NetXPTO - LinkPlanner

5 de Setembro de 2017

Conteúdo

Capítulo 1

Introduction

Capítulo 2

Simulator Structure

LinkPlanner is a signals open-source simulator.

The major entity is the system.

A system comprises a set of blocks.

The blocks interact with each other through signals.

2.1 System

You can run the System

Capítulo 3

Development Cycle

The NetXPTO-LinkPlanner has been developed by several people using git as a version control system. The NetXPTO-LinkPlanner repository is located in the GitHub site http://github.com/netxpto/linkplanner. The more updated functional version of the software is in the branch master. Master should be considered a functional beta version of the software. Periodically new releases are delivered from the master branch under the branch name Release<Year><Month><Day>. The integration of the work of all people is performed by Armando Nolasco Pinto in the branch Develop. Each developer has is how branch with his/her name.

Capítulo 4

Visualizer

visualizer

Capítulo 5

Case Studies

5.1 QPSK Transmitter

This system simulates a QPSK transmitter. A schematic representation of this system is shown in figure 5.1.

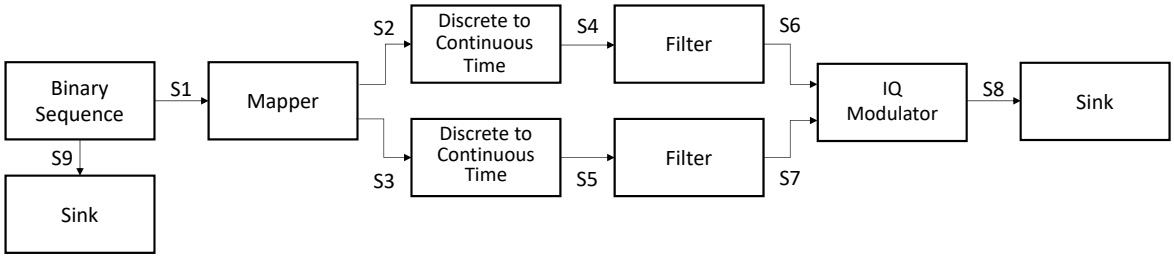


Figura 5.1: QPSK transmitter block diagram.

System Input Parameters

Parameter: *sourceMode*

Description: Specifies the operation mode of the binary source.

Accepted Values: PseudoRandom, Random, DeterministicAppendZeros, DeterministicCyclic.

Parameter: *patternLength*

Description: Specifies the pattern length used my the source in the PseudoRandom mode.

Accepted Values: Integer between 1 and 32.

Parameter: *bitStream*

Description: Specifies the bit stream generated by the source in the DeterministicCyclic and DeterministicAppendZeros mode.

Accepted Values: "XXX..", where X is 0 or 1.

Parameter: *bitPeriod*

Description: Specifies the bit period, i.e. the inverse of the bit-rate.

Accepted Values: Any positive real value.

Parameter: *iqAmplitudes*

Description: Specifies the IQ amplitudes.

Accepted Values: Any four par of real values, for instance { { 1,1 },{ -1,1 },{ -1,-1 },{ 1,-1 } }, the first value correspond to the "00", the second to the "01", the third to the "10"and the forth to the "11".

Parameter: *numberOfBits*

Description: Specifies the number of bits generated by the binary source.

Accepted Values: Any positive integer value.

Parameter: *numberOfSamplesPerSymbol*

Description: Specifies the number of samples per symbol.

Accepted Values: Any positive integer value.

Parameter: *rollOffFactor*

Description: Specifies the roll off factor in the raised-cosine filter.

Accepted Values: A real value between 0 and 1.

Parameter: *impulseResponseTimeLength*

Description: Specifies the impulse response window time width in symbol periods.

Accepted Values: Any positive integer value.

5.2 Quantum Noise

Introduction

This document describes an emission and detection system which is used to simulate the effects of quantum noise. The system is based on the transmission of coherent states as the support of information. The following section introduces some aspects about coherent states, with special focus on quantum noise.

We start by defining number states $|n\rangle$ (or Fock states), which correspond to states with perfectly fixed number of photons. [1] Associated to those states are two operators, the creation $\hat{a}^\dagger$ and annihilation $\hat{a}$ operators, which in a simple way, remove or add one photon from a given number state. [2] Their action is defined as:

$$\hat{a}\,|n\rangle = \sqrt{n}\,|n-1\rangle \qquad \hat{a}^\dagger\,|n\rangle = \sqrt{n+1}\,|n+1\rangle$$

$$\hat{n}\,|n\rangle = n\,|n\rangle$$

in which $\hat{n} = \hat{a}^\dagger\hat{a}$ is the number operator. Therefore, number states are eigenvectors of the number operator.

Coherent states have properties that closelly resemble classical electromagnetic waves, and are generated by single-mode lasers well above the thereshold. [3] We can defined them, using number states in the followi■manner:

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}}\,|n\rangle$$

in which α is the sole parameter that caracterizes it. [4] [5] [6] In fact, if we calculate the expected number of photons with $\langle\alpha|\,\hat{n}\,|\alpha\rangle$ we will obtain $|\alpha|^2$. The coherent state is an eigenstate of the annihilation operator, $\hat{a}\,|\alpha\rangle = \alpha\,|\alpha\rangle$.

Using the creation and annihilation operators, we can define two quadrature operators:[7]

$$\hat{X} = \frac{1}{2}\left(\hat{a}^\dagger + \hat{a}\right) \qquad \hat{Y} = \frac{i}{2}\left(\hat{a}^\dagger - \hat{a}\right)$$

The expected value of these two operators, using a coherent state $|\alpha\rangle$ are:

$$\langle\hat{X}\rangle = \mathrm{Re}(\alpha) \qquad \langle\hat{Y}\rangle = \mathrm{Im}(\alpha)$$

[1] Ulf, p.21
[2] REFERENCE
[3] Loudon, p.190
[4] (number state)Loudon, p.137
[5] Loudon, p.184
[6] Loudon, p.186
[7] Loudon, p.138, (4.3.36)

We see that the expected value of these operators give us the real and imaginary part of α. Now, we can obtain the uncertainty of these operators, using:

$$\text{Var}(\hat{X}) = \langle \hat{X}^2 \rangle - \langle \hat{X} \rangle^2$$

(VALERA A PENA GENERALIZAR A QUADRATURA?)
For each of these quadrature operators the variance will be:

$$\text{Var}(\hat{X}) = \text{Var}(\hat{Y}) = \frac{1}{4}$$

This result show us that for both quadratures, the variance of measurement is the same and independent of the value of α.

—WARNING: WORK IN PROGRESS—
Experimental setting
Given an state $|\alpha\rangle$, we can use the homodyne technique to measure the phase of this signal (S), relative to the phase of a local oscillator (LO). This technique consists in the interference of the input signal with a local oscilator which has the same frequency as the input signal, but an overwhelming largest power.
Operators X and Y correspond to the measurement of the in-phase (X) and quadrature (Y), which will depend on the phase of the local oscillator.????

Noise sources
There are several sources of noise, when measuring a quadrature (QUEM DIZ ISSO?). One of them is quantum noise which is related to oscillations in the electromagnetic field. [8] Other source of noise is thermal noise which is related to the electronic operations.??????

Quantum Noise
The quantum mechanical description of homodyne detection uses quantum operators to describe the effect of each component in the system. So, we start with the operators $\hat{a}_S$ and $\hat{a}_{LO}$ corresponding to the annihilation operator for the signal and local oscillator, which we be the inputs in a beam divisor. The outputs will be $\hat{a}_3$ and $\hat{a}_4$. We will use a balanced division, therefore, we can write the output as:

$$\hat{a}_3 = \frac{1}{\sqrt{2}}(\hat{a}_S + \hat{a}_{LO}) \qquad \hat{a}_4 = \frac{1}{\sqrt{2}}(\hat{a}_S - \hat{a}_{LO})$$

The final output of a homodyne measurement will be proportional to the difference between the photocurrents in arm 3 and 4. Then

$$I_{34} = I_3 - I_4 \sim \langle \hat{n}_3 - \hat{n}_4 \rangle$$

We can define an operator that describes the difference of number of photons in arm 3 and arm 4:

$$\hat{m} = \hat{a}_3^\dagger \hat{a}_3 - \hat{a}_4^\dagger \hat{a}_4$$

[8] REFERENCIA

If we assume the local oscillator as the coherent state $|\beta\rangle$, then the expected value of this measurement will be:

$$\langle m\rangle = 2|\alpha||\beta|\cos(\theta_\alpha - \theta_\beta) \qquad \text{Var}(m) = |\alpha|^2 + |\beta|^2$$

The local oscillator normally has a greater power than the signal (VER REFERENCIAS), then $|\alpha| \ll |\beta|$. If we use as unit, $2|\beta|$, then these two quantities can be simplified to:

$$\langle m\rangle = |\alpha|\cos(\theta_\alpha - \theta_\beta) \qquad \text{Var}(m) =\approx \frac{1}{4}$$

[9]

We can measure the two quadratures simultaneously. We divide the signal in two beams. One of the beams interferes with a local oscillator and the other will interfere with another oscillator that has a phase difference of $\pi/2$ relative to the other local oscillator.
Experimentaly, the optical hybrid (SEE SCHEME BELLOW) will take this role of mixing the signal with the two local oscillators.

SIMULATION IMPLEMENTATION:
In the simulator, we can model each arm's photocurrent (I_3 and I_4) and then obtain I_{34} by a simple subtraction.
The expected value and variance of $\hat{n}_3$ is:

$$\langle \hat{n}_3\rangle = \frac{1}{2}\left(|\alpha|^2 + |\beta|^2 + 2\cos(\phi_\alpha - \phi_\beta)\right) \qquad \text{Var}(\hat{n}_3) = \langle \hat{n}_3\rangle$$

Calculating the expected value and variance for $\hat{n}_4$, we will also obtain $\text{Var}(\hat{n}_4) = \langle \hat{n}_4\rangle$, therefore both processes can be modelled by Poisson distributions:

$$n_i \sim \text{Poisson}(\langle n_i\rangle)$$

In the simulation, a photodiode has 2 inputs. Each input corresponds to a mean complex amplitude $\bar{\Psi}_i$. This amplitude corresponds to the power $\bar{P}_i = |\bar{\Psi}_i|^2 = \langle n_i\rangle P_\lambda$ in which P_λ is the power of a single photon. What we want is the power after adding noise, $P_i = n_i P_\lambda$. To model n_i, we will use a gaussian aproximation of $\sqrt{n_i}$, for $n_i \gg 0$:

$$\sqrt{n_i} \sim \text{Gaussian}(\sqrt{\langle n_i\rangle}, 1/2)$$

Therefore:

$$n_i \approx \left(\sqrt{\langle n_i\rangle} + \frac{1}{2}G\right)^2 = \langle n_i\rangle + \sqrt{\langle n_i\rangle}G + \frac{1}{4}G^2$$

in which G is a random variable with a gaussian distribution with mean 0 and variance 1.
The output power becomes:

$$P_i = n_i P_\lambda = \bar{P}_i + \sqrt{P_\lambda \bar{P}}G + \frac{1}{4}P_\lambda G^2$$

[9] Referencia indirecta: Livro: Hans, p.207

Thermal noise
[10] Thermal noise is generated by electron in response to temperature.
(TIRADO DE PAPERS)

$$\langle i_T \rangle = 4K_B T_0 B / R_L$$

In which K_B it's Boltzmann's constant, T_0 is the absolute temperature and R_L is the receiver load impedance.

