MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS Y DE CONTROL POR LA UNIVERSIDAD COMPLUTENSE DE MADRID Y LA UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Trabajo Fin de Máster

TÍTULO Desarrollo de una herramienta software interactiva en Sysquake para enseñar/aprender las características básicas de las señales PRBS y multiseno usadas en Identificación de Sistemas

AUTOR	Javier Esteban Escaño
DIRECTOR	José Manuel Díaz Martínez
CURSO	2021-2022
CONVOCATORIA	Septiembre

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS Y DE CONTROL POR LA UNIVERSIDAD COMPLUTENSE DE MADRID Y LA UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Desarrollo de una herramienta software interactiva en Sysquake para enseñar/aprender las características básicas de las señales PRBS y multiseno usadas en Identificación de Sistemas

Proyecto tipo A: Proyecto específico propuesto por un profesor

AUTOR	Javier Esteban Escaño
DIRECTOR	José Manuel Díaz Martínez
CURSO	2021-2022
CONVOCATORIA	Septiembre 2022

Espacio reservado para la hoja de calificaciones, que será introducida por la secretaría del Master después de la defensa del mismo

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Javier Esteban Escaño

ESTEBAN ESCAÑO JAVIER - 25430511D

Firmado digitalmente por ESTEBAN ESCAÑO JAVIER - 25430511D
DN: C=ES, SERIALNUMBER=IDCES-25430511D, G=JAVIER, SN=ESTEBAN ESCAÑO, CN=ESTEBAN ESCAÑO JAVIER - 25430511D
Razón: Soy el autor de este documento
Ubicación:
Fecha: 2022.09.08 03:51:57 +02'00'
Foxit PDF Reader Versión: 12.0.1

Firma del alumno

Resumen del proyecto

La obtención de modelos matemáticos de sistemas dinámicos tiene una gran importancia en muchas áreas de la ciencia y de la ingeniería. Éstos se suelen usar para simulación, control o predicción del comportamiento de procesos reales. Existen dos métodos fundamentales de obtención de estos modelos: la modelización matemática y la identificación de sistemas. En la *modelización matemática* se utilizan leyes físicas, químicas, económicas, etc., para describir la dinámica de un proceso o fenómeno. En la *identificación de sistemas* se somete al sistema a una serie de experimentos y se registran los datos de entrada y salida. Posteriormente, se escoge la estructura de un modelo y se ajustan sus parámetros con los datos experimentales medidos. Ambas formas de modelización no se deben ver como separadas o excluyentes. En muchos casos los procesos son tan complejos que no es posible obtener un modelo usando únicamente principios físicos. En tal caso se requiere obligatoriamente el uso de técnicas de identificación. No obstante, para la elección de estas técnicas es importante todo el conocimiento físico previo que se tenga del sistema. También puede ocurrir que se obtenga un modelo a partir del análisis físico del sistema, pero existan parámetros que no se conozcan y que puedan ser estimados mediante identificación.

La identificación de sistemas es una materia transversal en el área de Ingeniería de Sistemas y Automática. En un primer acercamiento a esta materia se deben estudiar los fundamentos básicos tanto teóricos como prácticos de la identificación de sistemas. Algunos de estos fundamentos resultan complicados de aprender para los estudiantes de últimos cursos de grado o de máster. Para facilitar su aprendizaje, el uso de herramientas software interactivas puede resultar de gran utilidad, ya que está demostrado que el uso de estas herramientas mejora el aprendizaje de conceptos.

Para poder identificar un sistema, es necesario determinar la relación matemática entre la entrada y salida del proceso. Por tanto, se precisa de una señal de entrada adecuada para excitar el sistema y descubrir su comportamiento de la manera más precisa posible. Este TFM se centrará en el estudio de las señales de entrada utilizadas para la identificación de sistemas y en el diseño de una herramienta interactiva que permita su generación de una forma simple e intuitiva.

El primer paso en la metodología de Identificación de Sistemas consiste en diseñar las señales de entrada con que se va excitar un sistema para la obtención de datos de entrada y salida. La señal PRBS y la señal multiseno, son los dos tipos de señales más utilizadas para esta tarea. Por consiguiente, el estudiante debe conocer y comprender cuales son las principales características en el dominio temporal y frecuencial de estos dos tipos de señales.

En este TFM se presenta el diseño y desarrollo en Sysquake de una herramienta software interactiva que permite enseñar/aprender las características básicas en el dominio del tiempo y de la frecuencia de las señales PRBS y multiseno. También se han incluido ejemplos de uso de la herramienta y un manual de usuario para conocer su funcionamiento, así como su código fuente para que pueda ser ampliado y mejorado. Por último, se han establecido líneas de mejora futura de la herramienta.

Abstract

Obtaining mathematical models of dynamic systems have relevant importance in many areas of science and engineering. These are often used for simulation, control or prediction of the behaviour of actual processes. There are two fundamental methods for obtaining these models: mathematical modelling and system identification. In mathematical modelling, physical, chemical, economic laws, etc., are used in order to describe the dynamics of a process or phenomenon. In system identification, the system is exposed to a series of experiments and the input and output

data are recorded. Subsequently, the structure of a model is chosen and its parameters are adjusted with the measured experimental data. Both forms of modelling should not be seen as separate or exclusive. In many cases, the processes are so complex that it is not possible to obtain a model using only physical principles. In this case, the use of identification techniques is mandatory. However, all prior physical knowledge of the system is key for these techniques. It can also happen that a model is found from the physical analysis of the system, but some parameters are unknown and they can be estimated through identification techniques.

The identification of systems is a transversal subject in the area of Systems Engineering and Automation. In the first approach to this matter, the elementary theoretical and practical foundations of systems identification should be studied. Some of these fundaments are difficult to learn for students in their final bachelor's or master's courses. To facilitate their learning, the use of interactive software tools can be very useful, since it has been shown that the use of these tools improves the learning of these ideas.

To identify a system, it is necessary to determine the mathematical relationship between the input and output of the process. Therefore, a suitable input signal is needed to excite the system and discover its behaviour as precisely as possible. This TFM will focus on the study of the input signals used for the identification of systems and on the design of an interactive tool that allows their generation simply and intuitively.

The first step in the System Identification methodology consists in the designing of the input signals with which a system will be excited to obtain input and output data. The PRBS and the multisine signal are the two types of signals most frequently used for this task. Therefore, the student must know and understand what the main characteristics in the temporal and frequency domain of these two types of signals are.

This TFM presents the design and development in Sysquake of an interactive software tool that allows teaching/learning of the fundamental characteristics in the time and frequency domain of PRBS and multisine signals. Examples of how to use the tool and a user manual have also been included to learn how it works and its source code so it can be expanded and improved. Finally, future improvement lines for the tool have been established.

Lista de palabras clave

Identificación de sistemas, Sysquake, señales PRBS, señales multiseno.

Contenido

Autorización iv
Resumen del proyecto v
Abstract v
Lista de palabras clave vi
Lista de figuras ix
Lista de tablas xi
Lista de acrónimos xiii
1 Introducción. 1
1.1 Motivación del proyecto. 1
1.2 Objetivos del proyecto. 2
1.3 Estructura del proyecto. 3
2 Aprendizaje de la materia Identificación de Sistemas. 4
2.1 Identificación de sistemas. 4
2.1.1 Elementos que intervienen en la identificación 6
2.1.2 Hechos relacionados con la identificación 6
2.1.3 Tipos de modelos 7
2.1.4 Pasos a seguir durante la identificación de un sistema 8
2.2 Aprendizaje con metodologías basadas en la interactividad. 10
2.2.1 Técnicas activas que mejoran la retención de los estudiantes 10
2.2.2 La enseñanza interactiva en la ingeniería de control 11
2.2.3 Metodología de aprendizaje interactivo 14
2.3 Estado del arte. 15
3 Diseño de la señal de excitación: señales PRBS y señales multiseno. 17
3.1 Importancia de las señales de entrada en la identificación de sistemas. 17
3.1.1 Tipos de señales 18
3.1.2 Índices de calidad de las señales de perturbación 19
3.2 Introducción a la variable aleatoria 21
3.2.1 Concepto de variable aleatoria 21
3.2.2 Funciones de distribución de probabilidad 22
3.2.3 Funciones de densidad de probabilidad 23
3.2.4 Valores medios y momentos 23
3.3 Introducción a los procesos aleatorios 25
3.3.1 Proceso aleatorio estacionario y no estacionario 25
3.3.2 Proceso ergódico y no ergódico 25
3.4 Introducción a las funciones de correlación 26
3.4.1 Propiedades de las funciones de autocorrelación para señales estacionarias 28
3.4.2 Cálculo de funciones de correlación en la práctica 29
3.5 Introducción a la transformada y series de Fourier para señales continuas 31
3.5.1 Señales no periódicas 31
3.5.2 Señales periódicas 32
3.6 Introducción a la transformada y series de Fourier para señales discretas 33
3.6.1 Transformada discreta de Fourier 33
3.6.2 El problema del aliasing y del spectral leakage. 36
3.7 Señales PRBS. 38
3.7.1 Propiedades de las señales PRBS 38
3.7.2 Generación de las señales PRBS 48

3.7.3 Otros tipos de señales PRBS 51
3.7.4 Consideraciones sobre el diseño de señales PRBS....52
3.8 Señales multiseno. 53
3.8.1 Elección de la fase inicial de la señal multiseno 54
3.8.2 Función de autocorrelación de una señal multiseno....57
3.8.3 Espectro de potencia de una señal multiseno 59
4 Diseño e implementación de la herramienta....61
4.1 Generación de señales PRBS....61
4.1.1 Generación de señales PRBS usando Matlab....61
4.1.2 Señales PRBS generadas por Matlab y Sysquake 65
4.1.3 Generación de señales PRBS usando Sysquake....71
4.2 Generación de señales multiseno....74
4.2.1 Cálculo de las señales multiseno usando Matlab....74
4.2.2 Definición de la señal multiseno en el dominio del tiempo 76
4.2.3 Generación de señales multiseno usando Sysquake....80
4.3 Estructura del programa en Sysquake 82
4.3.1 Diagramas de casos de uso....83
4.3.3 Diagramas de clases....121
5 Ejemplos de uso de la herramienta. 127
5.1 Generación de señales PRBS....127
5.2 Aplicación de una señal PRBS a la identificación de sistemas 131
5.3 Generación de señales multiseno....132
5.4 Aplicación de una señal multiseno a la identificación de un sistema....134
5.5 Diferencias de la señal usando distintas fases iniciales....136
6 Conclusiones y trabajos futuros....139
6.1 Trabajos futuros....139
Bibliografía....141
7 Apéndice A: Manual de uso de la herramienta....148
7.1 Menús de la aplicación 148
7.2 Señal PRBS 150
7.3 Señal multiseno 159
8 Apéndice B: Código fuente de la herramienta....167

Lista de figuras

Figura 1: Esquema básico de la identificación de sistemas 2
Figura 2: Proceso de identificación de un sistema 5
Figura 3: Sistema identificado 6
Figura 4: Pasos a seguir para la identificación del modelo 8
Figura 5: Sistema de muestreo 9
Figura 6: función temporal aleatoria. 22
Figura 7: Ejemplo de función de distribución de probabilidad 23
Figura 8: Señal de banda limitada. 37
Figura 9: Ancho de banda de la señal muestreada 37
Figura 10: Espectro discreto afectado por el Aliasing 37
Figura 11: función de autocorrelación de la señal PRBS 41
Figura 12: Espectro de densidad de energía de una señal PRBS 44
Figura 13: Correlación cruzada de la función $\boldsymbol{Rx\tau}$ 47
Figura 14: Generador de señal PRBS con XOR 48
Figura 15: señal multiseno de 10 componentes y fase nula 55
Figura 16: Señal multiseno de 10 componentes con fase inicial aleatoria 55
Figura 17: Señal multiseno de 10 componentes con fase según algoritmo Newman 56
Figura 18: Señal multiseno de 10 componentes con fase según algoritmo Schroeder 56
Figura 19: Espectro de una señal multiseno 60
Figura 20: Señal PRBS generada por Matlab para un registro de 5 bits 62
Figura 21: Resultado de la comparación entre el resultado de idinput y gen_prbs_raw 65
Figura 22: Señal PRBS 68
Figura 23: Señal PRBS en Sysquake 68
Figura 24: Gráfico generado por la función gen_prbs 71
Figura 25: Señal PRBS en Sysquake 73
Figura 26: Señal multiseno de 12 componentes 75
Figura 27: espectro de la señal multiseno 75
Figura 28: Señal multiseno de frecuencias determinísticas 77
Figura 29: Respuesta en frecuencias de la señal multiseno 77
Figura 30: Señal multiseno de frecuencias aleatorias 79
Figura 31: Respuesta en frecuencias de la señal multiseno 79
Figura 32: Señal multiseno generada con Sysquake 81
Figura 33: Espectro de la señal multiseno en Sysquake 82
Figura 34: Diagrama de casos de uso principal. 83
Figura 35: Casos de uso nivel 2, PRBS signal 85
Figura 36: Casos de uso nivel 2 Multisine signal 92
Figura 37: Casos de uso nivel 3, configure signal pannel 100
Figura 38: Casos de uso nivel 3, configure autocorrelation pannel. 103
Figura 39:Caso de uso nivel 3, configure spectrum pannel. 106
Figura 40: Casos de uso nivel 3, configure signal pannel 111
Figura 41: Casos de uso nivel 3, configure autocorrelation pannel 113
Figura 42: Casos de uso nivel 3, configure spectrum pannel 116
Figura 43: diagrama de clases 121
Figura 44: Pantalla principal de la aplicación 127
Figura 45: Control de la amplitud de la señal. 128

Figura 46: Señal de amplitud ±5v y 63 bits de longitud. 128
Figura 47: Cuadro de selección del periodo de muestreo. 129
Figura 48:Señal PRBS que se repite durante 3 ciclos. 129
Figura 49: Valores de la función de autocorrelación en el instante inicial y en el resto de elementos de la secuencia. 129
Figura 50: Detalle de la función de autocorrelación y espectro de una secuencia de 7 bits y 3 ciclos. 130
Figura 51: señal PRBS diseñada 132
Figura 52: Pantalla de generación de señal multiseno. 132
Figura 53: Señal multiseno con todas las fases nulas. 134
Figura 54: Señal multiseno resultado. 136
Figura 55: Señal multiseno con fase aleatoria. 137
Figura 56: Segunda señal multiseno con fase aleatoria. 137
Figura 57: Señal multiseno con fase nula. 138
Figura 58: Icono del programa 148
Figura 59: Pantalla inicial de la aplicación. 148
Figura 60: Opciones principales del menú *options*. 149
Figura 61: Submenú *sampling time*, Ts para la señal PRBS 150
Figura 62: Submenú *sampling time*, Ts para el caso de la señal multiseno 150
Figura 63: Zona de parámetros de la señal PRBS. 151
Figura 64: Zona de información de la señal PRBS. 151
Figura 65: Ruleta de configuración de los gráficos de la aplicación. 151
Figura 66: Panel de configuración del gráfico Signal. 152
Figura 67: Escala bloqueada entre 10 y 20 segundos. 153
Figura 68: Opción Distinguish cycles activa. 153
Figura 69: Panel de configuración de la función de autocorrelación. 154
Figura 70: Función de autocorrelación incluyendo intervalos por debajo de cero, de forma discreta y sin rejilla de reloj 155
Figura 71: Panel de control de configuración del espectro de la señal. 155
Figura 72: Espectro de la señal en tamaño mediano. 156
Figura 73: Espectro de la señal PRBS continuo y en escala logarítmica. 157
Figura 74: Espectro de la señal continua y discreta superpuestos mostrando frecuencias negativas. 158
Figura 75: Ancho de banda de la señal discreta 158
Figura 76: Efecto del Aliasing en una señal PRBS discreta. 159
Figura 77: señal multiseno 159
Figura 78: Parámetros básicos de la señal multiseno. 160
Figura 79: Información de la señal multiseno. 161
Figura 80: Panel de control del gráfico de la señal multiseno 162
Figura 81: Modificar la amplitud de la señal multiseno arrastrando y soltando 162
Figura 82: Panel de control de la función de autocorrelación 163
Figura 83: Panel de control del gráfico espectro de la señal multiseno 165
Figura 84: Grafico del espectro de la señal multiseno en tamaño mediano 165
Figura 85: Espectro de la señal multiseno simetrico respecto a cero 166
Figura 86: Ancho de banda de la señal multiseno 166

Lista de tablas

Tabla 1: Ejemplo de secuencia desplazada y sumada un bit 40
Tabla 2: Tabla de multiplicación de la autocorrelación de una PRBS 41
Tabla 3: Secuencia PRBS de máxima longitud para n=4 y realimentaciones en los estados 1 y 4 49
Tabla 4: Secuencia PRBS de máxima longitud para n=4 y realimentaciones en los estados 3 y 4 50
Tabla 5: Estados que se realimentan a la entrada del registro con XOR (fuente (Godfrey, 1993, p. 27)) 52
Tabla 6: Estados realimentados en el registro de desplazamiento 66
Tabla 7: Caso de uso, PRBS signal. 83
Tabla 8: Caso de uso, multisine signal. 84
Tabla 9: Caso de uso, select shift register size. 86
Tabla 10: Caso de uso, select clock period. 86
Tabla 11: Caso de uso, change time units. 87
Tabla 12: Caso de uso, select signal amplitude. 87
Tabla 13: Caso de uso, select cycles number. 88
Tabla 14: Caso de uso, change sampling time 88
Tabla 15: Caso de uso, choose continuous view 89
Tabla 16: Caso de uso, choose discrete view 89
Tabla 17: Caso de uso, configure signal pannel 89
Tabla 18: Caso de uso, configure autocorrelation pannel 90
Tabla 19: Caso de uso, configure spectrum pannel 90
Tabla 20: Caso de uso, update information pannel & graphics 91
Tabla 21: Caso de uso, update graphics 91
Tabla 22: Caso de uso, select sinusoids phase. 93
Tabla 23: Caso de uso, select sinusoids number. 93
Tabla 24: Caso de uso, change time units. 94
Tabla 25: Caso de uso, change sequence length 94
Tabla 26: Caso de uso, select signal period 95
Tabla 27: Caso de uso, select signal amplitude. 95
Tabla 28: Caso de uso, select cycles number. 96
Tabla 29: Caso de uso, view sampling time 96
Tabla 30: Caso de uso, choose continuous view 97
Tabla 31: Caso de uso, choose discrete view 97
Tabla 32: Caso de uso, configure signal pannel 97
Tabla 33: Caso de uso, configure autocorrelation pannel 98
Tabla 34: Caso de uso, configure spectrum pannel 98
Tabla 35: Caso de uso, update information pannel & graphics 99
Tabla 36: Caso de uso, update graphics 99
Tabla 37: Caso de uso, update scale limits 101
Tabla 38: Caso de uso, plot continuous PRBS 101
Tabla 39: Caso de uso, plot discrete PRBS 102
Tabla 40: Caso de uso, plot clock period grid. 102
Tabla 41: Caso de uso, distinguish cycles. 102
Tabla 42: Caso de uso, drag amplitude and clock period. 103

Tabla 43: Caso de uso, update scale limits 104
Tabla 44: Caso de uso, plot continuous autocorrelation 104
Tabla 45: Caso de uso, plot discrete autocorrelation 105
Tabla 46: Caso de uso, only positive lags 105
Tabla 47: Caso de uso, plot clock period grid 105
Tabla 48: Caso de uso, change size. 107
Tabla 49: Caso de uso, view legend. 107
Tabla 50: Caso de uso, update scale limits 108
Tabla 51: Caso de uso, plot continuous spectrum 109
Tabla 52: Caso de uso, plot discrete spectrum 109
Tabla 53: Caso de uso, plot double-sided spectrum. 109
Tabla 54: caso de uso, highligth effective band. 110
Tabla 55: Caso de uso, drag effective band. 110
Tabla 56: Caso de uso, update scale limits 111
Tabla 57: Caso de uso, plot multisine continuous 112
Tabla 58: Caso de uso, plot multisine discrete 112
Tabla 59: Caso de uso, distinguish cycles. 112
Tabla 60: Caso de uso, drag amplitude and clock period. 113
Tabla 61: Caso de uso, update scale limits 114
Tabla 62: Caso de uso, plot continuous autocorrelation 114
Tabla 63: Caso de uso, plot discrete autocorrelation 115
Tabla 64: Caso de uso, only positive lags 115
Tabla 65: Caso de uso, change size. 116
Tabla 66: Caso de uso, view legend. 117
Tabla 67: Caso de uso, update scale limits 118
Tabla 68: Caso de uso, plot continuous spectrum 119
Tabla 69: Caso de uso, plot discrete spectrum 119
Tabla 70: Caso de uso, plot double-sided spectrum. 119
Tabla 71: caso de uso, highligth effective band. 120
Tabla 72: Caso de uso, drag effective band and power. 120
Tabla 73: Clase Signals_t 122
Tabla 74: Clase Signal_PRBS 122
Tabla 75: Clase Signal_multisine 123
Tabla 76: Clase Aux 123
Tabla 77: Clase Parameters 124
Tabla 78: Clase Par_PRBS 124
Tabla 79: Clase Par_multisine 124
Tabla 80: Clase Signal 125
Tabla 81: Clase PRBS 125
Tabla 82: Clase Multisine 126

Lista de acrónimos

BYOD	Bring your own device
DFT	Discrete Fourier transform
DIB	Discrete interval binary
DIT	Discrete interval ternary
DP	Director del proyecto
GFs	Galois fields
GPCIT	Generalized predictive control interactive tool
HAB	Hall binary
ILM	Interactive learning modules
ITCLSD	Interactive tool for closed-loop sharpening linear control design
ITSIE	Interactive tool for system identification education
ITTSAE	Interactive tool for time series analysis education
ITTSAE-TSA	Interactive tool for time series analysis education – time series analyser
ITTSAE-TSG	Interactive tool for time series analysis education – time series generator
IRPRB	Inverse-repeat pseudo-random binary
LCSD	Linear control system design
LFSR	Linear feedback shift register
MLB	Maximum length binary
MLBS	Maximum length binary sequence
MLMH	Multi-level multi-harmonic
PID	Proporcional Integral Derivado
PIPS	Performance index for perturbation signals
PRBS	Pseudo-random binary sequence
PRIS	Pseudo-random impulse sequence
PRML	Pseudo-random multilevel
QRB	Quadratic residue binary
QRT	Quadratic residue ternary
RBS	Random binary signal
RCLSD	Robust closed-loop sharping linear control system design
RMS	Root mean square
SISO-QFTIT	Single input – single output – quantitative feedback theory interactive tool
SOH	Sum of Harmonics
SRG	Shift register generator
TDF	Transformada discreta de Fourier
TFM	Trabajo Fin de Master
TIG	Template interactive generator
TPB	Twin prime binary
UNED	Universidad Nacional de Educación a Distancia
XOR	Exclusive OR

1 Introducción.

Durante las últimas décadas, ha habido un creciente reconocimiento de la importancia de la identificación de sistemas y las múltiples aplicaciones que tiene en muchos ámbitos de la ciencia. En los años 70 bastaba que el sistema a controlar funcionara, hoy en día debe ser óptimo y para conseguir esto se precisa de un buen diseño de las señales de entrada utilizadas para identificar el sistema. Una adecuada elección de la señal de perturbación y del método para procesar la relación entre la entrada y la salida del sistema, lleva a resaltar diferentes aspectos de éste. Por ejemplo, algunas señales son particularmente buenas a la hora de excluir los efectos de las no linealidades del sistema, mientras que, por el contrario, otras pueden servir para la identificación de estas no linealidades. Una característica común de todas estas señales es que se utilizan para identificar la dinámica entrada-salida de un sistema continuo a partir de una señal de perturbación digital y de una salida muestreada digitalmente.

Este trabajo de fin de master se centrará en el diseño y características de dos de las señales de excitación más comunes en la identificación de sistemas, las señales binarias pseudo aleatorias de longitud máxima y las señales multiseno.

1.1 Motivación del proyecto.

La principal motivación para la elaboración de este TFM (Trabajo Fin de Master) es la obligación de su realización para los alumnos de primer año de doctorado que no hayan hecho el master de Ingeniería de Sistemas y Control de la UNED/UCM.

Por tanto, la elaboración de este TFM va a ir motivada por la necesidad del estudiante de doctorado de ir introduciéndose en lo que será su campo de investigación: la identificación de sistemas y el uso de software interactivo para la implementación de herramientas didácticas.

El objetivo final de la identificación de sistemas es la obtención de modelos matemáticos de sistemas dinámicos. Estas técnicas tienen una gran importancia en muchas áreas de la ciencia y de la ingeniería. Estos modelos, en muchos casos, no se pueden obtener por otros procedimientos debido a la complejidad del propio modelo, su falta de linealidad o su naturaleza cambiante.

En el campo de la Ingeniería de Sistemas y la Automática los modelos obtenidos se utilizan para el diseño de controladores, simulación o predicción del comportamiento de los sistemas. Una posible técnica para la obtención de modelos es la identificación de sistemas, que consiste en excitar al sistema con unas señales de entrada, registrar los datos de las entradas y las salidas (en el dominio temporal o frecuencial), realizar un tratamiento de los datos (eliminación de medias y tendencias, filtrado y detección de outliers), y haciendo uso de dichos datos de entrada-salida obtener modelos no paramétricos y modelos paramétricos (ver Figura 1).

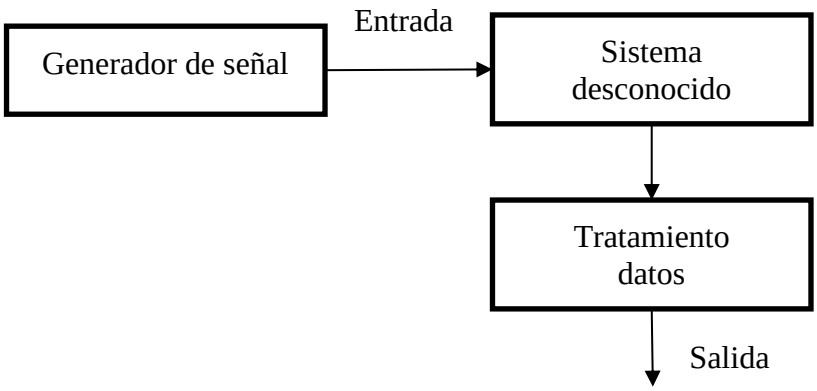

Figura 1: Esquema básico de la identificación de sistemas

En el caso de los modelos paramétricos, se debe seleccionar primero una estructura del modelo y posteriormente estimar sus parámetros haciendo uso de los datos de entrada-salida disponibles. El modelo estimado debe ser validado sometiéndole a diferentes test. Si la validación no resulta positiva se deben probar otras estructuras y obtener nuevos conjuntos de datos diseñando nuevamente las señales de entrada de excitación.

La identificación de sistemas es una metodología de obtención de modelos que requiere de herramientas software para su realización. La herramienta software más utilizada por su potencia y versatilidad es la toolbox de identificación de Matlab. Sin embargo, desde el punto de vista del aprendizaje y la enseñanza de la identificación de sistemas, se echan en falta herramientas software interactivas complementarias de carácter gratuito que permitan ilustrar adecuadamente los conceptos involucrados y los efectos de las decisiones que hay que tomar en cada etapa de esta metodología.

Un software con un lenguaje similar al usado en los scripts de Matlab que permite implementar herramientas software de alta interactividad y con interfaces gráficas simples y agradables es Sysquake (Calerga, 2022a) de Calerga desarrollado por Yves Piguet.

Como ya se dijo anteriormente, este trabajo fin de master se realiza por estar en el primer año de investigación de una tesis doctoral. La tesis quiere abordar el problema de la generación de herramientas didácticas para la identificación de sistemas. Dentro del ámbito de investigación de la tesis una de las partes que se debe abordar es la generación de señales utilizadas en la identificación de sistemas.

En este TFM, se van a generar señales de dos familias distintas:

- Señales PRBS (Pseudo-Random Binary Signals), que son señales binarias pseudo aleatorias.
- Señales multiseno, que son señales composición de señales sinusoidales puras desfasadas entre sí.

La generación de estas señales se va a realizar con la citada herramienta interactiva Sysquake que facilita el diseño de herramientas relacionadas con la ingeniería de control.

1.2 Objetivos del proyecto.

Los objetivos de este TFM son:

- Ilustrar las características en el dominio del tiempo y la frecuencia de las dos señales más utilizadas en la identificación de sistemas, las señales PRBS y multiseno.

- Diseñar e implementar en Sysquake una herramienta software interactiva que permita ilustrar los conceptos y decisiones fundamentales de la generación de señales que permitan la identificación de sistemas.

1.3 Estructura del proyecto.

En los siguientes capítulos se va a desarrollar el TFM persiguiendo conseguir los objetivos propuestos previamente, para ello se propone la siguiente estructura para el resto del trabajo:

- El segundo capítulo se va a realizar una introducción general a la identificación de sistemas, para ello se investigarán las aplicaciones en las que se precisa la identificación de sistemas y por qué etapas pasa la misma. A continuación, se estudiará el efecto de la interactividad en la enseñanza y cómo esta puede mejorar la comprensión de los alumnos en una materia dada. Por último, este segundo capítulo se cierra con el estudio del estado del arte relacionado con las herramientas software existentes en Matlab o Sysquake con finalidad docente o no para identificación de sistemas y diseño de señales de entrada (PRBS, multiseno).
- El tercer capítulo se centrará en el diseño de las citadas señales de excitación, haciendo especial hincapié en las señales PRBS y señales multiseno, incidiendo en el fundamento teórico precisado para la generación de las citadas señales y sus propiedades. Además, se tratará de explicar la importancia de diseñar adecuadamente la entrada que va excitar el sistema.
- El cuarto capítulo se centrará en el diseño e implementación de la herramienta Sysquake para la generación de las señales, esta herramienta servirá para facilitar el manejo de las mismas y llegar a una mejor comprensión de las características de cada señal.
- En el quinto capítulo se abordará el diseño de diversos ejemplos de uso de la herramienta, estos ejemplos deberán de tener un propósito didáctico y servir como prácticas de diversas asignaturas relacionadas con el control y la identificación de sistemas.
- El sexto capítulo abordará las conclusiones del TFM y los posibles trabajos futuros y líneas de investigación que abre la generación de señales para la identificación de sistemas.
- Por último, el TFM se completa con dos apéndices que contienen un manual de uso de la herramienta creada y el código fuente del programa.

2 Aprendizaje de la materia Identificación de Sistemas.

El diseño de sistemas de control es uno de los objetivos claves para la industria 4.0 ya que permiten no solo regular las características básicas del proceso a controlar sino además optimizar el comportamiento de dicho proceso en circunstancias cambiantes.

El problema en muchos casos es que para poder diseñar el regulador más adecuado se debe tener un conocimiento profundo del sistema a controlar. Solo en unos pocos casos se pueden encontrar sistemas sencillos a controlar. En la mayoría de casos será necesario realizar algunas simplificaciones y linealizaciones con el objetivo de obtener un modelo matemático que responda al sistema y que permita diseñar un controlador óptimo para ese sistema. En la mayoría de los casos el ingeniero de control deberá ir testeando y simulando controladores de tipo PID o variantes del mismo hasta encontrar una solución satisfactoria para la resolución del problema planteado. En este sentido hay que hacer mención de los algoritmos de autotune de reguladores PID que ofrecen herramientas como Matlab.

Es por este motivo por el que han ido apareciendo distintos mecanismos de control denominados inteligentes, robustos o adaptativos que partiendo de un modelo del sistema parcialmente conocido o totalmente desconocido pretenden ir aproximándose a una solución de control óptima. Ya que hoy en día, el desafío no es simplemente hacer que los sistemas de regulación funcionen, sino que lo hagan de una forma que permita reducir costes de producción y aumentar la vida útil de los elementos implicados en un proceso industrial.

En los años 60 del siglo pasado comenzó a experimentarse con señales de perturbación añadidas a las señales de control convencionales con el objetivo de identificar la función de transferencia del sistema que se estaba controlando (Liu & Gao, 2012). Esta técnica se aplicó en el control de procesos industriales que no pueden detenerse nunca como por ejemplo: una central nuclear (Godfrey, 1969), una planta química, centrales eléctricas, papeleras o altos hornos. Todos estos procesos tienen en común dos características: es demasiado caro o peligroso detener el proceso para reevaluar la función de transferencia del sistema en un momento dado y una mejora en el rendimiento del sistema a través de una mejora en el proceso de control genera un importante beneficio económico.

La selección de esta señal de perturbación va a desempeñar un papel decisivo en la calidad de la función de transferencia obtenida para el sistema y por tanto influir finalmente sobre la calidad del controlador y el resultado final del sistema regulado (Godfrey, 1969).

Los controles avanzados como por ejemplo los modelos de control predictivos, basan su funcionamiento en maximizar la precisión del modelo, de forma que en cada ciclo de funcionamiento deben recalcular el modelo del sistema en esas condiciones. En casos como éste es crítica la elección de las señales de perturbación que permiten medir el modelo del sistema con la máxima precisión (Mayne, 2014).

2.1 Identificación de sistemas.

Llamaremos **modelo** a una expresión matemática que describe las características básicas del proceso que pretendemos estudiar. Para llegar a este modelo hay dos caminos:

- Buscar ecuaciones diferenciales basadas en los principios físicos que rigen el proceso.
- Hacer una aproximación práctica basada en la observación del comportamiento del sistema cuando es sometido a diferentes tipos de entradas.

Esta segunda aproximación suele ser más acertada cuando se pretende analizar sistemas complejos, en los que las relaciones entre sus diferentes componentes no tienen por qué ser evidentes.

Esto nos lleva a una primera idea del significado de la identificación de sistemas atribuida a Tangirala: *la identificación de sistemas tiene que ver con los medios y técnicas utilizados para estudiar un proceso o sistema a través de la observación o la experimentación con el objetivo de obtener una descripción matemática útil del sistemas*[1] (Tangirala, 2014).

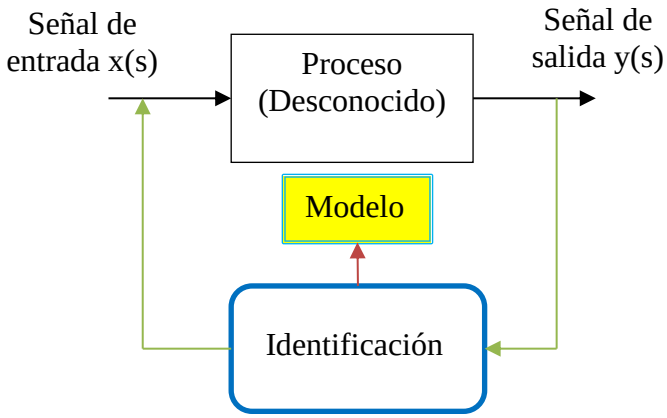

Figura 2: Proceso de identificación de un sistema

La Figura 2 resume de una forma gráfica el proceso interactivo de obtención del modelo del sistema. Este modelo obtenido por este procedimiento consiste en dos componentes: una descripción matemática de las relaciones causa-efecto del proceso a controlar, comúnmente conocido como **modelo determinístico** y un modelo estadístico-matemático que describe las incertidumbres del proceso conocido como **modelo estocástico**. Este segundo modelo aparece cuando tenemos en cuenta los errores de medida, las incertidumbres del proceso y los errores de modelado.

De forma que tendremos dos maneras básicas de aproximarnos al problema de la identificación de sistemas, por una parte, un sistema teórico basado en las leyes de la física y en segundo lugar uno más práctico basado en la observación del sistema en funcionamiento. La segunda aproximación tiene algunas ventajas (Tangirala, 2014):

- Se puede diseñar un modelo con un conocimiento mínimo del proceso que deseamos controlar.
- Mayor flexibilidad a la hora de determinar la estructura del modelo
- Facilidad para escribir un algoritmo que represente el modelo.

A estos modelos obtenidos con estos principios se les denomina **modelos de caja negra**, ya que desconocemos los principios físicos del proceso que estamos estudiando. Un problema que puede surgir a veces con este tipo de modelos es que funcionan bien con señales similares a las señales de test usadas para su diseño, pero podemos obtener resultados inesperados con señales diferentes.

[1] Traducción propia del inglés.

2.1.1 Elementos que intervienen en la identificación

Además del proceso que queremos identificar hay algunos elementos que intervienen siempre en el proceso y que es importante recapitular en este apartado:

- **Actuadores**: son los elementos que a partir de la señal de entrada generan la señal capaz de excitar el sistema. Por ejemplo, un motor, las resistencias de un horno o un diodo laser en un sistema de comunicación.
- **Sensores**: son los elementos encargados de medir la señal que vamos a procesar. Realmente solo conocemos el valor de salida del sistema por la medida que obtenemos del sensor, de forma que cualquier error cometido por el sensor será un error en la variable de salida.
- **Perturbaciones**: todo proceso está afectado por perturbaciones. Por ejemplo, una antena de microondas sufre desalineaciones debidas al efecto del viento o el sistema de guiado de un vehículo está afectado por las imperfecciones del terreno. Además de estas perturbaciones obvias existen otras que no lo son tanto, como el ruido eléctrico que afecta a los sensores y a las señales de entrada de los actuadores. Algunas de estas perturbaciones podrán medirse con sensores externos y otras no.
- **Señales de entrada**: las señales de entrada son aquellas variables que consideramos que tienen una influencia significativa sobre la salida y será el punto en donde se introducirán las señales de prueba. Estas señales serán una combinación de señales que podremos manipular durante el experimento con otras que no se podrán manipular ya que son críticas para el funcionamiento del proceso. Estas señales se generarán digitalmente usando un ordenador.
- **Señales de salida**: es la respuesta del sistema y estará influenciada por la respuesta de cada uno de los elementos expuestos previamente. Estas señales se muestrearán digitalmente con un ordenador y se almacenarán para su posterior proceso.

La Figura 3 ilustra la relación de los elementos involucrados en el proceso de identificación. Es interesante darse cuenta de que lo será posible identificar no es el proceso en sí mismo sino los elementos dentro de la caja negra.

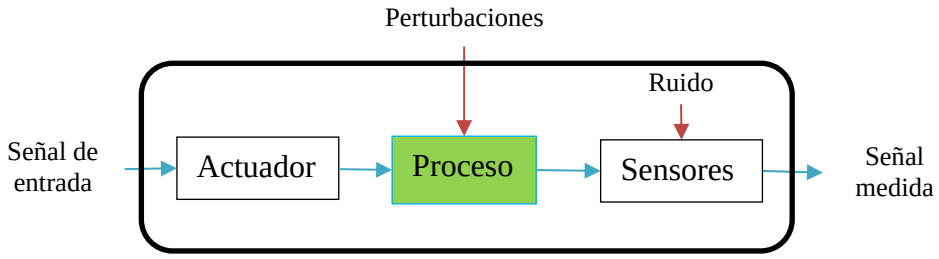

Figura 3: Sistema identificado

2.1.2 Hechos relacionados con la identificación

Hay varias afirmaciones relacionadas con la fidelidad y precisión de los modelos identificados: (Tangirala, 2014)

1. No es posible construir un modelo fidedigno a partir de un conjunto finito de muestras. Cualquier modelo estimado a partir de datos contaminados con errores nunca puede ser fidedigno. En muchos casos se produce una estimación errónea de

la estructura del modelo (En sistemas LTI considerar un número inferior de polos y ceros) de forma que el resultado obtenido es distinto del modelo real.

2. No es posible estimar un modelo preciso a partir de un conjunto finito de muestras. Cada vez que se repite un experimento se obtienen resultados diferentes, esto es debido a la variabilidad del ruido y perturbaciones que afectan al sistema. El proceso de identificación en la mayoría de los casos estima que esa varianza de los datos es nula, lo que es falso y lleva a un modelo erróneo.
3. La fidelidad y precisión del modelo identificado, depende críticamente de:
    a. El tipo de entrada, forma y niveles.
    b. La relación señal ruido de las señales manejadas en el experimento.

   Como Fisher (1925) propone, la calidad del modelo final depende de la información presente en los datos. Por tanto, una de las tareas más importantes en la identificación de sistemas será el diseño de las señales de entrada ya que la información que no esté presente en ellas no podrá ser extraída del sistema. Esto unido al desconocimiento de la estructura real del modelo hace que en muchos casos el diseño de estas señales de entrada sea un proceso interactivo.

El modelo obtenido por estos métodos será siempre un modelo en tiempo discreto, ya que las mediciones de las variables de salida se realizarán con conversores analógico digitales con los que obtendremos un conjunto de medidas en unos instantes concretos del tiempo. Por lo tanto, la frecuencia de muestreo de este sistema de medición deberá ser otro parámetro a tener en cuenta en el desarrollo del modelo.

2.1.3 Tipos de modelos

Podemos encontrar diferencias entre la definición de un modelo, dependiendo en gran medida del objetivo que se persigue obtener con el citado modelo. Así podemos hablar de:

- **Modelos cuantitativos o cualitativos**: en algunos casos al hablar de un sistema simple no interesa conocer su comportamiento exacto sino simplemente las reglas en que se basa. Por ejemplo, decimos que, si pisamos el acelerador, el coche aumenta su velocidad sin que nos interese poner números a ese modelo de comportamiento. En esos casos hablamos de modelos cualitativos. Los modelos que perseguiremos en este trabajo serán de tipo cuantitativo, donde interesa conocer un valor exacto o al menos aproximado de la relación entre las variables de entrada y salida del sistema.
- **Modelos determinísticos o estocásticos**: un modelo que explique la relación entre la salida y la entrada es un modelo determinístico y un modelo que explique los efectos de las perturbaciones y el ruido es un modelo estocástico.
- **Modelos paramétricos y no paramétricos**: En un modelo paramétrico en primer lugar debemos decidir la estructura del modelo y a continuación estimar el valor de los parámetros a partir de la información que tenemos de la entrada y la salida. En un modelo no paramétrico no partimos de ninguna estructura específica por lo que el modelo presenta únicamente la información proporcionada por la relación de valores obtenidos durante el experimento. La desventaja de un modelo paramétrico es que un error en la estimación de la estructura puede llevar a un resultado erróneo. La ventaja de un modelo no paramétrico es que no necesita un conocimiento preciso de la estructura del proceso a modelar.

2.1.4 Pasos a seguir durante la identificación de un sistema

En la Figura 4 se muestran los pasos a seguir para la identificación de un sistema o proceso. Vamos a profundizar en el significado de cada uno de estos pasos:

- **Generación y adquisición de datos**: este primer paso es el más importante en el proceso de identificación, ya que el resultado final dependerá de la calidad y confianza de los datos capturados. La Figura 5 ilustra el funcionamiento del proceso de muestreo de las señales del sistema. La secuencia $u[k]$ es la entrada del sistema que se conecta a un conversor digital analógico que realiza la función de un retenedor de orden 0, de forma que a su salida tenemos la señal continua $u(t)$. Esta señal excita al actuador que es una combinación de sistemas electromecánicos que generan la entrada precisada por el proceso[2]. A continuación, el proceso a evaluar genera la salida de la variable que se está estudiando $x(t)$. Esta variable se mide usando un sensor adecuado que proporciona la salida $y(t)$ en forma de una variable continua. Esta señal continua debe discretizarse usando un conversor digital analógico que la convierte en la secuencia de salida $y[k]$. Otra de las decisiones a tomar durante esta fase tiene que ver con la elección del periodo de muestreo y precisión para la secuencia de entrada y de salida. Por tanto, del conversor A/D y D/A. Lo lógico es que ambas sean del mismo valor y que deban estar sincronizadas para que entrada y salida lo estén.
  La elección de la señal de entrada también va a jugar un papel decisivo en la identificación del sistema. Si la señal de entrada no contiene suficiente información en un rango de frecuencias importante para el sistema, el resultado no será útil. Uno de los problemas que pueden surgir, es que muchas veces desconocemos la naturaleza del sistema y por tanto cual sería la señal más adecuada. Esto lleva a que muchas veces éste sea un proceso de ensayo y error.

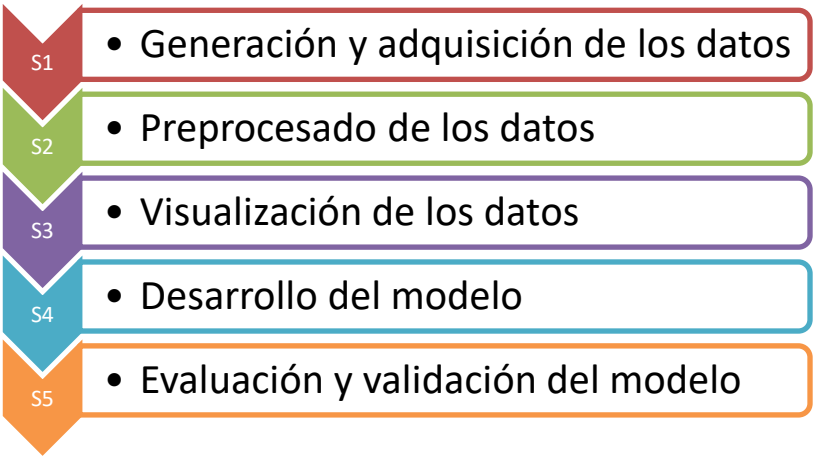

Figura 4: Pasos a seguir para la identificación del modelo

- **Preprocesado de los datos**: los datos generados en la entrada del sistema y obtenidos en su salida, serán almacenados en algún tipo de soporte informático. Antes de su procesado deberán revisarse para comprobar que no existe algún problema que afecte a la calidad del proceso de identificación. En muchos casos un

[2] El tipo de actuador depende del proceso a evaluar, si por ejemplo estamos modelando un horno, el actuador será el conjunto formado por un amplificador de potencia y las resistencias calefactoras del horno.

filtrado será una elección correcta para resolver esta fase del proceso de identificación. Los problemas más comunes con los datos son:

a. Datos perdidos: en ocasiones al realizar la lectura en el conversor D/A una falta de sincronización del proceso, puede dar lugar a la pérdida de uno o varios datos de salida. Esos datos perdidos pueden por ejemplo ignorarse o sustituirse por el valor medio de la secuencia en ese instante.
b. Valores raros: al analizar los resultados, podemos descubrir valores fuera de rango que respondan a errores de medición. Su tratamiento puede ser similar al de los datos perdidos.

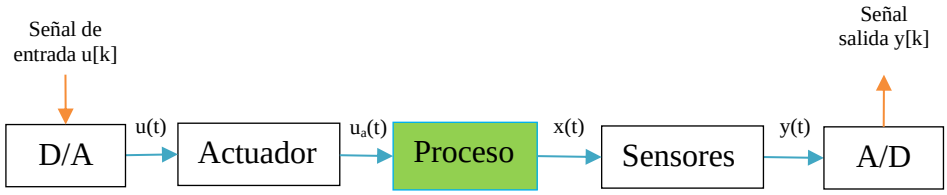

Figura 5: Sistema de muestreo

- **Visualización de los datos**: Esta es una fase asociada a la anterior ya que una inspección visual de los datos ayudará a localizar esos datos desaparecidos o con valores extraños. Además, podemos detectar otros efectos indeseables como que por ejemplo la señal de entrada tenga un nivel inferior al del ruido del sistema. Esta visualización se puede hacer en el dominio del tiempo o en el dominio de la frecuencia, en ambos casos se puede extraer información interesante y complementaria.
- **Desarrollo del modelo**: Esta será la parte fundamental del proceso de identificación del sistema. Este proceso de divide en dos fases:
  a. Determinación de la estructura del modelo.
  b. Estimación de los parámetros del modelo.

  Esta fase junto con la última parte del proceso de identificación será una tarea iterativa, ya que una vez determinado el modelo se deberá verificar si este cumple las especificaciones. En caso contrario, se deberá evaluar un modelo diferente hasta conseguir verificar las especificaciones del problema a resolver.
- **Evaluación y validación del modelo**: No se debe olvidar que un modelo es una abstracción matemática de una realidad física que es el sistema a identificar, por este motivo puede ocurrir que el modelo se ajuste perfectamente a los datos usados en la simulación pero que luego no encaje con otros datos. Es por este motivo que en estos experimentos se debe tener un conjunto de datos con los que ajustar el modelo y un segundo conjunto de datos con los que validar el modelo. El uso de métodos estadísticos, pueden indicarnos el grado de ajuste del modelo a los datos.

Además de las consideraciones realizadas hasta ahora hay que tener en cuenta que disponer de algún conocimiento previo del sistema a identificar puede reducir en gran medida el esfuerzo de identificación del sistema.

2.2 Aprendizaje con metodologías basadas en la interactividad.

En docencia el objetivo último es que los estudiantes aprendan las materias que componen un curso. Investigaciones realizadas desde los años 90 (Bonwell & Eison, 1991) prueban que cuanto más involucrados están los estudiantes en su proceso de enseñanza más aprenden. Sin embargo, en muchas instituciones se siguen usando formas pasivas de aprendizaje.

El concepto del **aprendizaje activo**, esto es, incrementar el compromiso de los estudiantes en el proceso de aprendizaje, es una técnica imprescindible para incrementar la efectividad de la enseñanza, que en casi todas las ocasiones no supone un incremento de coste.

La mayoría de los docentes usan la clase magistral como medio de docencia evitando usar técnicas de aprendizaje activo. Estos docentes confían en este mecanismo porque es el que se ha usado tradicionalmente durante decenios y se sienten culpables si intentan aplicar otras técnicas. Sin embargo, estudios como el Stuart y Rutherford (1978) desvelan que un estudiante entre los últimos cursos altamente motivado solo es capaz de mantener la atención entre 10 y 15 minutos. Un segundo estudio desvela que durante una clase de 50 minutos las cantidad de notas que toman los alumnos y su calidad van decayendo conforme avanza la clase (Dickinson & Verner, 1967; Milton & Penner, 1986; Verner & Dickinson, 1967). Eso no quiere decir que por parte de los alumnos también exista un acomodamiento a estas técnicas pasivas y que muchos de ellos se sientan más cómodos con la clase magistral ya que no exige una participación activa del estudiante (Chisholm, 1989). La clase magistral no resulta ser más efectiva que otros métodos como por ejemplo los foros de discusión (Costin, 1972) en la adquisición de hechos y nociones básicas, aunque los foros de discusión son más efectivos para el desarrollo de la capacidad de resolver problemas de los estudiantes.

2.2.1 Técnicas activas que mejoran la retención de los estudiantes

En esta sección se van a relatar una serie de técnicas de aprendizaje activo, de forma no sistemática, que alguna de ellas será aplicada posteriormente en el desarrollo de las herramientas incluidas en este TFM.

Una de las técnicas activas que se puede utilizar es la de la **experimentación**. Realizar experimentos durante las clases puede servir para estimular la curiosidad de los estudiantes y para mejora su comprensión de los conceptos, sobre todo cuando el experimento invita a la participación de los estudiantes (Okpala & Onocha, 1988; Phillips et al., 1984).

Una segunda técnica es la de **resolución de problemas**. Dewey (1983) propone un modelo clásico de toma de decisiones en 4 pasos:

- Definir el problema.
- Diagnosticar posibles razones para que aparezca el problema.
- Buscar soluciones alternativas.
- Evaluar las alternativas y escoger la solución más apropiada.

Este esquema ha servido como base para dos aproximaciones a la solución de problemas: el estudio de casos y el diseño guiado.

Los **estudios de caso** se utilizan en la investigación cualitativa para recopilar datos a partir de relatos de personas que han tenido que enfrentar un problema o suceso (McLeod & Fisher, 1978). Aunque son métodos propios de la psicología, también se pueden utilizar en otras áreas proponiendo a los estudiantes la resolución de cuestiones a partir de un relato previo de un suceso. Esto facilita la entrada en el campo de estudio y mejora la retención de datos mostrados al estudiante durante el relato del caso (Romm & Mahler, 1986).

El **diseño guiado** fue desarrollado en la Universidad West Virginia al final de los 70 y es una ampliación del modelo de Dewey. Los pasos que debe seguir el proceso son (Wales et al., 1986):

- Descubrir el estado del problema a resolver y establecer un objetivo.
- Recopilar información relevante.
- Generar posibles soluciones para el problema.
- Listar las restricciones de lo que se puede conseguir con cada posible solución.
- Escoger una posible solución.
- Analizar los factores importantes que deben ser considerados en el desarrollo de una solución detallada.
- Sintetizar la solución detallada.
- Evaluar la solución final.
- Recomendar las acciones adecuadas a la solución escogida.

El diseño guiado pone por escrito los pasos que cada uno sigue de forma inconsciente en su día a día al afrontar un problema. De forma que esta técnica resulta útil en las aproximaciones docentes de resolución de problemas.

La **enseñanza con apoyo de ordenadores** se ha posicionado como una de las tecnologías más prometedoras en la enseñanza activa. Ya desde los comienzos de su uso en la docencia se han presentado como una manera de enganchar al estudiante en la asignatura (Kulik & Kulik, 1987). Además, varios estudios demuestran que estudiantes con diversos estilos diferentes de aprendizaje obtienen beneficios similares del uso de ordenadores en el aula (Johnston & Gardner, 1989; Macgregor et al., 1988). La enseñanza con ordenador puede tener sus contraprestaciones, como por ejemplo el coste de las instalaciones y su mantenimiento para la institución. Aunque movimientos como el BYOD (Bring your own device) han aportado una segunda opción para las instituciones a coste cero (Ballagas et al., 2004; Miller & Welsh, 2017; Olalere et al., 2015).

El **aprendizaje cooperativo** puede mejorar el aprendizaje de los alumnos a la vez que favorece sus habilidades sociales como la gestión de conflictos y la toma de decisiones. Esta metodología se desarrolla en torno a la realización de un proyecto, la duración de este puede ser desde una clase al curso completo y esta técnica ha sido utilizada en diversos niveles educativos que van desde infantil hasta la universidad (J. Cooper, 1990). Probablemente la tarea más compleja en esta técnica sea la puntuación de cada alumno (Cohen & Tellez, 1994). Se considera que el grupo funcionará mejor si una parte de la nota es independiente para cada miembro del grupo (Slavin, 1983). Un segundo efecto de esta técnica es que reduce el absentismo a las clases. Se ha observado que en algunos grupos se puede reducir el absentismo del 50% al 1% de estudiantes (Slavin, 1983).

Los **juegos y las simulaciones** se pueden usar para mejorar la predisposición de los estudiantes en las clases. Estas técnicas también promueven el trabajo en grupo y suelen generan elevados niveles de motivación y entusiasmo. En un estudio realizado se encontró que los estudiantes sometidos a prácticas de simulación obtenían mejores notas que los que asistieron a una clase convencional (Fraas, 1980).

Hay que hacer notar que para aplicar estas técnicas hay que seguir las reglas de cada una de ellas, ya que una innovación mal aplicada puede dar lugar al resultado contrario al que se esperaba y provocar el tedio de los estudiantes o su pérdida de interés por la asignatura.

2.2.2 La enseñanza interactiva en la ingeniería de control

La ingeniería de control es un campo lleno de retos, en el que el estudiante debe llegar a tener una buena base en diferentes materias matemáticas (Braatz, 2013):

- Algebra lineal.
- Cálculo.
- Ecuaciones diferenciales.
- Variable Compleja.

Además, es un campo interdisciplinar ya que combina conocimientos de Física, Química, Economía o Electrónica entre otros con las técnicas propias del control. Los avances en éste campo se logran a base de una combinación de modelado, simulación, matemáticas y experimentación (Dormido Bencomo, 2002).

Los conceptos del control automático se pueden representar en muchos casos de forma gráfica, esta representación suele ser mucho más intuitiva y ayuda en muchos casos a resolver los problemas más fácilmente (Brown & Thaler, 1953; Clark, 1962; Eckman, 1958; Elgerd, 1967; Gardner & Barnes, 1943; Gibson, 1963; Graham & McRuer, 1962; Naslin, 1962; Newton et al., 1958). Esto lleva a la idea de la **visualización**, que permite explicar complejas relaciones abstractas por medio de representaciones gráficas (Dormido Bencomo, 2002, 2004). Como ya se ha dicho, la teoría de control se apoya fuertemente en una modelización matemática de la realidad, pero nuestra naturaleza humana encuentra más sencillas estas representaciones visuales. Por ejemplo, la representación matemática en el dominio de la frecuencia de un sistema es compleja y precisa de un conocimiento detallado de la transformada de Fourier y de la interpretación de los polos y ceros en una función de transferencia. Sin embargo, en un diagrama de Bode se puede ver de forma intuitiva el valor de la ganancia del sistema y de su fase a una determinada frecuencia o la estabilidad relativa del sistema. Otro ejemplo son los diagramas de polos y ceros del sistema que apoyándose en el método del lugar de las raíces permiten localizar un parámetro de la función de transferencia. Desde la aparición de los ordenadores personales diversos programas como Matlab, Scilab, MatrixX, Octave o Labview han permitido visualizar conceptos relacionados con los sistemas de control de una forma muy eficiente, y han servido como herramientas de apoyo al diseño de sistema de control.

Por otra parte, el proceso de diseño de un sistema de control involucra dos fases, la síntesis y el análisis. En la primera fase de síntesis se calculan los parámetros del sistema como incógnitas de un sistema de ecuaciones lineales y en la fase de análisis se verifica si con esos parámetros el sistema cumple las especificaciones propuestas para ese diseño. El proceso es iterativo ya que, si en la fase de análisis no se consiguen cumplir todas las restricciones impuestas al sistema, se deben recalcular de nuevo los parámetros o se pueden añadir nuevos subsistemas de control que mejoren las prestaciones del primer diseño[3]. En este aspecto las herramientas de control clásicas carecen de la necesaria interactividad que podría simplificar el proceso.

Sin embargo, la tecnología actual con el apoyo de los programas de ordenador, permite integrar las dos fases en una sola, modificando los parámetros y viendo inmediatamente el efecto que tiene dicha modificación. Esto consigue dinamizar el proceso de diseño y que los estudiantes tengan una mejor percepción de lo que se persigue con el proceso de diseño, ya que siguiendo las técnicas convencionales muchas veces se pierden en la complejidad matemática del problema sin llegar a entender el objetivo del ejercicio que están realizando. A este tipo de metodología de diseño la denominamos **interactiva**. Se puede definir la interactividad como la acción mutua realizada por dos o más objetos que permite el intercambio de información entre ellos (Guzmán Sánchez et al., 2013). Esta capacidad de mantener un flujo de información entre los objetos de aprendizaje es un mecanismo fundamental en la adquisición de conocimiento.

Uno de los primeros sistemas de diseño interactivo lo encontramos en el trabajo de Wittenmark, (1998) donde se nos presenta una herramienta que contiene un conjunto de gráficos

[3] Por ejemplo, un avance o atraso de fase.

dinámicos denominados *pinturas dinámicas*. De forma que estos gráficos responden a las acciones de los alumnos y al modificar un elemento o parámetro, el resto de las ventanas se actualizan para adaptarse al nuevo valor. La ventaja de esta herramienta es que a diferencia de otras como Octave o Matlab el estudiante no tiene que escribir código para conseguir que el gráfico aparezca en la pantalla. Por tanto, el estudiante se centra en la tarea principal que es entender el problema de control que se le está presentando y buscando una solución, sin tener que estar pendiente de cerrar un paréntesis olvidado o corregir una coma en el código.

Herramientas similares a la descrita son:

- Ictools (Johansson et al., 1998).
- CCSdemo (Wittenmark et al., 1998)
- Syquake (Calerga, 2022b; Piguet et al., 1999).
- Easy Java Simulations (EJS) (Esquembre, 2004).
- Interactive Learning Modules for PID (Guzmán Sánchez et al., 2006).
- Uso de JGrafchart para diseño interactivo de módulos PID (Theorin & Johnsson, 2014).
- Paquetes intrac, idpac, modpac, simnon, synpac y polpac (Åström, 1995, 2008)
- Paquetes user friendly CACSD (Ravn, 1989)

El aprendizaje de la teoría de control puede resultar confuso para un estudiante que entra en el campo por primera vez. Existen dos dominios diferentes, el del tiempo y el de la frecuencia y en cada uno de los dos dominios se estudian partes del comportamiento de un sistema. En el dominio del tiempo se estudia principalmente la sobreoscilación del sistema, su tiempo de asentamiento y el error de posición. En el dominio de la frecuencia se estudia la estabilidad relativa del sistema y ancho de banda. El análisis del comportamiento de un sistema en estos dos dominios resulta complejo porque el alumno no suele llegar a conectar en su mente unos resultados con otros. Sin embargo, el uso de herramientas interactivas como Sysquake permite en una misma pantalla ver el efecto que tienen cambios en alguno de los parámetros del sistema en ambos dominios, por lo que resulta más sencillo entender que ambos dominios están correlacionados entre sí.

Una segunda ventaja adicional de estas técnicas es que los métodos analíticos tradicionales sirven para estudiar sistemas de primer o segundo orden puros y en cualquier otro caso las aproximaciones analíticas no son precisas, pero la simulación del comportamiento del sistema con ordenador sí lo es. Por este motivo, herramientas como Octave o MatLab se han hecho imprescindibles en el estudio de la ingeniería de control. Sin embargo, los resultados que se obtienen con estos programas carecen de interactividad y si el estudiante no obtiene el resultado deseado en el primer intento debe volver a modificar el código y simularlo de nuevo. Este proceso se simplifica con el uso de interfaces interactivos como el presente en Sysquake, ya que con el mismo programa se pueden simular todas las posibles variantes del sistema y ver el grado de ajuste de cada opción instantáneamente.

De forma que Sysquake se posiciona como la herramienta ideal para aplicar el diseño interactivo en la enseñanza de los sistemas de control (Guzmán Sánchez et al., 2016). Otra ventaja adicional es que se pueden generar programas ejecutables, por lo que no es necesaria una licencia adicional por cada estudiante que precise usar la herramienta. Esto no ocurre con la herramienta más extendida en la ingeniería de control que es Matlab y que precisa que cada ordenador en el que se utilice disponga de su propia licencia de uso. Aunque Matlab tiene una política de precios reducidos para estudiantes, en la mayoría de los casos sigue suponiendo un coste elevado para estos o para la institución que les da soporte, sobre todo en el caso de cursos introductorios.

Algunos investigadores han explorado la posibilidad de desarrollar herramientas interactivas basadas en Sysquake con el propósito de generar utilidades que se puedan usar en

cursos universitarios de ingeniería de control. Algunos ejemplos, la mayoría de ellos desarrollados en la UNED, son:

- GPCIT (Generalized predictive control interactive tool) (Guzmán Sánchez et al., 2005).
- ITSIE (Interactive tool for system identification education) (Guzmán Sánchez et al., 2016).
- ILM (Interactive learning modules) (Guzmán Sánchez et al., 2006).
- ITCLSD (Interactive tool for closed-loop sharpening linear control design) (Díaz Martínez et al., 2018, 2019).
- ITTSAE (Interactive tool for time series analysis education) (Díaz Martínez et al., 2014c, 2015, 2016).
- ITTSAE-TSA (Interactive tool for time series análisis education - time series analyser) (Díaz Martínez et al., 2014a).
- ITTSAE-TSG (Interactive tool for time series análisis education – time series generator) (Díaz Martínez et al., 2014b).
- LCSD (Linear control system design) (Díaz Martínez & Dormido Bencomo, 2017; Marin et al., 2020).
- RCLSD (Robust closed-loop shaping linear control systems design) (Díaz Martínez et al., 2021; Díaz Martínez, Costa-Castelló, et al., 2020; Díaz Martínez, Dormido Bencomo, et al., 2020).
- SISO-QFTIT (Single input – single output – quantitative feedback theory interactive tool) (Baldini et al., 2015a, 2015b, 2016; Costa-Castelló et al., 2018; Díaz Martínez, 2020; Luis et al., 2011).
- TIG (Template interactive generator) (Díaz Martínez et al., 2007, 2008; Díaz Martínez, Dormido Bencomo, & Aranda, 2006; Díaz Martínez, Dormido Bencomo, Aranda, et al., 2006; Díaz Martínez & Dormido Bencomo, 2007).

Algunas de estas herramientas siendo muy útiles, son demasiado complejas para un curso introductorio. En estos casos será conveniente tener un conjunto de fichas o ejemplos que ilustren el manejo de la herramienta y su fundamento teórico. Encontramos un ejemplo de esta metodología en (Guzmán Sánchez et al., 2016), donde se presentan fichas con los conceptos teóricos a desarrollar en el uso de una parte de la herramienta y ejercicios propuestos para aplicar la teoría introducida.

En la siguiente sección se profundiza en la propuesta realizada por este grupo de investigadores y se resume su metodología docente presentada en (Guzmán Sánchez et al., 2016; Guzmán Sánchez, Costa-Castelló, et al., 2012).

2.2.3 Metodología de aprendizaje interactivo

Como ya se ha dicho este apartado es un resumen de la propuesta presentada en (Guzmán Sánchez et al., 2016; Guzmán Sánchez, Costa-Castelló, et al., 2012). La idea clave es usar la interactividad para acelerar el proceso de enseñanza y hacer las lecciones más atractivas para los estudiantes. Un segundo objetivo será promover el trabajo individual de los estudiantes. Los componentes fundamentales de este proyecto serán las **fichas de aprendizaje** que se componen de los siguientes elementos:

- Un programa interactivo y eminentemente gráfico, que además sea autoexplicativo y que permita visualizar los conceptos que se quieren transmitir en esa ficha. El programa debe incluir ejemplos predefinidos pensados para ilustrar algunos casos particulares.

- Cada aplicación se acompañará de un pequeño dosier que describa los conceptos que pretende ilustrar de forma analítica. Se trata de que se llegue a entender el comportamiento de la aplicación. El dosier incluirá referencias a libros y artículos donde ampliar los conceptos introducidos.
- Aunque la aplicación sea autoexplicativa, el dosier debe integrar un pequeño manual de uso.
- La ficha de aprendizaje debe incluir algunos ejercicios para ser realizados usando la aplicación. El objetivo de estos ejemplos debe ser el de guiar el aprendizaje del estudiante forzándole a prestar atención en los puntos clave. Los ejercicios deben seguir la filosofía de *aprendizaje por descubrimiento* y combinar el conocimiento cualitativo obtenido al interaccionar con la herramienta con el conocimiento cuantitativo obtenido al resolver los ejercicios.

Como ejemplo se pueden revisar la ficha creada para la aplicación *Material to Understand the Effect of Adding a Real Zero in a SecondOrder Process* en (Guzmán Sánchez et al., 2016).

De las diferentes técnicas activas mencionadas previamente, la metodología de aprendizaje propuesta combina:

- Experimentación.
- Resolución de problemas.
- Diseño guiado.
- Enseñanza con apoyo de ordenadores.
- Simulación.

Esto no quiere decir esta metodología que vaya a funcionar con todos los perfiles de aprendizaje de los estudiantes, pero sí con una gran cantidad. Se deberá evaluar en un experimento la idoneidad de este diseño y buscar posibilidades de mejora.

2.3 Estado del arte.

En este apartado se van a estudiar las herramientas software disponibles para la identificación de sistemas, centrando su estudio en las utilidades que tienen estas herramientas para la generación de señales de entrada a los sistemas a identificar.

La herramienta Matlab de MathWorks dispone de una serie de toolbox orientados a la regulación automática y el control. Uno de estos toolbox es el de identificación de sistemas[4] que posee funciones de apoyo y bloques de simulink diseñados para esta tarea. La función que permite generar una señal con características específicas para estimular un sistema y estudiar su respuesta es *idinput* (MathWorks, 2022). Con esta función se pueden generar tanto señales PRBS como multiseno, que son objeto de este TFM. Usando diferentes toolbox se pueden generar otros tipos de señales:

- SOH (Multisine sum of harmonics), señales multiseno suma de armónicos. Estas señales pueden tomar cualquier valor entre un mínimo y un máximo. Se pueden generar usando la función *msinclip* del toolbox FDIDENT.
- DIB (Discrete Interval Binary) y DIT (Discrete Interval Ternary), señales de intervalo discreto binario y ternario. Se pueden generar utilizando la función *dibs* del toolbox FDIDENT.
- MLMH (Multi-level Multi-harmonic), señales multinivel multiarmónicas, se pueden generar usando el paquete de Matlab *multilev*. Este mismo paquete también se puede utilizar para generar señales DIB y DIT.

