Laboratory manual

Scilab Virtual Software Laboratory

UbiLAB

Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

Hochschule Anhalt
Anhalt University of Applied Sciences

University of Maribor

Erasmus+

NATIONAL AGENCY
for European Educational Programmes and Mobility

3 Scilab Virtual Software Laboratory

3.1 Introduction

The Virtual Software Laboratories (VSL) developed via the UbiLAB platform, can be used and implemented in a vast variety of online courses and applications. VSLs are implemented in the UbiLAB framework through the Moodle LMS (however, everything described below can be easily implemented on other Learning Management Systems, such as Open edX).

In order to define and organize a VSL in the UbiLAB framework several elements need to be incorporated and connected. First, the access to the VSL needs to be configured. This is realized by using the Apache Guacamole framework and integrating it via the iFrame plugin in the LMS. Namely, Apache Guacamole is an open-source remote desktop gateway that allows users to access their desktops or applications through a web browser. It provides a way for users to access their desktops and applications from anywhere with an internet connection. The Guacamole server is installed on a server machine and configured to allow connections to remote desktops or applications. It uses standard protocols like VNC and RDP to communicate with the remote desktop, and it provides a web interface that allows users to access their remote desktops or applications from any device with a web browser.

The Guacamole server can be configured to enable access to different software environments (Linux and Windows-based virtual machines, Docker or LXC containers and similar) equipped with the needed software applications for the online course, in our case Scilab. These environments actually present the resources in the LMS that the student needs to reserve. The reservation process is done through the Scheduling module implemented into the LMS, described later.

When the Guacamole server is configured, the appropriate web interface can be embedded within an LMS iFrame. The iFrame is a web feature that can be easily embedded in different LMS platforms (Moodle). It allows embedding external web pages or applications (in this case the Guacamole web interface) within the LMS course page. An iFrame (short for inline frame) is a HTML element that can display the contents of another web page within a frame on your own/current web page. By embedding the Guacamole web interface within an iFrame, the students can access the remote virtual laboratory software environments or applications without leaving the LMS platform.

When the resources in the course have been configured, in such a way that one virtual machine (virtual software environment) is one resource that can be used by one (or several, depending on the configuration/logic of the exercise) students. Then by using the implemented Scheduler module from the LMS (Moodle), teachers can organize the laboratory exercises for the entire semester, create suitable laboratory slots (even for the entire semester), define laboratory exercise durations, set the maximum number of participants per exercise, and specify which students or groups can sign up for each exercise. The scheduler is connected to the actually definer resource in the course as explained previously. Then, students can sign up for available exercise slots directly from within the course, and teachers can manage and view their exercises through the Scheduler interface. Students are allowed access to the resource (virtual software environment) just during the time slot he/she reserved.

3.2 Scilab

Scilab is a free, open-source numerical computation software package for scientific and engineering applications. It provides a powerful computing environment for solving complex mathematical problems, creating and manipulating graphs and visualizations, and performing data analysis.

Scilab is similar in functionality to other popular numerical computing environments such as MATLAB, GNU Octave, and Python's NumPy library. It includes a large library of built-in mathematical functions, as well as tools for linear algebra, signal processing, optimization, statistics, and more.

Scilab also supports the development of user-defined functions and modules, making it highly customizable and extensible. It can be used for a wide range of applications, including scientific research, engineering design, and education.

Overall, Scilab is a powerful and flexible software package that provides a comprehensive set of tools for numerical computing and scientific analysis.

3.3 Laboratory experiment 1: AM Modulation

3.3.1 Goals of the experiment

The objective of this laboratory exercise is to provide a practical understanding of amplitude modulation (AM) using Xcos, a graphical modeling and simulation tool in Scilab. Amplitude modulation is a widely used technique in communication systems for transmitting information by varying the amplitude of a carrier wave. In this lab, we will use Xcos to simulate an AM system, explore the key components, and analyze the effects of different parameters on the modulation process.