The values of B is dependent in the configuration imposed in the system and the R_L value is dependent in the internal setup of the various components. Therefore, we can obtain a experimental value in order to configure the simulation.

Functional Description

The simulation setup is described by diagram in figure 5.2. We start by generating a state from one of the four available ones.???? Then, the signal is received in a Hybrid Detector??? where the signal is compared with a local oscillator giving four different signals in it's output. Two of those signals are detected by a photodiode which output will be the difference of the two photocurrents. The other two signals will be also be detected by another photodiode, which will obtain the other quadrature of the signal.????? (TEM QUE FICAR MELHOR EXPLICADO).

System Blocks	netxpto Blocks
M - Quadrature Amplitude Modulator	MQamTransmitter
Local Oscillator	LocalOscillator
-	OpticalHybrid
Photodiode	Photodiode
-	Sampler

Required files

Header Files

[10] Mark Fox, p. 96

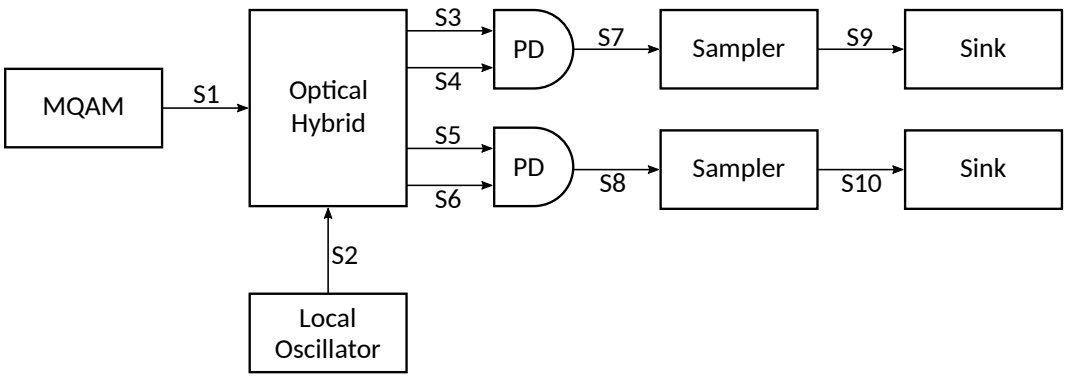


Figura 5.2: Overview of the optical system being simulated.

File	Description
netxpto.h	Generic purpose simulator definitions.
m_qam_transmitter.h	Outputs a QPSK modulated optical signal.??
local_oscillator.h	Generates continuous coherent signal.
optical_hybrid.h	—
photodiode.h	Converts two optical signals to a difference of photocurrents.??
sampler.h	Samples the input signal.??
sink.h	Closes any unused signals.

Source Files

File	Description
netxpto.cpp	Generic purpose simulator definitions.
m_qam_transmitter.cpp	Outputs a QPSK modulated optical signal.??
local_oscillator.cpp	Generates continuous coherent signal.
optical_hybrid.cpp	—
photodiode.cpp	Converts two optical signals to a difference of photocurrents.??
sampler.cpp	Samples the input signal.??
sink.cpp	Closes any unused signals.

System Input Parameters

This system takes into account the following input parameters:

System Parameters	Description
numberOfBitsGenerated	Gives the number of bits to be simulated
bitPeriod	Sets the time between adjacent bits
wavelength	Sets the wavelength of the local oscillator in the MQAM????
samplesPerSymbol	Establishes the number of samples each bit in the string is given
localOscillatorPower1	Sets the optical power, in units of W, of the local oscillator inside the MQAM
localOscillatorPower2	Sets the optical power, in units of W, of the local oscillator used for Bob's measurements
localOscillatorPhase	Sets the initial phase of the local oscillator used in the detection
transferMatrix	Sets the transfer matrix of the beam splitter used in the homodyne detector
responsivity	Sets the responsivity of the photodiodes used in the homodyne detectors
bufferLength	Sets the length of the buffer used in the signals
iqAmplitudeValues	Sets the amplitude of the states used in the MQAM????
shotNoise	Chooses if quantum shot noise is used in the simulation
samplesToSkip	Sets the number of samples to skip when writing out some of the signal files.

Inputs

This system takes no inputs.

Outputs

The system outputs the following objects:

Parameter: Signals:

Description: Binary Sequence used in the MQAM; (S_0)

Description: Local Oscillator used in the MQAM; (S_1)

Description: Local Oscillator used in the detection; (S_2)

Description: Optical Hybrid Outputs; (S_3, S_4, S_5, S_6)

Description: In phase Photodiode output; (S_7)

Description: Quadrature Photodiode output; (S_8)

Description: In phase Sampler output; (S_9)

Description: Quadrature Sampler output; (S_{10})

Simulation Results

The objective of this simulation was to get the (quantum noise???) associated to the detection of coherent states.

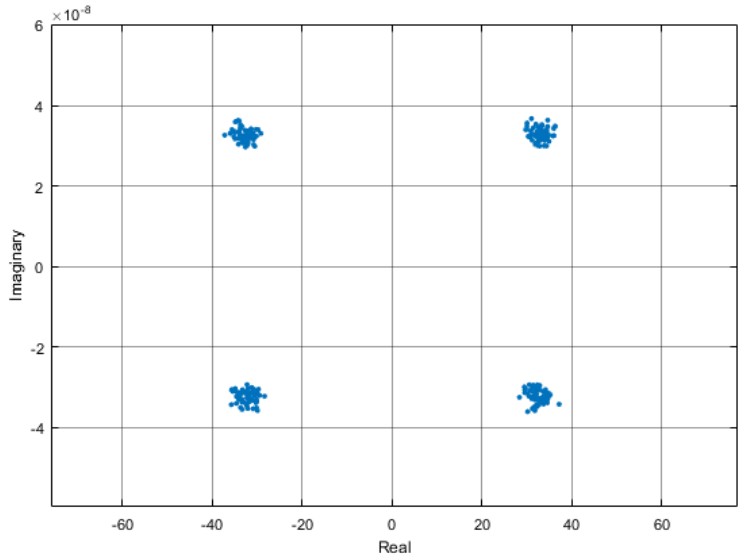


Figura 5.3: Simulation of a constelation of 4 states (n = 100)

We expect that the variance is invariant with the number of photons sent from Alice. The plot in 5.4 show that the simulation also shows this invariance with the number of photons.

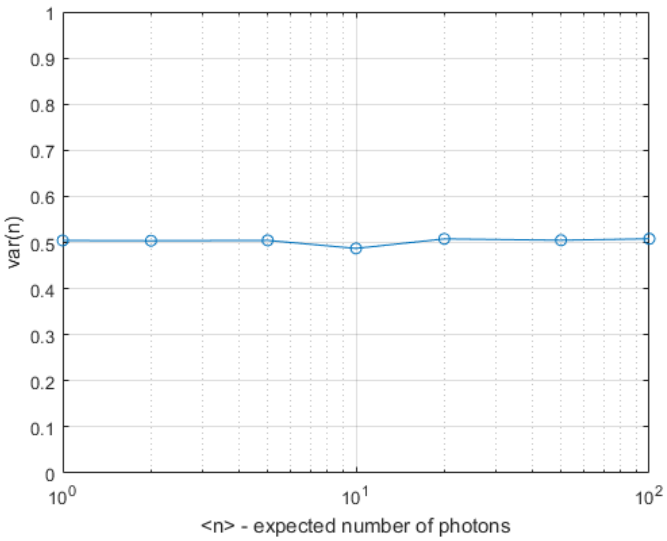


Figura 5.4: Simulation of the variance of n.

We can conclude that the expected variance will give us Var(X) $= \frac{1}{2}$.
The results obtained in our simulations are in accordance with the theoretical prevision???

Known Problems

1. —-

5.3 Continuos Variable Quantum Transmission System

In this section a continuous varible quantum transmission system is analyzed. The results here presented follow closely the [1]. In [1], the security of a continuous variable quantum key distribution (CV-QKD) system is studied theoretically, here we complete that theoretical study with simulations results.

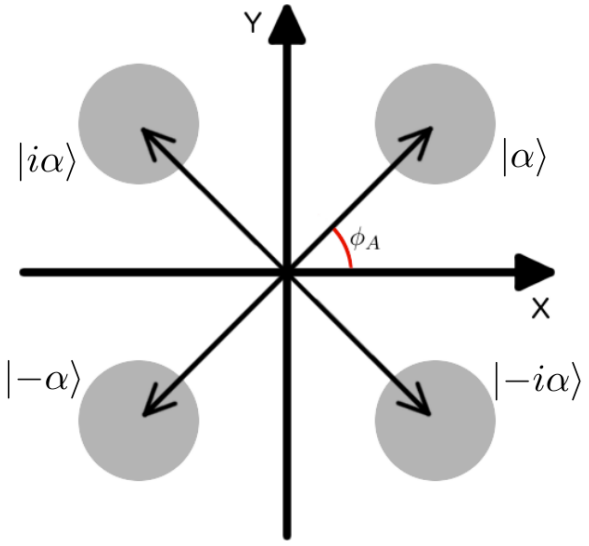


Figura 5.5: State constellation

The state constellation used in the system is presented in Figure 5.5. The emitter (usually named Alice) is going to use two basis, the 45^o base and the -45^o base. In the 45^o base, Alice sends one of two values, 1 and -1, which correspond to the states $|\alpha\rangle$ and $|-\alpha\rangle$. In the -45^o base, Alice can also send one of two values, 1 and -1, which correspond to the states $|-i\alpha\rangle$ and $|i\alpha\rangle$. At the end Alice is going to send one of the four states $|\alpha\rangle$, $|-\alpha\rangle$, $|-i\alpha\rangle$, and $|i\alpha\rangle$, with equal probability.

Because we don't know à prior which state is going be transmitted, neither which basis is going to be used, and to incorporate our "ignorance" in the system description, we can work with the density operator. The density operator is a proper tool to describe "statistical mixtures". A "statistical mixtures"is one state, from a possible set, but we don't know which state it is. There is no state superposition.

Since all states have the same probability of occurring, the state density operator is given by:

$$\hat{\rho} = \frac{1}{4}\left(|\alpha\rangle\langle\alpha| + |-\alpha\rangle\langle-\alpha| + |i\alpha\rangle\langle i\alpha| + |-i\alpha\rangle\langle-i\alpha|\right). \tag{5.1}$$

The probability to detect at the receiver the state $|\alpha\rangle$ is given by

$$P(\alpha) = \langle\alpha|\,\hat{\rho}\,|\alpha\rangle = \frac{1}{4}. \tag{5.2}$$

Note that the density operator is equivalent to the wave function in terms of the system description.

From the receiver perspective, i.e. from the Bob perspective, and after knowing the base used by Alice. The density operator can be reduce to,

$$\hat{\rho}_1 = \frac{1}{2}\left(|\alpha\rangle\langle\alpha| + |-\alpha\rangle\langle-\alpha|\right), \tag{5.3}$$

$$\hat{\rho}_2 = \frac{1}{2}\left(|i\alpha\rangle\langle i\alpha| + |-i\alpha\rangle\langle-i\alpha|\right). \tag{5.4}$$

where 1 corresponds to the 45^{o} base and -1 corresponds to -45^{o}.