[4] System Identification Toolbox

- PRBS una segunda forma de generar este tipo de señal es con el paquete de Matlab *prs*, que además de señales PRBS, puede generar señales QRB (Quadratic residue binary), HAB (Hall binary) y TPB (twin prime binary).
- IRPRB (Inverse-repeat pseudo-random binary signals), señales binarias pesudo aleatorias repetida invertida. Esta señal es una modificación de la PRBS donde se añade al final la misma señal invertida, esto tiene la propiedad de eliminar los armónicos pares. Se puede generar usando el paquete *prs*.
- QRT (Quadratic residue ternary), estas señales son similares a las QRB donde el último dígito del periodo se ha hecho cero. Se pueden generar usando el paquete *prs*.

La herramienta ITSIE, Interactive Tool for System Identification Education (Guzmán Sánchez et al., 2009; Guzmán Sánchez, Rivera, et al., 2012) desarrollada por profesores de la Uned y basada en Sysquake. Esta herramienta permite la identificación de un sistema en lazo abierto de tipo SISO (Single input single output) de quinto grado y sujeto a múltiples fuentes de ruido. Esta herramienta permite la generación de señales PRBS y multiseno como sistema para excitar el sistema a identificar.

También se pueden encontrar algoritmos de generación de señales en libros y artículos como los de Tan y Godfrey (Godfrey et al., 2006; Tan & Godfrey, 2009, 2019, Capítulo 8). En el capítulo 8 de su libro publicado en el año 2019, se puede encontrar código de Matlab que por medio de funciones y pequeñas aplicaciones basadas en Matlab, permite la generación de señales de tipo:

- MLB, máximum length binary
- QRB, quadratic residue binary
- HAB, hall binary
- TPB, twin prime binary
- QRT, quadratic residue ternary
- PRML, pseudo random multilevel
- GFs, Galois fields
- Multisine, tomado la fase inicial del algoritmo
  - Schroeder
  - Random
- DIB, discrete interval binary
- Señales de espectro óptimo
- MLMH, multilevel multiharmonic

La mayoría de estas señales escapan del alcance de este TFG donde solo trataremos las señales PRBS de tipo MLB y las señales multiseno.

En un estudio de la literatura reciente sobre la generación de señales para la identificación de sistemas, se han encontrado muchas referencias a nuevos sistemas de análisis utilizados en la identificación de sistemas tanto lineales como no lineales (Doyle et al., 2015; González et al., 2021; Pillonetto et al., 2022; Sarker et al., 2022; Steenis & Rivera, 2011; Winward et al., 2022), pero no se han encontrado nuevas referencias a herramientas de generación de las señales necesarias para los citados sistemas de identificación.

Otra categoría de sistemas de generación de señales para la identificación de sistemas, es la de generadores de señales físicas para aplicar a sistemas reales en vez de a simulaciones. Un ejemplo lo encontramos en el diseño de generadores PRBS y PRIS (Pseudorandom Impulse Sequence) propuestos por Mwaniki (2020).

3 Diseño de la señal de excitación: señales PRBS y señales multiseno.

Las señales periódicas se han utilizado ampliamente en la identificación de sistemas (Darnell, 1993; van den Bos, 1991). Según Tan (Tan & Godfrey, 2002), existen dos categorías para las señales de excitación periódicas:

1. Las señales optimizadas por ordenador. Señales de este tipo son:
    a. Las señales multiseno construidas como una combinación de señales armónicas (Tan & Godfrey, 2019). Estas pueden ser generadas, por ejemplo, usando el toolbox de identificación de sistemas de Matlab (Ljung et al., 2012).
    b. Señales multinivel y multiarmónicas (McCormack et al., 1995).
2. Las señales PRBS (Pseudorandom binary signal, señal binaria pseudoaleatoria). Dentro de este tipo de señales destacan las siguientes:
    a. Señales binarias de longitud máxima (MLB, Maximum length binary) (Zierler, 1959), que también se pueden generar usando el toolbox de identificación de sistemas de MatLab (Ljung et al., 2012).
    b. Señales binarias de longitud máxima multinivel, que se pueden generar a partir del paquete GALOIS (Barker, 2000).

3.1 Importancia de las señales de entrada en la identificación de sistemas.

En los años 60 del siglo pasado surge la necesidad de mejorar el control de los procesos industriales existentes o de diseñar nuevas plantas optimizadas. Esto llevó a un interés por mejorar el conocimiento en la dinámica de sistemas. Los métodos basados en el dominio del tiempo fueron los primeros utilizados, pero en muchos casos la respuesta al escalón o al impulso no ofrecían resultados satisfactorios. Esto era bien porque la presencia de ruido distorsionaba los resultados o porque los tiempos de asentamiento eran tan largos que era complicado mantener el proceso estable durante tanto tiempo.

Esto movió el foco de interés al uso de señales binarias pseudoaleatorias que aplicadas a la entrada del sistema permitían obtener una relación matemática entre la entrada y salida del sistema. Lo que finalmente llevó a conseguir una buena estimación de la respuesta al impulso de la planta y de su función de transferencia. Esté método se usó con éxito en el control de centrales nucleares, procesos químicos, refinerías, fábricas de papel y acero. Estas señales binarias se añaden a la señal de mando como una señal de perturbación y del estudio de los resultados se llega a un modelo del sistema.

Por tanto, en el proceso de identificación de un sistema o planta es muy importante la selección de señales de perturbación adecuadas que permitan el diseño de un experimento que lleve a la identificación de la planta. Un mejor diseño de estas señales tendrá como resultado un modelo más preciso y esto llevará a un mejor rendimiento del controlador. Técnicas de control avanzadas, como por ejemplo el control predictivo, necesitan de un modelo preciso para maximizar su capacidad (Mayne, 2014). Ya que en cada ciclo del proceso se reevalúa el modelo comparando su salida predicha con la real para optimizar el proceso en el siguiente ciclo.

Más aun, un diseño cuidadoso de estas señales permitirá descubrir la parte de contribución lineal, distorsiones no lineales y perturbaciones. Esto dará al ingeniero de control herramientas para decidir si debe aplicar un control de tipo lineal, adaptativo de parámetros variables o no lineal.

Cuando hablamos de la identificación de sistemas en la industria, la señal insertada en el sistema debe recopilar la máxima cantidad de información dentro del tiempo asignado al

experimento. En proyectos de control predictivo, el diseño de las señales, testeo de la planta e identificación del modelo puede consumir hasta el 50% del tiempo del proyecto (Darby & Nikolaou, 2012). En este aspecto señales de banda ancha que exciten al sistema con varias frecuencias de forma simultánea son preferibles a señales de una única frecuencia que obliguen a repetir el experimento múltiples veces.

Si el objetivo perseguido con la identificación de un sistema es múltiple, probablemente se precisarán diferentes señales de entrada. Por ejemplo, la identificación de no linealidades en el sistema, precisará de una señal distinta a la necesaria para identificar la parte lineal principal del mismo que a la vez minimice los efectos no lineales. También se deben adquirir compromisos en el diseño de estas señales. En el caso de la selección de la amplitud, cuanto mayor sea ésta mejor será la relación señal ruido en la salida del sistema, pero por contra, se debe evitar saturar el sistema o sacarlo de su régimen de funcionamiento lineal.

3.1.1 Tipos de señales

Las principales características que debe cumplir una señal de entrada utilizada para la identificación de sistemas son:

- La señal debe contener suficientes frecuencias para poder excitar todas las partes del sistema, se dice que proporcione una excitación persistente. Una señal tiene **excitación persistente** de orden *N* si su espectro es no nulo para *N* frecuencias diferentes.
- La entrada tiene que tener una amplitud limitada.
- Es preferible que la señal sea periódica y reproducible.

Existen diferentes tipos de señales disponibles para la identificación de sistemas, entre las más usadas se encuentran las siguientes:

- **Señal escalón**: una señal escalón toma valor cero para t<0 y A para t>0, se utiliza comúnmente para obtener las características principales de un sistema en el dominio del tiempo. Estas características fundamentales son:
  a. el tiempo de asentamiento,
  b. la sobreoscilación y
  c. el error de posición.

  Para el problema de la identificación de un sistema, estas características permiten conocer algunos parámetros de la función de transferencia cuando es conocida previamente la forma de esta. Por ejemplo, saber la ubicación del polo dominante del sistema o si éste es de tipo real o complejo. En el dominio de la frecuencia permite obtener la ganancia estática del sistema.
  Cuando el valor del escalón A=1 se denomina de escalón unitario.
  La transformada de Laplace de esta señal es A/s y su transformada Z en caso de ser una secuencia es $A/(1-z^{-1})$.
- **Señal impulso (pulso)**: Las señales impulso se usan normalmente como señales de excitación en el análisis de sistemas porque son fáciles de aplicar y de medir sus resultados. Una segunda ventaja es que es la señal de excitación más corta posible (Tan & Godfrey, 2019). La función pulso $\delta_{\Delta}(t - t_k)$ se puede definir como:

$$\delta_{\Delta}(t-t_k)=\begin{cases}0\ para\ t<t_k\\ \dfrac{1}{\Delta t}\ para\ t_k\le t\le t_k+\Delta t\\ 0\ para\ t\ge t_k+\Delta t\end{cases} \quad [1]$$

La función pulso $\delta_{\Delta}(t-t_k)$ tiene área unitaria. Cuando Δt se aproxima a cero, se convierte en la función impulso o delta de Dirac. La función impulso $\delta(t-t_k)$ tiene las propiedades:

$$\int_0^{\infty}\delta(t-t_k)dt=1 \quad [2]$$

$$\int_0^{\infty}f(t)\delta(t-t_k)dt=f(t_k)$$

Estas propiedades se cumplen p1ara cualquier función *f(t)* que sea continua en t_k

- **Señal PRBS**: Es una señal determinística y por lo tanto reproducible, pero con propiedades propias de una señal aleatoria. Una posible forma de generar una señal PRBS es usando registros de desplazamiento lineales realimentados de *n* bits, lo que genera señales de longitud máxima de 2^n-1 bits. El ancho de banda de la señal generada puede alterarse modificando el periodo del reloj del registro. Las señales PRBS posee un factor de cresta mínimo de 1.

- **Señal multiseno**: una señal multiseno es una combinación de señales sinusoidales de distinta frecuencia y fase inicial.

$$u[k]=\sum_{j=1}^{N}a_j\sin(w_jk+\phi_j)\quad 0\le w_1\le w_2\le\cdots\le w_N\le \quad [3]$$

Este tipo de señales son muy útiles para determinar el comportamiento del sistema a las frecuencias de la señal multiseno. Por otra parte, la fase inicial de las ondas seno tiene una influencia relevante sobre el factor de cresta de la señal.
Uno de los posibles inconvenientes de este tipo de señales es que las entradas del sistema son normalmente discretas y solo aceptan un número limitado de valores. Sin embargo, la señal sinusoidal tiene infinitos valores diferentes dentro de un rango acotado (Tan & Godfrey, 2002).

3.1.2 Índices de calidad de las señales de perturbación

En este apartado se van a definir una serie de parámetros que será de utilidad a la hora de caracterizar las señales estudiadas en este TFM. Estos parámetros permiten determinar la idoneidad de una determinada señal a la hora de usarlos en la identificación de sistemas. Dado el mecanismo de generación de señales que se utilizará, se asume que las señales con las que se trabaja son de tipo discreto.

3.1.2.1 Factor de cresta

En muchos casos la media de la señal es un valor interesante. Sin embargo, en el caso de señales con valores positivos y negativos suele ser más interesante el valor rms (root mean square value), que se define como la raíz cuadrada del cuadrado del valor instantáneo de la señal en un momento dado, ya que en muchos casos el valor medio de la señal a lo largo de un intervalo de tiempo suele ser nulo.

Para el caso de una señal discreta se puede expresar por medio de la ecuación [4].

$$u_{rms} = \sqrt{\frac{1}{N}\sum_{0}^{N-1} u(k)^2} \qquad [4]$$

En el caso de señales continuas $u=u(t)$ o discretas $u=u(k)$ el valor u_{pico} será el valor máximo de la señal considerado su valor absoluto en el intervalo de tiempo estimado.

De forma que se define el factor de cresta como (Tan & Godfrey, 2019):

$$u_{cresta} = \frac{u_{pico}}{u_{rms}} \qquad [5]$$

3.1.2.2 Índice de perturbación de una señal PIPS

La dispersión de una señal estacionaria u, puede ser expresado por su medida normalizada λ_u definida por Godfrey (Godfrey et al., 1999) como:

$$\lambda_u = \frac{u_{max} - u_{min}}{2\sigma_u} \qquad [6]$$

Donde u_{max} es el valor máximo de la señal, u_{min} el valor mínimo y σ_u su desviación estándar. El valor de la dispersión λ_u no puede ser inferior a 1 pero no tiene un límite superior. Frente a este índice es preferible uno que se exprese porcentualmente, donde 0% signifique el peor rendimiento posible para la señal y 100% el mejor.

Esto lleva a la definición del índice de perturbación de señales PIPS (Performance index for perturbation signals) (Godfrey et al., 1999).

$$PIPS = 100 * {\lambda_u}^{-1} = \frac{200\sqrt{u_{rms}^2 - u_{mean}^2}}{u_{max} - u_{min}}\,\%, u_{max} > u_{min} \qquad [7]$$

De forma que resulta que PIPS es independiente del valor medio y de la amplitud de la señal. Un ejemplo de señal que obtiene un índice PIPS del 100% es una señal binaria donde u_{max} y u_{min} estén distribuidas uniformemente en el tiempo.

En el caso de una señal binaria de máxima longitud MLBS[5], la expresión [7] se puede simplificar como:

[5] Maximum length binary sequence

$$PIPS = 100\sqrt{\frac{N^2-1}{N^2}}\% \quad [8]$$

Siendo *N* la longitud de la secuencia MLBS y que tiende al 100% conforme N aumenta.

3.1.2.3 Índice de amigabilidad

Doyle (Doyle et al., 2015) define el índice de amigabilidad *f* de una secuencia de entrada u_k, $k \in \{1,..,N\}$ como

$$f(\%) = 100(1 - \frac{n_T}{N-1}) \quad [9]$$

Donde *N* es la longitud de la secuencia y n_T es el número de cambios de valor de la señal. Una entrada será 100% amigable si la entrada no cambia. Por ejemplo, una señal de tipo escalón se considerará 100% amigable. Mientras que, una señal cuadrada, que cambia en cada ciclo de reloj será 0% amigable.

3.2 Introducción a la variable aleatoria

3.2.1 Concepto de variable aleatoria

Un experimento aleatorio es por ejemplo lanzar una moneda al aire y anotar cuantas caras o cruces recogemos en un número determinado de tiradas. La probabilidad de que en cada tirada aparezca cara o cruz es siempre la misma ya que cada tirada es independiente de la anterior. Algo similar ocurre cuando lanzamos un dado de 6 caras, el resultado será siempre uno de los 6 valores posibles.

Sin embargo, hay circunstancias en las que el valor esperado del experimento es desconocido. Las resistencias usadas en electrónica tienen tolerancias que van del 0.1% al 10%, esto quiere decir que el valor de cada resistencia es desconocido pero acotado dentro de un rango definido por su tolerancia. Si por ejemplo, se tienen resistencias de 10 ohm de una tolerancia del 1%, se sabrá que su valor estará entre 9.9 ohm y 10.1 ohm. Si tenemos una caja de resistencias de este tipo y sacamos una al azar podremos obtener cualquier valor dentro de ese rango, pero si buscamos una resistencia de 10.0050 ohm, la probabilidad de encontrarla dentro de la caja es cero. Diremos que el valor de la resistencia sacada del interior de la caja es una **variable aleatoria**.

Se puede asociar este concepto de variable aleatoria con el de una **función temporal aleatoria**. Considere el valor de la señal *x(t)* representada en la Figura 6, esta señal en un instante determinado puede tomar cualquier valor dentro de un rango conocido *±A*. Si realizamos el experimento de medir el valor de la señal en el instante t_i, obtendremos el valor $X(t_i)$ o X_i y decimos que es una muestra de la función temporal aleatoria. En otro instante de tiempo podremos obtener cualquier otro valor ya que el número de valores diferentes que podemos encontrar dentro del rango de la función es infinito.

Desde la perspectiva que nos ocupa, una variable aleatoria es la descripción numérica del resultado de un experimento aleatorio. Si $S=\{\alpha\}$ es el conjunto de todos los posibles valores de salida del experimento aleatorio. Cuando la salida es α, la variable aleatoria *X* tendrá el valor denotado como $X(\alpha)$. Desde este punto de vista la variable aleatoria es simplemente una función que obtiene valores reales sobre el espacio de la muestra. En aplicaciones de ingeniería

normalmente no es necesario considerar explícitamente el espacio de la muestra. Normalmente es solo necesario ser capaz de asignar probabilidades a diferentes eventos asociados con las variables aleatorias de interés. Estas probabilidades normalmente se pueden inferir de las propiedades físicas de la variable aleatoria analizada.

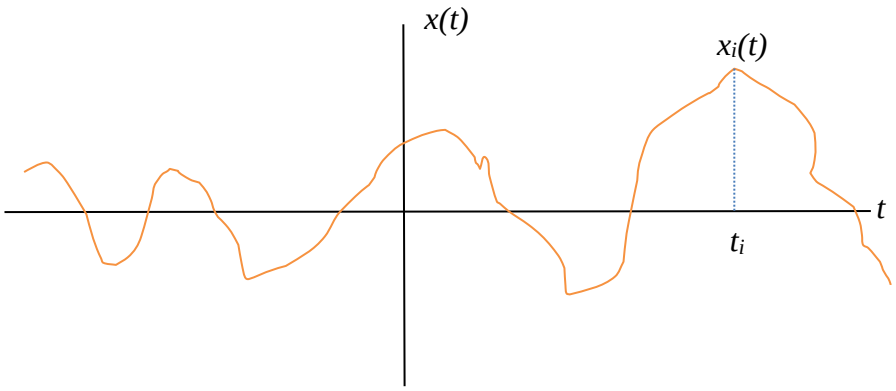

Figura 6: función temporal aleatoria.

3.2.2 Funciones de distribución de probabilidad

Sea *X* una variable aleatoria y *x* cualquier valor permitido para esa variable. La **función de distribución de probabilidad** se define como que la probabilidad del evento de observar la variable aleatoria *X* sea menor o igual que el valor *x*. Esto es:

$$F_x(x) = \Pr\,(X \leq x) \qquad [10]$$

Dado que la función de distribución de probabilidad es una probabilidad, debe satisfacer los axiomas y propiedades de la probabilidad. Sin embargo, también es una función de *x*, así como el valor posible de la variable aleatoria *X* y por último debe estar definida para todos los valores de *x*. Algunas de estas propiedades y restricciones para la función de distribución de probabilidad $F_x(x)$ son:

1. $0 \leq F_x(x) \leq 1, \quad -\infty < x < \infty$
2. $F_x(-\infty) = 0, \; F_x(\infty) = 1$ [11]
3. $F_x(x)$ es no decreciente cuando *x* aumenta.
4. $\Pr(x_1 < X \leq x_2) = F_x(x_2) - F_x(x_1)$

La Figura 7 muestra un posible ejemplo de función de distribución de probabilidad donde la variable x puede tomar valores entre $-\infty$ y ∞. La función de distribución de probabilidad también puede ser utilizada para expresar el evento de que se observe que la variable aleatoria X sea mayor que x. Dado que este evento es simplemente el complementario del definido por la función $F_x(x)$, éste se puede expresar como:

$$\Pr(X > x) = 1 - F_x(x) \qquad [12]$$

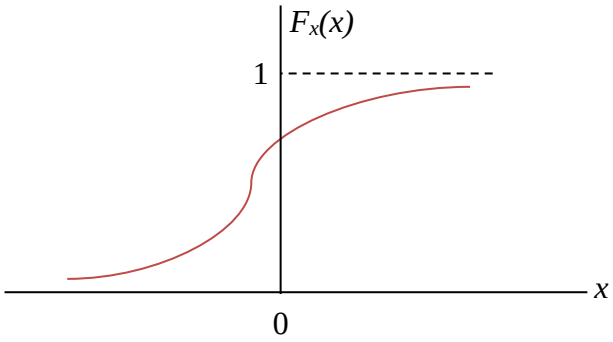

Figura 7: Ejemplo de función de distribución de probabilidad

3.2.3 Funciones de densidad de probabilidad

Aunque una función de distribución es una descripción completa de un modelo de probabilidad para una variable aleatoria, no es la forma más conveniente para muchos casos prácticos. Para estos es preferible usar la derivada de *F(x)* en vez de *F(x)*. Esta derivada es denominada la **función de densidad de probabilidad** y cuando existe es definida por la siguiente expresión.

$$f_x(x) = \lim_{e \to 0} \frac{F_x(x+e) - F_x(x)}{e} = \frac{dF_x(x)}{dx} \qquad [13]$$

El significado físico de la función de densidad de probabilidad se describe mejor en términos de la probabilidad del elemento $f_x(x)dx$ que se puede interpretar como:

$$f_x(x)dx = \Pr\,(x < X \leq x + dx) \qquad [14]$$

La ecuación anterior indica que la probabilidad del elemento $f_x(x)dx$ es la probabilidad del evento de que la variable aleatoria *X* caiga en el rango de posibles valores entre *x* y *x+dx*.

Dado que $f_x(x)$ es una función de densidad y no una probabilidad, no es necesario que su valor sea inferior a 1. Sus propiedades principales son:

1. $f_x(x) \geq 0 \quad -\infty < x < \infty$
2. $\int_{-\infty}^{\infty} f_x(x)dx = 1$
3. $F_x(x) = \int_{-\infty}^{x} f_x(u)du$
4. $\int_{x_1}^{x_2} f_x(x)dx = \Pr\,(x_1 < X \leq x_2)$

[15]

3.2.4 Valores medios y momentos

Uno de los conceptos más importantes asociados con métodos estadísticos es poder encontrar los valores medios de variables aleatorias o de funciones de variables aleatorias. La forma más común para encontrar valores medios de funciones en ingeniería es integrándolas en un intervalo de tiempo y dividiéndolas por la longitud del intervalo. En el caso de funciones del tiempo carece de sentido considerar solo el valor de la función en un instante de tiempo y será

necesario encontrar el valor medio integrando sobre el rango de posibles valores que la variable aleatoria puede tomar. La siguiente expresión representa este cálculo.

$$\bar{X} = E[X] = \int_{-\infty}^{\infty} x f(x) dx \qquad [16]$$

El símbolo $E[X]$ se conoce como el **valor esperado de *X*** o la **esperanza matemática de *X***. En muchos casos que tienen un interés práctico, el valor medio de una variable aleatoria es igual que la media temporal de cualquier función muestreada del proceso aleatorio al que pertenece la variable aleatoria. En estos casos, encontrar el valor medio de una tensión o corriente aleatoria es como encontrar su componente de continua.

El valor esperado de una función *x* se puede obtener de una forma similar con:

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x) dx \qquad [17]$$

Una función particularmente interesante es $g(x) = x^n$ ya que lleva a la expresión del **momento general** de la variable aleatoria:

$$\overline{X^n} = E[X^n] = \int_{-\infty}^{\infty} x^n f(x) dx \qquad [18]$$

Los momentos más importantes se dan para los valores de *n=1* que es el **valor medio** visto previamente y para *n=2* que lleva a la expresión de **valor cuadrático medio**.

$$\overline{X^2} = E[X^2] = \int_{-\infty}^{\infty} x^2 f(x) dx \qquad [19]$$

La importancia del valor cuadrático medio radica en que se puede interpretar como la media del cuadrado de una tensión o corriente aleatorias. En estos casos el valor cuadrático medio es proporcional a la potencia media disipada en una resistencia y su raíz cuadrada es el valor rms o valor efectivo del voltaje o corrientes aleatorios.

También es posible definir el **momento central**, que son los momentos de la diferencia entre la variable aleatoria y su valor medio. Entonces el momento central enésimo será:

$$\overline{(X - \bar{X})^n} = E[(X - \bar{X})^n] = \int_{-\infty}^{\infty} (x - \bar{X})^n f(x) dx \qquad [20]$$

El momento central para *n=1* es cero y para *n=2* tenemos un caso muy importante denominado **varianza** que normalmente se simboliza con σ^2

$$\sigma^2 = \overline{(X - \bar{X})^2} = \int_{-\infty}^{\infty} (x - \bar{X})^2 f(x) dx \qquad [21]$$

La varianza también puede ser expresada de una forma alternativa usando la suma de valores esperados. Esto es:

$$E[X_1 + X_2 + \cdots + X_m] = E[X_1] + E[X_2] + \cdots + E[X_m] \quad [22]$$

Entonces

$$\sigma^2 = E[(X - \bar{X})^2] = E[X^2 - 2X\bar{X} + (\bar{X})^2] = E[X^2] - 2E[X]\bar{X} + (\bar{X})^2 = \overline{X^2} - 2\bar{X}\bar{X} + (\bar{X})^2 = \overline{X^2} - (\bar{X})^2 \quad [23]$$

De forma que la varianza es la diferencia entre el valor cuadrático medio y el cuadrado del valor medio. La raíz cuadrada de la varianza σ se le conoce como la **desviación estándar**.

3.3 Introducción a los procesos aleatorios

Un **proceso aleatorio** es una colección de funciones del tiempo con una probabilidad asociada. La descripción de esta probabilidad consiste en la unión de las funciones de densidad de probabilidad de las funciones del proceso en instantes específicos del tiempo.

La colección completa de funciones del tiempo es una agrupación y se designará como $\{x(t)\}$, donde cualquier miembro particular de la agrupación $x(t)$ es una función muestreada de la agrupación. En general solo una función muestreada del proceso aleatorio puede ser observada. El resto de funciones muestreadas representan todas las otras posibles realizaciones que podrían haber ocurrido pero que no lo hicieron.

Una función muestreada aleatoria se representa como $X(t)$. Los valores de $X(t)$ en un instante t_1 definen una variable aleatoria llamada $X(t_1)$ o simplemente X_1.

3.3.1 Proceso aleatorio estacionario y no estacionario

Se puede definir una función de densidad de probabilidad para una variable aleatoria de la forma $X(t_1)$, pero no se ha dicho nada de si existe una dependencia entre la función de densidad y el instante de tiempo t_1. Si la función de densidad del proceso no depende de la elección del origen temporal, el **proceso** se dice que es **estacionario**. En este caso las medias y momentos discutidas previamente son constantes y no dependen del valor absoluto del tiempo.

Si cualquiera de las funciones de densidad de probabilidad, cambian con la elección del origen temporal, decimos que el **proceso** es **no estacionario**. En este caso, una o más de las medias o momentos dependen del tiempo.

En un sentido estricto, no hay procesos aleatorios estacionarios que existan físicamente, ya que cualquier proceso tuvo que comenzar en algún momento del pasado y terminar en algún momento del futuro y evolucionar o envejecer con el tiempo. También ocurre que hay muchos sistemas físicos en los que el proceso no cambia de forma significativa mientras es observado. En estos casos asumir que es un proceso de tipo estacionario lleva a un modelo matemático bastante aproximado a la realidad.

3.3.2 Proceso ergódico y no ergódico

Algunos procesos aleatorios estacionarios tienen la propiedad de que casi todos los miembros de su agrupación exhiben el mismo comportamiento estadístico que la agrupación al completo. Entonces es posible determinar su comportamiento estadístico examinando solo una función muestreada tipo. A estos procesos se les denomina **ergódicos**.

Para un proceso ergódico, las medias y momentos se pueden determinar con las medias temporales o con las medias de la agrupación. Por ejemplo, el momento enésimo de un proceso ergódico vendrá dado por:

$$\overline{X^n} = \int_{-\infty}^{\infty} x^n f(x)dx = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} X^n(t)dt \qquad [24]$$

Se debe resaltar que esta condición no puede existir a no ser que el proceso sea estacionario. De forma que un proceso ergódico es también estacionario.

Un proceso que no posee la propiedad anterior es **no ergódico**. Todos los procesos no estacionarios son no ergódicos, aunque también es posible que un proceso estacionario sea no ergódico.

3.4 Introducción a las funciones de correlación

Decimos que dos variables aleatorias están correlacionadas entre sí cuando una depende de alguna manera de la otra. Si tenemos dos variables aleatorias X e Y que toman valores x e y, entonces el valor esperado para su producto $E[XY]$ es conocido como su **correlación**. La ecuación [25] la define como:

$$E[XY] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xyf(x,y)dxdy = \overline{XY} \qquad [25]$$

Si las dos variables aleatorias son definidas como muestras de un proceso aleatorio en dos instantes de tiempo diferentes, entonces este valor esperado depende en como de rápido puedan cambiar estas funciones del tiempo. Se podría esperar que las variables aleatorias estuvieran fuertemente correlacionadas cuando los dos instantes de tiempo son muy cercanos, ya que, éstas no pueden cambiar lo suficientemente deprisa como para ser muy diferentes. Por otra parte, se podría esperar muy poca correlación entre las muestras de las variables cuando los instantes de tiempo estén muy separados, ya que será más sencillo que sean diferentes, sobre todo si el proceso muestreado no es periódico. Dado que la correlación depende en como de rápidamente cambien los valores de una variable aleatoria con respecto al tiempo, se espera que esta correlación esté también relacionada con la forma en que el contenido de energía de un proceso aleatorio está distribuido con respecto a la frecuencia. Esto es porque una función temporal debe tener una energía significativa en altas frecuencias para ser capaz de cambiar rápidamente con respecto al tiempo.

La definición de correlación presentada previamente genera un valor numérico ya que las variables aleatorias no tienen por qué estar asociadas a una función temporal. Sin embargo, es posible definir una nueva función en que cada par de variables aleatorias estén relacionadas por la separación temporal entre ambas y la correlación sea una función de dicha separación. De forma que se hace necesario definir la función de correlación en la que su argumento es el tiempo de separación entre dos variables aleatorias. Si las dos variables provienen del mismo proceso aleatorio, entonces esta función se denomina **función de autocorrelación**. Si por el contrario provienen de procesos aleatorios diferentes entonces se denomina **función de correlación cruzada** (G. R. Cooper & Mcgillem, 1997).

Si X(t) es una función muestra de un proceso aleatorio, se pueden definir las variables aleatorias.

$$X_1 = X(t_1)$$
$$X_2 = X(t_2) \qquad [26]$$

Entonces a partir de la ecuación [25] la función de autocorrelación se define como

$$R_{xx}(t_1,t_2) = E[X_1X_2] = \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} x_1x_2 f(x_1,x_2)dx_2 \quad [27]$$

donde $f(x_1,x_2)$ es la función de densidad probabilidad conjunta.

Esta definición es válida tanto para procesos estacionarios como no estacionarios, recordemos que un proceso o señal estacionaria sus componentes de frecuencia no cambian con el tiempo. La media de la señal x se designará como $\mu_x(t)$.

Se define la **covarianza** de dos variables aleatorias X e Y que tienen medias μ_x y μ_y como:

$$E\big[(X-\mu_x)(Y-\mu_y)\big] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\mu_x)(y-\mu_y)f(x,y)dxdy = \overline{(X-\bar{X})(Y-\bar{Y})} \quad [28]$$

La covarianza es más conveniente que la correlación en el caso en que las variables tengan una media no nula. Se define la **función de autocovarianza** $C_{XX}(t_1,t_2)$ como:

$$C_{xx}(t_1,t_2) = E\{[x(t_1)-\mu_x(t_1)].[x(t_2)-\mu_x(t_2)]\} \quad [29]$$

De forma que a partir de ambas definiciones es fácilmente deducible que:

$$R_{xx}(t_1,t_2) = C_{xx}(t_1,t_2) + \mu_x(t_1)\mu_x(t_2) \quad [30]$$

Las señales que vamos a estudiar son de tipo estacionario. Aunque es difícil encontrar señales que sean puramente estacionarias, hablaremos de señales estacionarias en sentido amplio o señales débilmente estacionarias, que solo precisan que su valor medio no cambie con el tiempo. La función de autocorrelación solo depende de la diferencia de tiempo $|t_2 - t_1|$ y haciendo esta diferencia igual a τ tenemos:

$$R_{xx}(\tau) = E[x(t)x(t+\tau)] \quad [31]$$

Para la mayoría de las señales estacionarias que tienen un interés práctico, se puede demostrar que el **valor esperado**, es equivalente a la media de la señal realizada en un intervalo de tiempo infinito. A esto se le conoce como la hipótesis ergódica. Para una señal ergódica y estacionaria x(t), el valor de su esperanza matemática:

$$E[x(t)] = \mu_x = \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} x(t)dt \quad [32]$$

De forma que la función de autocorrelación será en este caso:

$$R_{xx}(\tau) = \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} x(t)x(t+\tau)dt \quad [33]$$

y el valor de la función de autocovarianza de la señal será:

$$C_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} [x(t) - \mu_x][x(t+\tau) - \mu_x]dt \quad [34]$$

Si x(t) es una señal periódica de periodo T, entonces las medias infinitas de las que se hablaban previamente se convierten en medias en un tiempo kT donde k es un entero positivo mayor o igual a uno. Por ejemplo, para una señal periódica,

$$R_{xx}(\tau) = \frac{1}{T} \int_{0}^{T} x(t).x(t+\tau)dt \quad [35]$$

Se obtendrá un resultado similar para la función de correlación cruzada $E[x(t).y(t+\tau)]$ que se realice entre dos señales ergódicas x(t) e y(t):

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t).y(t+\tau)dt \quad [36]$$

Esta expresión general se podrá simplificar en el caso de que ambas funciones sean periódicas de periodo T.

$$R_{xy}(\tau) = \frac{1}{T} \int_{0}^{T} x(t).y(t+\tau)dt \quad [37]$$

El sentido de esta operación es que tomamos la señal y desfasada τ unidades de tiempo y multiplicamos las señales $x(t)$ e $y(t+\tau)$. La integral de este producto dividido por el periodo T es el valor tomado como correlación cruzada.

3.4.1 Propiedades de las funciones de autocorrelación para señales estacionarias

A partir de la ecuación [33] en el caso en que $\tau = 0$, tenemos

$$R_{xx}(0) = E[x^2(t)] \quad [38]$$

De forma que la función de autocorrelación de una señal con retardo nulo es igual al **error cuadrático medio**. De igual forma, a partir de la ecuación [34] tenemos:

$$C_{xx}(0) = E[(x(t) - \mu_x)^2] = \sigma_x^2 \quad [39]$$

Por lo que la función de autocovarianza de la señal con retardo nulo es igual a la varianza de la señal.

Para un proceso estacionario

$$R_{xx}(\tau) = R_{xx}(-\tau) \quad [40]$$

Por lo tanto, la función de autocorrelación es una función par de τ, lo mismo ocurre para la función de autocovarianza.

$$R_{xx}(0) \geq |R_{xx}(\tau)| \quad [41]$$

El valor de la función de autocorrelación para $\tau = 0$ es mayor o igual que la magnitud de la función de autocorrelación para cualquier otro valor de τ. Para mostrar esta relación, sabemos que:

$$E\{[x(t) \pm x(t+\tau)]^2\} \geq 0 \quad [42]$$

Reorganizando la anterior inecuación

$$E[x^2(t)] + E[x^2(t+\tau)] \geq \pm 2E[x(t).x(t+\tau)] \quad [43]$$

Entonces finalmente tenemos:

$$2R_{xx}(0) \geq \pm 2R_{xx}(\tau) \quad [44]$$

Expresión que nos permite justificar la ecuación [41].
Si tenemos que z(t)=x(t)+y(t) entonces

$$R_{zz}(\tau) = R_{xx}(\tau) + R_{xy}(\tau) + R_{yx}(\tau) + R_{yy}(\tau) \quad [45]$$

En el caso en que x e y no estén correlacionadas, se tiene que:

$$R_{zz}(\tau) = R_{xx}(\tau) + R_{yy}(\tau) \quad [46]$$

En general las propiedades mostradas en este apartado no se pueden aplicar a las funciones de correlación cruzada.

3.4.2 Cálculo de funciones de correlación en la práctica

El cálculo teórico de las funciones de autocorrelación y correlación cruzada, se mostró en las ecuaciones [36] y [37].

$$R_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x(t+\tau)dt$$

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t).y(t+\tau)dt$$

Este cálculo implica el siguiente proceso:

1. Desplazar la señal y(t) adelantándola un tiempo τ
2. Multiplicar los valores de las señales
3. Calcular la media del resultado sobre un largo periodo de tiempo (infinito en teoría)

Para poder realizar un cálculo realista, se deben aplicar algunos cambios a este procedimiento. En primer lugar, en vez de adelantar la señal y, atrasaremos la señal x, ya que a no ser que tengamos un conocimiento previo de la señal el cálculo no sería posible. Entonces las expresiones anteriores quedan sustituidas por:

$$R_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x(t+\tau)dt$$
$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t).y(t+\tau)dt \quad [47]$$

En segundo lugar, será necesario calcular la función de correlación en un tiempo finito T_f, de forma que el resultado que obtendremos será aproximado con respecto al verdadero.

$$\hat{R}_{xx}(\tau) = \frac{1}{T_f} \int_{0}^{T_f} x(t-\tau)x(t)dt$$
$$\hat{R}_{xy}(\tau) = \frac{1}{T_f} \int_{0}^{T_f} x(t-\tau).y(t)dt \quad [48]$$

Cuando solo tengamos disponible una longitud limitada de la señal, habrá una pérdida al final dado que el tiempo τ es distinto de cero. Las ecuaciones [48] en este caso quedan modificadas como:

$$\hat{R}_{xx}(\tau) = \frac{1}{T_f - \tau} \int_{0}^{T_f-\tau} x(t-\tau)x(t)dt$$
$$\hat{R}_{xy}(\tau) = \frac{1}{T_f - \tau} \int_{0}^{T_f-\tau} x(t-\tau).y(t)dt \quad [49]$$

Lo normal es que el periodo de desfase τ sea muy inferior a la longitud de la secuencia T_f.

Si las funciones de correlación son tratadas digitalmente con un total de N muestras distribuidas equidistantemente en el tiempo T_f, entonces las expresiones de las ecuaciones [48], para una longitud de datos sin limitaciones, quedan como:

$$\hat{R}_{xx}(k) = \frac{1}{N} \sum_{r=0}^{N-1} x_{r-k}x_r$$
$$\hat{R}_{xy}(k) = \frac{1}{N} \sum_{r=0}^{N-1} x_{r-k}y_r \quad [50]$$

Mientras que para el caso en que la longitud de la secuencia esté limitada, se tendrán las siguientes expresiones.

$$\hat{R}_{xx}(k) = \frac{1}{N-k} \sum_{r=0}^{N-k-1} x_{r-k}x_r$$
$$\hat{R}_{xy}(k) = \frac{1}{N-k} \sum_{r=0}^{N-k-1} x_{r-k}y_r \quad [51]$$

Si x(t) es una función periódica de periodo T, entonces las ecuaciones [48] y [50] son exactamente las funciones de autocorrelación cuando $T_f = kT$, donde k es un entero mayor o igual que uno. Similarmente si x(t) e y(t) son funciones periódicas de periodo T las expresiones [48] y [50] son las funciones de correlación cruzada cuando $T_f = kT$.

Bendat y Piersol (1966), dedujeron una expresión para la varianza de una función de correlación cruzada $\hat{R}_{xy}(\tau)$ tomando medidas sobre un periodo de tiempo finito T_f en vez de para un tiempo infinito, asumiendo que x(t) e y(t) son señales con media cero y función de densidad de probabilidad gaussiana.

$$Var[R_{xy}(\tau)] = \frac{1}{T_f}\int_{-\infty}^{\infty}[R_{xx}(u)R_{yy}(u) + R_{xy}(u+\tau)R_{xy}(\tau-u)]du \qquad [52]$$

La correspondiente expresión para la varianza de la función de autocorrelación se obtiene al sustituir y por x,

$$Var[R_{xx}(\tau)] = \frac{1}{T_f}\int_{-\infty}^{\infty}[(R_{xx}(u))^2 + R_{xx}(u+\tau)R_{xx}(u-\tau)]du \qquad [53]$$

Entonces para $\tau = 0$, tenemos

$$Var[R_{xx}(0)] = \frac{4}{T_f}\int_{0}^{\infty}[R_{xx}(u)]^2 du \qquad [54]$$

Muchos sistemas de análisis calculan las funciones de correlación usando la transformada rápida de Fourier para pasar del dominio del tiempo a la frecuencia y viceversa. Este uso de la transformada rápida de Fourier hace que las funciones de correlación resultantes sean periódicas independientemente de si las señales originales lo eran o no, por lo que se debe tener cuidado al interpretar estos resultados.

3.5 Introducción a la transformada y series de Fourier para señales continuas

El creciente interés en la identificación de sistemas en el dominio del tiempo es principalmente debida a la eficiencia computacional del algoritmo de la transformada rápida de Fourier (Godfrey, 1993). Éste es un algoritmo de procesado de señal discreta. Por lo tanto, la interpretación de sus resultados debe hacerse con cuidado. En el dominio de la frecuencia es necesario distinguir cuidadosamente entre señales periódicas y no periódicas.

3.5.1 Señales no periódicas

El procesado de la señal en el dominio de la frecuencia está basado en la transformada de Fourier. Para una señal no periódica x(t), la transformada de Fourier continua es dada por la siguiente expresión:

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt = \int_{-\infty}^{\infty} x(t)\cos\omega t\, dt - j\int_{-\infty}^{\infty} x(t)\sin\omega t\, dt \qquad [55]$$

La señal x(t) se puede obtener de $X(j\omega)$ usando la transformada inversa de Fourier

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)\, e^{j\omega t} dt \qquad [56]$$

Otra pareja de ecuaciones de gran interés en la identificación de sistemas en el dominio del tiempo son la **densidad espectral de potencia o espectro de potencia** $S_{xx}(\omega)$:

$$S_{xx}(\omega) = \int_{-\infty}^{\infty} R_{xx}(\tau) e^{-j\omega t} d\tau \qquad [57]$$

y la función de autocorrelación de la señal $R_{xx}(\tau)$

$$R_{xx}(\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{xx}(\omega) e^{j\omega t} d\omega \qquad [58]$$

Dado que la función de autocorrelación es una función par de τ, $S_{xx}(\omega)$ es una función real de ω y se puede obtener fácilmente de la siguiente expresión.

$$S_{xx}(\omega) = 2\int_{0}^{\infty} R_{xx}(\tau) \cos \omega\tau \, d\tau \qquad [59]$$

Las unidades del espectro de densidad de energía $S_{xx}(\omega)$ son voltios al cuadrado partido por hercios. Integrando S_{xx} entre dos frecuencias f_1 y f_2 se obtiene la energía contenida en la señal entre esas dos frecuencias (Godfrey, 1993).

3.5.2 Señales periódicas

Las ecuaciones [55] y [56] se aplican a señales continuas no periódicas. Si la señal es periódica, entonces las transformada de Fourier tiene que ser reemplazada por las series de Fourier. Si el periodo de la señal es *T*, entonces la señal solo puede contener energía a frecuencias armónicas *f=k/T* donde *k* es un entero mayor o igual a la unidad.

La forma más común de serie de Fourier es la trigonométrica, la cual tiene la forma dada por:

$$x(t) = \sum_{k=-\infty}^{\infty} a_k \cos\left(\frac{2\pi k t}{T}\right) + \sum_{k=-\infty}^{\infty} b_k \sin\left(\frac{2\pi k t}{T}\right) \qquad [60]$$

donde

$$a_k = \frac{1}{T}\int_{0}^{T} x(t) \cos\left(\frac{2\pi k t}{T}\right) dt\,, \qquad k = 0, \pm 1, \pm 2, \ldots$$

$$b_k = \frac{1}{T}\int_{0}^{T} x(t) \sin\left(\frac{2\pi k t}{T}\right) dt\,, \qquad k = 0, \pm 1, \pm 2, \ldots \qquad [61]$$

A primera vista no es muy similar a la transformada de Fourier vista en el apartado anterior, pero si la expresamos en la forma exponencial se verán más similares (Balmer, 1991).

$$x(t) = \sum_{k=-\infty}^{\infty} X_k \exp\left(j\frac{2\pi kt}{T}\right) \qquad [62]$$

$$X_k = \frac{1}{2}(a_k - jb_k) = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\exp\left(-j\frac{2\pi kt}{T}\right)dt\,, k = 0, \pm 1, \pm 2, \ldots$$

El rango de integración debe coincidir con el periodo de la señal, pero puede comenzar en cualquier punto dentro de ese periodo. En la ecuación [62] el rango se ha tomado de $-T/2$ a $T/2$ para que sea más parecido a la ecuación [55] de la transformada de Fourier. A veces es posible escoger la posición del origen temporal para facilitar los cálculos, eliminando los términos coseno de una función impar del tiempo o los términos seno de una función par.

Hay una relación similar entre la función de autocorrelación $R_{xx}(\tau)$ de una señal periódica y la de su espectro de potencia $S_{xx}(f = k/T)$, que es la cantidad de energía de la señal a la frecuencia $f = k/T$ Hz. El espectro se representa como una línea espectral y la señal no tiene energía a otras frecuencias. Si la señal x(t) es una tensión, entonces las unidades de S_{xx} son voltios al cuadrado.

Dado que $R_{xx}(\tau)$ es una función par de τ, solo los términos coseno tienen que tenerse en cuenta en la serie de Fourier. Por tanto, en caso de frecuencia nula:

$$S_{xx}(f = 0) = \frac{1}{T}\int_0^T R_{xx}(\tau)d\tau = \frac{2}{T}\int_0^{T/2} R_{xx}(\tau)d\tau \qquad [63]$$

Para una frecuencia $f = k/T$ tenemos:

$$S_{xx}\left(f = \frac{k}{T}\right) = \frac{1}{T}\int_0^T R_{xx}(\tau)\cos\left(\frac{2\pi k\tau}{T}\right)d\tau = \\ = \frac{2}{T}\int_0^{T/2} R_{xx}(\tau)\cos\left(\frac{2\pi k\tau}{T}\right)d\tau\,, k = \pm 1, \pm 2, \ldots \qquad [64]$$

3.6 Introducción a la transformada y series de Fourier para señales discretas

La mayor parte del procesado de señal que se hace hoy en día es digital. Ya que la mayoría de las señales de los sistemas físicos son continuas, el primer paso suele ser muestrear la señal x(t) en intervalos regulares de tiempo Δ para formar la secuencia $x_r, r = 0,1, \ldots, (N-1)$. El bloque de señal muestreado tiene longitud $T_0 = N.\Delta$

3.6.1 Transformada discreta de Fourier

La transformada discreta de Fourier X_k se obtiene de la secuencia x_r como:

$$X_k = \frac{1}{N}\sum_{r=0}^{N-1} x_r \exp\left(-j\frac{2\pi kr}{N}\right) = \frac{1}{N}\sum_{r=0}^{N-1} x_r \left(\cos\frac{2\pi kr}{N} - j\sin\frac{2\pi kr}{N}\right) \quad [65]$$

donde $k = 0,1,\dots,(N-1)$.

Dado que X_k es una secuencia de N números complejos, que proporciona una versión muestreada de la transformada continua de Fourier $X(j\omega)$ a frecuencias armónicamente relacionadas con frecuencias $k/(N*\Delta)$ Hz, donde k es un entero. Las propiedades cíclicas de las funciones seno y coseno hacen que la secuencia X_k sea cíclica con un periodo $1/\Delta$ Hz. A este ciclo se le denomina frecuencia de repetición de la transformada discreta de Fourier. Asimismo, la magnitud de X_k es una función par de la frecuencia $1/2\Delta$ Hz, mientras la fase de X_k es una función impar de esa frecuencia. La frecuencia $1/2\Delta$ Hz es denominada la frecuencia de plegado de Nyquist. La información útil de la TDF[6] está contenida en los armónicos $0,1/N\Delta,2/N\Delta,\dots,1/2\Delta$ Hz. La transformada no puede extraer información de los armónicos superiores a estos. Es importante tener en cuenta que la transformada continua $X(j\omega)$ no tiene este efecto de plegado.

La secuencia de tiempo x_r se puede obtener a partir de la secuencia X_k usando la transformada discreta inversa de Fourier:

$$x_r = \sum_{k=0}^{N-1} X_k \exp\left(j\frac{2\pi kr}{N}\right) \quad [66]$$

Donde $r = 0,1,\dots,(N-1)$. De forma que x_r es una secuencia periódica donde $x_{r+N} = x_r$ incluso si la función original x(t) no lo es. La posición exacta del factor 1/N en la TDF representa un factor de escala en la amplitud.

El espectro de potencia discreto $S_{xx}(k)$ de una señal x(t) se puede encontrar tomando la TDF de la función de autocorrelación $R_{xx}(\tau)$ que ha sido muestreada a intervalos de tiempo regular Δ para producir la secuencia $R_{xx}(r), r = 0,1,\dots,(N-1)$

$$S_{xx}(k) = \frac{1}{N}\sum_{r=0}^{N-1} R_{xx}(r)\exp\left(-j\frac{2\pi kr}{N}\right) \quad [67]$$

Dado que $R_{xx}(r)$ es simétrica con respecto a $r = 0$, $S_{xx}(k)$ es una secuencia de números reales que puede ser expresada como:

$$S_{xx}(k) = \frac{1}{N}\sum_{r=0}^{N-1} R_{xx}(r)\cos\left(\frac{2\pi kr}{N}\right) \quad [68]$$

La transformada inversa correspondiente a esta será:

$$R_{xx}(r) = \sum_{k=0}^{N-1} S_{xx}(k)\exp\left(\frac{2\pi kr}{N}\right) \quad [69]$$

[6] Transformada discreta de Fourier, en inglés se suele usar DFT por Discrete Fourier Transform.

Esta función es periódica con periodo $N\Delta$ aunque la función de correlación original $R_{xx}(\tau)$ no lo fuera.

La ecuación [69] lleva a una forma alternativa de calcular el espectro de potencia discreto. A partir de la ecuación [50] sabemos que

$$R_{xx}(r) = \frac{1}{N}\sum_{i=0}^{N-1} x_i . x_{i+r} \qquad [70]$$

Pero por la ecuación [66]

$$x_{i+r} = \sum_{k=0}^{N-1} X_k \exp\left(j\frac{2\pi k(i+r)}{N}\right) \qquad [71]$$

Entonces a partir de las ecuaciones [70] y [71] sustituyendo se tiene:

$$R_{xx}(r) = \frac{1}{N}\sum_{i=0}^{N-1} x_i . \left[\sum_{k=0}^{N-1} X_k \exp\left(j\frac{2\pi ki}{N}\right)\exp\left(j\frac{2\pi kr}{N}\right)\right] \qquad [72]$$

$$= \sum_{k=0}^{N-1}\left\{\left[\frac{1}{N}\sum_{i=0}^{N-1} x_i . \exp\left(j\frac{2\pi ki}{N}\right) . X_k . \exp\left(j\frac{2\pi kr}{N}\right)\right]\right\} = \sum_{k=0}^{N-1} X_k^* . X_k . \exp\left(j\frac{2\pi kr}{N}\right)$$

Donde X_k^* es el complejo conjugado de X_k. Comparando las ecuaciones [69] y [72] se puede ver que:

$$S_{xx}(k) = X_k^* . X_k \qquad [73]$$

Esta es la aproximación utilizada para calcular el espectro de potencia en la mayoría de analizadores de espectro.

Para una señal no periódica x(t), el espectro de potencia discreto $S_{xx}(k)$ proporciona una estimación del espectro de densidad de energía $S_{xx}(f)$ a la frecuencia cíclica $f = k/N\Delta$, sujeta a un factor de escala. Dado que el intervalo de frecuencia en la TDF es $1/N\Delta$, lo que se está obteniendo es la integral del espectro de densidad de energía continuo sobre el intervalo de frecuencia $1/N\Delta$. Por lo que el espectro discreto debe multiplicarse por $N\Delta = T_0$ para dar una estimación del espectro de densidad de energía. Entonces:

$$\hat{S}_{xx}\left(f = \frac{k}{N.\Delta}\right) = T_0 . S_{xx}(k) \qquad [74]$$

En el caso de tener una señal periódica x(t) el factor de escala T_0 no es necesario. El espectro discreto proporcionará una estimación de la energía espectral continua. Generando un bloque de longitud T_0 que será un múltiplo entero del periodo de la señal. Si no se cumple este criterio, entonces aparecerá un desplazamiento en el espectro obtenido.

3.6.2 El problema del aliasing y del spectral leakage.

El muestreo periódico de una señal continua transforma la señal en una secuencia de valores discretos. Este muestreo es realizado utilizando conversores analógico digitales cuya salida es una secuencia de valores discretizados[7]. Los valores de la secuencia coinciden con los de la señal muestreada en los instantes de muestreo, desconociéndose el valor real de la señal en el resto de instantes del tiempo. La pregunta clave será si al muestrear la señal se está recogiendo toda la información de la misma o se está perdiendo alguna información.

Si consideramos la señal sinusoidal de frecuencia f_0:

$$x(t) = \sin(2\pi f_0 t) \qquad [75]$$

Si ahora muestreamos la señal a una velocidad de f_s muestras por segundo, lo que implica que cada $t_s = 1/f_s$ segundos se tomará una muestra.

$$\begin{aligned} x_0 &= \sin(2\pi f_0 0 t_s) \\ x_1 &= \sin(2\pi f_0 1 t_s) \\ x_2 &= \sin(2\pi f_0 2 t_s) \\ &.... \\ x_n &= \sin(2\pi f_0 n t_s) \end{aligned} \qquad [76]$$

Si consideramos que la muestra n se toma en el instante en que $x_0 = x_n$, ambas muestras estarán separadas entre sí 2π radianes, podremos escribir la ecuación [76] como:

$$x_n = \sin(2\pi f_0 n t_s) = \sin(2\pi f_0 n t_s + 2\pi m) = \sin\left(2\pi(f_0 + \frac{m}{n t_s}) n t_s\right) \qquad [77]$$

Si se hace que *m* sea múltiplo de n entonces *m=kn* y se puede modificar la ecuación [77] como:

$$x_n = \sin\left(2\pi(f_0 + \frac{k}{t_s}) n t_s\right) \qquad [78]$$

Dado que $t_s = 1/f_s$ también se podrá escribir la ecuación anterior como:

$$x_n = \sin(2\pi f_0 n t_s) = \sin(2\pi(f_0 + k f_s) n t_s) \qquad [79]$$

Por tanto, los términos generados en f_0 y $f_0 + k f_s$ son iguales. Resulta por tanto que los valores muestreados para una señal de frecuencia f_0 y otra de frecuencia $f_0 + k f_s$ serán idénticos y por lo tanto indistinguibles unos de otros. Otra consecuencia de este resultado es que una secuencia generada por ordenador representará un número infinito de señales sinusoidales diferentes. Con respecto al espectro de una señal generada como una serie de valores muestreados, éste aparecerá repetido a intervalos $\pm k f_s$, ya que *k* puede tomar valores negativos.

A este fenómeno se le denomina con el nombre de **aliasing o plegamiento**, las frecuencias f_0 y $f_0 + k f_s$ se convierten en *alias* unas de otras. Si el periodo de muestreo es t_s, a la frecuencia

[7] Se dice discreto porque un conversor analógico digital de n bits de precisión solo puede distinguir entre 2^n valores diferentes.

$1/2t_s = f_s/2$ se le denomina frecuencia de Nyquist o frecuencia de plegamiento. Si el espectro del sistema contiene frecuencias por encima de la de Nyquist, estas se sumarán a las frecuencias inferiores distorsionando su contenido de energía por el efecto del aliasing. Podemos pensar también que, a la frecuencia de Nyquist, estos componentes del espectro se pliegan sobre los de frecuencia inferior sumándose a estos. Por tanto, si es posible, se deberá evitar que el espectro de la señal periódica discreta supere a la frecuencia de Nyquist. Reduciendo el periodo de muestreo se aumentará la frecuencia de Nyquist y se reducirá el efecto del aliasing.

Como ejemplo, considérese el caso de una señal de banda limitada de frecuencia de corte B, el espectro de la señal continua tendrá el aspecto de la Figura 8.

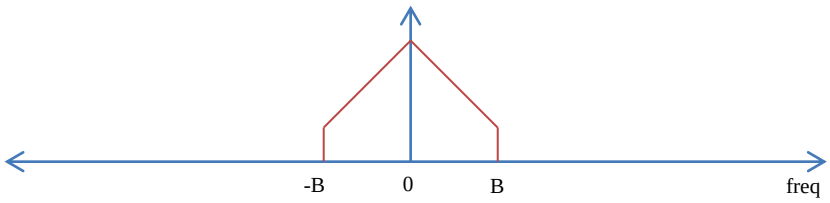

Figura 8: Señal de banda limitada.

El ancho de banda de la señal muestreada, se repite a intervalos $\pm kf_s$ sí tenemos el caso en que la frecuencia de Nyquist es superior a la frecuencia de corte $f_s/2 > B$ tendremos un espectro con la forma de la Figura 9.

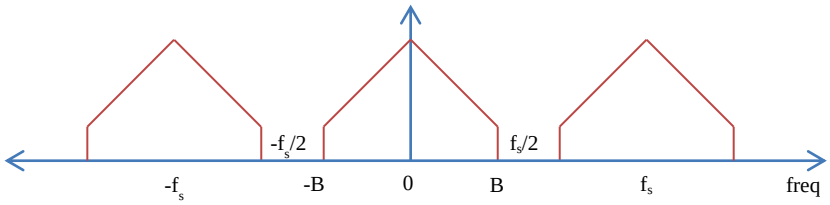

Figura 9: Ancho de banda de la señal muestreada

Si, por el contrario, la frecuencia de Nyquist es inferior a la frecuencia de corte $\frac{f_s}{2} < B$, se presentará el espectro mostrado en la Figura 10.

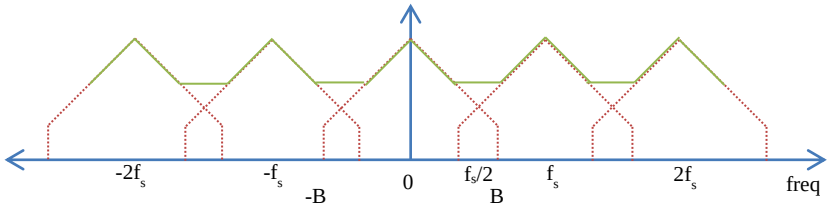

Figura 10: Espectro discreto afectado por el Aliasing

En este caso se puede observar que el espectro resultante es el descrito por la línea verde. Este nuevo espectro es suma de las frecuencias de un espectro con las frecuencias alias del siguiente espectro. El efecto final es una distorsión del espectro resultante, que tendrá un contenido en energía superior al esperado por la señal sin muestrear.

El **spectral leakage** o **fuga espectral** se produce al utilizar la transformada discreta de Fourier, para calcular el espectro de una señal muestreada. Dado que la transformada discreta de Fourier está diseñada para operar sobre un conjunto finito de valores N a una frecuencia de muestreo f_s, con el objeto de producir una transformada discreta de N-puntos cuyas salidas están asociadas con las frecuencias analizadas $f_a(m)$, tendremos que:

$$f_a(m) = \frac{mf_s}{N}, con\ m = 0,1,2,\dots,N-1 \qquad [80]$$

De forma que la transformada producirá respuestas correctas cuando la secuencia de entrada tenga energía exactamente en las frecuencias analizadas. Si la secuencia contiene energía en frecuencias que no sean múltiplos exactos de $\frac{f_s}{N}$ parte de esta energía aparecerá en todos los componentes de la transformada desvirtuando el resultado.

Para evitar este problema en primer lugar se debe garantizar que la frecuencia de muestreo sea al menor al doble de la frecuencia máxima a analizar. La segunda acción es garantizar que se toma una ventana de la muestra que contenga un número entero de periodos de la señal muestreada. Se han desarrollado distintos algoritmos para mejorar este problema, como por ejemplo el de Núñez (2007).

3.7 Señales PRBS.

Las señales PRBS se han utilizado ampliamente en la identificación de sistemas. Existen varios tipos de señales PRBS siendo las señales MLBS (maximum-length binary sequences, secuencias binarias de longitud máxima) son las más conocidas ya que se pueden generar muy fácilmente a través de un registro de desplazamiento realimentado.

3.7.1 Propiedades de las señales PRBS

Las señales PRBS poseen las siguientes propiedades:

(1) Al ser una señal binaria solo tiene dos niveles $\pm A$.

(2) Esta señal es de tipo discreto y síncrono, por lo tanto, solo puede cambiar su valor en los instantes que coincidan con el ciclo de reloj, $t = 0, \Delta t, 2\Delta t, \dots$

(3) Esta señal es de tipo determinístico, es decir, siempre que reiniciemos la secuencia obtendremos los mismos valores. Por lo tanto, cualquier experimento que realicemos con esta señal será repetible. (Esta es la principal diferencia con una señal discreta binaria aleatoria).

(4) Es una señal periódica con periodo $T = N\Delta t$, siendo N siempre un número impar.

(5) Para cualquier periodo la señal está a nivel 1 durante $\frac{1}{2}(N+1)$ ciclos y $\frac{1}{2}(N-1)$ ciclos a nivel 0. De forma que durante un periodo completo de la señal el número de unos excede en una unidad al de ceros.

(6) Las señales PRBS más usadas son MLBS, a partir de este punto solo se considerarán este tipo de señales PRBS.

(7) Para un registro de desplazamiento de *n* bits cada *n*-tupla de dígitos binarios aparece exactamente una vez, excepto la *n*-tupla de todos los bits a cero que no aparece. Si todas las posibles *n*-tuplas, excepto la todos los bits a cero, aparecen en la secuencia antes de volver a repetirse, tendremos una secuencia de longitud máxima y esta será de $N=2^n-1$ bits.

(8) Todas las secuencias binarias de longitud máxima tienen la propiedad de desplazar y sumar. Si una secuencia binaria de longitud máxima se suma, en módulo dos, a la misma secuencia retrasada r dígitos con respecto a la original, la secuencia resultante es de nuevo la misma pero retrasada en q dígitos, donde *r* y *q* son enteros en el rango,

$$1 \leq r, \qquad q \leq N-1$$

Esta propiedad se puede expresar de la forma

$$D^r x \oplus x = (D^r \oplus I)x = D^q x \qquad [81]$$

Esto es, para cada entero q, existe un entero r tal que

$$T^q \oplus T^r = I \qquad [82]$$

Donde *T* es la matriz de transición del registro de desplazamiento e *I* es la matriz unitaria.

Obviamente ya que solo una señal PRBS puede ser generada a partir de una determinada configuración del registro de desplazamiento, la secuencia obtenida a partir de la suma de la secuencia desplazada, solo puede ser ella misma.

(9) Si tenemos una señal PRBS cuyos dos estados son 0 y 1. A continuación, sumamos en módulo dos esta secuencia con ella misma retrasada únicamente un dígito. La secuencia resultante tendrá un estado 1 en cada posición correspondiente con los puntos en que la secuencia cambie de 1 a 0, o viceversa, en la secuencia original y cero en el resto de posiciones.

Si la secuencia original es:

$$x = C1C2C3 \ldots Cr \ldots CN \qquad [83]$$

Donde Cr puede ser un uno o un cero. Si la nueva secuencia se denomina x^1, entonces

$$x^1 = (D \oplus I)x \qquad [84]$$

Por tanto

$$C_r^1 = Cr \oplus Cr - 1 \qquad [85]$$

Donde C_r^1 es un elemento de la nueva secuencia x^1. Esta ecuación implica que C_r^1 y $Cr - 1$ son valores lógicos diferentes. Esto además implica que $Cr - 1$ es el último dígito de un cambio

de estado de lógica 1 (o lógica 0) y que Cr es el primer dígito de un cambio de estado de lógica 0 (o lógica 1).

De la propiedad 8, de desplazamiento y suma, se deduce que esta secuencia resultante, está retrasada por k_s dígitos de la original, donde k_s es el valor de q correspondiente con *r-1* en la ecuación [81]

Esta propiedad se ilustra en la Tabla 1, donde aparece una secuencia de longitud 15 generada en un registro de desplazamiento de 4 bits

Tabla 1: Ejemplo de secuencia desplazada y sumada un bit

Secuencia original x	1 1 1 1 0 0 0 1 0 0 1 1 0 1 0
Secuencia retrasada 1 dígito	0 1 1 1 1 0 0 0 1 0 0 1 1 0 1
$x^1 = (D \oplus I)x$	1 0 0 0 1 0 0 1 1 0 1 0 1 1 1

El valor del entero k_s es función de la configuración del registro de desplazamiento. No existe una fórmula explícita para este valor, pero puede ser determinado a partir de la propiedad 11.

(10) Si a las sucesivas ocurrencias de uno de los estados de un PRBS le denominamos bloque y estos bloques los ponemos en una tabla, se encontrará que hay 2^{n-1} bloques en la secuencia. De estos la mitad son de un dígito de longitud, la cuarta parte de dos dígitos, una octava parte de tres dígitos, etc. El número de tipos de bloques distintos es (n+1)/2. Existen un mismo número de bloques de cada estado, excepto que hay un bloque de n unos, no hay un bloque de *n* ceros, pero sí que hay un bloque de *n*-1 ceros, pero no uno de *n*-1 unos. A partir de la secuencia de la Tabla 1, se encuentra que se tienen:

- 4 bloques de 1 dígito
- 2 bloques de 2 dígitos
- 1 bloque de tres estados a 0
- 1 bloque de cuatro estados a 1

El número total de bloques es $8 = 2^{4-1}$.

(11) Para cualquier PRBS $(k_s - 1)$ es igual al número de dígitos que hay desde el principio de un bloque de *n*-1 estados lógicos cero hasta, pero sin incluirlo, el principio de un bloque de *n* estados lógicos a uno. Esto es, si la secuencia x comienza con *n* estados lógicos a uno, la secuencia definida como x^1 comienza con *n*-1 estados lógicos a cero.

$$x^1 = (D \oplus I)x = D^{k_s - 1}x \qquad [86]$$

Esto aparece como una consecuencia directa de la propiedad 8, dado que la secuencia x^1 debe tener su bloque de *n*-1 estados a valor cero correspondientes a los últimos *n*-1 estados del bloque de *n* estados a valor uno de la secuencia original x. Por la propiedad 3 se deduce que x^1 debe ser una versión desplazada de la PRBS original.

En la Tabla 1 se puede ver que el resultado de la secuencia x^1 esta desplazada 12 dígitos con respecto a la secuencia original x, mientras que el número de dígitos desde e incluyendo el principio de bloque de tres dígitos a cero, hasta el bloque de cuatro dígitos a uno, pero sin incluir el principio de este es de 11 dígitos.

(12) A partir de este punto se va a asumir que los dos estados de la señal PRBS previamente definidos como 1 y 0 van a corresponder con las tensiones $+A$ y $-A$ respectivamente. Este requisito es necesario desde el momento en que estas señales se van a utilizar como señales de pseudoruido blanco para ser introducidas en el sistema a identificar. El periodo de tiempo Δt será el intervalo transcurrido entre cualquier transición de la señal y será el periodo de la señal de reloj.

El número de transiciones de la señal de un estado a otro será de 2^{n-1}

(13) Si los dos niveles de la señal PRBS son $\pm A$ la función de autocorrelación cuya expresión aparece en la ecuación [47], de la señal PRBS es la de la Figura 11.

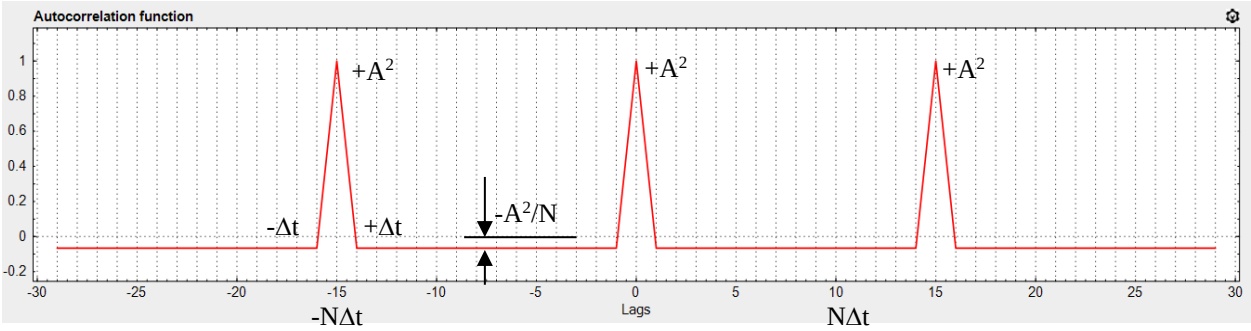

Figura 11: función de autocorrelación de la señal PRBS

De forma que la señal PRBS, como se indicó en la propiedad 4, tiene un periodo $T = N\Delta t$ donde Δt es el periodo de la señal de reloj y es por tanto el intervalo básico de tiempo entre bits. Dado que $N = 2^n - 1$ es un número impar, habrá una pequeña componente de continua en la media de la secuencia x(t) en cada ciclo, aunque en la mayoría de los casos, sobre todo cuando N es grande, puede ser despreciada.

