BULGARIAN ACADEMY OF SCIENCES
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS • 2023 • Vol. **80**, pp. 23-32
p-ISSN: 2738-7356; e-ISSN: 2738-7364
https://doi.org/10.7546/PECR.80.23.03

Wireless Data Transmission and Neural Networks: Using Amplitude Modulation and Demodulation

Krasimir Markov

Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Acad. Georgi Bonchev Str., bl. 2, 1113 Sofia, Bulgaria
E-mail: krasikasi@abv.bg

Abstract: This article presents the implementation of amplitude modulation (AM) and demodulation using a neural network. The signal parameters are initially defined, using the sampling frequency, the number of samples, the amplitudes of the carrier and modulating signals, and the carrier frequency. Then, a bit string to be modulated is provided. Using the defined parameters and the bit string, a modulating signal is generated for each symbol. Depending on the value of the bit (0 or 1), a modulating signal with amplitude or zero value is generated. The modulating signal is then multiplied by the carrier wave, and the resulting modulated signal is outputted. After modulation, white Gaussian noise is generated and added to the modulated signal. The signal-to-noise ratio (SNR) is used for this purpose, defined in decibels. The result is a noisy modulated signal. Next, a neural network with hidden layers and activation functions is constructed. The neural network is trained using the noisy amplitude-modulated signal and the clean amplitude-modulated signal, with the latter used as the target data for training. Upon completing the training, the neural network is used to predict the modulated signal. Demodulation of the predicted signal is performed, and the bits are decoded. The article presents a comprehensive process of amplitude modulation and demodulation using a neural network in the MATLAB programming environment. The presented results and the decoded output of the network demonstrate the effectiveness of the proposed implementation.

Keywords: *Amplitude Modulation (AM), Neural Network (NN), Signal-to-Noise Ratio (SNR), Prediction, Decoding, MATLAB*.

1. Introduction

Wireless data transmission is an important technology that allows us to communicate and exchange information without the need for physical connections. One widely used technique for wireless data transmission is amplitude modulation. In combination with neural networks, amplitude modulation can help us transmit and decode data with high accuracy, even in conditions of noise and interference.

Amplitude Modulation (AM) is a technique for transmitting information that utilizes changes in the amplitude of the carrier signal to convey data. In the process of AM, the data is embedded into the carrier signal by varying its amplitude according to the data values. At the receiver's end, the reverse process occurs – decoding the modulated signal and recovering the original data. With the use of neural networks, amplitude modulation can be optimized and improved. Neural networks are mathematical models inspired by the workings of the human brain, capable of "learning" to recognize and process complex patterned data. When combined with amplitude modulation, neural networks can be trained to accurately recognize and decode data, even in the presence of noise and interference. This combination of AM modulation and neural networks has a wide range of applications. For example, in wireless communications such as mobile phones and Wi-Fi, RFID networks. AM modulation is used for transmitting voice, data, and other types of information. The use of neural networks in this context can enhance the quality of data transmission and reception, particularly under conditions of noise and interference.

2. Amplitude Modulation and Demodulation

Amplitude modulation (AM) is a technique for transmitting information where the amplitude of the carrier signal is varied according to the data we want to transmit [3, 4, 6]. Modulation is a process of changing the parameters of a signal, called the carrying signal, under the influence of another one, called the modulating signal [8, 21]. The carrying signal has a much higher frequency than the modulating one and is usually changed by the sinusoidal or cosine theorem. Depending on the parameter that is being modulated we can distinguish amplitude modulation (AM), frequency modulation (FM) and phase modulation (PM). The last two are more well-known with their joint name angular modulation. When the carrier signal has rectangular form, an impulse modulation occurs. As with the other type, this can also be split into amplitude impulse modulation (AIM), frequency impulse modulation (FIM), phase impulse modulation (PIM) and wide impulse modulation (WIM). When the carrier signal is harmonic, and the modulating one has a rectangular form, the terms of manipulation are similar – amplitude, frequency and phase.

The mathematical model of an amplitude modulated signal follows:

$$A_{am}(t) = A.(1 + m.cos(2\pi.f_0.t + \Psi_o).\cos(2\pi.f.t + \Psi) \quad (1)$$

where $a(t)$ is carrying signal:

$$a_o(t) = A.cos(2\pi.f.t + \Psi) \quad (2)$$

where $a_0(t)$ is modulating signal:

$$a_o(t) = A.cos_0(2.\pi.f_0.t + \Psi_0) \quad (3)$$

The frequency of the modulating signal is much smaller than the one of the carrying signal ($\omega_0 \ll \omega$).