Single Base Homodyne Detection

The probability of obtaining a quadrature $\hat{X}_\phi = \hat{X}_1\cos\phi + \hat{X}_2\sin\phi$ when measuring the coherent state $|\alpha\rangle$ is given by the following gaussian distribution:

$$|\langle X_\phi|\alpha\rangle|^2 = \sqrt{\frac{2}{\pi}}e^{-2(X_\phi-\alpha\cos\phi)^2}, \tag{5.5}$$

We can define the "correct"and "wrong"basis measurement probability density, respectively, as:

$$\langle X_i|\,\hat{\rho}_j\,|X_i\rangle = \begin{cases} \frac{1}{\sqrt{2\pi}}\left(e^{-2(X_i-\alpha)^2} + e^{-2(X_i+\alpha)}\right), & i = j \\ \sqrt{\frac{2}{\pi}}e^{-2X_i^2}, & i \neq j \end{cases}. \tag{5.6}$$

The post selection efficiency (PSE) can be defined as the probability of a measurement in the correct basis yields a result that satisfies the limit value X_0:

$$\begin{aligned} P(X_0,\alpha) &= \int_{-\infty}^{-X_0}\langle X_1|\,\hat{\rho}_1\,|X_1\rangle\,dX_1 + \int_{X_0}^{\infty}\langle X_1|\,\hat{\rho}_1\,|X_1\rangle\,dX_1 \\ &= \frac{1}{2}\left[\text{erfc}(\sqrt{2}(X_0+\alpha)) + \text{erfc}(\sqrt{2}(X_0-\alpha))\right]. \end{aligned} \tag{5.7}$$

The bit error rate (BER) is the normalized probability of, after choosing the correct basis, obtaining the wrong bit value:

$$Q(X_0,\alpha) = \frac{1}{P(X_0,\alpha)}\int_{-\infty}^{-X_0}|\langle X_i|\alpha\rangle|\,dX_i = \frac{\text{erfc}\left(\sqrt{2}(X_0+\alpha)\right)}{2P(X_0,n)} \tag{5.8}$$

Double Homodyne setup

In our proposed double homodyne protocol both quadratures are measured simultaneously, as such the concept of correct and wrong basis measurements has no value. Our protocol also makes use of a locally generated Local Oscillator (LO), obtained from a different laser than the one used to generate the signal, thus we have to take into account the phase drift between both lasers. High intensity reference pulses are sent periodically to allow for an

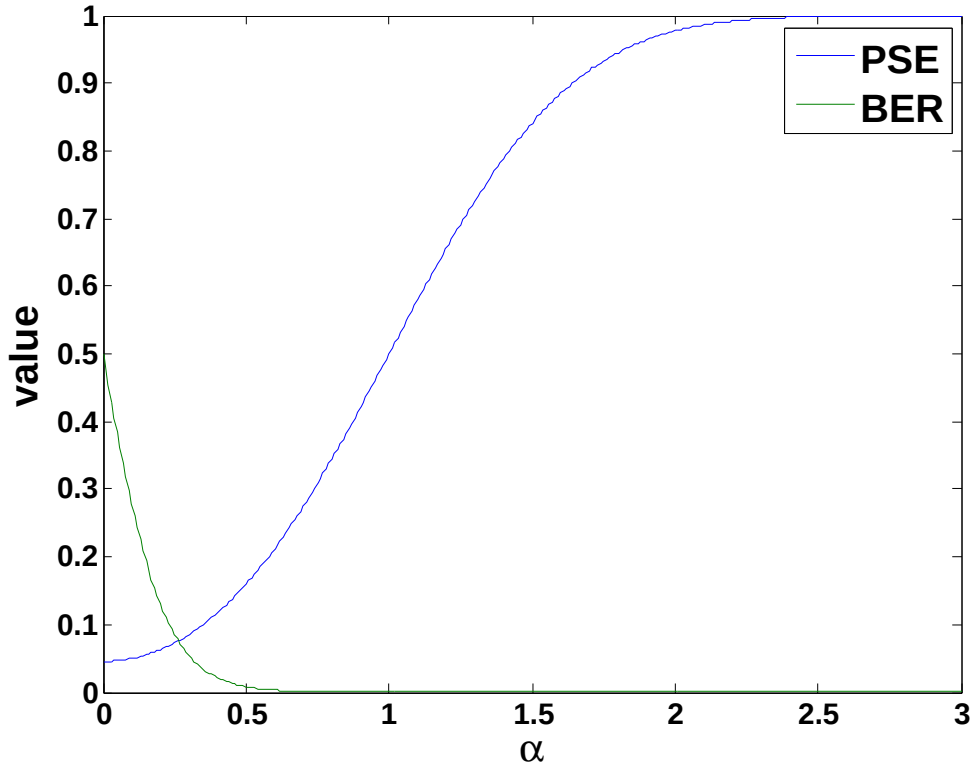


Figura 5.6: BER and PSE in function of α for the single homodyne setup. $X_0 = 1$ was used

estimation of the phase drift. The double homodyne setup requires the signal to be divided into the two utilized detectors, so each measurement is made on a coherent state with half the amplitude of the incoming signal $\alpha \to \frac{\alpha}{\sqrt{2}}$

For each incoming pulse we measure quadratures X_ϕ and Y_ϕ. ϕ has contributions from both the encoded angle, θ, and the phase difference between lasers, ϵ, we assume $\phi = \theta + \epsilon$. On the reference pulses no phase is encoded, that is $\theta = 0$, thus ϵ can be estimated. Assuming ϵ doesn't change between a reference pulse and the following signal pulse, the measured quadratures can be cast into the originally sent quadratures X_θ and Y_θ via:

$$\begin{aligned} X_\theta &= X_\phi \cos\epsilon - Y_\phi \sin\epsilon \\ Y_\theta &= X_\phi \sin\epsilon + Y_\phi \sin\epsilon \end{aligned} \tag{5.9}$$

Assuming an announcement of the coding basis, the density operators (5.3) and (5.4) still apply. We can now define the probability density of obtaining results X_θ and Y_θ, assuming

a state in the X_1 base was sent, as:

$$\langle X_\theta|\hat{\rho}_1|X_\theta\rangle = \frac{\sqrt{\frac{2}{\pi}}}{4}\left(e^{-2\left(x_\theta-\frac{\alpha}{\sqrt{2}}\cos\theta\right)^2}+e^{-2\left(x_\theta+\frac{\alpha}{\sqrt{2}}\cos\theta\right)^2}\right), \tag{5.10}$$

$$\langle Y_\theta|\hat{\rho}_1|Y_\theta\rangle = \frac{\sqrt{\frac{2}{\pi}}}{4}\left(e^{-2\left(y_\theta-\frac{\alpha}{\sqrt{2}}\sin\theta\right)^2}+e^{-2\left(y_\theta+\frac{\alpha}{\sqrt{2}}\sin\theta\right)^2}\right). \tag{5.11}$$

Now each state needs to satisfy two limit values, X_0 and Y_0, to be accepted. Thus, the PSE is now defined as:

$$\begin{aligned}P_{DH}(X_0,Y_0,\alpha) = &\int_{-\infty}^{-X_0}\langle X_\theta|\hat{\rho}_1|X_\theta\rangle\,dx_\theta\int_{-\infty}^{-Y_0}\langle Y_\theta|\hat{\rho}_1|Y_\theta\rangle\,dy_\theta+\\ &\int_{X_0}^{\infty}\langle X_\theta|\hat{\rho}_1|X_\theta\rangle\,dx_\theta\int_{Y_0}^{\infty}\langle Y_\theta|\hat{\rho}_1|Y_\theta\rangle\,dy_\theta\\ &=\frac{1}{4}\left\{\text{erfc}\left[\sqrt{2}\left(X_0-\frac{\alpha}{\sqrt{2}}\cos\theta\right)\right]+\text{erfc}\left[\sqrt{2}\left(X_0+\frac{\alpha}{\sqrt{2}}\cos\theta\right)\right]\right\}\\ &\left\{\text{erfc}\left[\sqrt{2}\left(Y_0-\frac{\alpha}{\sqrt{2}}\sin\theta\right)\right]+\text{erfc}\left[\sqrt{2}\left(Y_0+\frac{\alpha}{\sqrt{2}}\sin\theta\right)\right]\right\},\end{aligned} \tag{5.12}$$

The DH subscript denotes Double Homodyne. In a somewhat similar manner, the BER is now defined as:

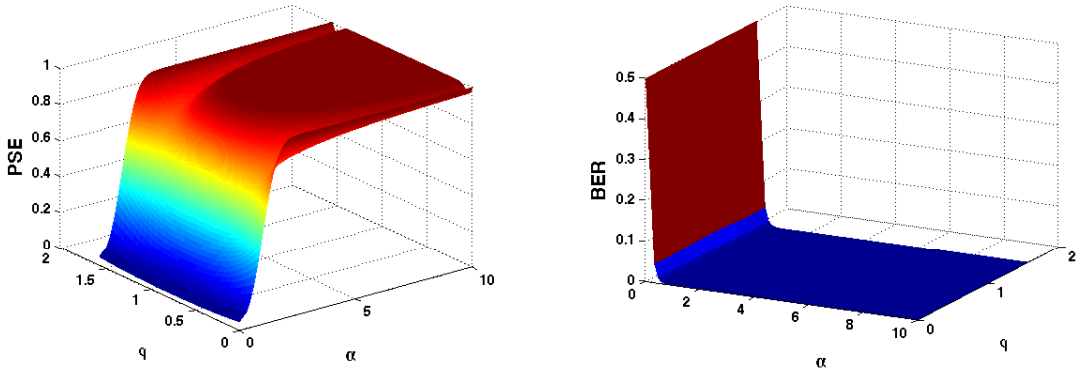
$$\begin{aligned}Q_{DH}(X_0,Y_0,\alpha) = &\frac{1}{P_{DH}}\left(\int_{-\infty}^{-X_0}\left|\langle X_\theta|\frac{\alpha}{\sqrt{2}}\rangle\right|^2dx_\theta\int_{-\infty}^{-Y_0}\left|\langle Y_\theta|\frac{\alpha}{\sqrt{2}}\rangle\right|^2dy_\theta+\right.\\ &\left.\int_{X_0}^{\infty}\left|\langle X_\theta|-\frac{\alpha}{\sqrt{2}}\rangle\right|^2dx_\theta\int_{Y_0}^{\infty}\left|\langle Y_\theta|-\frac{\alpha}{\sqrt{2}}\rangle\right|^2dy_\theta\right)\\ &=\frac{1}{2P_{DH}}\text{erfc}\left[\sqrt{2}\left(X_0+\frac{\alpha}{\sqrt{2}}\cos\theta\right)\right]\text{erfc}\left[\sqrt{2}\left(Y_0+\frac{\alpha}{\sqrt{2}}sin\theta\right)\right],\end{aligned} \tag{5.13}$$

note that, in this definition for BER, only values $\theta\in\left[0,\frac{\pi}{2}\right]$ make sense (the sent state was α).

Functional Description

Simplified diagrams of the systems being simulated are presented in Figures 5.8a. and 5.8b. Two optical signals are generated, one with a constant power level of 10 dBm and the other with power in multiples of the power corresponding to a single photon per sampling time (6.4078e$\times10^{-13}$ W for a sampling time of 200 ns). The two signals are mixed, with a Balanced Beam Splitter in the single homodyne case and with a 90º Optical Hybrid in the double homodyne one, and are subsequently evaluated with recourse to Homodyne Receivers.