Dado que, aunque las PRBS son señales continuas, se pueden considerar como una secuencia, ya que sus valores solo cambian en los instantes marcados por la señal de reloj. La ecuación [50] nos ofrece una forma sencilla de calcular la secuencia de autocorrelación.

$$\hat{R}_{xx}(k) = \frac{1}{N}\sum_{r=0}^{N-1} x_{r-k}x_r$$

Donde cada valor es un entero que corresponde con los desplazamientos de la secuencia original. Esta ecuación puede ser evaluada usando la tabla de multiplicación mostrada en la Tabla 2.

Tabla 2: Tabla de multiplicación de la autocorrelación de una PRBS

Termino 1	Término 2	Resultado
+a	+a	$+a^2$
+a	-a	$-a^2$
-a	+a	$-a^2$
-a	-a	$+a^2$

De forma que el resultado es una secuencia y se puede escribir como:

$$R_{xx}(k) = \frac{1}{N}\{Número\ de\ digitos\ coincidentes - Número\ de\ dígitos\ distintos\}$$

Donde coincidentes quiere decir que los dígitos tienen el mismo signo y distintos que tienen signo diferente. Sin embargo, dado que la secuencia es de tipo PRBS y satisface la propiedad 5, la ecuación anterior se puede expresar como:

$$R_{xx}(k) = \frac{1}{N}\{N - 2 * Número\ de\ dígitos\ distintos\} \quad [87]$$

De las propiedades previas, se deduce que la función de autocorrelación debe contener (N+1)/2 dígitos distintos y (N-1)/2 dígitos coincidentes.

La ecuación [87] se puede escribir como:

$$R_{xx}(k) = \frac{A^2}{N}\{N - 2(N+1)/2\} = -\frac{A^2}{N}\ \ para\ k \neq 0$$
$$R_{xx}(k) = \frac{A^2 N}{N} = A^2\ \ para\ \ k = 0 \quad [88]$$

Esto es, la función de autocorrelación siempre toma el valor A^2 para el término inicial de la señal PRBS y $-A^2/N$ para el resto de términos. Cuando N sea grande, la función de autocorrelación se irá aproximando a la ideal, que es A^2 para los términos iniciales de cada periodo y 0 para el resto.

Se ha obtenido el valor de la función de autocorrelación en los instantes de muestreo, pero esta es una señal continua, de forma que parece razonable que el valor de la función de autocorrelación cambie de forma lineal entre dos instantes consecutivos del periodo de muestreo.

Consideremos el valor de τ de la ecuación [47] que representa a la variable continua.

$$k\Delta t < \tau < (k+1)\Delta t, k = 0,1,2,\dots,N$$

Se va a tener en cuenta el primer rango $0 < \tau < \Delta t$.

Dado que hay 2^{n-1} ciclos o pasos por cero, entonces en cada intervalo Δt se restará del valor de $R_{xx}(k)$ un bloque de superficie $A^2\tau/N\Delta t$ y a la vez se sustituirá por un bloque $-A^2\tau/N\Delta t$

La función de autocorrelación para τ en este rango es entonces:

$$R_{xx}(\tau) = R_{xx}(0) - \frac{N+1}{2}\frac{A^2\tau}{N\Delta t} - \frac{N+1}{2}\frac{A^2\tau}{N\Delta t} = A^2 - \frac{(N+1)A^2\tau}{N\Delta t} \quad [89]$$

En conclusión

$$R_{xx}(\tau) = A^2\left[1 - \frac{\tau\,(N+1)}{N\Delta t}\right]\ \ para\ 0 < \tau < \Delta t \quad [90]$$

Si ahora se considera el rango k$\Delta t < \tau < (k+1)\Delta t$ con $k \neq 0$. Se puede ver que el área de los bloques que se quitan es igual que la de los bloques que se añaden, entonces para $\tau \geq \Delta t$

$$R_{xx}(\tau) = -A^2/N \quad [91]$$

El mismo argumento se puede aplicar para el rango $(N-1)\Delta t < \tau < N\Delta t$, de forma que la ecuación para una función de autocorrelación para una señal PRBS de longitud máxima es:

$$R_{xx}(\tau) = A^2\left[1 - \frac{\tau}{\Delta t}.\frac{(N+1)}{N}\right] \quad para\ 0 \le \tau < \Delta t \quad [92]$$
$$R_{xx}(\tau) = -\frac{A^2}{N} \quad para\ \tau \ge \Delta t$$

Que es la ecuación generatriz de la función de autocorrelación mostrada en la Figura 11.

Si la función de autocorrelación de cualquier secuencia de periodo genérico *B* no es mayor que 1/(*B*-1) para los desplazamientos distintos de cero, a la **secuencia** se la denomina **perfecta**. Como ha quedado establecido en la ecuación [92] todas las secuencias PRBS son perfectas.

(14) Como la señal PRBS es periódica su **espectro de potencia** es una espectro de líneas cuya envolvente es de la forma {(*sin x)/x*}, encontrándose el primer valor nulo a la frecuencia del impulso de reloj del registro de desplazamiento.

A partir del teorema Wiener-Khitchin (Kschischang, 2017) se sabe que el espectro de potencia es la transformada de Fourier de la función de autocorrelación como vimos en el apartado 3.5.1 ecuación [57], que se reproduce a continuación.

$$S_{xx}(\omega) = \int_{-\infty}^{\infty} R_{xx}(\tau)e^{-j\omega t}d\tau$$

Como ya se indicó previamente, $R_{xx}(\tau)$ para una señal PRBS es una función par de τ y en el caso de las señales PRBS es cíclica con periodo $T = N\Delta t$. Recordemos que la función de autocorrelación está descrita por una señal continua de valor $-A^2/N$ para la mayor parte del tiempo y un triángulo de amplitud $A^2(N+1)/N$ en el instante $\tau = 0$ y descendiendo a cero en $\tau = \Delta t$. Se puede rescribir la ecuación [57] como:

$$S_{xx}(\omega) = 2\int_0^T R_{xx}(\tau)\cos\omega\tau\, d\tau \quad [93]$$
$$= 2\int_0^{\Delta t} A^2\left[1 - \frac{\tau}{\Delta t}.\frac{(N+1)}{N}\right]\cos\omega\tau\, d\tau + 2\int_{\Delta t}^{N\Delta t} -\frac{A^2}{N}\cos\omega\tau\, d\tau$$

Una solución para estas integrales es:

$$S_{xx}(\omega) = \left\{-\frac{2A^2}{N\omega}\sin(\omega\Delta t) - \frac{2A^2(N+1)}{N\omega^2\Delta t}[\cos(\omega\Delta t) - 1] - \frac{2A^2}{N\omega}\sin(\omega N\Delta t)\right. \quad [94]$$
$$\left. + \frac{2A^2}{N\omega}\sin(\omega\Delta t)\right\} = \left\{\frac{2A^2(N+1)}{N\omega^2\Delta t}[2\sin^2(\omega\Delta t/2)] - \frac{2A^2}{N\omega}\sin(\omega N\Delta t)\right\}$$

La frecuencia fundamental o armónico principal de la señal es $2\pi/N\Delta t$ rad/sec que coincide con la frecuencia del pulso de reloj del registro de desplazamiento. Como la secuencia

cambia de nivel en intervalos discretos de tiempo, las frecuencias en el espectro de densidad de energía también tienen que ser discretas. El espectro es por tanto espectro de líneas con separaciones armónicas de $2\pi/N\Delta t$ rad/sec. El segundo término en la ecuación [94] es siempre nulo y la ecuación puede ser reescrita como:

$$S_{xx}(\omega) = \frac{A^2(N+1)\Delta t}{N}\sum_{r=1}^{N}\left[\frac{sin(r\omega\Delta t/2)}{(r\omega\Delta t/2)}\right]^2 = \frac{A^2(N+1)\Delta t}{N}\sum_{r=1}^{N}\left[\frac{sin(r\pi/N)}{(r\pi/N)}\right]^2 \quad [95]$$

Nótese que $\frac{sin(r\pi/N)}{(r\pi/N)}$ es aproximadamente 1 cuando $r \ll N$. El espectro de densidad de energía toma pues un valor de $\frac{A^2(N+1)\Delta t}{N}$ a bajas frecuencias. El punto que marca el ancho de banda es aquel en que la ganancia del espectro cae 3 decibelios, cuya frecuencia viene dada por

$$\left[\frac{sin(r\pi/N)}{(r\pi/N)}\right]^2 = 0.7071 \; o \; cuando \; r \cong N/3 \quad [96]$$

El ancho de banda efectivo cubierto por una señal PRBS de máxima longitud es:

$$f = \frac{1}{N\Delta t} \;\; hasta \;\; \frac{1}{3\Delta t} \; c/s \quad [97]$$

Donde Δt es el periodo de la señal de reloj. Por lo tanto, se puede decir que el rango de frecuencias de la señal PRBS queda seleccionado al escoger la frecuencia de reloj base de la señal.

La Figura 12 ilustra la forma que toma el espectro de densidad de energía de una PRBS. La ganancia de cada punto se calcula según la ecuación [95], el ancho de banda por la ecuación [96] marcado en la gráfica en color verde y el ancho de banda efectivo cubierto por la señal PRBS, viene determinado por la ecuación [97].

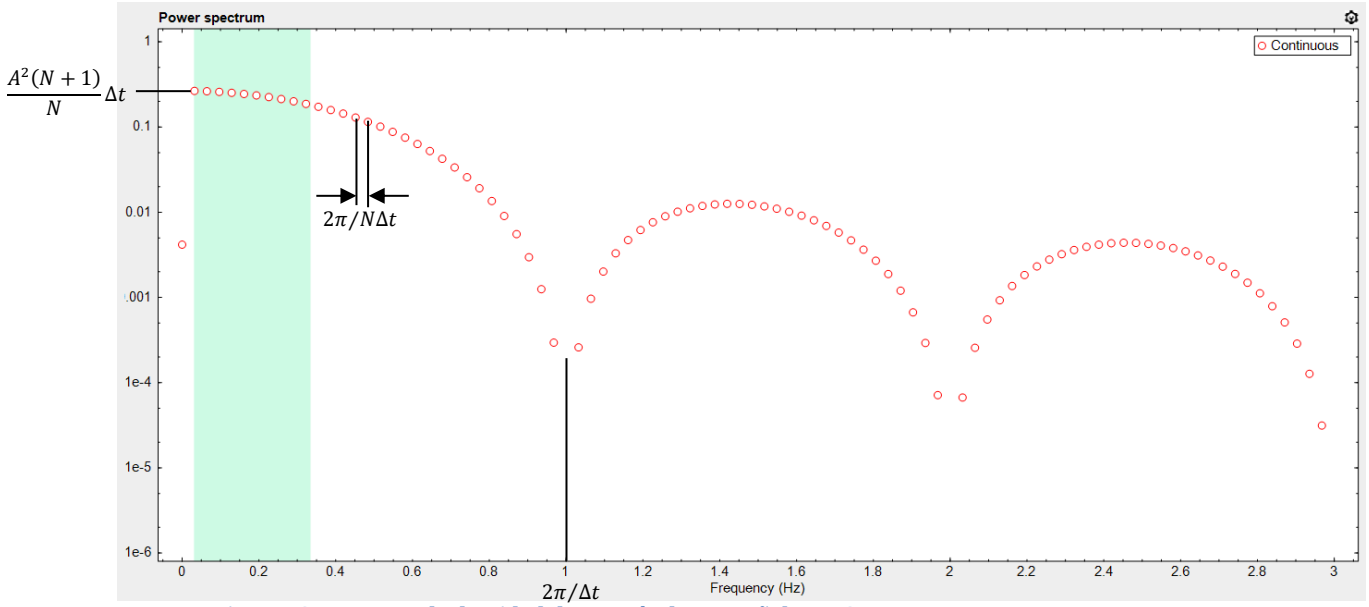

Figura 12: Espectro de densidad de energía de una señal PRBS

De forma que para $S_{xx}(0)$ tendremos:

$$S_{xx}(0) = \frac{A^2}{N^2} \quad [98]$$

Y para $S_{xx}\left(f = \frac{r}{T}\right)$ será:

$$S_{xx}\left(f = \frac{r}{T}\right) = \frac{A^2(N+1)}{N^2}\frac{\sin^2(r\pi/N)}{(r\pi/N)^2}, r = \pm 1, \pm 2, \ldots \quad [99]$$

(15) Cuando una señal PRBS de máxima longitud se utiliza como entrada de un sistema para obtener su correlación cruzada. Por ejemplo, para obtener su respuesta al impulso *g*(*t*), La integral de convolución.

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} g(s).R_{xx}(\tau - s)ds \quad [100]$$

Donde $R_{xy}(\tau)$ es la función de correlación cruzada entre la entrada y la salida y $R_{xx}(\tau - s)$ es la función de autocorrelación de la entrada y *g*(*t*) es la respuesta del sistema a la entrada impulso. Dado que la entrada es una señal PRBS esta ecuación se puede reescribir como:

$$R_{xy}(\tau) = \int_0^T g(s)R_{xx}(\tau - s)ds = \frac{A^2(N+1)}{N}\int_0^T g(s)R_{xx}(\tau - s)ds - \frac{A^2}{N}\int_0^T g(s)ds \quad [101]$$

Donde $R_{xx}(\tau)$ es un triángulo de altura unitaria y base $2\Delta t$ de forma que:

$$R_{xx}(\tau) = \frac{\tau + \Delta t}{\Delta t} \quad para\ -\Delta t \le \tau \le 0$$
$$R_{xx}(\tau) = \frac{\Delta t - \tau}{\Delta t} \quad para\ 0 \le \tau \le \Delta t \quad [102]$$

Expandiendo la función de la respuesta al impulso *g*(*s*) en torno a $s = \tau$ se tiene:

$$g(s) = g(\tau) + (s - \tau)g'(\tau) + \frac{(s-\tau)^2}{2!}g''(\tau) + \cdots \quad [103]$$

Entonces en una primera aproximación la integral de convolución se convierte en:

$$R_{xy}(\tau) = A_0\frac{A^2(N+1)}{N}g(\tau) - \frac{A^2}{N}\int_0^T g(s)ds \quad [104]$$

Donde

$$A_0 = \int_0^T R_{xx}(\tau - s)ds = -\int_\tau^{\tau - T} R_{xx}(X)dX \quad [105]$$

Donde

$$R_{xx}(X) = \frac{X + \Delta t}{\Delta t} \quad para \ -\Delta t \leq X \leq 0$$
$$R_{xx}(X) = \frac{\Delta t - X}{\Delta t} \quad para\ 0 \leq X \leq \Delta t \qquad [106]$$

Para un sistema físico estable, la respuesta al impulso debe decaer hasta cero en un tiempo interior a T, de forma que esta es la única parte de $R_{xx}(\tau)$ para $\tau < T$ que debe ser considerada.

Quedan entonces dos casos para considerar. En primer lugar, cuando $\tau \geq \Delta t$ y en segundo lugar cuando $\tau = 0$. En el primer caso $\tau \geq \Delta t$

$$A_0 = -\left\{\int_{\Delta t}^{0} \frac{\Delta t - X}{\Delta t} dX + \int_{0}^{-\Delta t} \frac{X + \Delta t}{\Delta t} dX\right\} = \Delta t \qquad [107]$$

Entonces para $\tau \geq \Delta t$ se tiene que:

$$R_{xy}(\tau) = \frac{A^2(N+1)\Delta t}{N} g(\tau) - \frac{A^2}{N}\int_0^T g(s)ds \qquad [108]$$

En el segundo caso cuando $\tau = 0$, se asume que la respuesta al impulso tiene que haber decaído hasta cero en un tiempo inferior al del ciclo T, entonces el segundo triángulo no necesita tenerse en cuenta. En este caso, por tanto:

$$A_0 = -\left\{\int_{0}^{-\Delta t} \frac{X + \Delta t}{\Delta t} dX\right\} = \frac{\Delta t}{2} \qquad [109]$$

Entonces para $\tau = 0$

$$R_{xy}(\tau) = \frac{A^2(N+1)\Delta t}{2N} g(\tau) - \frac{A^2}{N}\int_0^T g(s)ds \qquad [110]$$

El segundo término de la parte derecha

$$\frac{A^2}{N}\int_0^T g(s)ds = constante$$

Previendo que el periodo T es mayor que el tiempo de asentamiento del sistema bajo consideración y que $g(x)$ no cambia de valor $g(\tau)$ en el intervalo $(\tau - \Delta t) \leq x \leq (\tau + \Delta t)$.

De forma similar para cualquier valor de τ en el intervalo $0 < \tau < \Delta t$

$$A_0 = -\left\{\int_{\Delta t}^{0} \frac{\Delta t - X}{\Delta t} dX + \int_{0}^{-\Delta t} \frac{X + \Delta t}{\Delta t} dX\right\} = \frac{\Delta t}{2} + \tau\left\{1 - \frac{\tau}{2\Delta t}\right\} \qquad [111]$$

Entonces en este rango

$$R_{xy}(\tau) = \frac{A^2(N+1)}{N}\left[\frac{\Delta t}{2} + \tau\left\{1 - \frac{\tau}{2\Delta t}\right\}\right] g(\tau) - \frac{A^2}{N}\int_0^T g(s)ds \qquad [112]$$

(16) La correlación cruzada de una señal PRBS de longitud máxima cuyos estados son +*A* y -*A*, denominada $x(t)$ y una secuencia similar cuyos estados son +*A* y 0 respectivamente, denominada $\bar{x}(t)$ tiene la forma de la Figura 13. La relación entre las dos expresiones puede ser expresada como:

$$\bar{x}(t) = \frac{1}{2}\{x(t) + a\} \qquad [113]$$

Entonces la función de correlación cruzada $R_{x\bar{x}}(\tau)$ será

$$R_{x\bar{x}}(\tau) = \frac{1}{2}R_{xx} + \frac{A}{2T}\int_0^T x(t)dt = \frac{1}{2}\left\{R_{xx} + \frac{A^2}{N}\right\} \qquad [114]$$

Dado que $\int_0^T x(t)dt = a\Delta t$ para cualquier secuencia binaria de longitud máxima.

Figura 13: Correlación cruzada de la función $R_{x\bar{x}}(\tau)$

(17) De las propiedades anteriores se hace evidente que, si las técnicas convencionales de correlación cruzada se tienen que utilizar con una señal de entrada de tipo PRBS de longitud máxima, el hecho de que la secuencia sea binaria, simplifica la complejidad de la operación de producto necesaria para calcular la función de correlación cruzada.

Dado que hay solo dos posibles valores $\pm A$, el producto de la salida del sistema por la secuencia retrasada se convierte en una operación de control de puerta, donde o la salida o su inversa son seleccionadas, seguidas por un estado de amplificación para restaurar el valor de la constante *A*.

(18) Si una señal PRBS de longitud máxima de longitud $N = 2^n - 1$ se muestrea cada *f* bits, donde *f* y *N* son coprimos[8], la secuencia resultante será también una señal PRBS si f está en

[8] Son aquellos pares de números que no tienen divisores en común, salvo el 1

el rango $1 \leq f \leq (N-1)$. La secuencia resultante será o la misma que la original, o estará invertida en el tiempo, dependiendo del valor de f.

Por otra parte, si f y N no son coprimos, la secuencia resultante no será PRBS de longitud máxima.

(19) La función de autocovarianza de una señal PRBS es periódica y es similar a la del ruido blanco (Rivera, 2007).

3.7.2 Generación de las señales PRBS

Para las señales PRBS de tipo MLBS la longitud máxima es $N=2^n-1$ bits. Por lo tanto, el periodo T de esta señal será:

$$T = \frac{N}{F} = \frac{2^n - 1}{F} \qquad [115]$$

Siendo F la frecuencia de reloj de la señal.

Como ya se ha dicho, la forma más común de generar este tipo de señales es usando un registro de desplazamiento de n estados. Al primero de los bits del registro de estado se le realimentará una señal proveniente de una operación XOR, o suma módulo 2, entre dos de los bits del registro de desplazamiento. La lógica de la función XOR es:

$$1 \oplus 1 = 0 \oplus 0 = 0 \;\; y \;\; 1 \oplus 0 = 0 \oplus 1 = 1$$

En la práctica los valores 0 y 1 solo tienen sentido como operadores lógicos y deben ser sustituidos por dos valores de tensión. Por ejemplo, el 1 lógico equivale a una tensión positiva de valor $+A$ y el 0 lógico a una tensión negativa de valor $-A$.

Vemos un ejemplo de generador de señal PRBS que usa un registro de desplazamiento de 8 bits en la Figura 14, que puede generar secuencias de hasta 255 símbolos.

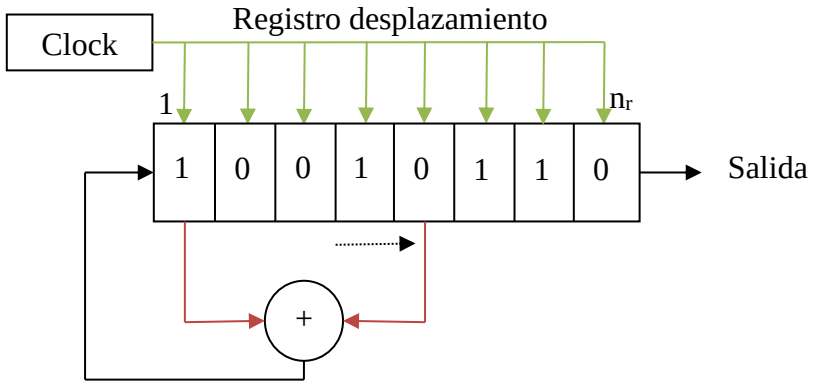

Figura 14: Generador de señal PRBS con XOR

Es fácil darse cuenta de que independientemente de la longitud del registro, si el estado inicial del registro de desplazamiento es con todos sus bits a cero, da igual a que bits del registro de desplazamiento conectemos las entradas de la XOR, que el resultado insertado en el registro de

desplazamiento será siempre nulo. Por ese motivo, el estado 000..0 nunca puede aparecer dentro del registro y por tanto, no se pueden conseguir secuencias de longitud 2^n.

Si tomamos por ejemplo un registro de desplazamiento de 4 bits, que tenga como estados de realimentación el 1 y el 4 y que inicialmente tenga el estado 1111, el resultado de los distintos desplazamientos es que aparece en la Tabla 3. La secuencia inicial se repite después de los 2^n-1 estados. Cada uno de los posibles estados aparece en el registro de desplazamiento exactamente una vez. Esto se cumple para todas las secuencias MLSB.

Si las realimentaciones desde los estados *k, l, m,…n* producen una secuencia MLSB, entonces las realimentaciones desde los estados *n-k, n-l, n-m,…, n* producirán una segunda secuencia que será igual que la primera pero invertida. Esto se puede apreciar en la Tabla 4, donde se presenta la secuencia generada por la realimentación de los estados 3 y 4, que es igual que la anterior pero invertida. Considere el bit 4 como bit de salida del registro.

Tabla 3: Secuencia PRBS de máxima longitud para n=4 y realimentaciones en los estados 1 y 4

Impulso de reloj	**Bit del registro de desplazamiento**			
	1	**2**	**3**	**4**
1	1	1	1	1
2	0	1	1	1
3	1	0	1	1
4	0	1	0	1
5	1	0	1	0
6	1	1	0	1
7	0	1	1	0
8	0	0	1	1
9	1	0	0	1
10	0	1	0	0
11	0	0	1	0
12	0	0	0	1
13	1	0	0	0
14	1	1	0	0
15	1	1	1	0
16	**1**	**1**	**1**	**1**

Tabla 4: Secuencia PRBS de máxima longitud para n=4 y realimentaciones en los estados 3 y 4

Impulso de reloj	**Bit del registro de desplazamiento**			
	1	**2**	**3**	**4**
1	1	1	1	1
2	0	1	1	1
3	0	0	1	1
4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
7	0	0	1	0
8	1	0	0	1
9	1	1	0	0
10	0	1	1	0
11	1	0	1	1
12	0	1	0	1
13	1	0	1	0
14	1	1	0	1
15	1	1	1	0
16	**1**	**1**	**1**	**1**

Muy pocas de las posibles conexiones de realimentación llevan a una secuencia de máxima longitud. Si definimos el polinomio característico del registro de desplazamiento D como:

$$f(D) = \sum_{k=0}^{n} a_k D^k \ (modulo\ 2) \qquad [116]$$

Donde a_k es cero si ese bit del registro no se suma y uno si se incluye en la realimentación del registro. El polinomio resultante debe ser irreducible y primitivo para conseguir una secuencia de longitud máxima. Un polinomio de grado n es irreducible si no es divisible por cualquier polinomio de grado inferior a n, pero superior a cero. Un polinomio es primitivo si y solo si no es divisible por $x^r \oplus 1$, donde $\oplus$ es la operación XOR, para cualquier $r < 2^n-1$ (Godfrey, 1993; Peterson & Weldon, 1972).

Si el polinomio $c_n x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ es primitivo módulo 2, entonces la ecuación característica correspondiente en retardos D introducida por el registro de desplazamiento (Briggs & Godfrey, 1966).

$$c_n D^n + c_{n-1}D^{n-1} + \cdots + c_1 D + c_0 = 0 \ \ módulo\ 2 \qquad [117]$$

Las conexiones de realimentación al primer estado del registro de desplazamiento vienen dadas por:

$$c_0 X = -c_1 DX - \cdots - c_{n-1}D^{n-1}X - c_n D^n X \ \ módulo\ 2 \qquad [118]$$

Donde X es la secuencia de entrada al registro de desplazamiento, DX es la secuencia en la salida del primer estado del registro y siguiendo así hasta $D^n X$ que es la secuencia en la salida del

último estado del registro de desplazamiento de n bits. Dado que la resta en módulo 2 es la misma operación que la suma[9] en módulo 2, la configuración de la realimentación viene dada por:

$$c_0X = c_1DX + \cdots + c_{n-1}D^{n-1}X + c_nD^nX \ \ módulo\ 2 \qquad [119]$$

Para secuencias MLSB $\boldsymbol{c_0 = c_n = 1}$ y el resto de coeficientes pueden ser 1 o 0. Por ejemplo la ecuación característica del registro de desplazamiento usado para generar la Tabla 3 es:

$$D^4 + D + 1 = 0, \quad módulo\ 2$$

Que corresponde con el polinomio primitivo $\boldsymbol{x^4 + x + 1 = 0}$ en módulo 2. La Tabla 5 tomada de Godfrey (1993, p. 27), muestra los estados a realimentar desde N=2 hasta 100. Además de las opciones mostradas en la tabla, existen otras que generan una secuencia de longitud máxima.

Para realizar funciones XOR de más de 2 bits, se puede utilizar la propiedad asociativa y dividir la función XOR de n bits en asociaciones de XOR de 2 bits. Otra opción es acudir a la definición de la función XOR que nos dice que salida toma valor 1 si en la entrada hay un número impar de bits a 1. Aplicando la definición, bastará son sumar el número de bits a uno y si el resultado es impar la salida de la XOR es 1 o 0 en caso contrario.

3.7.3 Otros tipos de señales PRBS

Las señales de las que hemos hablado hasta ahora son las de longitud máxima o MLBS, que son las más utilizadas en la identificación de sistemas. Sin embargo, estas secuencias de longitud máxima dejan muchos huecos en medio. Por ejemplo, para *n*=10 tenemos una secuencia de 1023 bits y para *n*=11 tenemos otra secuencia de 2047 bits, ¿Qué pasaría si precisásemos una secuencia de longitud intermedia? Por ejemplo 1500, pues deberíamos descubrir si existe algún otro tipo de secuencia que permita generar una secuencia más próxima a esa longitud.

Otros tipos de señales PRBS son:

1. Señales QRB (Quadratic Residue Binary Signals), para estas señales N=4k-1 teniendo que ser N primo, por tanto genera señales de longitudes: 3, 7, 11, 19, 23, 31, 43, 59, 67, 71, 79, 83, 107, 127,… (Charters, 2009; Egidi & Manzini, 2013).
2. Señales HAB (Hall Binary Signals), también conocidas como sHall's sextic residues para las cuales $N=4k^2+27$ y además primo como en el caso anterior. Con este tipo de señales se pueden generar secuencias de longitud: 31, 43, 127, 223, 283, 811, 1051, 1471, 1627,…(Lee et al., 2015).
3. Señales TPB (Twin Prime Binary Signals), para estas señales H=k(k+2) con k y k+2 primos, en este caso las longitudes de secuencias son: 15, 35, 143, 323, 899, 1763, 3599, 5183, …(Dai et al., 2009).
4. Señales QRT (Quadratic Residue Ternary Signals), son similares a las QRB, pero ahora N=4k±1 donde N sea primo, en esta caso las longitudes son: 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,…(Tan & Godfrey, 2019)

Estas señales pueden ser útiles para rellenar los huecos que aparecen entre las señales MLSB. Sobre todo, las últimas, las de tipo QRT ya que proporcionan más longitudes de secuencias distintas que las anteriores.

[9] Una operación XOR en ambos casos.

Tabla 5: Estados que se realimentan a la entrada del registro con XOR (fuente (Godfrey, 1993, p. 27)**)**

n	Estados realimentados	n	Estados realimentados	n	Estados realimentados
2	1,2	35	2,35	68	1,5,7,68
3	2,3	36	1,2,4,5,6,36	69	2,5,6,69
4	3,4	37	1,2,3,4,5,37	70	1,3,5,70
5	3,5	38	1,5,6,38	71	1,3,5,71
6	5,6	39	4,39	72	1,2,3,4,6,72
7	4,7	40	3,4,5,40	73	2,3,4,73
8	4,5,6,8	41	3,41	74	3,4,7,74
9	5,9	42	1,2,3,4,5,42	75	1,3,6,75
10	7,10	43	3,4,6,43	76	2,4,5,76
11	9,11	44	2,5,6,44	77	2,5,6,77
12	6,8,11,12	45	1,3,4,45	78	1,2,7,78
13	9,10,12,13	46	1,2,3,4,8,46	79	2,3,4,79
14	4,8,13,14	47	5,47	80	1,2,3,5,7,80
15	14,15	48	1,2,4,5,7,48	81	4,81
16	4,13,15,16	49	4,5,6,49	82	1,4,6,7,8,82
17	14,17	50	2,3,4,50	83	2,4,7,83
18	11,18	51	1,3,6,51	84	1,3,5,7,8,84
19	14,17,18,19	52	3,52	85	1,2,8,85
20	17,20	53	1,2,6,53	86	2,5,6,86
21	19,21	54	2,3,4,5,6,54	87	1,5,7,87
22	21,22	55	1,2,6,55	88	1,3,4,5,8,88
23	18,23	56	2,4,7,56	89	3,5,6,89
24	17,22,23,24	57	2,3,5,57	90	2,3,5,90
25	22,25	58	1,5,6,58	91	2,3,5,6,7,91
26	20,24,25,26	59	1,3,4,5,6,59	92	2,5,6,92
27	22,25,26,27	60	1,60	93	2,93
28	25,28	61	1,2,5,61	94	1,5,6,94
29	27,29	62	3,5,6,62	95	1,2,4,5,6,95
30	7,28,29,30	63	1,63	96	2,3,4,6,7,96
31	28,31	64	1,3,4,64	97	6,97
32	10,30,31,32	65	1,3,4,65	98	1,2,3,4,7,98
33	20,33	66	2,3,5,6,8,66	99	4,5,7,99
34	7,32,33,34	67	1,2,5,67	100	2,7,8,100

Hasta ahora nos hemos centrado en señales de tipo binario. En la sección 2.3 del libro de Tan (2019) se pueden encontrar señales de tipo multinivel pseudoaleatorias que también se podrían usar para la identificación de sistemas. Sin embargo, en este TFM se preferirá el uso de señales multiseno como señales multinivel.

3.7.4 Consideraciones sobre el diseño de señales PRBS

A continuación se van a hacer una serie de consideraciones prácticas a tener en cuenta en el diseño de señales PRBS utilizadas para la identificación de sistemas y extraídas de Godfrey (1993, p. 34)

- La amplitud de la señal PRBS, en un caso práctico, será impuesta por limitaciones de seguridad, (no saturar ninguna parte del sistema a identificar o no sacar al sistema fuera de sus parámetros de funcionamiento estable) o linealidad, (mantener las variables del proceso dentro de su rango lineal).
- El intervalo del periodo de reloj Δt debe ser escogido cuidadosamente: un valor demasiado pequeño llevará a una variabilidad demasiado pequeña en la respuesta, mientras que un valor demasiado grande llevará a que la convolución inversa de la ecuación [117] no sea válida.
- El periodo de la señal $N\Delta t$ debe ser superior al tiempo de asentamiento del proceso, pero no mucho más grande. Será mejor calcular la media sobre varios periodos de señal cortos que sobre uno grande.
- Si hay una desviación de la forma $a_1 t$ o $a_2 t^2$ en la salida y(t), la señal PRBS debería estar basada en las secuencias MLBS citadas por Barker (Barker, 1967), que permiten minimizar el efecto del término $a_2 t^2$ y la correlación debería comenzar en los puntos indicados para eliminar el efecto de $a_1 t$.

3.8 Señales multiseno.

Como ya se indicó en el apartado 3.1.1 en la página 19, una señal multiseno es una suma de sinusoides. Como en el caso de la señal PRBS se trata de una señal determinística periódica que puede expresarse tal y como figura en la ecuación [120]

$$u(t) = \sum_{i=1}^{F} A_i \cos(2\pi f_i t + \phi_i)\,, 0 \le t \le T \qquad [120]$$

En esta expresión:

- T es el periodo de la señal. Nótese que, si seleccionan N puntos o muestras de la señal, entonces el periodo de muestreo T_s es

$$T_s = \frac{T}{N} \;\rightarrow\; T = NT_s \qquad [121]$$

- F es el número de señales sinusoidales, es decir de armónicos de la señal. Nótese que para reconstruir bien la señal y evitar el aliasing se debe cumplir que:

$$F < \frac{N}{2} \qquad [122]$$

  Por lo que, si el valor de F está fijado de antemano, entonces N tiene el siguiente valor mínimo

$$N > 2F \qquad [123]$$

- A_i es la amplitud relativa de cada sinusoide.
- ϕ_i es la fase de cada sinusoide.
- f_i es la frecuencia de la sinusoide (armónico i), donde

$$f_i = \frac{i}{T}, \qquad i = 1, 2, 3, \dots, F \qquad [124]$$

El primer armónico o armónico fundamental será:

$$f_1 = \frac{1}{T} \qquad [125]$$

De esta forma podemos situar el espectro cubierto por la señal multiseno, su banda de frecuencias será

$$[f_1, f_F] = \left[\frac{1}{T}, \frac{F}{T}\right] \qquad [126]$$

La señal multiseno puede ser implementada en el dominio del tiempo, generándose a partir de una suma de señales sinusoidales como sugiere la ecuación [120], o en el dominio de la frecuencia especificando la amplitud y fase de cada armónico y calculando la transformada inversa de Fourier. La segunda forma resulta computacionalmente más efectiva cuando el número de armónicos es elevado (Tan & Godfrey, 2019).

La respuesta de un sistema ante una señal sinusoidal es otra señal sinusoidal de la misma frecuencia, pero distinta fase y amplitud. De forma que, con este tipo de entrada, se podrá determinar la respuesta del sistema ante las frecuencias principales de la señal multiseno, obteniendo la amplitud y fase de salida para cada una de las frecuencias de entrada.

En la práctica las señales multiseno se generan con ordenador y aunque es una señal de tipo continuo, se debe generar a partir de un sistema muestreado. Existen varias formas de reconstruir una señal multiseno a partir de una secuencia de valores muestreados, las dos posibilidades más comunes son:

- Señales de banda limitada, estas señales se pueden generar a partir de una señal muestreada y señal de reloj de frecuencia muy elevada o un filtro adecuado que permita la reconstrucción de la señal de banda limitada
- Uso de un retenedor de orden cero, en este caso la señal reconstruida atraviesa el retenedor que tiene un espectro $\sin^2 x / x^2$. Para anular el efecto del retenedor se hará necesario el uso de una compensación para anular el espectro del retenedor. En el caso de señales discretas no será necesaria esta compensación (Tan & Godfrey, 2019).

3.8.1 Elección de la fase inicial de la señal multiseno

Cuando el objetivo de la señal multiseno es la identificación de sistemas lineales, se deberán ajustar las fases de los distintos armónicos para minimizar el valor pico-pico de la señal. El factor de cresta (vea apartado 3.1.2.1 en la página 20), será el parámetro que deberemos minimizar para mejorar la calidad de la señal. Un segundo factor a considerar es el índice PIPS (vea apartado 3.1.2.2 en la página 20), que en este caso interesará maximizar.

De todas las posibles elecciones para la fase de las sinusoides la peor es que todas tengan fase nula. Ya que, al ser una señal suma de cosenos, todas sus crestas coinciden en el instante inicial, tal y como puede verse en la Figura 15.

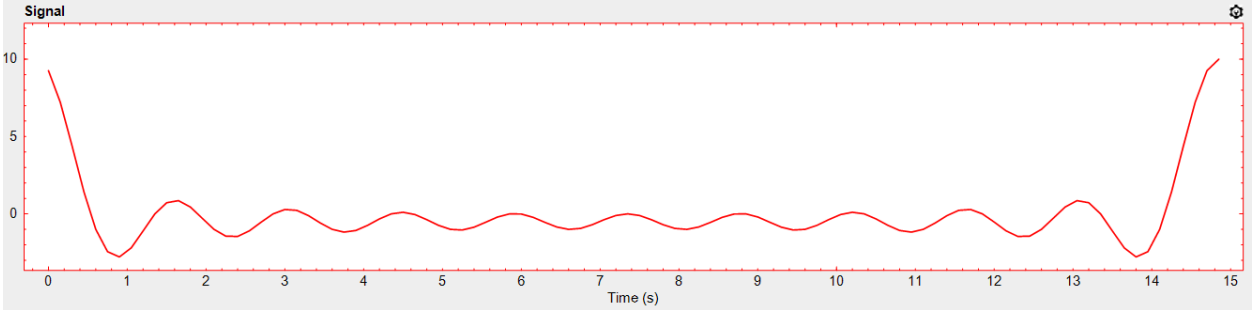

Figura 15: señal multiseno de 10 componentes y fase nula

En este caso el factor de cresta es $\sqrt{2F}$ que en el caso de la señal de la Figura 15 que es una composición de 10 sinusoides toma un valor de 4.472. El valor de PIPS para esta señal es de 34.988%.

La segunda opción es que la fase se tome de forma aleatoria para cada una de las señales entre 0 y 2π. En la Figura 16 puede verse un ejemplo de este tipo de señal. Con este diseño de la fase de las señales seno se pueden conseguir factores de cresta mayores a $\sqrt{logF}$ (Gersho et al., 1979). En el caso de la señal de la Figura 16 el factor de cresta es de 2.536 y el PIPS de 41.214%. Lo que representa una mejora con respecto al caso anterior. Pero al ser la fase aleatoria, la señal no es determinista y los factores de calidad de la señal varían en cada nueva señal.

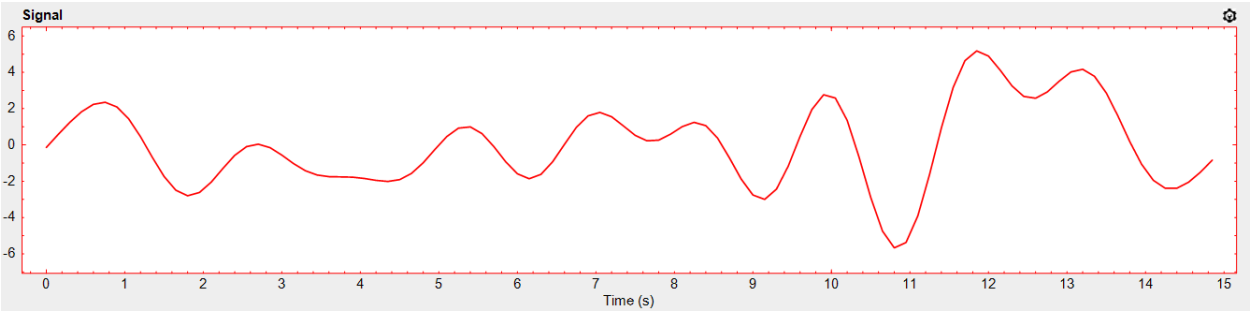

Figura 16: Señal multiseno de 10 componentes con fase inicial aleatoria

El diseño de fase propuesto por Newman (Newman, 1965) utiliza la siguiente fórmula para las fases

$$\phi_k = \frac{(k-1)^2\pi}{F} \qquad [127]$$

Está probado por Boyd (Boyd, 1986) que con esta elección de fase el factor de cresta es menor o igual a 2. Este diseño ha sido pensado para sistemas en que todos los armónicos están dentro de la banda de interés. En la Figura 17 se puede ver un ejemplo de esta señal en que el factor de cresta es de 1.851 y el índice PIPS del 59.568%

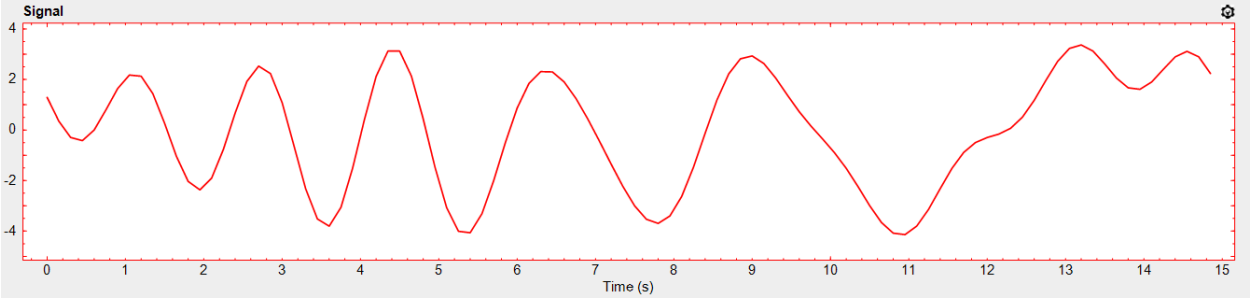

Figura 17: Señal multiseno de 10 componentes con fase según algoritmo Newman

La fase de cada una de las sinusoides puede calcularse según la fórmula de Schroeder (Schroeder, 1970)

$$\phi_k = \phi_1 - 2\pi \sum_{i=1}^{k-1} (k-i)p_i \qquad [128]$$

El objetivo de esta fórmula es minimizar los picos que pueden aparecer en la señal generada para de esa forma reducir el factor de cresta y maximizar el índice PIPS. Donde ϕ_1 es la fase inicial que se puede escoger arbitrariamente o fijar a cero para conseguir una señal determinista. p_i es la potencia relativa del armónico i.

Para el caso particular de que se requiera un espectro uniforme todos los armónicos tendrán la misma potencia y $p_i = 1/F$ siendo F el número total de armónicos y que la fase inicial $\phi_1 = 0$. Entonces la expresión [128] se puede simplificar como:

$$\phi_k = \frac{k(k-1)\pi}{F} \qquad [129]$$

En la Figura 18 se puede ver un ejemplo de señal multiseno de 10 componentes armónicas generada con el algoritmo de Schroeder. En este caso el factor de cresta es de 1.894 y el índice PIPS de 58.474%. Comparándolo con el resultado obtenido del algoritmo de Newman, se aprecia que los resultados son muy similares.

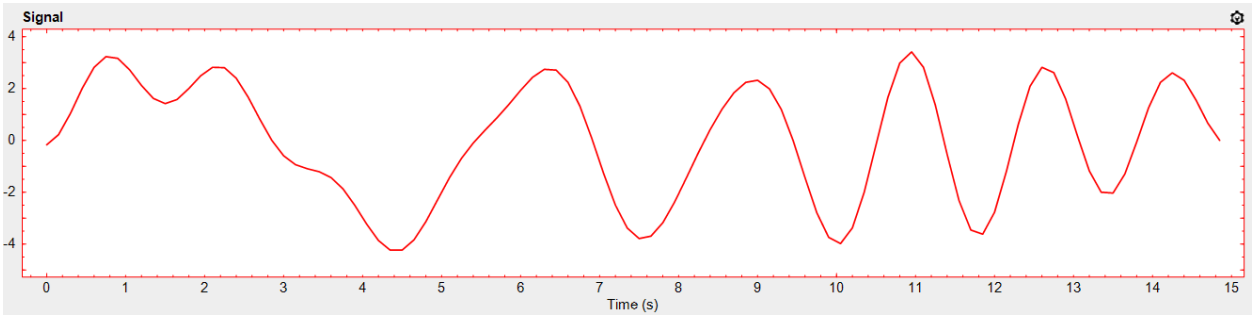

Figura 18: Señal multiseno de 10 componentes con fase según algoritmo Schroeder

En general el algoritmo de Schroeder genera bajos factores de cresta y proporciona un espectro de amplitud uniforme si los armónicos son consecutivos o impares (Tan & Godfrey, 2019).

Los siguientes algoritmos se aprovechan de la capacidad de optimización que ofrecen los ordenadores para generar la señal multiseno. El primero de ellos es el usado por la función *msinclip* de Matlab incluido en el toolbox de identificación de sistemas en el dominio de la frecuencia (Kollár et al., 1994, 2003). Este algoritmo está basado en el intercambio entre el dominio del tiempo y el de la frecuencia. Al comienzo se pueden escoger entre fases aleatorias o basadas en el algoritmo de Schroeder. A continuación, se determinan las amplitudes de cada armónico. Se combinan las señales en el dominio de la frecuencia y se calcula la transformada discreta de Fourier para pasar al dominio del tiempo. En el dominio del tiempo las señales se recortan a un nivel adecuado y se vuelve a calcular la transformada discreta de Fourier. En el dominio de la frecuencia se mantienen las nuevas fases mientras que la amplitud del espectro se limita a un determinado valor. El procedimiento se repite de forma iterativa hasta que no se consigue una reducción en el factor de cresta con respecto a la iteración anterior.

El segundo de los algoritmos es denominado de norma infinita o de minimización de la norma l_P (Guillaume et al., 1991) y está basado en el algoritmo de Polya (Bell & Polya, 1945). La norma l_P viene dada por la expresión:

$$l_p(u) = \left[\frac{1}{T_N}\int_0^{T_N} |u(t)|^p dt\right]^{1/p}, p \geq 1 \qquad [130]$$

Normalmente el algoritmo se comienza para p=2. Una vez que se obtienen un conjunto de fases que minimizan la ecuación [130] para p=2, el valor de p se aumenta a 4. Cuando la ecuación [130] se minimiza para 4, el valor de p se aumenta a 8. Durante el procedimiento, p va tomando los valores 2, 4, 8, 16, 32, 64,…etc.

Este algoritmo está disponible en el toolbox de identificación de sistemas en el dominio de la frecuencia (Kollár et al., 1994, 2003) en la función *crestmin*. Este algoritmo obtiene mejores resultados que el de intercambio entre el dominio del tiempo y la frecuencia (Pintelon & Schoukens, 2001).

3.8.2 Función de autocorrelación de una señal multiseno

Si consideramos la señal multiseno como la definida en la ecuación [120], donde todas las amplitudes son iguales a A_i =1/F

$$u(t) = \sum_{i=1}^{F} A_i \cos(2\pi f_i t + \phi_i), 0 \leq t \leq T$$

Entonces una de estas sinusoides será

$$u_i(t) = \frac{\cos(2\pi f_i t + \phi_i)}{F} \qquad [131]$$

A partir de la ecuación [33], se tiene que:

$$R_{xxi}(\tau) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} \frac{1}{F^2} cos(2\pi f_i t + \phi_i) cos(2\pi f_i t + 2\pi f_i \tau + \phi_i) dt^{10}$$
$$= \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} \frac{1}{2F^2} [cos(2\pi f_i \tau) + cos(4\pi f_i t + 2\pi f_i \tau + 2\phi_i)] dt$$
$$= \frac{1}{2F^2} \lim_{T\to\infty} \frac{1}{2T} \left[t\, cos(2\pi f_i \tau) + \frac{sin(4\pi f_i t + 2\pi f_i \tau + 2\phi_i)}{2\pi f_i} \right]^{T} \quad [132]$$
$$= \frac{1}{2F^2} \lim_{T\to\infty} \frac{1}{2T} (T - (-T))\, cos(2\pi f_i \tau)$$
$$+ \frac{sin(4\pi f_i T + 2\pi f_i \tau + 2\phi_i) - sin(4\pi f_i (-T) + 2\pi f_i \tau + 2\phi_i)}{2\pi f_i} =$$
$$= \frac{1}{2F^2} \lim_{T\to\infty} cos(2\pi f_i \tau) + \frac{1}{2T} \frac{sin(4\pi f_i T + 2\pi f_i \tau + 2\phi_i)}{\pi f_i}$$

Al tomar límites, el primero de los términos será el mostrado en la ecuación [133], ya que no hay ningún término dependiente de T

$$R_{xxi}(\tau) = \frac{1}{2F^2} cos(2\pi f_i \tau) \quad [133]$$

En el caso del segundo término, el seno del ángulo está afectado por el valor de *T*, que cuando *T* tiende a infinito es desconocido. Lo único que se sabe del término $sin(4\pi f_i T + 2\pi f_i \tau + 2\phi_i)$ es que su valor estará entre -1 y 1. De forma que el término $\frac{1}{2T}$ al tomar el límite con T tendiendo a infinito hará nulo todo el segundo término de la expresión. Por tanto, la ecuación [133] será el resultado total de la expresión. Extendiendo el resultado al resto de los armónicos de la señal se tendrá:

$$R_{xx}(\tau) = \sum_{i=1}^{F} \frac{1}{2F^2} cos(2\pi f_i \tau) \quad [134]$$

Y en el caso en que las amplitudes de las sinusoides no fueran iguales, pero si constantes, se tendrá la siguiente expresión para la función de autocorrelación de la señal multiseno.

$$R_{xx}(\tau) = \sum_{i=1}^{F} \frac{1}{2} A_i^2 cos(2\pi f_i \tau) \quad [135]$$

La función de autocorrelación ha perdido la información de la fase inicial de cada una de las sinusoides. Por lo tanto, el resultado será el mismo independientemente del algoritmo empleado para seleccionar las fases iniciales de las sinusoides.

[10] $\cos a + \cos b = 2\cos\frac{a+b}{2} * \cos\frac{a-b}{2}$

3.8.3 Espectro de potencia de una señal multiseno

Si de nuevo consideramos la señal multiseno definida en la ecuación [120], donde todas las amplitudes son iguales a A_i =1/F

$$u(t) = \sum_{i=1}^{F} A_i \cos(2\pi f_i t + \phi_i), 0 \le t \le T$$

Entonces cada una de estas sinusoides será igual que la presentada en la ecuación [131]. En el apartado anterior, se encontró que la función de autocorrelación de dicha función es la presentada en la ecuación [133]

$$R_{xxi}(\tau) = \frac{1}{2F^2} cos(2\pi f_i \tau)$$

A partir de la ecuación [64], haciendo el cambio de variable $2\pi f_i = \omega_i$ teniendo en cuenta que el periodo $T_i = 2\pi/\omega_i$ se tendrá:

$$S_{xxi}\left(f = \frac{k\omega_i}{2\pi}\right) = \frac{\omega_i}{2\pi}\int_0^{2\pi/\omega_i} R_{xxi}(\tau)\cos\left(\frac{2\pi k\tau\omega_i}{2\pi}\right)d\tau = \frac{\omega_i}{4\pi F^2}\int_0^{2\pi/\omega_i} cos(\omega_i\tau)\cos(k\tau\omega_i)\,d\tau \;, k = 0, \pm1, \pm2, \ldots \quad [136]$$

Para k=0

$$S_{xxi}(f=0) = \frac{\omega_i}{4\pi F^2}\int_0^{2\pi/\omega_i} cos(\omega_i\tau)d\tau = \frac{\omega_i}{4\pi F^2\omega_i}[sin(\omega_i\tau)]_0^{2\pi/\omega_i} = 0 \quad [137]$$

Para k=1

$$S_{xxi}\left(f = \frac{\omega_i}{2\pi}\right) = \frac{\omega_i}{4\pi F^2}\int_0^{2\pi/\omega_i} cos(\omega_i\tau)\cos(\omega_i\tau)\,d\tau = \frac{\omega_i}{4\pi F^2}\int_0^{2\pi/\omega_i}\frac{1}{2}\big(1 + cos(2\omega_i\tau)\big)d\tau = \frac{\omega_i}{8\pi F^2}\left[\tau + \frac{\sin 2\omega_i\tau}{2\omega_i}\right]_0^{2\pi/\omega_i} = \frac{1}{4F^2} \quad [138]$$

De igual forma para k=-1

$$S_{xxi}\left(f = -\frac{\omega_i}{2\pi}\right) = \frac{1}{4F^2} \quad [139]$$

Para k=2

$$S_{xxi}\left(f = \frac{2\omega_i}{2\pi}\right) = \frac{\omega_i}{4\pi F^2}\int_0^{\frac{2\pi}{\omega_i}} cos(\omega_i\tau)\cos(2\tau\omega_i)\,d\tau = \frac{\omega_i}{24\pi F^2}[3\sin\omega_i\tau - \sin 3\omega_i\tau]_0^{2\pi/\omega_i} \quad [140]$$

Que resulta ser cero, y ocurrirá los mismo para valores mayores de k. Por tanto, el espectro de la función seno presenta una sola línea a la frecuencia f_i y $-f_i$. Para la señal multiseno tendremos un grupo de líneas a la frecuencia de cada uno de los armónicos.

En el caso en que cada una de las componentes de la señal multiseno tuviera una amplitud diferente, el resultado para cada una de ellas será:

$$S_{xxi}\left(f = \pm\frac{\omega_i}{2\pi}\right) = \frac{A_i^2}{4} \qquad [141]$$

Si por ejemplo tenemos una señal multiseno de 10 componentes, el resultado será similar al de la Figura 19.

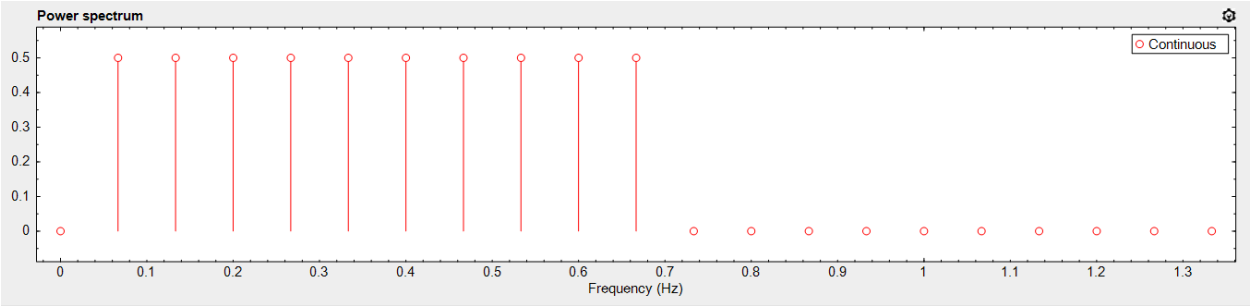

Figura 19: Espectro de una señal multiseno

4 Diseño e implementación de la herramienta.

En los siguientes apartados se va a detallar el proceso de diseño de la herramienta presentada como resultado de este TFG. Como idioma de la herramienta se ha escogido el idioma inglés para facilitar su distribución dentro del mundo académico.

En un primer lugar se realizó una validación de las señales generadas por Sysquake usando como sistema de comparación el software de Matlab.

4.1 Generación de señales PRBS

En este apartado se va a discutir el algoritmo utilizado para generar la señal PRBS.

4.1.1 Generación de señales PRBS usando Matlab

La herramienta Matlab de MathWorks dispone de una serie de toolbox orientados a la regulación automática y el control. Uno de estos toolbox es el de identificación de sistemas que posee funciones de apoyo para esta tarea. La función que permite generar una señal con características específicas para estimular un sistema y estudiar su respuesta es *idinput* (MathWorks, 2022).

Esta función admite varios parámetros diferentes. Para la generación de señales PRBS va a interesar el uso de 4 parámetros.

idinput([T, Un, Np], Tipo, Banda, Rango)

- El primero de los parámetros [*T, Nu, Np*] indica que se va a generar una señal de entrada binaria periódica de *Nu*-canales de periodo *T* y número total de periodos *Np*. Cada canal tiene la misma longitud *T*Np*.
- El segundo de los parámetros es el *Tipo* de señal. Los posibles tipos de señales que se pueden generar son:
  - ‘rbs’ Random binary signal, señal binaria aleatoria
  - ‘rgs’ Random Gaussian signal, señal Gaussiana aleatoria
  - ‘prbs’ Pseudo-random binary signal, señal binaria pseudo-aleatoria
  - ‘sine’ Sum-of-sinusoidals signal, señal multiseno
- El tercer parámetro *Banda*, representa el rango de frecuencias de la señal. En el caso de las señales PRBS la banda debe especificar el inverso del ciclo de reloj de la señal.
- El último parámetro *Rango*, especifica el rango de amplitudes de la señal.

El siguiente ejemplo, muestra cómo utilizando el comando *idinput* generar una señal de entrada de tipo PRBS y mostrarla en un gráfico que se puede ver en la Figura 20.

```
%Inicialización
clear all
close all

Ts=1;        %Periodo de muestreo del objeto idata
Tsw=4;       %Número de unidades de tiempo del periodo de la señal
nr=5;        %Tamaño del registro de desplazamiento
N=(2^nr)-1;  %Longitud de la secuencia

a=1;         %Amplitud de la secuencia
Banda=[0,1/Tsw];     %Banda de frecuencias desde 0 hasta 1/Tsw
T0=N*Tsw;    %Periodo de la señal
```

```
Np=1;        %Número de repeticiones de la señal


%[T0 1 Np] =[Periodo de la señal T0=N*Tsw, numero de canales, Número de
periodos]
prbs_gen=idinput([T0 1 Np],'prbs',Banda,[-a,a]); %Señal generada

%Representacion grafica
u = iddata([],prbs_gen,Ts);
figure(1);
plot(u);
title('PRBS input signal');
```

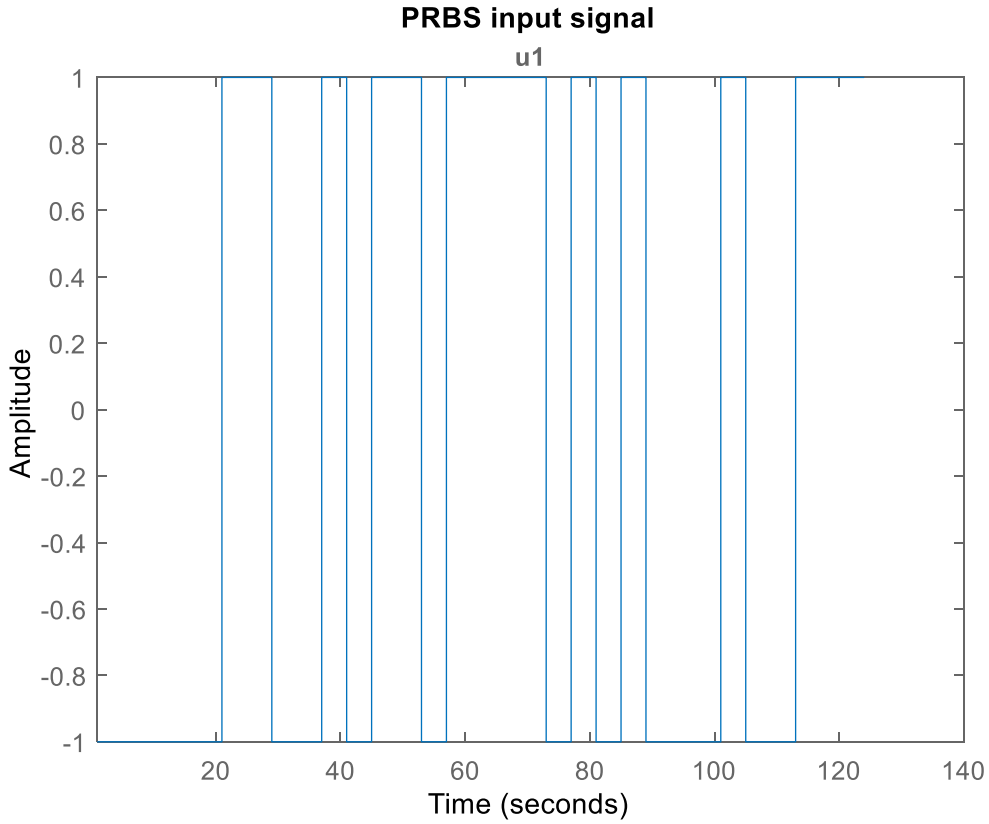

Figura 20: Señal PRBS generada por Matlab para un registro de 5 bits

Al ser la señal generada en la Figura 20 de un periodo de 4 unidades de tiempo y un registro de 5 bits, da una longitud de 124 unidades de tiempo para la secuencia creada.

A continuación, se ha fijado como objetivo determinar cómo genera Matlab esta secuencia. Ya que el objetivo final del TFM es poder generar secuencias PRBS usando Sysquake, Matlab va a ser una herramienta que permita comparar los resultados obtenidos con Sysquake para así poder validar las secuencias generadas.

El siguiente paso es desarrollar un algoritmo que sin necesidad de utilizar el comando *idinput* genere la misma secuencia que éste. A este algoritmo le denomino gen_prbs_raw. A continuación, se muestra el código del mismo:

```
% definition of pseudo-random binary sequences generator
```

```
% a output value +a / -a
% Tsw number of cicles per conmutation
% n size of shift register
% Np number of signal periods
% return signal prbs and time vector
function [prbs, time] = gen_prbs_raw(a, Tsw, n, Np)

prbs =[]; %Initial value
if (n>24)
    printf(sError register too bigs);
    return
end
permut = [1,2,0,0;1,3,0,0;1,4,0,0;2,5,0,0;1,6,0,0;1,7,0,0;1,2,7,8;4,9,0,0;...
    3,10,0,0;9,11,0,0;6,8,11,12;9,10,12,13;4,8,13,14;14,15,0,0;4,13,15,16;...
    14,17,0,0;11,18,0,0;1,2,6,19;17,20,0,0;19,21,0,0;21,22,0,0;...
    18,23,0,0;17,22,23,24];
reg = [linspace(1,1,n)]; %Initial state 11111...1
bits = ((2^n) - 1) * Np;

for j = 1:bits
    if (reg(n) == 0)
      prbs = [prbs linspace(a,a,Tsw)];
    else
      prbs = [prbs linspace(-a,-a,Tsw)];
    end
    %delay Tsw
    if (permut(n-1,3)==0)
    c = xor( reg(permut(n-1,1)), reg(permut(n-1,2)));
    elseif (permut(n-1,4)==0)
    c = xor(xor( reg(permut(n-1,1)), reg(permut(n-1,2))), reg(permut(n-
1,3)));
    else
    c = xor(xor( reg(permut(n-1,1)), reg(permut(n-1,2))), xor(reg(permut(n-
1,3)), reg(permut(n-1,4))));
    end
    for i = n:-1:2
    reg(i)=reg(i-1);
    end
    reg(1) = c;
end
%prbs = transpose(prbs)
time = 0:bits*Tsw-1;
%time = 0:clk:(bits*Tsw*clk)-clk;
end
```

El funcionamiento del algoritmo es muy simple, se declara la función a la que se le pasan 4 parámetros: *a* – amplitud, *Tsw* – periodo de la señal, *n* – tamaño del registro y *Np* – número de repeticiones de la secuencia.

En primer lugar, se inicializa el vector que almacena la secuencia *prbs* como un vector vacío que será el resultado del algoritmo.

El tamaño máximo de secuencia que se puede genera es de 2^{24}-1, por ello si n es mayor que 24 se genera un error. Este tamaño no representa ninguna limitación del algoritmo, es una elección de diseño.

La matriz de permutaciones, *permut*, indica que bits del registro de desplazamiento van a realizar una función XOR, un 0 en la matriz indica que esa posición no es necesaria. Se

comenzaron utilizando las realimentaciones definidas en la Tabla 5 y conforme se encontraban las utilizadas por Matlab se fueron adaptando.

A continuación, se inicializa la variable *reg* que actúa como registro de desplazamiento con todos sus bits a uno.

Se define la variable bits como el número total de bits que tendrá la secuencia. El bucle for se repite desde *j* = 1 hasta bits.

En cada iteración del bucle se van añadiendo los bits correspondientes al vector *prbs*.

Se consulta el valor de bit de más peso del registro de desplazamiento, si este es 1 se añaden *Tws* ciclos de reloj al registro *prbs* con valor -a y en caso contrario se añade a.

El siguiente paso es realizar la realimentación determinada en el registro *perm* para el numero de bits del registro actual. Esta realimentación puede ser una XOR de 2, 3 o 4 posiciones del registro de desplazamiento de ahí que se verifique si para esa línea la matriz tiene 2, 3 o 4 componentes y se haga una operación XOR u otra. El resultado de la XOR va a parar a la variable c.

A continuación, se realiza el desplazamiento dentro del registro, pasando el bit *i*-1 al bit *i* hasta introducir en el bit de menos peso el nuevo valor c calculado previamente.

El tiempo time se determina como un vector lineal desde 0 hasta bits*Tsw-1.

Y el resultado que proporciona este algoritmo se compara con el generado por el toolbox de Matlab. El programa se denomina *comparador_PRBS.m* y se presenta a continuación:

```
%Inicialización
clear all
close all

Ts=1;
Tsw=4;  %number of states generated for every output in the shift register
nr=5;   %number of bits in the shift register
N=(2^nr)-1; %Number el sequence lenght

a=1;    %Maximum value of output signal
Banda=[0,1/Tsw];
T0=N*Tsw;
Np=1; %Number of complete cicles that signal takes

%[T0 1 Np] =[Periodo de la señal T0=N*Tsw, numero de canales, Número de
periodos]
prbs_gen=idinput([T0 1 Np],'prbs',Banda,[-a,a]);
[prbs,t]=gen_prbs_raw(a,Tsw,nr,Np);

%Representacion grafica
u = iddata([],prbs_gen,Ts);
u2= iddata([],transpose(prbs),Ts);
figure(2);
plot(u);
hold on;
plot(u2);
title('PRBS input signal');
```

La gráfica obtenida como resultado se presenta en la Figura 21.

