|  |  |
| --- | --- |
|  | Al-Nahrain University  College of Information Engineering  Information and Communications Engineering Department |

**Classification of Digital Modulation Signals Based on AI**

A Final Year Project

Submitted to the College of Information Engineering / Information and Communications Engineering Department in Partial Fulfillment of the Requirements for the Degree of

**Bachelor of Science**

**IN**

**Information and Communications Engineering**

By

**Ali Faisal Ghazi**

Supervised By

**Dr. Asmaa Hameed Majeed**

|  |  |
| --- | --- |
| **Shawal** | **1446** |
| **April** | **2025** |

**Certificate**

We certify, as an examining committee, that we have read this final year project entitled “**Classification of Digital Modulation Signals Based on AI**”, examined the student **Ali Faisal Ghazi** in its content and found it meets the standard of a final year project report for the degree of Bachelor of Science in Information and Communications Engineering.

|  |  |
| --- | --- |
| Signature:  Name:  Date: / /2025  (Member) | Signature:  Name:  Date: / /2025  (Member) |
|  |  |
| Signature:  Name:  Date: / /2025  (Chairman) | Signature:  Name:  Date: / /2025  (Head of Department) |
|  |  |

**Abstract**

Automatic classification of digital modulation signals plays a key role in modern communication systems, especially in areas like spectrum monitoring, cognitive radio, and military signal intelligence. Traditional approaches often rely on manually engineered features and multiple demodulators or measurement tools like spectrum analyzers, making the process inefficient and less scalable. In this project, an AI-based approach is proposed for the automatic classification of common digital modulation schemes.

The system uses spectrograms—time-frequency visual representations of the signals—as input for the classification process. By converting raw modulated signals into spectrogram images, the model is able to capture both temporal and frequency domain features, which improves its ability to distinguish between modulation types. These spectrograms are then fed into a machine learning model trained using supervised learning techniques on a dataset that includes various modulation types such as BPSK, QPSK, and QAM, under different Signal-to-Noise Ratio (SNR) conditions.

The performance of different AI models is evaluated, and classification accuracy is measured across multiple noise levels to simulate real-world environments. The results show that the proposed method achieves high accuracy in identifying modulation types, demonstrating its potential to replace traditional classification systems with a more flexible, efficient, and scalable solution.

**List of Contents**

|  |  |
| --- | --- |
| **Abstract** | i |
| **List of Contents** | ii |
| **List of Symbols and Abbreviations** | iv |
| **List of Tables** | v |
| **List of Figures** | vi |
| **Chapter 1: Introduction** | **1** |
| * 1. Background   2. Modulation Terminology | 1  1 |
| 1.2.1 Amplitude Shift Keying (ASK) | 2 |
| 1.2.2 Phase Shift Keying (PSK) | 3 |
| 1.2.3 Frequency Shift Keying (FSK) | 4 |
| 1.2.4 Quadrature Amplitude Modulation (QAM) | 5 |
| 1.2.5 Chirp Modulation | 5 |

1.3 Introduction to machine learning in signal processing 6

1.4 History 7

1.5 Brief Description of the Work 8

1.6 Aim 9

1.7 Report Layout 9

**Chapter 2: Literature Review 10**

2.1 Introduction 10

2.2 Traditional Methods of Modulation Classification 10

2.3 AI and Machine Learning in Signal Processing 10

2.4 Spectrogram-Based Classification 11

2.5 Gaps in Existing Work 12

**Chapter 3: Methodology**   **13**

3.1 Overview of the Approach 13

3.2 Simulation of Modulation Schemes 13

3.2.1 Amplitude Shift Keying (ASK) 14

3.2.2 Frequency Shift Keying (FSK) 18

3.2.3 Phase Shift Keying (PSK) 20

3.2.4 Quadrature Amplitude Modulation (QAM) 23

3.2.5 Chirp Spread Spectrum (CSS) 25

3.3 Dataset Generation 28

3.3.1 Modulation and Parameter Configuration 28

3.3.2 Signal Generation and Modulation 28

3.3.3 Spectrogram Conversion 29

3.3.4 Visual Characteristics of Spectrograms 29

3.3.5 Labeling and Storage 30

3.4 Data Preprocessing 31

3.4.1 Loading and Formatting 31

3.4.2 Data Splitting 31

3.4.3 Normalization and Input Configuration 31

3.4.4 Class and Label Overview 31

3.5 Tools and Software Environment 32

3.5.1 Software 32

3.5.2 Hardware 32

3.5.3 Reproducibility and Script Structure 33

**Chapter 4: Implementation**   **34**

4.1 Data Preprocessing 34

4.2 Model Architecture 35

4.3 Dataset Preparation and Input Pipeline 36

4.4 Training Configuration 36

4.5 Training Process 36

**Chapter 5: Results**  **37**

5.1 Evaluation Metrics 38

5.2 Noiseless Predictions (∞ dB SNR) 38

5.3 Predictions at 10, 8, 5 dB SNR 40

5.4 Summary and Interpretation 44

**Chapter 6: Conclusion and Future Work** 45

6.1 Conclusion 45

6.2 Future Work 46

6.3 Final Thoughts 46

References 47

**List of Symbols and Abbreviations**

|  |  |
| --- | --- |
| ML | Machine Learning |
| AI | Artificial Intelligence |
| AMC | Automatic Modulation Classification |
| CNN | Convolutional Neural Network |
| SNR | Signal-to-Noise Ratio (dB) |
| ASK | Amplitude Shift Keying |
| FSK | Frequency Shift Keying |
| PSK | Phase Shift Keying |
| DPSK / DQPSK | Differential Phase Shift Keying / Differential Quadrature PSK |
| QAM | Quadrature Amplitude Modulation |
| CSS / Chirp | Chirp Spread Spectrum |
| M | Modulation Order (e.g., M = 2, 4, 8, 16...) |
| Fs | Sampling Frequency (Hz) |
| f₀, f\_c, f\_base | Carrier Frequency / Base Frequency (Hz) |
| T | Total Duration of Signal (seconds) |
| N | Number of Bits per Symbol (log₂M) |
| t | Time vector |
| s(t) or s\_t | Modulated signal in time domain |
| SpecImg | Spectrogram image |
| AWGN | Additive White Gaussian Noise |
| N | Number of Bits per Symbol (log₂M) |
| BT | Bandwidth-Time Product (in filters) |
| FFT | Fast Fourier Transform |
| ReLU | Rectified Linear Unit (activation function) |
| Dropout | Regularization technique used in training neural networks |
| Epoch | One complete pass through the training dataset |
| Mini-Batch | A small, randomly selected portion of the dataset used for one training step |
| Softmax | Activation function used for classification output |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| 4.1 | Summarizes the neural network architecture that’s used | 35 |
| 5.1 | Incoming noiseless signals predictions | 39 |
| 5.2 | Incoming 10dB SNR signals predictions | 41 |
| 5.3 | Incoming 8dB SNR signals predictions | 42 |
| 5.4 | Incoming 5dB SNR signals predictions | 43 |
|  |  |  |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| 1.1 | 2-level ASK Modulation | 2 |
| 1.2 | 4-level ASK | 3 |
| 1.3 | 2-level PSK modulation | 3 |
| 1.4 | 4-level PSK modulation or QPSK | 4 |
| 1.5 | 2-level FSK | 4 |
| 1.6 | 4-level FSK | 5 |
| 1.7 | 16-Level QAM | 5 |
| 1.8 | Chirp Modulation | 6 |
| 3.1 | Shows an example of a 32-level ASK modulation generated in matlab | 16 |
| 3.2 | Same modulation as figure 3.1 but with 10dB SNR added | 16 |
| 3.3 | Same modulation shown in figure 3.1 with 8dB SNR | 17 |
| 3.4 | Same modulation shown in figure 3.1 with 5dB SNR | 17 |
| 3.5 | Shows an example of a 32-level FSK modulation generated in matlab | 18 |
| 3.6 | Same modulation as figure 3.5 but with 10dB SNR added | 19 |
| 3.7 | Same modulation shown in figure 3.5 with 8dB SNR | 19 |
| 3.8 | Same modulation shown in figure 3.5 with 5dB SNR |  |
| 3.9 | Shows an example of a 32-level PSK modulation generated in matlab | 20 |
| 3.10 | Same modulation as figure 3.9 but with 10dB SNR added | 21 |
| 3.11 | Same modulation shown in figure 3.9 with 8dB SNR | 21 |
| 3.12 | Same modulation shown in figure 3.9 with 5dB SNR | 22 |
| 3.13 | Shows an example of a 32-level QAM modulation generated in matlab | 23 |
| 3.14 | Same modulation as figure 3.13 but with 10dB SNR added | 24 |
| 3.15 | Same modulation shown in figure 3.13 with 8dB SNR | 24 |
| 3.16 | Same modulation shown in figure 3.13 with 5dB SNR | 25 |
| 3.17 | Shows an example of a 32-level Chirp modulation generated in matlab | 26 |
| 3.18 | Same modulation as figure 3.17 but with 10dB SNR added | 26 |
| 3.19 | Same modulation shown in figure 3.17 with 8dB SNR | 27 |
| 3.20 | Same modulation shown in figure 3.17 with 5dB SNR | 27 |
| 3.21 | Sample of generated and stored modulations spectrograms | 30 |

**Chapter 1**

**Introduction**

* 1. **Background**

In modern communication systems, especially with how much wireless tech has expanded, being able to identify what kind of digital modulation a signal is using is becoming more important. This is especially true in areas like cognitive radio, military signal monitoring, and any system that needs to automatically understand and react to signal types.

Traditionally, modulation classification was done using hardware tools like spectrum analyzers or multiple demodulators. While that works, it’s not very efficient — it’s slow, can be limited in accuracy, and doesn’t scale well when systems need to be dynamic or automated.

With AI and machine learning becoming more common in signal processing, a more automated and flexible approach is now possible. In this project, the focus is on using AI to classify digital modulation signals. Instead of relying on raw time-domain signals or handcrafted features, the approach here uses spectrograms — basically visual representations of how the signal behaves over time and frequency. These spectrograms are then used as input to a model that learns to recognize different modulation types.

* 1. **Modulation Terminology**

To understand how digital modulation classification works, it's important to first break down some of the key terms used in the process. Modulation itself is the technique of changing a carrier signal in order to transmit information. In digital communication systems, this usually involves encoding binary data (0s and 1s) by adjusting certain properties of the carrier wave — like its amplitude, frequency, or phase.

In digital communication systems, once the data has been processed and encoded, it needs to be transformed into a format that can be effectively transmitted over physical channels. This is where digital modulation comes in. It plays a critical role in shaping the signal for transmission, ensuring that the information can travel across the channel reliably and efficiently. Different modulation techniques offer various trade-offs between data rate, power efficiency, and resistance to noise — making the choice of modulation scheme an important design consideration.

The binary sequence at the output of the channel encoder is passed to the digital

modulator, which serves as the interface to the communication channel. Since nearly

all the communication channels encountered in practice are capable of transmitting

electrical signals (waveforms), the primary purpose of the digital modulator is to map

the binary information sequence into signal waveforms.[1] Digital signals can be broadly categorized based on their modulation techniques.

* + 1. **Amplitude Shift Keying (ASK):**

Amplitude Shift Keying (ASK) is the first digital modulation scheme we will discuss because amplitude modulation is the simplest to visualize of the three sinusoid properties.

s(t)=Am⋅cos(2πfc​t) for bit 1

s(t)=0 for bit 0

Here is an example of 2-level ASK, called 2-ASK:

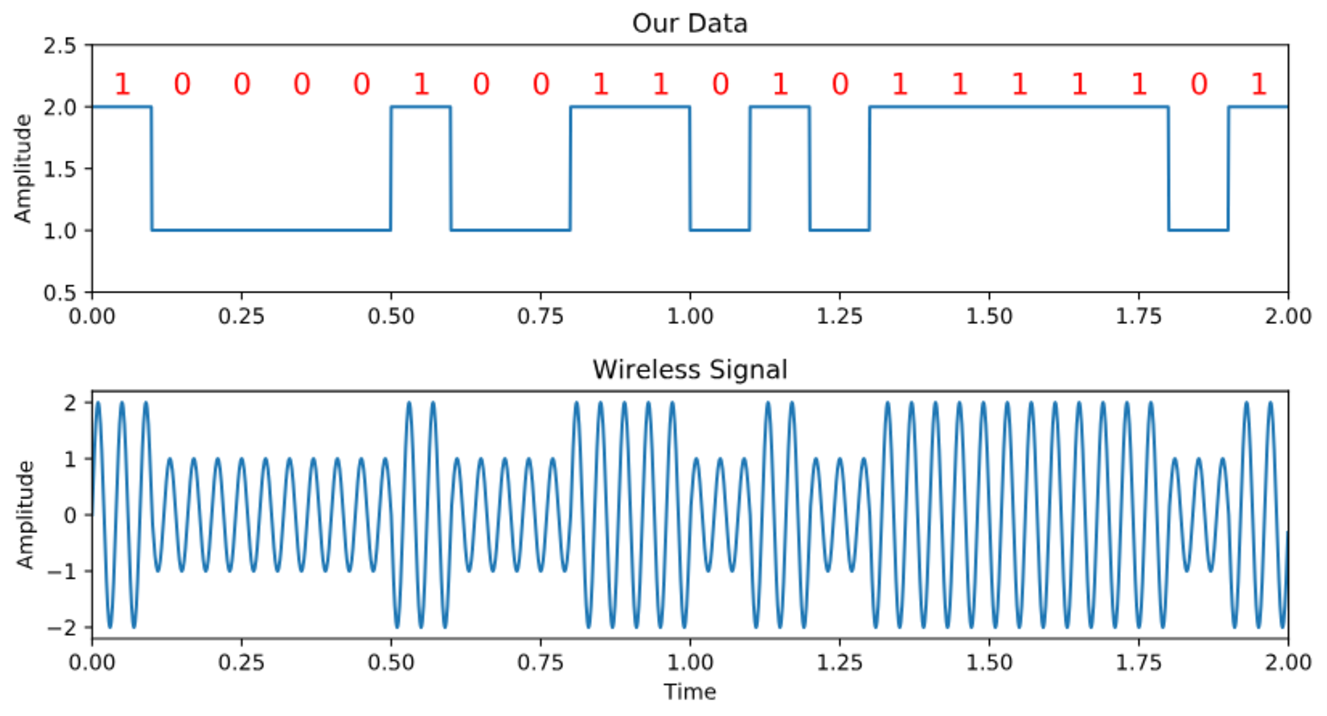
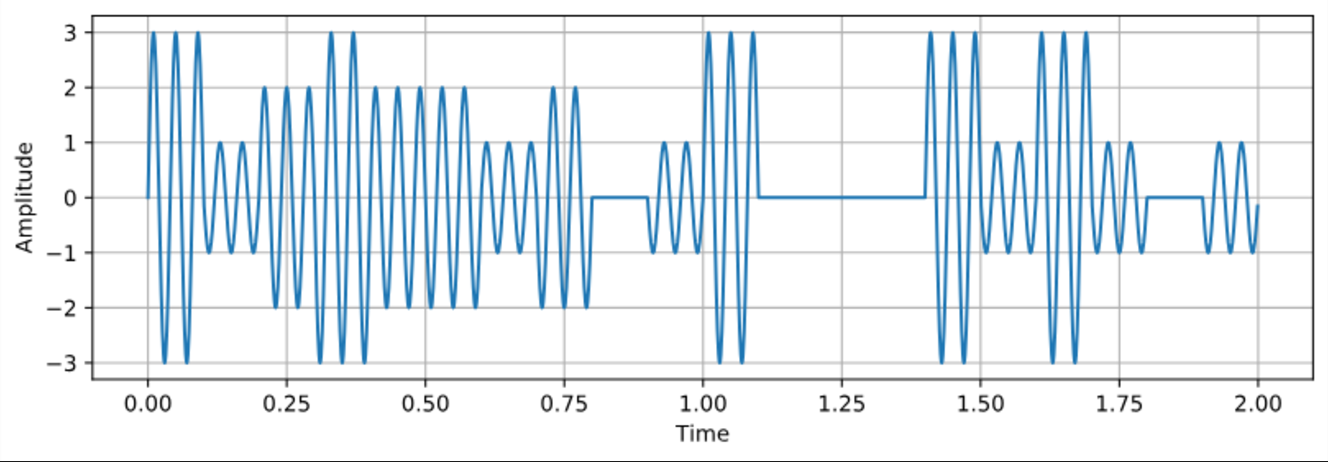


Figure 1.1 2-level ASK Modulation

We can use more than two levels, allowing for more bits per symbol. Below shows an example of 4-ASK.

  
Figure 1.2 4-level ASK

* + 1. **Phase Shift Keying (PSK):**

Utilizes Phase changes to represent symbols. Here, the phase of the carrier wave is changed to represent data. For instance, a 180° phase shift might separate a '0' from a '1' in Binary PSK (BPSK). Higher-order PSK (like QPSK or 8-PSK) uses more phase angles to encode multiple bits at once.