3.3.2 Experimental setup

- Launch Scilab on the virtual machine and open the Xcos environment.
- Familiarize yourself with the Xcos interface, which consists of a palette of blocks, a model editor area, and a simulation toolbar.
- Locate the blocks specific to AM modulation in the Xcos palette:
    - GENSIN_f: sine wave generator;
    - CONST_m: constant value generator.
    - PRODUCT: element-wise multiplication of its vector inputs
    - BIGSOM_f: performs addition on its scalar or vector inputs
    - CSCOPE: displays its input with respect to simulation time
    - CLOCK_s: generates a regular train of events that are scheduled by parameter

3.3.3 Experimental procedure

1. Start by creating a new Xcos model. Drag and drop the necessary blocks from the palette onto the model editor area to build the AM modulation system (as shown in Figure 3-1). Connect the blocks to establish the signal flow.

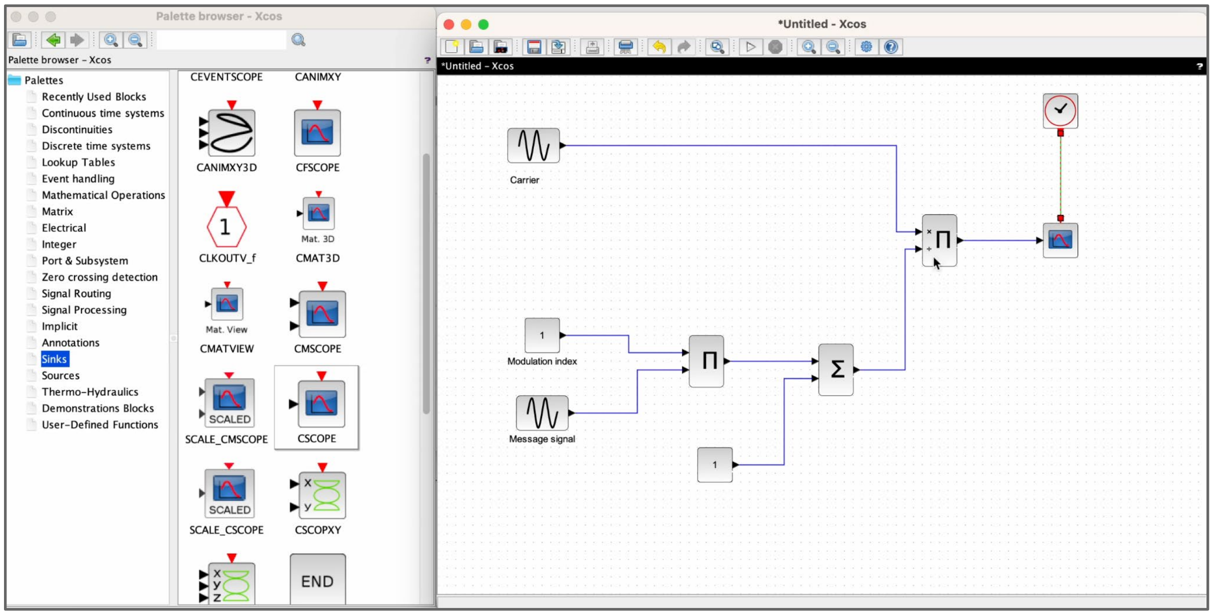
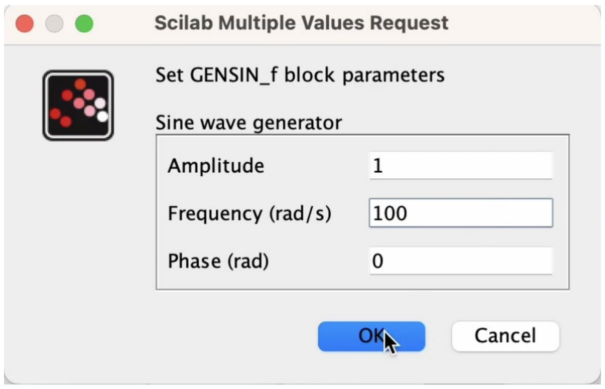


Figure 3-1. Scheme of the AM modulation system.