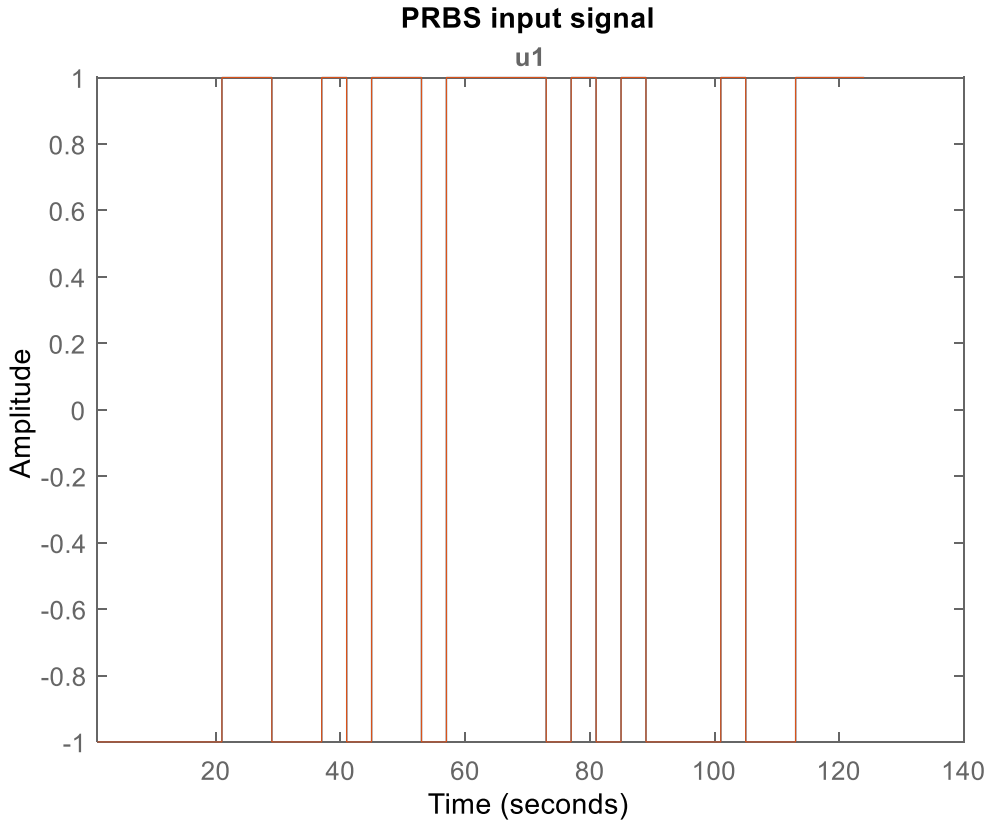

Figura 21: Resultado de la comparación entre el resultado de idinput y gen_prbs_raw

4.1.2 Señales PRBS generadas por Matlab y Sysquake

En la Tabla 5, se encuentran los estados de realimentación para longitudes de registro de hasta 100 bits. Para realizar el programa en Sysquake y verificar los resultados obtenidos, se pensó en utilizar Matlab como referencia. Sin embargo, al realizar las primeras pruebas se vio que Matlab no seguía las mismas reglas de realimentación que las proporcionadas por Godfrey en la citada tabla. Después de realizar una pequeña investigación se descubrió que Matlab siempre comenzaba con todos los bits del registro de desplazamiento a 1 lógico y que la salida era invertida. De forma que la salida +A corresponde con el estado 0 de salida y -A con el estado 1. Después de localizar cuales son las realimentaciones utilizadas por Matlab en el registro de desplazamiento se obtuvo el resultado de la Tabla 6 que muestra las realimentaciones usadas para registros de longitud de 2 a 24 bits.

Tabla 6: Estados realimentados en el registro de desplazamiento

N	Estados realimentados
2	1,2
3	1,3
4	1,4
5	2,5
6	1,6
7	1,7
8	1,2,7,8
9	4,9
10	3,10
11	9,11
12	6,8,11,12
13	9,10,12,13
14	4,8,13,14
15	14,15
16	4,13,15,16
17	14,17
18	11,18
19	1,2,6,19
20	17,20
21	19,21
22	21,22
23	18,23
24	17,22,23,24

El algoritmo utilizado en Matlab para descubrir estas permutaciones es de fuerza bruta, compara el resultado del vector binario obtenido por *idinput* con una variante de algoritmo expuesto anteriormente gen_prbs_perm, adaptado para generar un vector binario comparable con el generado por *idinput*. Si se encuentra coincidencia con la permutación probada se saca un mensaje de éxito y en caso contrario de fracaso. A continuación, se presenta este código para el caso de longitud 19, para otros casos basta con sustituir el tamaño del registro por otro número en la variable nr.

```
clear all;
close all;

Ts=0.1;
Tsw=1;       %Periodo de la señal
nr=19;       %Tamaño del registro
N=(2^nr)-1; %Longitud de la secuencia

a=1;         %Amplitud de la secuencia
Banda=[0,1/Tsw];     %Ancho de banda
T0=N*Tsw;    %Periodo de la señal
Np=1;        %Repeticiones de la señal

%[T0 1 Np] =[Periodo de la señal T0=N*Tsw, numero de canales, Número de
periodos]
```

```
prbs_gen=idinput([T0 1 Np],'prbs',Banda,[-a,a]);
for i=1:16
    for j=i+1:17
        for k=j+1:18
            prbs=gen_prbs_perm(a,Tsw,nr,Np,[i j k nr]);
            if(prbs == prbs_gen)
                fprintf('Permutación encontrada %d %d %d para n=%d\n', i, j,
k, nr);
            else
                fprintf('Permutacion fallida %d %d %d para n=%d\n', i, j, k,
nr);
            end
        end
    end
end

%definition of pseudo-random binary sequences generator
% a output value +a / -a
% Tsw number of cicles per conmutation
% n size of register
% Np number of signal periods
function [prbs] = gen_prbs_perm(a, Tsw, n, Np, permut)

prbs =[]; %Initial value
if (n>24)
    printf(sError register too bigs);
    return
end
reg = [linspace(1,1,n)]; %Initial state 11111...1
bits = ((2^n) - 1) * Np;

for j = 1:bits
    if (reg(n) == 0)
      prbs = [prbs linspace(a,a,Tsw)];
    else
      prbs = [prbs linspace(-a,-a,Tsw)];
    end
    %delay Tsw
    if (permut(1,3)==0)
    c = xor( reg(permut(1,1)), reg(permut(1,2)));
    elseif (permut(1,4)==0)
    c = xor(xor( reg(permut(1,1)), reg(permut(1,2))), reg(permut(1,3)));
    else
    c = xor(xor( reg(permut(1,1)), reg(permut(1,2))), xor(reg(permut(1,3)),
reg(permut(1,4))));
    end
    for i = n:-1:2
    reg(i)=reg(i-1);
    end
    reg(1) = c;
end
prbs=transpose(prbs);
end
```

La señal resultante se puede ver en la Figura 22.

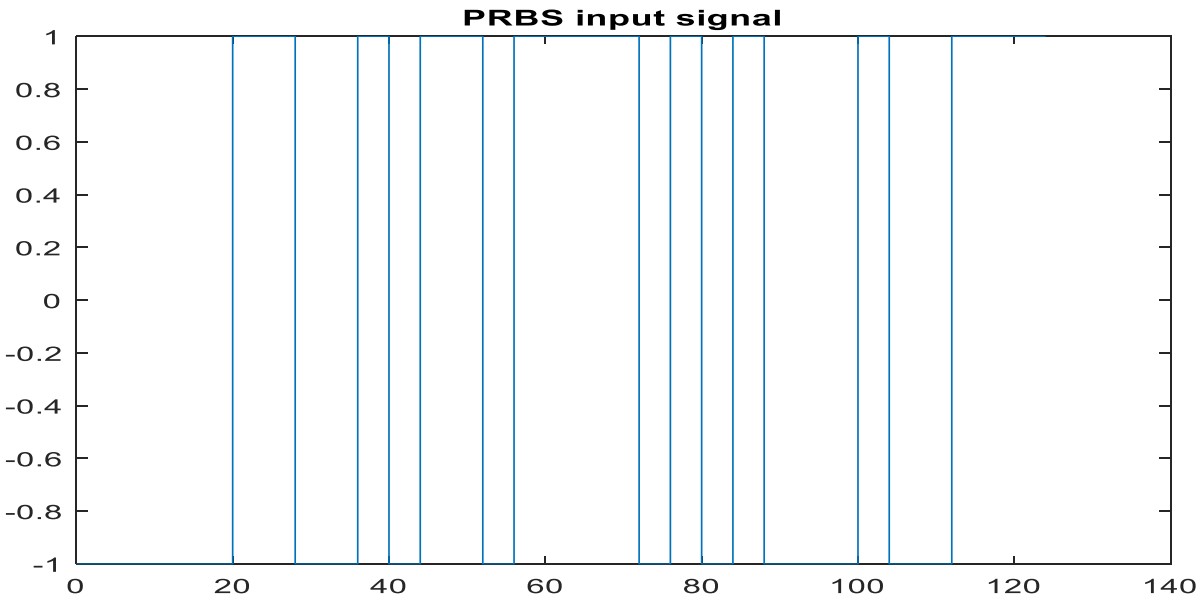

Figura 22: Señal PRBS

La misma señal generada por Sysquake puede verse en la Figura 23.

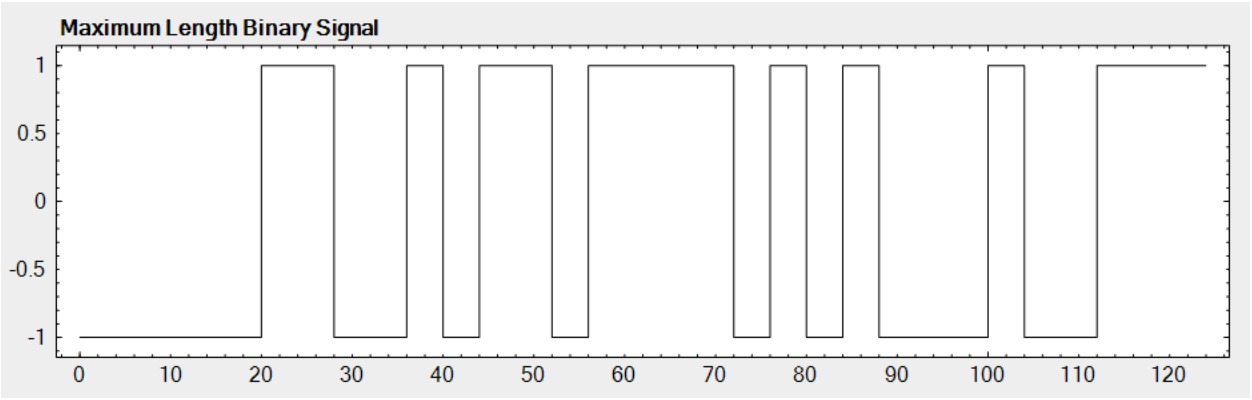

Figura 23: Señal PRBS en Sysquake

Una vez calculada la Tabla 6, los resultados obtenidos ya coincidían para cualquier longitud de registro. Se descartó realizar la búsqueda para registros de longitud superior a 24 bits porque para la identificación de sistemas las secuencias de longitudes inferiores son más que suficientes. Por otra parte, se consideró que para el TFM no se precisarán secuencias generadas a partir de registros de desplazamiento de más de 11 bits, por lo que la herramienta final solo admite secuencias de hasta 11 bits.

La señal que el sistema debe generar para realizar la identificación de sistemas, es la obtenida con la función *gen_prbs_raw*. Sin embargo, para representar gráficamente la función todavía se debe usar el objeto del toolbox *iddata*. Se propone por tanto eliminar esta necesidad modificando de nuevo la función para que genere un vector de datos representable por la función plot. Esta nueva función es *gen_prbs* y se presenta a continuación.

```
% definition of pseudo-random binary sequences generator
```

```
% a output value +a / -a
% Tsw number of cicles per conmutation
% n size of shift register
% Np number of signal periods
% return signal prbs and time vector
function [prbs, time] = gen_prbs(a, Tsw, n, Np)

prbs =[]; %Initial value
time =[];
if (n>24)
    printf("Error register too bigs");
    return
end
permut = [1,2,0,0;1,3,0,0;1,4,0,0;2,5,0,0;1,6,0,0;1,7,0,0;1,2,7,8;4,9,0,0;...
    3,10,0,0;9,11,0,0;6,8,11,12;9,10,12,13;4,8,13,14;14,15,0,0;4,13,15,16;...
    14,17,0,0;11,18,0,0;1,2,6,19;17,20,0,0;19,21,0,0;21,22,0,0;...
    18,23,0,0;17,22,23,24];
reg = [linspace(1,1,n)]; %Initial state 11111...1
bits = ((2^n) - 1) * Np;
state_ant = reg(n);
t=0;
for j = 1:bits
    if (reg(n) ~= state_ant)
        time = [time t];
        if(state_ant == 0)
            prbs = [prbs a];
        else
            prbs = [prbs -a];
        end
        state_ant = reg(n);
    end
    time = [time t:Tsw+t-1];
    t = t + Tsw;
    if (reg(n) == 0)
      prbs = [prbs linspace(a,a,Tsw)];
    else
      prbs = [prbs linspace(-a,-a,Tsw)];
    end
    %delay Tsw
    if (permut(n-1,3)==0)
        c = xor( reg(permut(n-1,1)), reg(permut(n-1,2)));
    elseif (permut(n-1,4)==0)
        c = xor(xor( reg(permut(n-1,1)), reg(permut(n-1,2))), reg(permut(n-
1,3)));
    else
        c = xor(xor( reg(permut(n-1,1)), reg(permut(n-1,2))),
xor(reg(permut(n-1,3)), reg(permut(n-1,4))));
    end
    for i = n:-1:2
        reg(i)=reg(i-1);
    end
    reg(1) = c;
end
if(state_ant == 0)
    prbs = [prbs a];
else
    prbs = [prbs -a];
end
time = [time t];
```

```
%prbs = transpose(prbs)
%time = 0:Tsw*bits-1;
end
```

La diferencia entre esta versión y la anterior está en que hay que modificar el vector de tiempo para que cuando la señal cambie el vector tenga un punto con el instante de tiempo *k* y el valor -*a* y un segundo punto con el instante *k* y el valor *a*, de esa forma *plot* podrá representar esa transición como una línea recta que en el instante *k* vaya del valor -*a* al valor *a*.

El programa de test de la función es el siguiente:

```
%Inicialización
clear all
close all

%Ts=1
Tsw=4;  %number of states generated for every output in the shift register
nr=5;   %number of bits in the shift register
N=(2^nr)-1; %Number el sequence lenght

a=1;    %Maximum value of output signal
Banda=[0,1/Tsw];
T0=N*Tsw;
Np=1; %Number of complete cicles that signal takes

[prbs,t]=gen_prbs(a,Tsw,nr,Np);

%Representacion grafica
figure(2);
plot(t,prbs);
[p,time]=gen_prbs_raw(a,Tsw,nr,Np);
title('PRBS input signal');
```

El vector [prbs, t] contiene la señal a representar por *plot* y [p, time] la señal a introducir al sistema a ensayar. La Figura 24 muestra el resultado de la ejecución del programa.

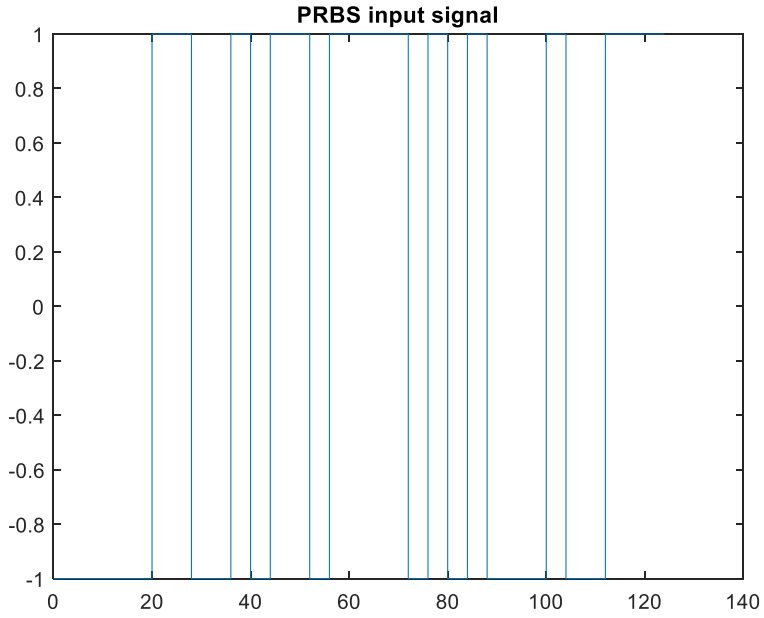

Figura 24: Gráfico generado por la función gen_prbs

4.1.3 Generación de señales PRBS usando Sysquake

El principal objetivo del TFM es generar las señales de entrada utilizadas en la identificación de sistemas usando Sysquake. La interactividad inherente a Sysquake permitirá de una forma sencilla personalizar la señal generada.

El primer paso es trasladar los dos algoritmos de generación de la señal PRBS: *gen_prbs* y *gen_prbs_RAW*. A continuación, se muestra el código en Sysquake.

```
%-------------------------------------------------
function (prbs, time) = gen_prbs_RAW(a, Tsw, nr, Np)
% definition of pseudo-random binary sequences generator
% a output value +a / -a
% Tsw number of cicles per conmutation
% n size of shift register
% Np number of signal periods
% clk time per clock cicle in seconds
% return signal prbs and time vector


  prbs =[]; //Initial value
  if nr > 24.0
    fprintf('Error register too big');
    return;
  end
  permut                                                                    =
[1,2,0,0;1,3,0,0;1,4,0,0;2,5,0,0;1,6,0,0;1,7,0,0;1,2,7,8;4,9,0,0;3,10,0,0;9,1
1,0,0;6,8,11,12;9,10,12,13;4,8,13,14;14,15,0,0;4,13,15,16;14,17,0,0;11,18,0,0
;1,2,6,19;17,20,0,0;19,21,0,0;21,22,0,0;18,23,0,0;17,22,23,24];
  reg = [linspace(1,1,nr)]; //Initial state 11111...1
  bits = ((2^nr) - 1) * Np;
```

```
  for j = 1:bits
    if reg(nr) == 0
      prbs = [prbs;transpose(linspace(a,a,Tsw))];
    else
      prbs = [prbs;transpose(linspace(-a,-a,Tsw))];
    end
    //delay Tsw
    if permut(nr-1,3)==0
      c = xor( reg(permut(nr-1,1)), reg(permut(nr-1,2)));
    elseif permut(nr-1,4)==0
      c = xor(xor( reg(permut(nr-1,1)), reg(permut(nr-1,2))), reg(permut(nr-
1,3)));
    else
      c     =     xor(xor(     reg(permut(nr-1,1)),     reg(permut(nr-1,2))),
xor(reg(permut(nr-1,3)), reg(permut(nr-1,4))));
    end
    for i = nr:-1:2
      reg(i)=reg(i-1);
    end
    reg(1) = c;
  end
  time = 0:bits*Tsw-1;
  //time = 0:clk:(bits*Tsw*clk)-clk;

%-------------------------------------------------
function (prbs, time) = gen_prbs(a, Tsw, nr, Np)
% definition of pseudo-random binary sequences generator
% a output value +a / -a
% Tsw number of cicles per conmutation
% n size of shift register
% Np number of signal periods
% clk time per clock cicle in seconds
% return signal prbs and time vector


  prbs =[]; //Initial value
  time =[];
  if nr > 24.0
    fprintf('Error register too big');
    return;
  end
  permut                                                                    =
[1,2,0,0;1,3,0,0;1,4,0,0;2,5,0,0;1,6,0,0;1,7,0,0;1,2,7,8;4,9,0,0;3,10,0,0;9,1
1,0,0;6,8,11,12;9,10,12,13;4,8,13,14;14,15,0,0;4,13,15,16;14,17,0,0;11,18,0,0
;1,2,6,19;17,20,0,0;19,21,0,0;21,22,0,0;18,23,0,0;17,22,23,24];
  reg = [linspace(1,1,nr)]; //Initial state 11111...1
  bits = ((2^nr) - 1) * Np;
  state_ant = reg(nr);
  t=0;

  for j = 1:bits
    if reg(nr) ~= state_ant
      time = [time;t];
      if state_ant == 0
        prbs = [prbs;a];
      else
        prbs = [prbs;-a];
      end
      state_ant = reg(nr);
```

```
    end
    time = [time;transpose(t:(Tsw+t-1))];
    t = t + Tsw;
    if reg(nr) == 0
      prbs = [prbs;transpose(linspace(a,a,Tsw))];
    else
      prbs = [prbs;transpose(linspace(-a,-a,Tsw))];
    end
    //delay Tsw
    if permut(nr-1,3)==0
      c = xor( reg(permut(nr-1,1)), reg(permut(nr-1,2)));
    elseif permut(nr-1,4)==0
      c = xor(xor( reg(permut(nr-1,1)), reg(permut(nr-1,2))), reg(permut(nr-
1,3)));
    else
      c     =     xor(xor(     reg(permut(nr-1,1)),     reg(permut(nr-1,2))),
xor(reg(permut(nr-1,3)), reg(permut(nr-1,4))));
    end
    for i = nr:-1:2
      reg(i)=reg(i-1);
    end
    reg(1) = c;
  end
  if state_ant == 0
    prbs = [prbs;a];
  else
    prbs = [prbs;-a];
  end
  time = [time;t];
  //time = 0:clk:(bits*Tsw*clk)-clk;
```

Las únicas diferencias entre estos algoritmos y los de Matlab tienen que ver con la forma en que Sysquake y Matlab trabajan con los vectores. La función *gen_prbs_RAW* genera la señal necesaria para realizar la identificación de sistemas y la función *gen_prbs* genera el vector necesario para dibujar la forma de onda de la señal PRBS.

En la Figura 25 puede apreciarse que el resultado de este algoritmo es el mismo que se había obtenido con Matlab que se representó en la Figura 24.

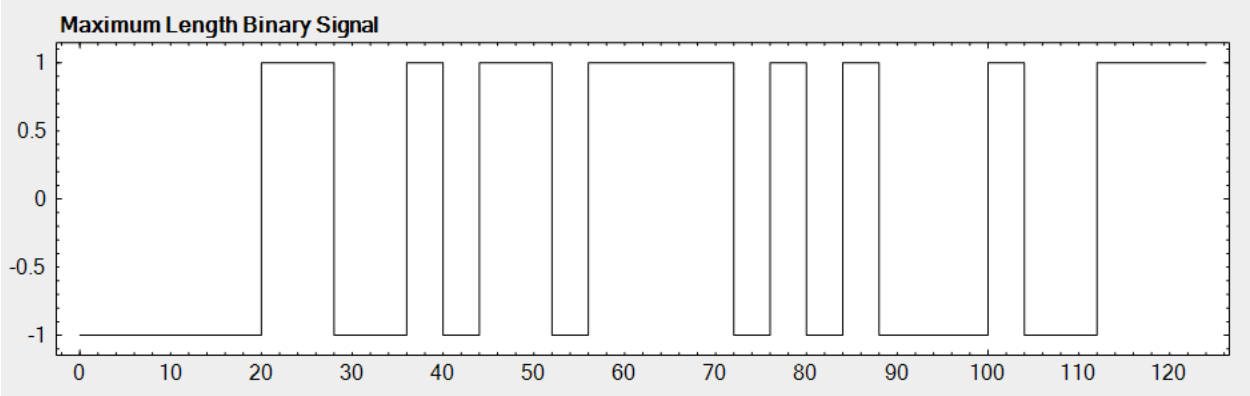

Figura 25: Señal PRBS en Sysquake

4.2 Generación de señales multiseno

4.2.1 Cálculo de las señales multiseno usando Matlab

Igual que en el caso de la señal PRBS, Matlab va a ser una herramienta útil para generar las señales multiseno. De nuevo la función *idinput* permite generar este tipo de señal. En este caso el segundo parámetro permite seleccionar el tipo de señal deseado a *'sine'*.

A continuación, se presenta el código Matlab que permite obtener la señal multiseno.

```
%Inicialización
clear all
close all

Ts = 0.1;            %Periodo de muestreo in segundos
Samples = 100;       %Número de muestras por periodo
Np = 3;              %Número de periodos
a = 1;               %Amplitud máxima
Range = [-a a];      %Rango de valores

Tsw = 1;             %Periodo de la señal
Banda=[0,1/Tsw];     %Banda de frecuencias
NumSinusoids = 12;   %Numero de sinusoides generadas
NumTrials = 10;      %Se intentan Numtrials ajustes de fase para encontrar la
menor dispersión de señal
GridSkip = 2;        %Se usa para caracterizar la frecuencia de las sinusoides
SineData = [NumSinusoids,NumTrials,GridSkip];
%frequencies of the sinusoids from the intersection of the frequency grid
2*pi*[1:GridSkip:fix(Period/2)]/Period and the passband pi*Band
%[Samples 1 Np] =[muestras por periodo, numero de canales, Número de
periodos]
sine_gen=idinput([Samples 1 Np],'sine',Banda,Range,SineData);

%Representacion grafica
u = iddata([],sine_gen,Ts,'Timeunit','seconds');
figure(3);
plot(u);
title('Multisine input signal')

%Espectro de la señal
figure(4);
ufft = fft(sine_gen);       %Transformada rapida de fourier
Fs = 2*pi/Ts;        %Frecuencia de muestreo en radianes por segundo
L = length(sine_gen);
w = (0:L-1)*Fs/L;
stem(w(1:L/2), abs(ufft(1:L/2)))     %Plot hasta frecuencia Tsw
title('Single-sided amplitude spectrum of multisine signal')
xlabel('Frequency (rad/sec)')
ylabel('Amplitude')
```

Las gráficas que resultan del programa anterior se muestran en la Figura 26 y Figura 27. La Figura 26, presenta la forma de onda resultado de una señal de 12 componentes que se repite en 3 ciclos. Cada ciclo toma 100 muestras a un periodo de muestreo de 0.1 segundos. Todas las sinusoides tienen la misma cantidad de energía como se puede apreciar en la Figura 27, que representa el espectro de la señal, y sus frecuencias son aleatorias.

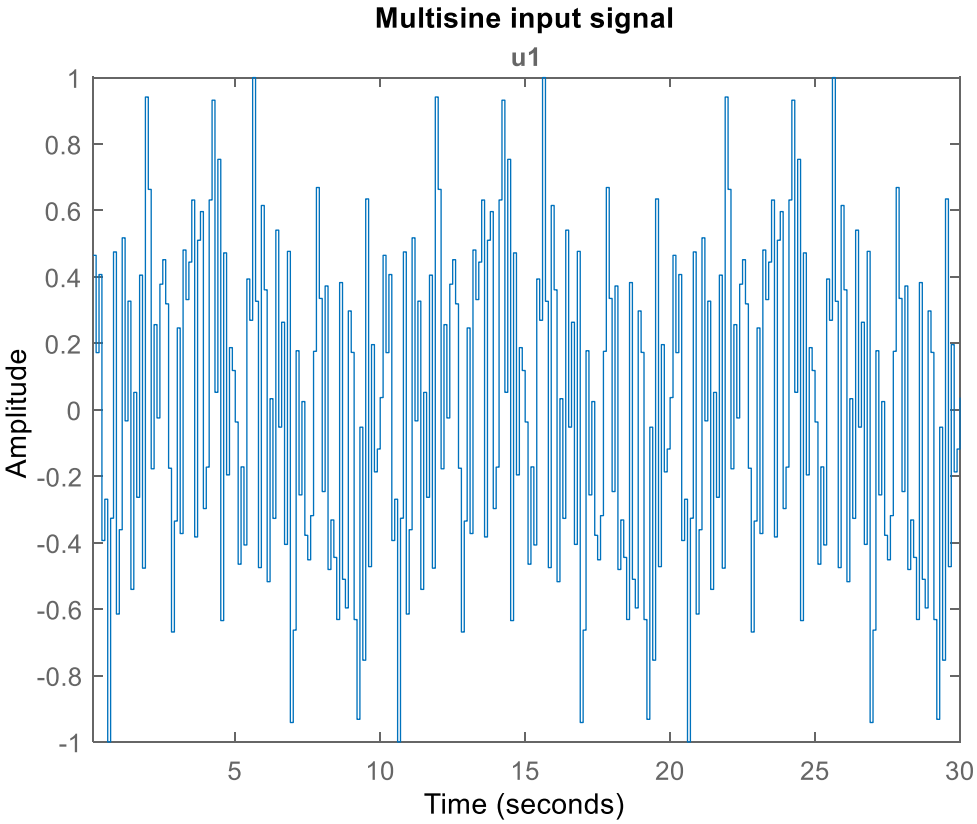

Figura 26: Señal multiseno de 12 componentes

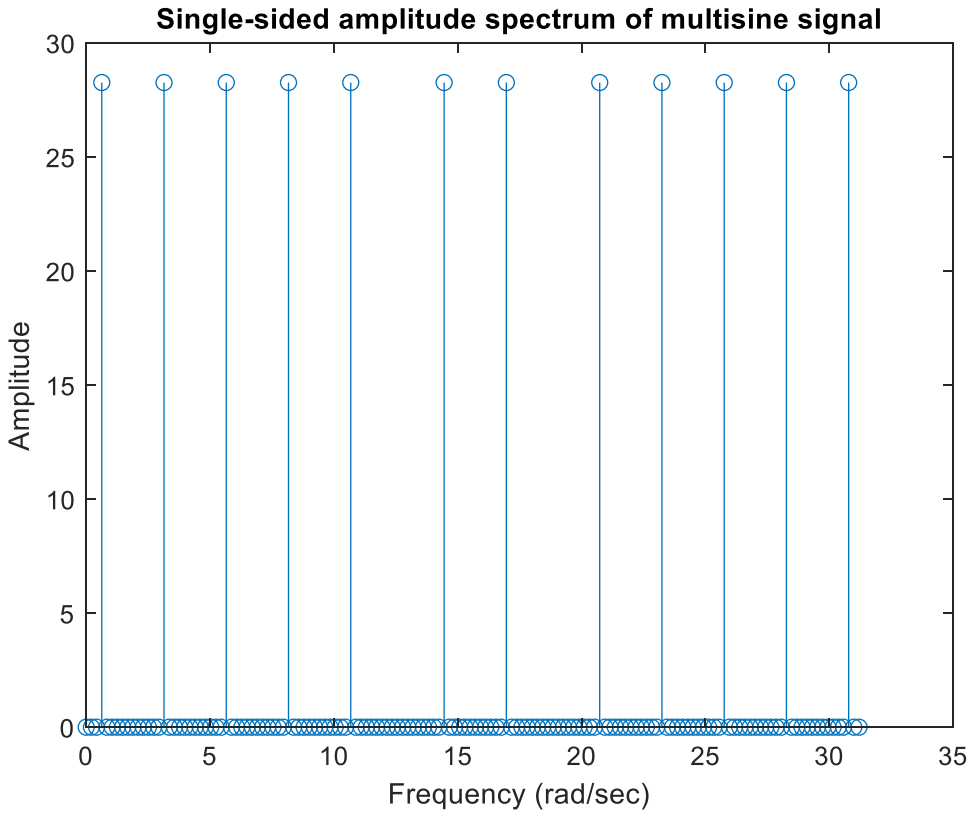

Figura 27: espectro de la señal multiseno

4.2.2 Definición de la señal multiseno en el dominio del tiempo

Una de las primeras opciones para definir la señal multiseno, es hacerlo en el dominio del tiempo como una suma de señales sinusoidales tal y como se expresa en la ecuación [120]. Las frecuencias y las fases de las señales se pueden obtener de forma determinística o aleatoria. Las ecuaciones de Schroeder (Schroeder, 1970) y Newman (Newman, 1965) para las fases de las sinusoides, garantizan minimizar el factor de cresta de la señal.

Si se genera una señal con sus frecuencias equidistantemente distribuidas, la función necesaria en Matlab será:

```
function [U, t] = generateMultisine(Sr, length, ns, banda, Np)
%GenerateMultiSine is a function to generate multisine signal
%   Sr: sampling rate in seconds
%   lenght: número total de muestras
%   ns: número de sinusoides
%   banda: frecuencia inicial y final
%   Np: Número de ciclos completos
%   U => excitation signal in frequency domain, single sided
%
%

   k = (1:ns);
   phase = (pi*k.*(k-1))/ns;
   U=zeros(1,length*Np);
   t = linspace(0,Np/Sr, length*Np);
   init = 1;
   fin = length;
   for j = 1:Np
       for i = k
           U(init:fin) = U(init:fin) + sin(2*pi*i*t(1:length)/(1/Sr) +
phase(i))/ns;
       end
       init = init + length;
       fin = fin + length;
   end

end
```

En este caso la línea encargada de generar la suma de sinusoides será

```
U(init:fin) = U(init:fin) + sin(2*pi*i*t(1:length)/(1/Sr) + phase(i))/ns;
```

Que va sumando por tramos de la señal el seno de cada una de las componentes, varible i con la fase generada con el algoritmo de Schroeder

```
phase = (pi*k.*(k-1))/ns;
```

El resultado obtenido para un periodo de muestreo de 0.1 segundos, 100 muestras por ciclo, 12 componente para la señal y 3 repeticiones de cada ciclo, se puede ver en la Figura 28 y Figura 29.

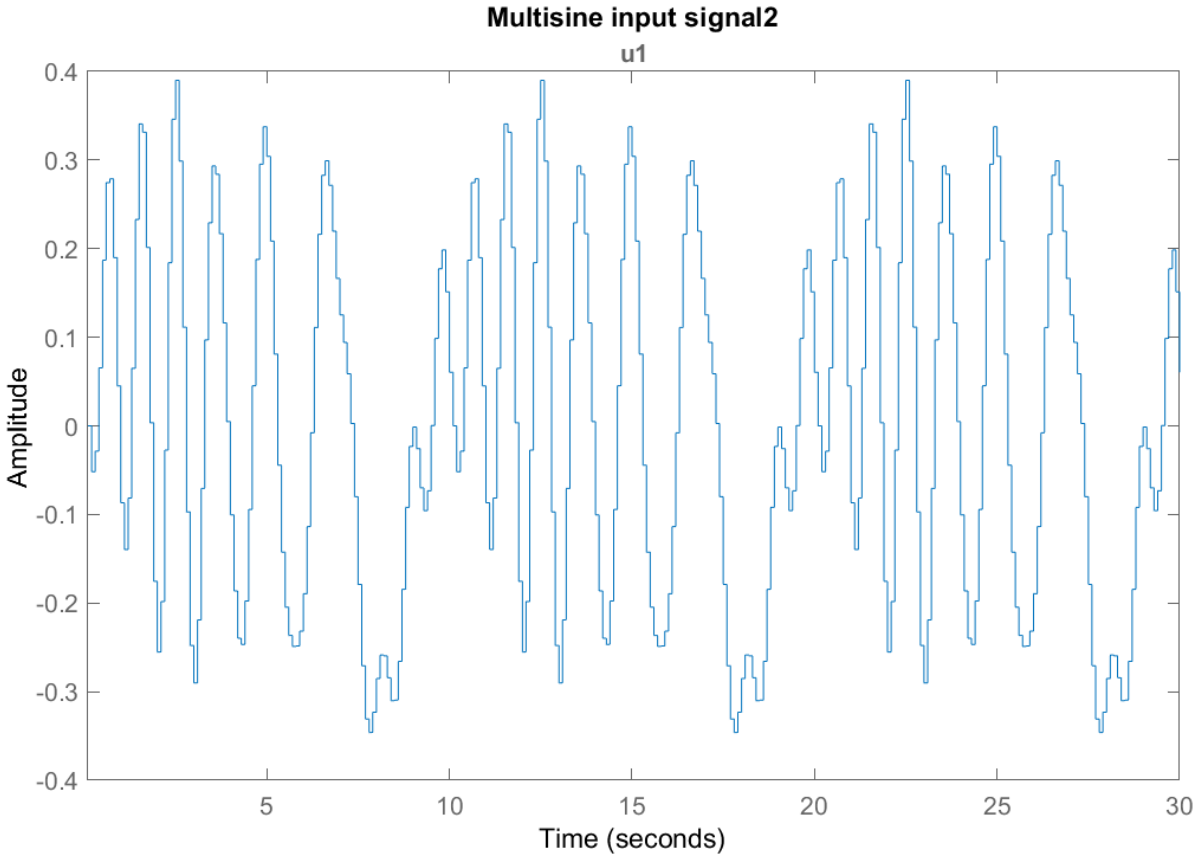

Figura 28: Señal multiseno de frecuencias determinísticas

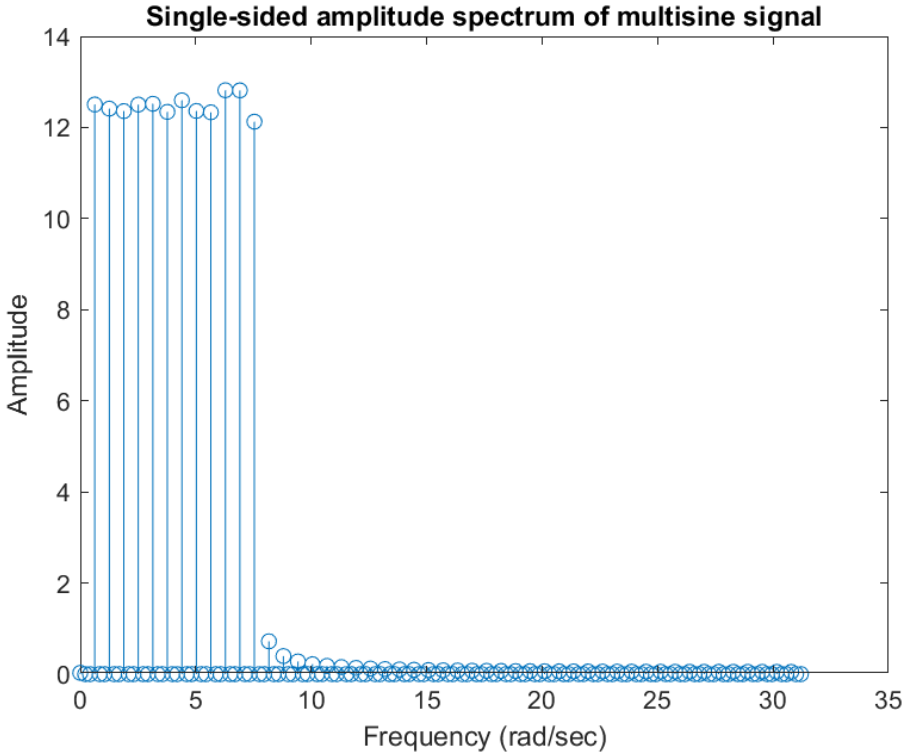

Figura 29: Respuesta en frecuencias de la señal multiseno

Del resultado obtenido en la Figura 29, se puede apreciar que no coincide con el espectro calculado en el apartado 3.8.3. La diferencia estriba en que en Matlab la señal generada es discreta y los errores en la amplitud de los armónicos en el espectro tienen que ver con la frecuencia de discretización.

Si ahora se genera una señal similar a la anterior, pero con frecuencias aleatorias dentro de un determinado ancho de banda, el programa a utilizar será:

```
function [U, t] = generateMultisine(Sr,length, ns, banda, Np)
%GenerateMultiSine is a function to generate multisine signal
%   Sr: sampling rate in seconds
%   lenght: número total de muestras
%   ns: número de sinusoides
%   banda: frecuencia inicial y final
%   Np: Número de ciclos completos
%   U => excitation signal in frequency domain, single sided
%
%

   k = (1:ns);
   phase = (pi*k.*(k-1))/ns;
   f = randsample(banda,ns);
   U=zeros(1,length*Np);
   t = linspace(0,Np/Sr, length*Np);
   init = 1;
   fin = length;
   for j = 1:Np
       for i = k
           U(init:fin) = U(init:fin) + sin(2*pi*f(i)*t(1:length)/(1/Sr) +
phase(i))/ns;
       end
       init = init + length;
       fin = fin + length;
   end

end
```

En este caso se ha añadido la generación de un vector de frecuencias aleatorias dentro del ancho de banda definido por la variable *banda*.

```
      f = randsample(banda,ns);
```

El código que genera las señales de frecuencias aleatorias es muy similar al del caso anterior

```
U(init:fin) = U(init:fin) + sin(2*pi*f(i)*t(1:length)/(1/Sr) + phase(i))/ns;
```

El resultado del algoritmo puede apreciarse en la Figura 30 y Figura 31. El ancho de banda obtenido es más complejo que en el caso anterior visto en la Figura 29, en que se representaba un filtro pasa bajos. Una de las características deseables en esta señal que no ha conseguido el algoritmo anterior es que presente el mismo contenido de energía para todas las frecuencias fundamentales.

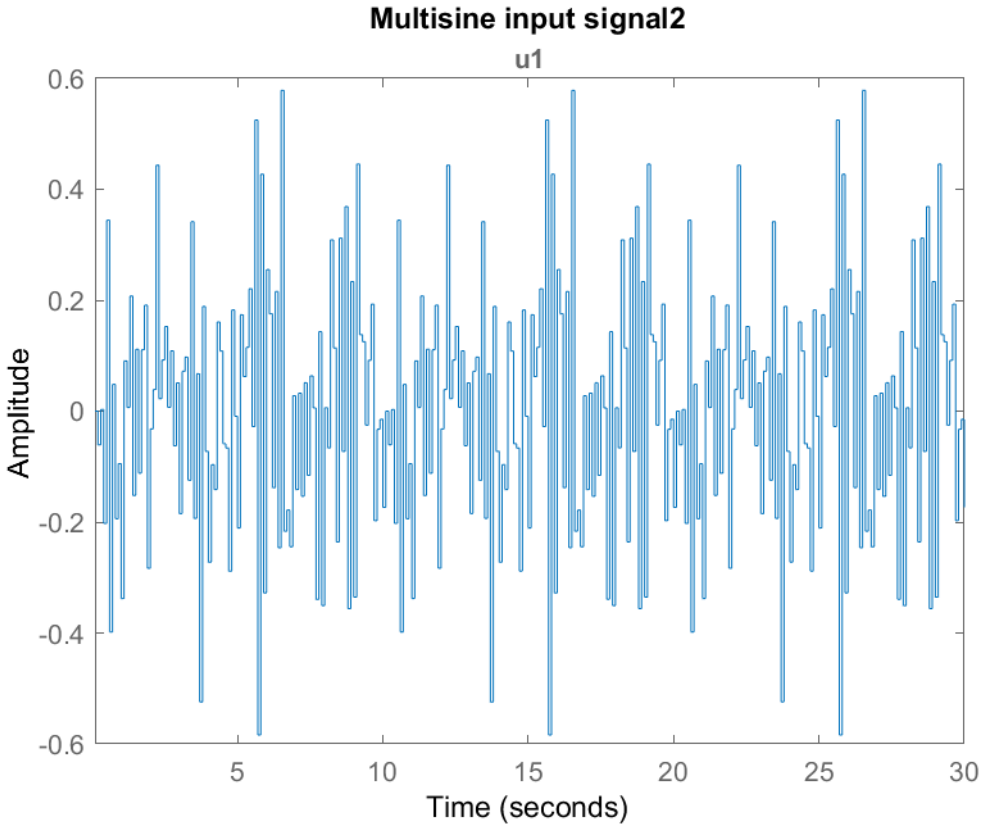

Figura 30: Señal multiseno de frecuencias aleatorias

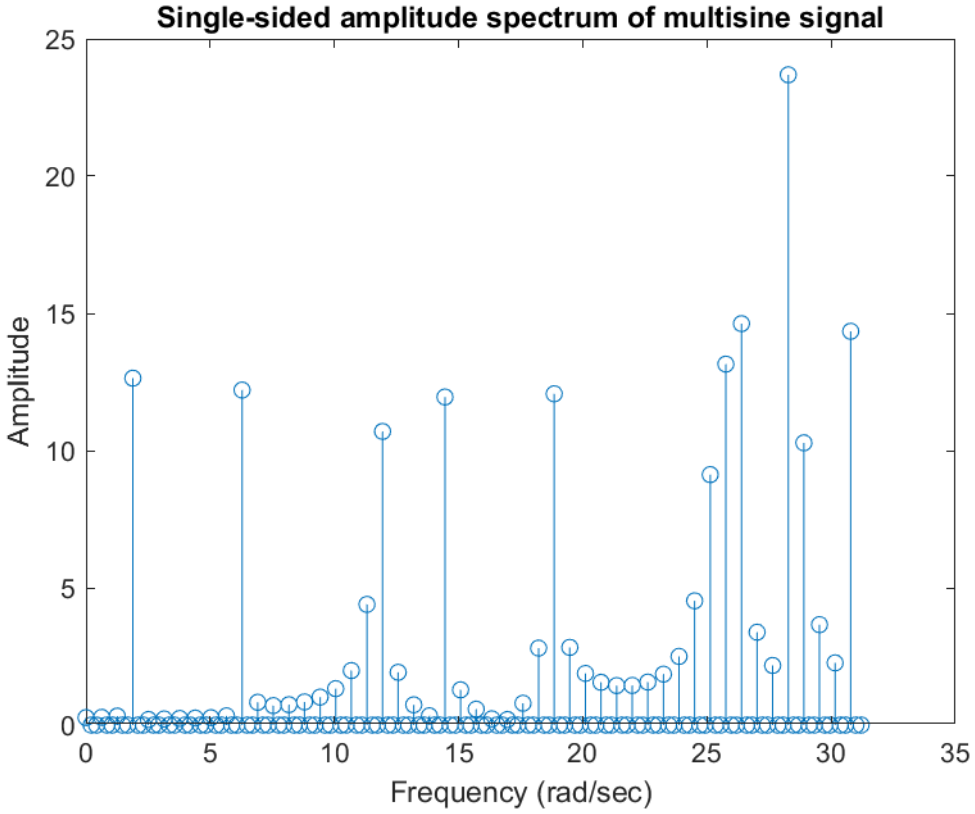

Figura 31: Respuesta en frecuencias de la señal multiseno

El programa principal usado para ejecutar las dos funciones anteriores, se presenta a continuación:

```
%Inicialización
clear all
close all

Ts = 0.1;            %Periodo de muestreo in segundos
Samples = 100;       %Número de muestras por periodo
Np = 3;              %Número de periodos

Tsw = 0.01;             %Periodo de la señal
Banda=[0,1/Tsw];     %Banda de frecuencias
NumSinusoids = 12;   %Numero de sinusoides generadas

[u2, t] = generateMultisine(Samples/Ts, Samples, NumSinusoids, 0:1/Tsw, Np);
t = linspace(0,Samples*Np*Ts,Samples*Np);
figure(4);
u = iddata([],transpose(u2),Ts,'Timeunit','seconds');
plot(u);
title('Multisine input signal2')

%Espectro de la señal
figure(5);
ufft = fft(u2);       %Transformada rapida de fourier
Fs = 2*pi/Ts;        %Frecuencia de muestreo en radianes por segundo
L = length(u2);
w = (0:L-1)*Fs/L;
stem(w(1:L/2), abs(ufft(1:L/2)))     %Plot hasta frecuencia Tsw
title('Single-sided amplitude spectrum of multisine signal')
xlabel('Frequency (rad/sec)')
ylabel('Amplitude')
```

4.2.3 Generación de señales multiseno usando Sysquake

Para la generación de señales multiseno con Sysquake, se va a escoger como en el caso de Matlab la generación de la señal en el dominio del tiempo y que las frecuencias de la señal estén armónicamente distribuidas. La fase de la señal se podrá escoger nula, aleatoria o según el algoritmo de Schroeder.

La función utilizada para generar la señal será:

```
%%-------------------------------------------------
function (msine, time) = gen_msine(F, a, dphase, N, T, Nc);
%gen_msine

      f=[1:F]; % Harmonics

      %Amplitude
      A=a*ones(1,F); %All sinusoids have equal amplitude

      %Sampling time:
      Ts=T/N;

      omega=1:F;
      omega=omega'*2*pi/N/Ts;       %Frequency of sinusoids
```

```
switch dphase                    %depends on the phase election
case 1 %zeros phases
      phi=zeros(1,F);

case 2 %random phases
      phi=2*pi*rand(1,F);

case 3 %Schroeder phases
      phi(1)=0;
      pk=abs(A.^2)/sum(abs(A.^2));
      for k=2:F
            phi(k)=-2*pi*[k-1:-1:1]*[pk(1:k-1)]'; %Schoroeder formula
            %phi(k)=pi*((k-1)^2)/F;              % Newman formula
      end
end

A=A(:); % be sure column vector
phi=phi(:);

for h=1:N
      u(h,1)=sum(A.*cos(omega.*h*Ts+phi));
end

msine=repmat(u,Nc,1);  % then repeat it, nc times

t=0:Ts:N*Ts*Nc-Ts;  % create the corresponding time vector
time=t'; % transpose the vector, to be tidy
```

Donde *T* es el periodo de muestreo, *F* es el número total de armónicos de la señal, *N* es el número de muestras. *Nc* será el número de veces que se repetirá la señal. El parámetro *a*, representa la amplitud de cada sinusoide y *dphase* la elección del usuario para calcular la fase inicial de las sinusoides. La respuesta de la función viene en el vector de amplitudes *msine* y el vector de tiempo *time*.

La señal resultante será:

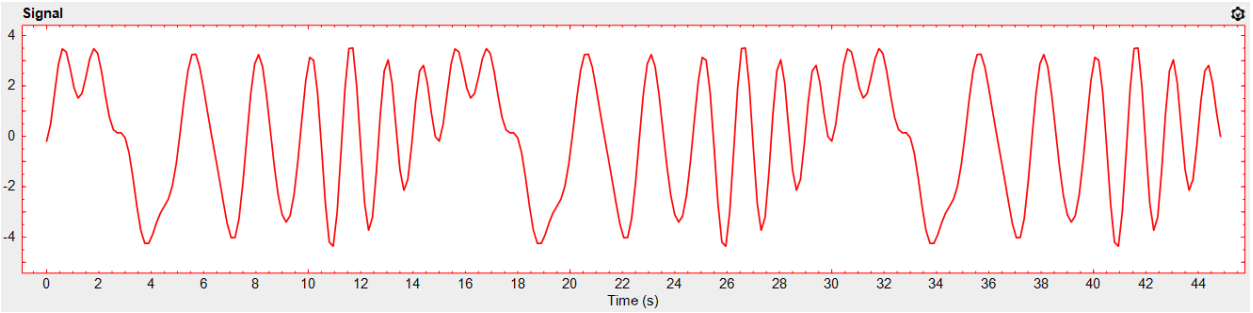

Figura 32: Señal multiseno generada con Sysquake

En la señal mostrada en la Figura 28 se obtenía una respuesta similar con la principal diferencia de que en Sysquake la señal se ha representado como continua y en Matlab la señal se mostraba atravesando un retenedor de orden cero que la distorsionaba.

En el caso del espectro de la señal, el resultado obtenido por Sysquake es el representado en la Figura 33:

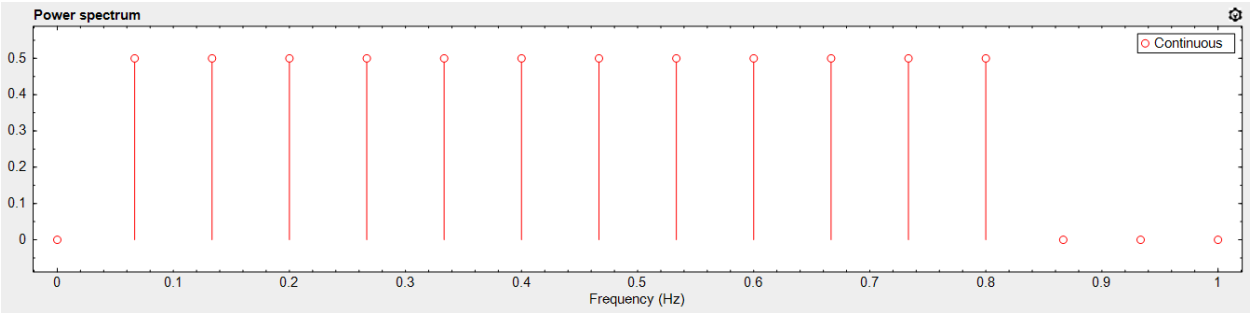

Figura 33: Espectro de la señal multiseno en Sysquake

4.3 Estructura del programa en Sysquake

La programación en Sysquake es una combinación del lenguaje de scripts usado en Matlab con algunas estructuras propias de Sysquake. Antes de comenzar a programar se debe definir la estructura y organización del código que se va a escribir, la ingeniería del software es la rama de la ciencia encargada de su definición y describe el funcionamiento del programa a través de una serie de diagramas normalizados y la documentación asociada a los mismos.

4.3.1 Diagramas de casos de uso

Los diagramas de casos de uso permiten descubrir los requisitos del sistema y la relación entre el sistema y sus usuarios. La documentación de cada caso de uso especifica las funcionalidades que realiza el caso de uso, en qué momento se puede ejecutar y los efectos que sobre el sistema que genera su ejecución.

4.3.1.1 Nivel 1

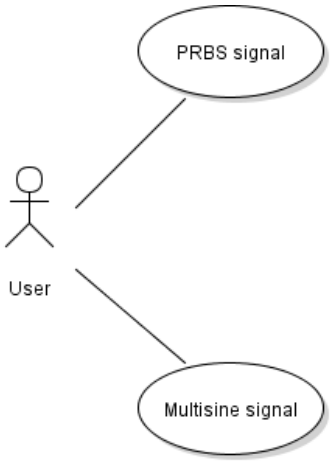

Figura 34: Diagrama de casos de uso principal.

Tabla 7: Caso de uso, PRBS signal.

Caso de uso:	**1.1 PRBS signal.**
Dependencias:	Ninguna.
Precondiciones:	Ninguna.
Descripción:	El usuario selecciona en el menú la generación de señales de tipo PRBS.
Secuencia normal:	1. En el menú *signal* se selecciona la opción *PRBS.* 2. Se establece la amplitud por defecto de la señal a 1. 3. Se establece la longitud del registro de desplazamiento a 4 bits. 4. Se establece el periodo de la señal de reloj a 1. 5. Se establece el número de ciclos de la señal a 1. 6. Se actualizan los valores del panel de información. 7. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro.
Excepciones:	Ninguna.

Tabla 8: Caso de uso, multisine signal.

Caso de uso:	**1.2 Multisine signal.**
Dependencias:	Ninguna.
Precondiciones:	Ninguna.
Descripción:	El usuario selecciona en el menú la generación de señales de tipo Multiseno.
Secuencia normal:	1. En el menú *signal* se selecciona la opción *multisine.* 2. Se selecciona el algoritmo de fase a Schroeder. 3. Se establece la amplitud por defecto de la señal a 1. 4. Se establece el número de sinusoides armónicas a 10. 5. Se establece la longitud de la secuencia a 100 muestras. 6. Se establece el periodo de la señal de reloj a 1. 7. Se establece el número de ciclos de la señal a 1. 8. Se actualizan los valores del panel de información. 9. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro.
Excepciones:	Ninguna.

4.3.1.2 Nivel 2

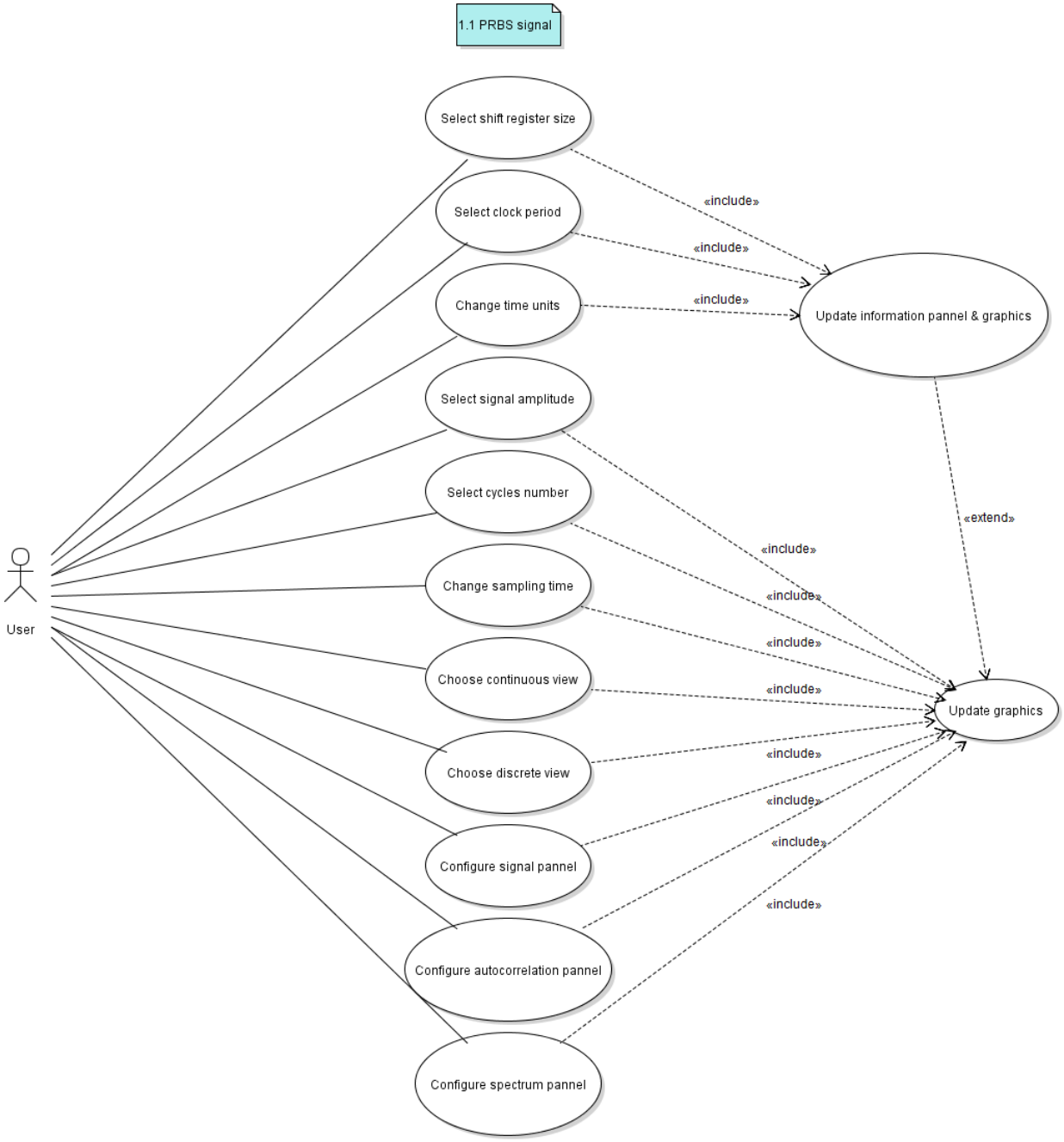

Figura 35: Casos de uso nivel 2, PRBS signal.

Tabla 9: Caso de uso, select shift register size.

Caso de uso:	**2.1.1 Select shift register size.**
Dependencias:	2.1.12 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona el número de bits que tendrá el registro de desplazamiento con el que se genera la señal PRBS.
Secuencia normal:	1. En la pantalla principal se introduce un nuevo número de bits para el registro de desplazamiento o se aumenta o disminuye el número de bits en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno del número de bits del registro de desplazamiento. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro con el nuevo número de bits para el registro de desplazamiento.
Excepciones:	1. Si se detecta un valor fuera de rango, se genera un cuadro de dialogo de error. Límite entre 2 y 11. 2. No se actualiza ninguna variable ni gráfico.

Tabla 10: Caso de uso, select clock period.

Caso de uso:	**2.1.2 Select clock period.**
Dependencias:	2.1.12 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona el número de ciclos de la señal de reloj con el que se genera la señal PRBS.
Secuencia normal:	1. En la pantalla principal se introduce un nuevo número de ciclos de la señal de reloj o se aumenta o disminuye el número de ciclos en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno del número ciclos de la señal de reloj. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro con el nuevo número de ciclos de la señal de reloj.
Excepciones:	1. Si se detecta un valor fuera de rango, se genera un cuadro de dialogo de error. Límite entre 1 y 100. 2. No se actualiza ninguna variable ni gráfico.

Tabla 11: Caso de uso, change time units.

Caso de uso:	**2.1.3 Change time units.**
Dependencias:	2.1.12 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona la unidad de tiempo básica con el que se genera la señal PRBS desde el menú *Time units*. La unidad de tiempo se puede fijar desde minutos a microsegundos.
Secuencia normal:	1. En el menú *Time units* se selecciona una nueva unidad de tiempo. 2. Se actualiza el valor interno de la unidad de tiempo utilizada. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro con el nuevo parámetro de unidad de tiempo.
Excepciones:	Ninguna.

Tabla 12: Caso de uso, select signal amplitude.

Caso de uso:	**2.1.4 Select signal amplitude.**
Dependencias:	2.1.13 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona la nueva amplitud para la señal PRBS. La señal es siempre simétrica con respecto a cero. Solo se admiten valores positivos.
Secuencia normal:	1. En la pantalla principal se introduce un nuevo valor para la amplitud de la señal o se aumenta o disminuye la amplitud en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno de la amplitud de la señal. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro con la nueva amplitud.
Excepciones:	1. Si se introduce un valor negativo, se toma su valor absoluto. 2. Se sigue el flujo normal del caso de uso.

Tabla 13: Caso de uso, select cycles number.

Caso de uso:	**2.1.5 Select cycles number.**
Dependencias:	2.1.13 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona el nuevo valor de ciclos de la señal PRBS. La señal se repetirá de forma idéntica el número de veces indicado por este control.
Secuencia normal:	1. En la pantalla principal se introduce un nuevo valor para el número de ciclos de la señal o se aumenta o disminuye el número de ciclos en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno del número de ciclos. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro con el nuevo número de ciclos repetidos.
Excepciones:	1. Si la combinación del tamaño de registro, número de ciclos de reloj y número de ciclos excede los 10000 valores aparece un dialogo de error. 2. No se actualiza ninguna variable ni el gráfico.

Tabla 14: Caso de uso, change sampling time

Caso de uso:	**2.1.6 Change sampling time.**
Dependencias:	2.1.13 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona el número de muestras de la señal PRBS por unidad de tiempo.
Secuencia normal:	1. Desde el menú se escoge la opción *Sampling time* que permite escoger el nuevo periodo de muestreo. El menú mostrará un desplegable donde se incluirá la información del periodo de muestreo, la frecuencia a Nyquist y el número total de muestras de la señal PRBS. El desplegable permite introducir el valor numérico del número de muestras por ciclo de reloj o dispondrá de un cursor que aumenta o disminuye el número de muestras en función del movimiento en la barra de desplazamiento. 2. El usuario pulsa el botón OK. 3. Se actualiza el valor interno del periodo de muestreo. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro con el nuevo periodo de muestreo.
Excepciones:	1. Si el número de ciclos no es múltiplo de 2 o 5 aparece un cuadro de dialogo de error. 2. Si el usuario pulsa el botón cancel se anulan las acciones previas. 3. No se actualiza ninguna variable ni el gráfico.

Tabla 15: Caso de uso, choose continuous view

Caso de uso:	**2.1.7 Choose continuous view.**
Dependencias:	2.1.13 Update graphics. 2.1.8 Choose discrete view
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario escoge mostrar la señal de forma continua.
Secuencia normal:	1. Desde el menú *View* se escoge la opción *Continuous* que hace que la señal se vea de forma continua. Esta acción es compatible con escoger de forma simultánea la visualización de forma discreta de la señal. 2. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro de forma continua.
Excepciones:	Ninguna.

Tabla 16: Caso de uso, choose discrete view

Caso de uso:	**2.1.8 Choose discrete view.**
Dependencias:	2.1.13 Update graphics. 2.1.7 Choose continuous view
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario escoge mostrar la señal de forma continua.
Secuencia normal:	1. Desde el menú *View* se escoge la opción *Continuous* que hace que la señal se vea de forma continua. Esta acción es compatible con escoger de forma simultánea la visualización de forma discreta de la señal. 2. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal PRBS, su función de autocorrelación y su espectro de forma continua.
Excepciones:	Ninguna.

Tabla 17: Caso de uso, configure signal pannel

Caso de uso:	**2.1.9 Configure signal pannel.**
Dependencias:	2.1.13 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona el control gráfico que permite acceder a la configuración del panel que muestra la señal PRBS.
Secuencia normal:	1. Desde el submenú de configuración del panel se modifica la configuración del gráfico. 2. El usuario pulsa el botón OK. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	El gráfico de la señal PRBS queda modificado.
Excepciones:	1. Si el usuario pulsa el botón cancel se anulan las modificaciones. 2. No se actualiza ninguna variable ni el gráfico.

Tabla 18: Caso de uso, configure autocorrelation pannel

Caso de uso:	**2.1.10 Configure autocorrelation pannel.**
Dependencias:	2.1.13 Update graphics. 2.1.11 Configure spectrum pannel
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona el control gráfico que permite acceder a la configuración del panel que muestra la función de autocorrelación de la señal PRBS.
Secuencia normal:	1. Desde el submenú de configuración del panel se modifica la configuración del gráfico. 2. El usuario pulsa el botón OK. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	El gráfico de autocorrelación de la señal PRBS queda modificado.
Excepciones:	1. Si el usuario pulsa el botón cancel se anulan las modificaciones. 2. Si el usuario escoge ver el panel de espectro más grande el gráfico de autocorrelación se oculta. 3. No se actualiza ninguna variable ni el gráfico.

Tabla 19: Caso de uso, configure spectrum pannel

Caso de uso:	**2.1.11 Configure spectrum pannel.**
Dependencias:	2.1.13 Update graphics. 2.1.11 Configure correlation pannel
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario selecciona el control gráfico que permite acceder a la configuración del panel que muestra el espectro de potencia de la señal PRBS.
Secuencia normal:	1. Desde el submenú de configuración del panel se modifica la configuración del gráfico. 2. El usuario pulsa el botón OK. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	El gráfico de espectro de potencia de la señal PRBS queda modificado.
Excepciones:	1. Si el usuario pulsa el botón cancel se anulan las modificaciones. 2. Si el usuario escoge ver el panel de espectro más grande el gráfico de autocorrelación se oculta. 3. No se actualiza ninguna variable ni el gráfico.

Tabla 20: Caso de uso, update information pannel & graphics

Caso de uso:	**2.1.12 Update information pannel & graphics.**
Dependencias:	2.1.13 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario ha realizado alguna modificación en los parámetros de la señal PRBS y esto fuerza un cambio en la visualización del panel de información y de forma simultánea en los gráficos de la señal.
Secuencia normal:	1. Se actualiza el panel de información 2. Se actualizan las gráficas visualizadas.
Postcondiciones:	El panel de información y los gráficos de la señal PRBS quedan modificados.
Excepciones:	Ninguna

Tabla 21: Caso de uso, update graphics

Caso de uso:	**2.1.13 Update graphics.**
Dependencias:	2.1.12 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal PRBS.
Descripción:	El usuario ha realizado alguna modificación en los parámetros de la señal PRBS y esto fuerza un cambio en la visualización de los gráficos de la señal.
Secuencia normal:	1. Se actualizan las gráficas visualizadas.
Postcondiciones:	Los gráficos de la señal PRBS quedan modificados.
Excepciones:	Ninguna

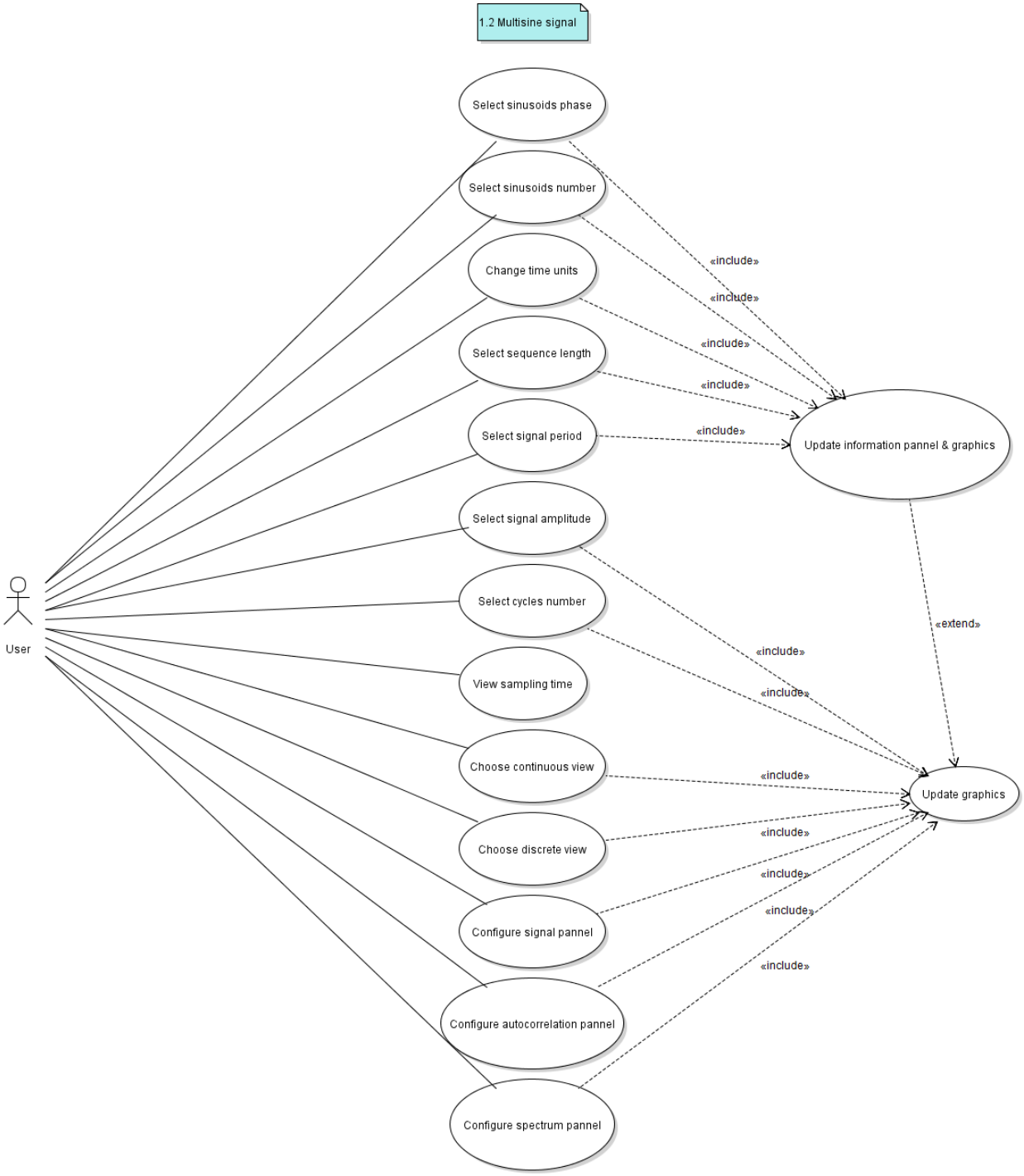

Figura 36: Casos de uso nivel 2 Multisine signal

Tabla 22: Caso de uso, select sinusoids phase.

