Digital Information Processing Lab 2025

Class 5: DTMF Tone Generation and Detection

Group 8 : Vision Matrix

23.06.2025

Name	Mat. Nr.	Email
Navya Sajeev Warrier	252782	navya.sajeev@st.ovgu.de
Sanjay Pulparambil Girish	249360	sanjay.pulparambil@st.ovgu.de

Double-click (or enter) to edit

Preparatory Task

Task 1: Introduction to DTMF Tones

1. Explain the concept of DTMF (Dual-Tone Multi-Frequency) tones and their use in telecommunication systems.

   DTMF works by generating a combination of two specific audio frequencies (tones) for each key pressed on a telephone keypad. The system uses a set of 8 frequencies: 4 low-frequency tones and 4 high-frequency tones. These are arranged in a 4x4 matrix to represent digits (0–9), characters (*, #), and four additional control keys (A–D, used mostly in military or proprietary systems). [1]

   The standard DTMF frequency table is:

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Each key on the keypad produces a tone that is the sum of one frequency from the low group and one from the high group. For example:

- Pressing ‘1’ generates tones at 697 Hz and 1209 Hz.
- Pressing ‘0’ generates tones at 941 Hz and 1336 Hz.

These tones are within the range of human speech (300 Hz to 3400 Hz), which means they can be transmitted over standard voice-grade telephone lines without requiring special equipment.

Technical Working

When a key is pressed:

1. Two oscillators generate sine waves at the corresponding low and high frequencies.
2. These signals are summed together and transmitted as a single tone over the line.
3. At the receiving end, a DTMF decoder uses a digital signal processor or filter bank to detect and identify the two frequency components.
4. The combination is then mapped back to the corresponding key or command.

Applications in Telecommunication :

1. **Telephone Dialing:** DTMF tones allow users to dial phone numbers by simply pressing buttons. This input is sent directly to telephone exchanges to establish the call.
2. **Interactive Voice Response (IVR):** DTMF is used in automated systems where users navigate menus by pressing keys (e.g., “Press 1 for English, Press 2 for Support”).
3. **Remote Control:** Systems such as voicemail, security gates, or radio repeaters can be remotely operated via DTMF tones.
4. **Mobile Communications:** In early mobile and analog systems, DTMF was used for control and navigation.
5. **Telebanking & Payments:** DTMF enables secure entry of PINs and account numbers during financial transactions over the phone.

2. Discuss the typical frequency ranges and combinations associated with each digit in DTMF.

Frequency Groups in DTMF DTMF uses a matrix of 4 low frequencies and 4 high frequencies, totaling 8 distinct frequencies. These frequencies are standardized to minimize distortion, prevent interference with human speech, and allow accurate decoding. [2] [3]

Low-frequency group (rows):

697 Hz

770 Hz

852 Hz

941 Hz

High-frequency group (columns):

1209 Hz

1336 Hz

1477 Hz

1633 Hz

When a user presses a key, the telephone or device generates two tones: one selected from the low group and one from the high group. These are added together (not modulated), forming the unique DTMF signal for that key.

3. Describe the advantages and applications of DTMF tones in phone systems, automated menus, and touch-tone dialing.

Advantages of DTMF Tones

1. **Speed and Efficiency:** Unlike pulse dialing, where numbers are transmitted through a series of electrical pulses (one for each digit), DTMF tones transmit information instantly with the press of a key. This significantly reduces the time it takes to place a call or input data.
2. **Reliability and Accuracy:** Each DTMF signal is uniquely identified by a specific combination of frequencies, minimizing errors during transmission. The system is highly resistant to misinterpretation by the telephone exchange equipment, making it more reliable than earlier analog methods.
3. **Ease of Use:** DTMF-enabled systems are intuitive and user-friendly. Users can interact with them simply by pressing buttons on a keypad, even from mobile or landline phones.
4. **Supports Automation:** DTMF tones can be interpreted by computerized systems, making them ideal for automating interactions without the need for human operators. This enables scalable, efficient customer service operations.
5. **Cost-Effectiveness:** Since DTMF operates over standard voice lines and doesn't require additional infrastructure, it is a low-cost solution for both service providers and end users.
6. **Compatibility:** DTMF is widely supported across telecommunication systems and works over both analog and digital networks, making it a universally accepted standard.