2. Set the frequency and amplitude of the carrier signal and the message signal (sine wave generators), as shown in Figure 3-2 and Figure 3-3.



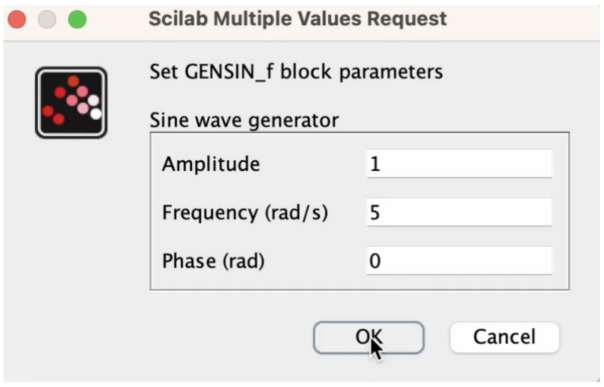
Scilab Multiple Values Request

Set GENSIN_f block parameters

Sine wave generator

Amplitude	1
Frequency (rad/s)	100
Phase (rad)	0

Figure 3-2. Carrier signal.



Scilab Multiple Values Request

Set GENSIN_f block parameters

Sine wave generator

Amplitude	1
Frequency (rad/s)	5
Phase (rad)	0

Figure 3-3. Modulation signal.

3. Run the simulation and observe the modulated signal on the Scope block. Pay attention to the changes in amplitude and frequency as a result of the modulation process.
4. Vary the parameters of the modulating signal, such as frequency and amplitude, to observe their effects on the modulated signal. Note any changes in amplitude modulation depth or spectral content. Additionally, vary the modulation index.

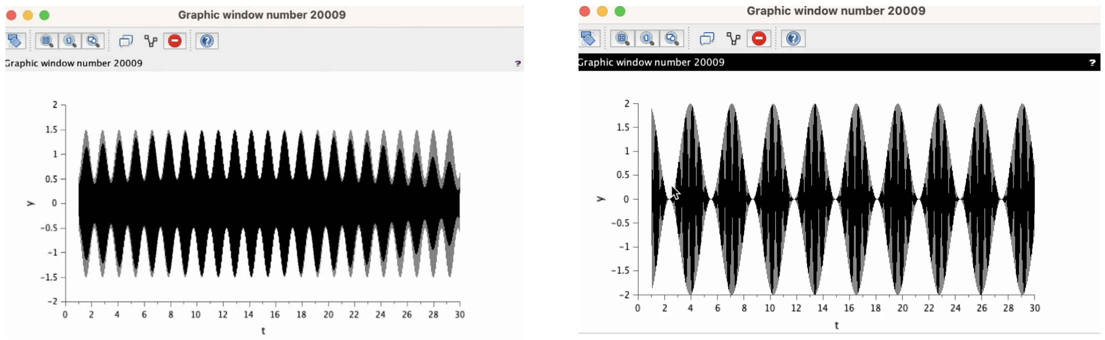


Figure 3-4. Modulated signal for different system parameters.

5. Experiment with different carrier frequencies while keeping the modulating signal constant. Observe how the carrier frequency affects the characteristics of the modulated signal (Figure 3-4).
6. Use Xcos's simulation features to analyze the demodulation process. Introduce an AM Demodulator block and connect it to the modulated signal. Connect the output of the demodulator to another Scope block to visualize the demodulated signal.

3.3.4 Experimental results

This laboratory exercise provided a practical understanding of amplitude modulation using Xcos in Scilab. By building and simulating an AM system, you have observed the effects of different parameters on the modulation and demodulation processes. Through Xcos's graphical modeling capabilities, you have gained insight into the modulation depth, carrier frequency, and spectral content of the modulated signal. This knowledge will serve as a foundation for further exploration of modulation techniques and their applications in communication systems.