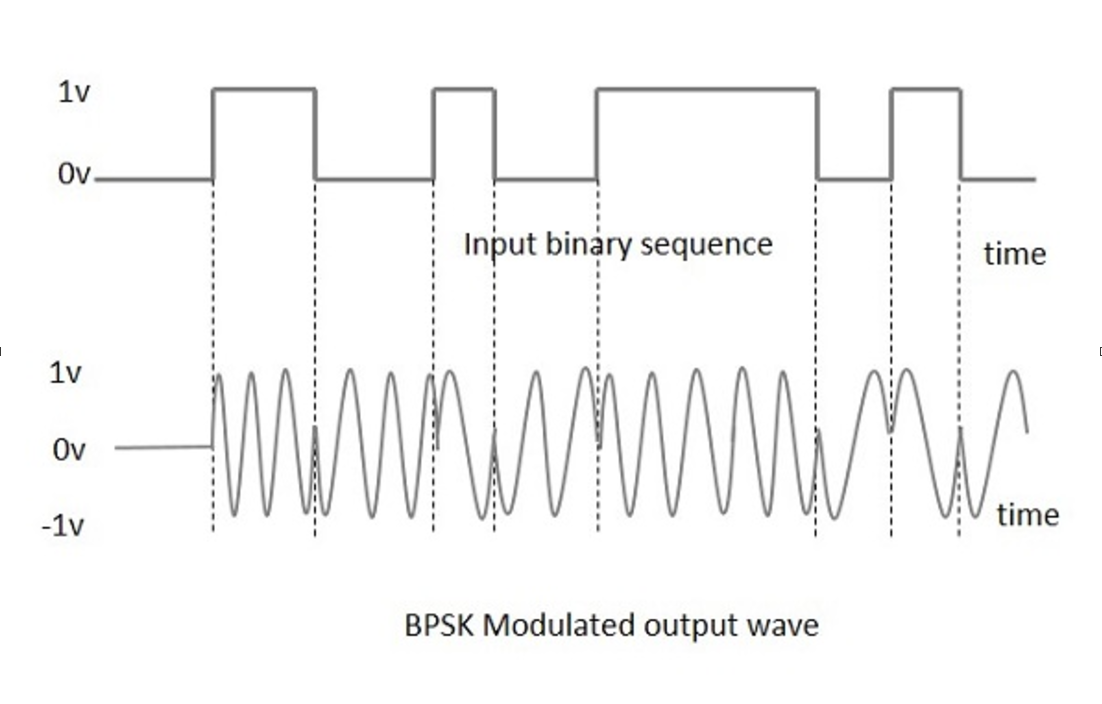


Figure 1.3 2-level PSK modulation

When you use 4-level modulation in PSK, it becomes a widely used modulation known as QPSK.

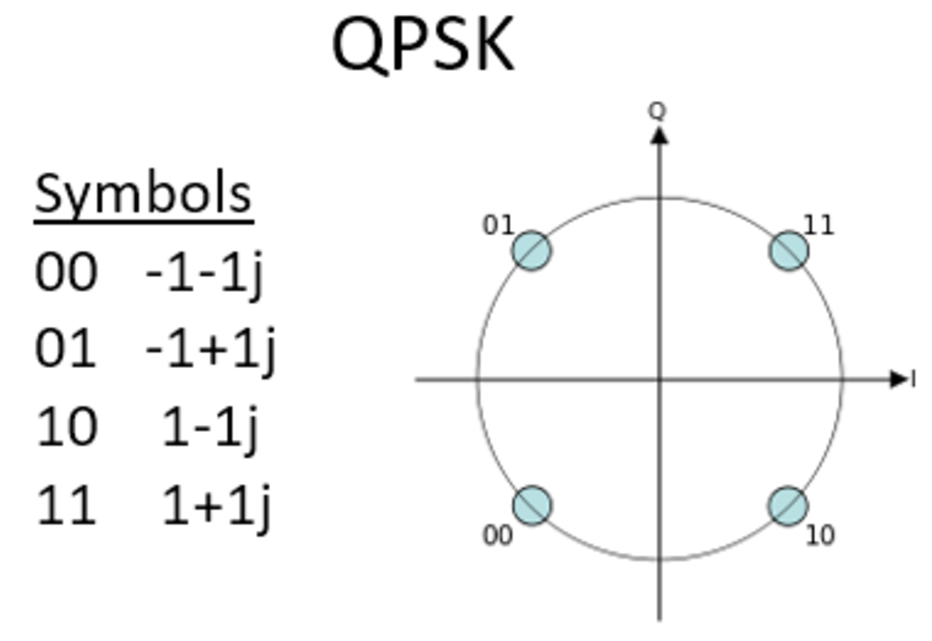


Figure 1.4 4-level PSK modulation or QPSK

DPSK is a variation of PSK where the information is encoded in the difference between phases of successive symbols instead of absolute phase values. This reduces the need for a coherent phase reference at the receiver. Like PSK, M-ary DPSK can be used (e.g., DQPSK for M = 4), allowing the system to transmit more bits per symbol while keeping receiver complexity low.

* + 1. **Frequency Shift Keying (FSK):**

Frequency Shift Keying (FSK) is a technique that conveys data by varying the frequency of a carrier signal. In FSK, different frequencies represent 0 and 1. Δf, or the frequency deviation, defines the difference between the two frequencies used.

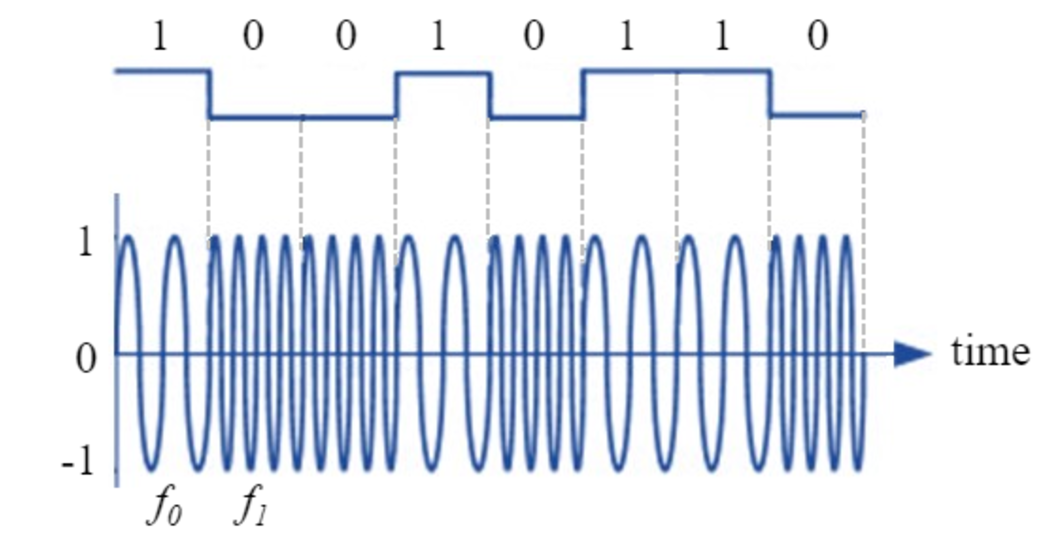


Figure 1.5 2-level FSK

When having multiple bits per symbol, each state gets its own frequency.



Figure 1.6 4-level FSK

* + 1. **Quadrature Amplitude Modulation (QAM):**

QAM combines both amplitude and phase modulation, allowing it to carry more data. For example, in 16-QAM, each symbol can represent 4 bits of information. It’s commonly used in modern systems like Wi-Fi and LTE but is more sensitive to noise.

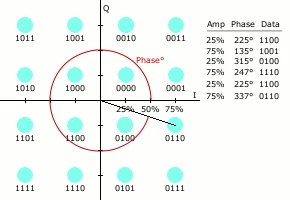
****

Figure 1.7 16-Level QAM

* + 1. **Chirp Modulation**

This is a frequency-modulated signal where the frequency increases or decreases over time. It's often used in radar and spread-spectrum systems because it's resilient to interference and works well for time-delay estimation.



Figure 1.8 Chirp Modulation

* 1. **Introduction to Machine Learning in Signal Processing**

In the field of Artificial Intelligence (AI), one of the most transformative and widely used techniques is Machine Learning (ML). It allows systems to automatically learn and improve from experience without being explicitly programmed for every specific task. By analyzing patterns and drawing inferences from data, machine learning enables computers to perform tasks that once required human intelligence.

* + 1. **What is Machine Learning?**

When the agent is a computer, we call it machine learning: a computer observes some data, builds a model based on the Machine learning data, and uses the model as both a hypothesis about the world and a piece of software that can solve problems.[2]

* + 1. **Supervised Learning**

In supervised learning the agent observes input-output pairs and learns a function that Supervised learning maps from input to output. For example, the inputs could be camera images, each one accompanied by an output saying “bus” or “pedestrian,” etc. An output like this is called a label. The agent learns a function that, when given a new image, predicts Label the appropriate label. In the case of braking actions (component 1 above), an input is the current state (speed and direction of the car, road condition), and an output is the

distance it took to stop. In this case a set of output values can be obtained by the agent

from its own percepts (after the fact); the environment is the teacher, and the agent

learns a function that maps states to stopping distance.[3]

* + 1. **Neural Networks**

The term ‘neural network’ has its origins in attempts to find mathematical representations of information processing in biological systems (McCulloch and Pitts,

1943; Widrow and Hoff, 1960; Rosenblatt, 1962; Rumelhart et al., 1986). Indeed,

it has been used very broadly to cover a wide range of different models, many of

which have been the subject of exaggerated claims regarding their biological plausibility. From the perspective of practical applications of pattern recognition, however, biological realism would impose entirely unnecessary constraints. Our focus in this chapter is therefore on neural networks as efficient models for statistical pattern recognition. In particular, we shall restrict our attention to the specific class of neural networks that have proven to be of greatest practical value, namely the multilayer perceptron.[4]

* 1. **History**

Modulation classification has been around for a long time, mostly handled by traditional hardware tools and manual techniques. In older systems, engineers would use things like spectrum analyzers, oscilloscopes, and demodulators to inspect signals and figure out their modulation type based on experience or predefined rules. These methods worked, but they weren’t scalable or efficient, especially in complex or noisy environments.

As communication systems got more advanced and dynamic, the idea of automatic modulation classification started to gain attention. Early approaches were mainly based on mathematical modeling and handcrafted features — using signal parameters like amplitude, phase, and frequency characteristics to build logic-based systems or rule engines that could tell the difference between modulation types.

With the rise of AI and machine learning, especially in the last decade, the focus shifted toward data-driven techniques. Instead of defining features manually, models like neural networks could be trained to learn the differences between signals on their own. More recently, approaches using spectrograms — which turn signals into visual patterns — have made it easier to apply image-based models like Convolutional Neural Networks (CNNs) for classification. This opened up a new path for building flexible, accurate, and fully automatic modulation classifiers that work well even under noisy or unknown conditions.

* 1. **Brief Description of the Work**

The work in this project mainly focuses on building an AI-based model to automatically classify digital modulation signals. The implementation was done using MATLAB, where multiple functions were written to simulate different types of modulation schemes. These include ASK, FSK, PSK, DPSK, QAM, and Chirp, with support for varying modulation orders (M-values) passed as arguments.

To prepare the dataset for training, a loop was created to generate thousands of modulated signal samples for each modulation type and each value of M. Each signal was then converted into a spectrogram image, which provides a time-frequency representation of the signal. These spectrograms were saved and later used as input for the training phase.

Supervised learning techniques were used to train a model on the generated dataset. After training, a set of manual tests were performed to evaluate the model’s accuracy and performance, and the results were recorded.

* 1. **Aim**

The aim of this work is to explore how artificial intelligence can be used to automatically classify digital modulation schemes. The idea behind choosing this topic is to better understand how machine learning can be applied to signal processing problems in a more modern, efficient way compared to traditional methods. By the end of the project, the goal was to build a working system that can take a modulated signal, process it into a spectrogram, and correctly identify its modulation type using a trained AI model.

* 1. **Report Layout**

To provide a clear and structured overview of the work carried out in this project, the report has been organized into six chapters. Each chapter focuses on a specific aspect of the project, guiding the reader from the initial motivation through to the final conclusions and recommendations. A brief description of each chapter is provided below:

Chapter 1: Introduces the concept of automatic modulation classification using AI and outlines the objectives, motivation, and scope of the project.

Chapter 2: Reviews existing literature on traditional and AI-based modulation classification methods, and highlights the advantages of using spectrograms and machine learning.

Chapter 3: Describes the methodology used to simulate digital modulation schemes in MATLAB, generate spectrograms, and prepare the dataset for training.

Chapter 4: Details the implementation of the AI model, training process, and evaluation using supervised learning techniques.

Chapter 5: Presents the results of the classification model, evaluates its performance, and discusses the effectiveness of the approach.

Chapter 6: Concludes the project, summarizes key findings, and suggests possible directions for future work.

**Chapter 2**

**Literature Review**

* 1. **Introduction**

This chapter presents a review of previous research and methods used in the classification of digital modulation schemes. It begins by discussing traditional techniques, then explores the emergence of artificial intelligence and machine learning in signal processing. Special focus is given to spectrogram-based approaches, which are closely related to the methodology used in this project. The chapter concludes by identifying gaps in existing work and justifying the need for the current approach.

* 1. **Traditional Methods of Modulation Classification**

Early modulation classification techniques primarily relied on rule-based systems and expert-designed features. These traditional approaches involved extracting statistical or deterministic features from the received signal—such as amplitude, phase, and frequency characteristics—and applying manually crafted decision rules to classify modulation types.

Examples include:

* Likelihood-based methods, which evaluate the probability of a signal belonging to a particular modulation class.
* Decision tree and threshold-based techniques, which use hard-coded logic to distinguish between signal patterns.
* Higher-order statistical analysis, like cumulants or cyclostationary analysis, to capture signal structure.

While these methods are interpretable and low in computational cost, they are often sensitive to noise and require domain-specific knowledge for feature engineering.

* 1. **AI and Machine Learning in Signal Processing**

With the growing availability of computational resources and data, machine learning (ML) methods have become increasingly popular for modulation classification tasks. Artificial neural networks (ANNs), in particular, have demonstrated the ability to learn complex patterns from raw or minimally processed data. Several notable studies include:

* “Automatic Recognition of the Digital Modulation Types Using Artificial Neural Networks” [5] – This work demonstrated how basic feedforward neural networks can distinguish between modulation types using time-domain features.
* “Classification of Modulation Signals Using Statistical Signal Characterization and Artificial Neural Networks” [6] – Combined feature extraction with ANN models to achieve improved accuracy.

More recently, deep learning techniques have gained traction: “Deep Learning-Based Automatic Modulation Classification Using Robust CNN Architecture for Cognitive Radio Networks” [7] – Utilized convolutional neural networks (CNNs) to automatically extract features from spectrograms, improving performance under noisy conditions. These approaches reduce the need for manual feature extraction and offer better adaptability to different signal conditions.

* 1. **Spectrogram-Based Classification**

Spectrograms provide a time-frequency representation of signals, making them suitable for image-based machine learning models such as CNNs. This technique transforms 1D signals into 2D images, capturing both spectral and temporal characteristics, which are useful for distinguishing between modulation types.

One relevant paper, “A New Approach to Signal Classification Using Spectral Correlation and Neural Networks,” [8] explored this idea by combining spectral correlation features with neural network classifiers. In this project, spectrograms are used as input images to a neural network, leveraging their rich representation of signal characteristics to enable robust classification across multiple modulation types and M-values.

* 1. **Gaps in Existing Work**

Despite the progress in applying AI to modulation classification, several challenges remain:

* Many existing approaches focus on a limited set of modulation types or assume fixed parameters.
* Few studies simulate a wide range of M-values or include more complex modulation schemes like Chirp.
* The use of spectrograms is still relatively underexplored, especially in MATLAB-based environments tailored for engineering applications.
* There's limited integration of simulation and classification pipelines in a single, automated framework.

This project addresses these gaps by:

* Simulating a diverse set of modulation schemes in MATLAB.
* Generating large-scale datasets with varied M-values.
* Using spectrograms as a standardized input format.
* Training a supervised learning model for robust and scalable modulation classification.

**Chapter 3**

**Methodology**

* 1. **Overview of the Approach**

The goal of this project is to automatically classify digital modulation schemes using artificial intelligence. The approach follows a structured pipeline consisting of the following key stages:

1. Signal Simulation: Generate synthetic signals for different modulation types using MATLAB.
2. Spectrogram Generation: Convert time-domain signals into time-frequency images (spectrograms).
3. Use a loop to generate a large dataset for training.
4. Model Training: Use a supervised learning algorithm to classify spectrograms based on modulation type.
5. Validation and Testing: Evaluate model performance using accuracy metrics and test cases.

This chapter details each step involved in the development and implementation of the system.

* 1. **Simulation of Modulation Schemes**

MATLAB was used to simulate digital modulation schemes:

* Amplitude Shift Keying (ASK)
* Frequency Shift Keying (FSK)
* Phase Shift Keying (PSK)
* Differential PSK (DPSK)
* Quadrature Amplitude Modulation (QAM)
* Chirp Modulation

A script was created for each modulation type which also allows for the symbol states to be passed, to allow for a wide variety of modulation types. We’ll be going through every modulation type and how it was generated.

Before going through the generation of the modulations, we’ll go over the common parameters used in the modulations. The following simulation parameters were used consistently across all modulation schemes to ensure a fair comparison and controlled environment for signal generation and analysis:

* input\_bits: A random binary sequence of length 1000 was generated using MATLAB’s randi function. This sequence served as the baseband data input for each modulation scheme.
* Fs (Sampling Frequency): The sampling frequency was set to 20,000 Hz. This value ensures that the generated waveforms are sampled well above the Nyquist rate relative to the carrier and symbol frequencies.
* symbol\_rate: The rate at which symbols are transmitted was fixed at 1,000 symbols per second. This determines the duration of each symbol.
* N: The number of bits per symbol, calculated as log2(M). For a modulation order of 32, this results in N = 5 bits per symbol.
* num\_symbols: The total number of symbols was computed as the ceiling of the ratio between the total number of bits and the number of bits per symbol. This accounts for all input bits within the modulation framework.
* symbol\_duration: The duration of a single symbol, defined as the inverse of the symbol rate. This sets the time span of each transmitted symbol.
* T: The total time duration of the signal, calculated as the product of the number of symbols and the symbol duration.
* f\_c (Carrier Frequency): The carrier frequency for modulation was set to 2,000 Hz, which served as the base frequency for schemes such as ASK, PSK, and QAM.
* f\_base: This is the base frequency used for schemes such as Frequency Shift Keying (FSK) and Chirp Modulation, where frequency variation is key. It was also set to 2,000 Hz.
* freq\_range: A frequency range of 3,000 Hz was defined to allow sufficient separation between frequencies in FSK and to define the bandwidth for chirp signals.
* SNR\_values: A set of Signal-to-Noise Ratio (SNR) values [10 dB, 8 dB, 5 dB] were used to evaluate the performance of the modulation schemes under varying noise conditions.
  + 1. **Amplitude Shift Keying (ASK)**

Amplitude Shift Keying (ASK) is a basic form of digital modulation where the amplitude of the carrier wave is varied in accordance with the digital data being transmitted. In this project, ASK modulation was implemented in MATLAB through a custom function designed to support M-ary ASK, meaning it can handle multiple amplitude levels depending on the modulation order (M).