The following relation represents the modulation coefficient and represents a measure of the depth of a modulation:

$$m = A_0 / A \quad (4)$$

For a successful transfer of the message sent, it (the coefficient) has to be less than 1 ($m < 1$) . If $m > 1$ the so called over modulation happens and on restoration of the initial signal at the receiver site, warping of the signal occurs. This leads to loss of information. In our script (Appendix 1), we use AM for transmitting binary data. When the bit is "1," we modulate the carrier signal with a specific amplitude, and when the bit is "0," the amplitude of the modulating signal is zero. Once we modulate the data, we generate a modulated signal that contains the information we want to transmit.

This modulated signal can be subjected to interference and noise during transmission. To recover the information from the modulated signal, we use demodulation. The demodulation is the process of recovering the original signal (modulating signal) from the modulated signal after it has been transmitted through the channel and received by the receiver. This is done to retrieve the data or information that was modulated onto the carrier signal before transmission.

The Nyquist-Shannon sampling theorem, also known as the Shannon sampling theorem, can be expressed as follows: If a function with a continuous spectrum does not contain frequency components higher than f_max, then that function can be fully and accurately reconstructed from its samples taken at a rate of at least twice the frequency f_max. This means that in order to avoid information loss during the sampling of a signal with a maximum frequency f_max, a sampling rate of at least twice the frequency f_max must be used.

In the script (Appendix 1), we employ a neural network for demodulation. The neural network is trained to predict the clean modulated signal using the noisy modulated signal as input. This allows us to recover the information and decode the transmitted data.

3. Training the Neural Network

In the script (Appendix 1), we use a neural network with two hidden layers to decode the transmitted data. The network is trained using the backpropagation algorithm. Training is performed by feeding the noisy modulated signal as input and the transmitted clean modulated signal as target data [8, 14-21].

We optimize the parameters of the network by adjusting the number of hidden neurons and the learning rate. Our goal is to achieve high accuracy in decoding the transmitted data. The neural network used in the script (Appendix 1) is known as a feedforward neural network. It is organized in sequential layers of neurons that propagate from the input layer through the hidden layers to the output layer.

Backpropagation is an algorithm used to train multilayer feedforward neural networks. The goal of the algorithm is to minimize the error between the network's predictions and the target outputs by adjusting the weights of the connections between neurons.

Here is how the backpropagation algorithm works:

- Weight initialization: Initially, the weights of the connections between neurons are initialized with random values.
- Forward calculation: An input signal is propagated through the network from the input layer to the output layer. The signal propagation in the network is computed by multiplying the input signal with the weights of the connections and applying an activation function to the result. This process continues until reaching the output layer.
- Error calculation: After obtaining the output predictions, the error is calculated by comparing the predictions with the target outputs.
- Backward propagation of error: The error is propagated backward from the output layer to the hidden layers. For each connection weight, the gradient is computed, indicating how a change in that weight would affect the error. The gradients are calculated using the chain rule and are used to update the weights.
- Weight update: Based on the computed gradients, the weights of the connections between neurons are updated. Typically, a gradient descent method is used, where the weights are adjusted with a certain learning rate that controls the magnitude of the change.
- Iteration of the process: Steps 2 to 5 are repeated for several epochs or until the desired accuracy of predictions is achieved.

This process of backpropagation allows the network to adapt and learn the relationships between the input data and the target outputs by adjusting the weights to minimize the error. With more training iterations, the network becomes better at predicting new input data.

4. Decoding the Transmitted Data

After training the neural network, we use it to decode the transmitted data. In our script (Appendix 1, 2), we divide the transmitted signal into segments and calculate the average amplitude of each segment [1, 2, 3, 9-13]. If the average amplitude exceeds half of the amplitude of the carrier signal, we assign "1" for decoding. Otherwise, we assign "0". This allows us to reconstruct the binary data that has been transmitted.

7. Conclusion

The use of amplitude modulation and demodulation in combination with neural networks provides a powerful tool for wireless data transmission and decoding. This technique allows us to transmit and recover data with high accuracy even in noisy and distorted conditions. By utilizing neural networks, we can train the system to handle various data transmission scenarios and achieve better results.
This article presented the fundamental idea and implementation of wireless data transmission and decoding using neural networks. In real-world applications, additional optimizations and improvements can be performed based on specific requirements and conditions. It is possible to convert the MATLAB script into VHDL or Verilog for implementation on programmable logic integrated circuits (FPGAs). I hope this article provides useful information and inspiration for further research and applications in the field of wireless data transmission and neural networks.

