audio("healing_ut_174hz.wav"))

# 2. Mix UT (174 Hz) and SOL (528 Hz) with equal amplitudes
mixed = mix_solfeggio_frequencies(
    frequencies=[SolfeggioFrequency["UT"], SolfeggioFrequency["SOL"]],
    amplitudes=[0.5, 0.5],
    duration=5
)

wavfile.write("mixed_ut_sol.wav", 44100, mixed)
print("✅ Saved: mixed_ut_sol.wav")
display(Audio("mixed_ut_sol.wav"))

# 3. Generate a binaural beat using SOL (528 Hz) with 7.83 Hz difference
```

```
wavfile.write("mixed_ut_sol.wav", 44100, mixed)
print("✅ Saved: mixed_ut_sol.wav")
display(Audio("mixed_ut_sol.wav"))

# 3. Generate a binaural beat using SOL (528 Hz) with 7.83 Hz difference
stereo_binaural = generate_solfeggio_binaural(
    base_freq=SolfeggioFrequency["SOL"],
    beat_frequency=7.83,
    duration=5
)

wavfile.write("binaural_sol_528_7.83.wav", 44100, stereo_binaural)
print("✅ Saved: binaural_sol_528_7.83.wav")
display(Audio("binaural_sol_528_7.83.wav"))
```

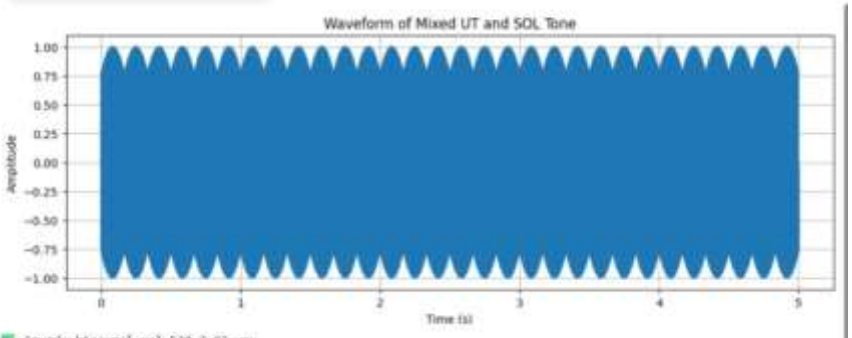
```
✅ Saved: healing_ut_174hz.wav
▶ 0:00 / 0:05
✅ Saved: mixed_ut_sol.wav
▶ 0:00 / 0:05
✅ Saved: binaural_sol_528_7.83.wav
▶ 0:00 / 0:05
```



Time (s)

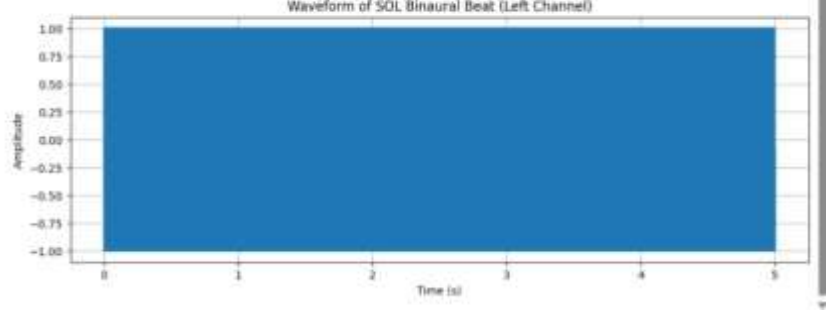
✅ Saved: mixed_ut_sol.wav

▶ 0:00 / 0:05



✅ Saved: binaural_sol_528_7.83.wav

▶ 0:00 / 0:05



Binaural Beat with carrier frequency:

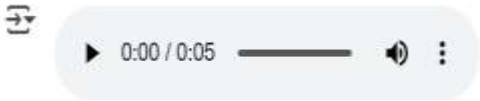
```python
import numpy as np
import scipy.io.wavfile as wavfile
from IPython.display import Audio, display

def generate_binaural_beat(
    base_frequency,
    beat_frequency,
    duration,
    sample_rate=44100,
    carrier_frequency=None,
    attack=0.0,
    decay=0.0
):
    t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)

    left_freq = base_frequency
    right_freq = base_frequency + beat_frequency

    # Generate sine waves
    left = np.sin(2 * np.pi * left_freq * t)
    right = np.sin(2 * np.pi * right_freq * t)

    # Optional amplitude modulation with a carrier frequency
    if carrier_frequency:
        carrier = np.sin(2 * np.pi * carrier_frequency * t)
        left *= carrier
        right *= carrier