The function askMModulate() accepts the following inputs:

* A binary input sequence (input\_bits)
* Sampling frequency (Fs)
* Carrier frequency (f\_c)
* Signal duration (T)
* Modulation order (M), which must be a power of 2

The modulation process begins by grouping the binary input into symbols. Each group contains log₂(M) bits, which are then mapped to an index representing one of M amplitude levels. These amplitudes are defined over a normalized range (e.g., 0.2 to 1.0) using MATLAB’s linspace() function to ensure even spacing across all M levels. The signal is then constructed symbol by symbol by multiplying each amplitude with a cosine carrier wave.

Each modulated symbol is assigned a portion of the total signal duration. The time vector t is divided accordingly, and each segment of the output signal s\_t is filled with a cosine wave at the carrier frequency and scaled by the respective amplitude. This method allows for flexible ASK modulation with various M values. For instance: M = 2 results in traditional binary ASK (only two amplitude levels), M = 4, 8, or 16 allows more data to be packed per symbol by using more distinct amplitude levels, increasing data rate but also making the signal more sensitive to amplitude distortion. This modulation approach provides a clean and scalable way to generate synthetic ASK signals for training the classification model.

We’ll be going over figures of modulations generated in MATLAB, with and without noise, and we’ll explore the visual differences between the spectrograms.

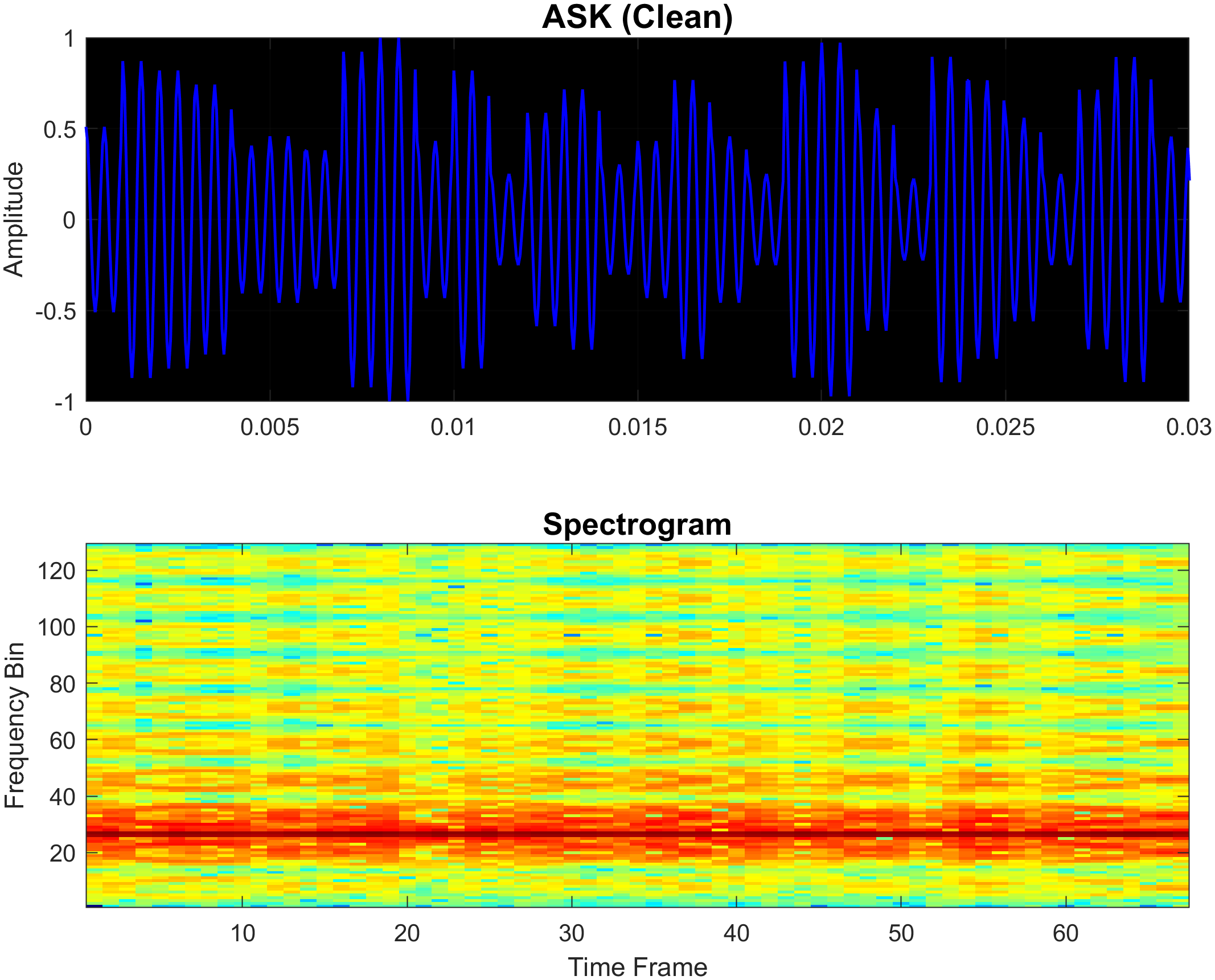


Figure 3.1 Shows an example of a 32-level ASK modulation generated in matlab

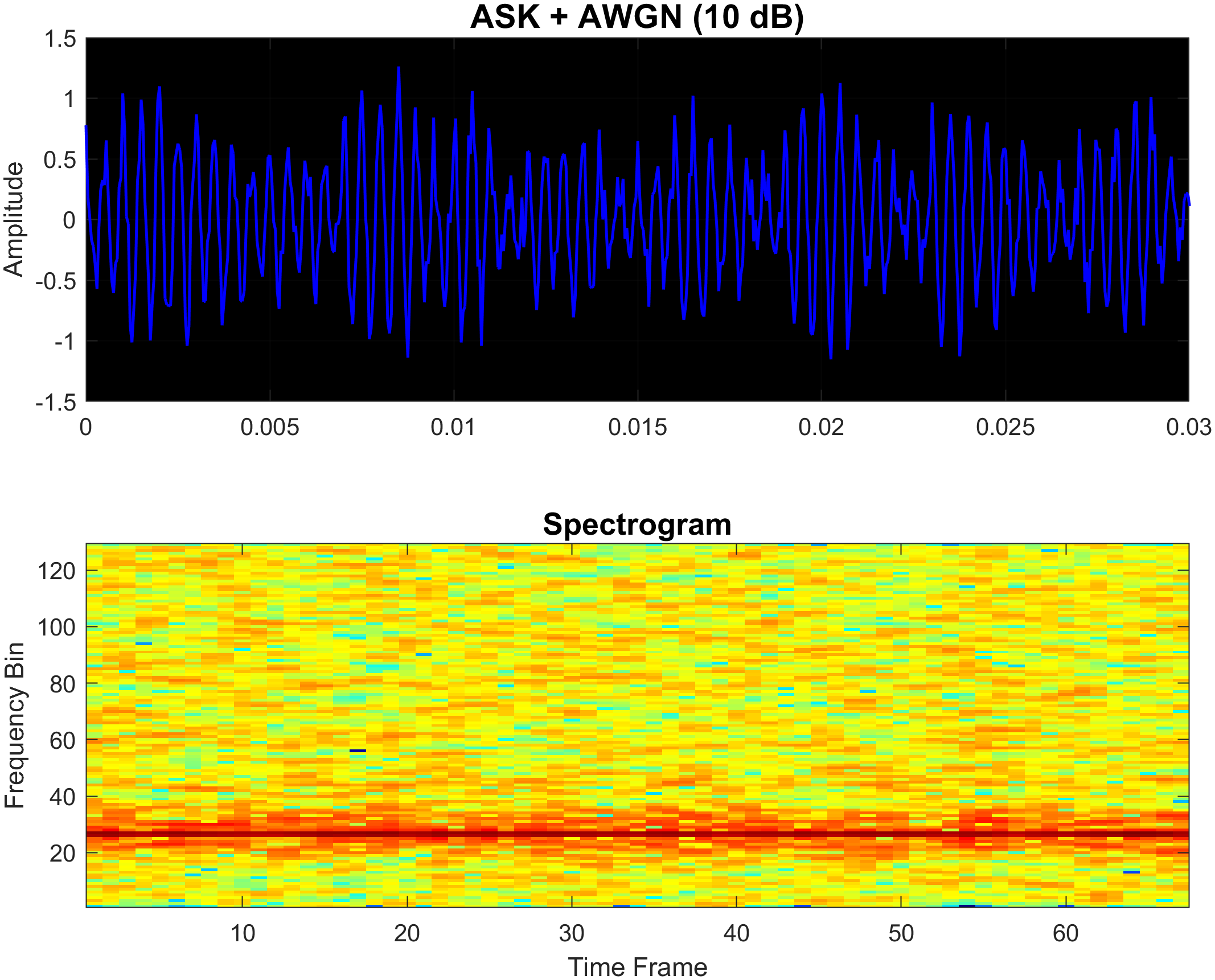


Figure 3.2 Same modulation as figure 3.1 but with 10dB SNR added

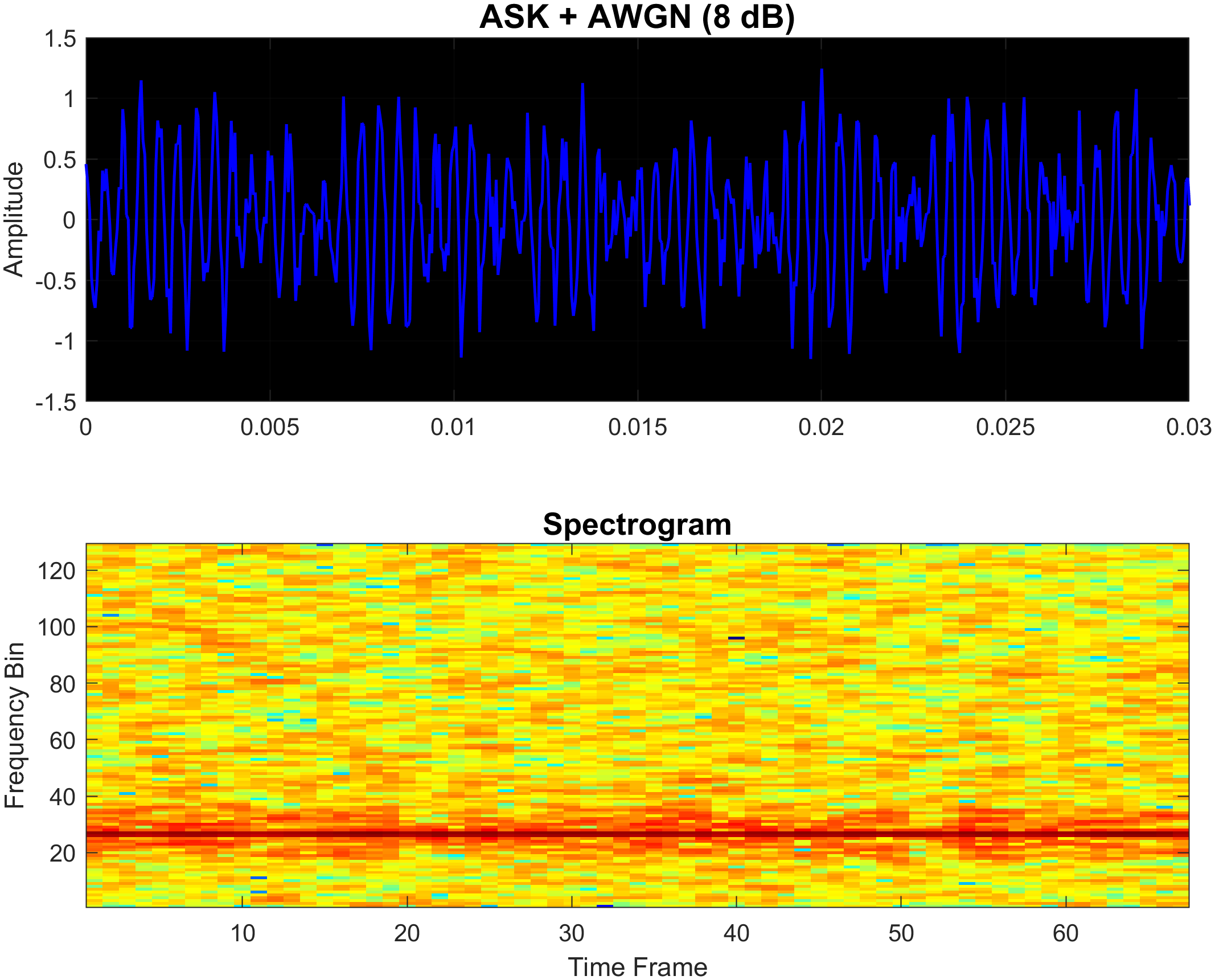


Figure 3.3 Same modulation shown in figure 3.1 with 8dB SNR

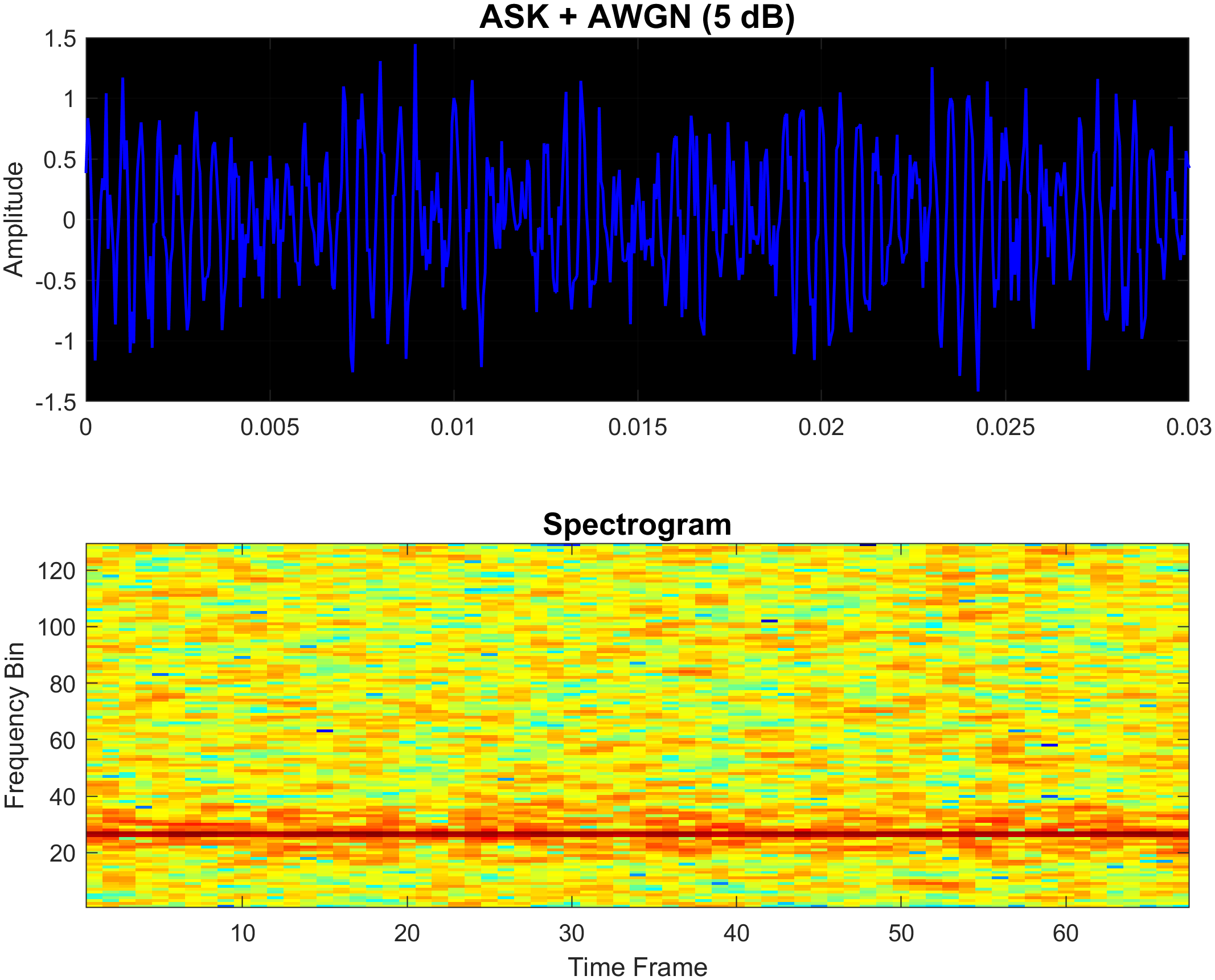


Figure 3.4 Same modulation shown in figure 3.1 with 5dB SNR

* + 1. **Frequency Shift Keying (FSK)**

Frequency Shift Keying (FSK) is a digital modulation technique where data is encoded by varying the frequency of the carrier signal. In M-ary FSK, bits are grouped into symbols, each mapped to one of M distinct frequencies. The custom MATLAB function fskMModulate() was developed to implement M-ary FSK. It takes the following inputs:

* Binary input sequence (input\_bits)
* Sampling frequency (Fs)
* Base frequency (f\_base)
* Total signal duration (T)
* Modulation order (M), which must be a power of 2

Each bit group is converted to a symbol index, which is mapped to a frequency within a defined range (e.g., ±500 Hz around the base frequency). A cosine wave is then generated for each symbol at its corresponding frequency and placed in its time slot.