Caso de uso:	**2.2.1 Select sinusoids phase.**
Dependencias:	2.2.14 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona el algoritmo que se utilizará para generar la fase inicial de la señal multiseno. Habrá tres opciones: fase cero, fase aleatoria y fase siguiendo el algoritmo de Schroeder.
Secuencia normal:	1. En la pantalla principal el usuario selecciona uno de los tres algoritmos a utilizar para el cálculo inicial de la fase de la señal multiseno. 2. Se actualiza la variable que almacena el tipo de algoritmo a utilizar. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro con el algoritmo seleccionado para las fases de las sinusoides.
Excepciones:	Ninguna.

Tabla 23: Caso de uso, select sinusoids number.

Caso de uso:	**2.2.2 Select sinusoids number.**
Dependencias:	2.2.14 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona el número de sinusoides con el que se genera la señal multiseno.
Secuencia normal:	1. En la pantalla principal se introduce un nuevo número de sinusoides de la señal o se aumenta o disminuye el número de sinusoides en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno del número de sinusoides de la señal. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro con el nuevo número de sinusoides.
Excepciones:	1. Si se detecta un valor fuera de rango, se genera un cuadro de dialogo de error. Límite establecido estará entre 1 y la longitud de la secuencia partido por 2. 2. No se actualiza ninguna variable ni gráfico.

Tabla 24: Caso de uso, change time units.

Caso de uso:	**2.2.3 Change time units.**
Dependencias:	2.2.14 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona la unidad de tiempo básica con el que se genera la señal multiseno desde el menú *Time units*. La unidad de tiempo se puede fijar desde minutos a microsegundos.
Secuencia normal:	1. En el menú *Time units* se selecciona una nueva unidad de tiempo. 2. Se actualiza el valor interno de la unidad de tiempo utilizada. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro con el nuevo parámetro de unidad de tiempo.
Excepciones:	Ninguna.

Tabla 25: Caso de uso, change sequence length

Caso de uso:	**2.2.4 Change sequence length.**
Dependencias:	2.2.14 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona la nueva longitud de la secuencia. La longitud de la secuencia debe ser siempre mayor que el doble del número de sinusoides. Seleccionando el periodo de la señal y el número de muestras queda determinado el periodo de muestreo de la señal.
Secuencia normal:	1. En la pantalla principal el usuario introduce una nueva longitud de la secuencia o se aumenta o disminuye su longitud en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno de la longitud de la secuencia de la señal multiseno. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro ajustada al nuevo parámetro de longitud de la secuencia.
Excepciones:	1. Si se detecta un valor fuera de rango, se genera un cuadro de dialogo de error. El límite establecido estará entre el doble del número de sinusoides más 1 y 2000. 2. No se actualiza ninguna variable ni gráfico.

Tabla 26: Caso de uso, select signal period

Caso de uso:	**2.2.5 Select signal period.**
Dependencias:	2.2.14 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona el periodo de la señal en unidades de la unidad de tiempo escogida. Seleccionando el periodo de la señal y el número de muestras queda determinado el periodo de muestreo de la señal.
Secuencia normal:	1. En la pantalla principal el usuario introduce el nuevo periodo de la señal en unidades de tiempo o se aumenta o disminuye el periodo en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno del periodo de la señal multiseno. 3. Se actualizan los valores del panel de información. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro ajustada al nuevo parámetro de periodo de la señal.
Excepciones:	1. Si se detecta un valor fuera de rango, se genera un cuadro de dialogo de error. El límite establecido estará entre 1 y 1000. 2. No se actualiza ninguna variable ni gráfico.

Tabla 27: Caso de uso, select signal amplitude.

Caso de uso:	**2.2.6 Select signal amplitude.**
Dependencias:	2.2.15 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona la nueva amplitud para la señal multiseno. La señal es siempre simétrica con respecto a cero. Solo se admiten valores positivos.
Secuencia normal:	1. En la pantalla principal se introduce un nuevo valor para la amplitud de la señal o se aumenta o disminuye la amplitud en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno de la amplitud de la señal. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro con la nueva amplitud.
Excepciones:	1. Si se introduce un valor negativo, se toma su valor absoluto. 2. Se sigue el flujo normal del caso de uso.

Tabla 28: Caso de uso, select cycles number.

Caso de uso:	**2.2.7 Select cycles number.**
Dependencias:	2.2.15 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona el nuevo valor de ciclos de la señal multiseno. La señal se repetirá de forma idéntica el número de veces indicado por este control.
Secuencia normal:	1. En la pantalla principal se introduce un nuevo valor para el número de ciclos de la señal o se aumenta o disminuye el número de ciclos en función del movimiento en la barra de desplazamiento. 2. Se actualiza el valor interno del número de ciclos. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro con el nuevo número de ciclos repetidos.
Excepciones:	1. Si se detecta un valor fuera de rango, se genera un cuadro de dialogo de error. El límite establecido estará entre 1 y 100. 2. No se actualiza ninguna variable ni el gráfico.

Tabla 29: Caso de uso, view sampling time

Caso de uso:	**2.2.8 View sampling time.**
Dependencias:	Ninguno.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario visualiza el número de muestras de la señal multiseno por unidad de tiempo. Para la señal multiseno, el periodo de muestreo queda determinado al seleccionar el número de muestras y el periodo de la señal.
Secuencia normal:	1. Desde el menú se escoge la opción *Sampling time* que permite visualizar el periodo de muestreo. El menú mostrará un desplegable donde se incluirá la información del periodo de muestreo, la frecuencia a Nyquist y el número total de muestras de la señal multiseno. 2. El usuario pulsa el botón OK. 3. Se actualiza el valor interno del periodo de muestreo. 4. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro con el nuevo periodo de muestreo.
Excepciones:	Ninguna.

Tabla 30: Caso de uso, choose continuous view

Caso de uso:	**2.2.9 Choose continuous view.**
Dependencias:	2.2.15 Update graphics. 2.2.10 Choose discrete view
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario escoge mostrar la señal de forma continua.
Secuencia normal:	1. Desde el menú *View* se escoge la opción *Continuous* que hace que la señal se vea de forma continua. Esta acción es compatible con escoger de forma simultánea la visualización de forma discreta de la señal. 2. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro de forma continua.
Excepciones:	Ninguna.

Tabla 31: Caso de uso, choose discrete view

Caso de uso:	**2.2.10 Choose discrete view.**
Dependencias:	2.2.15 Update graphics. 2.2.9 Choose continuous view
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario escoge mostrar la señal de forma continua.
Secuencia normal:	1. Desde el menú *View* se escoge la opción *Continuous* que hace que la señal se vea de forma continua. Esta acción es compatible con escoger de forma simultánea la visualización de forma discreta de la señal. 2. Se actualizan las gráficas visualizadas.
Postcondiciones:	La interface de usuario se ajusta para mostrar una señal multiseno, su función de autocorrelación y su espectro de forma continua.
Excepciones:	Ninguna.

Tabla 32: Caso de uso, configure signal pannel

Caso de uso:	**2.2.11 Configure signal pannel.**
Dependencias:	2.2.15 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona el control gráfico que permite acceder a la configuración del panel que muestra la señal multiseno.
Secuencia normal:	1. Desde el submenú de configuración del panel se modifica la configuración del gráfico. 2. El usuario pulsa el botón OK. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	El gráfico de la señal multiseno queda modificado.
Excepciones:	1. Si el usuario pulsa el botón cancel se anulan las modificaciones. 2. No se actualiza ninguna variable ni el gráfico.

Tabla 33: Caso de uso, configure autocorrelation pannel

Caso de uso:	**2.2.12 Configure autocorrelation pannel.**
Dependencias:	2.2.15 Update graphics. 2.2.13 Configure spectrum pannel
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona el control gráfico que permite acceder a la configuración del panel que muestra la función de autocorrelación de la señal multiseno.
Secuencia normal:	1. Desde el submenú de configuración del panel se modifica la configuración del gráfico. 2. El usuario pulsa el botón OK. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	El gráfico de autocorrelación de la señal multiseno queda modificado.
Excepciones:	1. Si el usuario pulsa el botón cancel se anulan las modificaciones. 2. Si el usuario escoge ver el panel de espectro más grande el gráfico de autocorrelación se oculta. 3. No se actualiza ninguna variable ni el gráfico.

Tabla 34: Caso de uso, configure spectrum pannel

Caso de uso:	**2.2.13 Configure spectrum pannel.**
Dependencias:	2.2.15 Update graphics. 2.2.12 Configure correlation pannel
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario selecciona el control gráfico que permite acceder a la configuración del panel que muestra el espectro de potencia de la señal multiseno.
Secuencia normal:	1. Desde el submenú de configuración del panel se modifica la configuración del gráfico. 2. El usuario pulsa el botón OK. 3. Se actualizan las gráficas visualizadas.
Postcondiciones:	El gráfico de espectro de potencia de la señal multiseno queda modificado.
Excepciones:	1. Si el usuario pulsa el botón cancel se anulan las modificaciones. 2. Si el usuario escoge ver el panel de espectro más grande el gráfico de autocorrelación se oculta. 3. No se actualiza ninguna variable ni el gráfico.

Tabla 35: Caso de uso, update information pannel & graphics

Caso de uso:	**2.2.14 Update information pannel & graphics.**
Dependencias:	2.2.15 Update graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario ha realizado alguna modificación en los parámetros de la señal multiseno y esto fuerza un cambio en la visualización del panel de información y de forma simultánea en los gráficos de la señal.
Secuencia normal:	1. Se actualiza el panel de información 2. Se actualizan las gráficas visualizadas.
Postcondiciones:	El panel de información y los gráficos de la señal multiseno quedan modificados.
Excepciones:	Ninguna

Tabla 36: Caso de uso, update graphics

Caso de uso:	**2.2.15 Update graphics.**
Dependencias:	2.2.14 Update information pannel & graphics.
Precondiciones:	Se ha seleccionado previamente la señal multiseno.
Descripción:	El usuario ha realizado alguna modificación en los parámetros de la señal multiseno y esto fuerza un cambio en la visualización de los gráficos de la señal.
Secuencia normal:	1. Se actualizan las gráficas visualizadas.
Postcondiciones:	Los gráficos de la señal multiseno quedan modificados.
Excepciones:	Ninguna

4.3.1.3 Nivel 3

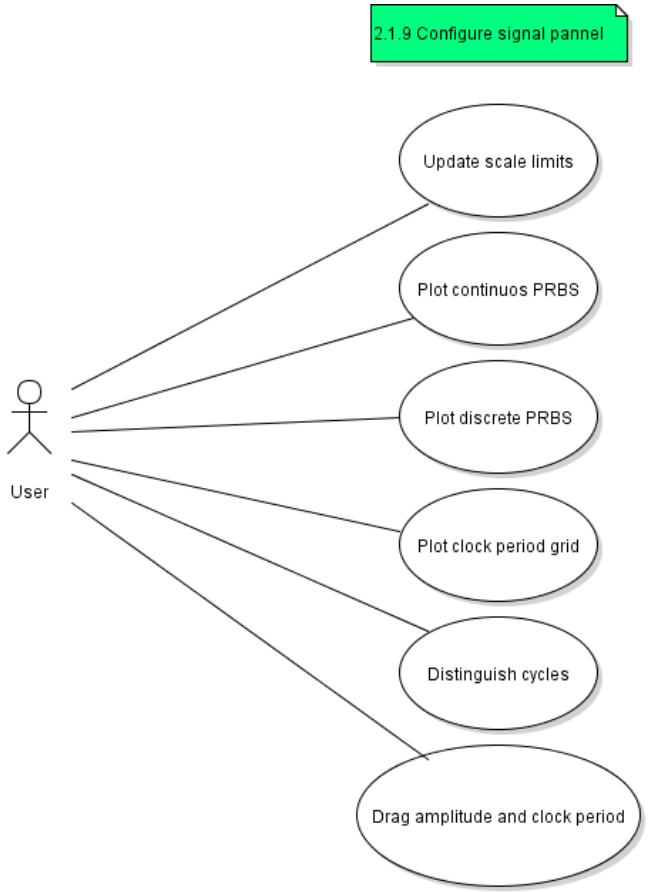

Figura 37: Casos de uso nivel 3, configure signal pannel

Tabla 37: Caso de uso, update scale limits

Caso de uso:	**3.1.9.1 Update scale limits.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la señal PRBS.
Descripción:	El usuario puede escoger dos tipos de escala: • *Variable*, el sistema ajusta la mejor opción de escala para visualizar toda la señal en el panel. • *Locked*, en este caso el usuario es libre de modificar el valor mínimo y máximo de los dos ejes de la señal, lo que le permitirá ver detalles de la misma.
Secuencia normal:	1. El usuario escoge entre una de las dos opciones de manejo de la escala. 2. En el caso de escoger escala *Locked*, puede seleccionar a continuación los nuevos valores de t_{min}, t_{max}, y_{min} e y_{max}. 3. La gráfica se actualiza con los nuevos valores de ejes.
Postcondiciones:	El panel gráfico *signal* de la señal PRBS queda modificado.
Excepciones:	1. Si estando seleccionada la opción de escala *Variable* el usuario intenta modificar los límites de los ejes el programa genera un mensaje de error 2. No se modifica la gráfica *signal*.

Tabla 38: Caso de uso, plot continuous PRBS

Caso de uso:	**3.1.9.2 Plot continuous PRBS.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la señal PRBS.
Descripción:	Si se selecciona este checkbox el gráfico de la señal se muestra de forma continua. Si se deselecciona la forma continua desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la señal de forma continua. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 39: Caso de uso, plot discrete PRBS

Caso de uso:	**3.1.9.3 Plot discrete PRBS.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la señal PRBS.
Descripción:	Si se selecciona este checkbox el gráfico de la señal se muestra de forma discreta. Si se deselecciona la forma discreta de la señal desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la señal de forma discreta. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 40: Caso de uso, plot clock period grid.

Caso de uso:	**3.1.9.4 Plot clock period grid.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la señal PRBS.
Descripción:	Si se selecciona este checkbox se muestra unas líneas verticales punteadas coincidentes con los instantes en que se producen los impulsos de reloj. Si se deselecciona la rejilla desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la rejilla del periodo de reloj. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 41: Caso de uso, distinguish cycles.

Caso de uso:	**3.1.9.5 Distinguish cycles.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la señal PRBS.
Descripción:	Si se selecciona este checkbox se crean bandas de color azul claro que permiten distinguir los distintos ciclos de la señal PRBS. Si se deselecciona las bandas de color desaparecen. Solo se verán bandas de color si la opción *cycles number* es mayor a 1.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar las bandas de color que distinguen los distintos ciclos de la señal PRBS. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 42: Caso de uso, drag amplitude and clock period.

Caso de uso:	**3.1.9.6 Drag amplitude and clock period.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la señal PRBS.
Descripción:	Si se selecciona este checkbox en el gráfico aparecen dos líneas que permiten modificar dinámicamente la amplitud de la señal y su periodo de reloj.
Secuencia normal:	1. El usuario escoge poder modificar la amplitud y periodo de reloj dinámicamente. 2. Si el usuario sube o baja la línea horizontal en el gráfico, la amplitud de la señal se modifica. 3. Si el usuario mueve a izquierda o derecha la línea vertical del gráfico, el periodo de reloj se modifica. 4. Los valores de las variables amplitud y periodo se modifican. 5. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal PRBS queda modificado. Los valores de amplitud y periodo de reloj de la señal PRBS se modifican.
Excepciones:	Ninguna.

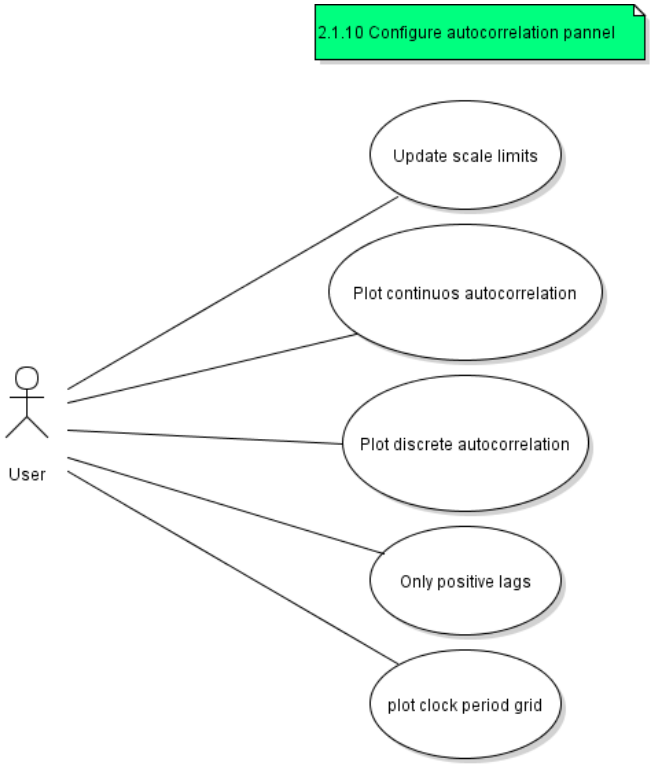

Figura 38: Casos de uso nivel 3, configure autocorrelation pannel.

Tabla 43: Caso de uso, update scale limits

Caso de uso:	**3.1.10.1 Update scale limits.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	El usuario puede escoger dos tipos de escala: • *Variable*, el sistema ajusta la mejor opción de escala para visualizar toda la función de autocorrelación en el panel. • *Locked*, en este caso el usuario es libre de modificar el valor mínimo y máximo de los dos ejes de la señal, lo que le permitirá ver detalles de la misma.
Secuencia normal:	1. El usuario escoge entre una de las dos opciones de manejo de la escala. 2. En el caso de escoger escala *Locked*, puede seleccionar a continuación los nuevos valores de x_{min}, x_{max}, y_{min} e y_{max}. 3. La gráfica se actualiza con los nuevos valores de ejes.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal PRBS queda modificado.
Excepciones:	1. Si estando seleccionada la opción de escala *Variable* el usuario intenta modificar los límites de los ejes el programa genera un mensaje de error 2. No se modifica la gráfica *autocorrelation*.

Tabla 44: Caso de uso, plot continuous autocorrelation

Caso de uso:	**3.1.10.2 Plot continuous autocorrelation.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	Si se selecciona este checkbox el gráfico de la función de autocorrelación se muestra de forma continua. Si se deselecciona la forma continua desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la función de autocorrelación de forma continua. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 45: Caso de uso, plot discrete autocorrelation

Caso de uso:	**3.1.10.3 Plot discrete autocorrelation.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	Si se selecciona este checkbox el gráfico de la función de autocorrelación se muestra de forma discreta. Si se deselecciona la forma discreta de la señal desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la función de autocorrelación de forma discreta. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 46: Caso de uso, only positive lags

Caso de uso:	**3.1.10.4 Only positive lags.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	Si se selecciona este checkbox el gráfico de la función de autocorrelación muestra solo su parte positiva en el tiempo. Si se deselecciona se muestra la función de autocorrelación simétrica con respecto a cero.
Secuencia normal:	1. El usuario escoge entre mostrar solo la parte positiva de la función de autocorrelación o mostrarla completa. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 47: Caso de uso, plot clock period grid

Caso de uso:	**3.1.10.5 Plot clock period grid.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	Si se selecciona este checkbox se muestra unas líneas verticales punteadas coincidentes con los instantes en que se producen los impulsos de reloj. Si se deselecciona la rejilla desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la rejilla del periodo de reloj. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

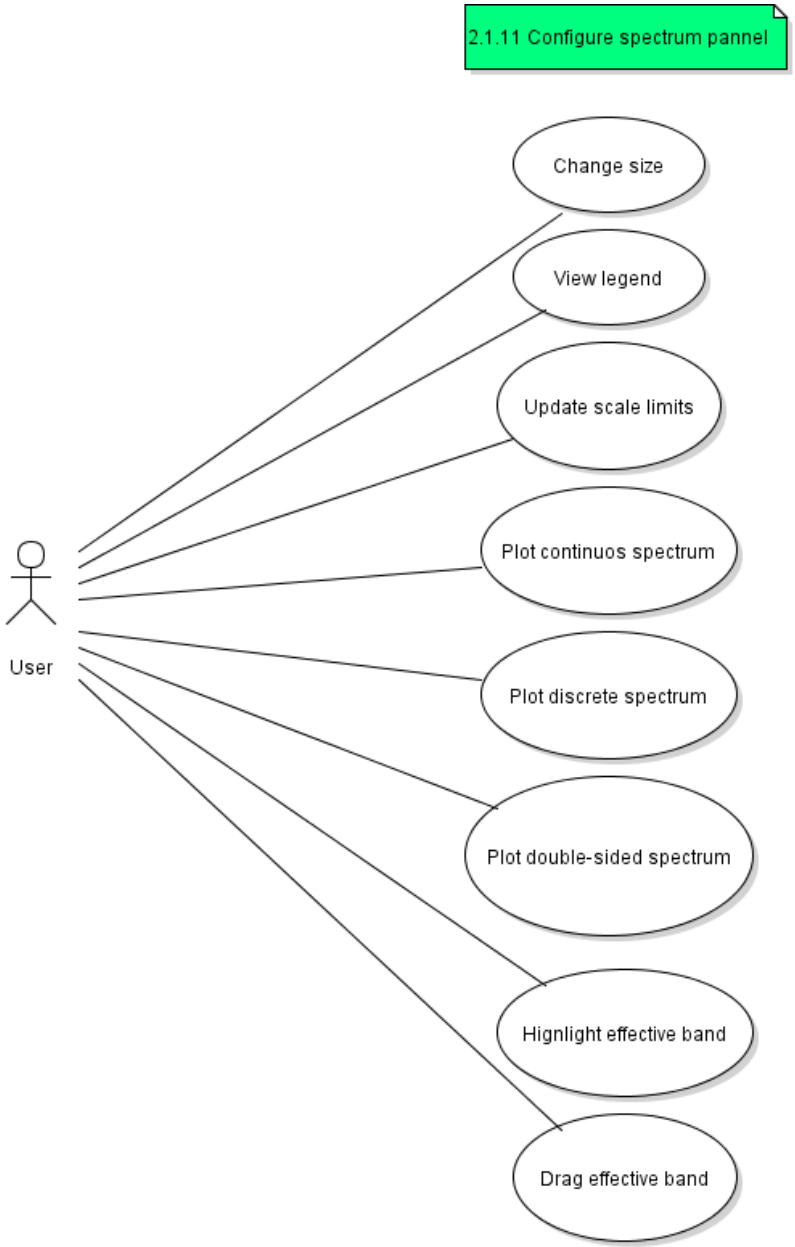

Figura 39:Caso de uso nivel 3, configure spectrum pannel.

Tabla 48: Caso de uso, change size.

Caso de uso:	**3.1.11.1 Change size.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Aparecen dos opciones *small / medium*, si se selecciona la opción *small* el gráfico ocupa un tercio del área de gráficos, (opción por defecto). Cuando se selecciona la opción *medium* el gráfico ocupa dos tercios del área de gráficos haciendo desaparecer el gráfico de la función de autocorrelación.
Secuencia normal:	1. El usuario escoge entre la opción *small* o *medium*. 2. La gráfica *power spectrum* cambia su tamaño para adaptarse a la elección.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 49: Caso de uso, view legend.

Caso de uso:	**3.1.11.2 View legend.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Aparecen dos opciones *yes / no*, que permiten visualizar o no la leyenda sobre la gráfica. La leyenda indica si se está viendo el espectro de la señal continua / discreta o ambas.
Secuencia normal:	1. El usuario escoge entre la ocultar o mostrar la leyenda. 2. La gráfica *power spectrum* se modifica para adaptarse a la elección.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 50: Caso de uso, update scale limits

Caso de uso:	**3.1.11.3 Update scale limits.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	El usuario puede escoger dos tipos de escala: • *Variable*, el sistema ajusta la mejor opción de escala para visualizar todo el espectro en el panel. • *Locked*, en este caso el usuario es libre de modificar el valor mínimo y máximo de los dos ejes de la señal, lo que le permitirá ver detalles del espectro. El usuario puede escoger el tipo de unidades para visualizar el eje de frecuencia: • *Hz*, la frecuencia se visualiza en hercios o ciclos por segundo. • *Rad/s*, la frecuencia se visualiza en radianes por segundo. • *Harmonic number*, en el eje x se ven los números de armónicos de la señal. Escala de los ejes x e y: • *Linear*, el usuario puede escoger ver cualquiera de los dos ejes en escala lineal. • *Logarithmic*, el usuario puede escoger ver cualquiera de los dos ejes en escala logarítmica.
Secuencia normal:	Elección de escala: 1. El usuario escoge entre una de las dos opciones de manejo de la escala. 2. En el caso de escoger escala *Locked*, puede seleccionar a continuación los nuevos valores de x_{min}, x_{max}, y_{min} e y_{max}. 3. La gráfica se actualiza con los nuevos valores de ejes. Elección de unidades: 1. El usuario escoge entre una de las tres opciones de unidades para el eje y. 2. La gráfica se actualiza con los nuevos valores de ejes. Elección de ejes: 1. El usuario escoge entre escala lineal o logarítmica en el eje x o y. 2. La gráfica se actualiza con los nuevos valores de ejes.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	1. Si estando seleccionada la opción de escala *Variable* el usuario intenta modificar los límites de los ejes el programa genera un mensaje de error 2. No se modifica la gráfica *signal*.

Tabla 51: Caso de uso, plot continuous spectrum

Caso de uso:	**3.1.11.4 Plot continuous spectrum.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox el gráfico del espectro de la señal se muestra de forma continua. Si se deselecciona la forma continua del espectro desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar el espectro de forma continua. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 52: Caso de uso, plot discrete spectrum

Caso de uso:	**3.1.11.5 Plot discrete spectrum.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox el gráfico del espectro de la señal se muestra de forma discreta. Si se deselecciona la forma discreta del espectro desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar el espectro de forma discreta. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 53: Caso de uso, plot double-sided spectrum.

Caso de uso:	**3.1.11.6 Plot double-sided spectrum.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox se muestra tanto la parte positiva como negativa del espectro de potencia de la señal. Cuando se deselecciona solo se muestra la parte positiva del esepctro.
Secuencia normal:	1. El usuario escoge entre mostrar el espectro de forma simétrica con respecto al origen o solo su parte positiva. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 54: caso de uso, highligth effective band.

Caso de uso:	**3.1.11.7 Highligth effective band.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox se colorea en verde la parte del espectro que coincide con el ancho de banda de la señal. Esto se produce en el momento en que la ganancia cae 3db.
Secuencia normal:	1. El usuario escoge entre mostrar el ancho de banda de la señal o no. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

Tabla 55: Caso de uso, drag effective band.

Caso de uso:	**3.1.11.8 Drag effective band.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal PRBS y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox en el gráfico aparece una línea vertical que permite modificar dinámicamente el ancho de banda de la señal.
Secuencia normal:	1. El usuario escoge poder modificar el ancho de banda del espectro de la señal dinámicamente. 2. Si el usuario mueve a izquierda o derecha la línea vertical del gráfico, el ancho de banda de la señal se modifica. 3. Los valores de las variables periodo de muestreo y reloj se modifican para ajustarse al nuevo ancho de banda. 4. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal PRBS queda modificado.
Excepciones:	Ninguna.

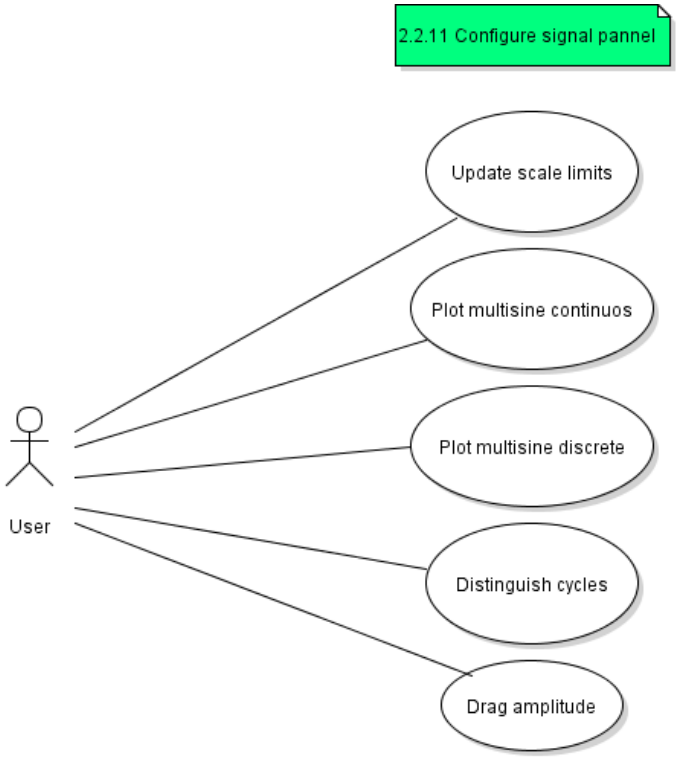

Figura 40: Casos de uso nivel 3, configure signal pannel

Tabla 56: Caso de uso, update scale limits

Caso de uso:	**3.2.11.1 Update scale limits.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la señal multiseno.
Descripción:	El usuario puede escoger dos tipos de escala: • *Variable*, el sistema ajusta la mejor opción de escala para visualizar toda la señal en el panel. • *Locked*, en este caso el usuario es libre de modificar el valor mínimo y máximo de los dos ejes de la señal, lo que le permitirá ver detalles de la misma.
Secuencia normal:	1. El usuario escoge entre una de las dos opciones de manejo de la escala. 2. En el caso de escoger escala *Locked*, puede seleccionar a continuación los nuevos valores de t_{min}, t_{max}, y_{min} e y_{max}. 3. La gráfica se actualiza con los nuevos valores de ejes.
Postcondiciones:	El panel gráfico *signal* de la señal multiseno queda modificado.
Excepciones:	1. Si estando seleccionada la opción de escala *Variable* el usuario intenta modificar los límites de los ejes el programa genera un mensaje de error 2. No se modifica la gráfica *signal*.

Tabla 57: Caso de uso, plot multisine continuous

Caso de uso:	**3.2.11.2 Plot multisine continuous.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la señal multiseno.
Descripción:	Si se selecciona este checkbox el gráfico de la señal se muestra de forma continua. Si se deselecciona la forma continua desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la señal de forma continua. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 58: Caso de uso, plot multisine discrete

Caso de uso:	**3.2.11.3 Plot multisine discrete.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la señal multiseno.
Descripción:	Si se selecciona este checkbox el gráfico de la señal se muestra de forma discreta. Si se deselecciona la forma discreta de la señal desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la señal de forma discreta. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 59: Caso de uso, distinguish cycles.

Caso de uso:	**3.2.11.4 Distinguish cycles.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la señal multiseno.
Descripción:	Si se selecciona este checkbox se crean bandas de color azul claro que permiten distinguir los distintos ciclos de la señal multiseno. Si se deselecciona las bandas de color desaparecen. Solo se verán bandas de color si la opción *cycles number* es mayor a 1.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar las bandas de color que distinguen los distintos ciclos de la señal multiseno. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 60: Caso de uso, drag amplitude and clock period.

Caso de uso:	**3.2.11.5 Drag amplitude.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la señal multiseno.
Descripción:	Si se selecciona este checkbox en el gráfico aparecen una línea que permiten modificar dinámicamente la amplitud de la señal.
Secuencia normal:	1. El usuario escoge poder modificar la amplitud dinámicamente. 2. Si el usuario sube o baja la línea horizontal en el gráfico, la amplitud de la señal se modifica. 3. El valor de la variable amplitud se modifica. 4. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *signal* de la señal multiseno queda modificado. El valores de amplitud de la señal multiseno se modifica.
Excepciones:	Ninguna.

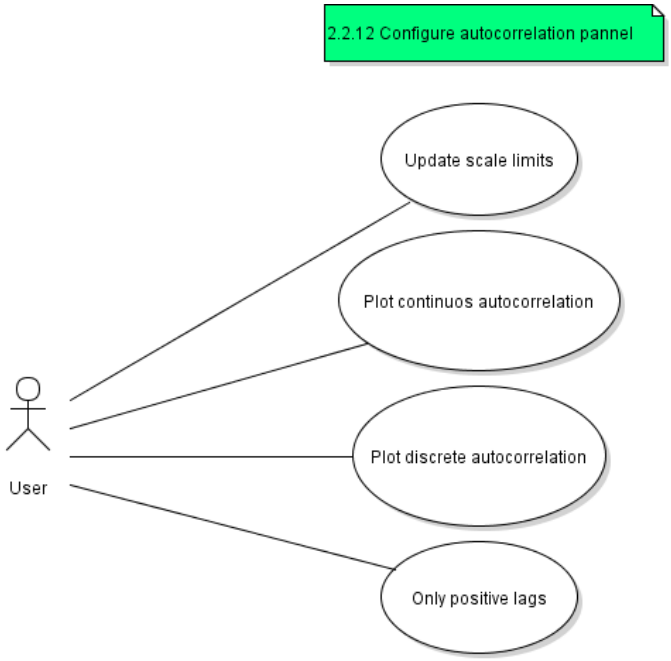

Figura 41: Casos de uso nivel 3, configure autocorrelation pannel

Tabla 61: Caso de uso, update scale limits

Caso de uso:	**3.2.12.1 Update scale limits.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	El usuario puede escoger dos tipos de escala: • *Variable*, el sistema ajusta la mejor opción de escala para visualizar toda la función de autocorrelación en el panel. • *Locked*, en este caso el usuario es libre de modificar el valor mínimo y máximo de los dos ejes de la señal, lo que le permitirá ver detalles de la misma.
Secuencia normal:	1. El usuario escoge entre una de las dos opciones de manejo de la escala. 2. En el caso de escoger escala *Locked*, puede seleccionar a continuación los nuevos valores de x_{min}, x_{max}, y_{min} e y_{max}. 3. La gráfica se actualiza con los nuevos valores de ejes.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal multiseno queda modificado.
Excepciones:	1. Si estando seleccionada la opción de escala *Variable* el usuario intenta modificar los límites de los ejes el programa genera un mensaje de error 2. No se modifica la gráfica *autocorrelation*.

Tabla 62: Caso de uso, plot continuous autocorrelation

Caso de uso:	**3.2.12.2 Plot continuous autocorrelation.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	Si se selecciona este checkbox el gráfico de la función de autocorrelación se muestra de forma continua. Si se deselecciona la forma continua desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la función de autocorrelación de forma continua. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 63: Caso de uso, plot discrete autocorrelation

Caso de uso:	**3.2.12.3 Plot discrete autocorrelation.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	Si se selecciona este checkbox el gráfico de la función de autocorrelación se muestra de forma discreta. Si se deselecciona la forma discreta de la señal desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar la función de autocorrelación de forma discreta. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 64: Caso de uso, only positive lags

Caso de uso:	**3.2.12.4 Only positive lags.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel de la función de autocorrelación.
Descripción:	Si se selecciona este checkbox el gráfico de la función de autocorrelación muestra solo su parte positiva en el tiempo. Si se deselecciona se muestra la función de autocorrelación simétrica con respecto a cero.
Secuencia normal:	1. El usuario escoge entre mostrar solo la parte positiva de la función de autocorrelación o mostrarla completa. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *autocorrelation function* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

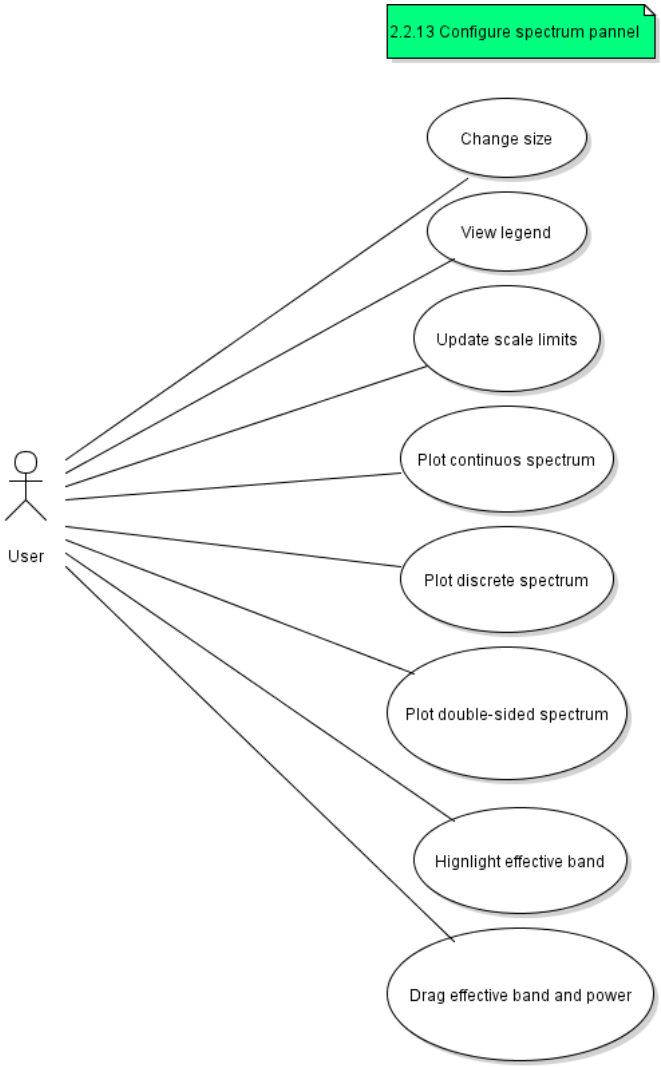

Figura 42: Casos de uso nivel 3, configure spectrum pannel

Tabla 65: Caso de uso, change size.

Caso de uso:	**3.2.12.1 Change size.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Aparecen dos opciones *small / medium*, si se selecciona la opción *small* el gráfico ocupa un tercio del área de gráficos, (opción por defecto). Cuando se selecciona la opción *medium* el gráfico ocupa dos tercios del área de gráficos haciendo desaparecer el gráfico de la función de autocorrelación.
Secuencia normal:	1. El usuario escoge entre la opción *small* o *medium*. 2. La gráfica *power spectrum* cambia su tamaño para adaptarse a la elección.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 66: Caso de uso, view legend.

Caso de uso:	**3.2.12.2 View legend.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Aparecen dos opciones *yes / no*, que permiten visualizar o no la leyenda sobre la gráfica. La leyenda indica si se está viendo el espectro de la señal continua / discreta o ambas.
Secuencia normal:	1. El usuario escoge entre la ocultar o mostrar la leyenda. 2. La gráfica *power spectrum* se modifica para adaptarse a la elección.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 67: Caso de uso, update scale limits

Caso de uso:	**3.2.12.3 Update scale limits.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	El usuario puede escoger dos tipos de escala: • *Variable*, el sistema ajusta la mejor opción de escala para visualizar todo el espectro en el panel. • *Locked*, en este caso el usuario es libre de modificar el valor mínimo y máximo de los dos ejes de la señal, lo que le permitirá ver detalles del espectro. El usuario puede escoger el tipo de unidades para visualizar el eje de frecuencia: • *Hz*, la frecuencia se visualiza en hercios o ciclos por segundo. • *Rad/s*, la frecuencia se visualiza en radianes por segundo. • *Harmonic number*, en el eje x se ven los números de armónicos de la señal. Escala de los ejes x e y: • *Linear*, el usuario puede escoger ver cualquiera de los dos ejes en escala lineal. • *Logarithmic*, el usuario puede escoger ver cualquiera de los dos ejes en escala logarítmica.
Secuencia normal:	Elección de escala: 1. El usuario escoge entre una de las dos opciones de manejo de la escala. 2. En el caso de escoger escala *Locked*, puede seleccionar a continuación los nuevos valores de x_{min}, x_{max}, y_{min} e y_{max}. 3. La gráfica se actualiza con los nuevos valores de ejes. Elección de unidades: 1. El usuario escoge entre una de las tres opciones de unidades para el eje y. 2. La gráfica se actualiza con los nuevos valores de ejes. Elección de ejes: 1. El usuario escoge entre escala lineal o logarítmica en el eje x o y. 2. La gráfica se actualiza con los nuevos valores de ejes.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	1. Si estando seleccionada la opción de escala *Variable* el usuario intenta modificar los límites de los ejes el programa genera un mensaje de error 2. No se modifica la gráfica *signal*.

Tabla 68: Caso de uso, plot continuous spectrum

Caso de uso:	**3.2.12.4 Plot continuous spectrum.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox el gráfico del espectro de la señal se muestra de forma continua. Si se deselecciona la forma continua del espectro desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar el espectro de forma continua. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 69: Caso de uso, plot discrete spectrum

Caso de uso:	**3.2.12.5 Plot discrete spectrum.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox el gráfico del espectro de la señal se muestra de forma discreta. Si se deselecciona la forma discreta del espectro desaparece.
Secuencia normal:	1. El usuario escoge entre mostrar o no mostrar el espectro de forma discreta. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 70: Caso de uso, plot double-sided spectrum.

Caso de uso:	**3.2.12.6 Plot double-sided spectrum.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox se muestra tanto la parte positiva como negativa del espectro de potencia de la señal. Cuando se deselecciona solo se muestra la parte positiva del esepctro.
Secuencia normal:	1. El usuario escoge entre mostrar el espectro de forma simétrica con respecto al origen o solo su parte positiva. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 71: caso de uso, highligth effective band.

Caso de uso:	**3.2.12.7 Highligth effective band.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox se colorea en verde la parte del espectro que coincide con el ancho de banda de la señal. Esto se produce en el momento en que la ganancia cae 3db.
Secuencia normal:	1. El usuario escoge entre mostrar el ancho de banda de la señal o no. 2. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

Tabla 72: Caso de uso, drag effective band and power.

Caso de uso:	**3.2.12.8 Drag effective band and power.**
Dependencias:	Ninguna.
Precondiciones:	Se ha seleccionado previamente la señal multiseno y se ha entrado en el menú de configuración del panel del espectro de potencia.
Descripción:	Si se selecciona este checkbox en el gráfico aparecen dos líneas que permiten modificar dinámicamente la amplitud de la señal y su ancho de banda.
Secuencia normal:	1. El usuario escoge poder modificar el ancho de banda del espectro de la señal dinámicamente. 2. Si el usuario mueve a izquierda o derecha la línea vertical del gráfico, el ancho de banda de la señal se modifica. 3. Si el usuario mueve la línea horizontal hacia arriba o abajo se modifica la amplitud de la señal y su energía. 4. Los valores de las variables amplitud, periodo de muestreo y frecuencia de reloj se modifican para ajustarse al nuevo ancho de banda y amplitud. 5. La gráfica se actualiza.
Postcondiciones:	El panel gráfico *power spectrum* de la señal multiseno queda modificado.
Excepciones:	Ninguna.

4.3.3 Diagramas de clases

Los diagramas de clases permiten definir los principales elementos del programa, su estructura y cómo se relacionan entre sí.

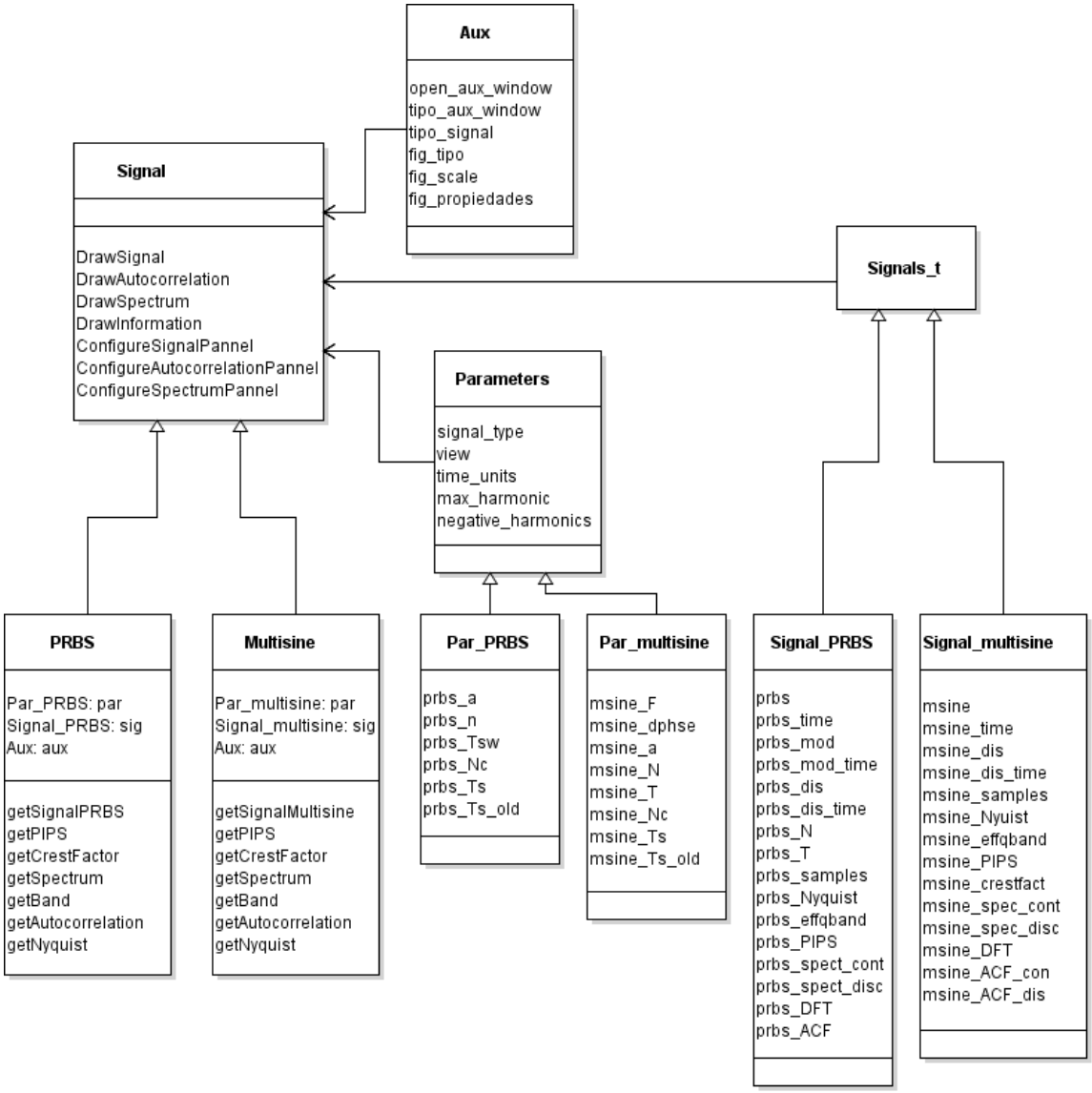

Figura 43: diagrama de clases

Tabla 73: Clase Signals_t

Nombre	Signals_t
Tipo	Abstract.
Relación	Clase padre de Signal_PRBS y Signal_mutisine.
Descripción	Clase abstracta que es la superclase de las secuencias de las señales y los atributos de las mismas.
Atributos	
Métodos	

Tabla 74: Clase Signal_PRBS

Nombre	Signal_PRBS
Tipo	Clase.
Relación	Hereda de Signals_t.
Descripción	Clase de la secuencia PRBS y sus atributos.
Atributos	-prbs: vector, vector con la secuencia. -prbs_time: vector, vector con los instantes de tiempo que corresponden a cada elemento de la secuencia -prbs_mod: vector, secuencia modificada para poder se ploteada -prbs_mod_time: vector, vector de tiempos modificados para poder ser ploteados. -prbs_dis: vector, vector de la secuencia muestreada. -prbs_dis_time: vector, vector de tiempo de la secuencia muestreada. -prbs_N: integer, numero de bits de la secuencia. -prbs_T: integer, periodo de la secuencia. -prbs_samples: integer, número de muestras de la secuencia. -prbs_Nyquist: vector, número de armónicos y frecuencia. -prbs_effqband: vector, número de armónico mínimo, máximo, frecuencia mínima y máxima. -prbs_PIPS: float, índice PIPS de la señal. -prbs_spect_cont: matrix, [número de armónico, frecuencia, espectro continuo]. -prbs_spect_disc: matrix, [número de armónico, frecuencia, espectro discreto]. -prbs_DFT: matrix, [número de armónico, frecuencia, DTF]. -prbs_ACF: matrix, [delay, ACF].
Métodos	

Tabla 75: Clase Signal_multisine

Nombre	Signal_multisine
Tipo	Clase.
Relación	Hereda de Signals_t.
Descripción	Clase de la secuencia multisine y sus atributos.
Atributos	-msine: vector, vector con la secuencia. -msine_time: vector, vector con los instantes de tiempo que corresponden a cada elemento de la secuencia -msine_dis: vector, vector de la secuencia muestreada. -msine_dis_time: vector, vector de tiempo de la secuencia muestreada. -msine_samples: integer, número de muestras de la secuencia. -msine_Nyquist: vector, número de armónicos y frecuencia. -msine_effqband: vector, número de armónico mínimo, máximo, frecuencia mínima y máxima. -msine_PIPS: float, índice PIPS de la señal. -msine_crestfact: float, índice factor de cresta de la señal. -msine_spect_cont: matrix, [número de armónico, frecuencia, espectro continuo]. -msine_spect_disc: matrix, [número de armónico, frecuencia, espectro discreto]. -msine_DFT: matrix, [número de armónico, frecuencia, DTF]. -msine_ACF_con: matrix, [delay, ACF] continuo. -msine_ACF_dis: matrix, [delay, ACF] discreto.
Métodos	

Tabla 76: Clase Aux

Nombre	Aux
Tipo	Clase
Relación	Ninguna.
Descripción	Clase auxiliar asociada a cada ventana de gráfico que almacena el valor de diversos atributos del gráfico.
Atributos	-open_aux_window: integer, indica si la ventana auxiliar está abierta o cerrada. -tipo_aux_window: integer, indica el contenido de la ventana auxiliar. -tipo_signal: integer, tipo de señal. -fig_tipo: integer, tipo de escala de la ventana. -fig_scale: vector, datos de la escala -fig_propiedades: integer, propiedades de la figura.
Métodos	

Tabla 77: Clase Parameters

Nombre	Parameters
Tipo	Abstract.
Relación	Clase padre de Par_PRBS y Par_mutisine.
Descripción	Clase abstracta que es la superclase de los parámetros utilizados para generar las secuencias.
Atributos	-signal_type: integer, tipo de señal. -View: integer, forma de visualizar la señal, continua o discreta. -time_units: integer, unidad de tiempo empleada -max_harmonic: integer, número máximo de armónico que se muestra en la gráfica de la señal. -negative_harmonics: boolean, los armónicos negativos se tienen en cuenta o no.
Métodos	

Tabla 78: Clase Par_PRBS

Nombre	Par_PRBS
Tipo	Clase.
Relación	Hereda de Parameters.
Descripción	Clase que almacena los parámetros utilizados para generar la secuencia PRBS.
Atributos	-prbs_a: float, amplitud de la señal. -prbs_n: integer, tamaño del registro de desplazamiento en bits. -prbs_Tsw: integer, número de unidades de tiempo del impulso de reloj. -prbs_Nc: integer, número de ciclos de la señal. -prbs_Ts: integer, periodo de muestreo actual. -prbs_Ts_old: integer, periodo de muestreo anterior.
Métodos	

Tabla 79: Clase Par_multisine

Nombre	Par_multisine
Tipo	Clase.
Relación	Hereda de Parameters.
Descripción	Clase que almacena los parámetros utilizados para generar la señal multiseno.
Atributos	-msine_F: integer, número de armónicos de la señal. -msine_dphase: integer, método para generar la fase inicial de la señal. -msine_a: float, amplitud de cada armónico. -msine_N: integer, longitud de la señal. -msine_T: integer, periodo de la señal. -msine_Nc: integer, número de ciclos de la señal. -msine_Ts: integer, periodo de muestreo de la señal. -msine_Ts_old: integer, periodo de muestreo anterior.
Métodos	

Tabla 80: Clase Signal

Nombre	Signal
Tipo	Abstract
Relación	Clase padre de PRBS y Multisine.
Descripción	Superclase que define los métodos que al menos deben contener las clases PRBS y Multisine.
Atributos	
Métodos	DrawSignal, dibuja el gráfico de la señal. DrawAutocorrelation, dibuja el gráfico de la función de autocorrelación de la señal. DrawSpectrum, dibuja el gráfico del espectro de la señal. DrawInformation, dibuja la información asociada a la señal. ConfigureSignalPannel, abre el panel de configuración de la señal. ConfigureAutocorrelationPannel, abre el panel de configuración de la función de autocorrelación. ConfigureSpectrumPannel, abre el panel de configuración del espectro de la señal.

Tabla 81: Clase PRBS

Nombre	PRBS
Tipo	Clase
Relación	Hereda de la clase Signal.
Descripción	Clase que contiene las operaciones básicas de la señal PRBS.
Atributos	Par_PRBS par Signal_PRBS sig Aux aux
Métodos	getSignalPRBS, método que devuelve un vector con la señal PRBS construida según la configuración actual. getPIPS, método que devuelve el valor del índice PIPS. getCrestFactor, método que devuelve el valor del fator de cresta. getSpectrum, método que devuelve un vector con el espectro de la señal PRBS. getBand, método que devuelve el ancho de banda actual. getAutocorrelation, método que devuelve un vector con la función de autocorrelación de la señal. getNyquist, método que devuelve la frecuencia de Nyquist.

Tabla 82: Clase Multisine

Nombre	Multisine
Tipo	Clase
Relación	Hereda de la clase Signal.
Descripción	Clase que contiene las operaciones básicas de la señal multiseno.
Atributos	Par_multisine: par Signal_multisine: sig Aux: aux
Métodos	getSignalMultisine, método que devuelve un vector con la señal multiseno construida según la configuración actual. getPIPS, método que devuelve el valor del índice PIPS. getCrestFactor, método que devuelve el valor del fator de cresta. getSpectrum, método que devuelve un vector con el espectro de la señal PRBS. getBand, método que devuelve el ancho de banda actual. getAutocorrelation, método que devuelve un vector con la función de autocorrelación de la señal. getNyquist, método que devuelve la frecuencia de Nyquist.

5 Ejemplos de uso de la herramienta.

En esta sección se van a describir una serie de ejemplos de uso de la herramienta, así como un resumen del fundamento teórico de cada uno de los ejemplos.

5.1 Generación de señales PRBS

En la herramienta software desarrollada, la señal PRBS se genera a partir de la información contenida en un registro de desplazamiento de *n* bits realimentado entre alguna de sus posiciones y el primero de sus bits con una función XOR (ver Figura 14). La salida del registro de desplazamiento invertida se toma como salida que genera la secuencia PRBS. La longitud máxima de la secuencia será 2^n-1 bits. Que una secuencia sea de longitud máxima o no depende de la realimentación escogida para la secuencia. La secuencia generada será una señal simétrica con respecto a 0 de amplitud ±*A*.

La herramienta se presenta en dos formas, como ejecutable *prbs_multisine.exe* o como script de Sysquake *prbs_multisine.sq*. En cualquiera de estas dos formas al iniciarla aparecerá el contenido de la Figura 44. En este primer ejemplo observaremos la influencia de cada uno de los parámetros en el diseño de una señal PRBS.

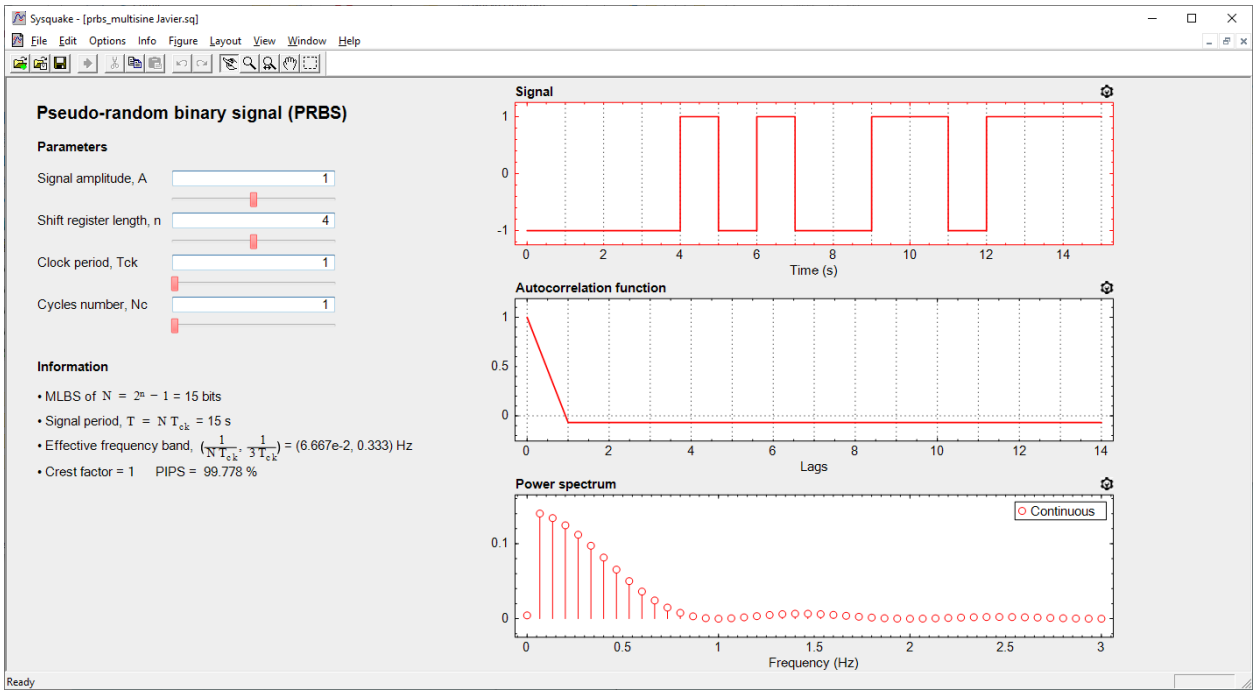

Figura 44: Pantalla principal de la aplicación

La **amplitud de la señal** es un factor importante a tener en cuenta a la hora de generar una entrada para un sistema a identificar. Por una parte, debe ser lo suficientemente grande como para superar al ruido presente en el sistema y a la vez no producir la saturación de los actuadores e introducir no linealidades en el sistema. Introduciendo un nuevo valor en el cuadro *Signal amplitude, A* o deslizando la barra que hay debajo del cuadro (ver Figura 45) se puede modificar su valor. Cualquier cambio realizado sobre los parámetros de la señal tiene un reflejo inmediato en las gráficas de la señal. Si se modifica el valor de la amplitud de la señal para conseguir un valor de 5 que supondrá generar una señal de ±5v de amplitud se observa

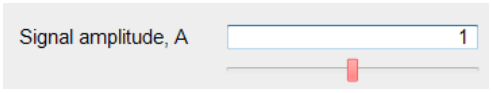

Figura 45: Control de la amplitud de la señal.

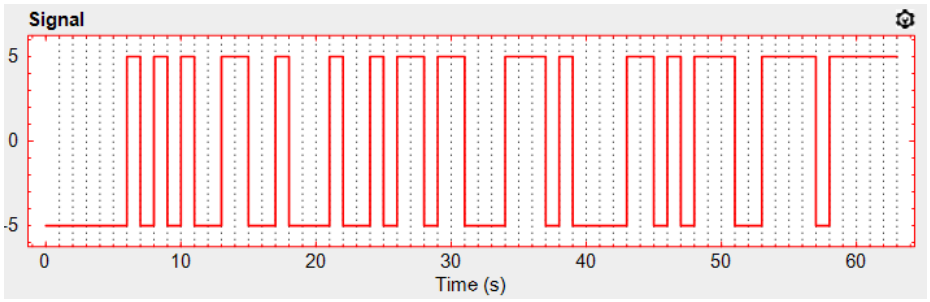

Figura 46: Señal de amplitud ±5v y 63 bits de longitud.

La **longitud del registro de desplazamiento** marca el tamaño de la secuencia. Por ejemplo, modifica el tamaño *n* a 6 bits, se conseguirá una secuencia de 2^6-1 = 63 bits (ver Figura 46). El tamaño máximo de secuencia que puede generar esta versión de la herramienta es de 2^{11}-1 = 2047 bits.

Al modificar el **periodo de reloj** se consigue que un bit de la secuencia dure más de un impulso de reloj. Este control se puede usar en combinación con el menú *time units* para fijar la duración deseada de cada bit. Si por ejemplo desea que cada bit de la secuencia tome 5 milisegundos bastará con escoger en el menú *time units* la unidad de tiempo en milisegundos y el periodo de reloj Tck = 5 ms.

Estos cambios afectarán al periodo de la señal y a su ancho de banda. El periodo de la señal *T = N*Tck* y el ancho de banda de la señal es el punto en que el espectro cae 3db desde la frecuencia base *BW=1/(N*TcK), 1/(3*Tck)*

El periodo de la señal será la duración de un bit por el número de bits de la secuencia. Por tanto, en este caso si cada bit dura 5ms y tenemos una secuencia de 63 bits, el periodo será de 315ms. En el caso del ancho de banda la frecuencia máxima será 1/3 de la frecuencia de reloj, y la frecuencia de reloj el inverso del periodo de reloj. Entonces en el caso anterior tendremos una frecuencia máxima de $1/15*10^{-3}$ = 66.66 Hz. La frecuencia mínima será la frecuencia de reloj partido por la longitud de la secuencia o 1/N*Tck, para los valores anteriores el resultado es de 3.175Hz.

El último parámetro que se puede modificar está en el menú *Sampling Time* y es el **periodo de muestreo** de la señal. La secuencia se puede muestrear una o varias veces por cada ciclo de reloj, (ver Figura 47). El principal efecto de este control es aumentar la frecuencia de Nyquist para reducir el posible efecto del aliasing sobre la señal. Si cada D ciclos de reloj se muestrea la señal, el periodo de muestreo será Ts = Tck/D. Con objeto de conseguir que el periodo de muestreo sea un número racional, no todos los valores de D son posibles, solo aquellos que sean múltiplos de 2 y 5.

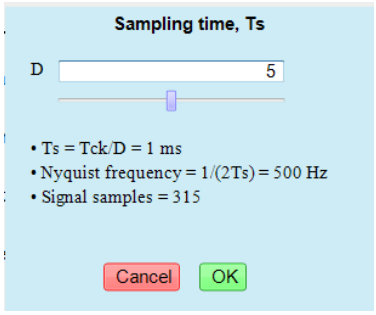

Figura 47: Cuadro de selección del periodo de muestreo.

La frecuencia de Nyquist es 1/2Ts. Con el periodo escogido de 5mS la frecuencia de Nyquist es de 500Hz, pero si se cambia el valor de D por 10, la frecuencia de Nyquist sube a 1000Hz. Ts = 5/10 = 0.5ms, $1/2*0.5*10^{-3}$=1000Hz.

Por último, si se desea que la señal se repita cíclicamente durante varios ciclos, se puede utilizar el control *Cycles number*, *Nc* y seleccionar el valor de repetición deseado. La aplicación limita la longitud máxima de la secuencia a 10000 muestras.

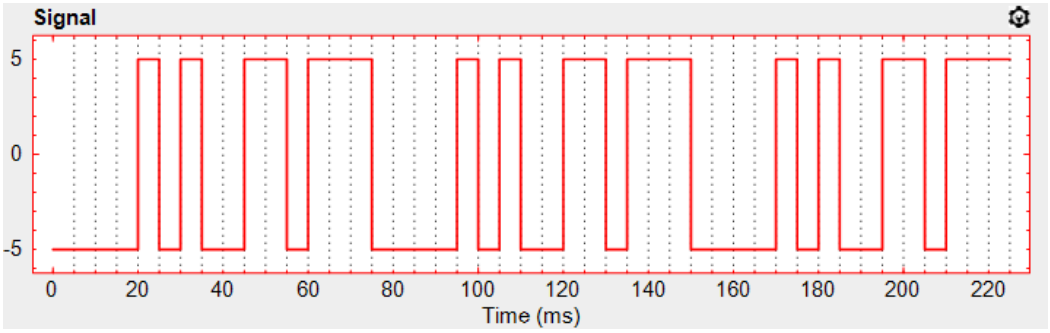

Figura 48:Señal PRBS que se repite durante 3 ciclos.

Con respecto a la función de autocorrelación en la propiedad 13 de la señal PRBS, vista en el apartado 3.7.1. Entonces estudiamos que la amplitud de la función para k=0 es A^2 y para el resto de valores de la secuencia es $-A^2/N$.

Si por ejemplo tomamos una secuencia de longitud 4 con A=5, tendremos en el índice 0 de la secuencia una amplitud de 25 y en el resto de valores -25/15 = -1.66. En la Figura 49 pueden verse estos resultados en la gráfica de autocorrelación de una secuencia de 15 bits longitud y 5 de amplitud.

Figura 49: Valores de la función de autocorrelación en el instante inicial y en el resto de elementos de la secuencia.

La función de autocorrelación es periódica. Por tanto, se repetirá en cada ciclo completo de la secuencia. Por ejemplo, en la Figura 50 puede verse la función de autocorrelación de una secuencia de 7 bits repetida en 3 ciclos.

Conforme aumente el valor de *N* por usar registros de desplazamiento más grandes, la parte negativa de la función de autocorrelación ira tendiendo a cero.

Con respecto al espectro de potencia de la señal, de las ecuaciones [98] y [99] se obtuvo la expresión:

$$S_{xx}(0) = \frac{A^2}{N^2}$$

$$S_{xx}\left(f = \frac{k}{T}\right) = \frac{A^2(N+1)}{N^2}\frac{\sin^2(r\pi/N)}{(r\pi/N)^2}, r = \pm 1, \pm 2, \ldots$$

Como la señal PRBS es periódica, el espectro de potencia es un espectro de líneas, es decir, las frecuencias en el espectro también son discretas. La frecuencia fundamental o primer armónico es $2\pi/N\Delta t$ rad/s, que también es la separación entre líneas. Las amplitudes serán decrecientes conforme aumente la frecuencia.

En el ejemplo visualizado en la Figura 48, la amplitud de la señal es 4 y la secuencia con un tamaño de registro de 3 bits es de 7 bits de longitud. Usando las fórmulas [98] y [99] los valores que toma el espectro de frecuencia en los primeros armónicos son:

$$S_{xx}(k) = \{0.326, 2.4415, 1.9819, 1.3697, 0.77043, \ldots\}$$

Se observa que coincide con el resultado obtenido en la Figura 48. Por otra parte, el periodo de reloj es de 4mS y su periodo de *N*Tck*=28ms o su frecuencia de 35.71Hz o 224rad/s, por lo que veremos las líneas espectrales situadas en múltiplos de esta frecuencia.