Acknowledgments

This work was supported by the Bulgarian Ministry of Education and Science under the National Research Programme „Smart crop production" approved by Decision of the Ministry Council No 866/26.11.2020 (No D01-65/19.03.2021).

References

1. Oppenheim, A. V., Willsky, A. S., Young, I. T.: Signals and Systems. Sofia, Technical Press. (1993).
2. Penev, N.: Contemporary Communication Systems and Technologies. Sofia, V. Nedkov. (2008).
3. Ifeachor, E., Jervis, B.: Digital Signal Processing: A Practical Approach. Edinburgh Gate: Pearson Education. (2002).
4. MATLAB – Communications System Toolbox-User's Guide. Retrieved from www.mathworks.com.
5. MATLAB – Control System Toolbox-User's Guide. Retrieved from www.mathworks.com.
6. MATLAB – Signal Processing Toolbox-User's Guide. Retrieved from www.mathworks.com.
7. MATLAB – Symbolic Math Toolbox-User's Guide. Retrieved from www.mathworks.com.
8. MATLAB – Neural Network Toolbox -User's Guide. Retrieved from www.mathworks.com.
9. Doukovska, L.: Hough target detectors with small values of signal-to-noise ratio. NATO Advanced Study Institute "Unexploded Ordnance Detection and Mitigation", Il Ciocco, Italy, (2008).

10. Doukovska, L.: Application of mathematical transform in detection algorithms. In: Proc. of the First International Symposium on Business Modelling and Software Design – BMSD'11, Sofia, Bulgaria, pp. 161–167, DOI: 10.5220/0004459801610167, (2011).
11. Doukovska, L.: Constant false alarm rate detectors in intensive noise environment conditions. Cybernetics and Information Technologies 10 (3), pp. 31– 48, (2010).
12. Doukovska, L., Angelova, D.: Comparative analysis of two techniques for moving target velocity estimation. In: Proc. of the 7-th European Radar Conference – EuRAD'10, Paris, France, pp. 431–434, (2010).
13. Kamenov, D., Sgurev, V., Doukovska, L.: Controlling Multiagent System for Sensor Networks - Software Architecture Modelling and Diagnostics. Proc. of Signal Processing Symposium - SPS'11, Jachranka, Poland, IEEEXplore, CD Proc. (2011).
14. Toskova, A., Toskov, B., Doukovska, L., Daskalov, B., Radeva, I.: Neural Networks in the Intelligent Educational Space. Proc. of the IEEE International Workshop on Advances in Neural Networks and Applications – ANNA 2018, VDE VERLAG GMBH, Berlin, IEEEXplore, 2018, ISBN:978-3-8007-4756-6, 107-112, (2018).
15. Shahpazov, V., Doukovska, L., Karastoyanov, D.: Artificial Intelligence Neural NetworksApplications in Forecasting Financial Markets and Stock Prices. Proc. of the International Symposium on Business Modeling and Software Design – BMSD'14, Luxembourg, Grand Duchy of Luxembourg, SCITEPRESS - Science and Technology Publications, 2014, ISBN:978-989-758-032-1, DOI:10.5220/0005427202820288, 282-288, (2014).
16. Shahpazov, V., Velev, V., Doukovska, L: Design and Application of Artificial Neural Networks for Predicting the Values of Indexes on the Bulgarian Stock Market. Proc. of the Signal Processing Symposium – SPS'13, Jachranka Village, Poland, IEEEXplore, ISBN:978-1-4673-6319-8-13, CD Proc. (2013).
17. Shahpazov, V., Doukovska, L.: Forecasting financial markets with artificial intelligence. Proc. of the International Workshop on Advanced Control and Optimisation: Step Ahead – ACOSA'14, Bankya, Bulgaria, Prof. Marin Drinov Publishing House, ISSN:1314-4634, 67-74, (2014).
18. Terziyska, M., Doukovska, L.: Semi fuzzy neural networks, Part 1: Nonlinear system identification. Proc. of the International Workshop on Advanced Control and Optimisation: Step Ahead – ACOSA'14, 2014, Bankya, Bulgaria, Prof. Marin Drinov Publishing House, ISSN:1314-4634, 18-23, (2014).
19. Terziyska, M., Doukovska, L.: Semi fuzzy neural networks, Part 2: Predictive control, Proc. of the International Workshop on Advanced Control and Optimisation: Step Ahead – ACOSA'14, Bankya, Bulgaria, Prof. Marin Drinov Publishing House, ISSN:1314-4634, 24-28, (2014).
20. Hu, Y. H., Hwang, J. N.: Neural Network Signal Processing. CRC Press. (1999).
21. Lagnese, M. C.: Neural Networks with MATLAB. Pearson, (2003).