Applications of DTMF Tones

1. **Touch-Tone Dialing:** The most basic and widespread use of DTMF is for dialing telephone numbers. When a user presses a key, a unique DTMF tone is generated and sent to the telephone exchange, which interprets it to route the call.
2. **Interactive Voice Response (IVR) Systems:** DTMF tones are the backbone of IVR systems used in customer service. Callers can navigate menus, enter account numbers, or respond to prompts without speaking, simply by pressing keys. This allows businesses to provide 24/7 service without staffing call centers continuously.
3. **Remote Control of Systems:** DTMF tones can be used to control remote devices over a phone line. For example, amateur radio operators use DTMF to activate repeaters or control equipment. Security systems may also use DTMF codes to arm or disarm alarms remotely.

4. **Banking and Financial Services:** Telephone banking often utilizes DTMF tones for user authentication and transaction processing. Customers can check balances, transfer funds, or pay bills using keypad inputs.
5. **Voicemail and Teleconferencing Systems:** Users can navigate voicemail systems or manage conference calls using DTMF inputs to skip messages, delete recordings, or mute participants. [2]

4. Explore the challenges and considerations in generating and detecting DTMF tones accurately.

Challenges and Considerations in Generating and Detecting DTMF Tones Accurately :

Dual-Tone Multi-Frequency (DTMF) tones are widely used in telecommunication systems for signaling user input through the telephone keypad. Each key press generates a unique combination of two audio frequencies, allowing quick and precise communication with automated systems. While DTMF technology is reliable and time-tested, accurately generating and detecting these tones comes with a number of challenges and considerations, especially in modern digital and hybrid communication environments.

1. Frequency Precision and Signal Integrity Accurate DTMF generation relies on producing two distinct frequencies per key press with strict tolerances—typically within ±1.5% of the specified values. Any deviation due to hardware imperfections, poor oscillator stability, or software miscalculations can result in incorrect tone generation. On the detection side, receivers must distinguish these tones accurately, even in the presence of slight distortions or variations. Therefore, signal generators and decoders must be precisely calibrated and meet the specifications outlined in standards like ITU-T Q.23.
2. Noise and Line Distortion DTMF tones can be affected by background noise, signal degradation, or distortion during transmission. Analog telephone lines, mobile networks, and even VoIP systems introduce varying degrees of noise and compression artifacts, which can interfere with tone recognition. For instance, echoes or crosstalk may distort the signal, leading to detection errors. To address this, detection algorithms must be robust enough to filter out unwanted noise and isolate the correct frequency pairs.
3. Timing Requirements Each DTMF tone must be transmitted for a minimum duration (typically 40 to 70 milliseconds), with an appropriate pause between digits. If tones are too short or closely spaced, the decoder may fail to register them. On fast keypads or **automated** systems, improper timing can result in missed or duplicate entries. Implementing proper timing controls and adhering to recommended tone durations and inter-digit spacing is essential for reliability.
4. Voice Compression Issues In modern networks, especially those using Voice over IP (VoIP), audio compression algorithms such as G.729 or G.723 may degrade DTMF tones or strip them out entirely. These codecs are optimized for human speech and may treat DTMF tones as background noise or tonal artifacts. To overcome this, many systems use

out-of-band DTMF signaling, where tones are encoded and transmitted separately from the voice stream (e.g., using RFC 2833 for RTP packets).

5. Talk-Off and False Detection A major challenge in DTMF detection is "talk-off," where voice or music mimics DTMF tones, triggering false responses. This occurs because speech occasionally contains frequency components similar to DTMF signals. To prevent this, detectors use validation techniques such as twist ratio analysis (comparing the amplitude of the two frequencies), minimum duration checks, and matching both frequencies simultaneously before registering a valid tone. [3] [4]
6. Amplitude and Twist Ratio DTMF tones consist of one low and one high-frequency tone played simultaneously. These must be transmitted at appropriate relative amplitudes, known as the twist ratio. If one frequency dominates significantly, the receiving system may fail to recognize the tone accurately. Ensuring correct amplitude balance during tone generation is critical for detection reliability.