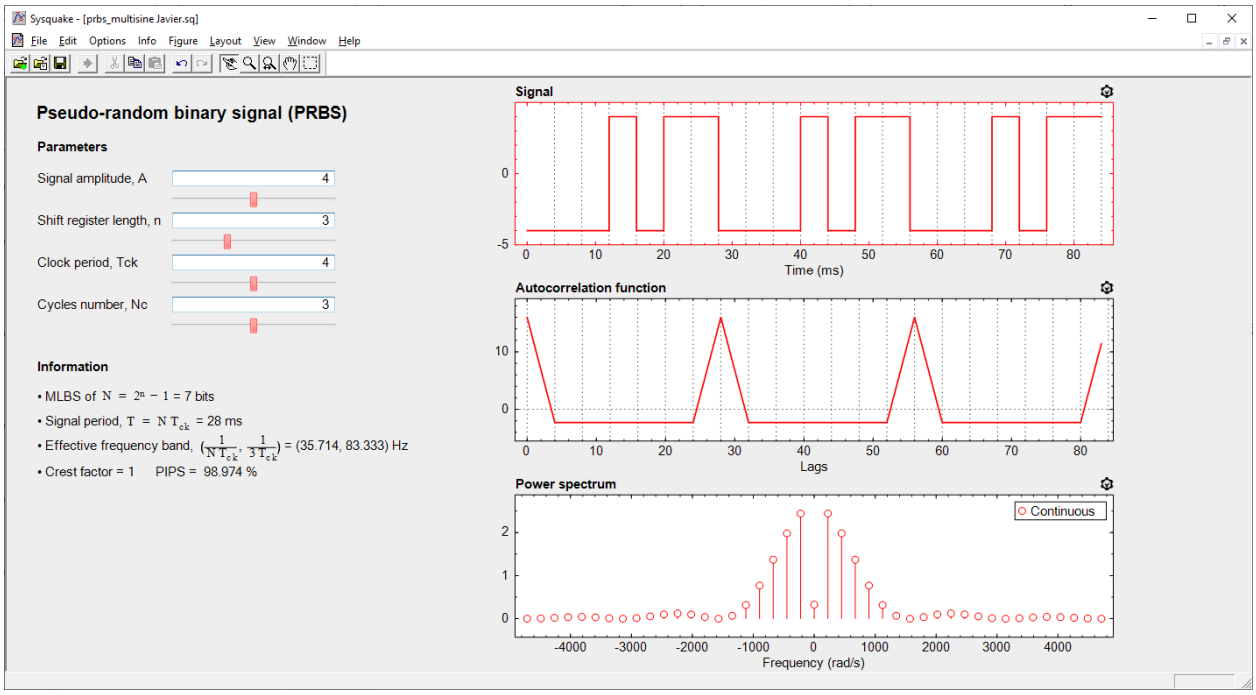

Figura 50: Detalle de la función de autocorrelación y espectro de una secuencia de 7 bits y 3 ciclos.

El ancho de banda, como vimos en la ecuación [97] de la señal ira de 1/N*Tck a 1/3*Tck, en este caso de 35.714Hz a 83.33Hz. De forma que el ancho de banda de la señal queda determinado por la longitud de la secuencia y su periodo de reloj.

Las ecuaciones [98] y [99] indican que la amplitud del espectro de potencia es proporcional al cuadrado de *A* e inversamente proporcional al cuadrado de *N*. Por lo tanto, al aumentar el tamaño del registro de desplazamiento disminuye la amplitud del espectro de potencia y al aumentar la amplitud de la señal aumenta la amplitud del espectro.

5.2 Aplicación de una señal PRBS a la identificación de sistemas

Se pretende utilizar una señal PRBS para identificar un sistema de función de transferencia desconocida. A partir de algunos experimentos iniciales, se sabe el ancho de banda de la señal necesaria tiene que ir de los 100Hz a los 500Hz. También se desea que su amplitud no supere los ±2V para no generar saturación en los amplificadores del sistema. Además, se desea que la señal tenga una duración mínima de 0.04 s.

A partir de estas condiciones iniciales determine los valores de los parámetros de la señal PRBS que cumpla con las especificaciones y configure el programa para que genere la señal necesaria.

Solución:

A partir de la ecuación [97] se puede determinar el valor del periodo de reloj *Tck* y longitud *N* de la señal:

$$500Hz = \frac{1}{3 * Tck} \quad \rightarrow \quad Tck \cong 666\mu S$$

$$100Hz = \frac{1}{N * Tck} \quad \rightarrow \quad N \geq 15$$

Como $N=2^n-1$, entonces el valor de *n* que verifica que *N* es mayor o igual a 15 es *n*=4 bits, lo que genera una secuencia de *N*=15 bits.

El periodo de la señal es, por tanto:

$$T = N * Tck = 15 * 666\mu S = 9{,}99mS \text{ s}$$

Como se desea que la señal generada tenga una duración mínima de *TF*=0.04s, entonces el número de ciclos de señal que se debe generar es

$$Nc = \frac{TF}{T} = 4.004 \rightarrow Nc = 5$$

Introduciendo estos valores en la herramienta se obtiene la señal PRBS que se muestra en la Figura 51. Se observa cómo cumple con las especificaciones de diseño dadas. En la Figura 51 el programa ha resaltado en verde el ancho de banda efectivo de la señal.

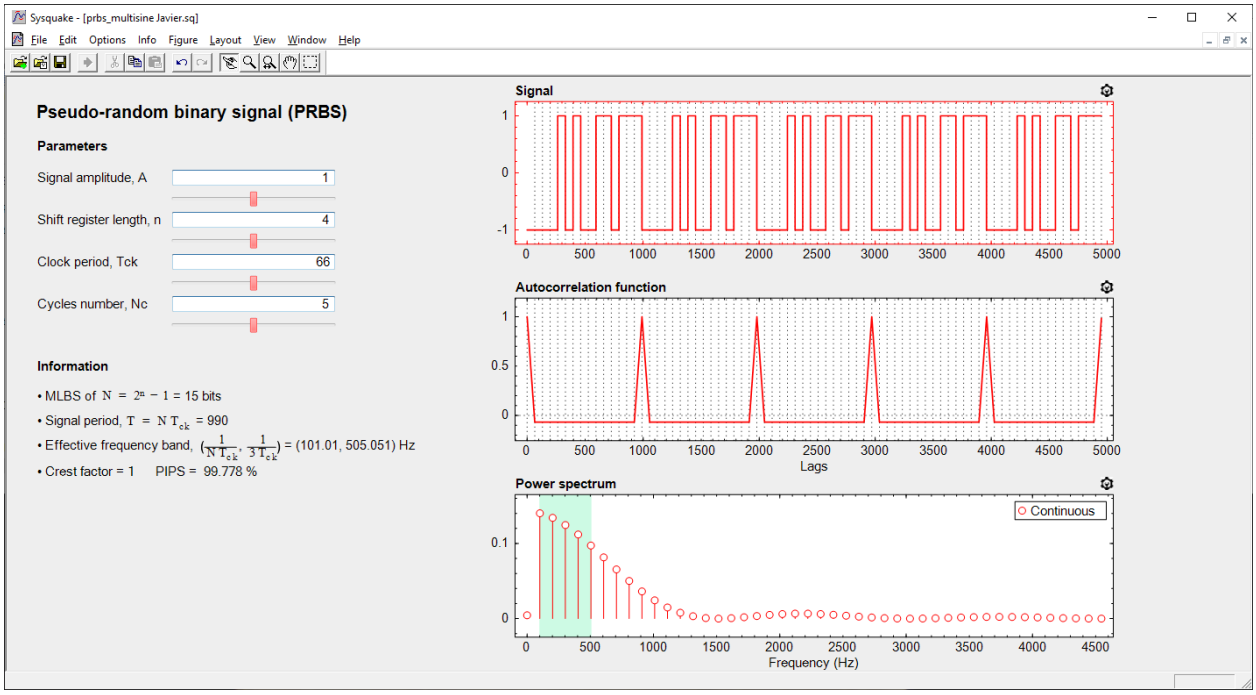

Figura 51: señal PRBS diseñada

5.3 Generación de señales multiseno

Dentro del menú *options*, submenú *signal* debe cambiar al tipo de señal *mutisine*. Una vez realizada la selección el aspecto de la pantalla principal cambiará para asemejarse al de la Figura 52.

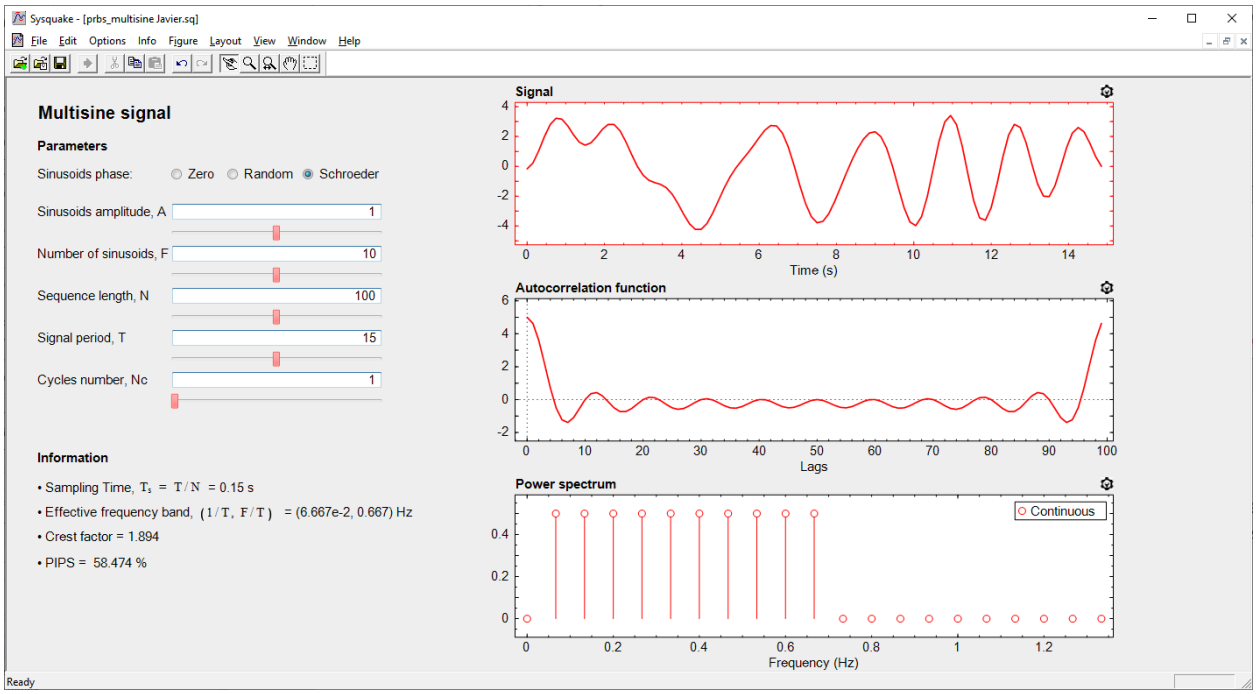

Figura 52: Pantalla de generación de señal multiseno.

Igual que en el caso de la señal PRBS puede modificar los controles para cambiar la señal generada. En este caso la función de cada uno de los controles es:

- *Sinusoids phase*: la fase inicial de las sinusoides es clave para mejorar el factor de cresta de la señal. El programa propone uno de tres algoritmos:
  - *Zero*: la fase inicial de todas las sinusoides es cero. Esta elección resulta ser la peor para el factor de cresta, ya que todas las sinusoides se suman en algún punto generando una diferencia muy grande entre el valor máximo y el valor medio de la señal.
  - *Random*: la fase inicial de todas las sinusoides es aleatoria entre 0 y 2π. El factor de cresta mejora notablemente con respecto al caso anterior, pero la señal generada no es determinista.
  - *Schroeder*: la fase inicial de las sinusoides sigue el algoritmo de Schroeder, vea sección 3.8.1. Se consigue un factor de cresta mejor al del caso aleatorio y además la señal es determinista.
- *Sinusoids amplitude, A:* Igual que en el caso de la señal PRBS con este control se modifica la amplitud de cada una de las sinusoides que forman la señal. En esta versión del programa no se pueden generar sinusoides de diferentes amplitudes.
- *Number of sinusoids:* Este control permite modificar el número de armónicos de la señal multiseno. La ecuación [120] representa a una señal multiseno genérica. El programa genera señales de la forma:

$$u(t) = \sum_{i=1}^{F} A\cos(2\pi F_{ck} i t + \phi_i), 0 \le t \le T \qquad [142]$$

  Siendo todas las señales armónicas de F_{ck} frecuencia base de la señal y F el número total de armónicos.
- *Sequence length*: Con este control se modifica el número de muestras total de la señal. Esta longitud de la secuencia debe ser al menos el doble del número de sinusoides de la señal. A mayor longitud de la secuencia, mayor precisión en la señal generada.
- *Signal period, T*: Periodo de la señal generada. Este parámetro junto con el anterior, determinan el periodo de muestreo de la señal multiseno. Además, fija la frecuencia del primer armónico de la señal que será 1/T. El ancho de banda de la señal irá desde el primer armónico 1/T al último F/T, siendo F el número total de armónicos.
- *Cicles number, Nc*: Igual que en el caso de la señal PRBS la señal se puede repetir el número de veces que indique este parámetro.
- Igual que en caso de la señal PRBS el menú *Time units* modifica la base de tiempo usada a la hora de determinar el periodo de la señal.
- En este caso el menú *Sampling Time* es meramente informativo.

En la ecuación [135] se obtuvo una expresión para la función de autocorrelación, que tendrá la misma forma que la señal multiseno con todas sus fases a 0, vea Figura 53.

De la citada ecuación [135], se puede deducir que en el caso del instante inicial todos los cosenos serán 1

$$R_{xx}(0) = \sum_{i=1}^{F} \frac{1}{2} A_i^2 cos(2\pi f_i 0) = \sum_{i=1}^{F} \frac{1}{2} A_i^2 \qquad [143]$$

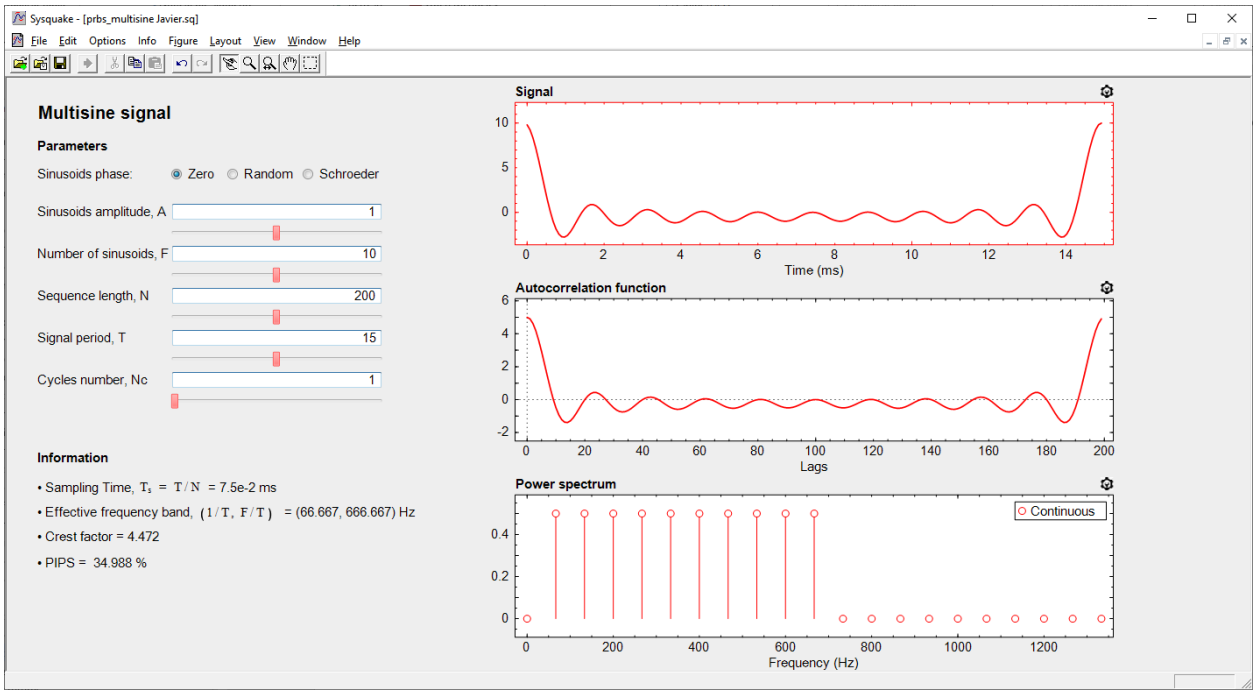

Figura 53: Señal multiseno con todas las fases nulas.

que será el valor máximo de la función de autocorrelación. En el caso de la Figura 53, al tomar *A* el valor de 1 y *F*=10, tendremos 10 veces la suma de ½, de forma que el máximo de la función de autocorrelación será 5.

En el caso del espectro de la señal, la ecuación [141] nos indicaba que la amplitud de cada uno de los armónicos de la señal es $\frac{A_i^2}{4}$ en el caso de un espectro completo, o sea mostrando tanto la parte negativa como la positiva. Si por el contrario solo se muestra la parte positiva del espectro, como en la Figura 53, la amplitud será $\frac{A_i^2}{2}$, o sea 0.5 ya que la amplitud *A* es 1. La señal tiene el mismo número de armónicos que de sinusoides el primero de ellos a frecuencia $1/T$ y el último a frecuencia F/T.

5.4 Aplicación de una señal multiseno a la identificación de un sistema

Se pretende utilizar una señal multiseno para identificar un sistema de función de transferencia desconocida. A partir de algunos experimentos iniciales, sabemos que el ancho de banda de la señal necesaria tiene que ir de los 100Hz a los 2000Hz. También queremos que la señal tenga una duración mínima de 0.1s y que su amplitud no supere los ±2V para no generar saturación en los amplificadores del sistema.

A partir de estas condiciones iniciales determine cuál es la señal más adecuada y configure el programa para que genere la señal necesaria.

Solución:

Como se ha visto en la sección anterior la frecuencia inferior del ancho de banda fija el periodo de la señal

$$F_{min} = \frac{1}{T} \rightarrow T = \frac{1}{100Hz} = 0.01 \text{ s}$$

Por tanto, habrá que fijar las unidades de tiempo en *centiseconds* y el periodo T a 1.
La frecuencia más alta del ancho de banda determina el número total de sinusoides.

$$F_{max} = \frac{F}{T} \quad \rightarrow \quad F = T * F_{max} = 0.01 * 2000 = 20 \text{ Hz}$$

De forma que el número de sinusoides será de 20.

Como se desea que la señal generada tenga una duración mínima de TF=0.1s, entonces el número de ciclos de señal que se debe generar es

$$Nc = \frac{TF}{T} = \frac{0.1}{0.01} = 10$$

Según el teorema del muestreo la frecuencia de muestreo deberá ser mayor al doble de la frecuencia máxima del ancho de banda de la señal, en este caso 4kHz o 0.25ms. Al fijar el periodo de muestreo y conocer el periodo de la señal, se podrá determinar el número de muestras mínimo de la señal.

$$T_D = \frac{T}{N} \quad \rightarrow \quad N = \frac{T}{T_D} = \frac{0.01}{0.00025} = 40$$

El programa solo admitirá valores mayores a 40.

Usando el algoritmo de Schroeder se consigue el menor valor de factor de cresta de los tres posibles algoritmos del programa. Esto ayuda a minimizar la amplitud máxima de la señal.
La ecuación [5] muestra la definición para el valor de cresta.

$$u_{cresta} = \frac{u_{pico}}{u_{rms}}$$

La complejidad de esta ecuación reside en calcular el valor de la tensión rms de la señal multiseno para obtener el valor de la tensión de pico. De forma que el carácter interactivo de la herramienta permitirá buscar fácilmente un valor de señal que no llegue a saturar el amplificador. En el caso de este ejercicio, una amplitud de 0.39 genera la señal de amplitud inferior a ±2v.

Introduciendo estos valores en la herramienta se obtiene la señal multiseno que se muestra en la Figura 54. Se observa cómo cumple con las especificaciones de diseño dadas.

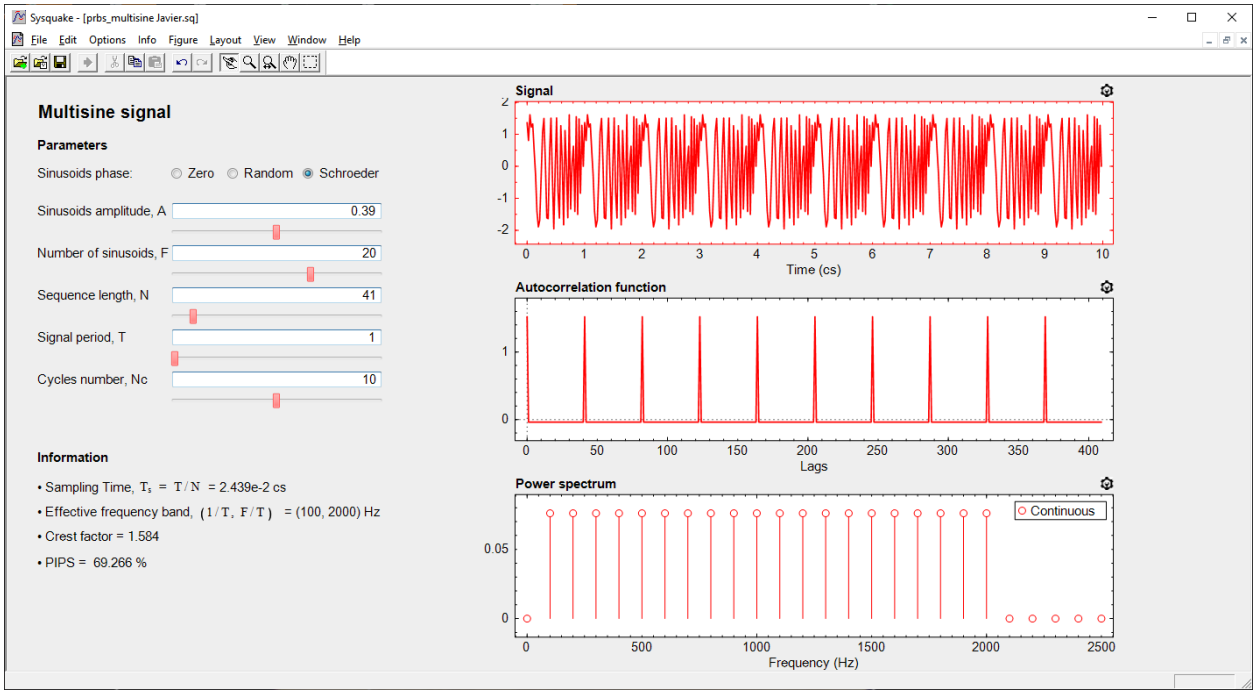

Figura 54: Señal multiseno resultado.

5.5 Diferencias de la señal usando distintas fases iniciales

En el apartado anterior se ha visto la señal generada usando el algoritmo de Schroeder. Si en vez de este algoritmo se usa el de fases aleatorias, el resultado obtenido es el de la Figura 55. La forma de la señal generada es diferente, aunque la diferencia más importante la encontramos en el factor de cresta de la nueva señal que ahora es de 1.972. Una segunda diferencia es que la señal no es determinista, si volvemos a renovar el cálculo de fase se obtiene la señal de la Figura 56 que ahora tiene un factor de cresta de 2.418 y una nueva forma de onda. Ninguna de estas señales cumplirá el requisito del límite de amplitud del ejemplo anterior y lo peor es que como cada modificación genera una señal diferente con un factor de cresta diferente será complicado realizar el ajuste del nivel máximo de la señal en caso de que sea necesario.

Si se utiliza la fase cero para generar la señal multiseno, se obtiene una señal como la de la Figura 57. Lo interesante de esta señal es que es similar a una señal impulso que puede ser útil para encontrar algunas características del sistema a identificar. En este caso el factor de cresta es el peor posible, pero la señal obtenida es determinista.

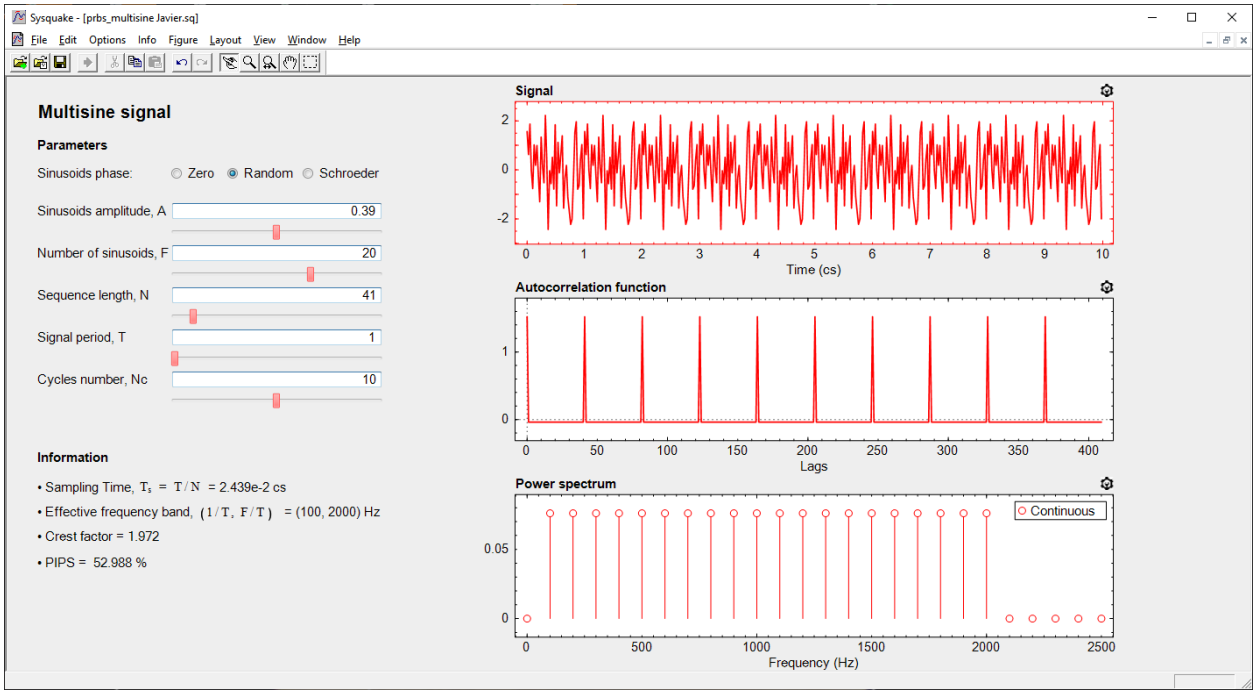

Figura 55: Señal multiseno con fase aleatoria.

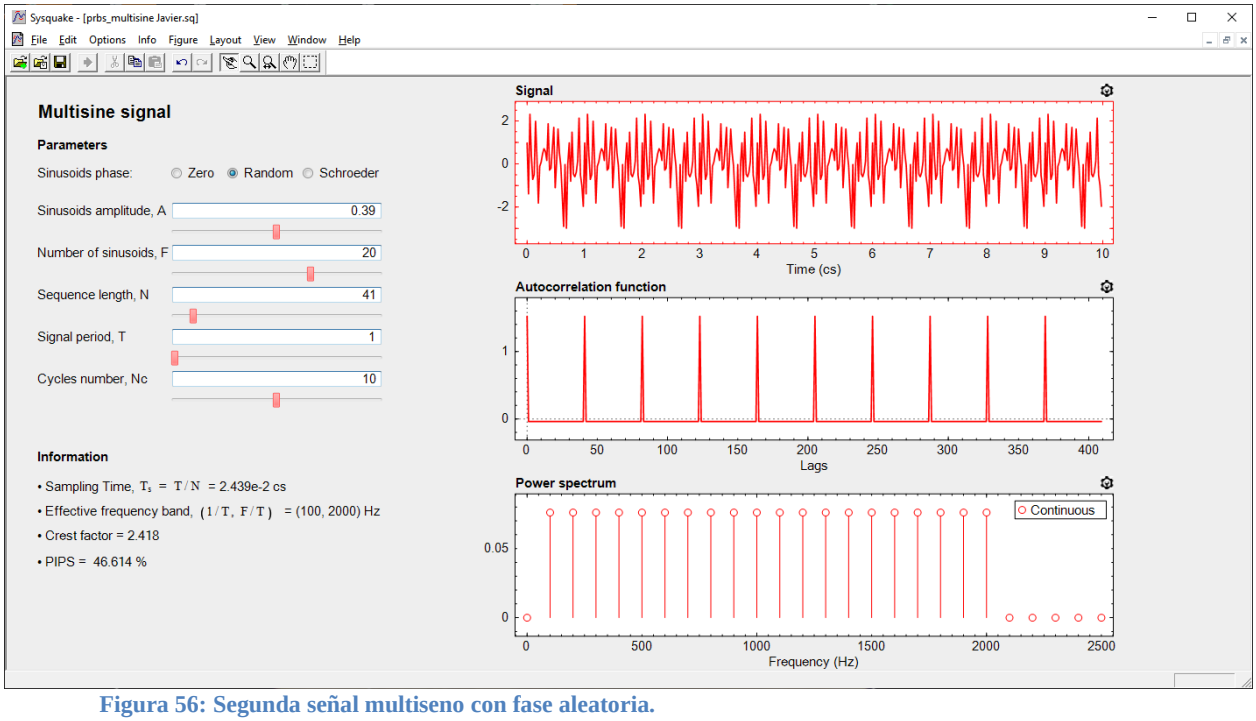

Figura 56: Segunda señal multiseno con fase aleatoria.

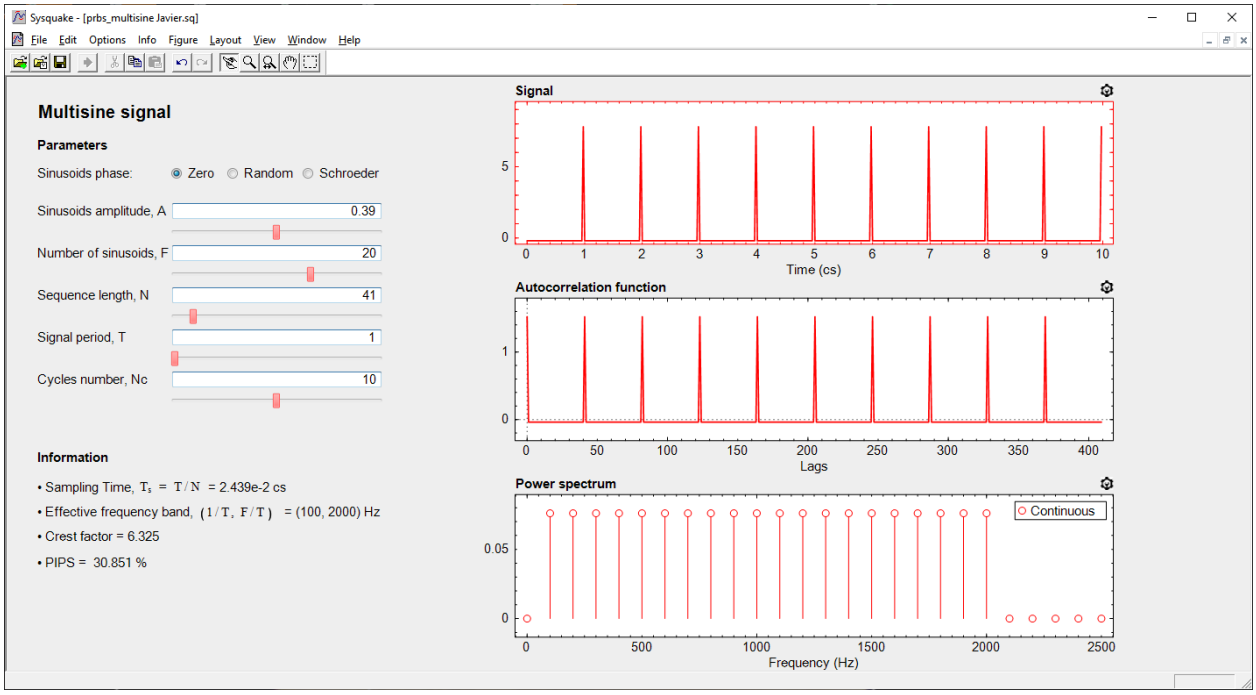

Figura 57: Señal multiseno con fase nula.

6 Conclusiones y trabajos futuros.

Después de completar este trabajo, los objetivos del proyecto fin de máster se han cumplido. Por una parte, se quería ilustrar con fines docentes las características en el dominio del tiempo y la frecuencia de las señales PRBS y multiseno. Durante el desarrollo de este TFM se han estudiado las propiedades de las funciones de autocorrelación y la expresión que toman en el caso de las señales PRBS y multiseno. También se ha estudiado la transformada de Fourier de señales continuas o discretas y el cálculo del espectro de potencia de las señales PRBS y multiseno. En el apartado 3.7 se han aplicado estos conocimientos previos a las señales PRBS estudiando cómo se generan, cuál es su función de autocorrelación y espectro. En el apartado 3.8 se ha estudiado la forma en que se generan las señales multiseno, su función de autocorrelación y espectro. Por todos estos motivos se considera cumplido el primer objetivo del TFM.

El segundo objetivo del TFM persigue diseñar un programa en Sysquake que permita generar señales de tipo PRBS y multiseno y que lo haga de forma interactiva e intuitiva. En el capítulo 4 se han estudiado los trozos de código necesarios para generar estas señales y se han comparado con las señales que genera Matlab para verificar que son correctas. Se ha definido en UML la estructura del programa y se ha realizado el programa *prbs_multisine.sq* que permite la generación de ambas señales. Como complemento se han añadido en el capítulo 5 una serie de ejemplos básicos de uso del generador de señales. Si este programa es intuitivo o no deberá determinarse con un posterior estudio de satisfacción de los usuarios.

Como complemento se han añadido dos apéndices: uno con el manual de uso del programa y otro con el código fuente para que pueda ser mejorado por cualquier usuario del mismo.

Por todo esto, se consideran cumplidos los objetivos propuestos inicialmente al comenzar el TFM. Además de estos logros también se ha comenzado a estudiar el proceso de identificación de sistemas, situando la generación de señales de entrada o perturbación como la fase inicial de este proceso. Otro elemento analizado en este trabajo es el de la importancia del uso de técnicas interactivas para mejorar la experiencia del alumno en las clases.

No obstante, dada la limitación temporal de este trabajo, han quedado fuera algunos aspectos que deberán ser desarrollados posteriormente durante la preparación de la tesis. Estos se detallan en el siguiente apartado.

6.1 Trabajos futuros

Hay una serie de mejoras sobre la herramienta que no se han podido implementar en esta primera versión. A continuación, se citan estas mejoras:

- Que la interfaz gráfica de la herramienta también se visualice correctamente en sistemas Mac OS X. Actualmente solo se visualiza correctamente en Windows.
- Añadir un menú Sesión que permita guardar la sesión de trabajo actual con los valores de configuración actuales y recuperar sesiones de trabajo previas almacenadas en el disco del ordenador, así como restaurar la herramienta a una configuración inicial.
- Crear informes a partir de los valores de configuración de la señal
- Exportar las señales creadas para poder usarlas en Matlab o como entrada de otros programas hechos con Sysquake.
- Añadir más interactividad a las figuras, como por ejemplo:
    - Que al situar el cursor encima de un punto de una gráfica muestre cuales son las coordenadas del punto.

    - En la gráfica del espectro de potencia, que el usuario pueda arrastrar los límites de la banda efectiva de frecuencia y se configuren automáticamente los valores de los parámetros de las señales.
- En el caso de las señales PRBS añadir un control que permitiera invertir la señal
- En el caso de las señales PRBS añadir un diagrama que permitiera ver cuál es la estructura del registro de desplazamiento que se está utilizando para generar la señal PRBS y cuáles son las conexiones de realimentación y valor inicial. Así mismo, que este valor inicial del registro se pudiera modificar y ver reflejado en la salida el efecto de este cambio.
- En el caso de las señales multiseno poder determinar de forma independiente la amplitud de cada una de las componentes de la señal.
- En el caso de las señales multiseno, poder generar la fase de las señales siguiendo el procedimiento de minimizar la norma l_P, para garantizar el valor mínimo posible del factor de cresta y máximo de PIPS.
- En el caso de las señales multiseno, al trabajar en discreto poder generar la señal de forma que atravesase un retenedor de orden 0 y analizar el efecto que tiene esto sobre el espectro de la señal y la función de autocorrelación.

Con respecto a los ejemplos de uso de la herramienta generados en el capítulo 5 se pueden completar con otro conjunto de ejercicios avanzados y problemas para resolver.

Con respecto a la metodología de aprendizaje interactivo se deberá diseñar un experimento que permita validar dicha metodología en el aprendizaje de la identificación de sistemas usando Sysquake.

Bibliografía

Åström, K. J. (1995). Computer-Aided Control Engineering. En M. K. Masten (Ed.), *Modern Control Systems*. IEEE - Institute of Electrical and Electronics Engineers Inc.

Åström, K. J. (2008). Computer Aided Modeling, Analysis and Design of Control Systems—A Perspective. *IEEE Control Systems*, *28*(2), 14-18. https://doi.org/10.1109/MCS.2008.915312

Baldini, P. N., Calandrini, G. L., Doñate, P. D. & Bambill, H. R. (2015a). *Introducción Temprana de Conceptos de Control Robusto : Experiencia Práctica y CAD*. *3*, 61-67.

Baldini, P. N., Calandrini, G. L., Doñate, P. D. & Bambill, H. R. (2015b). Uso de Software Interactivo como facilitador para la introducción temprana de conceptos de control robusto. *VAEP-RITA*, *3*(2), 61-67.

Baldini, P. N., Calandrini, G. L., Doñate, P. D. & Bambill, H. R. (2016). Early Introduction of Robust Control Concepts: Hands-On Experience and CAD. *Revista Iberoamericana de Tecnologias del Aprendizaje*, *11*(2), 56-62. https://doi.org/10.1109/RITA.2016.2553978

Ballagas, R., Rohs, M. & Sheridan, J. (2004). Byod: Bring your own device. En *vs.inf.ethz.ch*.

Balmer, L. (1991). *Signals and systems: an introduction*. Prentice Hall.

Barker, H. A. (1967). Choice of pseudorandom binary signals for system identification. *Electronics Letters*, *3*(11), 524-526. https://doi.org/10.1049/el:19670412

Barker, H. A. (2000). GALOIS - A Program for Generating Pseudo-Random Perturbation Signals. *IFAC Proceedings Volumes*, *33*(15), 505-507. https://doi.org/10.1016/s1474-6670(17)39800-2

Bell, E. T. & Polya, G. (1945). How to Solve It. A New Aspect of Mathematical Method. *The American Mathematical Monthly*, *52*(10), 575. https://doi.org/10.2307/2306109

Bendat, J. S. & Piersol, A. G. (1966). *Measurement and analysis of random data*. Wiley.

Bonwell, C. C. & Eison, J. A. (1991). Active Learning : Creating Excitement in the Classroom. En *Learning* (1.ª ed.). ERIC Clearinghouse on Higher Education.

Boyd, S. (1986). Multitone signals with low crest factor. *IEEE transactions on circuits and Systems*, *33*, 1018-1022.

Braatz, R. D. (2013). Teaching mathematics to control engineers. *IEEE Control Systems*, *33*(3), 66-67. https://doi.org/10.1109/MCS.2013.2249471

Briggs, P. A. N. & Godfrey, K. R. (1966). Pseudorandom signals for the dynamic analysis of multivariable systems. *Proceedings of the Institution of Electrical Engineers*, *113*(7), 1259-1267. https://doi.org/10.1049/piee.1966.0213

Brown, R. G. & Thaler, G. J. (1953). *Servo Mechanism Analysis*. McGraw-Hill Education.

Calerga. (2022a). *Sysquake*. https://www.calerga.com/products/Sysquake/index.html

Calerga. (2022b). *Sysquake – Documentation*. https://calerga.com/doc/index.html

Charters, P. (2009). Generalizing binary quadratic residue codes to higher power residues over larger fields. *Finite Fields and their Applications*, *15*(3), 404-413. https://doi.org/10.1016/j.ffa.2009.02.001

Chisholm, R. M. (1989). *Theory fo knowledge* (Third). Prentice-Hall international, inc.

Clark, R. N. (1962). *Introduction to Automatic Control Systems*. Wiley.

Cohen, M. D. & Tellez, K. (1994). Implementing Cooperative Learning for Language Minority Students. *Bilingual Research Journal*, *18*(1-2), 1-19. https://doi.org/10.1080/15235882.1994.10162655

Cooper, G. R. & Mcgillem, C. D. (1997). *Probabilistic Methods of Signal and System Analysis* (A. S. Sedra & M. R. Lightner (eds.); 3.ª ed.). Oxford University Press.

Cooper, J. (1990). *Cooperative Learning and College Instruction: Effective Use of Student*

Learning Teams.

Costa-Castelló, R., Carrero, N., Dormido Bencomo, S. & Fossas, E. (2018). Teaching, Analyzing, Designing and Interactively Simulating Sliding Mode Control. *IEEE Access*, *6*, 16783-16794. https://doi.org/10.1109/ACCESS.2018.2815043

Costin, F. (1972). Lecturing versus Other Methods of Teaching: A Review of Research. *British Journal of Educational Technology*, *3*(1), 4-31. https://doi.org/10.1111/j.1467-8535.1972.tb00570.x

Dai, Z., Gong, G. & Song, H. Y. (2009). A trace representation of binary Jacobi sequences. *Discrete Mathematics*, *309*(6), 1517-1527. https://doi.org/10.1016/j.disc.2008.02.024

Darby, M. L. & Nikolaou, M. (2012). MPC: Current practice and challenges. *Control Engineering Practice*, *20*(4), 328-342. https://doi.org/10.1016/j.conengprac.2011.12.004

Darnell, M. (1993). Periodic and non-periodic, binary and multi-level pseudo-random signals. En *Perturbation signals for system Identification* (1.ª ed.). Prentice Hall.

Dewey, J. (1983). *The middle works, 1899-1924*.

Díaz Martínez, J. M. (2020). *SISO-QFTIT*. UNED. https://www2.uned.es/itfe/QFTIT/

Díaz Martínez, J. M., Costa-Castelló, R. & Dormido Bencomo, S. (2018). *ITCLSD*. UNED. https://www2.uned.es/itfe/ITCLSD/ITCLSD.html

Díaz Martínez, J. M., Costa-Castelló, R. & Dormido Bencomo, S. (2019). Closed-Loop Shaping Linear Control System Design: An Interactive Teaching/Learning Approach [Focus on Education]. *IEEE Control Systems*, *39*(5), 58-74. https://doi.org/10.1109/MCS.2019.2925255

Díaz Martínez, J. M., Costa-Castelló, R. & Dormido Bencomo, S. (2020). *RCLSD*. UNED. https://www2.uned.es/itfe/RCLSD/RCLSD.html

Díaz Martínez, J. M., Costa-Castelló, R. & Dormido Bencomo, S. (2021). An Interactive Software Tool to Learn/Teach Robust Closed-Loop Shaping Control Systems Design. *IEEE Access*, *9*, 125805-125821. https://doi.org/10.1109/ACCESS.2021.3111129

Díaz Martínez, J. M. & Dormido Bencomo, S. (2007). An interactive approach to template generation in QFT methodology. *e-spacio.uned.es*.

Díaz Martínez, J. M. & Dormido Bencomo, S. (2017). *LCSD*. UNED. https://www2.uned.es/itfe/LCSD/LCSD.html

Díaz Martínez, J. M., Dormido Bencomo, S. & Aranda, J. (2006). *TIG*. UNED. https://www2.uned.es/itfe/TIG/main.html

Díaz Martínez, J. M., Dormido Bencomo, S. & Aranda, J. (2008). An interactive approach to template generation in quantitative feedback theory methodology. *Asian Journal of Control*, *10*(3), 361-367. https://doi.org/10.1002/ASJC.29

Díaz Martínez, J. M., Dormido Bencomo, S., Aranda, J., Dormido-Canto, S., Chaos, D. & Muñoz, R. (2006). Interactive software tools for robust control : application to marine systems. En *Chaos* (pp. 21-42).

Díaz Martínez, J. M., Dormido Bencomo, S., Aranda, J., Muñoz, R. & Chaos, D. (2007). Interactive generation of plant templates for robust control. *Proceedings of the American Control Conference*, 212-217. https://doi.org/10.1109/ACC.2007.4282627

Díaz Martínez, J. M., Dormido Bencomo, S. & Costa-Castelló, R. (2020). An interactive teaching/learning approach to the design of robust linear control systems using the closed-loop shaping methodology. *IFAC-PapersOnLine*, *53*(2), 17174-17178. https://doi.org/10.1016/j.ifacol.2020.12.1726

Díaz Martínez, J. M., Dormido Bencomo, S. & Rivera, D. E. (2014a). *ITTSAE-TSA*. UNED. https://www2.uned.es/itfe/ITTSAE/ITTSAE_TSA.html

Díaz Martínez, J. M., Dormido Bencomo, S. & Rivera, D. E. (2014b). *ITTSAE-TSG*. UNED.

https://www2.uned.es/itfe/ITTSAE/ITTSAE_TSG.html

Díaz Martínez, J. M., Dormido Bencomo, S. & Rivera, D. E. (2014c). *ITTSAE*. UNED. https://www2.uned.es/itfe/ITTSAE/ITTSAE.html

Díaz Martínez, J. M., Dormido Bencomo, S. & Rivera, D. E. (2015). Interactive Education for Time-Domain Time Series Analysis using ITTSAE. *IFAC-PapersOnLine*, *48*(28), 751-756. https://doi.org/10.1016/j.ifacol.2015.12.220

Díaz Martínez, J. M., Dormido Bencomo, S. & Rivera, D. E. (2016). ITTSAE: A set of interactive software tools for time series analysis education. *IEEE Control Systems*, *36*(3), 112-120. https://doi.org/10.1109/MCS.2016.2535914

Dickinson, G. & Verner, C. (1967). Attendance Patterns and Dropouts in Adult Night School Classes. *Adult Education Quarterly*, *18*(1), 24-33. https://doi.org/10.1177/074171366701800103

Dormido Bencomo, S. (2002). Control learning: Present and future. En *IFAC Proceedings Volumes (IFAC-PapersOnline)* (Vol. 15, Número 1). IFAC. https://doi.org/10.3182/20020721-6-es-1901.01641

Dormido Bencomo, S. (2004). Control learning: Present and future. *Annual Reviews in Control*, *28*(1), 115-136. https://doi.org/https://doi.org/10.1016/j.arcontrol.2003.12.002

Doyle, F. J., Parker, R. S., Pearson, R. K. & Ogunnaike, B. A. (2015). Plant-friendly identification of second-order Volterra models. *European Control Conference, ECC 1999 - Conference Proceedings*, 1096-1101. https://doi.org/10.23919/ecc.1999.7099455

Eckman, D. P. (1958). *Automatic Process Control*. Wiley. https://doi.org/10.1016/S0065-2377(08)60225-2

Egidi, L. & Manzini, G. (2013). Better spaced seeds using Quadratic Residues. *Journal of Computer and System Sciences*, *79*(7), 1144-1155. https://doi.org/10.1016/j.jcss.2013.03.002

Elgerd, O. I. (1967). Control Systems Theory. En *McGraw-Hill electrical and electronic engineering series*.

Esquembre, F. (2004). Easy Java Simulations: A software tool to create scientific simulations in Java. *Computer Physics Communications*, *156*(2), 199-204. https://doi.org/10.1016/S0010-4655(03)00440-5

Fisher, R. A. (1925). Theory of Statistical Estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, *22*(5), 700-725. https://doi.org/10.1017/S0305004100009580

Fraas, J. W. (1980). The use of seven simulation games in a college economics course. *Journal of Experimental Education*, *48*(4), 264-280. https://doi.org/10.1080/00220973.1980.11011744

Gardner, M. F. & Barnes, J. L. D. W. (1943). Transients in linear systems, studied by the Laplace transform. *Bulletin of the American Mathematical*, *1*.

Gersho, A., Gopinath, B. & Odlyzko, A. M. (1979). Coefficient Inaccuracy in Transversal Filtering. *Bell System Technical Journal*, *58*(10), 2301-2316. https://doi.org/10.1002/j.1538-7305.1979.tb02968.x

Gibson, J. E. (1963). *Nonlinear automatic control*. McGraw-Hill Education.

Godfrey, K. R. (1969). The Theory of the Correlation Method of Dynamic Analysis and its Application to Industrial Processes and Nuclear Power Plant. *Measurement and Control*, *2*(5), T65-T72. https://doi.org/10.1177/002029406900200506

Godfrey, K. R. (1993). *Introduction to perturbation signals for time-domain systems identification*. Prentice Hall.

Godfrey, K. R., Barker, H. A. & Tucker, A. J. (1999). Comparison of perturbation signals for linear system identification in the frequency domain. *IEE Proceedings: Control Theory and Applications*, *146*(6), 535-548. https://doi.org/10.1049/ip-cta:19990529

Godfrey, K. R., Tan, A. H., Barker, H. A. & Widanage, W. D. (2006). A SUITE OF WEB-BASED PROGRAMS FOR PERTURBATION SIGNAL DESIGN. *IFAC Proceedings Volumes*, *39*(1), 696-701. https://doi.org/10.3182/20060329-3-au-2901.00108

González, R. A., Rojas, C. R., Pan, S. & Welsh, J. S. (2021). Consistent identification of continuous-time systems under multisine input signal excitation. *Automatica*, *133*, 109859. https://doi.org/10.1016/j.automatica.2021.109859

Graham, D. & McRuer, D. (1962). Analysis of Nonlinear Control Systems. *Journal of the American Statistical Association*, *57*(298), 526. https://doi.org/10.2307/2281680

Guillaume, P., Schoukens, J., Pintelon, R. & Kollái, I. (1991). Crest-Factor Minimization Using Nonlinear Chebyshev Approximation Methods. *IEEE Transactions on Instrumentation and Measurement*, *40*(6), 982-989. https://doi.org/10.1109/19.119778

Guzmán Sánchez, J. L., Åström, K. J., Dormido Bencomo, S., Hägglund, T. & Piguet, Y. (2006). Interactive learning modules for PID control. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *7*(PART 1), 7-12. https://doi.org/10.3182/20060621-3-ES-2905.00003

Guzmán Sánchez, J. L., Berenguel Soria, M. & Dormido Bencomo, S. (2005). Interactive teaching of constrained generalized predictive control. *IEEE Control Systems*, *25*(2), 52-66. https://doi.org/10.1109/MCS.2005.1411384

Guzmán Sánchez, J. L., Costa-Castelló, R., Berenguel Soria, M. & Dormido Bencomo, S. (2012). *Control automático con herramientas interactivas*. Pearson Educación.

Guzmán Sánchez, J. L., Costa-Castelló, R., Dormido Bencomo, S. & Berenguel Soria, M. (2016). An Interactivity-Based Methodology to Support Control Education: How to Teach and Learn Using Simple Interactive Tools [Lecture Notes]. *IEEE Control Systems*, *36*(1), 63-76. https://doi.org/10.1109/MCS.2015.2495092

Guzmán Sánchez, J. L., Dormido Bencomo, S. & Berenguel Soria, M. (2013). Interactivity in education: An experience in the automatic control field. *Computer Applications in Engineering Education*, *21*(2), 360-371. https://doi.org/10.1002/cae.20480

Guzmán Sánchez, J. L., Rivera, D. E., Dormido Bencomo, S. & Berenguel Soria, M. (2009). ITSIE: An Interactive Software Tool for System Identification Education. *IFAC Proceedings Volumes*, *42*(10), 752-757. https://doi.org/10.3182/20090706-3-fr-2004.00125

Guzmán Sánchez, J. L., Rivera, D. E., Dormido Bencomo, S. & Berenguel Soria, M. (2012). An interactive software tool for system identification. *Advances in Engineering Software*, *45*(1), 115-123. https://doi.org/10.1016/J.ADVENGSOFT.2011.09.013

Johansson, M., Gafvert, M. & Åström, K. J. (1998). Interactive Tools for Education in Automatic Control. *IEEE Control Systems*, *18*(3), 33-40. https://doi.org/10.1109/37.687617

Johnston, J. & Gardner, S. (1989). *The Electronic classroom in higher education : a case for change*. National Center for Research to Improve Postsecondary Teaching and Learning.

Kollár, I., Pintelon, R., Rolain, Y., Schoukens, J. & Simon, G. (2003). Frequency Domain System Identification Toolbox for MATLAB: Automatic Processing - From Data to Models. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *36*(16), 1459-1463. https://doi.org/10.1016/S1474-6670(17)34966-2

Kollár, I., Pintelon, R. & Schoukens, J. (1994). Frequency Domain System Identification Toolbox. *IFAC Proceedings Volumes*, 8--12.

Kschischang, F. R. (2017). *The Wiener-Khinchin Theorem*.

Kulik, J. A. & Kulik, C. L. C. (1987). Review of recent research literature on computer-based instruction. En *Contemporary Educational Psychology* (Vol. 12, Número 3, pp. 222-230). https://doi.org/10.1016/S0361-476X(87)80027-9

Lee, C. D., Huang, Y. P., Chang, Y. & Chang, H. H. (2015). Perfect Gaussian integer sequences of odd period 2m - 1. *IEEE Signal Processing Letters*, *22*(7), 881-885.

https://doi.org/10.1109/LSP.2014.2375313

Liu, T. & Gao, F. (2012). *Industrial Process Identification and Control Design*. Springer International Publishing. https://doi.org/10.1007/978-3-030-03661-4

Ljung, L., Zhang, Q., Lindskog, P. & Juditsky, A. (2012). *System identification toolbox: User's guide* (1.ª ed.). The MathWorks, Inc.

Luis, J., Carlos, J., Berenguel Soria, M., Rodriguez, F. & Sanchez-Hermosill, J. (2011). A Frequency Domain Quantitative Technique for Robust Control System Design. En *Robust Control, Theory and Applications*. https://doi.org/10.5772/16027

Macgregor, S. K., Shapiro, J. Z. & Niemiec, R. (1988). Effects of a Computer-Augmented Learning Environment on Math Achievement for Students with Differing Cognitive Style. *Journal of Educational Computing Research*, *4*(4), 453-465. https://doi.org/10.2190/nbld-3eb6-4w47-yvgb

Marin, L., Vargas, H., Heradio, R., de la Torre, L., Díaz Martínez, J. M. & Dormido Bencomo, S. (2020). Evidence-based control engineering education: Evaluating the LCSD simulation tool. *IEEE Access*, *8*, 170183-170194. https://doi.org/10.1109/ACCESS.2020.3023910

MathWorks. (2022). *Generate input signals to support system identification - MATLAB idinput - MathWorks España*. https://es.mathworks.com/help/ident/ref/idinput.html

Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, *50*(12), 2967-2986. https://doi.org/10.1016/j.automatica.2014.10.128

McCormack, A. S., Godfrey, K. R. & Flower, J. O. (1995). Design of multilevel multiharmonic signals for system identification. *IEE Proceedings: Control Theory and Applications*, *142*(3), 247-252. https://doi.org/10.1049/ip-cta:19951844

McLeod, M. N. & Fisher, P. (1978). Pavor nocturnus in a schizophrenic patient: a review and case study. *American Journal of Psychiatry*, *135*(2), 235-236. https://doi.org/10.1176/ajp.135.2.235

Miller, S. & Welsh, K. E. (2017). Bring Your Own Device (BYOD) in higher education: Opportunities and challenges. En *Mobile Learning: Students' Perspectives, Applications and Challenges* (pp. 133-164).

Milton, O. & Penner, J. G. (1986). Why Many College Teachers Cannot Lecture: How to Avoid Communication Breakdown in the Classroom. *The Journal of Higher Education*, *57*(1), 115. https://doi.org/10.2307/1981473

Mwaniki, F. M. & Vermeulen, H. J. (2020). Characterization and Application of a Pseudorandom Impulse Sequence for Parameter Estimation Applications. *IEEE Transactions on Instrumentation and Measurement*, *69*(6), 3917-3927. https://doi.org/10.1109/TIM.2019.2937447

Naslin, P. (1962). Les Régimes Variables Dans Les Systémes Linéaires et non linéaires. En *The Dynamics of Linear and Non-Linear Systems*.

Newman, D. J. (1965). An L1 Extremal Problem for Polynomials. *Proceedings of the American Mathematical Society*, *16*, 1287-1290. https://doi.org/10.2307/2038582

Newton, G. C., Gould, L. A. & Kaiser, J. F. (1958). Analytical Design of Linear Feedback Controls. *The Journal of the Royal Aeronautical Society*, *62*(572), 610-610. https://doi.org/10.1017/s0368393100069479

Núñez, R. (2007). Obtención eficiente de espectros basada en un algoritmo mejorado de ventana ajustable. *Revista Mexicana de Fisica E*, *53*(1), 20-30.

Okpala, P. & Onocha, C. (1988). Student factors as correlates of achievement in physics. *Physics Education*, *23*(6), 361-364. https://doi.org/10.1088/0031-9120/23/6/311

Olalere, M., Abdullah, M. T., Mahmod, R. & Abdullah, A. (2015). A Review of Bring Your Own Device on Security Issues. *SAGE Open*, *5*(2). https://doi.org/10.1177/2158244015580372

Peterson, W. W. & Weldon, E. J. (1972). *Error-correcting codes* (Second). MIT Press.

Phillips, D. A., Phillips, P., Tanis, D. O. & Shakhashiri, B. Z. (1984). Why I Do Demonstrations / Lecture Demonstrations. *Goals*, *61*(11), 2.

Piguet, Y., Holmberg, U. & Longchamp, R. (1999). Instantaneous performance display for graphical control design methods. *IFAC Proceedings Volumes*, *32*(2), 6059-6064. https://doi.org/10.1016/S1474-6670(17)57034-2

Pillonetto, G., Chen, T., Chiuso, A., De Nicolao, G. & Ljung, L. (2022). *Regularized System Identification*. Springer International Publishing. https://doi.org/10.1007/978-3-030-95860-2

Pintelon, R. & Schoukens, J. (2001). System Idenfitication: A Frequency Domain Approach. En *Book*.

Ravn, O. (1989). On user-friendly interface construction for CACSD packages. *ieeexplore.ieee.org*, 35-40. https://doi.org/10.1109/cacsd.1989.69828

Rivera, D. E. (2007). *Introducción a la identificación de sistemas*. UNED.

Romm, T. & Mahler, S. (1986). A three dimensional model for using case studies in the academic classroom. *Higher Education*, *15*(6), 677-696. https://doi.org/10.1007/BF00141404

Sarker, M. T., Tan, A. H. & Vun Yap, T. T. (2022). Amplitude Spectrum Design for Multivariable System Identification in Open Loop. *2022 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 107-112. https://doi.org/10.1109/I2CACIS54679.2022.9815487

Schroeder, M. R. (1970). Synthesis of Low-Peak-Factor Signals and Binary Sequences with Low Autocorrelation. *IEEE Transactions on Information Theory*, *16*(1), 85-89. https://doi.org/10.1109/TIT.1970.1054411

Slavin, R. E. (1983). When does cooperative learning increase student achievement? *Psychological Bulletin*, *94*(3), 429-445. https://doi.org/10.1037/0033-2909.94.3.429

Steenis, R. & Rivera, D. E. (2011). Plant-friendly signal generation for system identification using a modified simultaneous perturbation stochastic approximation (SPSA) methodology. *IEEE Transactions on Control Systems Technology*, *19*(6), 1604-1612. https://doi.org/10.1109/TCST.2010.2089795

Stuart, J. & Rutherford, R. J. D. (1978). Medical student concentration during lectures. *The Lancet*, *312*(8088), 514-516. https://doi.org/10.1016/S0140-6736(78)92233-X

Tan, A. H. & Godfrey, K. R. (2002). The generation of binary and near-binary pseudorandom signals: An overview. *IEEE Transactions on Instrumentation and Measurement*, *51*(4), 583-588. https://doi.org/10.1109/TIM.2002.802243

Tan, A. H. & Godfrey, K. R. (2009). A guide to the design and selection of perturbation signals. *Proceedings of the IEEE Conference on Decision and Control*, 464-469. https://doi.org/10.1109/CDC.2009.5400077

Tan, A. H. & Godfrey, K. R. (2019). *Industrial Process Identification. Perturbation Signal Design and Applications.* (M. J. Grimble & M. A. Johnson (eds.); 1.ª ed.). Springer. https://doi.org/https://doi.org/10.1007/978-3-030-03661-4

Tangirala, A. K. (2014). Principles of System Identification: Theory and Practice. En *Principles of System Identification: Theory and Practice*. https://doi.org/10.1201/9781315222509

Theorin, A. & Johnsson, C. (2014). A Freely Available Interactive PID Learning Module. En *Reglermöte 2014*.

van den Bos, A. (1991). Periodic test signals - properties and use. *IEE Conference Publication*, *1*(332), 545-549.

Verner, C. & Dickinson, G. (1967). The Lecture, An Analysis and Review of Research. En *Adult Education Quarterly* (Vol. 17, Número 2, pp. 85-100). https://doi.org/10.1177/074171366701700204

Wales, C., Nardi, A. & Stager, R. (1986). *Professional decision-making*. Center for Guided Design, West Virginia University.

Winward, E., Yang, Z., Mason, B. & Cary, M. (2022). Excitation Signal Design for Generating Optimal Training Data for Complex Dynamic Systems. *IEEE Access*, *10*, 8653-8663. https://doi.org/10.1109/ACCESS.2021.3138166

Wittenmark, B., Haglund, H. & Johansson, M. (1998). Dynamic Pictures and Interactive Learning. *IEEE Control Systems*, *18*(3), 26-32. https://doi.org/10.1109/37.687616

Zierler, N. (1959). Linear Recurring Sequences. *Journal of the Society for Industrial and Applied Mathematics*, *7*(1), 31-48. https://doi.org/10.1137/0107003

7 Apéndice A: Manual de uso de la herramienta.

En este manual se va a detallar el funcionamiento de la herramienta diseñada en Sysquake para la generación de señales PRBS de tipo MLBS y multiseno. La herramienta se llama *prbs_multisine.exe* y puede ver su icono en la Figura 58.

Figura 58: Icono del programa

7.1 Menús de la aplicación

El aspecto del programa al iniciarse es el que puede ver en la Figura 59. El programa tiene tres menús: *options*, *info* y *help*.

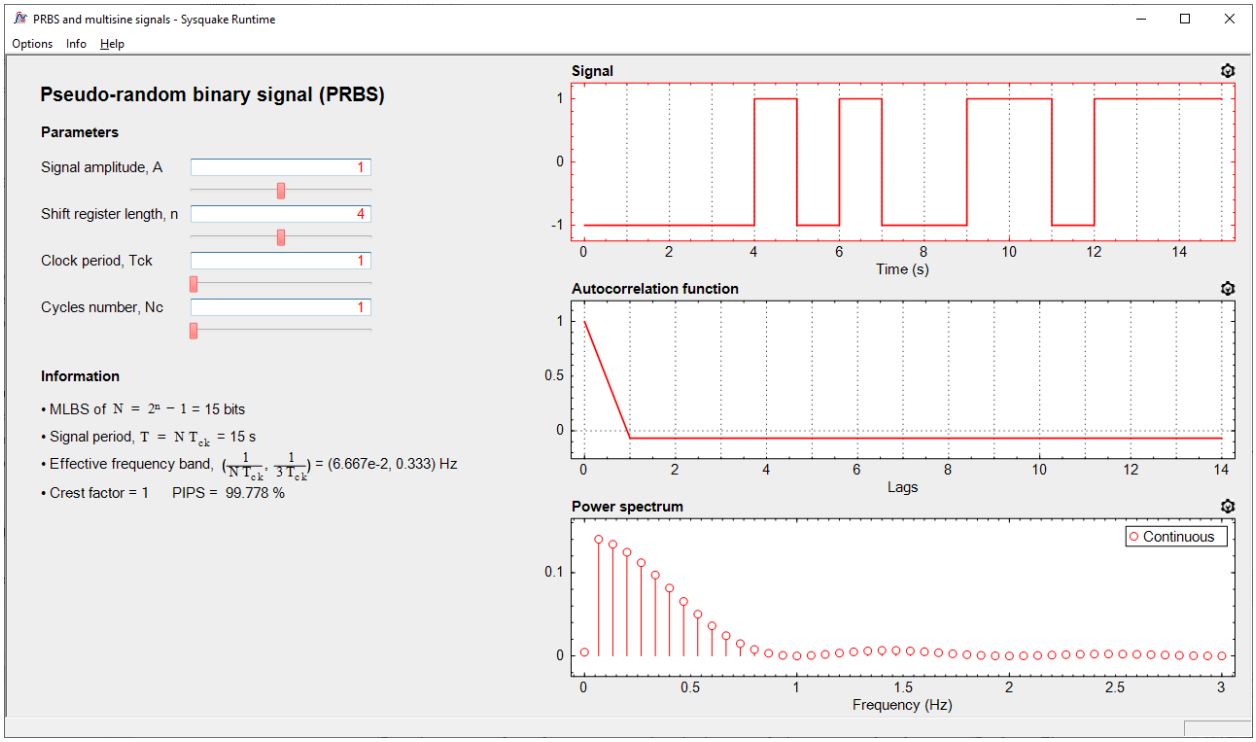

Figura 59: Pantalla inicial de la aplicación.

El menú *options* tiene 4 submenús (vea Figura 60) y tres de ellos tienen submenús adicionales. En la siguiente lista se pueden ver cada una de sus opciones.

- *Signal*: Permite escoger el tipo de señal a mostrar
  - *PRBS*: Señal binaria pseudoaleatoria de longitud máxima.

  - *Multisine*: Señal multiseno, compuesta por una suma de sinusoides de amplitud constante.
- *View*: Permite escoger en qué forma visualizamos la señal.
  - *Continuous*: Permite ver la señal como una línea continua.
  - *Discrete*: Permite ver la señal en forma discreta, solo el valor instantáneo de la señal en los instantes de muestreo.
  - *Both*: Permite ver la señal de forma continua y discreta simultáneamente.
- *Time units*: Permite escoger la unidad de tiempo:
  - *Minutes(min)*: Cada ciclo en la escala de tiempo durará 1 minuto, 60s.
  - *Seconds(s)*: Cada ciclo en la escala de tiempo durará 1 segundo, 1s.
  - *Deciseconds(ds)*: Cada ciclo en la escala de tiempo durará 1 décima de segundo 0.1s.
  - *Centiseconds(cs)*: Cada ciclo en la escala de tiempo durará 1 centésima de segundo, 0.01s.
  - *Miliseconds(ms)*: Cada ciclo en la escala de tiempo durará 1 milisegundo, 0.001s.
- *Sampling time*: En el caso de la señal PRBS, se despliega un panel de control que permite escoger el número de muestras por unidad de tiempo. De forma que entre la elección de los dos submenús (*Time units* y *Sampling time*), permite determinar la frecuencia de muestreo del sistema. Vea el panel desplegado en la Figura 61. El valor de *Ts* periodo de muestreo será el valor de la unidad temporal dividido por el número de muestras escogidas en el panel. Si, por ejemplo, tomamos 1s como unidad de tiempo y en el desplegable seleccionamos 10 muestras, tendremos un periodo de muestreo *Ts* de 0.1s o, dicho de otra forma, 10 muestras por unidad de tiempo. El submenú muestra además la frecuencia de Nyquist y el número de muestras totales de la señal generada. La frecuencia de Nyquist está relacionada con el efecto del Aliasing, para que esto no ocurra, la mayor parte del espectro de la señal debería estar por debajo de la frecuencia de Nyquist.
  En el caso de una señal multiseno, mostrado en la Figura 62, no se podrá modificar el número de muestras por unidad de tiempo, sólo se mostrará el valor del periodo de muestreo, la frecuencia de Nyquist y el número total de muestras de la señal. En este caso el periodo de muestreo que fijado por el periodo de la señal y el número de muestras.

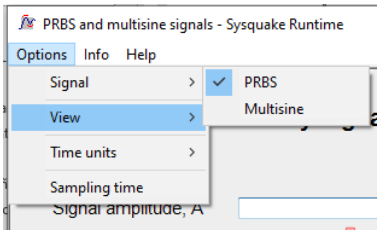

Figura 60: Opciones principales del menú *options*.