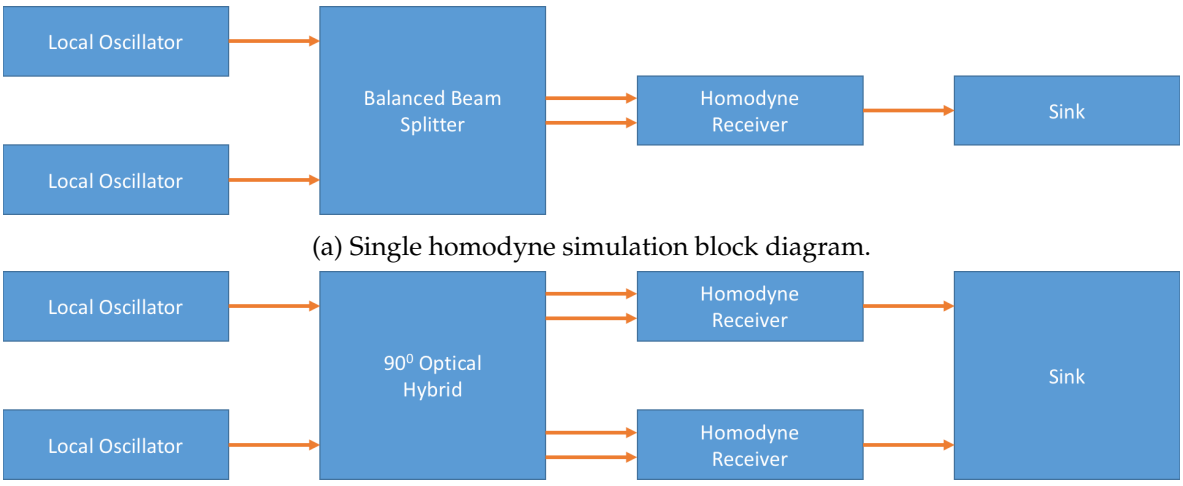
System Blocks	netxpto Blocks
Local Oscillator	LocalOscillator
Homodyne Receiver	I_HomodyneReceiver
Balanced Beam Splitter	BalancedBeamSplitter
90º Optical Hybrid	OpticalHybrid



(a) PSE in function of α and θ for the double homodyne setup. $X_0 = 1$ was used

(b) BER in function of α and θ for the double homodyne setup. $X_0 = 1$ was used

Figura 5.7: Theoretical results for double homodyne setup.



(a) Single homodyne simulation block diagram.

(b) Double homodyne simulation block diagram.

Figura 5.8: Block diagrams of both simulation results presented in this report.

Required files

Header Files

File	Description
netxpto.h	Generic purpose simulator definitions.
local_oscillator.h	Generates continuous coherent signal.
balanced_beam_splitter.h	Mixes the two input signals into two outputs.
optical_hybrid.h	Mixes the two input signals into four outputs.
homodyne_reciever.h	Performs coherent detection on the input signal.
sink.h	Closes any unused signals.

Source Files

File	Description
netxpto.cpp	Generic purpose simulator definitions.
local_oscillator.cpp	Generates continuous coherent signal.
balanced_beam_splitter.cpp	Mixes the two input signals into two outputs.
optical_hybrid.cpp	Mixes the two input signals into four outputs.
homodyne_reciever.cpp	Performs coherent detection on the input signal.
sink.cpp	Closes any unused signals.

System Input Parameters

This system takes into account the following input parameters:

System Parameters	Description
numberOfBitsGenerated	Gives the number of bits to be simulated
bitPeriod	Sets the time between adjacent bits
samplesPerSymbol	Establishes the number of samples each bit in the string is given
localOscillatorPower_dBm1	Sets the optical power, in units of dBm, at the reference output
localOscillatorPower2	Sets the optical power, in units of W, of the signal
localOscillatorPhase1	Sets the initial phase of the local oscillator used for reference
localOscillatorPhase2	Sets the initial phase of the local oscillator used for signal
transferMatrix	Sets the transfer matrix of the beam splitter used in the homodyne detector
responsivity	Sets the responsivity of the photodiodes used in the homodyne detector
amplification	Sets the amplification of the trans-impedance amplifier used in the homodyne detector
electricalNoiseAmplitude	Sets the amplitude of the gaussian thermal noise added in the homodyne detector
shotNoise	Chooses if quantum shot noise is used in the simulation

Inputs

This system takes no inputs.

Outputs

The single homodyne system outputs the following objects:

Parameter: Signals:

Description: Local Oscillator Optical Reference; (S_1)

Description: Local Oscillator Optical Signal; (S_2)

Description: Beam Splitter Outputs; (S_3, S_4)

Description: Homodyne Detector Electrical Output; (S_5)

The double homodyne system outputs the following objects:

Parameter: Signals:

Description: Local Oscillator Optical Reference; (S_1)

Description: Local Oscillator Optical Signal; (S_2)

Description: 90º Optical Hybrid Outputs; (S_3, S_4, S_5, S_6)

Description: Homodyne Detector Electrical Output; (S_7)

Simulation Results

Single homodyne results

The numerical results presented in Figure 5.9 were obtained with the simulation described by the block diagram in Figure 5.8a. Theoretical results are a direct trace of (5.8). One can see that the numerical results adhere quite well to the expected curve.

Double homodyne results

The numerical results presented in Figure 5.10 were obtained with the simulation described by the block diagram in Figure 5.8b. Theoretical results are a direct trace of (5.13) with $\theta = \frac{\pi}{4}$. One can see that the numerical results adhere quite well to the expected curve.

Known Problems

1. Homodyne Super-Block not functioning
2. 90º Optical Hybrid PDF needs to be written

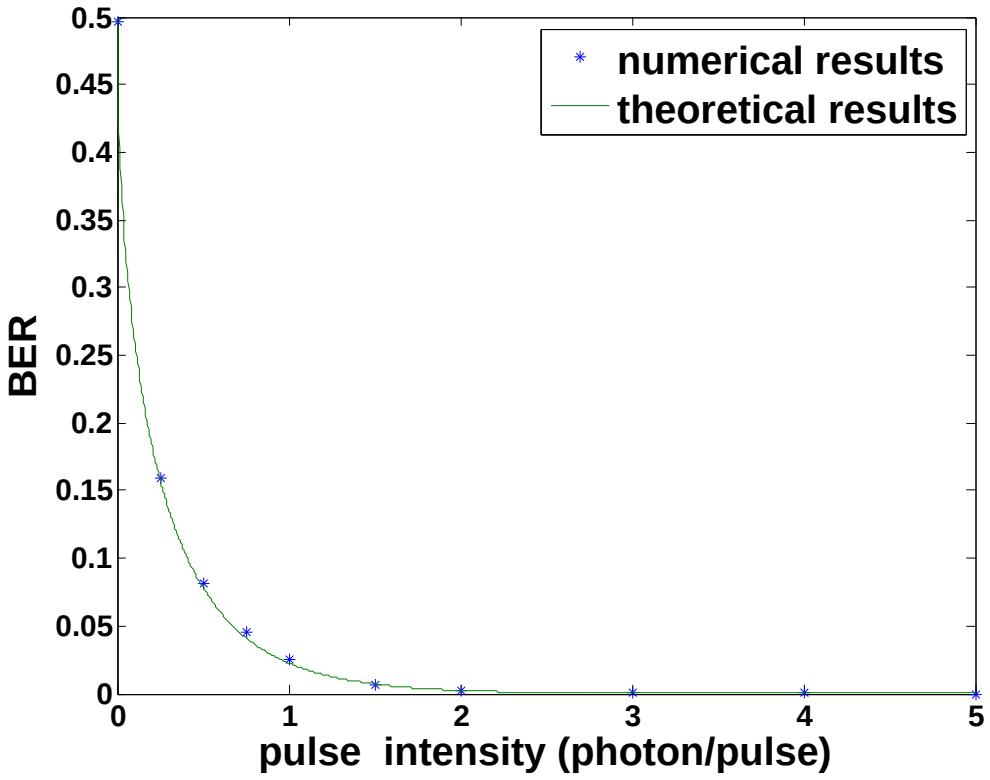


Figura 5.9: BER in function of α for the single homodyne setup. $X_0 = 0$ was used

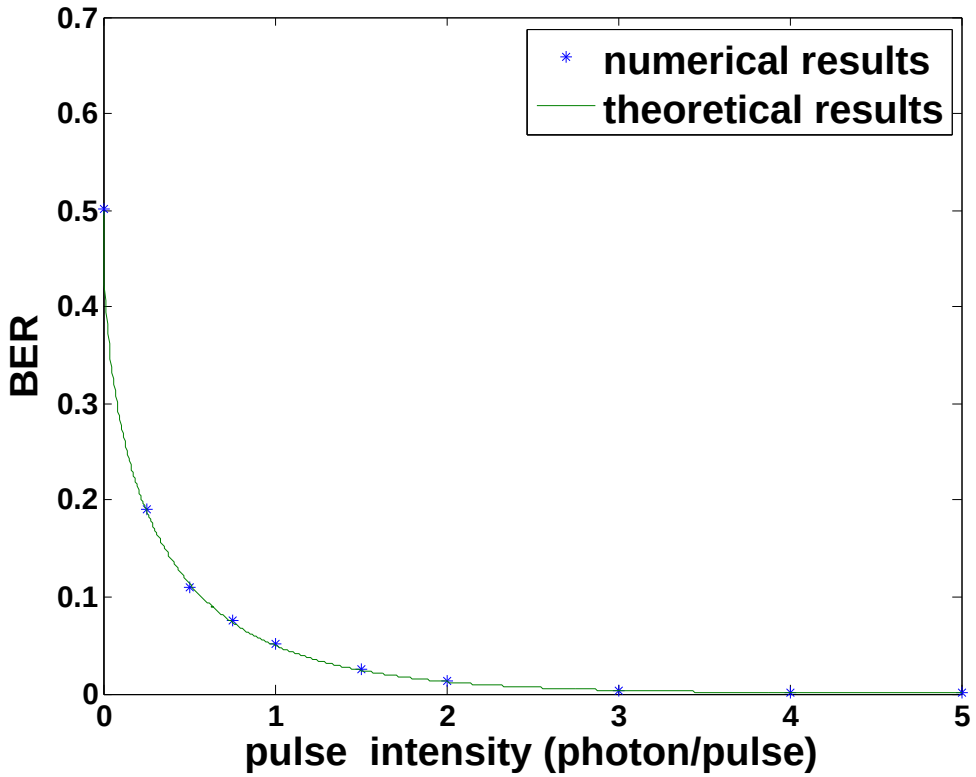


Figura 5.10: BER in function of α for the double homodyne setup. $X_0 = 0$ was used

Bibliografia

[1] Ryo Namiki and Takuya Hirano. Security of quantum cryptography using balanced homodyne detection. *Physical Review A*, 67(2):022308, 2003.

Capítulo 6

Library

6.1 Add

Input Parameters

This block takes no parameters.

Functional Description

This block accepts two signals and outputs one signal built from a sum of the two inputs. The input and output signals must be of the same type.