3.4 Laboratory experiment 2: AM Demodulation

3.4.1 Goals of the experiment

The objective of this laboratory exercise is to provide a practical understanding of amplitude demodulation using Xcos, a graphical modeling and simulation tool in Scilab. Amplitude demodulation is the process of extracting the original modulating signal from an amplitude-modulated (AM) carrier signal. In this lab, we will use Xcos to simulate an AM demodulation system, explore the key components, and analyze the effects of different parameters on the demodulation process.

3.4.2 Experimental setup

- Launch Scilab on the virtual machine and open the Xcos environment.
- Familiarize yourself with the Xcos interface, which consists of a palette of blocks, a model editor area, and a simulation toolbar.
- Locate the blocks specific to AM modulation in the Xcos palette:
  - GENSIN_f: sine wave generator;
  - CONST_m: constant value generator.
  - PRODUCT: element-wise multiplication of its vector inputs
  - BIGSOM_f: performs addition on its scalar or vector inputs
  - CSCOPE: displays its input with respect to simulation time
  - CLOCK_s: generates a regular train of events that are scheduled by parameter
  - CLR: SISO linear system represented by its rational transfer function Numerator/Denominator

3.4.3 Experimental procedure

1. Start by creating a new Xcos model. Drag and drop the necessary blocks from the palette onto the model editor area to build the AM demodulation system (as shown in Figure 3-5). Connect the blocks to establish the signal flow.

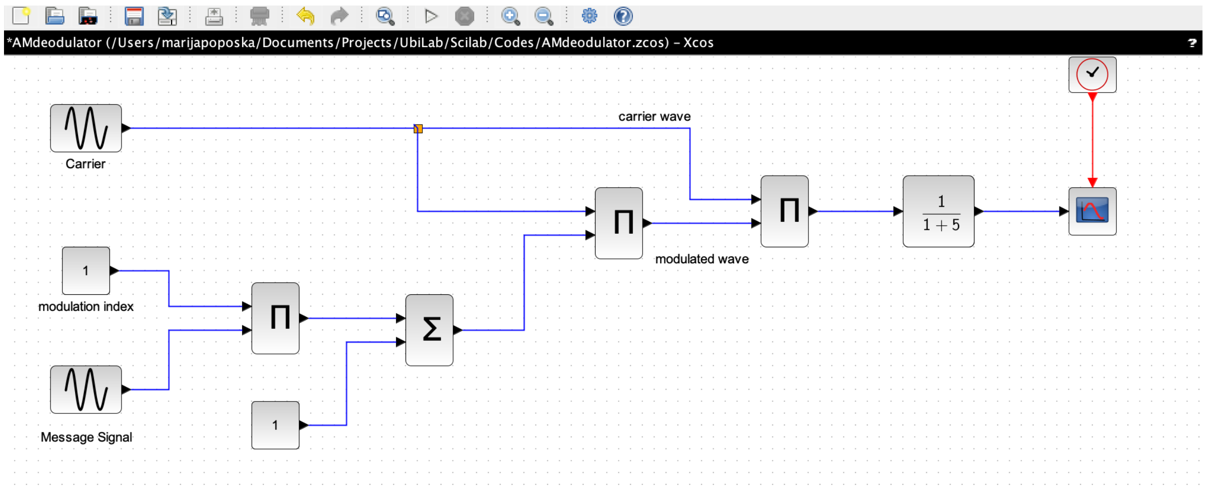
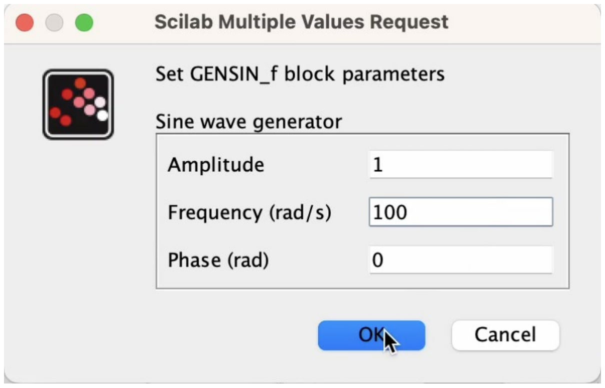