Task 2: Filtering Techniques for DTMF Tone Generation and Detection

1. Discuss the importance of accurate frequency generation and detection in DTMF systems.

   Accurate frequency generation and detection lie at the heart of reliable Dual-Tone Multi-Frequency (DTMF) systems. Each DTMF tone is formed by combining two specific audio frequencies: one from a low-frequency group (697 Hz to 941 Hz) and one from a high-frequency group (1209 Hz to 1633 Hz). These frequencies are uniquely associated with each key on a telephone keypad. When a user presses a button, the system sends the corresponding dual-tone signal over the communication channel. The receiver must then detect and decode this signal correctly to interpret the intended input. Any inaccuracies in generating or detecting these frequencies can compromise the entire system’s reliability.

   1. Frequency Uniqueness and Signal Recognition DTMF relies on the precise pairing of frequencies to identify digits. For example, pressing the “5” key generates tones at 770 Hz and 1336 Hz. If the frequency generation is imprecise—due to component tolerances, signal distortion, or software inaccuracies—the resulting tone may fall outside the acceptable range, leading to failed detection or misinterpretation by the receiving system. Similarly, if detection algorithms are not precise enough to distinguish between valid and distorted tones, it can lead to errors, such as registering the wrong digit or missing an input altogether.
   2. Avoiding Crosstalk and Ambiguity The DTMF frequency plan is designed so that no pair of tones overlaps with another key combination. This non-overlapping nature ensures that each key press is uniquely identifiable. Accurate generation is crucial to maintaining this distinction. Even minor deviations can cause tone overlap or ambiguous signals, especially when transmitted over noisy or lossy channels. The

receiver, in turn, must detect the exact frequencies with sufficient resolution to avoid confusing one key with another.

3. Compliance with Telecommunication Standards International standards, such as ITU-T Recommendation Q.23, define specific tolerances for DTMF frequencies—typically within ±1.5%. Systems that deviate from these standards may become incompatible with global telephony infrastructure. Accurate frequency generation ensures interoperability between devices, networks, and services across various regions and platforms.
4. Resisting Talk-Off and False Positives One of the risks in DTMF systems is "talk-off," where speech or background audio mimics DTMF frequencies, triggering false responses. Accurate detection algorithms must be able to distinguish true DTMF tones from random voice signals. This requires identifying both correct frequencies simultaneously and verifying their duration and amplitude. Inaccurate detection mechanisms increase susceptibility to false positives and degrade user experience, especially in automated services like IVR systems.
5. Critical for Automation and Security In applications like banking, voicemail access, and remote control systems, DTMF inputs are used for authentication and command execution. Any error in tone recognition could lead to incorrect operations or security vulnerabilities. Accurate frequency handling ensures that only valid inputs are recognized and acted upon.

2. Explain how filters can be used to shape the frequency response of a signal for generating and detecting DTMF tones.

   Using Filters to Shape the Frequency Response for DTMF Tone Generation and Detection Dual-Tone Multi-Frequency (DTMF) tones are the familiar sounds you hear when pressing keys on a telephone keypad. Each key generates a combination of two specific frequencies: one from a low-frequency group and one from a high-frequency group. To accurately generate and detect these tones, filters play a crucial role in shaping and analyzing the frequency response of signals.

   Tone Generation with Filters :

   When generating DTMF tones, filters can be used to isolate and shape the desired frequencies. For example, if we want to generate the tone for the key '5', we need to combine a 770 Hz tone (from the low-frequency group) with a 1336 Hz tone (from the high-frequency group). By using bandpass filters centered around these frequencies, we can ensure that only the required sine waves are passed through cleanly, while any unwanted noise or harmonics are suppressed. This results in a clear, well-defined dual-tone signal.

   Tone Detection with Filters

In the detection process, filters help us pick out the specific frequency components present in a received DTMF signal. Bandpass filters are again used—this time in parallel, each tuned to one of the standard DTMF frequencies. When a signal is received, each filter will respond strongly only if its center frequency is present in the signal. This allows us to determine exactly which two frequencies are present, and therefore which key was pressed. The cleaner and sharper the filters, the more reliable the detection.