Input Signals

Number: 2

Type: Real, Complex or Complex_XY signal (ContinuousTimeContinuousAmplitude)

Output Signals

Number: 1

Type: Real, Complex or Complex_XY signal (ContinuousTimeContinuousAmplitude)

6.2 Binary source

This block generates a sequence of binary values (1 or 0) and it can work in four different modes:

1. Random
2. PseudoRandom
3. DeterministicCyclic
4. DeterministicAppendZeros

This blocks doesn't accept any input signal. It produces any number of output signals.

Input Parameters

Parameter: mode{PseudoRandom}
(Random, PseudoRandom, DeterministicCyclic, DeterministicAppendZeros)

Parameter: probabilityOfZero{0.5}
(real $\in$ [0,1])

Parameter: patternLength{7}
(integer $\in$ [1,32])

Parameter: bitStream{"0100011101010101"}
(string of 0's and 1's)

Parameter: numberOfBits{-1}
(long int)

Parameter: bitPeriod{1.0/100e9}
(double)

Methods

BinarySource(vector⟨Signal *⟩ &InputSig, vector⟨Signal *⟩ &OutputSig) :Block(InputSig, OutputSig){};

void initialize(void);

bool runBlock(void);

void setMode(BinarySourceMode m) BinarySourceMode const getMode(void)

void setProbabilityOfZero(double pZero)

double const getProbabilityOfZero(void)

void setBitStream(string bStream)

string const getBitStream(void)

void setNumberOfBits(long int nOfBits)

long int const getNumberOfBits(void)

void setPatternLength(int pLength)

int const getPatternLength(void)

void setBitPeriod(double bPeriod)

double const getBitPeriod(void)

Functional description

The *mode* parameter allows the user to select between one of the four operation modes of the binary source.

Random Mode Generates a 0 with probability *probabilityOfZero* and a 1 with probability 1-*probabilityOfZero*.

Pseudorandom Mode Generates a pseudorandom sequence with period $2^{patternLength} - 1$.

DeterministicCyclic Mode Generates the sequence of 0's and 1's specified by *bitStream* and then repeats it.

DeterministicAppendZeros Mode Generates the sequence of 0's and 1's specified by *bitStream* and then it fills the rest of the buffer space with zeros.

Input Signals

Number: 0

Type: Binary (DiscreteTimeDiscreteAmplitude)

Output Signals

Number: 1 or more

Type: Binary (DiscreteTimeDiscreteAmplitude)

Examples

Random Mode

PseudoRandom Mode As an example consider a pseudorandom sequence with *patternLength*=3 which contains a total of 7 ($2^3 - 1$) bits. In this sequence it is possible to find every combination of 0's and 1's that compose a 3 bit long subsequence with the exception of 000. For this example the possible subsequences are 010, 110, 101, 100, 111, 001 and 100 (they appear in figure 6.1 numbered in this order). Some of these require wrap.

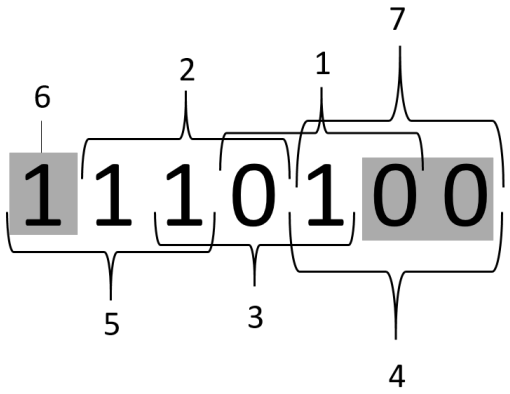


Figura 6.1: Example of a pseudorandom sequence with a pattern length equal to 3.

DeterministicCyclic Mode As an example take the *bit stream* '0100011101010101'. The generated binary signal is displayed in.

DeterministicAppendZeros Mode Take as an example the *bit stream* '0100011101010101'. The generated binary signal is displayed in 6.2.

Sugestions for future improvement

Implement an input signal that can work as trigger.

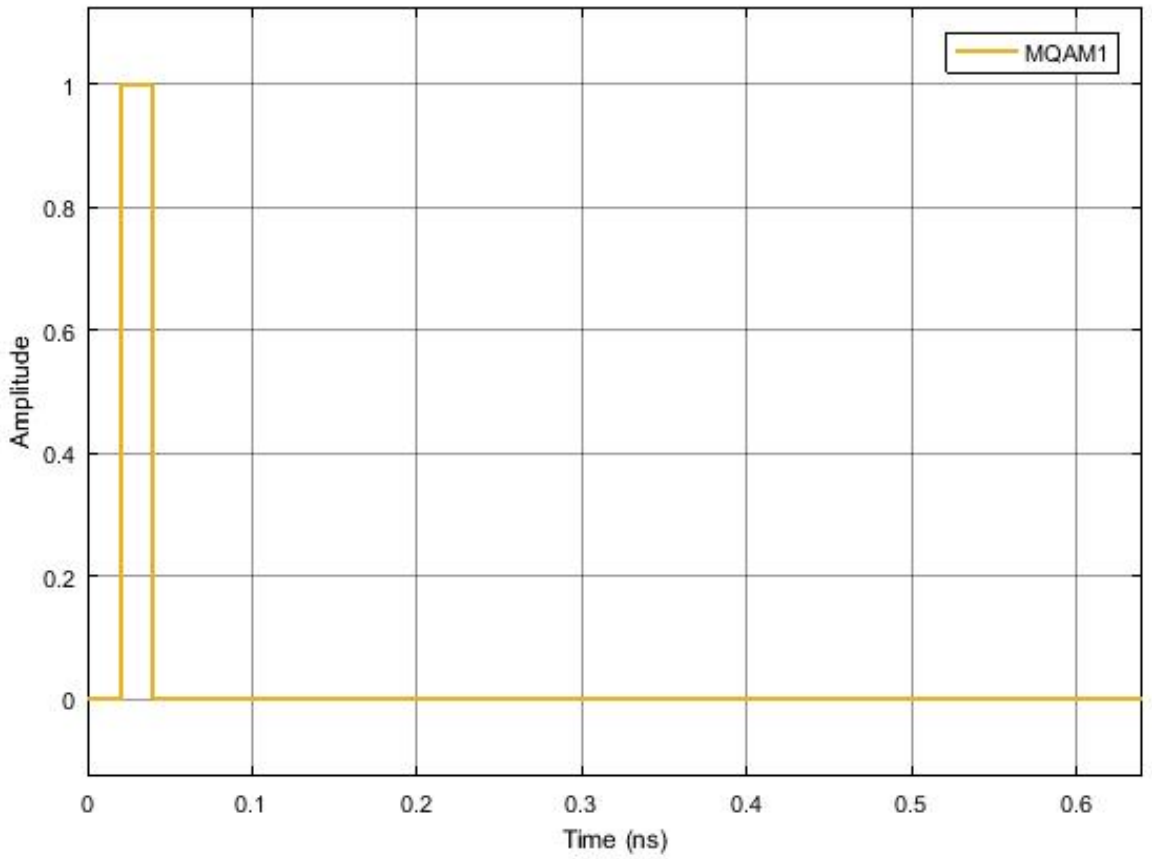


Figura 6.2: Binary signal generated by the block operating in the *Deterministic Append Zeros* mode with a binary sequence 01000...

6.3 Clock

This block doesn't accept any input signal. It outputs one signal that corresponds to a sequence of Dirac's delta functions with a user defined *period*.

Input Parameters

Parameter: period{ 0.0 };

Parameter: samplingPeriod{ 0.0 };

Methods

Clock()

Clock(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig, OutputSig)

void initialize(void)

bool runBlock(void)

void setClockPeriod(double per)

void setSamplingPeriod(double sPeriod)

Functional description

Input Signals

Number: 0

Output Signals

Number: 1

Type: Sequence of Dirac's delta functions. (TimeContinuousAmplitudeContinuousReal)

Examples

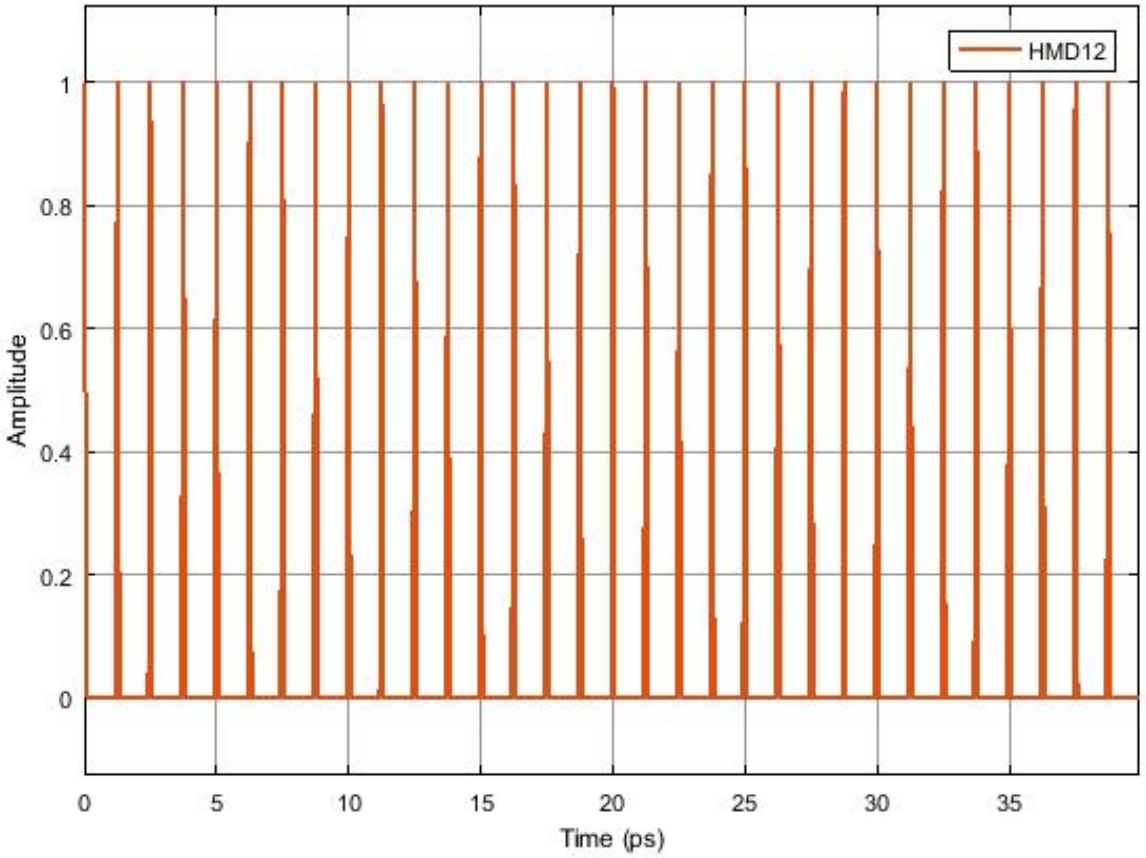


Figura 6.3: Example of the output signal of the clock

Sugestions for future improvement

6.4 Decoder

This block accepts a complex electrical signal and outputs a sequence of binary values (0's and 1's). Each point of the input signal corresponds to a pair of bits.