*AMdeodulator (/Users/marijapoposka/Documents/Projects/UbiLab/Scilab/Codes/AMdeodulator.zcos) – Xcos
Carrier
carrier wave
1
modulation index
Π
Σ
Message Signal
modulated wave
1
1 + 5

Figure 3-5. Scheme of the AM demodulation system.

2. Set the frequency and amplitude of the carrier signal and the message signal (sine wave generators), as shown in Figure 3-6 and Figure 3-7.



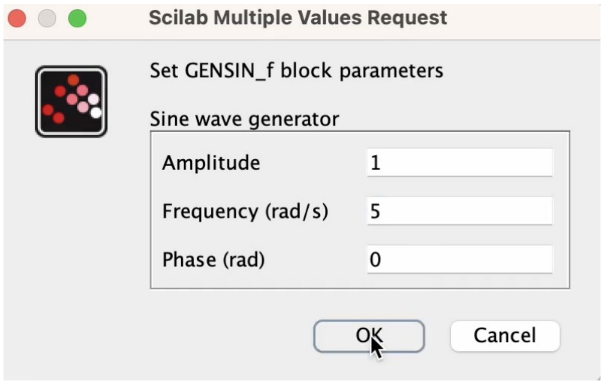
Scilab Multiple Values Request

Set GENSIN_f block parameters

Sine wave generator

Amplitude	1
Frequency (rad/s)	100
Phase (rad)	0

Figure 3-6. Carrier signal.



Scilab Multiple Values Request

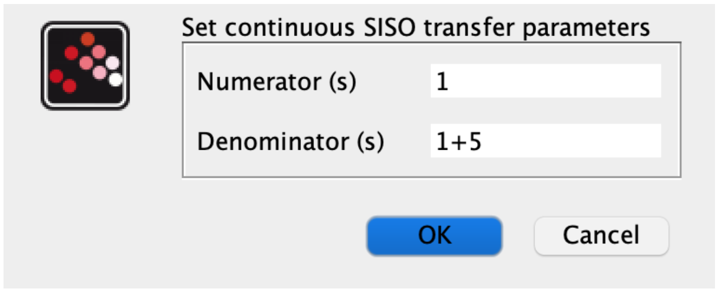
Set GENSIN_f block parameters

Sine wave generator

Amplitude	1
Frequency (rad/s)	5
Phase (rad)	0

Figure 3-7. Modulation signal.

3. Set the values of the low-pass filter (Figure 3-8).


Set continuous SISO transfer parameters

Numerator (s) 1

Denominator (s) 1+5

Figure 3-8. Low-pass filter.

4. Run the simulation and observe the demodulated signal on the Scope block. Pay attention to the changes in amplitude and frequency as a result of the demodulation process.
5. Vary the parameters of the modulating signal, such as frequency and amplitude, to observe their effects on the demodulated signal. Note any changes in amplitude modulation depth or spectral content. Additionally, vary the values of the low-pass filter.

3.4.4 Experimental results

This laboratory exercise provided a practical exploration of amplitude demodulation using Xcos in Scilab. By building and simulating an AM demodulation system, you have observed the effects of different parameters and techniques on the fidelity and accuracy of the demodulated signal. Through Xcos's graphical modeling capabilities, you have gained insight into envelope detection, synchronous demodulation, and the impact of noise on the demodulation process. This knowledge will serve as a foundation for further exploration of demodulation techniques and their applications in communication systems.