Figure 3.5 Shows an example of a 32-level FSK modulation generated in matlab

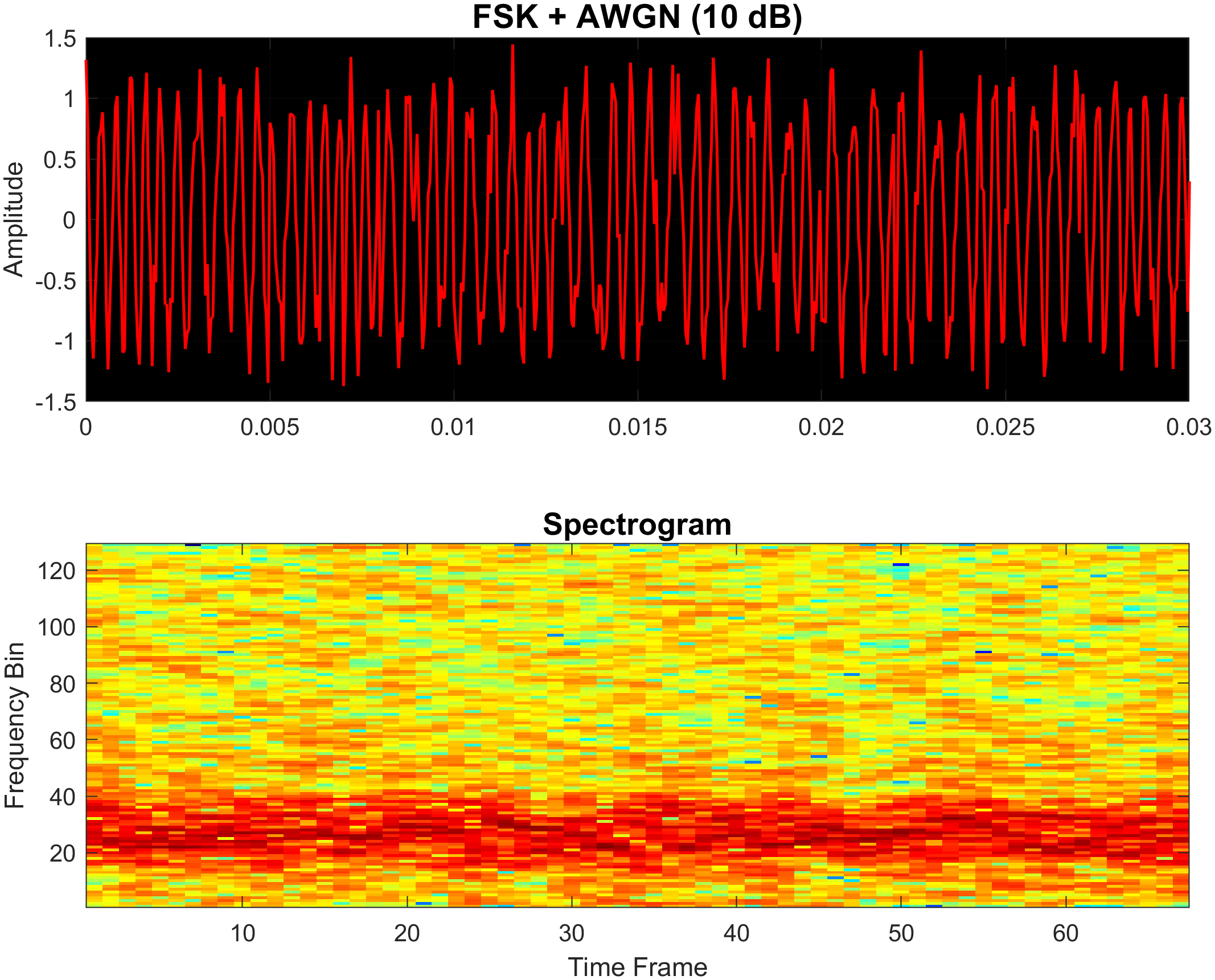


Figure 3.6 Same modulation as figure 3.5 but with 10dB SNR added

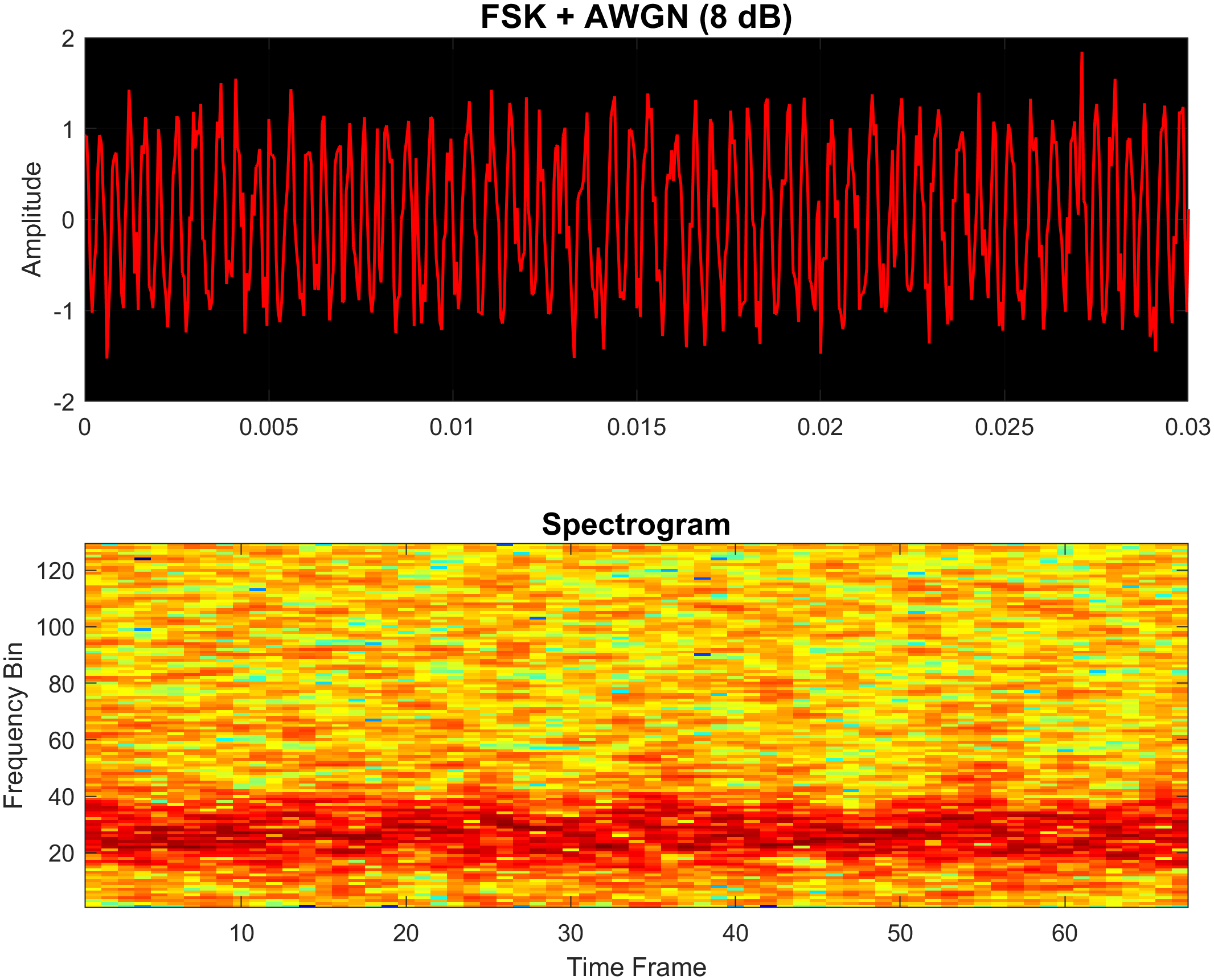


Figure 3.7 Same modulation shown in figure 3.5 with 8dB SNR

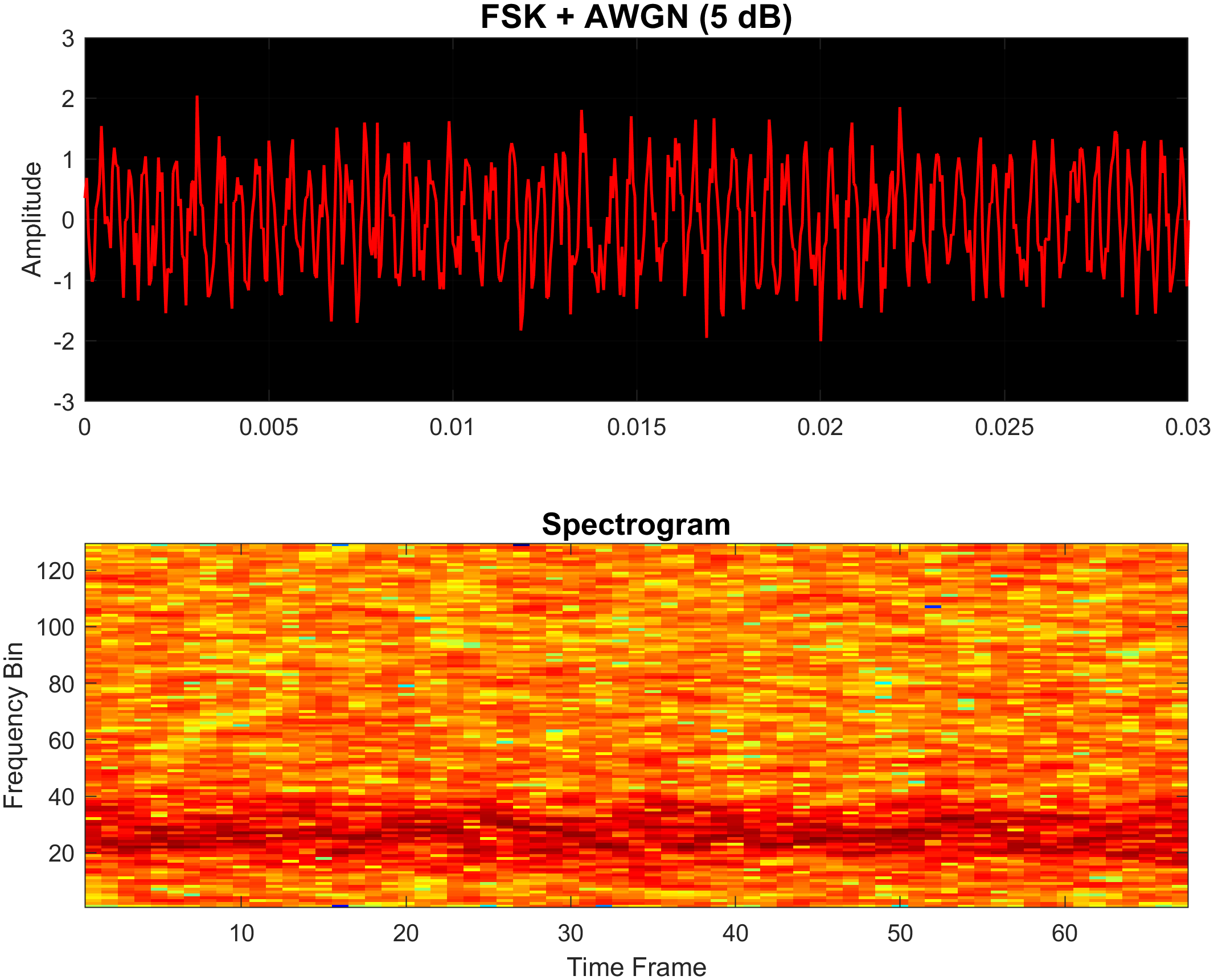


Figure 3.8 Same modulation shown in figure 3.5 with 5dB SNR

* + 1. **Phase Shift Keying (PSK)**

Phase Shift Keying (PSK) is a digital modulation scheme in which information is encoded in the phase of the carrier signal. In M-ary PSK, each symbol represents bits and is mapped to one of M distinct phase shifts. This project employs a MATLAB function, pskMModulate(), to simulate M-ary PSK. The function accepts:

* A binary input sequence (input\_bits)
* Sampling frequency (Fs)
* Carrier frequency (f\_c)
* Total signal duration (T)
* Modulation order (M), constrained to powers of 2

The input bits are grouped into symbols and converted into indices, each corresponding to a unique phase angle evenly spaced between 0 and 2π. For each symbol, a cosine wave is generated with the carrier frequency and its assigned phase offset. The modulated signal is constructed by sequentially placing each phase-shifted cosine wave into its corresponding time slot, based on the total duration and sampling frequency.