Why Filtering Matters ?

Without proper filtering, signals can become noisy or distorted, which can lead to false detections or missed tones. Filters help reduce interference, isolate important frequency components, and maintain the clarity of both generated and received tones. In practice, digital filters (like IIR or FIR filters) are often used in software implementations, while analog filters may be used in hardware-based systems.

3. Discuss the potential challenges in filtering out unwanted noise and harmonics while preserving the integrity of DTMF tones.

   Filtering out unwanted noise and harmonics while preserving the integrity of Dual-Tone Multi-Frequency (DTMF) tones presents a range of technical challenges, primarily due to the nature of the signal itself and the environment in which it is transmitted and processed. DTMF tones, used in telephone systems and interactive voice response (IVR) applications, consist of two simultaneous sinusoidal frequencies selected from a predefined set of low and high frequency groups. The goal of any filter in this context is to accurately detect and preserve these two specific tones while rejecting all other frequency components. This task, however, is complicated by several factors. [2] [3]

1. Narrow Frequency Tolerance

DTMF tones are defined by a precise set of eight frequencies (four low and four high), and each tone pair corresponds to a specific keypress. These frequencies are typically spaced apart enough to prevent overlap, but they still lie relatively close to each other within the 697–1633 Hz range. Consequently, filters must have sharp frequency selectivity to distinguish DTMF tones from noise or spurious signals without attenuating the tones themselves. Designing such narrowband filters that maintain minimal phase distortion is technically demanding.

2. Presence of Harmonics

   Each DTMF tone is a sum of two pure sinusoids. However, in practical scenarios, these signals may get distorted during transmission due to non-linearities in the communication channel, such as compression algorithms, amplifiers, or digital codecs (e.g., GSM or VoIP systems). These distortions introduce harmonics—integer multiples of the base DTMF frequencies—that can interfere with tone detection. Filtering out harmonics without also impacting the primary tones, especially since the second harmonics of lower DTMF tones can overlap with higher frequency DTMF tones, requires advanced filtering techniques and possibly adaptive signal processing.

3. Additive Noise

   Ambient noise, line noise, and background speech are common in telephony applications. Noise can mask or distort DTMF tones, particularly if the signal-to-noise ratio (SNR) is low. Conventional low-pass or band-pass filters may not suffice, as DTMF frequencies lie in the voice band and may overlap with the spectrum of human speech. This makes it difficult to distinguish tones from voiced sounds such as vowels or background music. Filters must be designed not only for frequency isolation but also for temporal characteristics to exploit the predictable duration and cadence of DTMF signals.

4. Real-Time Constraints

Many DTMF detection systems, especially those used in IVR or mobile applications, must operate in real-time. This limits the complexity of filtering algorithms, as they must process incoming data streams with minimal delay. Fast Fourier Transform (FFT)-based methods or Goertzel algorithms are commonly used, but they must be implemented efficiently to avoid latency while maintaining detection accuracy in noisy environments.

5. Multipath and Channel Impairments

In mobile networks or IP-based communication, multipath propagation and packet loss can alter the signal's amplitude and phase characteristics. Such impairments may shift the frequency content slightly or cause fading, further complicating the task of filtering and detection. Adaptive filtering or dynamic thresholding may be required to handle these issues, adding complexity to the system. [3]

Double-click (or enter) to edit

Reference

[1] ^ Suvad Selman and Raveendran Paramesran (2007). Comparative analysis of methods used in the design of DTMF tone detectors. 2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications. doi:https://doi.org/10.1109/ictmicc.2007.4448657.

[2] ^ Bagchi, S. and Mitra, S.K. (1999). Dual-Tone Multi-Frequency Signal Decoding. Springer eBooks, pp.173–196. doi:https://doi.org/10.1007/978-1-4615-4925-3_6.

[3] ^Proakis, J. and Manolakis, D. (n.d.). DIGITAL SIGNAL PROCESSING Principles, Algorithms, m l Applications. [online] Available at: https://uvceee.wordpress.com/wp-content/uploads/2016/09/digital_signal_processing_principles_algorithms_and_applications_third_edition.pdf.