Input Parameters

Parameter: `t_integer` m{ 4 }

Parameter: vector<`t_complex`> iqAmplitudes{ { 1.0, 1.0 },{ -1.0, 1.0 },{ -1.0, -1.0 },{ 1.0, -1.0 } };

Methods

Decoder()

Decoder(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig, OutputSig)

void initialize(void)

bool runBlock(void)

void setM(int mValue)

void getM()

void setIqAmplitudes(vector<`t_iqValues`> iqAmplitudesValues)

vector<`t_iqValues`>getIqAmplitudes()

Functional description

This block makes the correspondence between a complex electrical signal and pair of binary values using a predetermined constellation.

To do so it computes the distance in the complex plane between each value of the input signal and each value of the *iqAmplitudes* vector selecting only the shortest one. It then converts the point in the IQ plane to a pair of bits making the correspondence between the input signal and a pair of bits.

Input Signals

Number: 1

Type: Electrical complex (TimeContinuousAmplitudeContinuousReal)

Output Signals

Number: 1

Type: Binary

Examples

As an example take an input signal with positive real and imaginary parts. It would correspond to the first point of the *iqAmplitudes* vector and therefore it would be associated to the pair of bits 00.

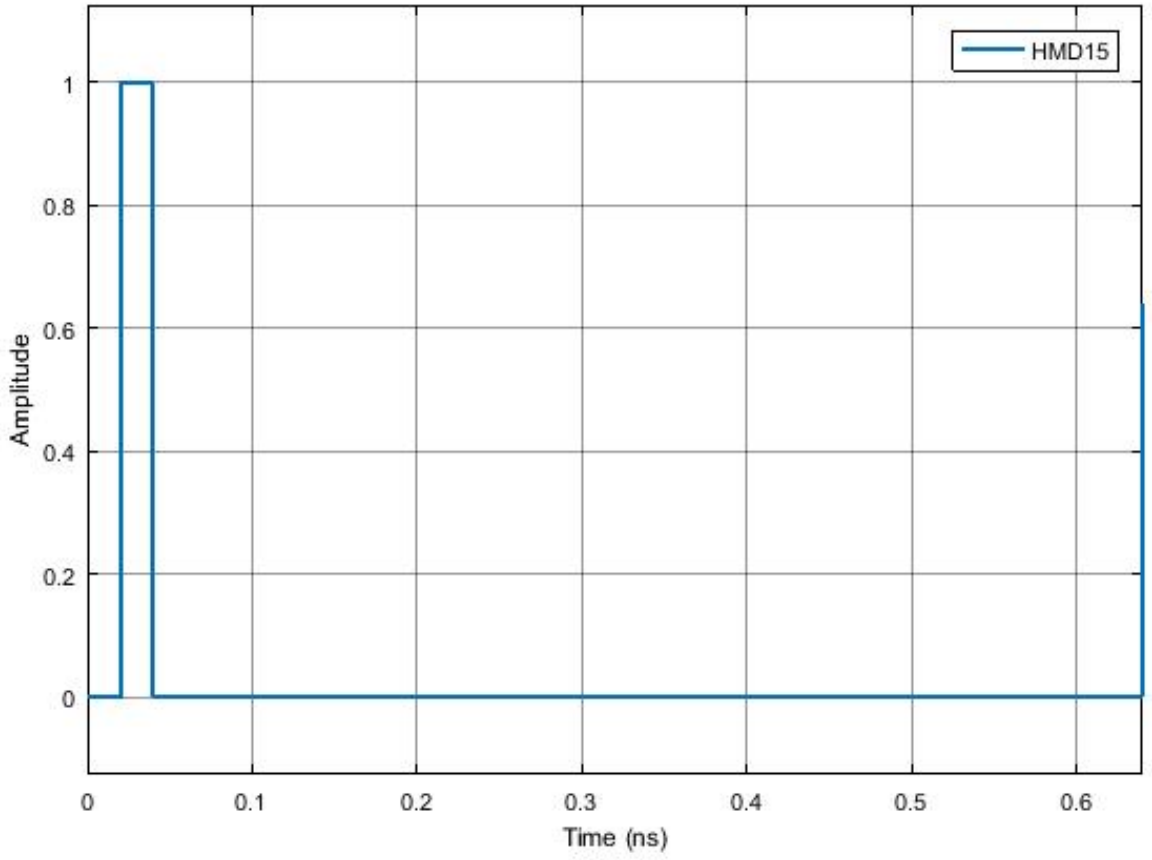


Figura 6.4: Example of the output signal of the decoder for a binary sequence 01. As expected it reproduces the initial bit stream

Sugestions for future improvement

6.5 Discrete to continuous time

This block converts a signal discrete in time to a signal continuous in time. It accepts one input signal that is a sequence of 1's and -1's and it produces one output signal that is a sequence of Dirac delta functions.

Input Parameters

Parameter: numberOfSamplesPerSymbol{8}
(int)

Methods

DiscreteToContinuousTime(vector<Signal *> &inputSignals, vector<Signal *> &outputSignals) :Block(inputSignals, outputSignals){};

void initialize(void);

bool runBlock(void);

void setNumberOfSamplesPerSymbol(int nSamplesPerSymbol)

int const getNumberOfSamplesPerSymbol(void)

Functional Description

This block reads the input signal buffer value, puts it in the output signal buffer and it fills the rest of the space available for that symbol with zeros. The space available in the buffer for each symbol is given by the parameter *numberOfSamplesPerSymbol*.

Input Signals

Number : 1

Type : Sequence of 1's and -1's. (DiscreteTimeDiscreteAmplitude)

Output Signals

Number : 1

Type : Sequence of Dirac delta functions (ContinuousTimeDiscreteAmplitude)

Example

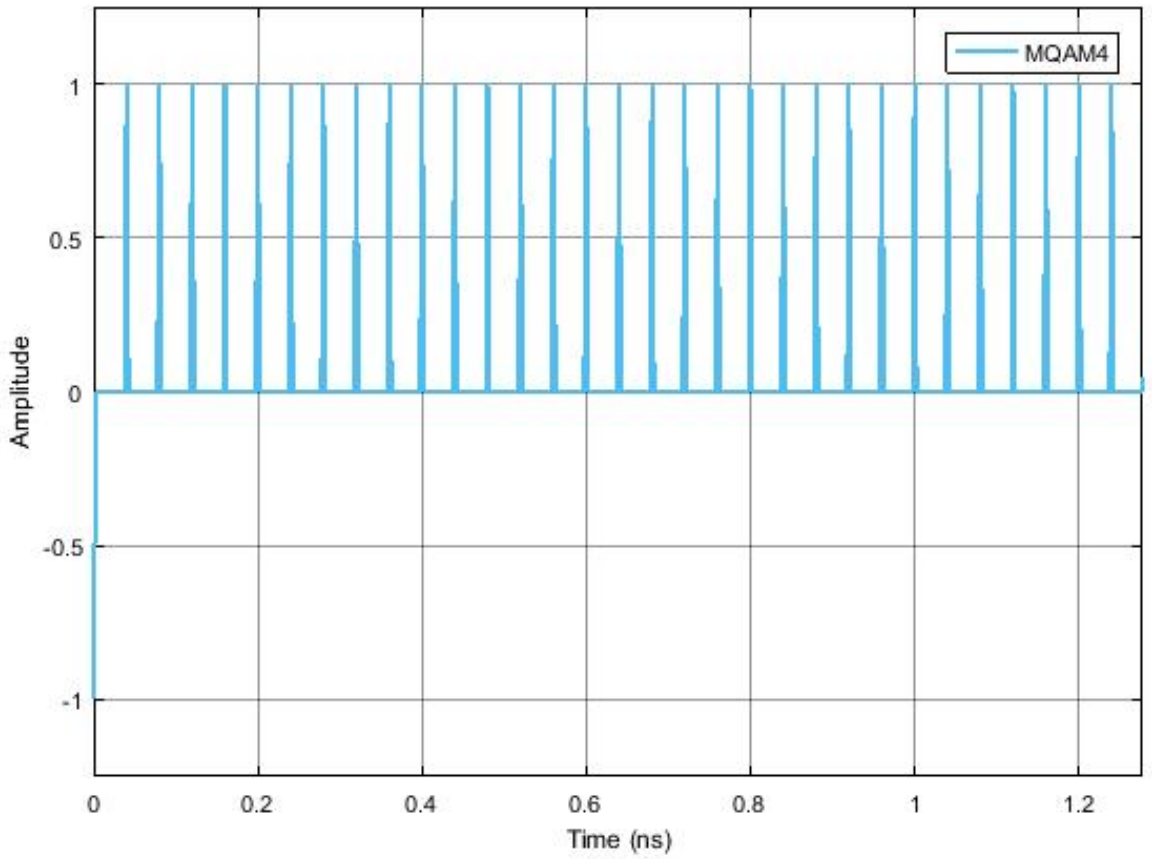


Figura 6.5: Example of the type of signal generated by this block for a binary sequence 0100...

6.6 Homodyne receiver

This block of code simulates the reception and demodulation of an optical signal (which is the input signal of the system) outputing a binary signal. A simplified schematic representation of this block is shown in figure 6.6.

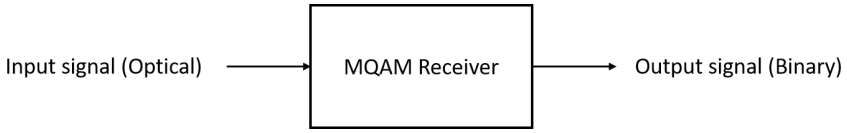


Figura 6.6: Basic configuration of the MQAM receiver

Functional description

This block accepts one optical input signal and outputs one binary signal that corresponds to the M-QAM demodulation of the input signal. It is a complex block (as it can be seen from figure 6.7) of code made up of several simpler blocks whose description can be found in the *lib* repository.

In can also be seen from figure 6.7 that there's an extra internal (generated inside the homodyne receiver block) input signal generated by the *Clock*. This block is used to provide the sampling frequency to the *Sampler*.

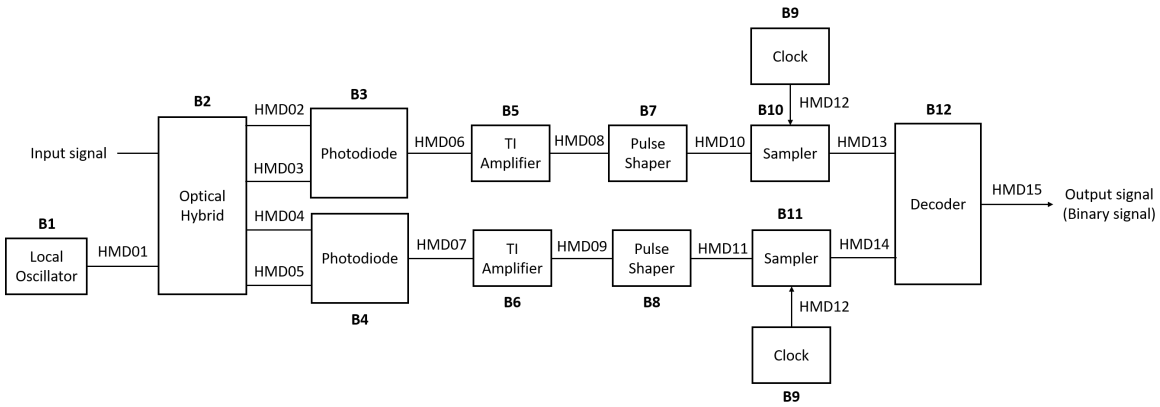


Figura 6.7: Schematic representation of the block homodyne receiver.