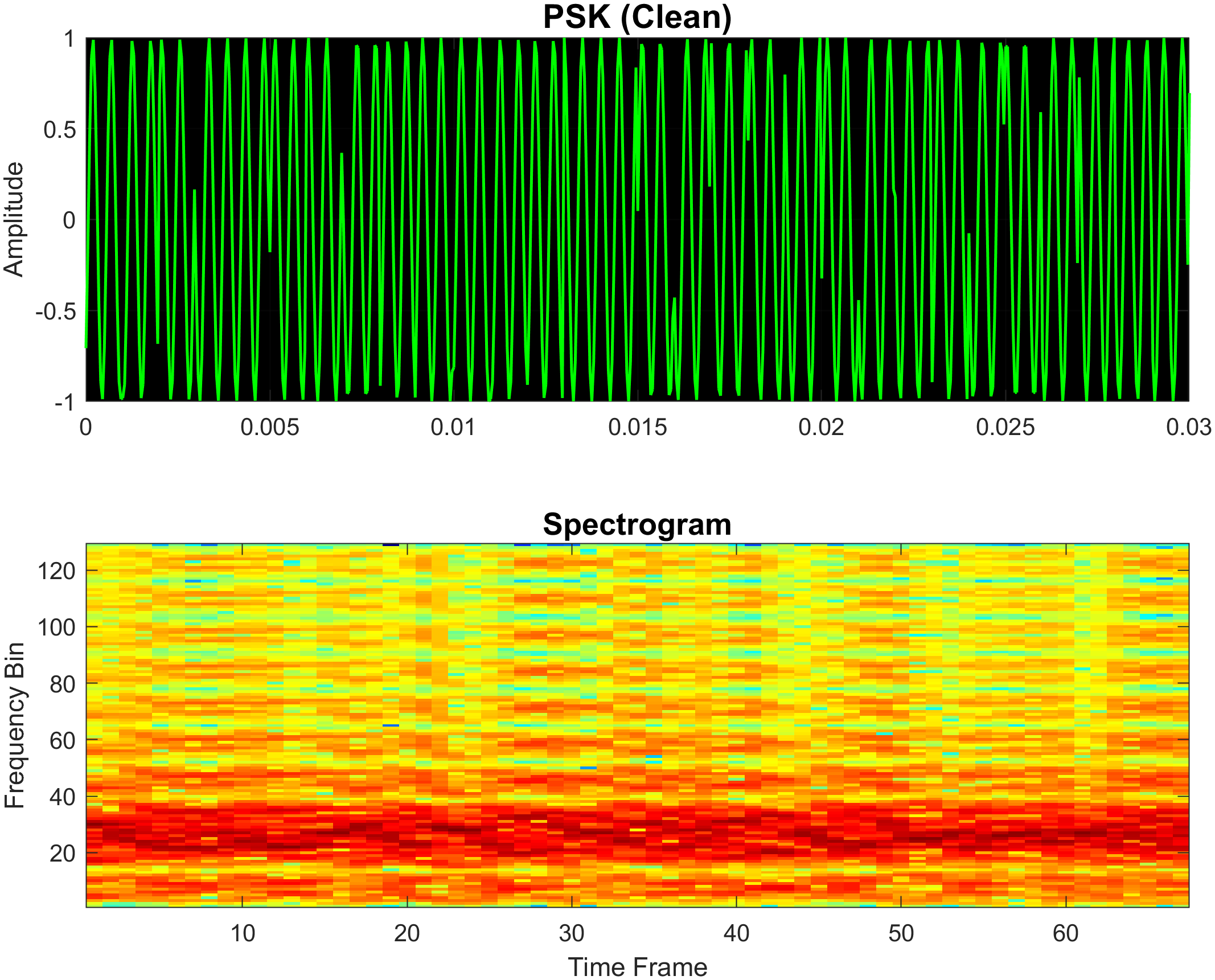


Figure 3.9 Shows an example of a 32-level PSK modulation generated in matlab

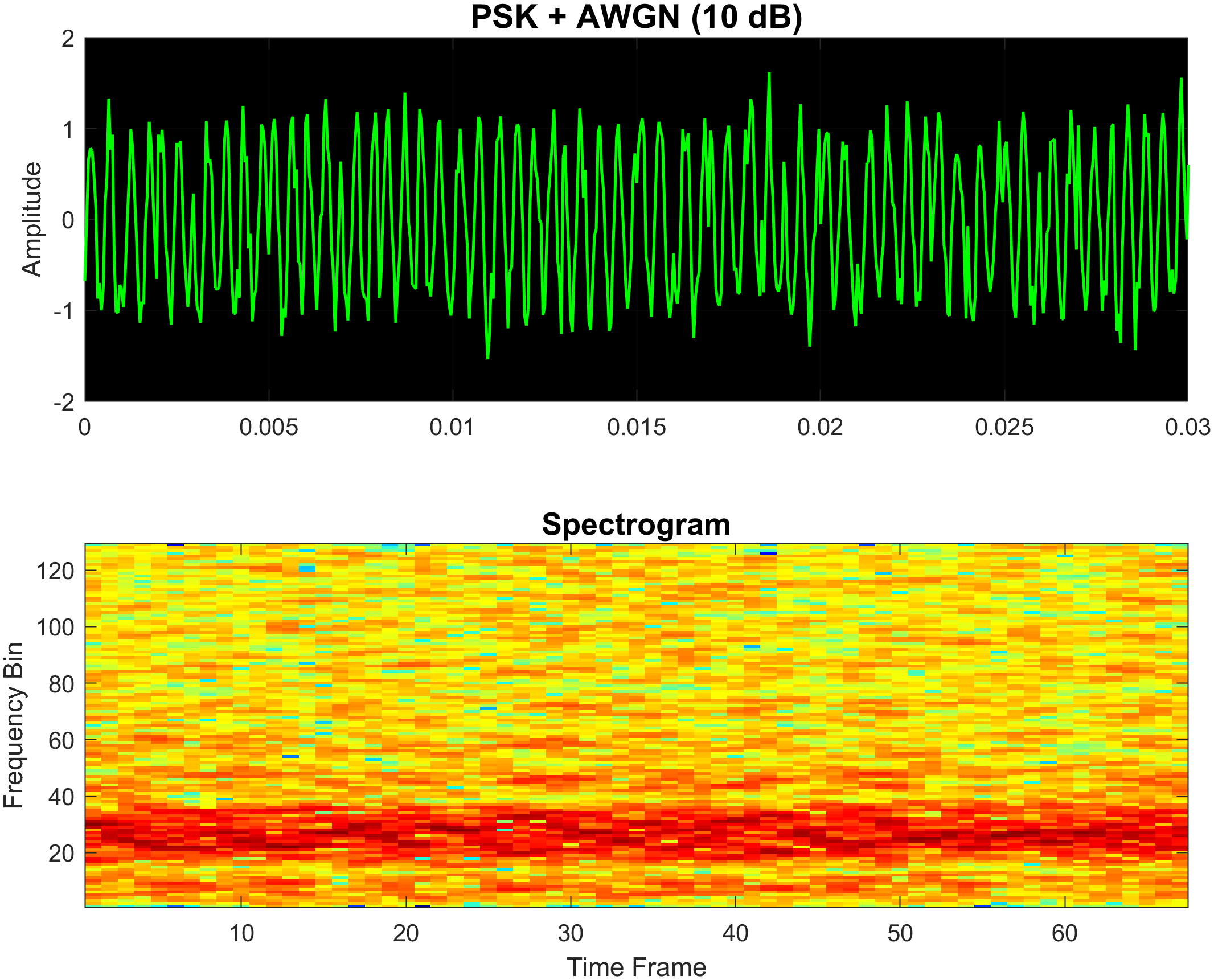


Figure 3.10 Same modulation as figure 3.9 but with 10dB SNR added

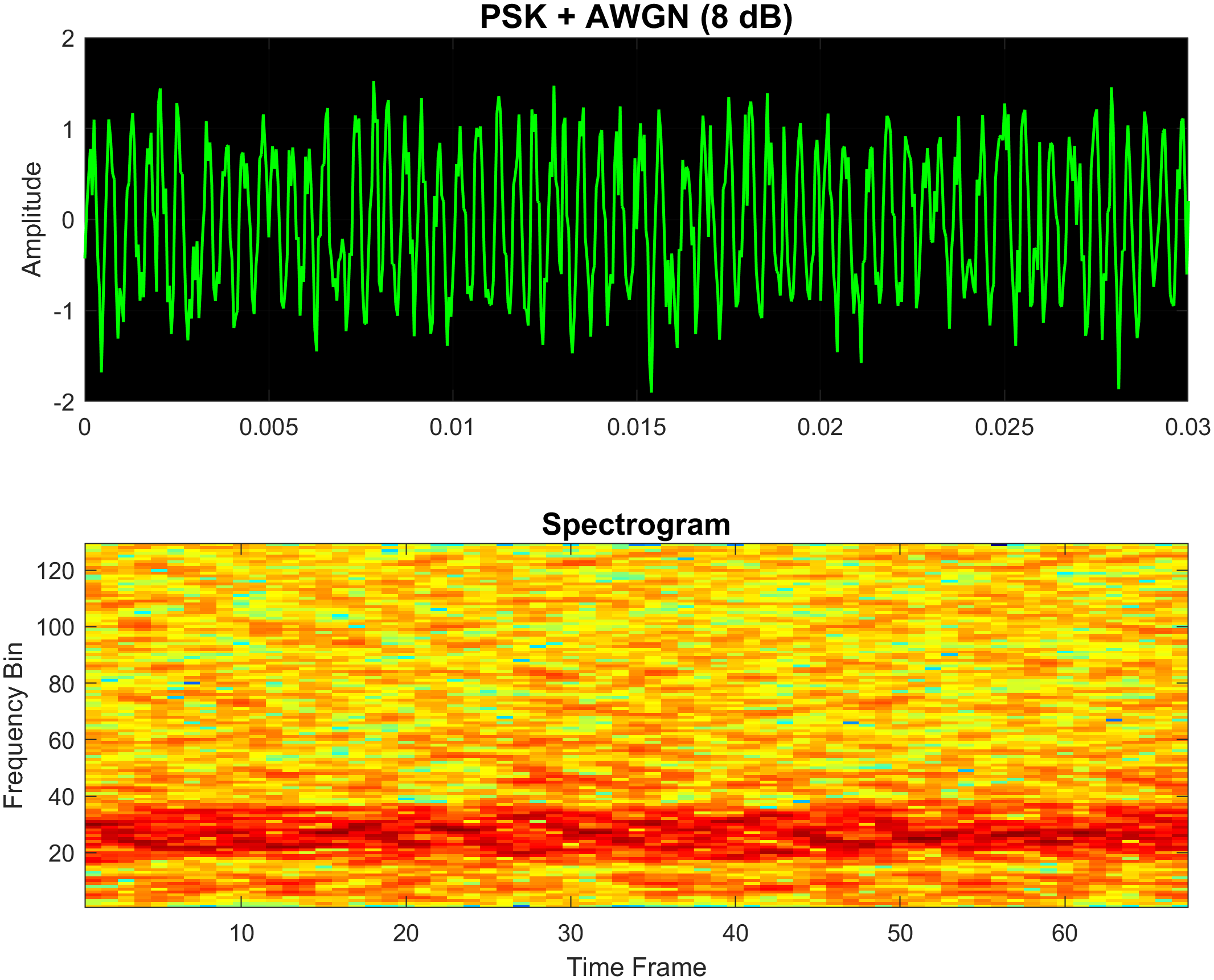


Figure 3.11 Same modulation shown in figure 3.9 with 8dB SNR

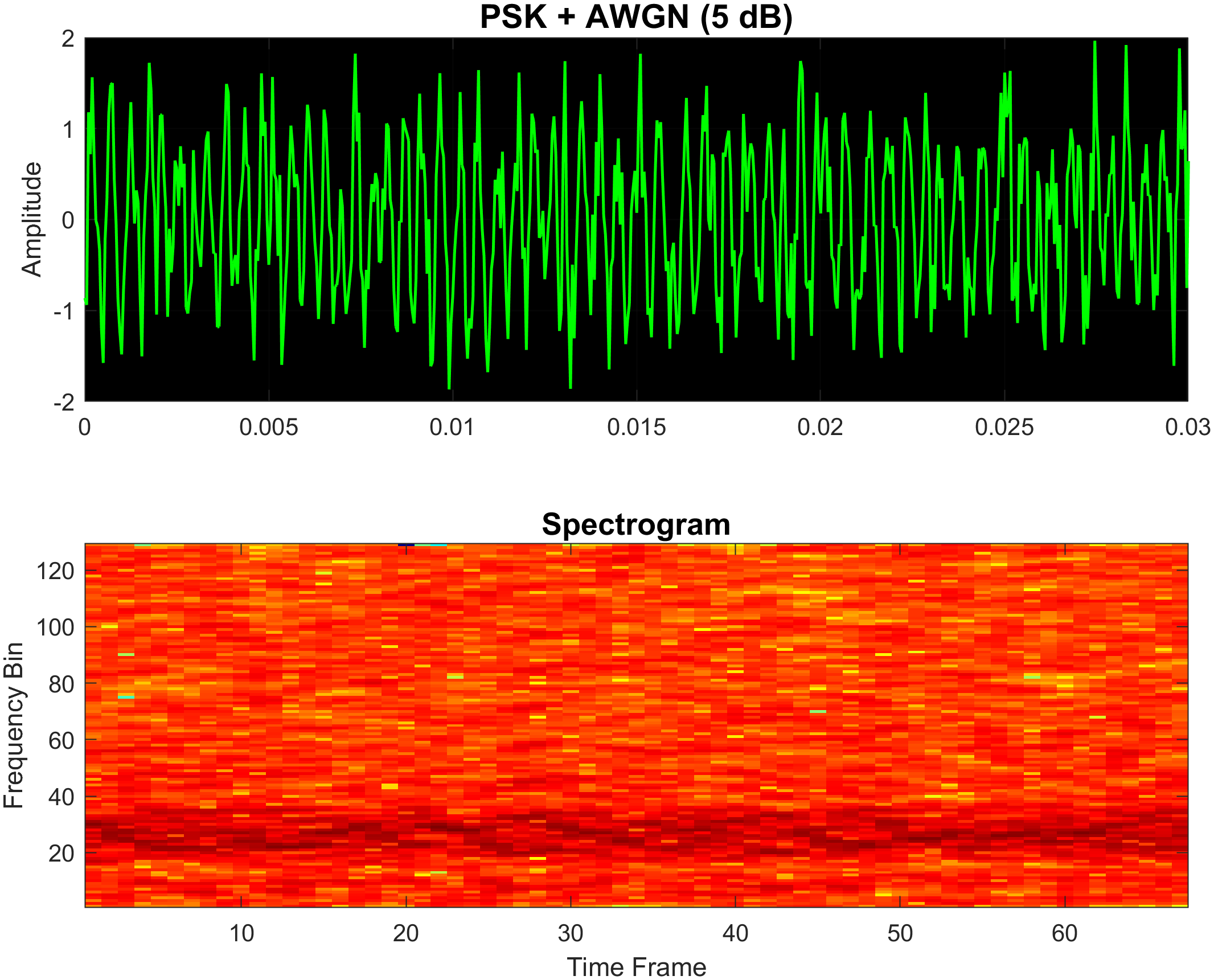


Figure 3.12 Same modulation shown in figure 3.9 with 5dB SNR

* + 1. **Quadrature Amplitude Modulation (QAM)**

Quadrature Amplitude Modulation (QAM) combines both amplitude and phase modulation, allowing the transmission of multiple bits per symbol. In M-ary QAM, ​bits are grouped into symbols, which are then mapped to points on a 2D constellation grid, defined by both amplitude and phase. The qamMModulate() function implements M-ary QAM in MATLAB, accepting the following inputs:

* A binary input sequence (input\_bits)
* Sampling frequency (Fs)
* Carrier frequency (f\_c)
* Total signal duration (T)
* Modulation order (M), which must be a power of 2

The input bits are grouped into symbols and converted into indices. These indices are then mapped to a custom QAM constellation, generated in polar coordinates. The constellation consists of multiple rings, each with different amplitudes and phase angles, to define the possible symbol points.