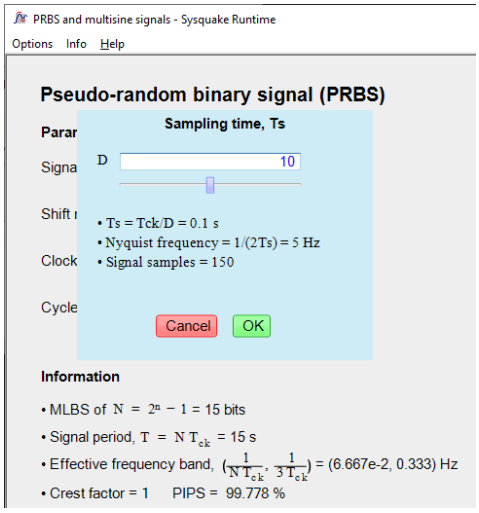

Figura 61: Submenú *sampling time*, Ts para la señal PRBS

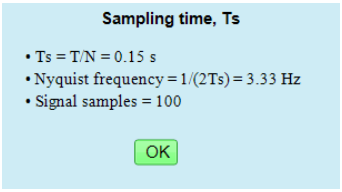

Figura 62: Submenú *sampling time*, Ts para el caso de la señal multiseno

El menú *info*, contiene información sobre los autores y un enlace a este manual de uso.

7.2 Señal PRBS

El panel de control de la señal PRBS puede verse en la Figura 59. El panel de control está dividido en tres zonas:

- *Parámeters*: En esta zona se pueden modificar los parámetros fundamentales de la señal PRBS. La modificación de todos los parámetros se puede hacer de dos formas, con un control deslizante o introduciendo el valor numérico deseado en el cuadro de texto. Véase la Figura 63
  - *Signal amplitude, A*: Este parámetro permite seleccionar la amplitud de la señal PRBS. La señal generada es simétrica con respecto a cero, de forma que, si la amplitud seleccionada es 2, se generará una señal de ±2 voltios.
  - *Shift register length, n*: Con este parámetro se determina la longitud del registro de desplazamiento empleado para generar la señal PRBS. Este a su vez determina la longitud de la señal generada 2^n-1 bits.
  - *Clock period, Tck*: Este parámetro indica el número de ciclos de reloj que dura cada bit de la señal.
  - *Cycles number, Nc*: Con este parámetro se determina el número de veces que se repetirá la señal completamente. Si, por ejemplo, se ha escogido una longitud de registro de 4 bits, lo que da origen a una secuencia de 15 bits y a Nc le damos un valor de 3. La señal obtenida tendrá una longitud total de 45 bits, o sea repetirá 3 veces el ciclo de la señal PRBS.
- *Information*: En esta zona de la pantalla se muestra la información básica de la señal generada. Vea Figura 64.

- *MLBS of N*, señal binaria de longitud máxima. En función del tamaño del registro de desplazamiento, la longitud mínima de la señal quedará establecida en 2^n-1 bits.
- *Signal period*, marca la duración en unidades de tiempo de la señal generada y será el número de bits de la señal por la duración del ciclo de reloj.
- *Effective frequency band*, banda de frecuencia efectiva de la señal. Esta banda va desde la frecuencia inferior que es el inverso del periodo de la señal hasta $1/3T_{ck}$ que marca la frecuencia en que la señal cae 3db con respecto a su valor máximo.
- *Crest factor:* factor de cresta de la señal que es igual a su valor máximo dividido por su valor rms. Vea fórmula [5] en sección 3.1.2.1.
- *PIPS:* índice de perturbación de la señal, vea fórmula [8] en sección 3.1.2.2.

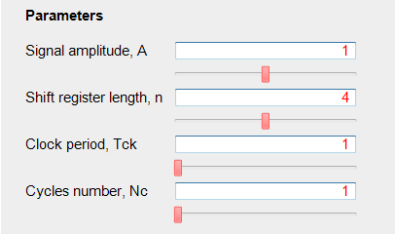

Figura 63: Zona de parámetros de la señal PRBS.

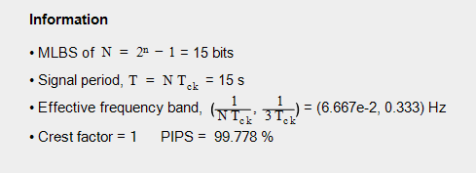

Figura 64: Zona de información de la señal PRBS.

- Gráficos: El programa presenta tres gráficos principales:
  - *Signal*: Representa la señal PRBS generada.
  - *Autocorrelation function*: Muestra la función de autocorrelación de la señal.
  - *Power spectrum*: Presenta el espectro de potencia en frecuencias de la señal generada.

  Cada uno de estos gráficos va acompañado de una ruleta de configuración, véase la Figura 65. A través de esta ruleta de configuración se pueden acceder a opciones complementarias de cada uno de los gráficos.

Figura 65: Ruleta de configuración de los gráficos de la aplicación.

En el caso del panel de configuración del gráfico *Signal*, representado en la Figura 66, hay opciones para configurar distintos elementos de la señal. Cada uno de los controles en checkbox se pueden activar de forma independiente.

- *Scale*: *Locked / Variable*, cuando la escala está en la opción *Variable* (por defecto), la escala se ajusta automáticamente en función de los datos a representar. Cuando la escala está en la opción *Locked*, se pueden usar los controles inferiores para aumentar o reducir el rango de valores mostrado en la gráfica. En la Figura 67 puede

verse un ejemplo de cómo hacer zoom en la escala temporal para apreciar con más detalle el valor de la señal entre los segundos 10 y 20.

- *Plot continuous PRBS*: Permite mostrar la señal de forma continua
- *Plot discrete PRBS*: Permite mostrar la señal de forma discreta, solo los instantes en que la señal es muestreada. Estos dos controles hacen la misma función que el menú *options* y submenú *View*, pero solo con respecto al gráfico de la señal.
- *Plot clock period grid*: oculta o muestra una rejilla donde aparecen los instantes del periodo de reloj *Tck*. En el caso de la Figura 67 pueden verse cada dos unidades de tiempo, que está fijada en décimas de segundo.
- *Distinguish cycles*: Con esta opción activada, en el caso en que el número de ciclos sea mayor que uno, el programa pondrá con un tono azulado en el fondo los ciclos pares para distinguirlos de los impares. En la Figura 68 puede verse esta opción activada para una señal de 3 ciclos.
- *Drag amplitude and clock period*: Esta opción permite modificar en el gráfico de forma interactiva la amplitud de la señal y el periodo de la señal de reloj. Tiene la misma función que los controles *Signal amplitude, A* y *Clock period, Tck*. Al activar esta opción, aparecen dos líneas de guiones rojas que permiten arrastrar y soltar modificando los valores indicados de amplitud y duración del ciclo de reloj, tanto para aumentar como para reducir su valor actual.

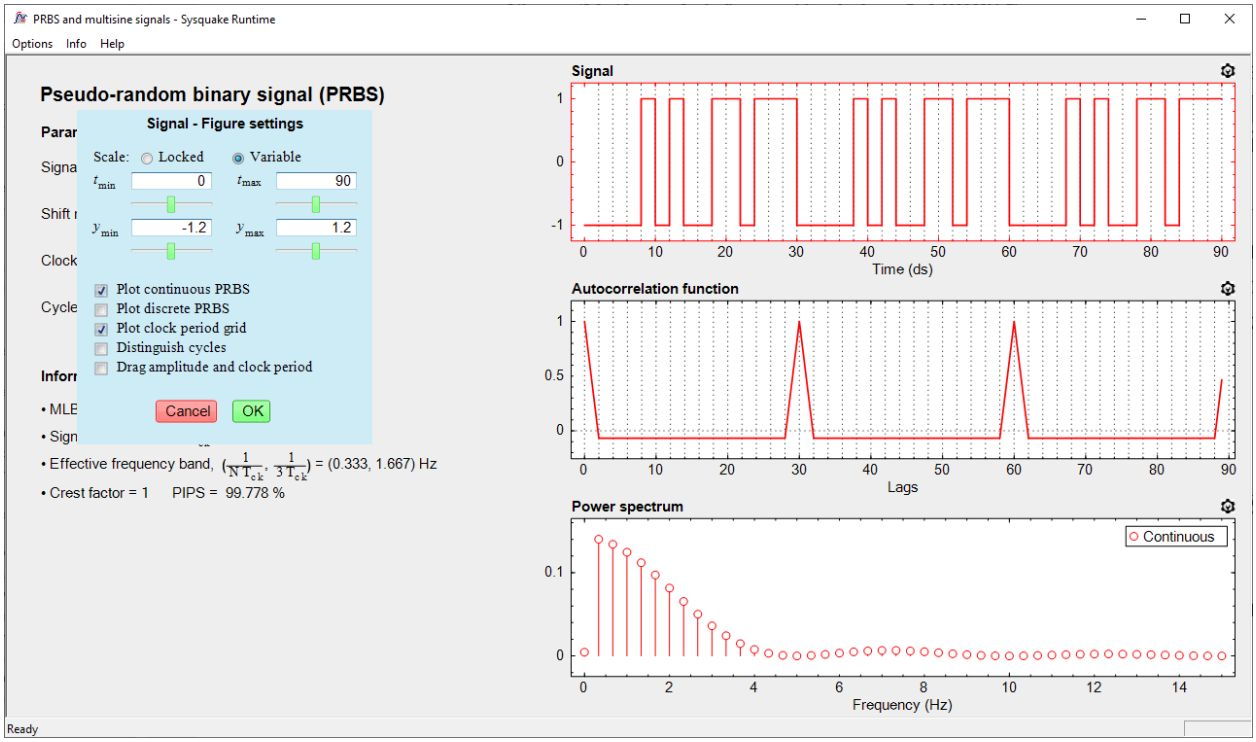

Figura 66: Panel de configuración del gráfico Signal.

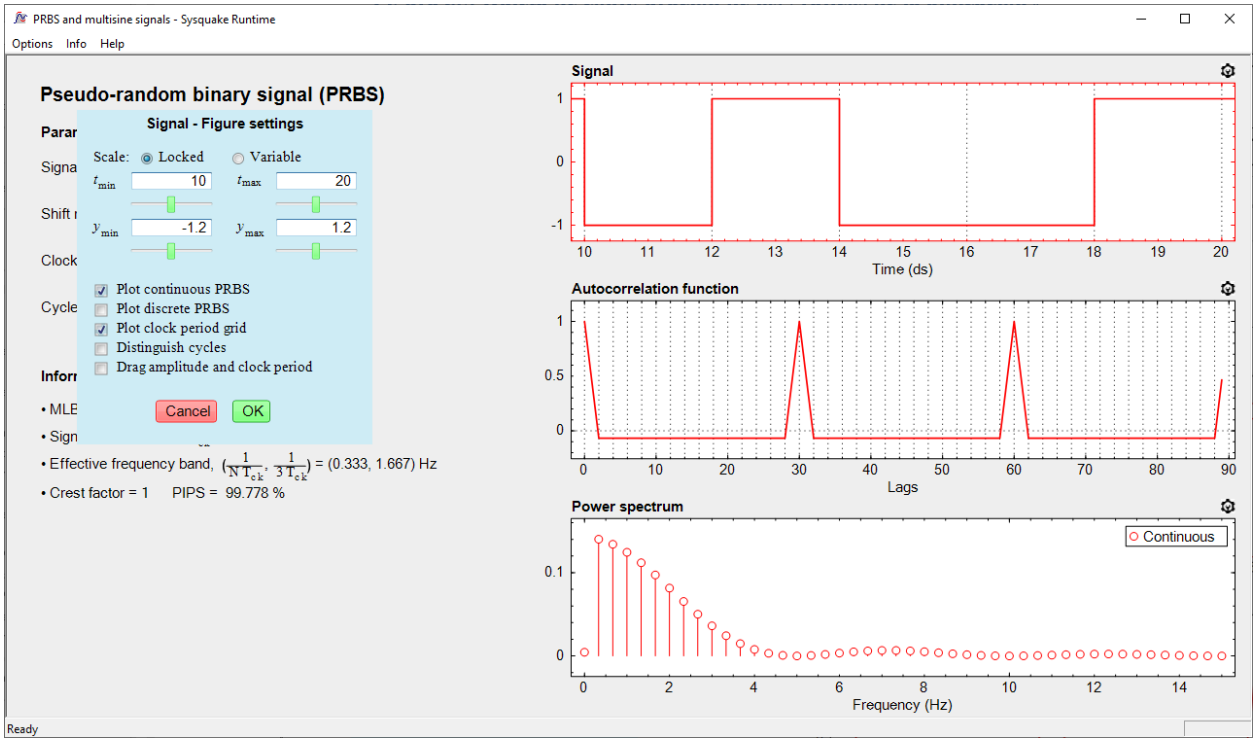

Figura 67: Escala bloqueada entre 10 y 20 segundos.

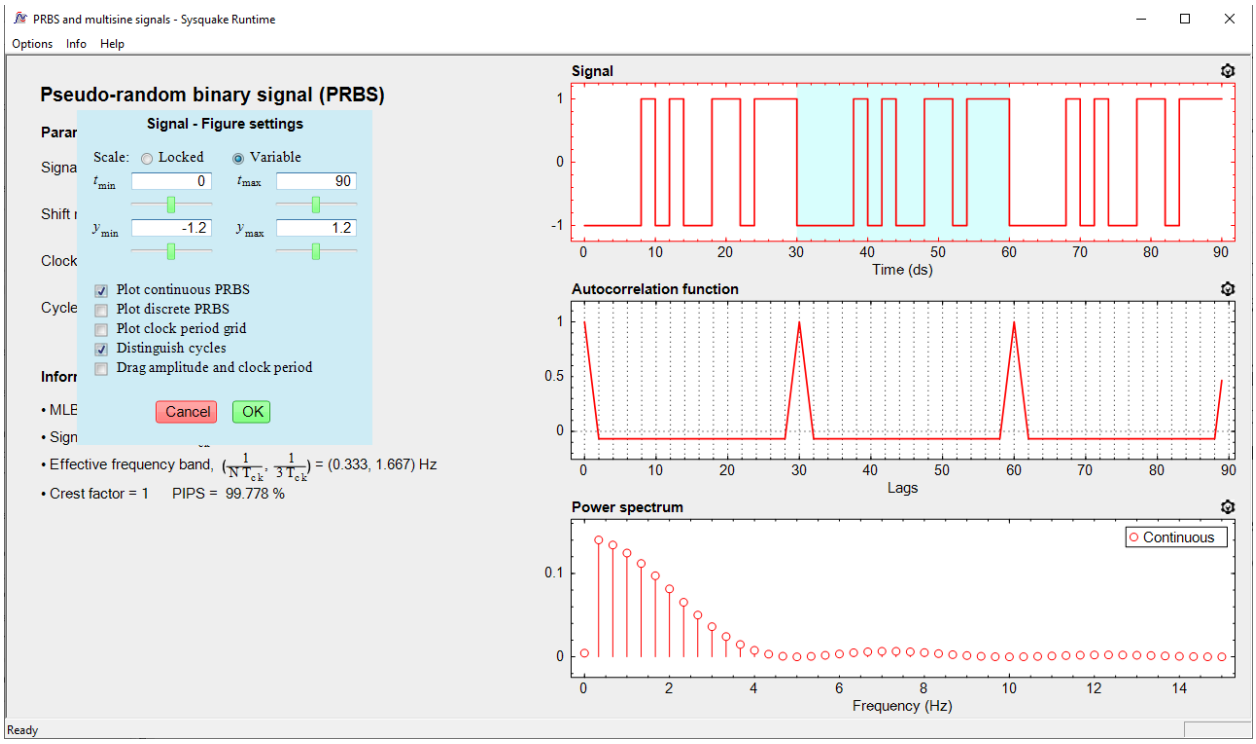

Figura 68: Opción Distinguish cycles activa.

En el caso del panel de configuración del gráfico *Autocorrelation function*, representado en la Figura 69, hay opciones que permiten configurar cómo se muestra la función de autocorrelación. Cada uno de los controles en checkbox se pueden activar de forma independiente.

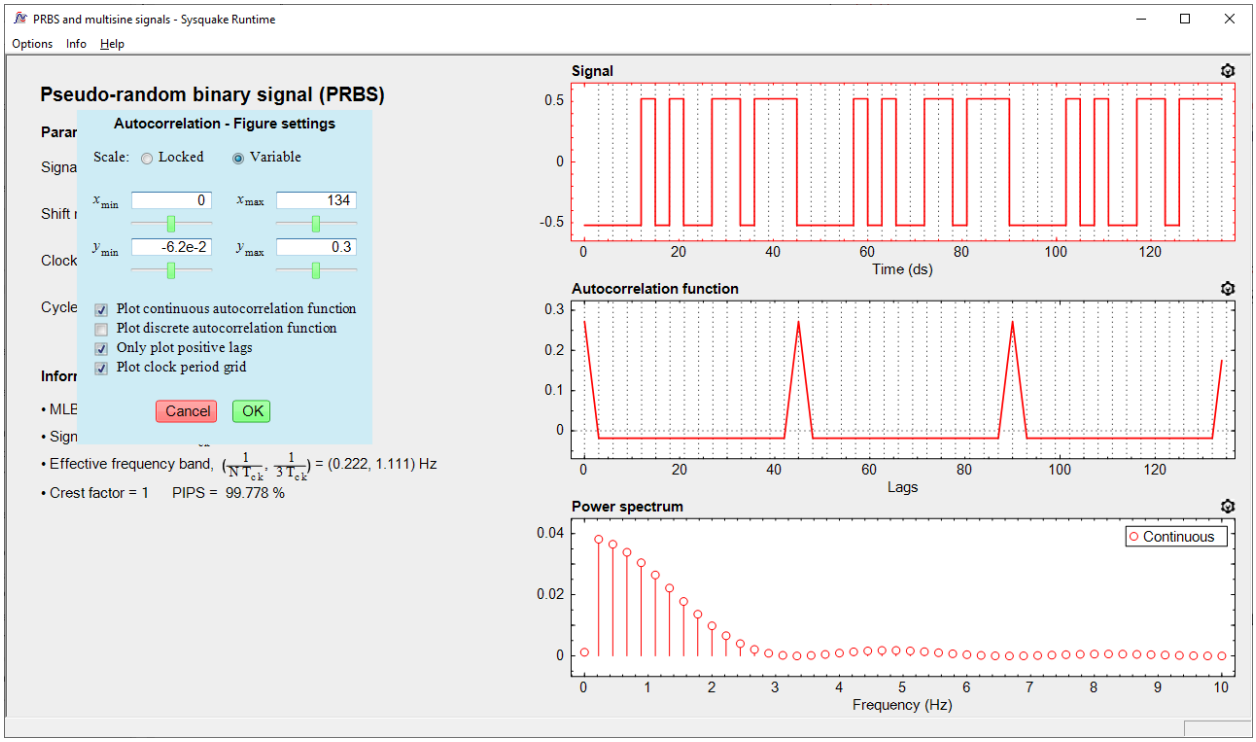

Figura 69: Panel de configuración de la función de autocorrelación.

- *Scale*: *Locked / Variable*, igual que en el caso anterior, cuando la escala está en la opción *Variable* (por defecto), la escala se ajusta automáticamente en función de los datos a representar. Cuando la escala está en la opción *Locked*, se pueden usar los controles inferiores para aumentar o reducir el rango de valores mostrado en la gráfica.
- *Plot continuous autocorrelation function*: Permite mostrar la función de autocorrelación de la señal con forma continua
- *Plot discrete autocorrelation function*: Permite mostrar la función de autocorrelación de la señal en forma discreta. Estos dos controles hacen la misma función que el menú *options* y submenú *View*, pero solo con respecto al gráfico de la función de autocorrelación.
- *Only plot positive lags*: En el dominio del tiempo la función de autocorrelación se extiende desde $-\infty$ a $+\infty$, con esta opción activada solo se representan los retardos (lags) positivos. En la Figura 70, puede verse la función de autocorrelación con esta opción desactivada.
- *Plot clock period grid*: oculta o muestra una rejilla donde aparecen los instantes del ciclo de reloj Tck. En el caso de la Figura 70 pueden verse esta utilidad desactivada.

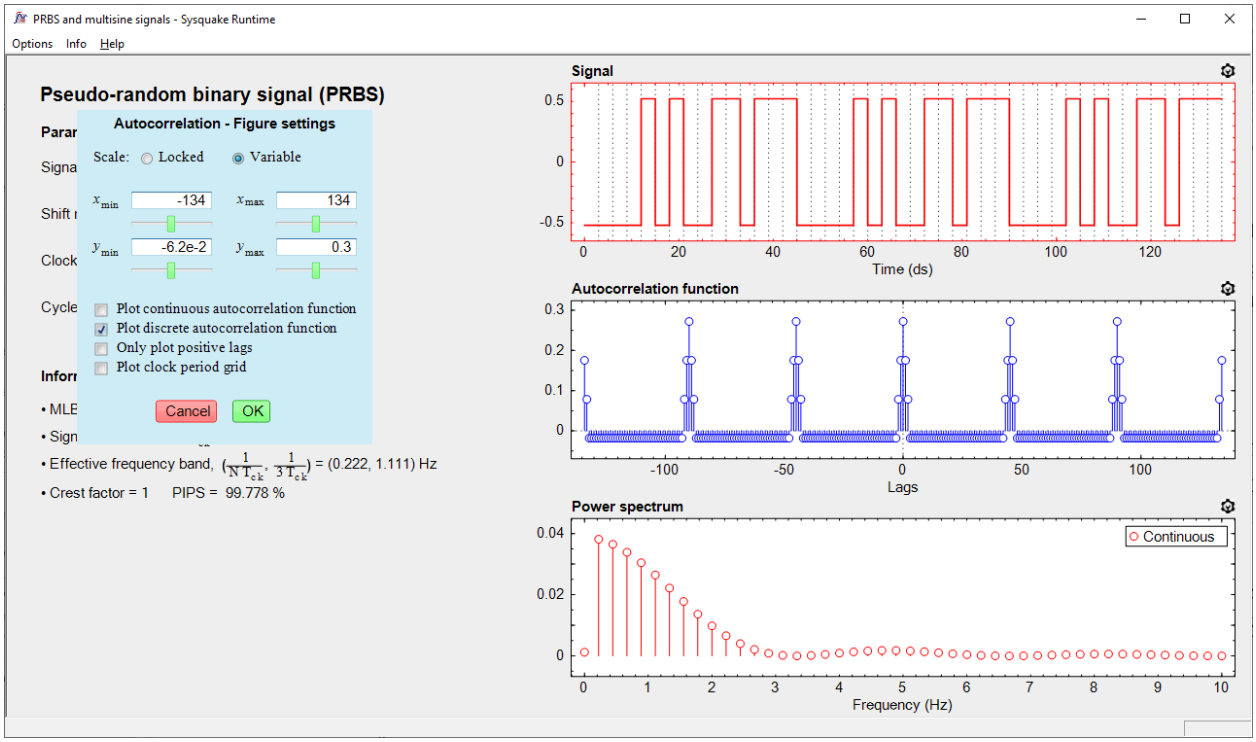

Figura 70: Función de autocorrelación incluyendo intervalos por debajo de cero, de forma discreta y sin rejilla de reloj.

En el caso del panel de configuración del gráfico *Power spectrum*, representado en la Figura 71, hay opciones que permiten configurar cómo se muestra el espectro de la señal PRBS. Al ser una señal periódica su espectro será de líneas. Cada uno de los controles en checkbox se pueden activar de forma independiente.

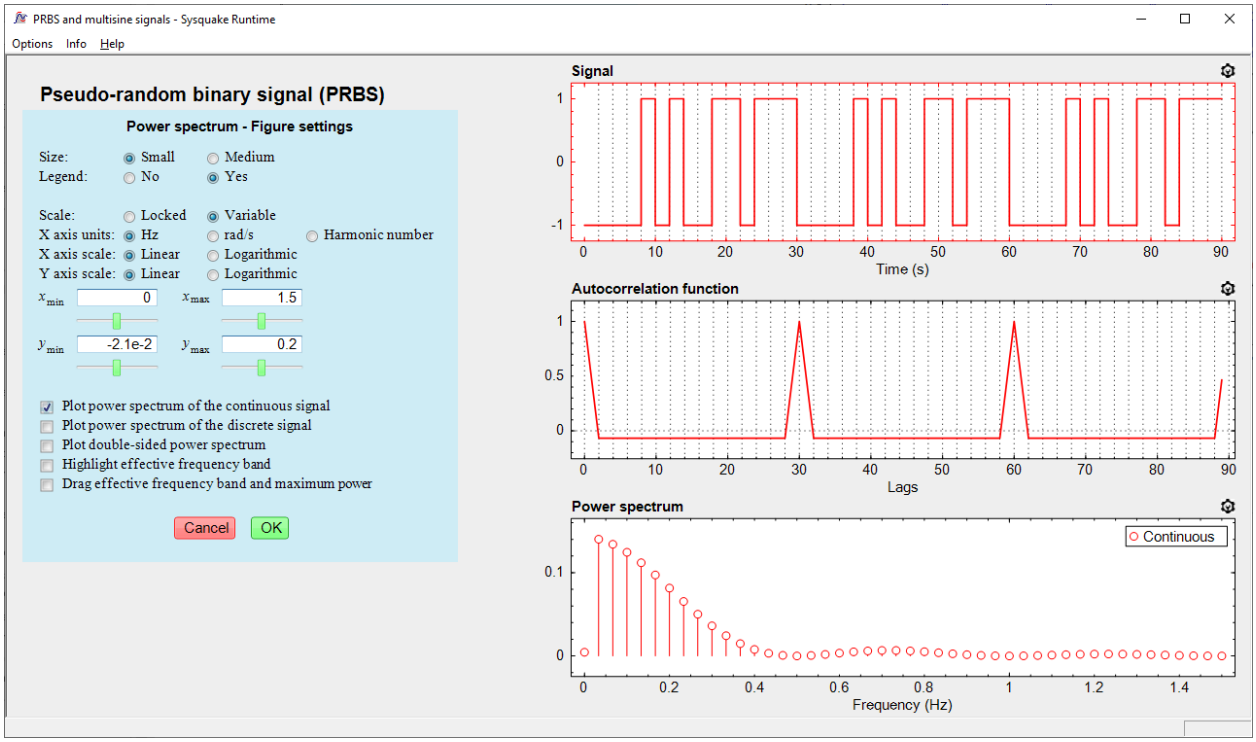

Figura 71: Panel de control de configuración del espectro de la señal.

- *Size: Small / Medium*: Con esta opción se puede escoger que el gráfico del espectro de potencia ocupe la zona que se ve en la Figura 71 o que se apropie del espacio de la función de autocorrelación y se vea como en la Figura 72.

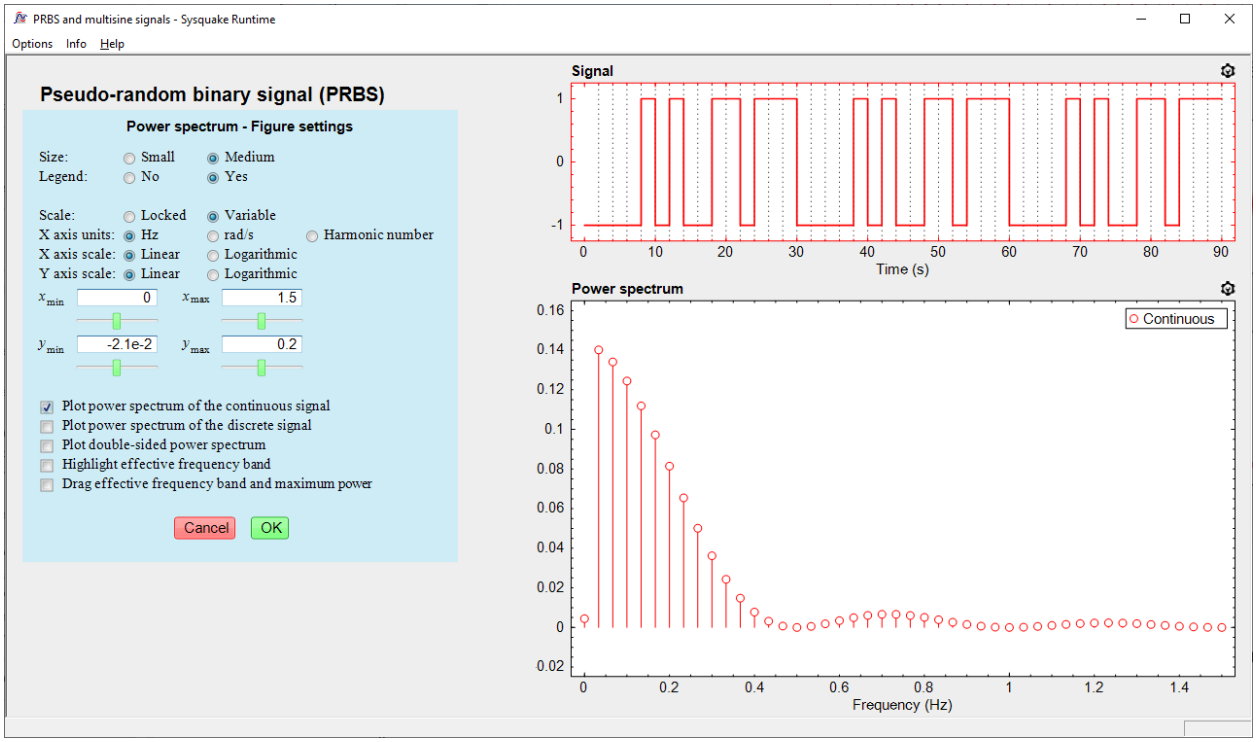

Figura 72: Espectro de la señal en tamaño mediano.

- *Legend: No / Yes*, Permite escoger si la leyenda que aparece en la parte superior derecha del gráfico es visible o no. La leyenda indica si la señal que estamos viendo es continua o discreta. En la Figura 72 puede verse la leyenda mostrada en el caso de una señal continua.
- *Scale*: *Locked / Variable*, igual que en el caso anterior, cuando la escala está en la opción *Variable* (por defecto), la escala se ajusta automáticamente en función de los datos a representar. Cuando la escala está en la opción *Locked*, se pueden usar los controles inferiores para aumentar o reducir el rango de valores mostrado en la gráfica.
- *X axis units: Hz, rad/s, Harmonic number*, este control permite escoger las unidades del eje X de la gráfica, en hercios, radianes por segundo o número de armónicos representados. Esta opción se combina con la siguiente que escoge la representación del eje como lineal o logarítmica.
- *X axis scale: lineal / logarithmic*, permite escoger si este eje tiene una representación lineal o logarítmica
- *Y axis scale: lineal / logarithmic*, permite escoger si este eje tiene una representación lineal o logarítmica. La Figura 73 representa ambos ejes de forma logarítmica para una señal PRBS continua.
- *Plot power spectrum of the contonuous signal*: Permite mostrar el espectro de la señal de forma continua

- *Plot power spectrum of the discrete signal*: Permite mostrar el espectro de la señal de forma discreta. En este caso aparece un control adicional que permite visualizar las frecuencias en que la señal produce aliasing.
- *Plot double-sided power spectrum*, el espectro de la señal se extiende desde $-\infty\ a\ +\infty$, con esta opción activada se puede ver la zona de frecuencias negativas. En la Figura 74 hay un ejemplo con esta opción activada.
- *Highlight effective frequency band*, cuando se activa este control se muestra en verde la zona correspondiente con la banda efectiva de frecuencia de la señal, o sea el punto en que la señal cae 3db con respecto a su máximo valor. La Figura 75 muestra un ejemplo de señal discreta con esta opción activa.
- *Drag effective frequency band and maximun power*, permite de forma dinámica seleccionar el ancho de banda de la señal y su energía máxima. Esta opción en esta versión de la herramienta todavía no está implementada.
- *Highlight contribution to aliasing*, con esta opción activa se marcarán en amarillo las frecuencias que contribuyen al efecto del Aliasing en la señal, o sea aquellas que están por encima de la frecuencia de Nyquist. Este fenómeno solo se produce en caso de señales discretas, por lo que si no está seleccionada la opción de visualizar el espectro discreto no se mostrará.

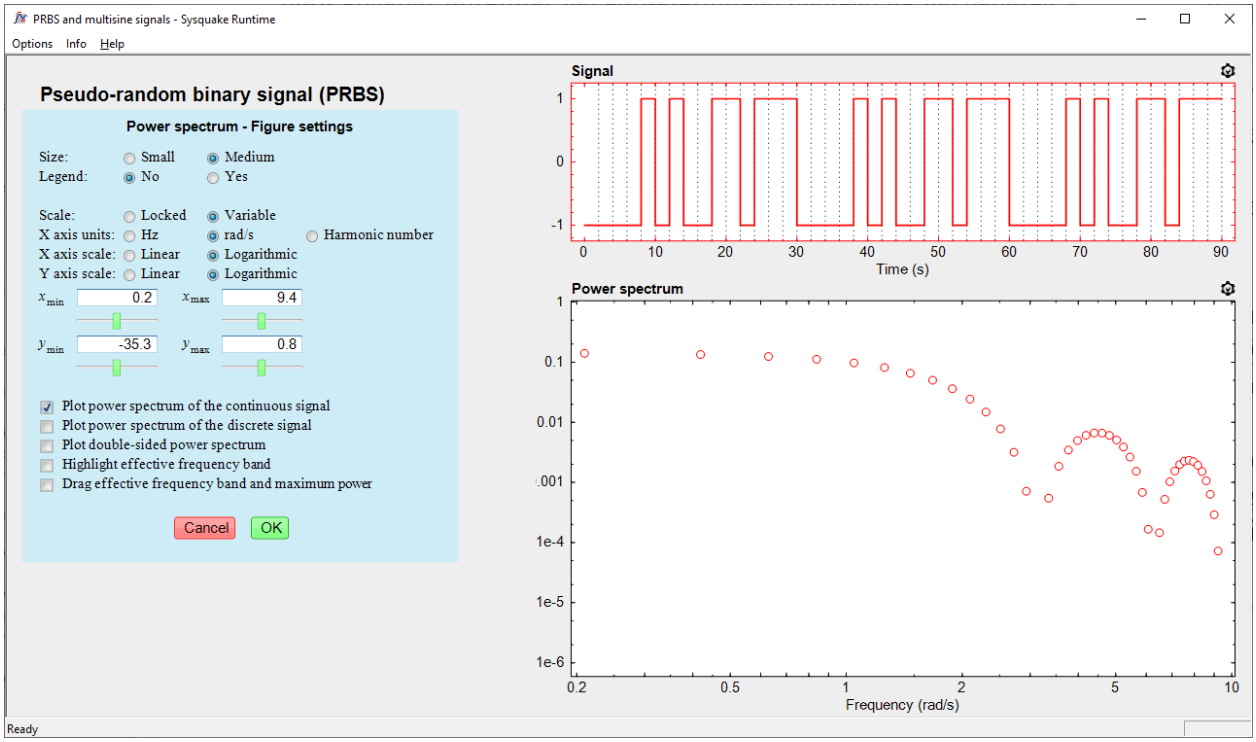

Figura 73: Espectro de la señal PRBS continuo y en escala logarítmica.

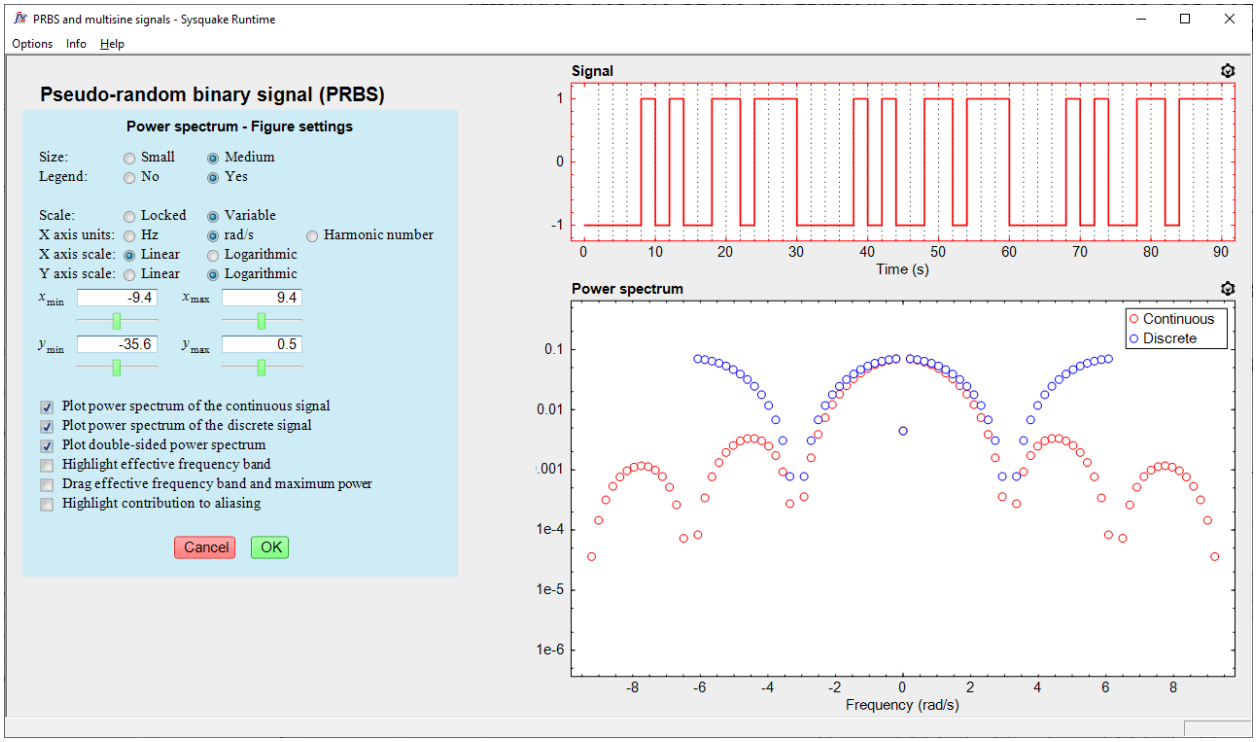

Figura 74: Espectro de la señal continua y discreta superpuestos mostrando frecuencias negativas.

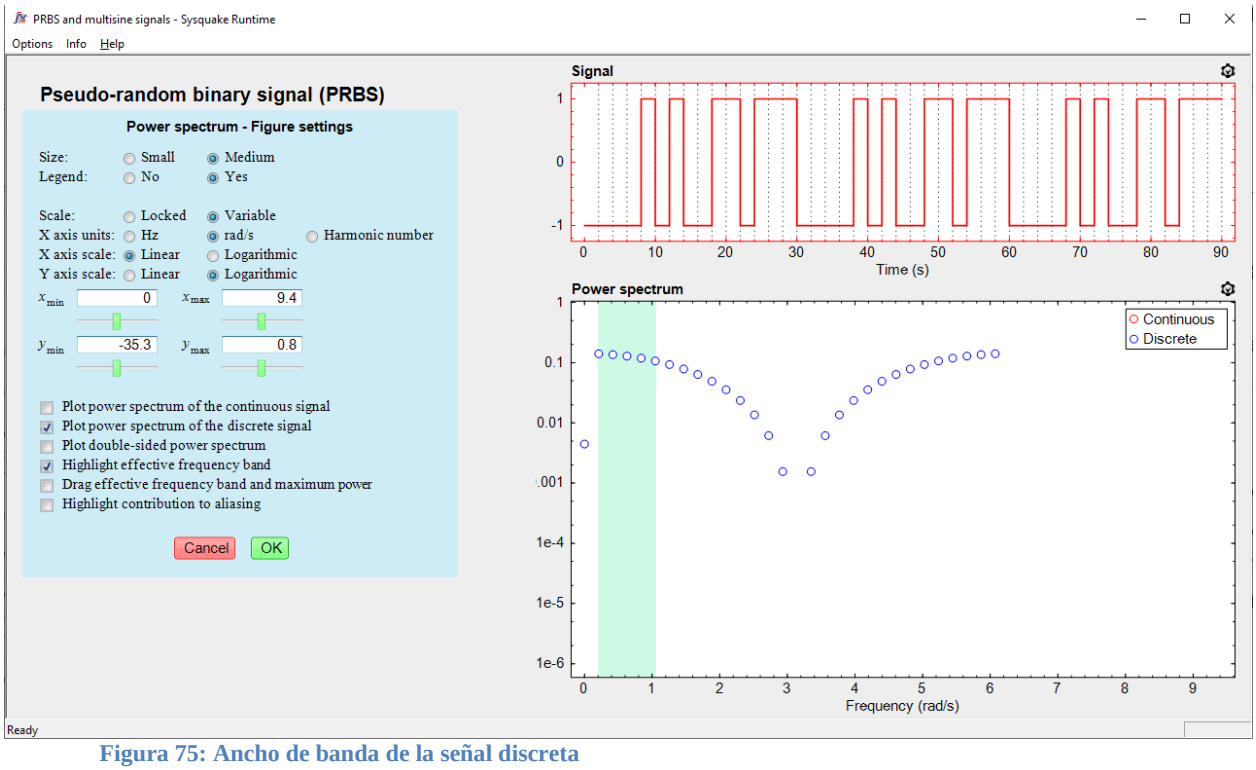

Figura 75: Ancho de banda de la señal discreta

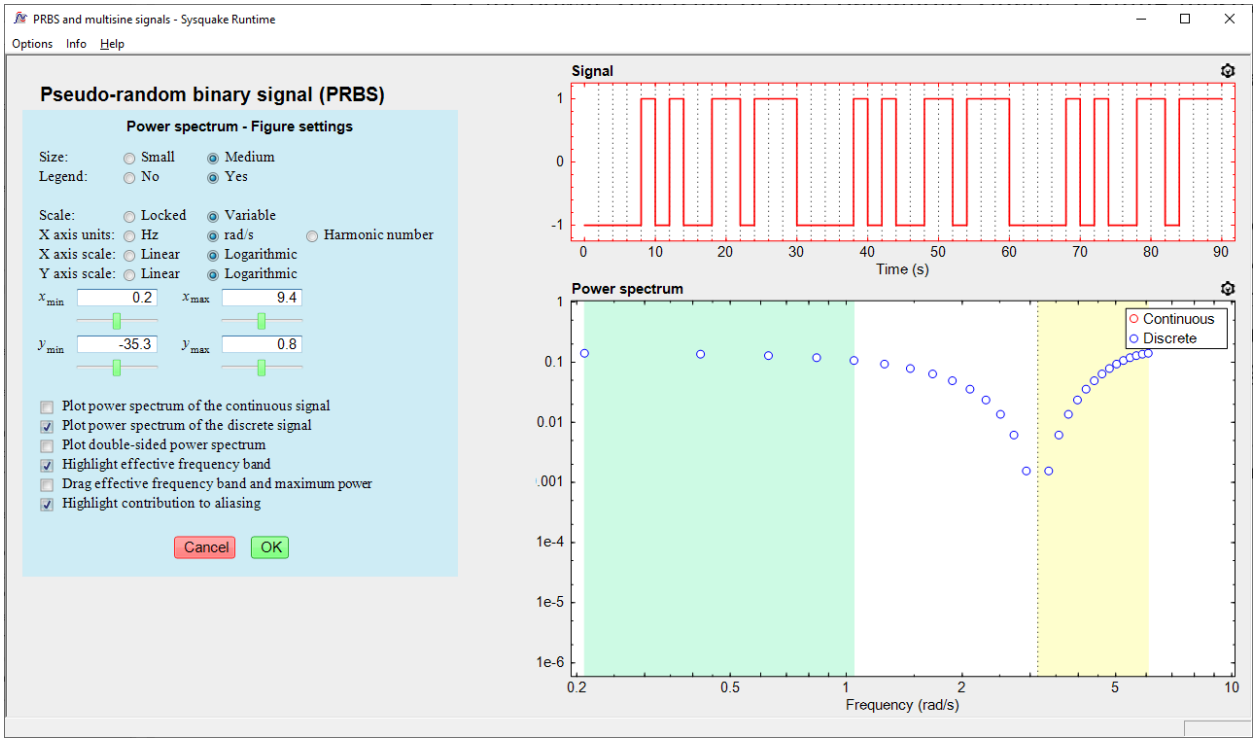

Figura 76: Efecto del Aliasing en una señal PRBS discreta.

7.3 Señal multiseno

Al igual que en el caso de la señal PRBS, el programa muestra 3 zonas diferenciadas para la señal multiseno: Parameters, information y Gráficos. El aspecto general del panel de la señal multiseno puede verse en la Figura 77.

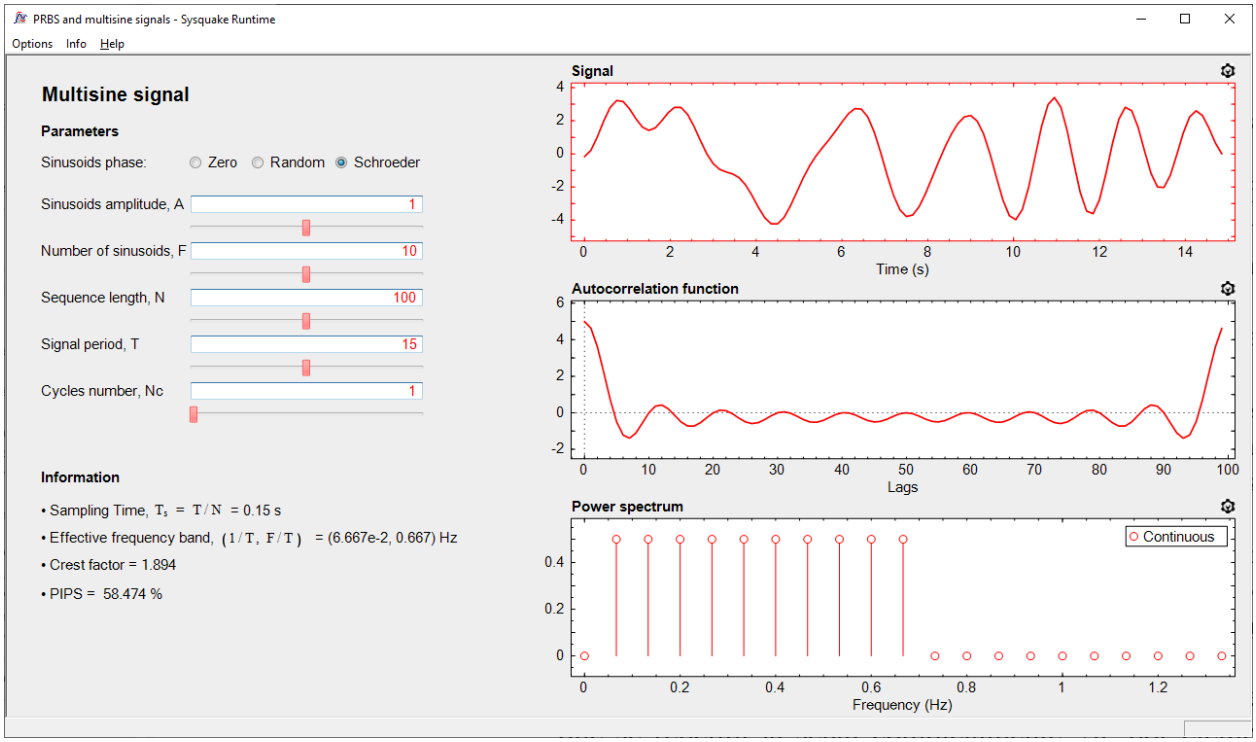

Figura 77: señal multiseno

- *Parámeters*: En esta zona se pueden modificar los parámetros fundamentales de la señal multiseno. La modificación de todos los parámetros se puede hacer de dos formas, con un control deslizante o introduciendo el valor numérico deseado en el cuadro de texto. Vea Figura 78.
    - *Sinusoids phase: Zero / Random /Schroeder*, permite escoger la fase inicial de las sinusoides entre uno de tres algoritmos. Fase cero para todas las señales, fase aleatoria diferente entre 0 y 2π para cada señal o fase siguiendo el algoritmo de Schroeder.
    - *Sinusoids amplitude, A*: Este parámetro permite seleccionar la amplitud de la señal multiseno. La señal generada es simétrica con respecto a cero, de forma que, si la amplitud seleccionada es 2, se generará una suma de señales sinusoidales de ±2 voltios. La amplitud instantánea dependerá de la fase y frecuencia de cada una de estas sinusoides.
    - *Number of sinusoids ,F*: Con este parámetro se establece el número total de sinusoides de la señal. En esta versión de la herramienta, todas las sinusoides están armónicamente relacionadas, siendo la primera de frecuencia f_0 y la última de frecuencia $F*f_0$.
    - *Sequence length, N*: Este parámetro indica el número de muestras de la señal.
    - *Signal period, T*: Este parámetro establece la duración en ciclos de la unidad de tiempo de la señal. Este parámetro junto al anterior determina el periodo de muestreo que será T/N.
    - *Cycles number, Nc*: Con este parámetro se determina el número de veces que se repetirá la señal completamente. Si, por ejemplo, a Nc le damos un valor de 3. La señal obtenida se repetirá 3 veces teniendo un periodo de 3T.

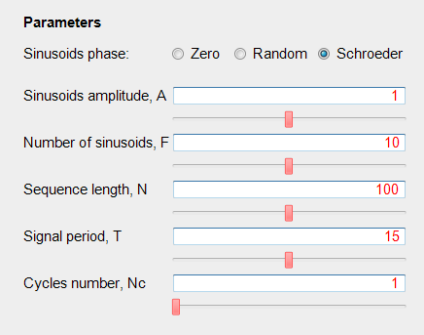

Figura 78: Parámetros básicos de la señal multiseno.

- *Information*: En esta zona de la pantalla se muestra la información básica de la señal generada. Vea Figura 79.
    - *Sampling Time, Ts*, periodo de muestreo igual a T/N unidades de tiempo.
    - *Effective frequency band*, banda de frecuencia efectiva de la señal. Esta banda va desde la frecuencia inferior que es el inverso del periodo de la señal hasta F/T que marca la frecuencia en que la señal cae 3db con respecto a su valor máximo.
    - *Crest factor:* factor de cresta de la señal que es igual a su valor máximo dividido por su valor rms. Vea fórmula [5] en sección 3.1.2.1.
    - *PIPS:* índice de perturbación de la señal, vea fórmula [8] en sección 3.1.2.2.

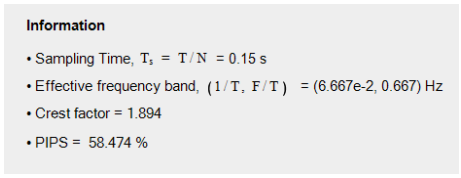

Figura 79: Información de la señal multiseno.

Al igual que en el caso de la señal PRBS hay tres gráficos que indican la forma de la señal, la función de autocorrelación y el espectro de la misma. Cada uno de estos gráficos tienen un panel de control que permite modificar algún parámetro de la visualización del mismo.

El primero de los paneles *Signal* es el que corresponde con el gráfico que muestra la señal multiseno, en él podemos encontrar las siguientes opciones:

- *Scale*: *Locked / Variable*, cuando la escala está en la opción *Variable* (por defecto), la escala se ajusta automáticamente en función de los datos a representar. Cuando la escala está en la opción *Locked*, se pueden usar los controles inferiores para aumentar o reducir el rango de valores mostrado en la gráfica.
- *Plot multisine continuous*: Permite mostrar la señal de forma continua
- *Plot multisine discrete*: Permite mostrar la señal de forma discreta, solo los instantes en que la señal es muestreada. Estos dos controles hacen la misma función que el menú *options* y submenú *View*, pero solo con respecto al gráfico de la señal.
- *Distinguish cycles*: Con esta opción activada, en el caso en que el número de ciclos sea mayor que uno, el programa pondrá con un tono azulado en el fondo los ciclos pares para distinguirlos de los impares. En la Figura 80 puede verse esta opción activada para una señal de 2 ciclos.
- *Drag amplitude*: Esta opción permite modificar en el gráfico de forma interactiva la amplitud de la señal. Tiene la misma función que el control *Sinusoids amplitude, A*. Al activar esta opción, aparecen una línea de guiones rojos que permiten arrastrar y soltar modificando el valor de amplitud, tanto para aumentar como para reducir su valor actual. En la Figura 81 puede verse esta opción activada.

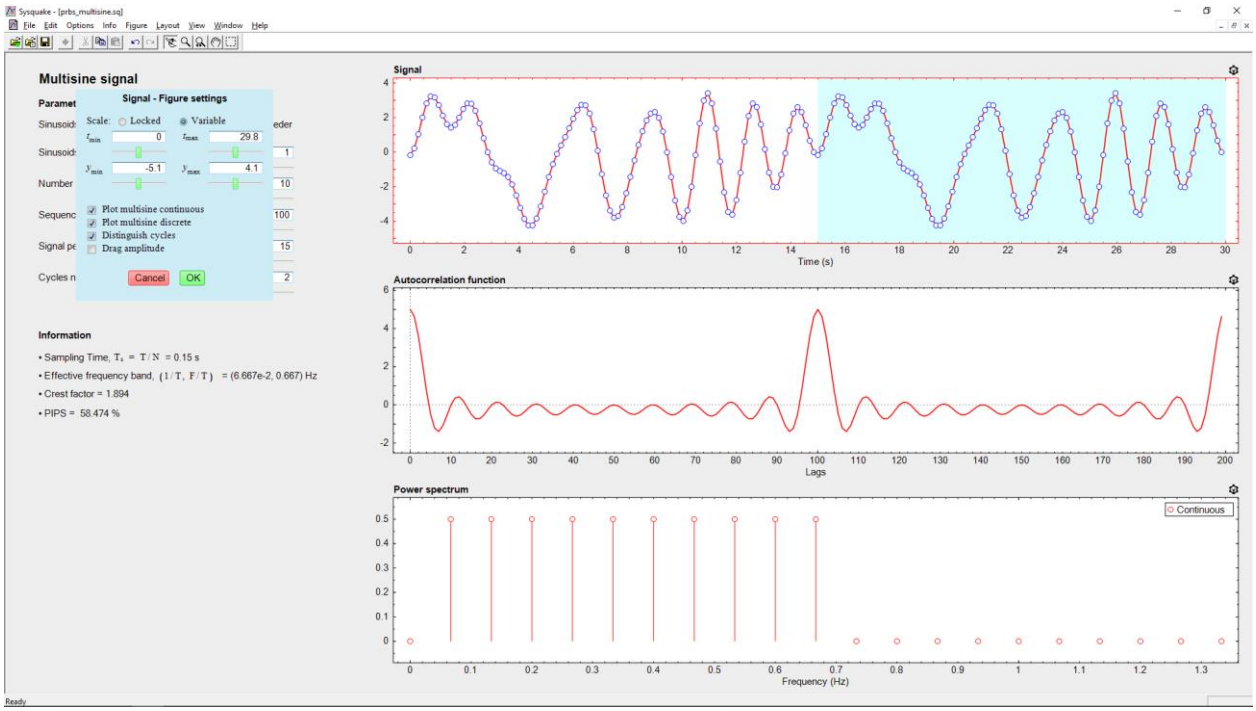

Figura 80: Panel de control del gráfico de la señal multiseno

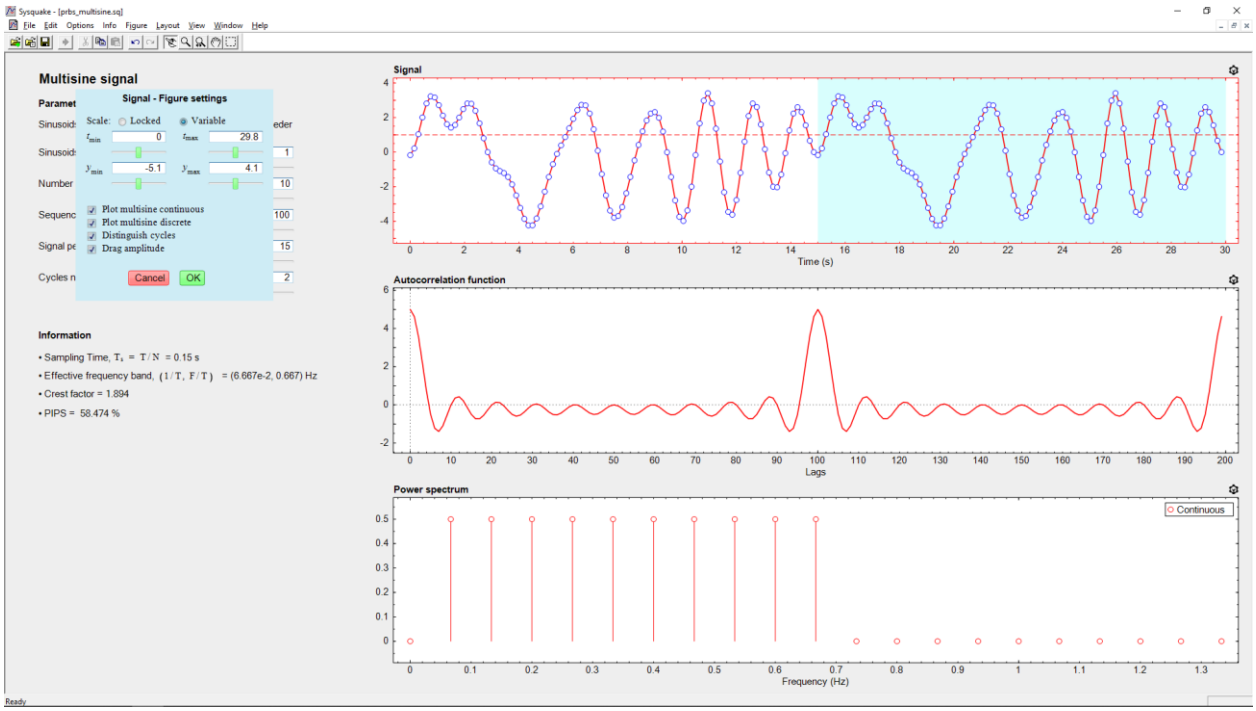

Figura 81: Modificar la amplitud de la señal multiseno arrastrando y soltando.

El segundo de los paneles gráficos *Autocorrelation function* muestra la función de autocorrelación de la señal multiseno. Las opciones asociadas a este panel pueden verse en la Figura 82. Las opciones que se pueden modificar en este caso son:

- *Scale*: *Locked / Variable*, igual que en el caso anterior, cuando la escala está en la opción *Variable* (por defecto), la escala se ajusta automáticamente en función de los datos a representar. Cuando la escala está en la opción *Locked*, se pueden usar

los controles inferiores para aumentar o reducir el rango de valores mostrado en la gráfica.

- *Plot continuous autocorrelation function*: Permite mostrar la función de autocorrelación de la señal con forma continua
- *Plot discrete autocorrelation function*: Permite mostrar la función de autocorrelación de la señal en forma discreta. Estos dos controles hacen la misma función que el menú *options* y submenú *View*, pero solo con respecto al gráfico de la función de autocorrelación.
- *Only plot positive lags*: En el dominio del tiempo la función de autocorrelación se extiende desde $-\infty\ a + \infty$, con esta opción activada solo se representan los retardos (lags) positivos.

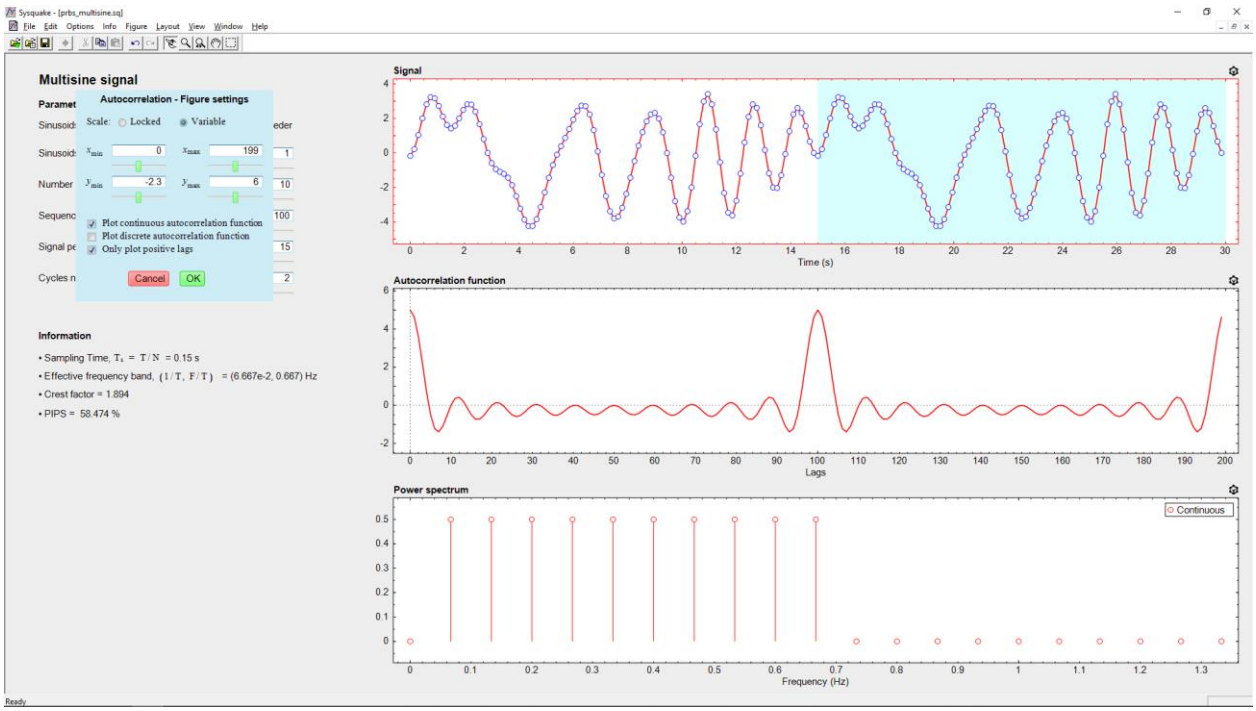

Figura 82: Panel de control de la función de autocorrelación

El último de los gráficos Power spectrum, muestra el espectro de la señal. Este gráfico tiene asociado un panel de control, (vea Figura 83), desde el cual se pueden ajustar los siguientes parámetros.

- *Size: Small / Medium*: Con esta opción se puede escoger que el gráfico del espectro ocupe la zona que se ve en la Figura 83 o que se apropie del espacio de la función de autocorrelación y se vea como en la Figura 84.
- *Legend: No / Yes*, Permite escoger si la leyenda que aparece en la parte superior derecha del gráfico es visible o no. La leyenda indica si la señal que estamos viendo es continua o discreta. En la Figura 84 puede verse la leyenda mostrada en el caso de una señal continua.
- *Scale*: *Locked / Variable*, igual que en el caso anterior, cuando la escala está en la opción *Variable* (por defecto), la escala se ajusta automáticamente en función de los datos a representar. Cuando la escala está en la opción *Locked*, se pueden usar los controles inferiores para aumentar o reducir el rango de valores mostrado en la gráfica.

- *X axis units: Hz, rad/s, Harmonic number*, este control permite escoger las unidades del eje X de la gráfica, en hercios, radianes por segundo o número de armónicos representados. Esta opción se combina con la siguiente que escoge la representación del eje como lineal o logarítmica.
- *X axis scale: lineal / logarithmic*, permite escoger si este eje tiene una representación lineal o logarítmica
- *Y axis scale: lineal / logarithmic*, permite escoger si este eje tiene una representación lineal o logarítmica.
- *Plot power spectrum of the contonuous signal*: Permite mostrar el espectro de la señal de forma continua
- *Plot power spectrum of the discrete signal*: Permite mostrar el espectro de la señal de forma discreta. En este caso aparece un control adicional que permite visualizar las frecuencias en que la señal produce Aliasing.
- *Plot double-sided power spectrum*, el espectro de la señal se extiende desde $-\infty\ a + \infty$, con esta opción activada se puede ver la zona de frecuencias negativas. En la Figura 85 hay un ejemplo con esta opción activada.
- *Highlight effective frequency band*, cuando se activa este control se muestra en verde la zona correspondiente con la banda efectiva de frecuencia de la señal, o sea el punto en que la señal cae 3db con respecto a su máximo valor. La Figura 86 muestra un ejemplo de señal discreta con esta opción activa.
- *Drag effective frequency band and maximun power*, permite de forma dinámica seleccionar el ancho de banda de la señal y su energía máxima.
- *Highlight contribution to aliasing*, con esta opción activa se marcarán en amarillo las frecuencias que contribuyen al efecto del aliasing en la señal, o sea aquellas que están por encima de la frecuencia de Nyquist. Este fenómeno solo se produce en caso de señales discretas, por lo que si no está seleccionada la opción de visualizar el espectro discreto no se mostrará. Al ser la señal multiseno continua tampoco se aprecia este efecto sobre la señal.