[4] ^ Pamuk, N. and Pamuk, Z. (2015). Dual Tone Multi Frequency (DTMF) signal generation and detection using MATLAB software. 2015 UBT International Conference. doi:https://doi.org/10.33107/ubt-ic.2015.97.

Double-click (or enter) to edit

Class 4: DTMF Tone Generation and Detection

Task 1: DTMF Tone Generation

In this exercise, you will implement a DTMF tone generator function that can generate a DTMF tone sequence from a given sequence of digits.

1. Explain the `generate_dtmf_tone_sequence` function that takes a sequence of digits, frequency table, duration, and pause duration as inputs and generates a corresponding DTMF tone sequence.

   The generate_dtmf_tone_sequence() function takes:

- A sequence of digits (like "123#"),
- A DTMF frequency table,
- A tone duration (how long each tone lasts),
- A pause duration (gap between tones),

and returns a concatenated waveform (audio signal) representing the complete DTMF tone sequence for that input.

2. Generate a wave file for the sequence "003916751040" using the `dtmf_frequencies` table, with a duration of 0.1s and pause duration of 0.05s and display the encoded information.
3. Generate a wave file of the song corresponding to your group number using `note_frequencies` (see list and help below). Listen/Identify the song and write down the name.

Task 2: DTMF Tone Detection

In this task, you will implement a DTMF tone detection function that can decode a given DTMF tone sequence and extract the corresponding digits.

1. Explain the `decode_dtmf_tone_sequence` function that takes a tone sequence, frequency table, duration, and pause duration as inputs and decodes the sequence of digits.
2. Decode the phone number from the provided wave file "numbers.wav" using the `dtmf_frequencies` table, with a duration of 0.0X seconds for each tone and a pause duration of 0.0X seconds. (find out the durations)
3. Test the `generate_dtmf_tone_sequence` and `decode_dtmf_tone_sequence` functions with the self-generated wave files from Task 1 and compare the decoded digits

to the original sequences. How does duration and pause times effect the quality of decoding ?

Optional: Decode the message from the wave file "secret_message.wav" and send an answer as an SMS to the decoded phone number to pass this class instantly without the need for a protocol submission.

```
import numpy as np
import soundfile as sf
from scipy.signal import find_peaks

# global parameters
sample_rate = 44100  # Hz

dtmf_frequencies = {
    '1': [697, 1209],
    '2': [697, 1336],
    '3': [697, 1477],
    'A': [697, 1633],
    '4': [770, 1209],
    '5': [770, 1336],
    '6': [770, 1477],
    'B': [770, 1633],
    '7': [852, 1209],
    '8': [852, 1336],
    '9': [852, 1477],
    'C': [852, 1633],
    '0': [941, 1336],
    'D': [941, 1633]
}


def generate_dtmf_tone(frequencies, duration):
    t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False)
    tone = np.sin(2 * np.pi * frequencies[0] * t) + np.sin(2 * np.pi * frequencie
    return tone


def generate_tone_sequence(sequence, frequency_table, duration, pause_duration):
    tone_sequence = np.array([])
    for digit in sequence:
        if digit in frequency_table:
            frequencies = frequency_table[digit]
            dtmf_tone = generate_dtmf_tone(frequencies, duration)
            tone_sequence = np.concatenate((tone_sequence, dtmf_tone))
            pause = np.zeros(int(pause_duration * sample_rate))
            tone_sequence = np.concatenate((tone_sequence, pause))
    return tone_sequence



def decode_tone_sequence(tone_sequence, segment_duration, pause_duration, dtmf_fr
    segment_length = int(sample_rate * segment_duration)
    pause_length = int(sample_rate * pause_duration)
```

```
    num_segments = len(tone_sequence) // (segment_length + pause_length)

    result = []

    for i in range(num_segments):
        start = i * (segment_length + pause_length)
        segment = tone_sequence[start:start + segment_length]

        spectrum = np.abs(np.fft.fft(segment))
        frequencies = np.fft.fftfreq(segment_length, 1 / sample_rate)
        sorted_indices = np.argsort(spectrum)[::-1]
        top_frequencies = np.round(frequencies[sorted_indices][:4]).astype(int)

        # Remove negative frequencies
        top_frequencies = top_frequencies[top_frequencies >= 0]
        result.append(sorted(top_frequencies))

    return result

def match_frequencies(dtmf_frequencies, detected_frequencies):
    detected_digits = []
    for detected in detected_frequencies:
        min_distance = float('inf')
        closest_digit = None
        for digit, freq in dtmf_frequencies.items():
            distance = np.sqrt((freq[0] - detected[0])**2 + (freq[1] - detected[1
            if distance < min_distance:
                min_distance = distance
                closest_digit = digit

        deviation = (min_distance / np.sqrt((detected[0])**2 + (detected[1])**2))
        print(f"Detected digit: {closest_digit} with deviation: {deviation}%")
        detected_digits.append(closest_digit)

    return detected_digits
```