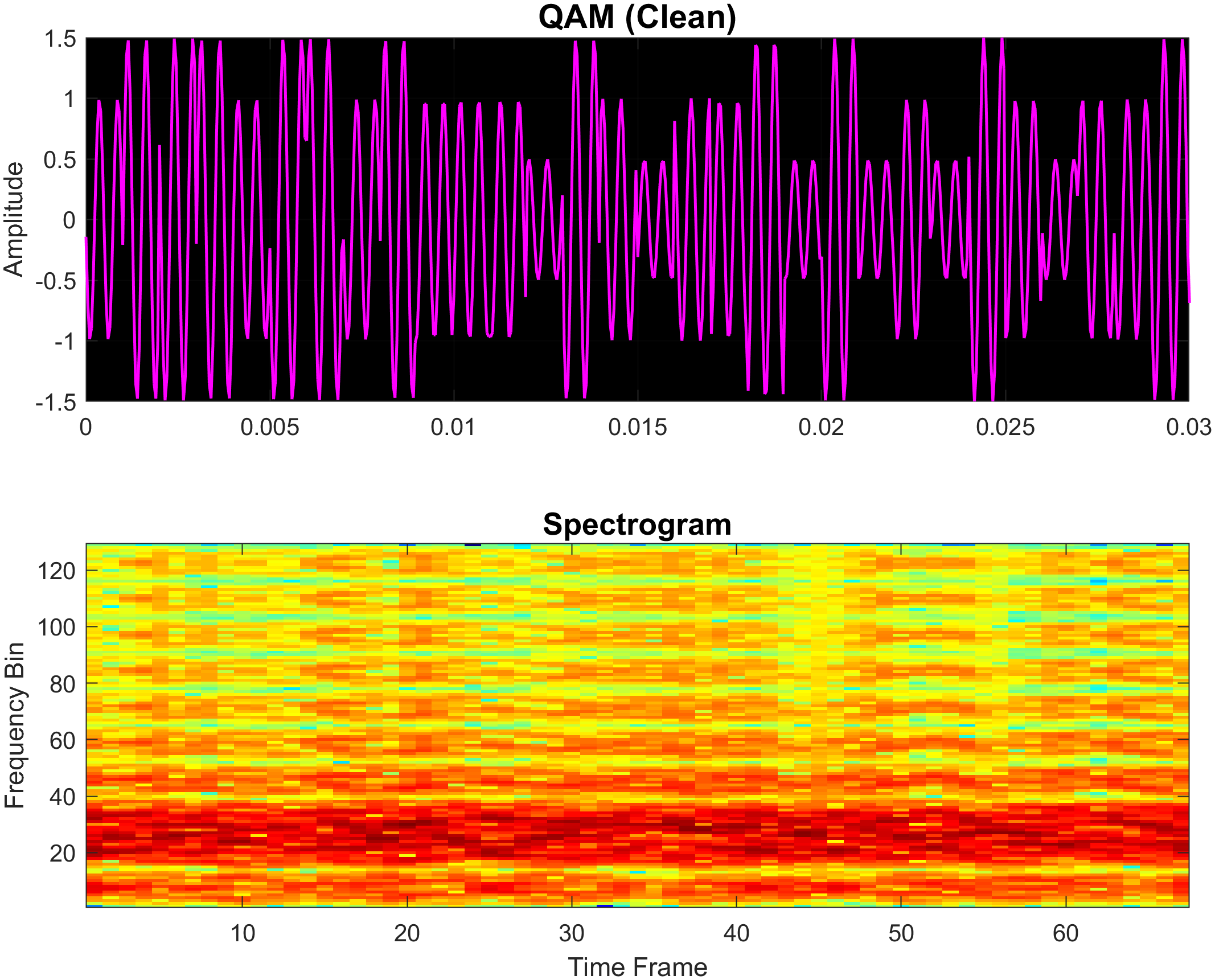


Figure 3.13 Shows an example of a 32-level QAM modulation generated in matlab

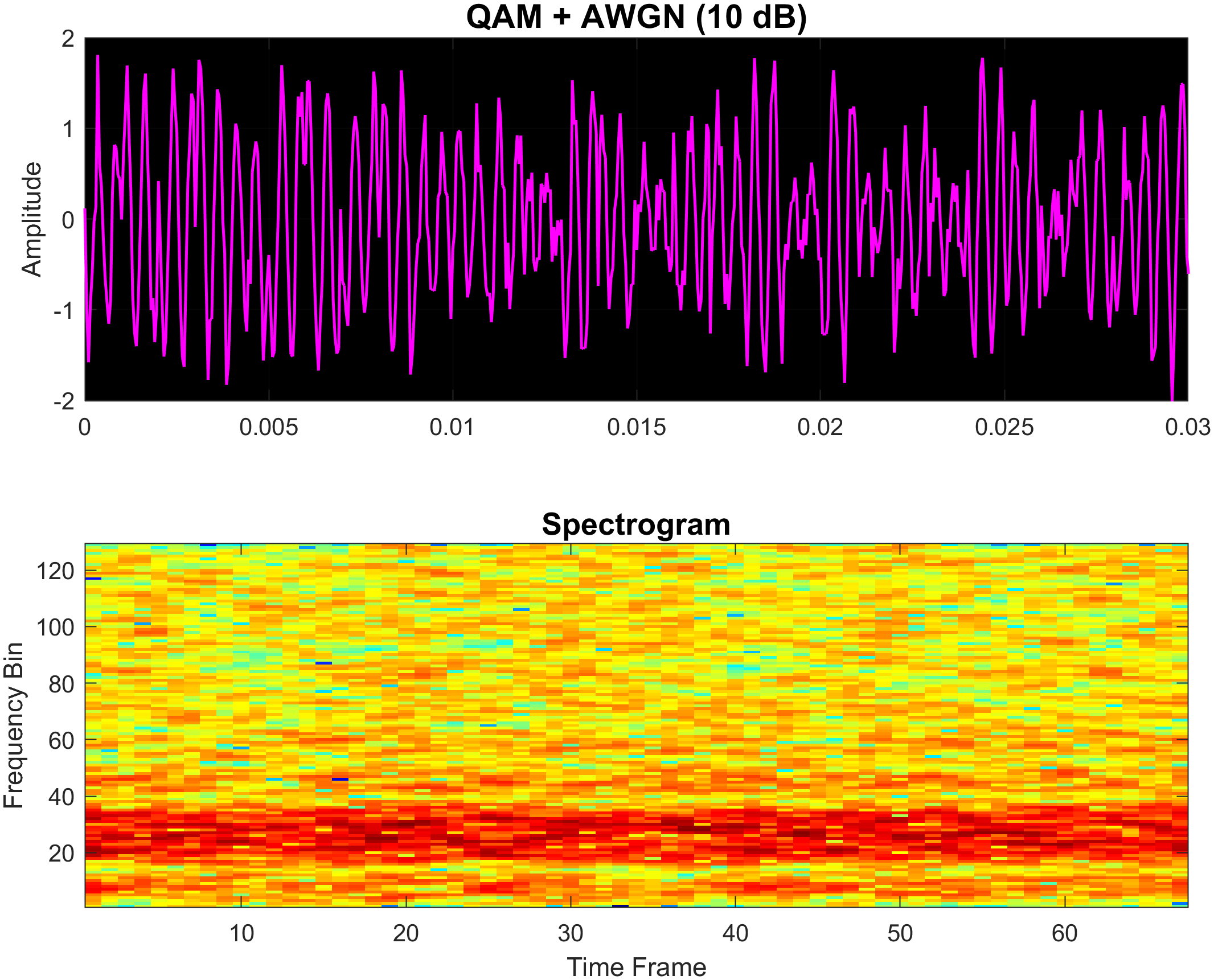


Figure 3.14 Same modulation as figure 3.13 but with 10dB SNR added

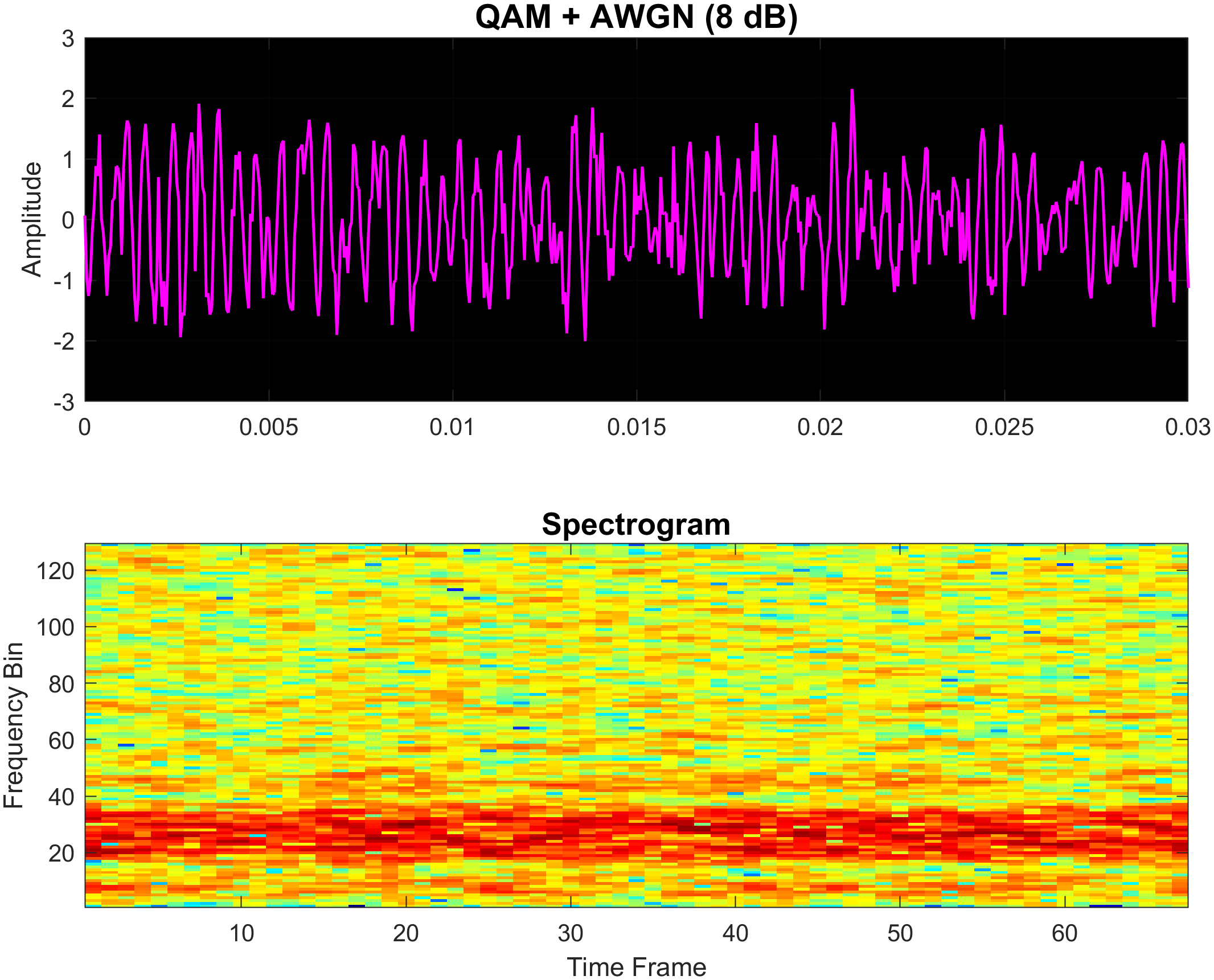


Figure 3.15 Same modulation shown in figure 3.13 with 8dB SNR

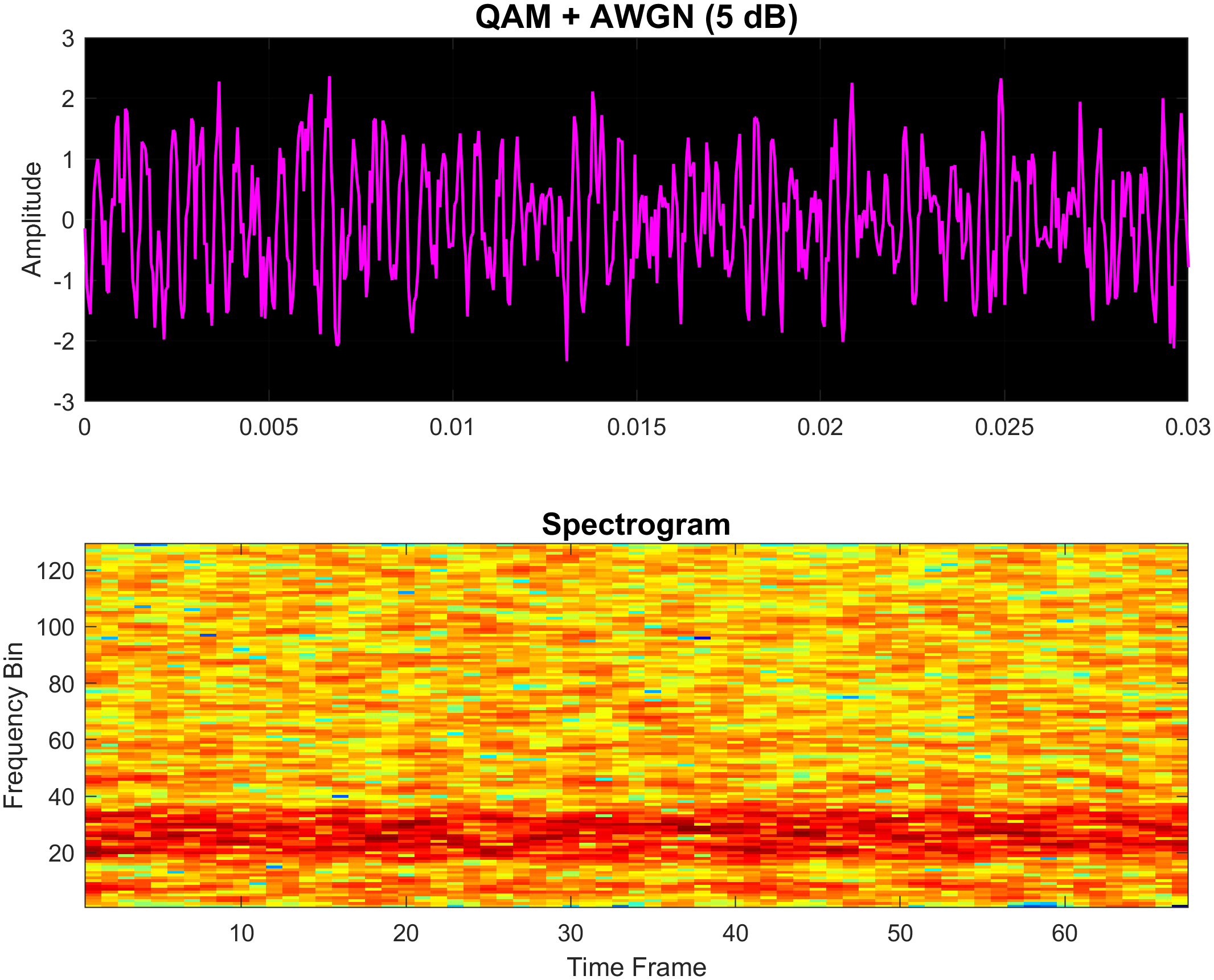


Figure 3.16 Same modulation shown in figure 3.13 with 5dB SNR

* + 1. **Chirp Spread Spectrum (CSS)**

Chirp Spread Spectrum (CSS) modulation encodes digital data by modulating the frequency of a chirp signal, which sweeps linearly over a frequency range. This technique is robust against noise and multipath interference, making it useful for low-SNR environments. The custom MATLAB function chirpModulate() implements M-ary chirp modulation and accepts:

* A binary input sequence (input\_bits)
* Sampling frequency (Fs)
* Carrier frequency (f\_c)
* Total signal duration (T)
* Modulation order (M), which must be a power of 2

The bitstream is divided into symbols, each representing bits. Symbols are mapped to unique frequency offsets within a normalized [0,1] range. For each symbol, a linear chirp signal is generated over the symbol duration, with its starting frequency determined by the symbol index. The signal is constructed by assigning each chirp to its respective time interval, resulting in a continuous waveform where frequency sweeps vary per symbol.