    # Apply fade-in (attack) and fade-out (decay)
    envelope = np.ones_like(t)
```

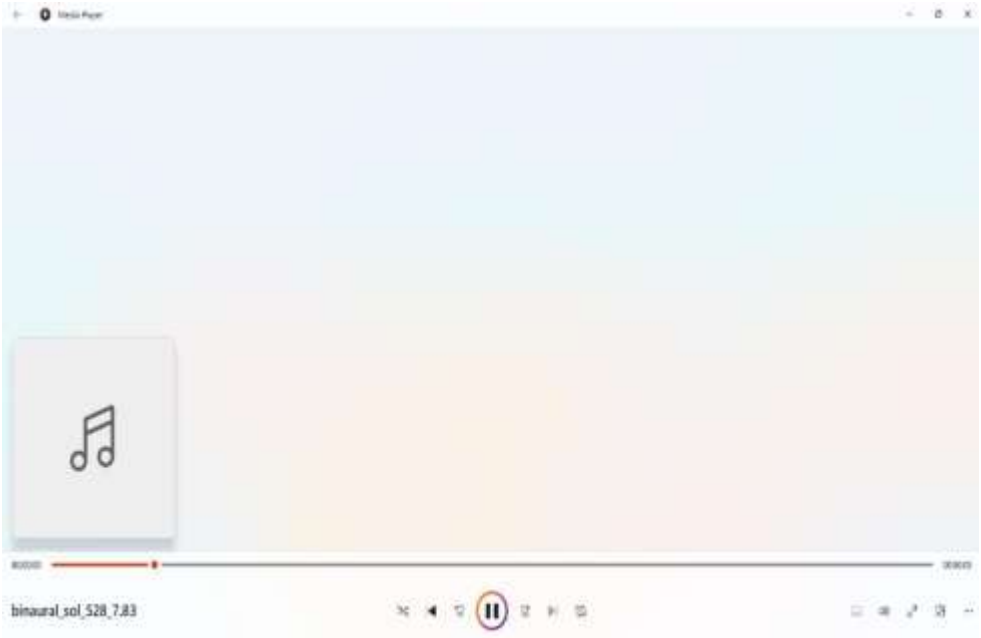
```python
    # Apply fade-in (attack) and fade-out (decay)
    envelope = np.ones_like(t)
    if attack > 0:
        attack_samples = int(sample_rate * attack)
        envelope[:attack_samples] *= np.linspace(0, 1, attack_samples)
    if decay > 0:
        decay_samples = int(sample_rate * decay)
        envelope[-decay_samples:] *= np.linspace(1, 0, decay_samples)

    left *= envelope
    right *= envelope

    stereo = np.vstack((left, right)).T.astype(np.float32)
    return stereo

# === Call the function ===
binaural = generate_binaural_beat(
    base_frequency=200,
    beat_frequency=10,
    duration=5,
    carrier_frequency=432,  # Optional amplitude modulation
    attack=0.5,             # Fade in
    decay=0.5               # Fade out
)

# Save and play
wavfile.write("custom_binaural.wav", 44100, binaural)
display(Audio("custom_binaural.wav"))
```

```python
import numpy as np
import scipy.io.wavfile as wavfile
from IPython.display import Audio, display
from typing import Optional, Tuple

def generate_binaural_beat(
    base_frequency: float,
    beat_frequency: float,
    duration: float,
    sample_rate: int = 44100,
    amplitude: float = 0.5,
    carrier_frequency: Optional[float] = None,
    attack: float = 0.1,
    decay: float = 0.1
) -> Tuple[np.ndarray, np.ndarray]:
    """
    Generate a stereo binaural beat signal.

    Parameters:
        base_frequency (float): Frequency for left ear (Hz).
        beat_frequency (float): Frequency difference (Hz).
        duration (float): Length of signal (seconds).
        sample_rate (int): Sampling rate (samples per second).
        amplitude (float): Volume scaling factor (0 to 1).
        carrier_frequency (Optional[float]): Frequency for amplitude modulation (Hz).
        attack (float): Fade-in time (seconds).
        decay (float): Fade-out time (seconds).

    Returns:
```

```python
    Returns:
        Tuple[np.ndarray, np.ndarray]: Left and right channel signals.
    """
    t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)

    # Core tones
    left = amplitude * np.sin(2 * np.pi * base_frequency * t)
    right = amplitude * np.sin(2 * np.pi * (base_frequency + beat_frequency) * t)

    # Optional carrier modulation
    if carrier_frequency:
        carrier = np.sin(2 * np.pi * carrier_frequency * t)
        left *= carrier
        right *= carrier

    # Apply fade-in and fade-out
    envelope = np.ones_like(t)
    if attack > 0:
        attack_samples = int(sample_rate * attack)
        envelope[:attack_samples] *= np.linspace(0, 1, attack_samples)
    if decay > 0:
        decay_samples = int(sample_rate * decay)
        envelope[-decay_samples:] *= np.linspace(1, 0, decay_samples)

    left *= envelope
    right *= envelope

    return left.astype(np.float32), right.astype(np.float32)
```

```
# === Use the function ===
left, right = generate_binaural_beat(
    base_frequency=200,
    beat_frequency=10,
    duration=5,
    amplitude=0.5,
    carrier_frequency=432,
    attack=0.5,
    decay=0.5
)