Songs

Here Are a few songs coded with `note_frequencies` DTMF table. Each group gets a song corresponding to the group number.

```
note_frequencies = {
    'q': [261.63, 392.00],
    'w': [293.66, 440.00],
    'e': [329.63, 493.88],
    'a': [349.23, 523.25],
    's': [392.00, 587.33],
    'd': [440.00, 659.25],
    'f': [493.88, 739.99],
    '1': [523.25, 783.99],
    '2': [587.33, 880.00],
```

```
    '3': [659.25, 987.77],
    'r': [698.46, 1046.50],
    't': [783.99, 1174.66],
    'y': [880.00, 1318.51],
    'u': [987.77, 1480.00],
    '4': [1046.50, 1567.98],
    'x': [0, 0]
}
```

```
song0 = 'qxqxqwexeweasx11sseeqqsaewq'
song1 = 'eeexeeexesqwexaaaaaeeeewwewxs'
song2 = 'ssds1fxssds21xsst31fdrr3121'
song3 = 'ewqweeexwwwxessxewqweeeewwewq'
song4 = 'ssweewxffddsxwsssweewxffddsx'
song5 = 'qweqqweqeasxeas'
song6 = 'qqssddsxaaeewwq'
song7 = 'eeassaewqqweewwxeeassaewqqwewqq'
```

The chosen songs are well known world wide. If you can't recocnise the song here is a list of possible solutions.

List of Songs: London Bridge is Falling Down, Twinkle Twinkle Little Star, Are You Sleeping? (Frère Jacques), Head Shoulders Knees and Toes, Row Row Row Your Boat, She'll Be Coming 'Round the Mountain, Humpty Dumpty, Jingle Bells, Old MacDonald Had a Farm, Mary Had a Little Lamb, Baa Baa Black Sheep, The Alphabet Song, If You're Happy and You Know It, This Old Man, Itsy Bitsy Spider, The Wheels on the Bus, Happy Birthday, Ode to Joy, Yankee Doodle, Five Little Ducks.

Example use

```
from IPython.display import Audio, display
from scipy.io import wavfile

def play(tone, rate=44100): display(Audio(tone, rate=rate))
```

Encoding

```
sequence = "123ABC"
duration = 0.1
pause_duration = 0.5

tone_sequence = generate_tone_sequence(sequence, dtmf_frequencies, duration, paus
play(tone_sequence)
sf.write("dtmf_sequence.wav", tone_sequence, sample_rate)
```

0:00 / 0:03

Decoding

```
detected_frequencies = decode_tone_sequence(tone_sequence, duration, pause_durati
print(detected_frequencies)
decoded_segments = match_frequencies(dtmf_frequencies, detected_frequencies)
print(decoded_segments)
```

```
[[700, 1210], [700, 1340], [700, 1480], [700, 1630], [770, 1630], [850, 1630]
Detected digit: 1 with deviation: 0.22621771308345115%
Detected digit: 2 with deviation: 0.3307271821659828%
Detected digit: 3 with deviation: 0.25914118950121584%
Detected digit: A with deviation: 0.23916343442975432%
Detected digit: B with deviation: 0.16641517941340644%
Detected digit: C with deviation: 0.19613354421355128%
['1', '2', '3', 'A', 'B', 'C']
```