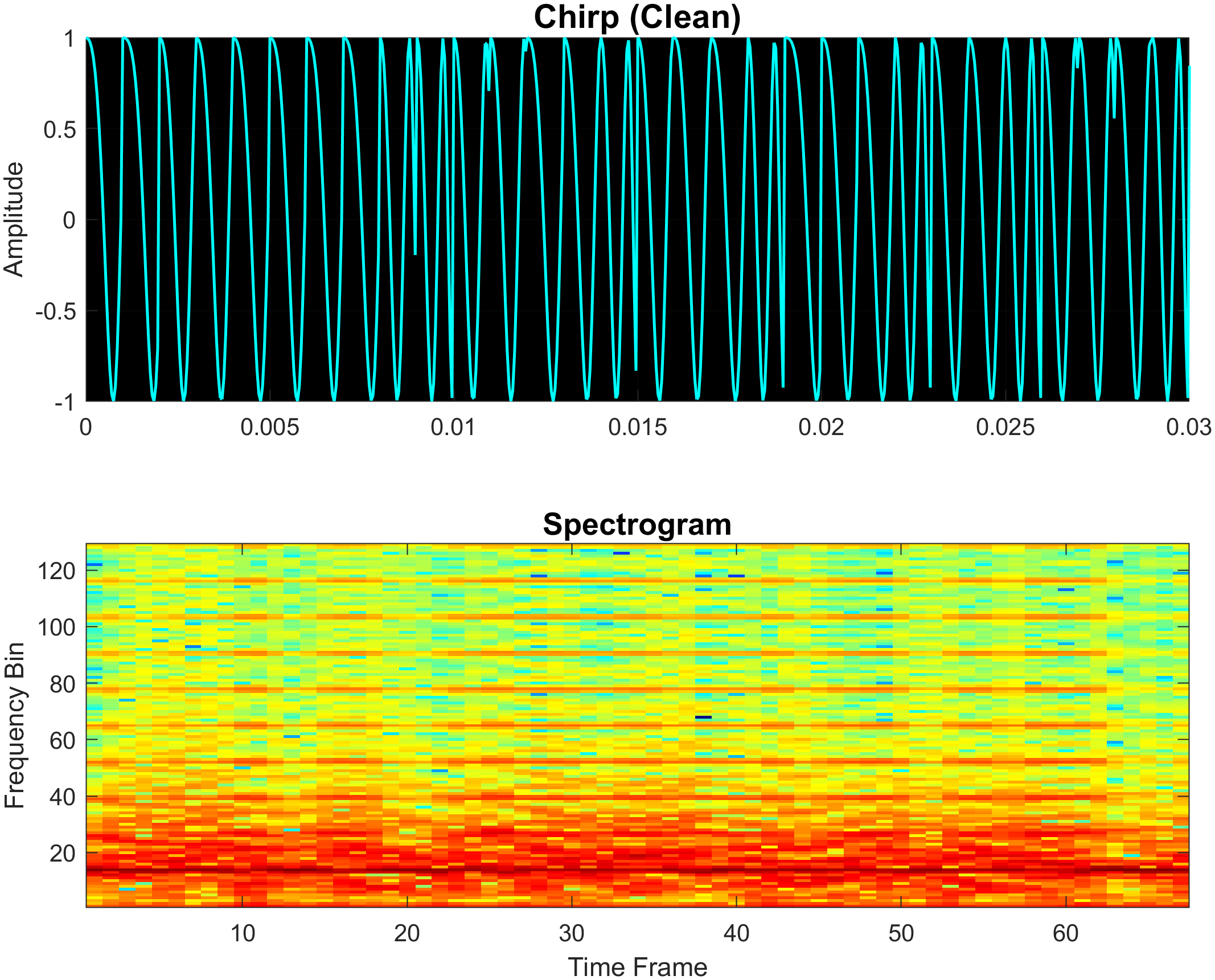


Figure 3.17 Shows an example of a 32-level Chirp modulation generated in matlab

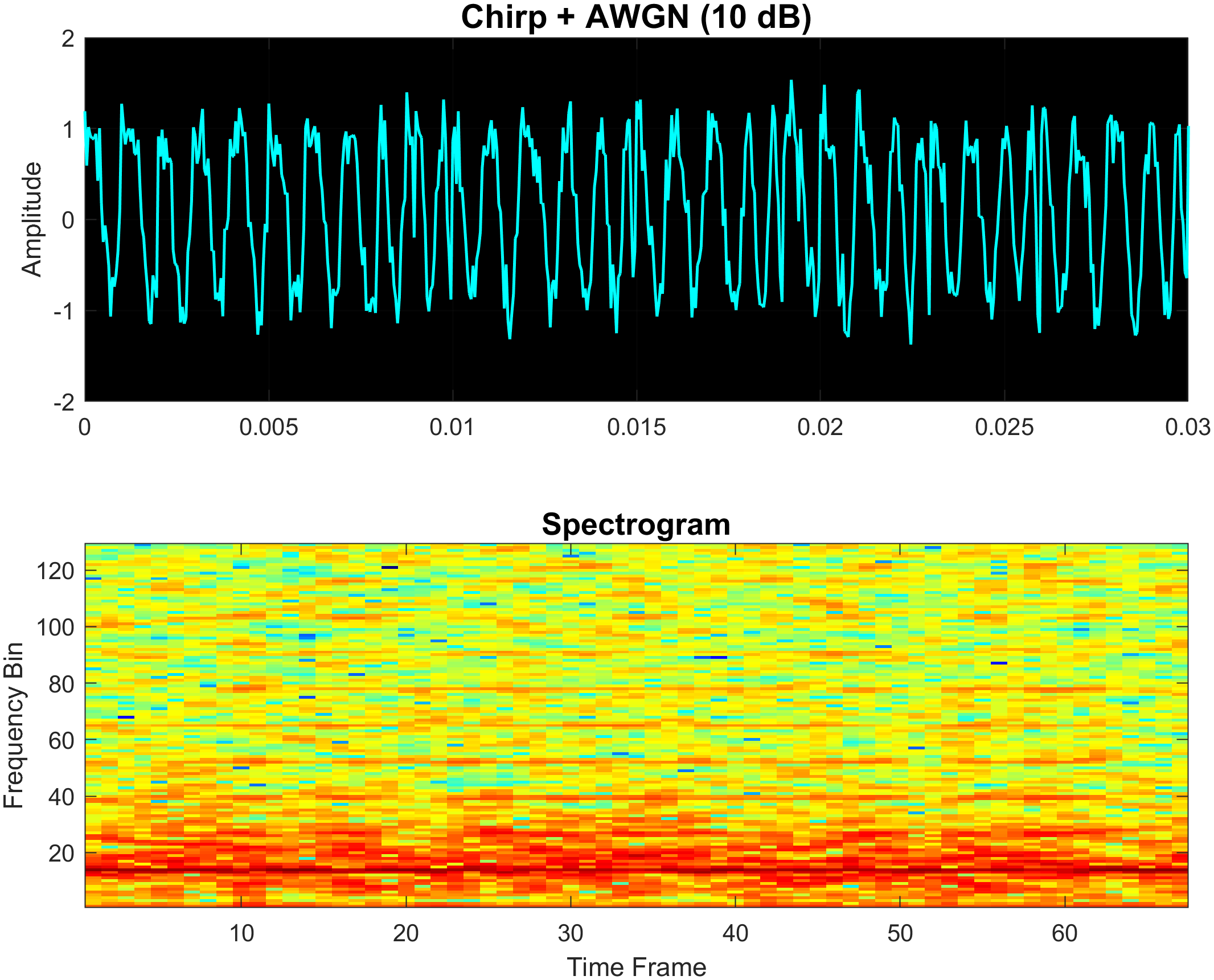


Figure 3.18 Same modulation as figure 3.17 but with 10dB SNR added

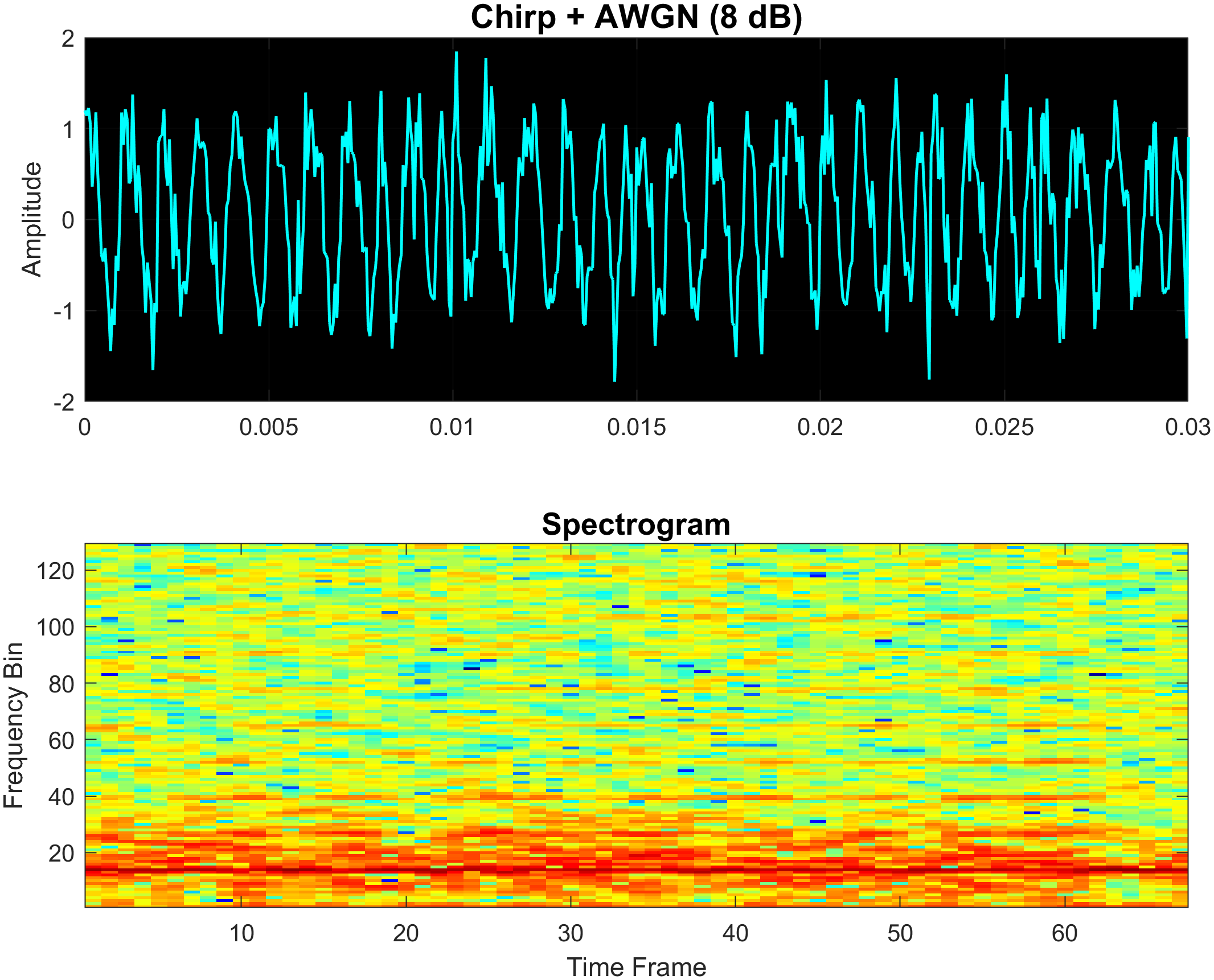


Figure 3.19 Same modulation shown in figure 3.17 with 8dB SNR

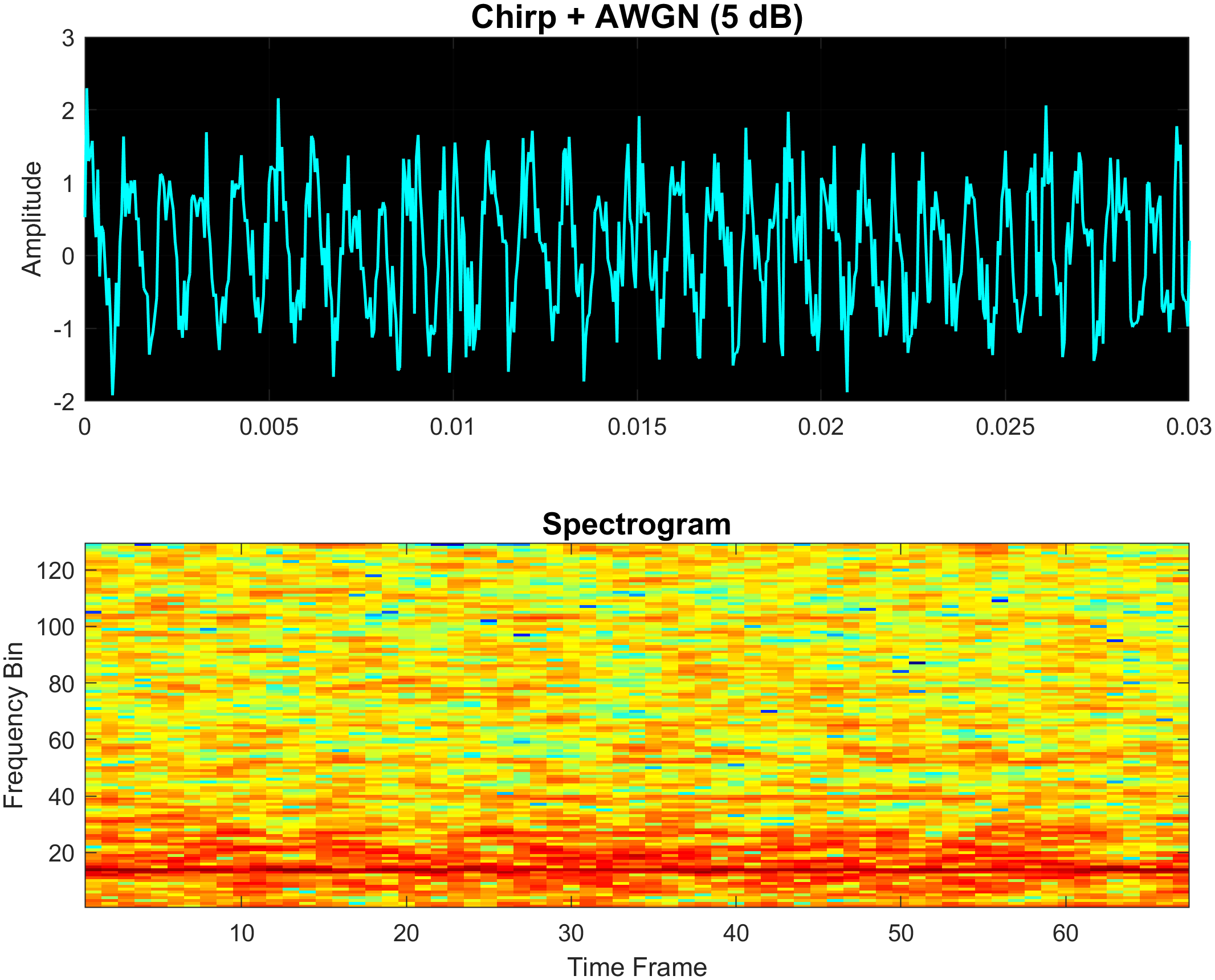


Figure 3.20 Same modulation shown in figure 3.17 with 5dB SNR

* 1. **Dataset Generation**

To enable the training and evaluation of modulation classification models, a large-scale dataset of spectrogram images was generated. The dataset captures a diverse set of modulation schemes under varying signal-to-noise ratio (SNR) conditions. This section outlines the methodology used to construct the dataset, including the modulation configurations, signal synthesis, noise application, spectrogram conversion, and final storage.

* + 1. **Modulation and Parameter Configuration**

The dataset includes six modulation types: ASK, FSK, PSK, QAM, Chirp, and DQPSK. Each modulation type is generated for multiple modulation orders (𝑀), which define the number of unique symbols. The modulation orders used were:

* ASK, FSK, PSK, Chirp: 𝑀=2, 4, 8, 16, 32, 64
* QAM: 𝑀=4, 16, 32, 64
* DQPSK: 𝑀=4

Each combination of modulation type and order was simulated across three SNR levels: clean (∞ dB), 10 dB, and 8 dB, to reflect different channel conditions. A total of 2,000 samples were generated per combination. Key signal parameters included:

* Sampling frequency: 20,000 Hz
* Symbol rate: 1,000 symbols/sec
* Carrier frequency (where applicable): 2,000 Hz
* Signal length: 8,000 samples (padded or truncated as needed)
* Output image size: 128 × 128 pixels
  + 1. **Signal Generation and Modulation**

For each sample:

* A random binary bitstream of 1,000 bits was generated.
* Bits were grouped into symbols based on the modulation order.
* A corresponding modulation function was called to generate the time-domain signal. These functions were previously implemented (see Sections 3.2.1–3.2.5).
* If the selected SNR was not infinite, Additive White Gaussian Noise (AWGN) was applied to the signal using MATLAB’s awgn() function.
  + 1. **Spectrogram Conversion**

Each time-domain signal was converted into a spectrogram, a 2D time-frequency representation commonly used for visual classification. MATLAB’s spectrogram() function was used with the following parameters:

* Window length: 256
* Overlap: 200
* FFT length: 256

The resulting power spectral density matrix was:

* Converted to logarithmic scale (dB),
* Resized to 128 × 128 pixels,
* Normalized using mat2gray() to scale the image data between 0 and 1.

This transformation ensures consistent image dimensions for input to machine learning models.

* + 1. **Visual Characteristics of Spectrograms**

The spectrogram images exhibit visually distinctive features for each modulation type.

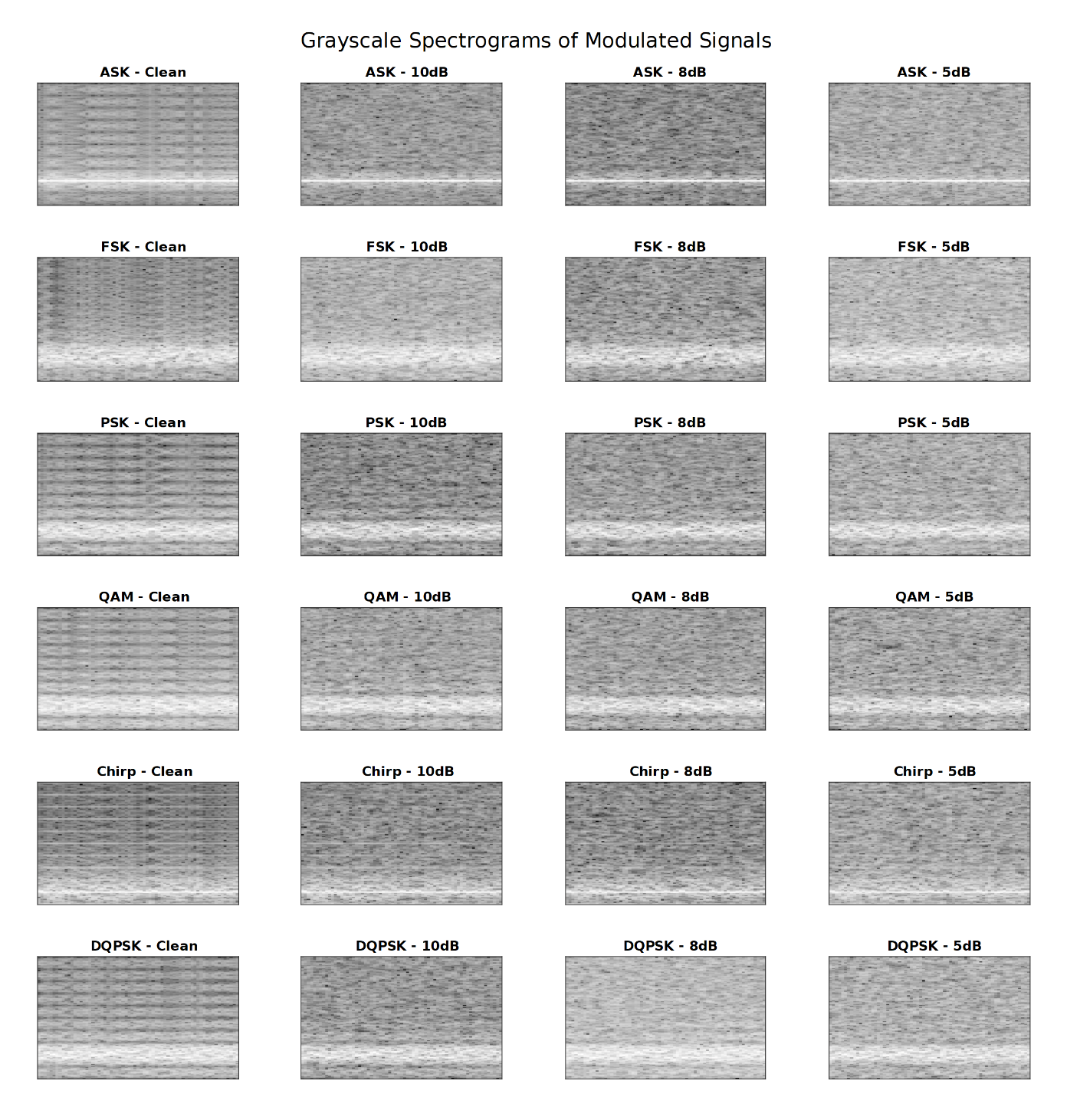


Figure 3.21 Sample of generated and stored modulations spectrograms

* + 1. **Labeling and Storage**

Each spectrogram was labeled using a format combining the modulation type and order (e.g., PSK\_16). All generated samples were stored in a 4D array, and labels were stored in a string array.

After generation, the arrays were trimmed to remove any unused preallocated space. The final dataset was saved in MATLAB’s .mat format (version 7.3) as spectrogram\_modulation\_dataset\_multiSNR.mat.

* 1. **Data Preprocessing**

Once the spectrogram dataset was generated and stored, a series of preprocessing steps were applied to prepare the data for supervised learning. These steps ensured that the data was in the correct format, appropriately normalized, and split into training and validation subsets for effective model evaluation.

* + 1. **Loading and Formatting**

The dataset was loaded from the .mat file generated in Section 3.3, containing two primary variables:

* spectrograms: a 4D array of size 128 × 128 × 1 × N, where N is the number of samples.
* labels: a string array of modulation labels (e.g., FSK\_8, QAM\_64).

The string labels were converted into categorical format using MATLAB’s categorical() function. This transformation is essential for classification tasks, allowing the training algorithm to treat the labels as discrete classes.

* + 1. **Data Splitting**

The dataset was split into training and validation subsets using stratified random sampling, ensuring that all modulation classes were proportionally represented in both sets. The cvpartition() function with a holdout ratio of 20% was used to allocate:

* 80% for training
* 20% for validation

This split is commonly adopted to balance training volume with effective evaluation, and it ensures that the model is validated on unseen data during each epoch.

* + 1. **Normalization and Input Configuration**

Each spectrogram image was already normalized to the [0, 1] range during the generation phase (Section 3.3.3). At the input layer of the convolutional neural network (CNN), the data was further zero-centered using the 'zerocenter' normalization option in MATLAB’s imageInputLayer. This helps standardize input distributions and improve learning stability. The input size for the network was set to 128 × 128 × 1, consistent with the dimensions of the spectrogram images.

* + 1. **Class and Label Overview**

The number of output classes was determined dynamically from the training labels using:

numClasses = numel(categories(YTrain));

This ensured compatibility with various configurations and allowed the same model architecture to support all modulation classes defined in the dataset.

* 1. **Tools and Software Environment**

All modulation simulations, dataset generation, and preprocessing tasks were performed using MATLAB, a high-level programming environment widely used in signal processing and engineering research. The following tools and configurations were used throughout the project:

* + 1. **Software**
* MATLAB Version: R2023b
* Toolboxes:
  + Signal Processing Toolbox (for spectrogram computation and signal transformations)
  + Deep Learning Toolbox (for model definition, training, and evaluation)
* Operating System: Windows 10 (64-bit)
  + 1. **Hardware**
* Processor: Ryzen 5 5600x 6 Core
* RAM: 32 GB
* GPU: NVIDIA GTX 1070 8GBs VRam, used for training acceleration via CUDA
* Storage: SSD with at least 100 GB of available space during generation and training phases
  + 1. **Reproducibility and Script Structure**
* askMModulate.m, fskMModulate.m, pskMModulate.m, etc. – Custom functions for modulation
* generateSpectrogramDataset.m – Handles dataset creation, noise addition, spectrogram conversion, and saving
* trainCNN.m – Loads the dataset, performs data preprocessing, and trains the classification model

**Chapter 4**

**Implementation**

This chapter presents the implementation of the deep learning model used for automatic modulation classification. It covers the model architecture, training strategy, evaluation setup, and validation process. The approach leverages supervised learning techniques using spectrogram representations of modulated signals as input.

* 1. **Data Preprocessing**

The model was built using **a Convolutional Neural Network (CNN)** architecture, chosen for its proven success in image-based pattern recognition. The spectrograms, previously generated from time-domain signals (Chapter 3), were treated as grayscale images of size 128 × 128 × 1. The CNN processes these spectrograms to learn temporal-frequency features that distinguish between modulation types and orders, even under varying noise levels.

The convolutional neural network used in this project was designed to balance classification performance with computational practicality. While deeper and more complex networks may improve accuracy marginally, they tend to require significantly more resources and training time—an issue encountered frequently during this project due to memory limitations and prolonged training cycles. The selected architecture contains three convolutional blocks, each increasing in depth (16, 32, 64 filters respectively). This progression allows the model to learn hierarchical features:

* Lower layers capture basic frequency patterns in spectrograms (edges, transitions),
* Higher layers recognize modulation-specific textures and time-frequency structures.

Adding more layers beyond this did not result in significantly better performance in early tests, but increased VRAM usage and training time noticeably. To prevent overfitting and improve generalization:

* A dropout layer was added after the fully connected layer, with a 50% drop rate.
* Batch normalization was used after each convolution to stabilize training and allow for faster convergence.

All convolution layers use small 3×3 kernels with 'same' padding, maintaining spatial dimensions while minimizing the number of parameters.

* 1. **Model Architecture**

The implemented CNN architecture includes three convolutional blocks followed by a fully connected layer. Each block consists of a convolutional layer, batch normalization, a ReLU activation, and max pooling to reduce spatial dimensions while preserving important features.

**Architecture Summary:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Description** | **Parameters** | **Output Size** |
| 1 | Input Layer | 128 × 128 × 1 | 128 × 128 × 1 |
| 2 | Conv2D + BN + ReLU + MaxPool | 16 filters, 3×3, stride 2 | 64 × 64 × 16 |
| 3 | Conv2D + BN + ReLU + MaxPool | 32 filters, 3×3, stride 2 | 32 × 32 × 32 |
| 4 | Conv2D + BN + ReLU + MaxPool | 64 filters, 3×3, stride 2 | 16 × 16 × 64 |
| 5 | Fully Connected + ReLU + Dropout | 128 units, 50% dropout | 128 |
| 6 | Output Layer (Fully Connected + Softmax) | numClasses units (multi-class) | 1 x numClasses |

Table 4.1 Summarizes the neural network architecture that’s used

This structure provided a good balance between:

* The ability to classify a wide range of modulation types and orders (including noisy samples),
* Keeping model size manageable (for deployment and GPU usage),
* Sufficient depth to learn modulation characteristics from spectrogram features.
  1. **Dataset Preparation and Input Pipeline**

The input data consists of spectrogram images generated from digitally modulated signals across six modulation types and multiple orders. Each sample was:

* Converted into a 128 × 128 grayscale image,
* Normalized and reshaped to match the model's input format.

To ensure balanced training and avoid class bias, the dataset was:

* Split into 80% training and 20% validation using cvpartition with stratified sampling,
* Shuffled at every epoch to improve generalization.

The number of classes (numClasses) was automatically determined based on the unique combinations of modulation type and order (e.g., 'ASK\_4', 'QAM\_16', etc.).

* 1. **Training Configuration**

Training was performed using the Adam optimizer, which provides adaptive learning rates for faster convergence. The model was trained for 100 epochs with a mini-batch size of 256, selected to utilize GPU resources efficiently.