Input parameters

This block has some input parameters that can be manipulated by the user in order oto change the basic configuration of the receiver. Each parameter has associated a function that allows for its change. In the following table (table 6.2) the input parameters and corresponding functions are summarized.

Input parameters	**Function**	Type	**Accepted values**
IQ amplitudes	setIqAmplitudes	Vector of coordinate points in the I-Q plane	**Example** for a 4-qam mapping: { { 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 } }
Local oscillator power (in dBm)	setLocalOscillatorOpticalPower_dBm	double(t_real)	Any double greater than zero
Local oscillator phase	setLocalOscillatorPhase	double(t_real)	Any double greater than zero
Responsivity of the photodiodes	setResponsivity	double(t_real)	$\in$ [0,1]
Amplification (of the TI amplifier)	setAmplification	double(t_real)	Positive real number
Noise amplitude (introduced by the TI amplifier)	setNoiseAmplitude	double(t_real)	Real number greater than zero
Samples to skipe	setSamplesToSkip	int(t_integer)	
Save internal signals	setSaveInternalSignals	bool	True or False
Sampling period	setSamplingPeriod	double	Givem by *symbolPeriod/samplesPerSy*

Tabela 6.1: List of input parameters of the block MQAM receiver

Methods

HomodyneReceiver(vector<Signal *> &inputSignal, vector<Signal *> &outputSignal) (**constructor**)

void setIqAmplitudes(vector<t_iqValues> iqAmplitudesValues)

vector<t_iqValues> const getIqAmplitudes(void)

void setLocalOscillatorSamplingPeriod(double sPeriod)

void setLocalOscillatorOpticalPower(double opticalPower)

void setLocalOscillatorOpticalPower_dBm(double opticalPower_dBm)

void setLocalOscillatorPhase(double lOscillatorPhase)

void setLocalOscillatorOpticalWavelength(double lOscillatorWavelength)

void setSamplingPeriod(double sPeriod)

void setResponsivity(t_real Responsivity)

void setAmplification(t_real Amplification)

void setNoiseAmplitude(t_real NoiseAmplitude)

void setImpulseResponseTimeLength(int impResponseTimeLength)

void setFilterType(PulseShaperFilter fType)

void setRollOffFactor(double rOffFactor)

void setClockPeriod(double per)

void setSamplesToSkip(int sToSkip)

Input Signals

Number: 1

Type: Optical signal

Output Signals

Number: 1

Type: Binary signal

Example

Sugestions for future improvement

6.7 IQ modulator

This blocks accepts one inupt signal continuous in both time and amplitude and it can produce either one or two output signals. It generates an optical signal and it can also generate a binary signal.

Input Parameters

Parameter: outputOpticalPower{1e-3}
(double)

Parameter: outputOpticalWavelength{1550e-9}
(double)

Parameter: outputOpticalFrequency{speed_of_light/outputOpticalWavelength}
(double)

Methods

IqModulator(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig, OutputSig){};

void initialize(void);

bool runBlock(void);

void setOutputOpticalPower(double outOpticalPower)

void setOutputOpticalPower_dBm(double outOpticalPower_dBm)

void setOutputOpticalWavelength(double outOpticalWavelength)

void setOutputOpticalFrequency(double outOpticalFrequency)

Functional Description

This block takes the two parts of the signal: in phase and in amplitude and it combines them to produce a complex signal that contains information about the amplitude and the phase.

This complex signal is multiplied by $\frac{1}{2}\sqrt{outputOpticalPower}$ in order to reintroduce the information about the energy (or power) of the signal. This signal corresponds to an optical signal and it can be a scalar or have two polarizations along perpendicular axis. It is the signal that is transmited to the receptor.

The binary signal is sent to the Bit Error Rate (BER) meaurement block.

Input Signals

Number : 2

Type : Sequence of impulses modulated by the filter (ContinuousTimeContiousAmplitude))

Output Signals

Number : 1 or 2

Type : Complex signal (optical) (ContinuousTimeContinuousAmplitude) and binary signal (DiscreteTimeDiscreteAmplitude)

Example

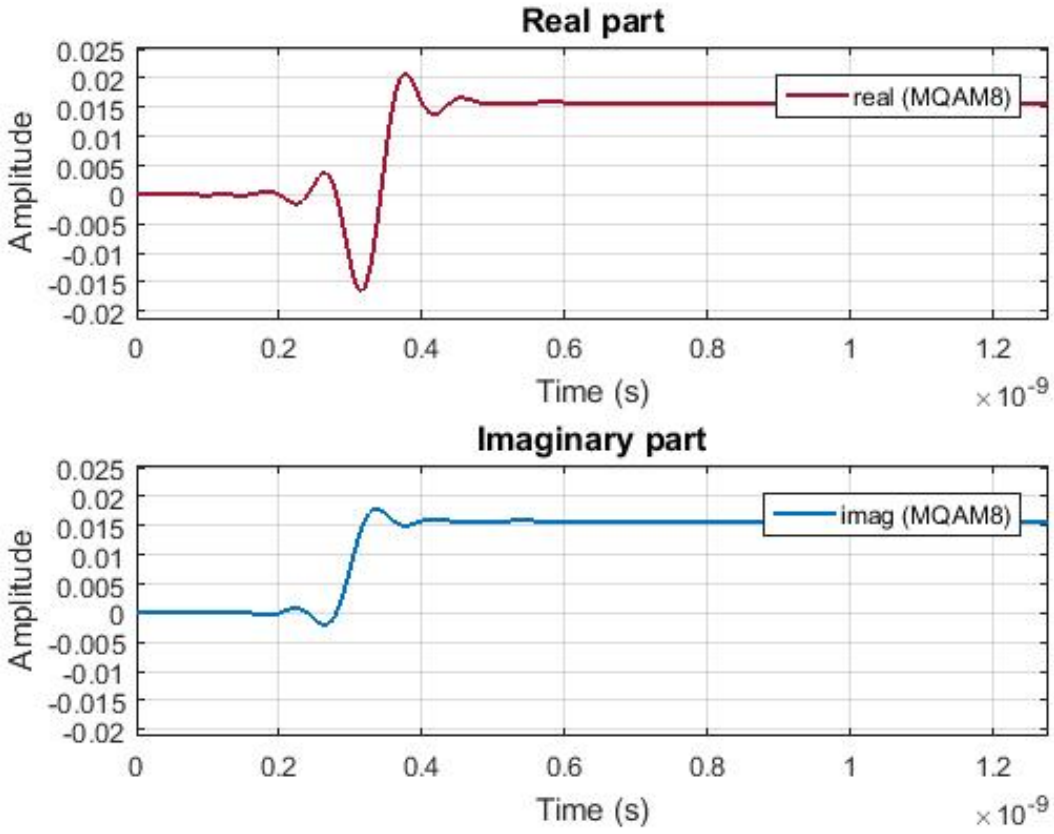


Figura 6.8: Example of a signal generated by this block for the initial binary signal 0100...

6.8 Local Oscillator

This block simulates a local oscillator which can have shot noise or not. It produces one output complex signal and it doesn't accept input signals.

Input Parameters

Parameter: opticalPower{ 1e-3 }

Parameter: wavelength{ 1550e-9 }

Parameter: frequency{ SPEED_OF_LIGHT / wavelength }

Parameter: phase{ 0 }

Parameter: samplingPeriod{ 0.0 }

Parameter: shotNoise{ false }

Methods

LocalOscillator()

LocalOscillator(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig, OutputSig){};

void initialize(void);

bool runBlock(void);

void setSamplingPeriod(double sPeriod);

void setOpticalPower(double oPower);

void setOpticalPower_dBm(double oPower_dBm);

void setWavelength(double wlength);

void setPhase(double lOscillatorPhase);

void setShotNoise(bool sNoise);

Functional description

This block generates a complex signal with a specified phase given by the input parameter *phase*.

It can have shot noise or not which corresponds to setting the *shotNoise* parameter to True or False, respectively. If there isn't shot noise the the output of this block is given by $0.5 * \sqrt{OpticalPower} * ComplexSignal$. If there's shot noise then a random gaussian distributed noise component is added to the *OpticalPower*.

Input Signals

Number: 0

Output Signals

Number: 1

Type: Optical signal

Examples

Sugestions for future improvement

6.9 MQAM mapper

This block does the mapping of the binary signal using a *m*-QAM modulation. It accepts one input signal of the binary type and it produces two output signals which are a sequence of 1's and -1's.

Input Parameters

Parameter: m{4}
(m should be of the form 2^n with n integer)

Parameter: iqAmplitudes{{ 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 }}

Methods

MQamMapper(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig, OutputSig) {};

void initialize(void);

bool runBlock(void);

void setM(int mValue);

void setIqAmplitudes(vector<t_iqValues> iqAmplitudesValues);

Functional Description

In the case of m=4 this block atributes to each pair of bits a point in the I-Q space. The constellation used is defined by the *iqAmplitudes* vector. The constellation used in this case is ilustrated in figure 6.9.

Input Signals

Number : 1

Type : Binary (DiscreteTimeDiscreteAmplitude)

Output Signals

Number : 2

Type : Sequence of 1's and -1's (DiscreteTimeDiscreteAmplitude)