3.5 Laboratory experiment 3: Digital Modulation Techniques

3.5.1 Goals of the experiment

The objective of this laboratory exercise is to provide a hands-on understanding of digital modulation techniques, specifically Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Phase Shift Keying (PSK). These techniques are widely used in digital communication systems to transmit digital data over a carrier signal by manipulating amplitude, frequency, or phase. In this lab, we will use Scilab to simulate and analyze ASK, FSK, and PSK systems, and investigate their characteristics and performance.

3.5.2 Experimental setup

- Launch Scilab on your computer and create a new script or open the Scilab editor.
- Familiarize yourself with the Scilab environment and basic programming concepts, such as variables, functions, and plotting capabilities.

3.5.3 Experimental procedure

1. Start by defining the necessary parameters for ASK, FSK, and PSK systems, such as the carrier frequency, symbol rate, modulation index, and phase offsets. Assign appropriate values to these parameters based on the desired characteristics of each modulation technique.

2. Generate a digital signal, such as a binary sequence, to be modulated using ASK, FSK, and PSK techniques. Assign appropriate values to represent the digital data.

```
clc;
clear;
xdel(winsid());
sym=10;//no. of symbols
g=[1 1 1 0 1 0 0 1 0 1 ]//binary data
f1=1;f2=2;//frequencies of carrier
t=0:2*%pi/99:2*%pi;//range of time
```

3. Implement the ASK modulation scheme by creating a function or series of Scilab commands to encode a digital signal onto an amplitude-modulated carrier wave. This can involve multiplying the digital signal with the carrier signal to achieve the desired modulation.

```
cp=[];bit=[];mod_ask=[];mod_fsk=[];mod_psk=[];cp1=[];cp2=[];
for n=1:length(g);
  if g(n)==0;
    die=zeros(1,100);
  else g(n)==1;
    die=ones(1,100);
  end
  c_ask=sin(f1*t);
  cp=[cp die];
  mod_ask=[mod_ask c_ask];
  end
ask=cp.*mod_ask;//ASK modulated signal
```

4. Implement the FSK modulation scheme by creating a function or series of Scilab commands to encode the digital signal onto frequency-modulated carrier waves. This can involve generating two carrier signals at different frequencies and switching between them based on the digital signal.

```
for n=1:length(g);
    if g(n)==0;
        die=ones(1,100);
        c_fsk=sin(f1*t);
    else g(n)==1;
        die=ones(1,100);
        c_fsk=sin(f2*t);
    end
    cp1=[cp1 die];
    mod_fsk=[mod_fsk c_fsk];
end
fsk=cp1.*mod_fsk;//FSK molated signal
```

5. Implement the PSK modulation scheme by creating a function or series of Scilab commands to encode the digital signal onto phase-modulated carrier waves. This can involve adjusting the phase of the carrier signal according to the digital signal.

```
for n=1:length(g);
    if g(n)==0;
```

```
            die=ones(1,100);
            c_psk=sin(f1*t);
        else g(n)==1;
            die=ones(1,100);
            c_psk=-sin(f1*t);
        end
        cp2=[cp2 die];
        mod_psk=[mod_psk c_psk];
    end
    psk=cp2.*mod_psk;//PSK modulated signal
```

6. Plot and analyze the modulated signals in the time domain to observe the effects of each modulation technique on the carrier wave.

```
subplot(4,1,1);plot(cp,'LineWidth',1.5);//plot binary signal
xgrid;
title('Binary Signal');//title
mtlb_axis([0 100*length(g) -2.5 2.5]); //axis range
subplot(4,1,2);plot(ask,'LineWidth',1.5);//plot of ASK modulated signal
xgrid;
title('ASK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//axis range
subplot(4,1,3);plot(fsk,'LineWidth',1.5);//plot of FSK modulated signal
xgrid;
title('FSK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//axis range
subplot(4,1,4);plot(psk,'LineWidth',1.5);//plot of PSK modulated signal
xgrid;
title('PSK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//range of axis
```

3.5.4 Experimental results

This laboratory exercise provided a practical exploration of digital modulation techniques, specifically ASK, FSK, and PSK, using Scilab. By implementing and simulating these techniques, you have gained insight into how they manipulate carrier signals to transmit digital data. Through analyzing the modulated signals and demodulating them to obtain the original digital data, you have evaluated the performance of each modulation technique (Figure 3-9). This knowledge will serve as a foundation for further exploration of digital communication systems and modulation schemes.