**Training Options:**

* Optimizer: Adam
* Initial Learning Rate: 1 × 10⁻⁴
* Epochs: 50
* Mini-Batch Size: 256
* Validation Frequency: Every 30 iterations
* Execution Environment: GPU
* Gradient Thresholding: 1 (to prevent exploding gradients)
* Shuffling: Performed at every epoch
  1. **Training Process**

The model was trained on the spectrogram dataset described in Chapter 3. After preprocessing, the dataset was split into training (80%) and validation (20%) sets using stratified sampling to preserve class balance. The training process was monitored using MATLAB’s training plot, which tracked both loss and accuracy metrics over epochs. The trained model was saved to disk for later evaluation and deployment.

**Chapter 5**

**Results**

This chapter presents a comprehensive evaluation of the trained convolutional neural network (CNN) for modulation classification. It covers overall accuracy, class-wise performance metrics, SNR-wise behavior, and other relevant indicators that highlight the effectiveness and robustness of the proposed approach.

* 1. **Evaluation Metrics**

The To evaluate the real-time inference capability:

* The trained model was loaded separately.
* Incoming signals were generated for known modulation schemes and orders (e.g., 4-QAM).
* For each signal, the softmax output (i.e., probability distribution across all modulation classes) was recorded.
* Tests were repeated under SNR conditions: ∞ dB (noiseless), 10 dB, 8 dB, and 5 dB.

This setup reflects a realistic usage scenario in which signals are received and classified on-the-fly.

* 1. **Noiseless Predictions (∞ dB SNR)**

This table shows model predictions for clean (noiseless) input signals. Each row represents a test case, with the true class and the top predicted probabilities.

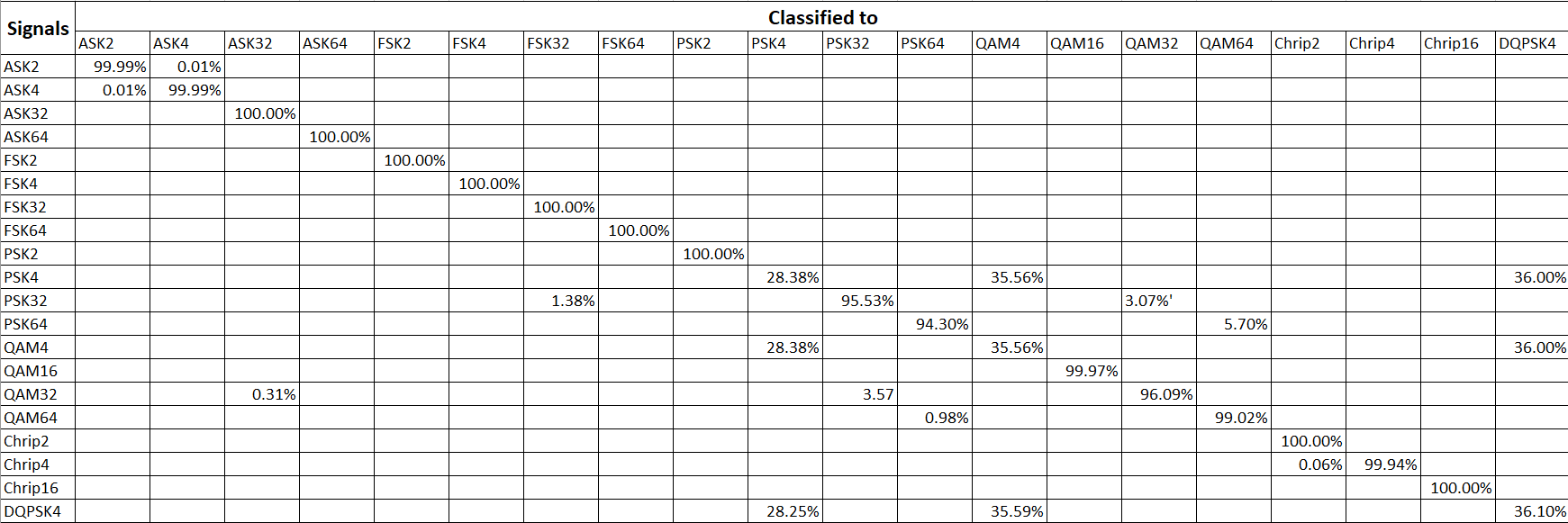


Table 5.1 Incoming noiseless signals predictions

* 1. **Predictions at 10, 8, 5 dB SNR**

At different SNR levels, signals begin to degrade slightly, but the model retains high performance. At As noise is introduced into the signal, the spectral clarity of modulated waveforms begins to deteriorate. Despite this, the model demonstrates strong robustness across all tested SNR levels, with only a gradual decrease in performance:

* Noiseless (∞ dB): 90% accuracy
* 10 dB SNR: 89% accuracy
* 8 dB SNR: 88% accuracy
* dB SNR: 84% accuracy

This gradual drop reflects the model’s ability to generalize across noise conditions, thanks to being trained on a multi-SNR dataset. However, at lower SNRs, specific patterns of misclassification become more noticeable. The most common confusion occurs between PSK, QAM, and DPSK. This is likely because their spectrograms often share similar symmetrical features—especially at higher modulation orders—making them harder to distinguish in the presence of noise. For example:

* A 4-QAM signal might be classified as 50% QAM, 50% PSK, especially under slight noise.
* DPSK occasionally gets misclassified as PSK, due to both schemes relying on phase changes and sharing closely aligned spectral structures.

Despite these challenges, the model maintains fairly strong confidence in its predictions, even under 5 dB SNR. The softmax outputs (prediction probabilities) reflect meaningful ambiguity, showing that the model isn't making random guesses but rather distributing probability across visually plausible classes.

This highlights one of the key strengths of using a spectrogram-based CNN approach: even when the prediction isn’t fully confident, the output probabilities can inform secondary decision layers or human-in-the-loop systems.

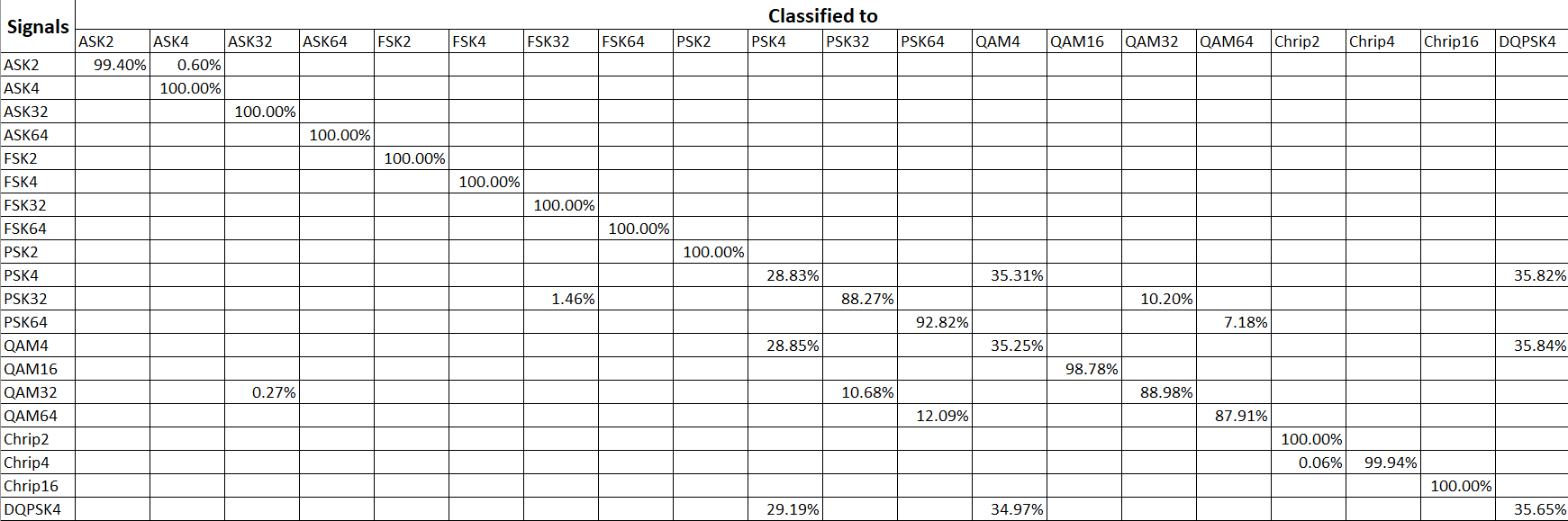


Table 5.2 Incoming 10dB SNR signals predictions



Table 5.3 Incoming 8dB SNR signals predictions

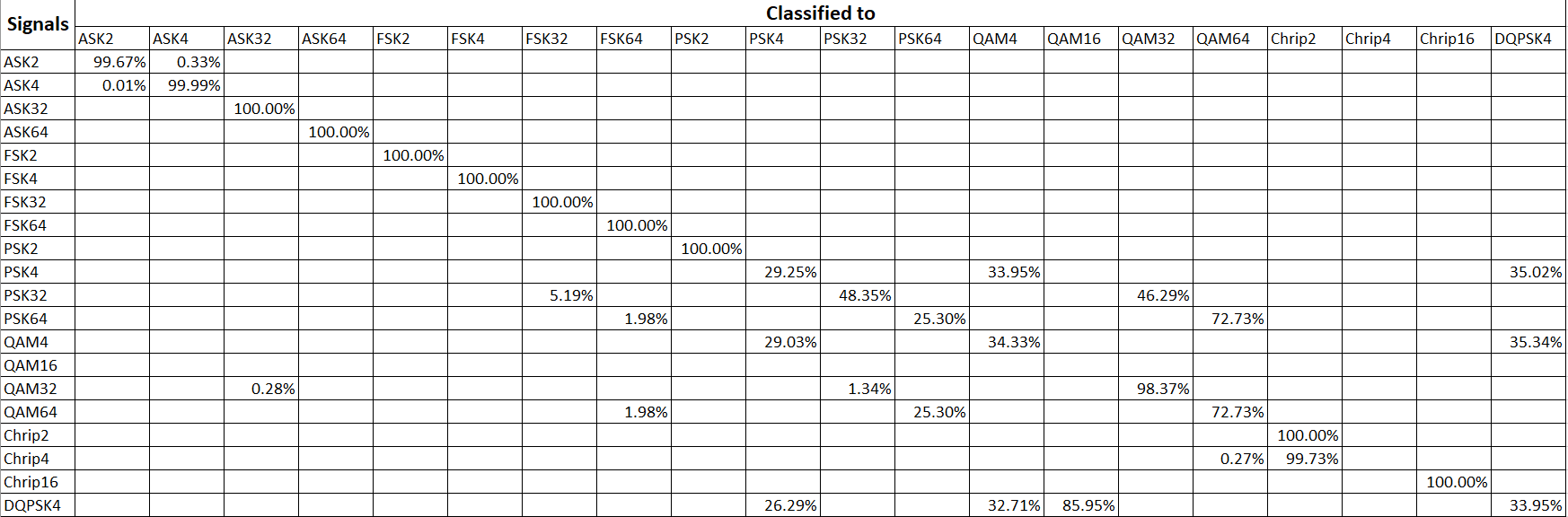


Table 5.4 Incoming 5dB SNR signals predictions

* 1. **Summary and Interpretation**
* The CNN performs exceptionally well in clean conditions, with 100% accuracy in several cases.
* At moderate SNRs (10 dB, 8 dB), the model remains reliable, though confidence margins tighten.
* At low SNR (5 dB), while raw accuracy drops, the model still outputs reasonable class probabilities, which can be useful in systems that implement confidence-aware decisions or post-filtering.
* Most misclassifications involve modulations with similar time-frequency features, such as QAM vs PSK or FSK vs ASK.

**Chapter 6**

**Conclusion and Future Work**

This chapter presents a comprehensive evaluation of the trained convolutional neural network (CNN) for modulation classification. It covers overall accuracy, class-wise performance metrics, SNR-wise behavior, and other relevant indicators that highlight the effectiveness and robustness of the proposed approach.

* 1. **Conclusion**

This project successfully explored and implemented a full pipeline for Automatic Modulation Classification (AMC) using spectrogram-based deep learning. The system simulated a variety of digital modulation schemes in MATLAB, including ASK, FSK, PSK, QAM, DPSK, and Chirp, across different modulation orders and signal-to-noise ratios.

Key achievements include:

* Generation of a large, diverse dataset of spectrograms across six modulation types, multiple M-orders, and varying SNR levels (∞, 10 dB, 8 dB, and 5 dB).
* Design and training of a custom CNN architecture optimized for modulation classification.
* Achieved strong classification performance, with:
  + 90% accuracy on clean signals
  + 84% accuracy at 5 dB SNR, showing robustness in challenging conditions
* Demonstrated real-world inference by testing the model on incoming, unseen signals and capturing confidence scores across all classes.

The use of spectrograms as input images proved to be highly effective. Visual transformations of modulated signals into time-frequency representations enabled the CNN to learn spatial features that distinguish between modulation types — even in noisy conditions. The model often exhibited high confidence in its decisions, and even when incorrect, the softmax outputs reflected reasonable class overlap, which can be useful in practical systems. Despite the project's success, several challenges were encountered:

* Dataset generation was memory- and storage-intensive.
* Training the model was computationally heavy, often pushing system memory to the limit.
* Similar modulation types (e.g., PSK vs. QAM, DPSK vs. PSK) remained a consistent source of confusion, especially under noise.
  1. **Future Work**

There are several directions in which this work can be expanded:

* Use a larger more diverse dataset to train a better model.
* Implement more complex modulation types like OFDM
* Implementing the AI Model in a real-world project
* Implementing a demodulation logic to bring back the original binary format for the signal.
* Model Optimization: Experiment with deeper or more efficient architectures (e.g., ResNet, MobileNet) to improve accuracy and reduce training time.
* Noise Robustness: Add more SNR levels, include real-world channel effects (e.g., fading, Doppler), or apply adversarial training.
* Feature Fusion: Combine spectrograms with other representations (e.g., IQ plots or raw time-domain waveforms) to give the model more information.
* Real-Time Deployment: Explore how the model can be integrated into SDR (Software Defined Radio) platforms or real-time communication systems.
* Confidence-Aware Classification: Use prediction probabilities to build systems that adapt behavior based on model certainty (e.g., fallback modes, human review).
  1. **Final Thoughts**

This project bridges signal processing and artificial intelligence, showing how deep learning can transform traditional communication system tasks. It not only automates modulation classification but does so in a way that is scalable, interpretable, and highly adaptable to future improvements.

As communication systems continue to grow in complexity, AI-driven modulation classification will likely play a key role in enabling autonomous, intelligent radios — and this work takes a meaningful step toward that vision.

**References**

[1] John G. Proakis and Masoud Salehi, “Digital Communications” McGraw-Hill, pp. 3, 2008

[2] Stuart J. Russell and Peter Norvig, “Artificial Intelligence A Modern Approach” Pearson Education, pp. 670-671, 2022

[3] Stuart J. Russell and Peter Norvig, “Artificial Intelligence A Modern Approach” Pearson Education, pp. 672, 2022

[4] Christopher M. Bishop, “Pattern Recognition and Machine Learning” Springer, pp. 266. 2006

[5] Saad S. Hreshee, “Automatic recognition of the digital modulation types using

the artificial neural networks” IJECE pp. 5871~5882 2020

[6] Abdulnasir Hossen, Fakhri Al-Wadahi & Joseph A. Jervase, “Classification of Modulation Signals Using Statistical Signal Characterization and Artificial Neural Networks” Elsevier Ltd 2006

[7] John A. Snoap and Dimitrie C. Popescu “Deep Learning-Based Automatic Modulation Classification Using Robust CNN Architecture for Cognitive Radio Networks” Old Dominion University 2023

[8] A. Fehske, J. Gaeddert and J. H. Reed “A New Approach to Signal Classification Using Spectral Correlation and Neural Networks” Virginia Polytechnic Institute and State University Mobile and Portable Radio Research Group 2005

**الخلاصة**

يهدف هذا المشروع إلى تطوير نموذج ذكاء اصطناعي قادر على تصنيف إشارات الاتصالات الرقمية بناءً على نمط التضمين المستخدم، وذلك باستخدام تقنيات التعلم الآلي وتحويل الإشارات إلى تمثيلات طيفية.

تم إنشاء مجموعة بيانات واسعة من إشارات التضمين المختلفة مثل تضمين السعة، وتضمين التردد، وتضمين الطور، والتضمين التفاضلي للطور، وتضمين السعة والطور المعقد، والإشارات الممسوحة تردديًا، باستخدام برنامج ماتلاب، مع أخذ مستويات مختلفة من نسبة الإشارة إلى الضوضاء في الاعتبار لمحاكاة ظروف بيئية متنوعة.

تم تحويل كل إشارة إلى صورة طيفية باستخدام تحويل فورييه قصير الزمن، ثم تم تدريب شبكة عصبية التفافية على هذه الصور لتتعلم كيفية التمييز بين أنماط التضمين المختلفة. أظهرت النتائج دقة عالية في التصنيف، حيث تجاوزت نسبة الدقة ٩٠٪ في حالة الإشارات الخالية من الضوضاء، وانخفضت تدريجيًا مع زيادة الضوضاء لتصل إلى ٨٤٪ عندما كانت نسبة الإشارة إلى الضوضاء ٥ ديسيبل. كما أظهرت النتائج أن النموذج قد يخلط أحيانًا بين أنماط تضمين الطور وتضمين السعة والطور المعقد والتضمين التفاضلي للطور، نظرًا للتشابه الطيفي بينها في بعض الحالات.

تُبيّن هذه الدراسة فعالية الدمج بين تقنيات معالجة الإشارات الرقمية والتعلم الآلي في تحسين أداء أنظمة تصنيف التضمين، وتفتح المجال أمام تطبيق الذكاء الاصطناعي في أنظمة الاتصالات الحديثة..

|  |  |
| --- | --- |
|  | **جامعة النهرين**  **كلية هندسة المعلومات**  **قسم هندسة المعلومات والاتصالات** |

**تصنيف إشارات التضمين الرقمي بالاعتماد على الذكاء الاصطناعي**

**مشروع السنة المنتهية**

**مقدم الى كلية هندسة المعلومات / قسم هندسة المعلومات والاتصالات كجزء من متطلبات نيل شهادة**

**البكالوريوس**

**في**

**هندسة المعلومات والاتصالات**

**من قبل**

**علي فيصل عازي**

**باشراف**

**د. أسماء حميد مجيد**

**شوال 1446**

**أبريل 2025**