# Combine into stereo and save
stereo = np.vstack((left, right)).T
wavfile.write("binaural_beat_with_fade.wav", 44100, stereo)
display(Audio("binaural_beat_with_fade.wav"))
```





Outputs:

- Speech intelligibility scores per condition (e.g., STOI, SII, HASPI)
- Effect of:
  - Different SNRs (e.g., -5 to +10 dB)
  - Reverberation levels (RT60)
  - Hearing-aid processing configurations
- Graphs: Intelligibility vs. SNR, RT60, etc

4 ACKNOWLEDGEMENT :

The author sincerely acknowledges the invaluable guidance, continuous support, and constructive feedback provided by Dr. S. China Venkateswarlu Dr . V. Siva Nagaraju faculty and Ms P.Ganga Bhavani and members of the Department of Electronics and Communication Engineering at the Institute of Aeronautical Engineering (IARE). Their expert advice and encouragement have been instrumental throughout the entire course of this research.

Special thanks are also extended to the faculty and staff of the Institute for providing a conducive academic environment and essential resources that greatly facilitated the successful completion of this work. The author appreciates the support and collaboration of peers and colleagues who contributed their time and expertise

REFERENCES:

- **General Binaural Speech Intelligibility**

1. **Bronkhorst,A.W.(2000).** ***The cocktail party phenomenon: A review of research on speech intelligibility in multiple-talker conditions. Acta Acustica united with Acustica, 86(1), 117–128.*** **[Classic review on binaural unmasking, spatial release from masking, and auditory scene analysis.]**
2. **Best, V., Marrone, N., Mason, C. R., & Kidd, G. Jr. (2008).**

The influence of non-spatial factors on measures of spatial release from masking.Journal of the Acoustical Society of America, 124(5), 3100–3111.

3. **Culling,J.F.,&Lavandier,M.(2021).** ***Speech intelligibility in rooms and the effects of binaural and spatial hearing.***
Nature Reviews Psychology, 1(8), 426–439. **[Excellent recent overview tying together binaural perception and room acoustics.]**

- **Reverberation and Speech Perception**

4. **Nábělek, A. K., & Pickett,J.M.(1974).** ***Monaural and binaural speech perception through hearing aids under reverberant and nonreverberant conditions. Journal of Speech and Hearing Research, 17(4), 724–739.***
5. **Warzybok, A., Rennies, J., Brand, T., Doclo, S., & Kollmeier, B. (2013).** ***Effects of spatial and temporal integration of a single early reflection on speech intelligibility.***
Journal of the Acoustical Society of America, 133(1), 269–282.

- **Noise**

6. **Brungart,D.S.(2001).**

Informational and energetic masking effects in the perception of two simultaneous talkers.
Journal of the Acoustical Society of America, 109(3), 1101–1109.

7. **Freyman, R.L.,Balakrishnan,U.,&Helfer,K.S.(2004).**

Effect of number of masking talkers and auditory priming on informational masking in speech recognition. Journal of the Acoustical Society of America, 115(5), 2246–2256.

- **Hearing Aids and Signal Processing**

8. **Neher, T., Grimm, G., Hohmann, V., & Kohlmeier, B. (2014).** ***Do hearing aid users exploit binaural cues in spatially complex environments?***
Ear and Hearing, 35(3), e52–e62.

9. **Doclo, S., Spriet, A., & Moonen, M. (2010).** ***Binaural signal processing for hearing aids: Opportunities and challenges.***
Proceedings of the International Symposium on Communications Control and Signal Processing (ISCCSP).

10. **Desloge, J. G., Rabinowitz, W. M., Zurek, P. M., & Delhorne, L. A. (1997).** ***Microphone-array hearing aids with binaural output. II. A two- microphone adaptive system. IEEE Transactions on Speech and Audio Processing, 5(6), 543–551.***

◆ **Models and Objective Measures**

11. **Beutelmann, R., Brand, T., & Kollmeier, B. (2010).** ***Revision, extension, and evaluation of a binaural speech intelligibility model.***
Journal of the Acoustical Society of America, 127(4), 2479–2497. **[Binaural SI model (BSIM) integrating binaural masking and room acoustics.]**

12. **Jelfs,S.,Lavandier, M.,&Culling,J.F. (2011).*****Revision and validation of a binaural model for speech intelligibility in noise.***

Hearing Research, 275(1–2), 96–104.

13. **Rennies, J., Schepker, H., Warzybok, A., & Kollmeier, B. (2016).** ***Modeling the effects of background noise and room reverberation on speech intelligibility.***
Journal of the Acoustical Society of America, 140(5), 2915–2928.