APPENDIX 1
MATLAB SCRIPT

```
% Signal Parameters
Fs = 100e6; % Sampling frequency
N = 1024; % Number of samples
dt = 1/Fs; % Time step
A1 = 5; % Carrier signal amplitude A
2 = 3.3; % Modulating signal amplitude
 fc = 13.56e6; % Carrier frequency
t = 0:dt:(N-1)*dt; % Time axis
carrier = A1cos(2pifct); % Carrier wave

% Modulating signal
bits = '01101101 01100001 01100111 01101110 01101001 01110100
'; % magnit
modulated_data = []; % Initialize empty array
bits = strrep(bits, ' ', ''); % Remove spaces from the string
% Generate modulating signal for each character
for i = 1:length(bits) if bits(i) == '1'
modulated_signal = A1 * cos(2pifc*t); % Modulating signal for
'1'
else
modulated_signal = zeros(size(t)); % Modulating signal for '0'
end modulated_
data = [modulated_data modulated_signal]; % Add modulating
signal to the array
end

% Amplitude modulation
carrier_replicated = repmat(carrier, 1, length(modulated_data)
/ length(carrier));
modulated_signal = carrier_replicated .* (1 + (A2/A1) *
modulated_data); % Formula
t = 0:dt:(length(modulated_signal)-1)*dt; % Update the length
of the t vector

% White Gaussian noise generation
SNR = 10; % Signal-to-noise ratio (dB)
noisy_signal = awgn(modulated_signal, SNR, 'measured');

% Generate data for the neural network
input_data = noisy_signal; % Noisy amplitude-modulated signal
target_data = modulated_signal; % Clean amplitude-modulated
signal (message)

% Create a neural network
hidden_units = 100; % Number of hidden neurons
net = feedforwardnet(hidden_units);
```

```
net.layers{1}.transferFcn = 'logsig'; % Activation function for
the first hidden layer net.layers{2}.transferFcn = 'tansig'; %
Activation function for the second hidden layer

% Set training parameters
net.trainParam.epochs = 100; % Number of epochs
net.trainParam.lr = 0.01; % Learning rate

% Train the neural network
net = train(net, input_data, target_data);
predicted_signal = sim(net, target_data);

% Demodulation
demodulated_signal = (predicted_signal ./ carrier_replicated -
1) * A1/A2;

% Decoding the network output
decoded_bits_predicted = ''; for i =
1:length(predicted_signal)/N
chunk_predicted = predicted_signal((i-1)N+1:iN);
average_amplitude_predicted = mean(abs(chunk_predicted));
 if average_amplitude_predicted > 0.5 * A1
 decoded_bits_predicted = [decoded_bits_predicted '1'];
else
decoded_bits_predicted = [decoded_bits_predicted '0'];
 end
end
```

APPENDIX 2

RESULTS

1. Decoding the network output:

```
decoded_message_predicted =
char(bin2dec(reshape(decoded_bits_predicted, 8, []).'));
disp('Decoded network output:');
disp(decoded_message_predicted);
 Output: Decoded network output: m a g n i t
The goal has been achieved. The modulated message has been
correctly recognized.
```

2. Time characteristics of the signals:

The carrier wave, the modulating signal and the modulated signal are shown on Fig. 1.

```
figure(1);
subplot(3, 1, 1);
plot(t, carrier_replicated);
```

```
title('Carrier Wave'); xlabel('Time'); ylabel('Amplitude');
 subplot(3, 1, 2);
 stem(t, modulated_data);
title('Modulating          Signal');          xlabel('Time');
ylabel('Amplitude');
subplot(3, 1, 3);
plot(t, modulated_signal);
 title('Modulated                                        Signal');
xlabel('Time');ylabel('Amplitude');
```