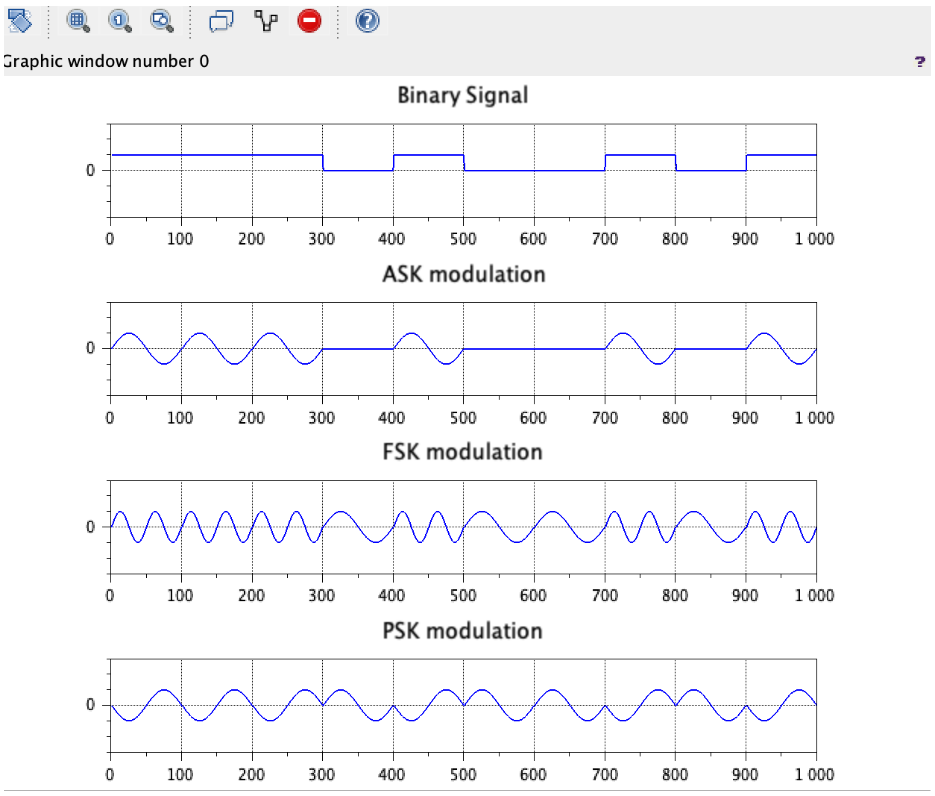


Figure 3-9. Results of the digital modulation techniques.

3.6 Conclusion

In conclusion, the Scilab experiment on analog and digital modulation provided valuable insights into the fundamental concepts and techniques used in communication systems. Through hands-on simulations, we explored both analog modulation methods, such as Amplitude Modulation (AM), as well as digital modulation techniques, including Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Phase Shift Keying (PSK).

Amplitude modulation allowed us to understand the relationship between the modulating signal and the carrier. On the other hand, digital modulation provided us with insights into how digital data is conveyed using discrete symbols. ASK, FSK, and PSK modulation schemes showcased diverse approaches to encoding digital information onto carrier waves, each with its unique strengths and applications.

By leveraging the capabilities of Scilab, we were able to visualize, analyze, and demodulate complex signals, enabling a deeper understanding of the performance and characteristics of each modulation technique.