Example

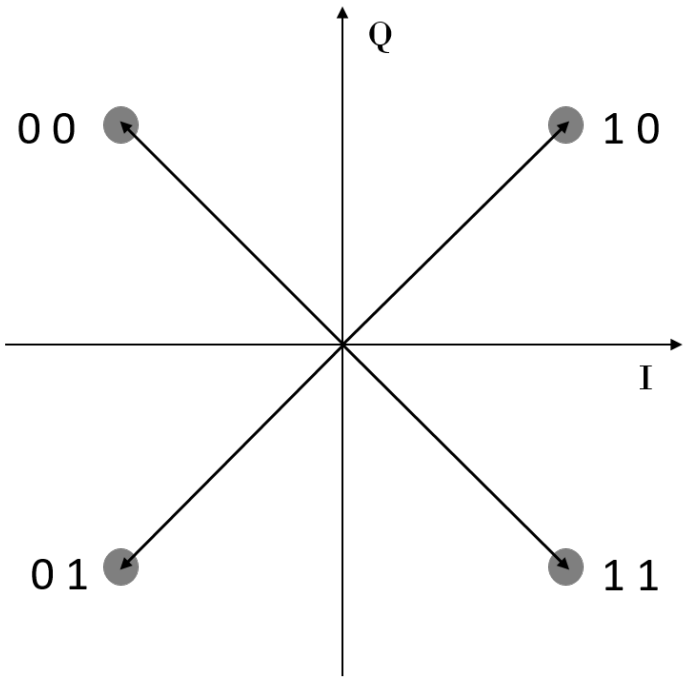


Figura 6.9: Constellation used to map the signal for m=4

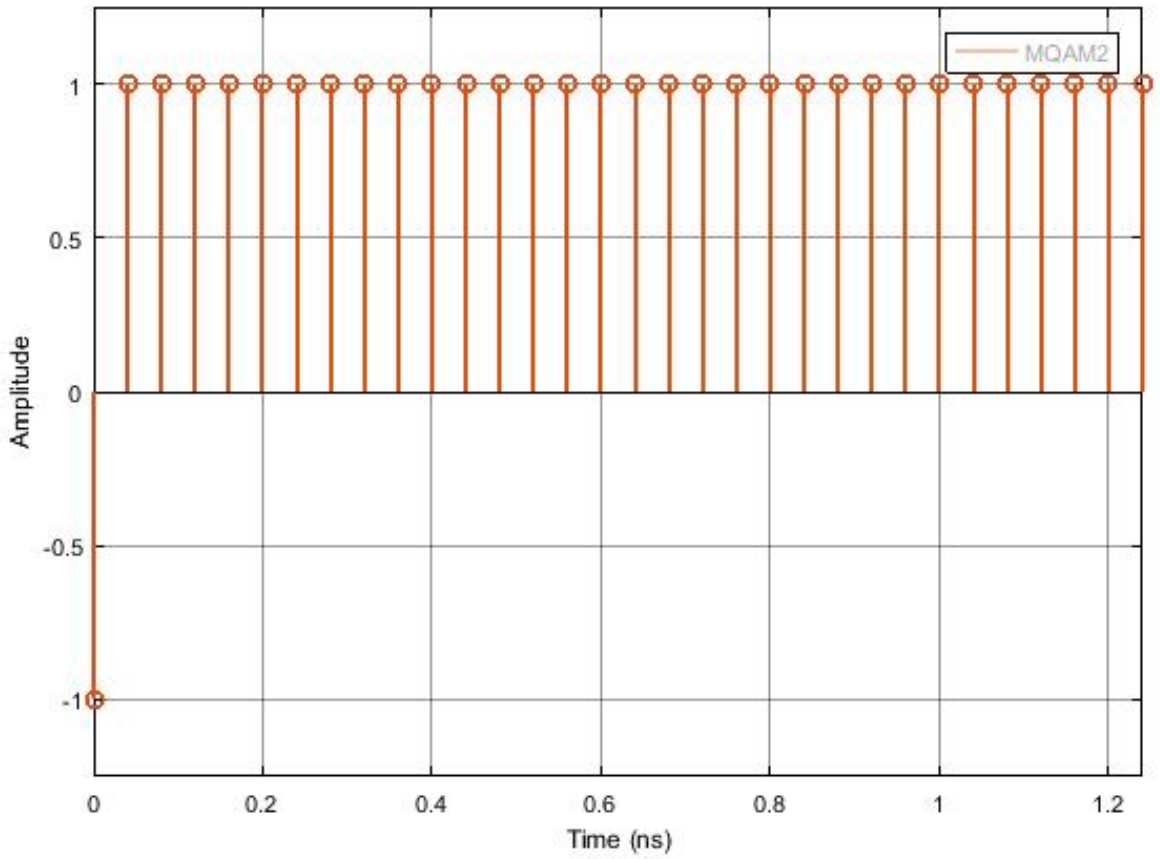


Figura 6.10: Example of the type of signal generated by this block for the initial binary signal 0100...

6.10 MQAM transmitter

This block generates a MQAM optical signal. It can also output the binary sequence. A schematic representation of this block is shown in figure 6.11.

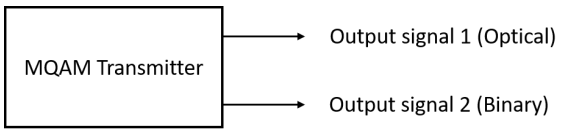


Figura 6.11: Basic configuration of the MQAM transmitter

Functional description

This block generates an optical signal (output signal 1 in figure 6.12). The binary signal generated in the internal block Binary Source (block B1 in figure 6.12) can be used to perform a Bit Error Rate (BER) measurement and in that sense it works as an extra output signal (output signal 2 in figure 6.12).

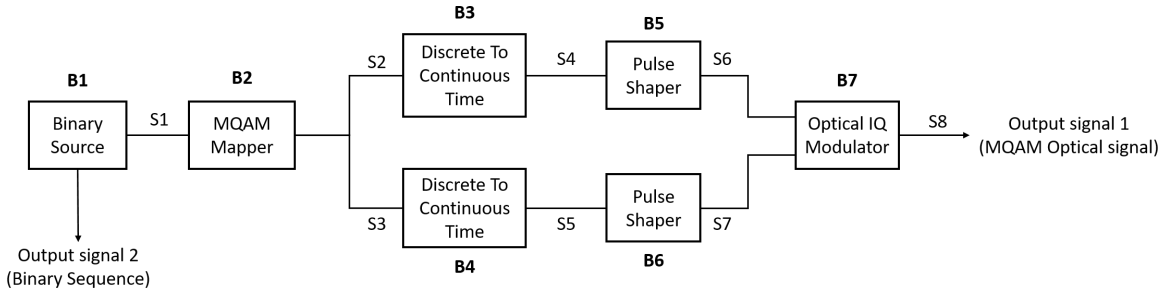


Figura 6.12: Schematic representation of the block MQAM transmitter.

Input parameters

This block has a special set of functions that allow the user to change the basic configuration of the transmitter. The list of input parameters, functions used to change them and the values that each one can take are summarized in table 6.2.

Input parameters	**Function**	Type	**Accepted values**
Mode	setMode()	string	PseudoRandom Random DeterministicAppendZeros DeterministicCyclic
Number of bits generated	setNumberOfBits()	int	Any integer
Pattern length	setPatternLength()	int	Real number greater than zero
Number of bits	setNumberOfBits()	long	Integer number greater than zero
Number of samples per symbol	setNumberOfSamplesPerSymbol()	int	Integer number of the type 2^n with n also integer
Roll of factor	setRollOfFactor()	double	$\in$ [0,1]
IQ amplitudes	setIqAmplitudes()	Vector of coordinate points in the I-Q plane	**Example** for a 4-qam mapping: { { 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 } }
Output optical power	setOutputOpticalPower()	int	Real number greater than zero
Save internal signals	setSaveInternalSignals()	bool	True or False

Tabela 6.2: List of input parameters of the block MQAM transmitter

Methods

MQamTransmitter(vector<Signal *> &inputSignal, vector<Signal *> &outputSignal); (**constructor**)

void set(int opt);

void setMode(BinarySourceMode m)

BinarySourceMode const getMode(void)

void setProbabilityOfZero(double pZero)

double const getProbabilityOfZero(void)

void setBitStream(string bStream)

string const getBitStream(void)

void setNumberOfBits(long int nOfBits)

long int const getNumberOfBits(void)

void setPatternLength(int pLength)

int const getPatternLength(void)

void setBitPeriod(double bPeriod)

double const getBitPeriod(void)

void setM(int mValue) int const getM(void)

void setIqAmplitudes(vector<t_iqValues> iqAmplitudesValues)

vector<t_iqValues> const getIqAmplitudes(void)

void setNumberOfSamplesPerSymbol(int n)

int const getNumberOfSamplesPerSymbol(void)

void setRollOffFactor(double rOffFactor)

double const getRollOffFactor(void)

void setSeeBeginningOfImpulseResponse(bool sBeginningOfImpulseResponse)

double const getSeeBeginningOfImpulseResponse(void)

void setOutputOpticalPower(t_real outOpticalPower)

t_real const getOutputOpticalPower(void)

void setOutputOpticalPower_dBm(t_real outOpticalPower_dBm)

t_real const getOutputOpticalPower_dBm(void)

Output Signals

Number: 1 optical and 1 binary (optional)

Type: Optical signal

Example

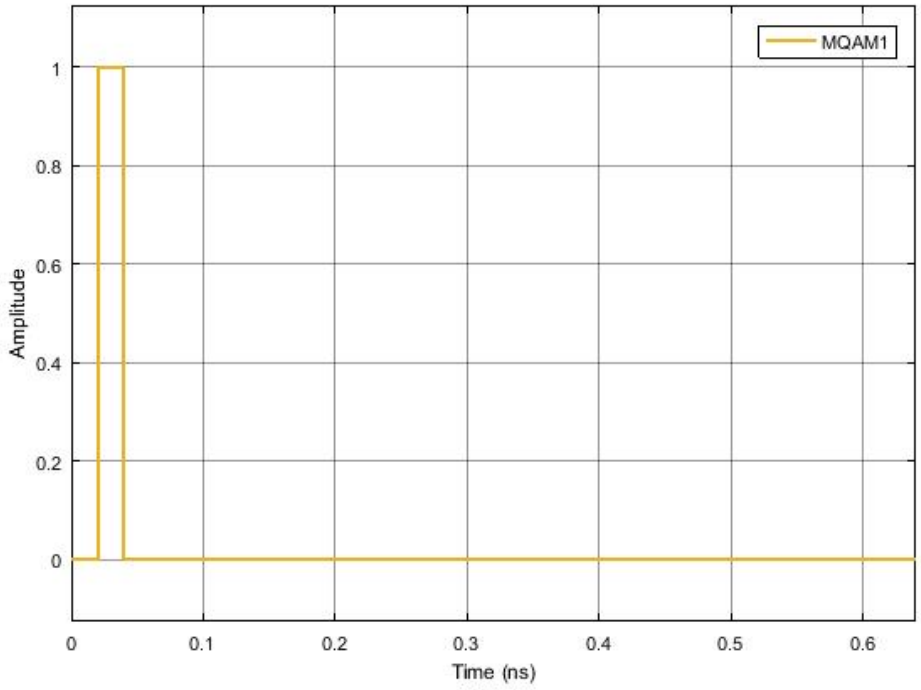


Figura 6.13: Example of the binary sequence generated by this block for a sequence 0100...

Sugestions for future improvement

Add to the system another block similar to this one in order to generate two optical signals with perpendicular polarizations. This would allow to combine the two optical signals and generate an optical signal with any type of polarization.

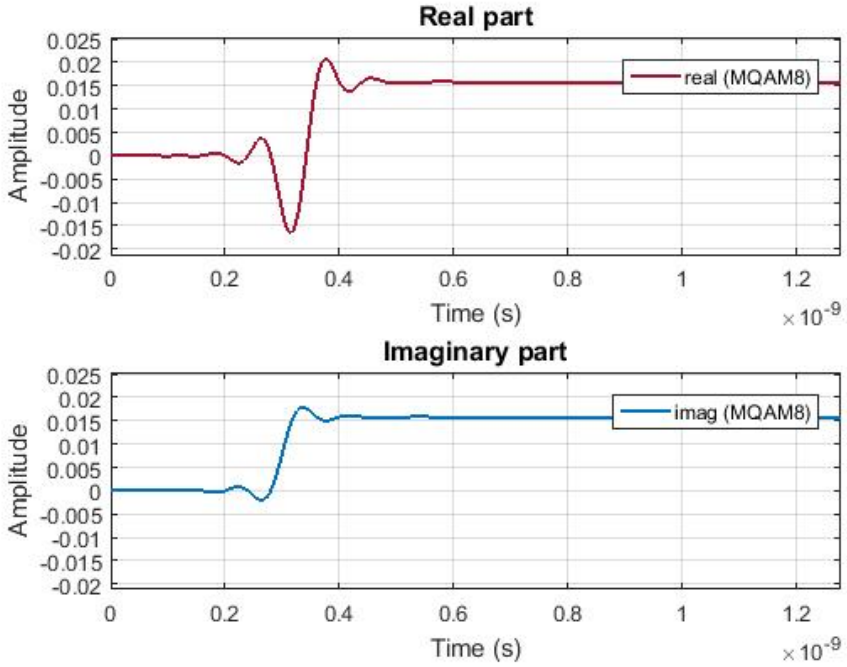


Figura 6.14: Example of the output optical signal generated by this block for a sequence 0100...