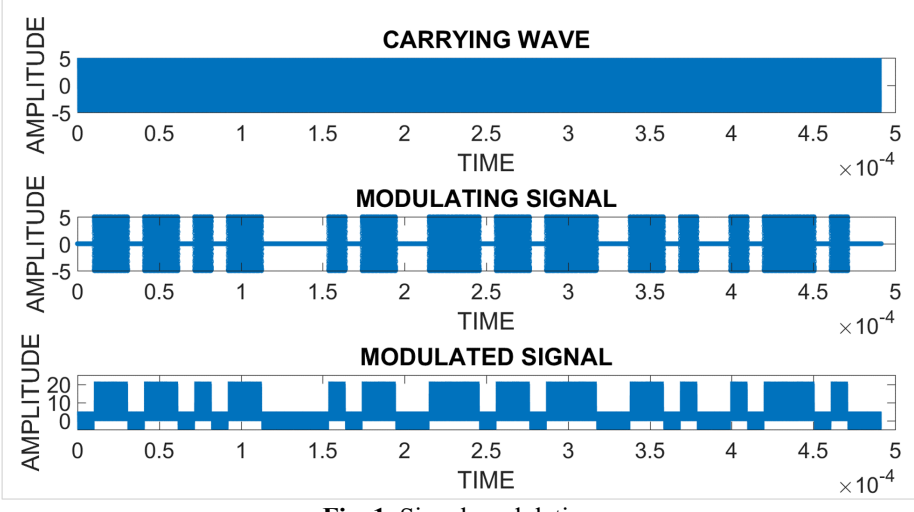

Fig. 1. Signal modulation

3. Time characteristics of the signals:

The real modulated signal and noisy modulated signal are shown on Fig. 2.

```
figure(2);
 subplot(2, 1, 1);
 plot(t, modulated_signal);
 title('Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
subplot(2, 1, 2);
 plot(t, noisy_signal);
 title('Noisy Modulated Signal');
xlabel('Time');
 ylabel('Amplitude');
```

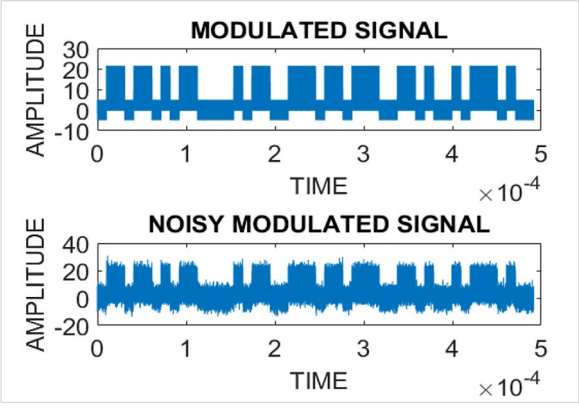

Fig. 2. Noise modulation

4. Time characteristics of the signals:

The results for real modulated signal, noisy modulated signal and output signal from a neural network are shown on Fig. 3.

```
figure(3);
 subplot(3, 1, 1);
plot(t, modulated_signal);
 title('Modulated          Signal');          xlabel('Time');
ylabel('Amplitude');
subplot(3, 1, 2);
plot(t, noisy_signal);
 title('Noisy     Modulated     Signal');     xlabel('Time');
ylabel('Amplitude');
subplot(3, 1, 3);
 plot(t, predicted_signal);
title('Output    Signal    from    the    Neural    Network');
xlabel('Time'); ylabel('Amplitude');
```

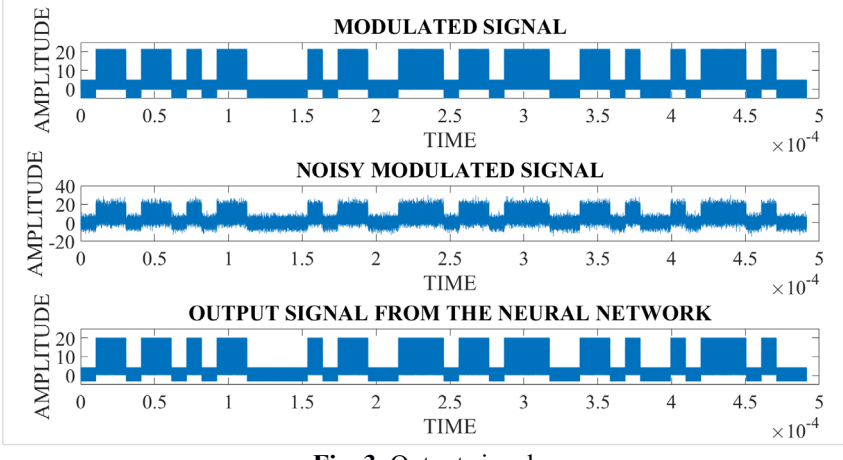

Fig. 3. Output signal