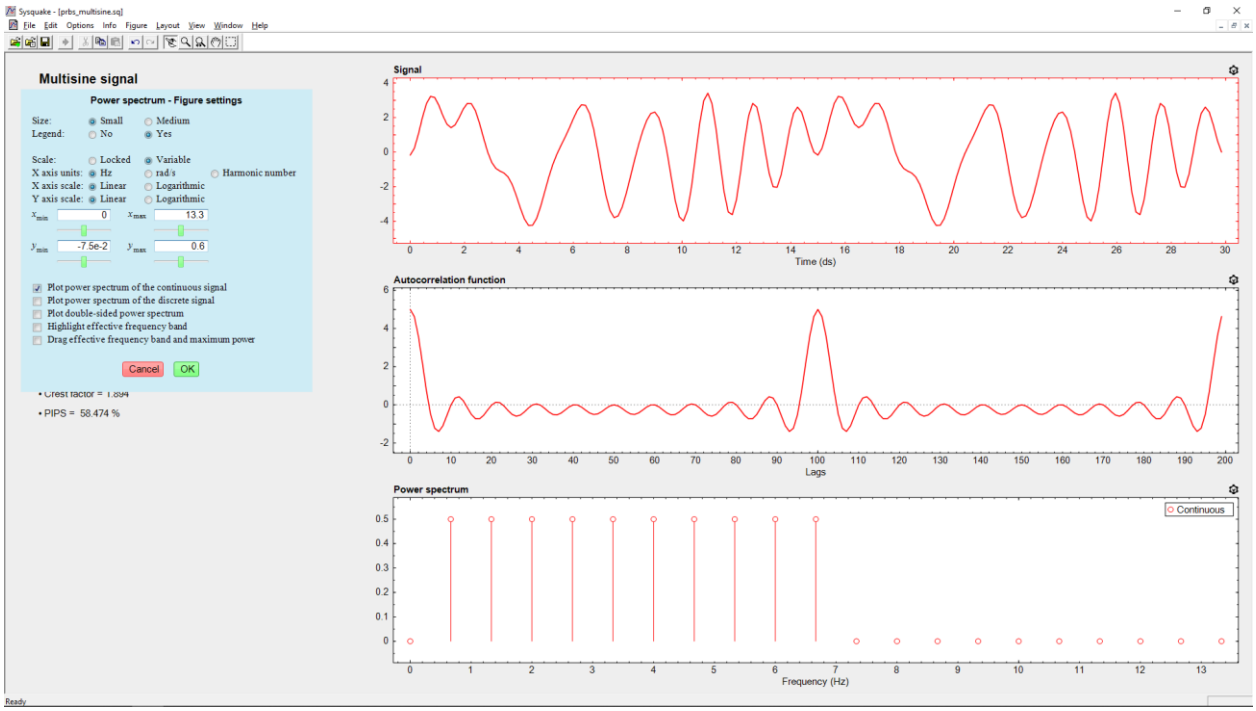

Figura 83: Panel de control del gráfico espectro de la señal multiseno

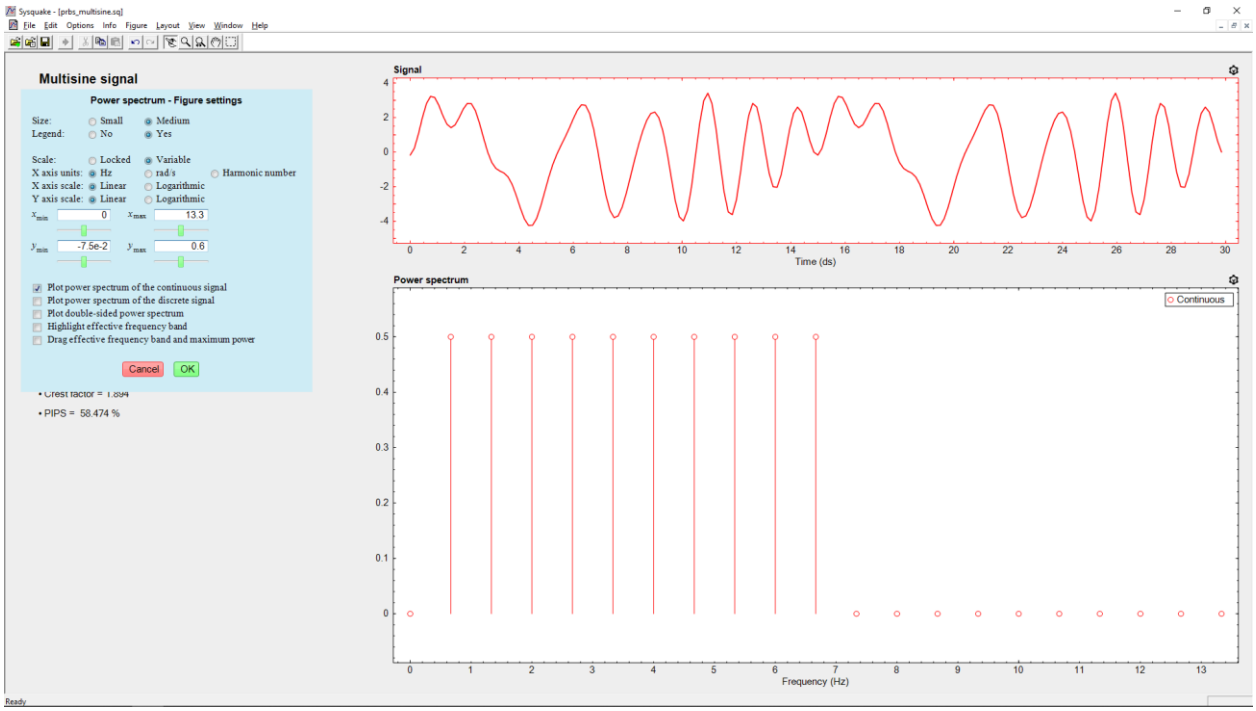

Figura 84: Grafico del espectro de la señal multiseno en tamaño mediano

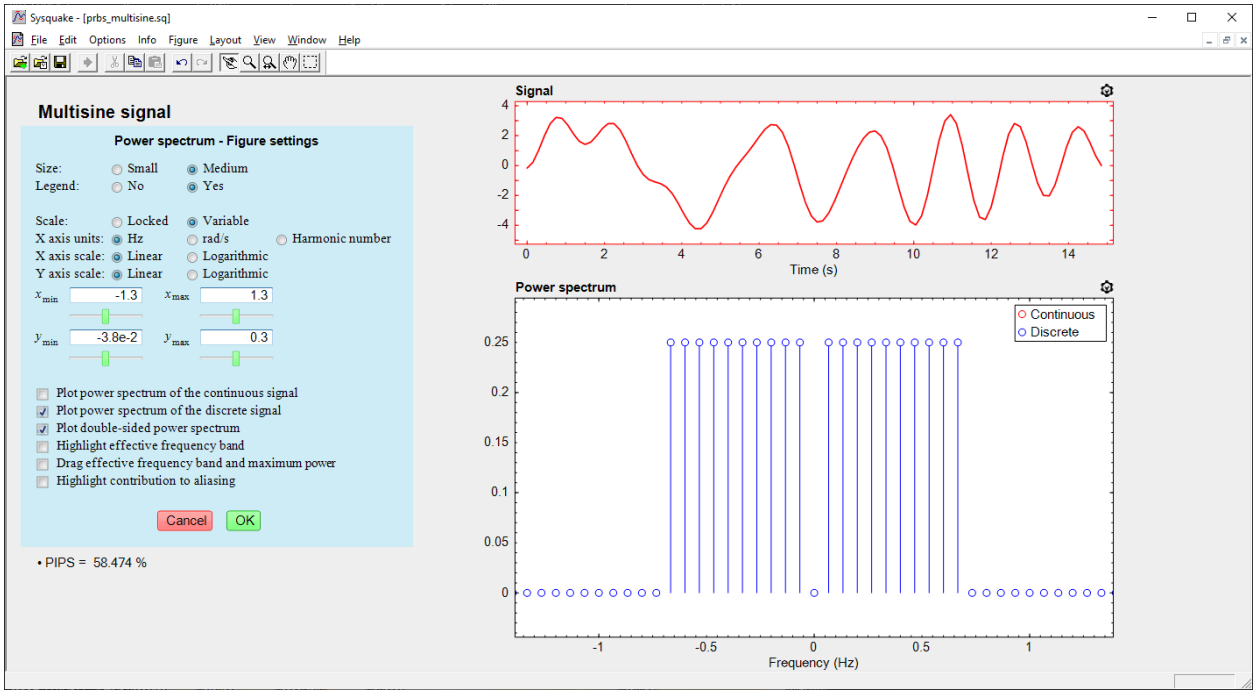

Figura 85: Espectro de la señal multiseno simetrico respecto a cero

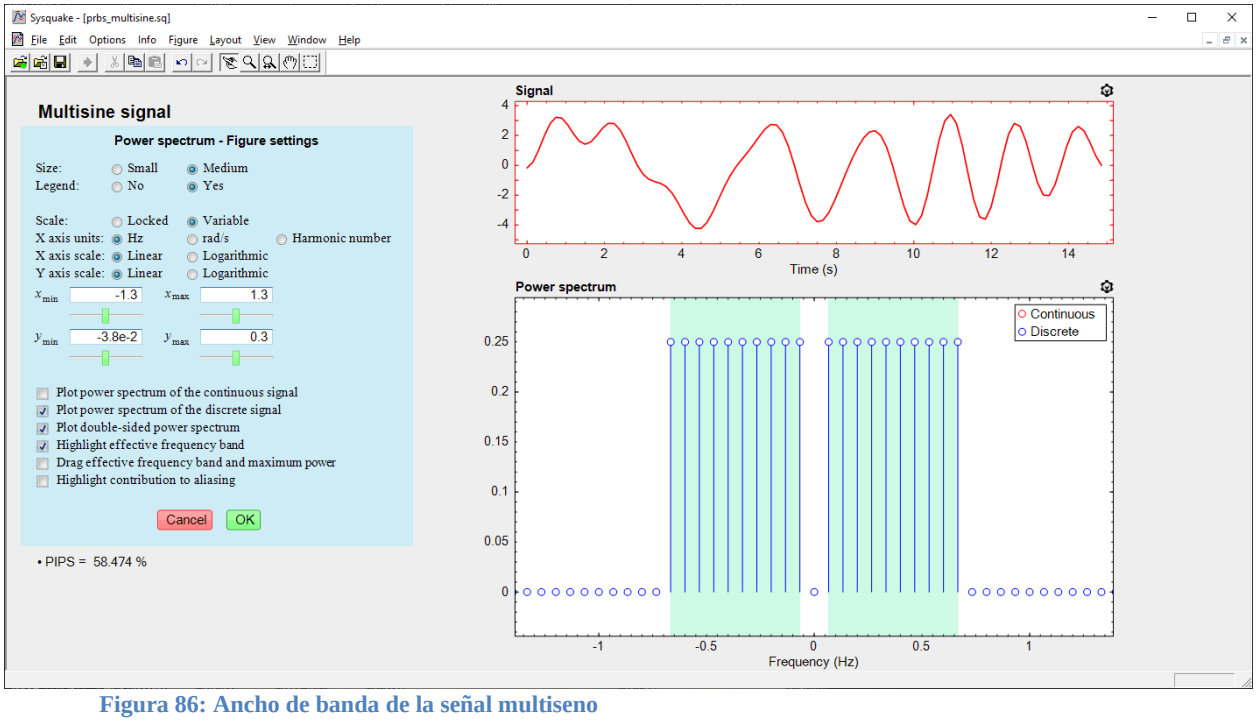

Figura 86: Ancho de banda de la señal multiseno

8 Apéndice B: Código fuente de la herramienta

```
 // charset=iso-8859-1
%%%
%%% Interactive software tool to teach/learn the main characteristics of PRBS (type
MLBS) and multisine signals
%%% by Javier Esteban and José Manuel Díaz (June 2022)

%TITLE
%====================================
title "PRBS and multisine signals"
%====================================

%VARIABLES
%====================================
variable par
                          % Structure with all the signal parameters
                          % --
                          % signal_type       1 - PRBS (MLBS); 2 - Multisine
                          % view                 1 - Continuous signal    2 - Discrete
signal
                          % time_units        1 Minutes; 0 Seconds; -1 deciseconds; -2
centiseconds; -3 miliseconds;
                          %
                          %
                          % max_harmonic                    Maximum  harmonic  that  is
plotted in the spectrum or DFT plot for continuous signal
                          % negative_harmonics              0 - Only positive harmonics
are computed; 1 - Negative and positive harmonics are computed


                          % --
                          % PRBS (MLBS) signal parameters:
                          % --
                          % prbs_a   Amplitude
                          % prbs_n   Length of the shift register       (positive
integer between 2 y 11)
                          % prbs_Tsw Clock pulse period                 (positive
integer between 1 y 50)
                          % prbs_Nc  Number of cycles of the signal (positive integer
between 1 y X)
                          % prbs_Ts                 Sampling  time  for  the  discrete
signal
                          % prbs_Ts_old             Sampling time

                          % --
                          % Multisine signal parameters:
                          % --
                          % msine_F    Number of harmonics (positive integer)

                          % msine_dphase  Method to generate the harmonic phases: 1 -
All the phases equal to 0, 2 - Random in uniform distribution in[0,2pi], 3 - Schroeder
phase design
                          % msine_a    Amplitude of each harmonic  Column vector Fx1
                          % msine_N           Sequence length  (number of samples of the
first cycle)
                          % msine_T       Period of the multisine signal.
                          % msine_Nc        Number of cycles of the signal (positive
integer between 1 y X)
                          % msine_Ts          Sampling  time  for  the  discrete  signal
(N*Ts=T)
                          % msine_Ts_old
```

```
variable signals
                        % Structure with the generated signals and its information
                        % prbs                    Column vector with the points of
the PRNS generated
                        % prbs_time      Column vector with time points of prbs
                        % prbs_mod    Row vector with the PRBS modified to be plotted
                        % prbs_mod_time  Row vector with the time points of prbs_mod
                        % prbs_dis    Row vector with the sampled PRBS
                        % prbs_dis_time  Row vector with the time points of prbs_dis
                        % prbs_N                   Number of bits of the MLBS, N=2^n-
1;                      (positive integer)
                        % prbs_T                   PRBS (MLBS) period, T= N·Tsw;
                              (positive integer)
                        % prbs_samples   PRBS (MLBS) number of samples
                        (positive integer)
                        %  prbs_Nyquist_freq    Nyquist_frequency         Row  vector
[harmonic number, frequency (Hz)]
                        % prbs_effqband     Effective  frequency  band  vector  1x4
[harmonic_number_min, harmonic_number_max, freq_min(Hz), freq_max(Hz)]

                        % prbs_PIPS      Performance index for perturbation signals
                        %  prbs_spectrum_con   Matrix with three columns: [harmonic
number, frequency (Hz), continuous power spectrum]
                        %  prbs_spectrum_dis   Matrix with three columns: [harmonic
number, frequency (Hz), discrete power spectrum]
                        % prbs_DFT      Matrix with three columns: [harmonic number,
frequency (Hz), DFT]
                        % prbs_ACF       Matrix with two columns:   [delay, ACF]

                        % msine              Column  vector  with  the  points  of  the
multisine signal
                        %  msine_time        Column  vector  with  time  points  of  the
multisine signal
                        % msine_dis    Row vector with the sampled the multisine signal
                        % msine_dis_time  Row vector with the time points of msine_dis
                        %  msine_samples    Number  of  samples  of  the  full  multisine
signal
                        %  msine_Nyquist_freq    Nyquist_frequency         Row  vector
[harmonic number, frequency (Hz)]
                        % msine_effqband    Effective  frequency  band  vector  1x4
[harmonic_number_min, harmonic_number_max, freq_min(Hz), freq_max(Hz)]

                        % msine_PIPS               Performance       index       for
perturbation signals
                        % msine_crestfactor      Crest factor
                        %  msine_spectrum_con   Matrix with three columns: [harmonic
number, frequency (Hz), continuous power spectrum]
                        %  msine_spectrum_dis   Matrix with three columns: [harmonic
number, frequency (Hz), discrete power spectrum]
                        % msine_DFT      Matrix with three columns: [harmonic number,
frequency (Hz), DFT]
                        % msine_ACF_con       Matrix with two columns:   [delay, ACF]
                        % msine_ACF_dis       Matrix with two columns:   [delay, ACF]




variable otras
                        % Structure with figure settings and general tool settings.

                        %open_aux_window  variable entera que indica si se abre la
ventana auxiliar: 0=cerrada, 1=abierta
```

```
                        %tipo_aux_window  cadena de caracteres que indica el contenido
de la ventana auxiliar:
                        %                  '1' = Signal - Figure Settings
                        %                    '2' = Autocorrelation function - Figure
Settings
                        %                  '3' = Power spectrum - Figure Settings
                        %                  '4' = DFT magnitude  - Figure Settings
                        %                  '5' = Time sampling  - Settings

                        %tipo_signal     1-PRBS, 2-Multisine

                        %fig1_tipo       vector fila 2x1 cada elemento indica el tipo
de la escala: 1 fija, 2 variable
                        %                 el elemento 1x1 para la escala_old antes de
abrir la ventana y el 1x2 para la escala despues de cerrar la ventana
                        %fig1_escala_old   vector fila de 1x4 con los valores  de
[xmin,xmax,ymin,ymax] antes de abrir la ventana o antes del arrastre
                        %fig1_escala        vector fila de 1x4 con los valores  de
[xmin,xmax,ymin,ymax] despues de cerrar la ventana
                        %fig1_prbs_mostrar vector fila  de 1x5 el significado de cada
elemento es el siguiente:
                        %                             Columna 1: Si vale 1 se dibuja
la señal PRBS continua.
                        %                             Columna 2: Si vale 1 se dibuja
la señal PRBS discreta.
                        %                             Columna 3: Si vale 1 se dibujan
lineas verticales en Tck, 2Tck, 3Tck,...
                        %                             Columna  4:  Si  vale  1  se
distinguen con zonas sombreadas los ciclos pares de la señal.
                        %                             Columna 5: Si vale 1 se dibujan
dos lineas una horizontal y otra vertical para poder configura  prbs_a y/o  prbs_Tck


                        %fig2_tipo       vector fila 2x1 cada elemento indica el tipo
de la escala: 1 fija, 2 variable
                        %                 el elemento 1x1 para la escala_old antes de
abrir la ventana y el 1x2 para la escala despues de cerrar la ventana
                        %fig2_escala_old   vector fila de 1x4 con los valores  de
[xmin,xmax,ymin,ymax] antes de abrir la ventana o antes del arrastre
                        %fig2_escala        vector fila de 1x4 con los valores  de
[xmin,xmax,ymin,ymax] despues de cerrar la ventana
                        %fig2_ACF_mostrar  vector fila  de 1x5 el significado de cada
elemento es el siguiente:
                        %                             Columna 1: Si vale 1 se dibuja
la ACF continua.
                        %                             Columna 2: Si vale 1 se dibuja
la ACF discreta.
                        %                             Columna 3: Si vale 1 se muestran
únicamente los lags positivos
                        %                             Columna 4: Si vale 1 se dibujan
lineas verticales en Tck, 2Tck, 3Tck,...

                        %fig3_tipo       vector fila 2x1 cada elemento indica el tipo
de la escala: 1 fija, 2 variable
                        %                 el elemento 1x1 para la escala_old antes de
abrir la ventana y el 1x2 para la escala despues de cerrar la ventana
                        %fig3_Xunits   escalar: 1 Cycles/s (Hz), 2 rad/s, 3 Harmonic
number
                        %fig3_Xscale   escalar: 1 Linear, 2 Logarithmic
                        %fig3_Yscale   escalar: 1 Linear, 2 Logarithmic
                        %fig3_escala_old   vector fila de 1x4 con los valores  de
[xmin,xmax,ymin,ymax] antes de abrir la ventana o antes del arrastre
                        %fig3_escala        vector fila de 1x4 con los valores  de
[xmin,xmax,ymin,ymax] despues de cerrar la ventana
```

```
                            %fig3_size        escalar para indicar el tamaño de la figura:
1 small, 2 medium
                            %fig3_legend          escalar para indicar si se muestra la
leyenda: si vale 2 se muestra la leyenda.

                            %fig3_spectrum_mostrar  vector fila  de 1x6 el significado de
cada elemento es el siguiente:
                            %                                   Columna 1: Si vale 1 se dibuja
el espectro de potencia de la señal continua.
                            %                                   Columna 2: Si vale 1 se dibuja
el espectro de potencia de la señal discreta.
                            %                                   Columna 3: Si vale 0 se muestran
el espectro de un solo lado (frecuencias positivas), si vale 2 se muestra el espectro
de doble lado.
                            %                                   Columna 4: Si vale 1 se dibuja
en verde la banda de frecuencia efectiva.
                            %                                   Columna 5: Si vale 1 se dibuja
en amarillo la zona de aliasing (empieza la frecuencia de Nyquist)

                            %                                   Columna 5: Si vale 1 se dibujan
dos lineas verticales para configurar N y Tck, y una línea horizontal para configurar A


%=====================================



%INIT
%=====================================
init (par, signals, otras) = init(_width,_height);
%=====================================

%MAKE
%=====================================

%=====================================

%RESIZE
%=====================================
resize resize(_width, _height, otras);
%=====================================

%KEYDOWN
%=====================================
keydown (otras)= keydownHandler(_key,_width,_height,otras);
%=====================================


%MENUS
%=====================================
/* beginmenu "Session"
       menu "Load session                              L"
       menu "Save session                              S"
       menu "Reset"
       separator
       menu "Generate report"
       menu "Export signal to Matlab"
endmenu
*/



beginmenu "Options"
       beginsubmenu "Signal"
```

```
            menu "PRBS"          _checkmark(par.signal_type==1)          (par,signals,
otras)=choose_signal(_height,_width,par,otras,1)
        menu "Multisine"  _checkmark(par.signal_type==2)                 (par,signals,
otras)=choose_signal(_height,_width,par,otras,2)
      endsubmenu
      separator
      beginsubmenu "View"
            menu "Continuous"          _checkmark(par.view==1)
(par,otras)=choose_view(1,par,otras)
        menu "Discrete"          _checkmark(par.view==2)
(par,otras)=choose_view(2,par,otras)
        menu "Both"                  _checkmark(par.view==3)
(par,otras)=choose_view(3,par,otras)
      endsubmenu
      separator
      beginsubmenu "Time units"
        menu "minutes (min)"     _checkmark(par.time_units==1)
(par,signals,otras)=choose_time_units(1,par,otras)
        menu        "seconds         (s)"                _checkmark(par.time_units==0)
(par,signals,otras)=choose_time_units(0,par,otras)
        menu       "deciseconds       (ds)"              _checkmark(par.time_units==-1)
(par,signals,otras)=choose_time_units(-1,par,otras)
        menu       "centiseconds      (cs)"              _checkmark(par.time_units==-2)
(par,signals,otras)=choose_time_units(-2,par,otras)
        menu       "milliseconds      (ms)"              _checkmark(par.time_units==-3)
(par,signals,otras)=choose_time_units(-3,par,otras)

      endsubmenu
      separator
      menu "Sampling time" (otras)=select_sampling_time(_height,_width,otras)
endmenu

beginmenu "Info"
    menu "About..." About()
    menu "Help..." Ayuda()
endmenu
%====================================

%FIGURES
%====================================
figure "Title"
      draw drawTitle(par)

figure "Parameters"
      draw drawParameters(par)
      mousedrag (par,signals,otras) = dragParameters(par,signals,otras,_id,_x1)

figure "Information"
      draw show_information(par,signals)

figure "Shift register"
      draw draw_register(par)

figure "Signal"
      draw drawSignal(par,signals,otras)
      mousedrag (par,signals,otras)=drag_fig1(_id,_x0,_x1,_y1,par,signals,otras)
      mouseover (_cursor,par,signals,otras)=over_fig1(_id,_x0,_y0,par,signals,otras)

figure "Autocorrelation Function"
      draw drawACF(par,signals,otras)

figure "Power Spectrum"
      draw drawSpectrum(par,signals,otras)

figure "DFT Magnitude"
```

```
       draw drawDFT(par,signals,otras)

figure "Title Figure Signal"
    draw drawTitleSIG

figure "Title Figure ACF"
       draw drawTitleACF

figure "Title Figure Power Spectrum"
       draw drawTitlePS

figure "Title Figure DFT Magnitude"
       draw drawTitleDFT

figure "Settings Figure Signal"
       draw dibujar_icono_settings
       mouseup (otras)=pulsar_icono_settings(_id,_height,_width,'1',otras)

figure "Settings Figure ACF"
       draw dibujar_icono_settings
       mouseup (otras)=pulsar_icono_settings(_id,_height,_width,'2',otras)

figure "Settings Figure Power Spectrum"
       draw dibujar_icono_settings
       mouseup (otras)=pulsar_icono_settings(_id,_height,_width,'3',otras)

figure "Settings Figure DFT Magnitude"
       draw dibujar_icono_settings
       mouseup (otras)=pulsar_icono_settings(_id,_height,_width,'4',otras)

figure "Virtual auxiliar window"
       draw draw_auxiliar_window(par,signals,otras)
       mousedrag
(otras,par,signals)=drag_auxiliar_window(_id,_ix,_nb,_x1,_y1,_height,_width,otras,par,
signals)
       mouseup
(otras,par,signals)=mouseup_auxiliar_window(_id,_ix,_nb,_x1,_y1,_height,_width,otras,p
ar,signals)

%=====================================
%FUNCTIONS CODE
%=====================================

userinterface noappmenus, notoolbar, nofigoptions, noselectall, noplotchoice

functions
{@

%-------------------------------------------------
%Libraries
%-------------------------------------------------
       use stdlib;
       use ruedas_dentadas.lml




%-------------------------------------------------
%Constants
%-------------------------------------------------
       define width0=1280;
       define height0=684;
       define size_font = ismac ? 16 : 11;
       define face_font='Arial';
```

```
       define color_fondo_vauxiliar='#CEECF5';

%------------------------------------------------
% Init function
%------------------------------------------------
function (par, signals, otras) = init(width,height)
%init

        par.signal_type=1; %PRBS
       (par, signals, otras) = inicializar(par,width,height);

%------------------------------------------------
function (par, signals, otras) = inicializar(par,width,height)

       par.view=1; %Continuous
       par.time_units=0; %Seconds

       %PRBS
       par.prbs_Ts=1;
       par.prbs_Ts_old=1;

       par.prbs_a=1;
       par.prbs_n=4;
       par.prbs_Tsw=1;
       par.prbs_Nc=1;

       %par.prbs_Ts=par.prbs_Tsw;

       par.max_harmonic=3*(2^par.prbs_n-1);
       par.negative_harmonics=0;

       %MULTISINE

       par.msine_F=10;
       par.msine_dphase=3;
       par.msine_a=1;
       par.msine_N=100;
       par.msine_T=15;
       par.msine_Nc=1;
       par.msine_Ts=par.msine_T/par.msine_N; %T=N*Ts --> Ts=T/N;
       par.msine_Ts_old=par.msine_Ts;



       otras.open_aux_window=0;
       otras.tipo_aux_window=[];

       otras.tipo_signal=par.signal_type;

       if otras.tipo_signal==2
              par.max_harmonic=2*par.msine_F;
       end

       otras.fig1_tipo=[2,2];     %Signal
       otras.fig1_escala=[0,15,-1.5,1.5];
       otras.fig1_escala_old = otras.fig1_escala;
       otras.fig1_prbs_mostrar=[1,0,1,0,0];

       otras.fig2_tipo=[2,2];     %Autocorrelation
       otras.fig2_escala=[-15,15,-0.5,1.5];
       otras.fig2_escala_old = otras.fig2_escala;
       otras.fig2_ACF_mostrar=[1,0,1,1];
```

```
       otras.fig3_tipo=[2,2];     %Power spectrum
       otras.fig3_Xunits=1; %Cycles/s (Hz)
       otras.fig3_Xscale=1; %Linear
       otras.fig3_Yscale=1; %Linear
       otras.fig3_escala=[-par.max_harmonic,par.max_harmonic,-0.1,0.5];
       otras.fig3_escala_old = otras.fig3_escala;
       otras.fig3_size=1;
       otras.fig3_legend=2;
       otras.fig3_spectrum_mostrar=[1,0,0,0,0,0];


       otras.fig4_tipo=[1,1];     %DFT magnitude
       otras.fig4_escala=[0,30,-0.1,0.5];
       otras.fig4_escala_old = otras.fig4_escala;




       otras.fig1_id=[];
       otras.fig1_x0=[];
       otras.fig1_y0=[];

       otras.fig3_type=1; % Plot the Power Spectrum in Figure 3.



       (signals,otras)=compute_signals(par,otras);

       initGraphics(width,height,otras);

%-------------------------------------------------
function (signals,otras)=compute_signals(par,otras)
%compute_signals

       if par.signal_type==1
              (signals.prbs,  signals.prbs_time)  =  gen_prbs(par.prbs_a,  par.prbs_Tsw,
par.prbs_n, par.prbs_Nc);
              (signals.prbs_mod,   signals.prbs_mod_time)   =   gen_prbs_mod(par.prbs_a,
par.prbs_Tsw, par.prbs_n, par.prbs_Nc);
              (signals.prbs_dis,   signals.prbs_dis_time)   =   gen_prbs_dis(par.prbs_a,
par.prbs_Tsw, par.prbs_n, par.prbs_Nc, par.prbs_Ts);

              signals.prbs_N=2^par.prbs_n -1;

              signals.prbs_T=signals.prbs_N * par.prbs_Tsw;
              signals.prbs_samples=signals.prbs_T;

              signals.prbs_PIPS=compute_PIPS(signals,par);
              signals.prbs_spectrum_cont=compute_spectrum_cont(signals, par);
              signals.prbs_spectrum_dis=compute_spectrum_dis(signals, par);

              signals.prbs_Nyquist_freq(1,2)=1/(2*par.prbs_Ts);

       signals.prbs_Nyquist_freq(1,1)=(signals.prbs_Nyquist_freq(1,2)*signals.prbs_T);
              Ts=convert2seconds(par.prbs_Ts,par.time_units);
              signals.prbs_Nyquist_freq(1,2)=1/(2*Ts);

              T=convert2seconds(signals.prbs_T,par.time_units);
              Tsw=convert2seconds(par.prbs_Tsw,par.time_units);
              signals.prbs_effqband(1,1)=1;
              signals.prbs_effqband(1,2)=signals.prbs_N/3;
              signals.prbs_effqband(1,3)=1/T;
              signals.prbs_effqband(1,4)=1/(3*Tsw);

              signals.prbs_DFT=compute_DFT(signals, par);
```

```
            signals.prbs_ACF=compute_ACF_dis(signals, par);

      else

            (signals.msine_dis,   signals.msine_dis_time)   =   gen_msine(par.msine_F,
par.msine_a, par.msine_dphase, par.msine_N, par.msine_T,par.msine_Nc);
            signals.msine=signals.msine_dis;
            signals.msine_time=signals.msine_dis_time;

            signals.msine_PIPS=compute_PIPS(signals,par);
            signals.msine_CF=compute_crest_factor(signals,par);

            %
            signals.msine_spectrum_dis=compute_spectrum_dis(signals, par);
            signals.msine_spectrum_cont=compute_spectrum_cont(signals, par);



            signals.msine_Nyquist_freq(1,2)=1/(2*par.msine_Ts);

      signals.msine_Nyquist_freq(1,1)=(signals.msine_Nyquist_freq(1,2)*par.msine_T);
            Ts=convert2seconds(par.msine_Ts,par.time_units);
            signals.msine_Nyquist_freq(1,2)=1/(2*Ts);

            T=convert2seconds(par.msine_T,par.time_units);
            signals.msine_effqband(1,1)=1;
            signals.msine_effqband(1,2)=par.msine_F;
            signals.msine_effqband(1,3)=1/T;
            signals.msine_effqband(1,4)=par.msine_F/T;



            signals.msine_DFT=compute_DFT(signals, par);
            signals.msine_ACF_dis=compute_ACF_dis(signals, par);
            signals.msine_ACF_con=compute_ACF_con(signals, par);




      end

      otras=calcular_escalas_variables(par,signals,otras);

%-------------------------------------------------
function (prbs, time) = gen_prbs_mod(a, Tsw, n, Np)
%gen_prbs_mod

      prbs =[]; //Initial value
      time =[];
      if n > 24.0
            fprintf('Error register too big');
            return;
      end

      permut                                                                    =
[1,2,0,0;1,3,0,0;1,4,0,0;2,5,0,0;1,6,0,0;1,7,0,0;1,2,7,8;4,9,0,0;3,10,0,0;9,11,0,0;6,8
,11,12;9,10,12,13;4,8,13,14;14,15,0,0;4,13,15,16;14,17,0,0;11,18,0,0;1,2,6,19;17,20,0,
0;19,21,0,0;21,22,0,0;18,23,0,0;17,22,23,24];
      reg = [linspace(1,1,n)]; //Initial state 11111...1
      bits = ((2^n) - 1) * Np;
      state_ant = reg(n);
      t=0;

      for j = 1:bits
```

```
            if reg(n) ~= state_ant
                  time = [time;t];
                  if state_ant == 0
                        prbs = [prbs;a];
                  else
                        prbs = [prbs;-a];
                  end
                  state_ant = reg(n);
            end
            time = [time;transpose(t:(Tsw+t-1))];
            t = t + Tsw;
            if reg(n) == 0
                  prbs = [prbs;transpose(linspace(a,a,Tsw))];
            else
                  prbs = [prbs;transpose(linspace(-a,-a,Tsw))];
            end

            %delay Tsw
            if permut(n-1,3)==0
                  c = xor( reg(permut(n-1,1)), reg(permut(n-1,2)));
            elseif permut(n-1,4)==0
                  c = xor(xor( reg(permut(n-1,1)), reg(permut(n-1,2))), reg(permut(n-
1,3)));
            else
                  c     =     xor(xor(     reg(permut(n-1,1)),     reg(permut(n-1,2))),
xor(reg(permut(n-1,3)), reg(permut(n-1,4))));
            end

            for i = n:-1:2
                  reg(i)=reg(i-1);
            end
            reg(1) = c;
      end

      if state_ant == 0
            prbs = [prbs;a];
      else
            prbs = [prbs;-a];
      end

      time = [time;t];
      %time = 0:clk:(bits*Tsw*clk)-clk;

%-------------------------------------------------
function (prbs, time) = gen_prbs(a, Tsw, n, Np)
%gen_prbs

      prbs =[]; %Initial value
      if n > 24
            fprintf('Error register too big');
            return;
      end
      permut                                                                    =
[1,2,0,0;1,3,0,0;1,4,0,0;2,5,0,0;1,6,0,0;1,7,0,0;1,2,7,8;4,9,0,0;3,10,0,0;9,11,0,0;6,8
,11,12;9,10,12,13;4,8,13,14;14,15,0,0;4,13,15,16;14,17,0,0;11,18,0,0;1,2,6,19;17,20,0,
0;19,21,0,0;21,22,0,0;18,23,0,0;17,22,23,24];
      reg = [linspace(1,1,n)]; %Initial state 11111...1
      bits = ((2^n) - 1) * Np;

      for j = 1:bits
            if reg(n) == 0
                  prbs = [prbs;transpose(linspace(a,a,Tsw))];
            else
                  prbs = [prbs;transpose(linspace(-a,-a,Tsw))];
            end
```

```
            %delay Tsw
            if permut(n-1,3)==0
                  c = xor( reg(permut(n-1,1)), reg(permut(n-1,2)));
            elseif permut(n-1,4)==0
                  c = xor(xor( reg(permut(n-1,1)), reg(permut(n-1,2))), reg(permut(n-
1,3)));
            else
                  c     =     xor(xor(     reg(permut(n-1,1)),     reg(permut(n-1,2))),
xor(reg(permut(n-1,3)), reg(permut(n-1,4))));
            end
            for i = n:-1:2
                  reg(i)=reg(i-1);
            end
            reg(1) = c;
      end

      time = 0:bits*Tsw-1;
      time=time';
      %time = 0:clk:(bits*Tsw*clk)-clk;


%--------------------------------------------------
function (prbs_dis, time_dis) = gen_prbs_dis(a, Tsw, n, Nc, Ts);
%gen_prbs_dis

            %PRBS discretizada con tiempo de muestreo Ts
            Tsw_virtual=(1/Ts)*Tsw;

            (prbs_dis, time_dis) = gen_prbs(a, Tsw_virtual, n, Nc);
            time_dis=time_dis*Ts;


            %[time_dis,prbs_dis]
            %L=length(time_dis)

%--------------------------------------------------
function (msine, time) = gen_msine_TAN(F, a, dphase, N, T, Nc);
%gen_msine_TAN

      f=[1:F]; % Harmonics

      %Amplitude
      A=a*ones(1,F);

      switch dphase
      case 1 %zeros phases
            phi=zeros(1,F);

      case 2 %random phases
            phi=2*pi*rand(1,F);

      case 3 %Schroeder phases
            pk=abs(A.^2)/sum(abs(A.^2));
            phi(1)=0;
            for k=2:F
                  phi(k)=-2*pi*[k-1:-1:1]*[pk(1:k-1)]';
            end
      end

      %obtain the signal u through the frequency domain
      U=zeros(N,1);
      U(f+1)=A.*exp(j.*phi);
      U(N-f+1)=conj(U(f+1));
```

```
        msine=real(ifft(U));

        %scale to obtain RMS of 1
        %urms=sqrt(mean(msine.^2));
        %°msine=msine/urms;

        if Nc>1
                x=msine;
                for h=2:Nc
                        x=[x;msine];
                end
                msine=x;
        end




        for h=0:(N*Nc-1)
                time(h+1,1)=h*T/N;
        end



%%--------------------------------------------------
function (msine, time) = gen_msine(F, a, dphase, N, T, Nc);
%gen_msine

        f=[1:F]; % Harmonics

        %Amplitude
        A=a*ones(1,F);

        %Sampling time:
        Ts=T/N;

        omega=1:F;
        omega=omega'*2*pi/N/Ts;



        %phi=cumsum((1:ns)'.*a); % create the phases

        switch dphase
        case 1 %zeros phases
                phi=zeros(1,F);

        case 2 %random phases
                phi=2*pi*rand(1,F);

        case 3 %Schroeder phases
                phi(1)=0;
                pk=abs(A.^2)/sum(abs(A.^2));
                for k=2:F
                        phi(k)=-2*pi*[k-1:-1:1]*[pk(1:k-1)]';
                end
        end


        A=A(:); % be sure column vector
        phi=phi(:);


        for h=1:N
                u(h,1)=sum(A.*cos(omega.*h*Ts+phi));
        end
```

```
       msine=repmat(u,Nc,1);  % then repeat it, nc times

       t=0:Ts:N*Ts*Nc-Ts;  % create the corresponding time vector
       time=t'; % transpose the vector, to be tidy



       %for h=0:(N*Nc-1)
       %      time(h+1,1)=h*T/N;
       %end




%--------------------------------------------------
function y=convert2seconds(x,time_units);
%convert2seconds

              if time_units<0
                     y=x*10^time_units; %seconds
              elseif       time_units==1
                     y=x*60; %seconds
              else
                     y=x; %seconds
              end

%-----------------------------
function PIPS=compute_PIPS(signals,par)
% compute_PIPS
       if par.signal_type==1
              u=signals.prbs;
       else
              u=signals.msine;
       end

    % Compute the PIPS of a signal according to the function calculate_PIPS.m  from PRS
software (https://sites.google.com/view/signaldesign/prs)
       DFT = abs(fft(u));
       for k = 2:length(u)
              DFT_squared(k) = DFT(k)*DFT(k);
       end
       PIPS = 200*sqrt(sum(DFT_squared))/length(u)/(max(u)-min(u));

%-----------------------------
function CF=compute_crest_factor(signals,par)
%compute_crest_factor
       if par.signal_type==1
              u=signals.prbs;
       else
              u=signals.msine;
       end

       %Crest Factor
       upeak=max(abs(u));
       urms=sqrt(mean(u.^2));

       CF=upeak/urms;


%-----------------------------
function spectrum=compute_spectrum_cont(signals, par);
%compute_spectrum_cont
```

```
%This function compute the one-sided power spectrum of a continuous PRBS signal or
multine signal

       if par.signal_type==1 %PRBS

              freq = [];
              power = [];

              %Expression of the power spectrum of a PRBS is given in pp.68-69 of
[Godfrey, 1993]
              % Harmonic k=0
              power0=par.prbs_a^2/signals.prbs_N^2;

              % Harmonics k>0
              for k=1:par.max_harmonic
                     T=convert2seconds(signals.prbs_T,par.time_units);
                     freq(k,1)=k/T;
                     R=(par.prbs_a^2*(signals.prbs_N+1)/signals.prbs_N^2);
                     C=(sin(k*pi/signals.prbs_N)/(k*pi/signals.prbs_N))^2;

                     power(k,1)=R*C;
              end

              if par.negative_harmonics==0 %one-side spectrum
                     %Anadir el harmónico k=0 y
                     freq=[0;freq];
                     power=[power0;2*power];
                     harmonics=(0:par.max_harmonic)';
              else  %Two-side spectrum
                     freq=[-flipud(freq);0;freq];
                     power=[flipud(power);par.prbs_a^2/signals.prbs_N^2;power];
                     harmonics=(-par.max_harmonic:par.max_harmonic)';

              end


              spectrum=[harmonics,freq,power];


       else %multisine

              freq = [];
              power = [];

              %Expression of the power spectrum of a sinusoids is given in pp.66 of
[Godfrey, 1993]
              % Harmonic k=0
              power0=0;

              % Harmonics k>0
              for k=1:par.max_harmonic
                     T=convert2seconds(par.msine_T,par.time_units);
                     freq(k,1)=k/T;
                     if k>par.msine_F
                            power(k,1)=0;
                     else
                            power(k,1)=0.25*par.msine_a^2;
                     end
              end

              if par.negative_harmonics==0 %one-side spectrum
                     %Anadir el harmónico k=0 y
                     freq=[0;freq];
                     power=[power0;2*power];
                     harmonics=(0:par.max_harmonic)';
```

```
            else  %Two-side spectrum
                  freq=[-flipud(freq);0;freq];
                  power=[flipud(power);0;power];
                  harmonics=(-par.max_harmonic:par.max_harmonic)';

            end


            spectrum=[harmonics,freq,power];


      end

%-----------------------------
function spectrum=compute_spectrum_dis(signals, par);
%compute_spectrum_dis
%This function compute the one-sided power spectrum of a discrete signal u

      if par.signal_type==1
            u=signals.prbs_dis;
            Nc=par.prbs_Nc;
            %T=N*Ts  -->N=T/Ts
            N=(signals.prbs_T)/par.prbs_Ts;
            T=convert2seconds(signals.prbs_T,par.time_units);

      else

            u=signals.msine_dis;
            Nc=par.msine_Nc;
            %T=N*Ts  -->N=T/Ts
            N=par.msine_N;
            T=convert2seconds(par.msine_T,par.time_units);


      end


      % Compute dicrete power spectrum from a sequence sampled in the first period of
the continuous signal.
      Lt=length(u);
      DFT = fft(u(1:Lt/Nc));
      L=length(DFT);

      %[DFT/L,conj(DFT)/L,(DFT.*conj(DFT))/L^2]
      power_dis=(DFT.*conj(DFT))/L^2;



      %Single-sided power spectrum
      if par.negative_harmonics==0 %One-sided power spectrum
            power_dis(2:end)=2*power_dis(2:end);

            for k=1:N-1
                  harmonics(k,1)=k;
                  freq(k,1)=k/T;
            end

            %Anadir el harmónico k=0;
            freq=[0;freq];
            harmonics=[0;harmonics];

      else %Double-sided power spectrum
            power_dis(2:end)=power_dis(2:end);
```

```
            for k=1:N-1
                  harmonics(k,1)=k;
                  freq(k,1)=k/T;
            end
            harmonics=[-flipud(harmonics);0;harmonics];
            freq=[-flipud(freq);0;freq];
            power_dis=[flipud(power_dis(2:end));power_dis(1);power_dis(2:end)];

      end

      spectrum=[harmonics,freq,power_dis];




%------------------------------
function DFT=compute_DFT(signals, par);
%compute_DFT

      if par.signal_type==1
            Nc=par.prbs_Nc;
            T=signals.prbs_T;
            u=signals.prbs;
            fin=Nc*T-1;


      else %Multisine
            N=par.msine_N;
            T=par.msine_T;
            Nc=par.msine_Nc;
            u=signals.msine;
            fin=Nc*N-1;

      end

      for k=1:fin
            DFT(k+1,1)=k; %Harmonic k;

            T=convert2seconds(T,par.time_units);
            DFT(k+1,2)=k/T; %Frequency harmonic k;
      end


      DFT(:,3)=abs(fft(u)); %DFT




%------------------------------
function ACF=compute_ACF_dis(signals, par);
%compute_ACF_dis



      if par.signal_type==1 %PRBS
            u=signals.prbs;
            fin=signals.prbs_T*par.prbs_Nc-1;

      else %Multisine
            u=signals.msine;
```

```
            fin=par.msine_N*par.prbs_Nc-1;
      end


      % Compute the autocorrelation function of a signal according to the function
autocorrelation.m  from PRS software (https://sites.google.com/view/signaldesign/prs)
      vector = (ifft(abs(fft(u)).^2))/length(u);
      ACF(:,2) = real(vector);

      for k=0:fin
            ACF(k+1,1)=k;%delay (lag)
      end


      %Insert negative lags
      temp(:,1)=[-flipud(ACF(2:end,1));ACF(:,1)];
      temp(:,2)=[flipud(ACF(2:end,2));ACF(:,2)];
      ACF=temp;


%------------------------------
function ACF=compute_ACF_con(signals, par);
%compute_ACF_con



      if par.signal_type==1 %PRBS

            %No es necesario calcularla ya ue coincide con la discreta.

      else %Multisine  --> Recordar que la ACF de una señal x(t)=V cos(wt+phi) es
(1/2)*V^2*cos(w*lag)
            a=par.msine_a;
            F=par.msine_F;
            N=par.msine_N;
            T=par.msine_T;
            Nc=par.msine_Nc;
            Ts=T/N;

            fin=T*par.prbs_Nc-1;

            %Amplitude
            A=(a.^2)/2*ones(1,F);

            %Frecuencias de los F armónicos
            omega=1:F;
            omega=omega'*2*pi/N/Ts;

            %ACF

            A=A(:); % be sure column vector
            omega=omega(:); % be sure column vector

            for k=0:N*Nc-1
                  ACF(k+1,1)=k;
                  ACF(k+1,2)=sum(A.*cos(omega.*k*Ts));
            end

            %Insert negative lags
            temp(:,1)=[-flipud(ACF(2:end,1));ACF(:,1)];
            temp(:,2)=[flipud(ACF(2:end,2));ACF(:,2)];
            ACF=temp;

      end
```

```
%-----------------------------
function otras=calcular_escalas_variables(par,signals,otras)
%calcular_escalas_variables

if par.signal_type==1 %PRBS
	if otras.fig1_tipo(2)==2 %Variable scale
		xmin=0;
		xmax=signals.prbs_mod_time(end);
		ymin=min(signals.prbs_mod);
		ymax=max(signals.prbs_mod);
		escala=[xmin,xmax,ymin-0.2*abs(ymin),ymax+0.2*abs(ymax)];
		otras.fig1_escala=escala;
		otras.fig1_escala_old=escala;
	end


	if otras.fig2_tipo(2)==2 %Variable scale
		xmin=min(signals.prbs_ACF(:,1));
		xmax=max(signals.prbs_ACF(:,1));
		ymin=min(signals.prbs_ACF(:,2));
		ymax=max(signals.prbs_ACF(:,2));
		rango=ymax-ymin;
		ymin=ymin-0.15*rango;
		ymax=ymax+0.15*rango;
		escala=[xmin,xmax,ymin,ymax];

		otras.fig2_escala=escala;
		otras.fig2_escala_old=escala;

		if otras.fig2_ACF_mostrar(3)==1  %Only plot positive lags
			otras.fig2_escala(1)=0;
			otras.fig2_escala_old(1)=0;
		end

	end



	if otras.fig3_tipo(2)==2 %Variable scale

		if otras.fig3_Xunits==1
			Xc=signals.prbs_spectrum_cont(:,2);
			Xd=signals.prbs_spectrum_dis(:,2);
		elseif otras.fig3_Xunits==2
			Xc=2*pi*signals.prbs_spectrum_cont(:,2);
			Xd=2*pi*signals.prbs_spectrum_dis(:,2);

		else
			Xc=signals.prbs_spectrum_cont(:,1);
			Xd=signals.prbs_spectrum_dis(:,1);
		end


		if par.max_harmonic==0
			xmin=0;
		else
			xmin=min(Xc);
		end
		if otras.fig3_Xscale==2
			xmin=Xc(2);
		end
		xmax=max(Xc);
		ymin=min(signals.prbs_spectrum_cont(:,3));
		ymax=max(signals.prbs_spectrum_cont(:,3));
```

```
            if otras.fig3_Yscale==1
                  rango=ymax-ymin;
                  ymin=ymin-0.15*rango;
                  ymax=ymax+0.15*rango;
            else
                  rango=log10(ymax)-log10(ymin);
                  ymin=log10(ymin)-0.05*rango;
                  ymax=log10(ymax)+0.05*rango;
            end

            escala=[xmin,xmax,ymin,ymax];

            otras.fig3_escala=escala;
            otras.fig3_escala_old=escala;

      end

else %multisine

      if otras.fig1_tipo(2)==2 %Variable scale
            xmin=0;
            xmax=signals.msine_time(end);
            ymin=min(signals.msine);
            ymax=max(signals.msine);
            escala=[xmin,xmax,ymin-0.2*abs(ymin),ymax+0.2*abs(ymax)];
            otras.fig1_escala=escala;
            otras.fig1_escala_old=escala;
      end


      if otras.fig2_tipo(2)==2 %Variable scale
            xmin=min(signals.msine_ACF_con(:,1));
            xmax=max(signals.msine_ACF_con(:,1));
            ymin=min(signals.msine_ACF_con(:,2));
            ymax=max(signals.msine_ACF_con(:,2));
            rango=ymax-ymin;
            ymin=ymin-0.15*rango;
            ymax=ymax+0.15*rango;
            escala=[xmin,xmax,ymin,ymax];

            otras.fig2_escala=escala;
            otras.fig2_escala_old=escala;

            if otras.fig2_ACF_mostrar(3)==1  %Only plot positive lags
                  otras.fig2_escala(1)=0;
                  otras.fig2_escala_old(1)=0;
            end

      end

      if otras.fig3_tipo(2)==2 %Variable scale

            if otras.fig3_Xunits==1
                  Xc=signals.msine_spectrum_cont(:,2);
                  Xd=signals.msine_spectrum_dis(:,2);
            elseif otras.fig3_Xunits==2
                  Xc=2*pi*signals.msine_spectrum_cont(:,2);
                  Xd=2*pi*signals.msine_spectrum_dis(:,2);

            else
                  Xc=signals.msine_spectrum_cont(:,1);
                  Xd=signals.msine_spectrum_dis(:,1);
            end
```

```
            if par.max_harmonic==0
                  xmin=0;
            else
                  xmin=min(Xc);
            end
            if otras.fig3_Xscale==2
                  xmin=Xc(2);
            end
            xmax=max(Xc);
            ymin=min(signals.msine_spectrum_cont(:,3));
            ymax=max(signals.msine_spectrum_cont(:,3));

            if otras.fig3_Yscale==1
                  rango=ymax-ymin;
                  ymin=ymin-0.15*rango;
                  ymax=ymax+0.15*rango;
            else
                  rango=log10(ymax)-log10(ymin);
                  ymin=log10(ymin)-0.05*rango;
                  ymax=log10(ymax)+0.05*rango;
            end

            escala=[xmin,xmax,ymin,ymax];

            otras.fig3_escala=escala;
            otras.fig3_escala_old=escala;


      end



end

%-------------------------------------------------------------------------
% Resize function
%-------------------------------------------------------------------------
function resize(width, height,otras)
%resize

      initGraphics(width,height,otras);

%-------------------------------------------------------------------------
% Keydown function
%-------------------------------------------------------------------------
function (otras)= keydownHandler(_key,_width,_height,otras);
%keydownHandler

%-------------------------------------------------------------------------
% Menu function
%-------------------------------------------------------------------------
function (par,signals,otras)=choose_time_units(a,par,otras)
%choose_time_unit

            par.time_units=a;

            (signals,otras)=compute_signals(par,otras);
            otras=calcular_escalas_variables(par,signals,otras);


%-------------------------------------------------------------------------
function (par,signals,otras)=choose_signal(height,width,par,otras,a)
%choose_signal
```

```
	par.signal_type=a;
	(par, signals, otras) = inicializar(par,width,height);


%------------------------------------------------------------------------
function (par,otras)=choose_view(a,par,otras)
%choose_view

		par.view=a;

		if par.view==1 % Continuous
			otras.fig1_prbs_mostrar=[1,0,1,0,0];
			otras.fig2_ACF_mostrar=[1,0,1,1];
			otras.fig3_spectrum_mostrar=[1,0,0,0,0];
		elseif par.view==2 % Discrete
			otras.fig1_prbs_mostrar=[0,1,1,0,0];
			otras.fig2_ACF_mostrar=[0,1,1,1];
			otras.fig3_spectrum_mostrar=[0,1,0,0,0];
		elseif par.view==3 % Both
			otras.fig1_prbs_mostrar=[1,1,1,0,0];
			otras.fig2_ACF_mostrar=[1,1,1,1];
			otras.fig3_spectrum_mostrar=[1,1,0,0,0,0];

		end


%------------------------------------------------------------------------
function (otras)=select_sampling_time(height,width,otras)
%select_sampling_time

			otras.open_aux_window=1;
			otras.tipo_aux_window='5';
			initGraphics(width,height,otras);

			otras.fig1_prbs_mostrar=[1,1,1,0,0];
			otras.fig2_ACF_mostrar=[1,1,1,1];
			otras.fig3_spectrum_mostrar=[1,1,0,0,1,0];

%------------------------------------------------------------------------
function About
    opt = dialogset('Title', 'LCSD', ...
    'Prompt', 'PRBS & Multisine signals\n\n\t José Manuel Díaz Martínez\n\t Javier
Esteban Escaño \n\n(c) Copyright 2022');

    ok = dialog(opt);

%------------------------------------------------------------------------
function Ayuda
	%if ~launchurl('http://www2.uned.es/itfe/LCSD/LCSD.html')
	%	dialog('Cannot launch\n http://www2.uned.es/itfe/LCSD/LCSD.html');
	%end

%------------------------------------------------------------------------
% Figure functions
%------------------------------------------------------------------------
function drawParameters(par)
%drawParameters



	scale([0 50 0 70]);


	fonttitles = fontset('Font', face_font, 'Bold', true , 'Size', size_font, 'Color',
[0 0 0]);
```

```
if par.signal_type==1 %PRBS (MLBS)
	if ismac()==1
		text(0,65,'Parameters',fonttitles);
	else
		text(4.5,65,'Parameters',fonttitles);
	end

	text('\n');

	settabs([20,25]);
	% a
	a_min=par.prbs_a-abs(par.prbs_a*0.25);
	a_max=par.prbs_a+abs(par.prbs_a*0.25);

	textfield('Signal amplitude, A', '%.2g', par.prbs_a, 'r', id=1);
	slider('\t', par.prbs_a, [a_min,a_max],'', 'r', id=1);

	% n
	n_min=par.prbs_n-2;
	if n_min<=1
		n_min=2;
	end
	n_max=par.prbs_n+2;
	if n_max>11
		n_max=11;
	end

	textfield('Shift register length, n', '%d', par.prbs_n, 'r', id=2);
	slider('\t', par.prbs_n, [n_min,n_max],'', 'r', id=2);

	% Tsw
	Tsw_min=par.prbs_Tsw-2;
	if Tsw_min<1
		Tsw_min=1;
	end
	Tsw_max=par.prbs_Tsw+2;
	if Tsw_max>100
		Tsw_max=100;
	end
	textfield('Clock period, Tck', '%d', par.prbs_Tsw, 'r', id=3);
	slider('\t', par.prbs_Tsw, [Tsw_min,Tsw_max],'', 'r', id=3);

	% Nc
	Nc_min=par.prbs_Nc-2;
	if Nc_min<1
		Nc_min=1;
	end
	Nc_max=par.prbs_Nc+2;
	if Nc_max>100
		Nc_max=100;
	end
	textfield('Cycles number, Nc', '%d', par.prbs_Nc, 'r', id=4);
	slider('\t', par.prbs_Nc, [Nc_min,Nc_max],'', 'r', id=4);

else %Multisine

	if ismac()==1
		text(0,65,'Parameters',fonttitles);
	else
```

```
                text(4.5,67,'Parameters',fonttitles);
            end

            % Phase
            % Radio buttons de selección de escala fija o variable
            settabs([20,8,11]);
            button('Sinusoids      phase:\tZero\tRandom\tSchroeder',par.msine_dphase,
'radiobutton', '', 6);
        text('\n');

            settabs([20,32]);

            % a Amplitude
            a_min=par.msine_a-abs(par.msine_a*0.25);
            a_max=par.msine_a+abs(par.msine_a*0.25);

            textfield('Sinusoids amplitude, A', '%.2g', par.msine_a, 'r', id=1);
            slider('\t', par.msine_a, [a_min,a_max],'', 'r', id=1);

            % F Number of harmonics
            F_min=par.msine_F-1;
            if F_min<=1
                F_min=1;
            end
            F_max=par.msine_F+1;
            if F_max> par.msine_N/2
                F_max=par.msine_N/2;
            end

            textfield('Number of sinusoids, F', '%d', par.msine_F, 'r', id=2);
            slider('\t', par.msine_F, [F_min,F_max],'', 'r', id=2);

            % N
            N_min=par.msine_N-10;
            if N_min<2*par.msine_F
                N_min=2*par.msine_F;
            end
            N_max=par.msine_N+10;
            if N_max>2000
                N_max=1999;
            end

            textfield('Sequence length, N', '%d', par.msine_N, 'r', id=3);
            slider('\t', par.msine_N, [N_min,N_max],'', 'r', id=3);

            % T
            T_min=par.msine_T-2;
            if T_min<1
                T_min=1;
            end
            T_max=par.msine_T+2;
            if T_max>1000
                T_max=1000;
            end
            textfield('Signal period, T', '%d', par.msine_T, 'r', id=4);
            slider('\t', par.msine_T, [T_min,T_max],'', 'r', id=4);

            % Nc
            Nc_min=par.msine_Nc-2;
            if Nc_min<1
                Nc_min=1;
            end
            Nc_max=par.msine_Nc+2;
            if Nc_max>100
                Nc_max=100;
```

```
            end
            textfield('Cycles number, Nc', '%d', par.msine_Nc, 'r', id=5);
            slider('\t', par.msine_Nc, [Nc_min,Nc_max],'', 'r', id=5);









      end


    plotoption noframe;
    plotoption nolabel;
      plotoption nomargin;



%--------------------------------------------------
function (par,signals,otras) = dragParameters(par,signals,otras,id,x1)
%dragParameters

      if isempty(id)
            cancel;
      end

      actualizar=0;

      if par.signal_type==1 %PRBS (MLBS)

            switch id
            case 1
                  if abs(x1)>0
                        par.prbs_a = abs(x1);
                        actualizar=1;
                  else
                        dialog('WARNING: The signal amplitude has to be greater than
0');
                  end
            case 2
                  if  x1>1 && x1 <12
                        if x1 -floor(x1)>=0.5
                              par.prbs_n = ceil(x1);
                        else
                              par.prbs_n = floor(x1);
                        end

                        par.max_harmonic=3*(2^par.prbs_n-1);
                        actualizar=1;
                  else
                        dialog('WARNING:  The  shift  register  length  has  to  be  an
integer between 2 and 11.');
                  end
            case 3
                  if  x1>=1 && x1 <101
                        if x1-floor(x1)>=0.5
                              par.prbs_Tsw = ceil(x1);
                        else
                              par.prbs_Tsw = floor(x1);
                        end
                        actualizar=1;
                  else
```

```
                        dialog('WARNING: The clock period has to be an integer between
1 and 100.');
                end
            case 4
                if  x1>=1 && x1 <101
                    if x1-floor(x1)>=0.5
                        par.prbs_Nc = ceil(x1);
                    else
                        par.prbs_Nc = floor(x1);
                    end
                    actualizar=1;
                else
                    dialog('WARNING: The cycles number has to be an integer
between 1 and 100.');
                end
            end



      else %Multisine

            switch id
            case 1
                if abs(x1)>0
                    par.msine_a = abs(x1);
                    actualizar=1;
                else
                    dialog('WARNING: The sinusoids amplitude has to be greater
than 0');
                end

            case 2
                if  x1>0 && x1 <floor(par.msine_N/2)
                    if x1 -floor(x1)>=0.5
                        par.msine_F = ceil(x1);
                    else
                        par.msine_F = floor(x1);
                    end

                    par.max_harmonic=par.msine_F+0.25*par.msine_F;
                    actualizar=1;
                else
                    cadena=sprintf('WARNING: The number of sinusoids has to be an
integer in the range [1, N/2) = [1, %d)',floor(par.msine_N/2));
                    dialog(cadena);
                end


            case 3
                if  x1>2*par.msine_F && x1 <2001
                    if x1 -floor(x1)>=0.5
                        par.msine_N = ceil(x1);
                    else
                        par.msine_N = floor(x1);
                    end

                    par.msine_Ts=par.msine_T/par.msine_N;

                    actualizar=1;
                else

                    cadena=sprintf('WARNING: The sequence length has to be an
integer between 2F+1=%d and 2000',2*par.msine_F+1);
                    dialog(cadena);
                end
```

```
            case 4
                  if  x1>=1 && x1 <1001
                        if x1-floor(x1)>=0.5
                              par.msine_T = ceil(x1);
                        else
                              par.msine_T = floor(x1);
                        end
                        par.msine_Ts=par.msine_T/par.msine_N;
                        actualizar=1;
                  else
                        dialog('WARNING: The signal period has to be an integer
between 1 and 1000.');
                  end

            case 5
                  if  x1>=1 && x1 <101
                        if x1-floor(x1)>=0.5
                              par.msine_Nc = ceil(x1);
                        else
                              par.msine_Nc = floor(x1);
                        end
                        actualizar=1;
                  else
                        dialog('WARNING: The cycles number has to be an integer
between 1 and 100.');
                  end

            case 6
                  par.msine_dphase=x1;
                  actualizar=1;
            end


      end
      if actualizar==1
            (signals,otras)=compute_signals(par,otras);
             otras=calcular_escalas_variables(par,signals,otras);
      end

%-------------------------------------------------
function show_information(par,signals)
%show_information

      scale([0 20 0 20]);


      fonttitles = fontset('Font', face_font, 'Bold', true , 'Size', size_font, 'Color',
[0 0 0]);
      fonttext = fontset('Font', 'Arial' , 'Size', size_font, 'Color', [0 0 0]);

      if ismac()==1
            text(0,15,'Information',fonttitles);
      else
            text(1.4,19,'Information',fonttitles);
      end



   if par.signal_type==1 %PRBS (MLBS)

            cadena=sprintf('• MLBS of                        = %d bits',signals.prbs_N);
            text(-0.4,14,cadena,'l',fonttext);

            cadena='$\\rm{N=2^{n}-1}$';
```

```
            math(2.7,13.9,cadena,'l',Size=size_font);

            cadena=sprintf('• Signal period,                    = %d',signals.prbs_T);
            %time_units         1 Minutes; 0 Seconds; -1 deciseconds; -2 centiseconds;
-3 miliseconds;
            if par.time_units==1
                  cadena=[cadena,' min'];
            elseif par.time_units==0
                  cadena=[cadena,' s'];
            elseif par.time_units==-1
                  cadena=[cadena,' ds'];
            elseif par.time_units==-2
                  cadena=[cadena,' cs'];
            elseif par.time_units==-3
                  cadena=[cadena,' ms'];
            end
            text(-0.4,10,cadena,'l',fonttext);

            cadena='$\\rm{T = N \\, T_{ck}}$';
            math(3.9,9.7,cadena,'l',Size=size_font);

            %cadena=sprintf('Harmonic  spacing,                                  =  %.3g
Hz',1/signals.prbs_T);
            %text(-0.4,6,cadena,'l',fonttext);

            %cadena='$\\rm{\\Delta f = 1/T}$';
            %math(6.3,5.7,cadena,'l',Size=size_font);

            cadena=sprintf('• Effective frequency band,                         =
(%.3g, %.3g) Hz',signals.prbs_effqband(3),signals.prbs_effqband(4));
            text(-0.4,6,cadena,'l',fonttext);

            cadena='$(\\rm{\\frac{1}{N T_{ck}},\\, \\frac{1}{3 T_{ck}}})$';
            math(7.5,5.7,cadena,'l',Size=size_font);



            cadena=sprintf('•  Crest  factor  =  %.3g           PIPS  =    %.3g  %',1,
signals.prbs_PIPS);
            text(-0.4,1.7,cadena,'l',Size=size_font);

      else

      %     cadena=sprintf('• MLBS of                     = %d bits',signals.prbs_N);
      %     text(-0.4,14,cadena,'l',fonttext);

      %     cadena='$\\rm{N=2^{n}-1}$';
      %     math(2.7,13.9,cadena,'l',Size=size_font);


            cadena=sprintf('•  Sampling  Time,                                   =
%.3g',par.msine_T/par.msine_N);
            %time_units         1 Minutes; 0 Seconds; -1 deciseconds; -2 centiseconds;
-3 miliseconds;
            if par.time_units==1
                  cadena=[cadena,' min'];
            elseif par.time_units==0
                  cadena=[cadena,' s'];
            elseif par.time_units==-1
                  cadena=[cadena,' ds'];
            elseif par.time_units==-2
                  cadena=[cadena,' cs'];
            elseif par.time_units==-3
                  cadena=[cadena,' ms'];
            end
```

```
              text(-0.4,14,cadena,'l',fonttext);

              cadena='$\\rm{T_{s} = T/N}$';
              math(4.5,13.9,cadena,'l',Size=size_font);

              %cadena=sprintf('Harmonic  spacing,                                    = %.3g
Hz',1/signals.prbs_T);
              %text(-0.4,6,cadena,'l',fonttext);

              %cadena='$\\rm{\\Delta f = 1/T}$';
              %math(6.3,5.7,cadena,'l',Size=size_font);

              cadena=sprintf('• Effective frequency band,                                =
(%.3g, %.3g) Hz',signals.msine_effqband(3),signals.msine_effqband(4));
              text(-0.4,10,cadena,'l',fonttext);

              cadena='$(\\rm{ 1/T,\\, F/T\\, })$';
              math(7.5,9.7,cadena,'l',Size=size_font);



              cadena=sprintf('• Crest factor = %.3g',signals.msine_CF);
              text(-0.4,6,cadena,'l',Size=size_font);

              cadena=sprintf('• PIPS =  %.3g %',signals.msine_PIPS);
              text(-0.4,1.7,cadena,'l',Size=size_font);



       end

    plotoption noframe;
    plotoption nolabel;
       plotoption nomargin;
       plotoption math;

%-------------------------------------------------
function draw_register(par)

    plotoption noframe;
    plotoption nolabel;
       plotoption nomargin;

%-------------------------------------------------
function drawSignal(par,signals,otras)
%drawSignal

       if par.signal_type==1
              drawPRBS(par,signals,otras);
       else
              drawmsine(par,signals,otras);
       end

       switch par.time_units
       case 2
              label('Time','');
       case 1
              label('Time (min)','');
       case 0
              label('Time (s)','');
       case -1
              label('Time (ds)','');
       case -2
              label('Time (cs)','');
       case -3
```

```
		label('Time (ms)','');

	end

%-----------------------------------------------------
function (par,signals,otras)=drag_fig1(id,x0,x1,y1,par,signals,otras)
	actualizar=0;

	if id==1 && x1>1 %Tsw
		if x1-x0>0 %drag to the right
			par.prbs_Tsw=ceil(x1);
		else %drag to the left
			par.prbs_Tsw=floor(x1);
		end
		actualizar=1;

	elseif id==2 && y1>0 %a
		if otras.tipo_signal==1
			par.prbs_a=y1;
		else
			par.msine_a=y1;
		end
		actualizar=1;
	end

	if actualizar==1
		(signals,otras)=compute_signals(par,otras);
	end


%-----------------------------------------------------
function (cursor,par,signals,otras)=over_fig1(id,x0,y0,par,signals,otras)
%overSignal
	if isempty(id)
		otras.fig1_id=[];
		otras.fig1_x0=[];
		otras.fig1_y0=[];
		cursor=false;
	else
		otras.fig1_id=id;
		otras.fig1_x0=x0;
		otras.fig1_y0=y0;
		cursor=true;
	end


%-------------------------------------------------
function drawPRBS(par,signals,otras)
%drawPRBS

	scale(otras.fig1_escala);
	sc=scale;



	%Distinguir los ciclos pares de la señal con un sombreado de color gris claro
	if otras.fig1_prbs_mostrar(4)==1
		for h=1:par.prbs_Nc
			if rem(h,2)==0 %para ciclos pares
				X_C=[signals.prbs_T*(h-
1),signals.prbs_T*h,signals.prbs_T*h,signals.prbs_T*(h-1)];
				Y_C=[sc(3),sc(3),sc(4),sc(4)];

				st=plotset(Fill=true,FillColor='#D8FEFD');
```

```
                        plot(X_C,Y_C,st)
                end
            end
        end



    %Plot vertical lines at x={Tsw,2Tsw,3Tsw,....}
        if otras.fig1_prbs_mostrar(3)==1
            N=(signals.prbs_T*par.prbs_Nc)/par.prbs_Tsw;
            st=plotset(LineStyle=':',Color='black');
            for h=1:N
                line([1,0],h*par.prbs_Tsw,st);

            end
        end

        %Plot horizontal lines at y=a and vertical line at x=Tsw. This lines can be drag
to configure the amplitude and the clock period
        if otras.fig1_prbs_mostrar(5)==1
            st=plotset(LineStyle='-', Color='red');
            line([1,0],par.prbs_Tsw,st,1);
            line([0,1],par.prbs_a,st,2)
        end



        %Plot continuous PRBS
        if otras.fig1_prbs_mostrar(1)==1
            st=plotset(LineStyle='', LineWidth=1.7,Color='red');
            plot(signals.prbs_mod_time', signals.prbs_mod',st);
        end

        %Plot discrete PRBS
        if otras.fig1_prbs_mostrar(2)==1
            %st=plotset(Marker='o',MarkerFaceColor='red',MarkerEdgeColor='red');
            st=plotset(Marker='o', MarkerEdgeColor='blue');
            plot(signals.prbs_dis_time', signals.prbs_dis',st);
        end



        plotoption nolabel;



%-------------------------------------------------
function drawmsine(par,signals,otras)
%drawmsine

        scale(otras.fig1_escala);
        sc=scale;




            %for h=0:(250*par.msine_Nc-1)
            %      time(h+1)=h*par.msine_T/250;
            %
        y(h+1)=par.msine_a*cos(((2*pi)/par.msine_T)*time(h+1))+par.msine_a*cos(((2*pi)*2
/par.msine_T)*time(h+1)-3.141592653589793);
```

```
            %end

            %st=plotset(LineStyle='', LineWidth=1.7,Color='red');
            %plot(time, y,st);

      %Dibujar ejes:
      %st=plotset(Color='black',LineStyle=':');
      %plot([sc(1),sc(2)],[0,0],st); %Eje X
      %plot([0,0],[sc(3),sc(4)],st); %Eje Y



      %Distinguir los ciclos pares de la señal con un sombreado de color gris claro
      if otras.fig1_prbs_mostrar(4)==1
            for h=1:par.msine_Nc
                  if rem(h,2)==0 %para ciclos pares
                        X_C=[par.msine_T*(h-
1),par.msine_T*h,par.msine_T*h,par.msine_T*(h-1)];
                        Y_C=[sc(3),sc(3),sc(4),sc(4)];

                        st=plotset(Fill=true,FillColor='#D8FEFD');

                        plot(X_C,Y_C,st)
                  end
            end
      end


      %Plot horizontal lines at y=a and vertical line at x=Tsw. This lines can be drag
to configure the amplitude and the clock period
      if otras.fig1_prbs_mostrar(5)==1
            st=plotset(LineStyle='-', Color='red');
            line([0,1],par.msine_a,st,2)
      end




      %Plot continuous msine
      if otras.fig1_prbs_mostrar(1)==1
            st=plotset(LineStyle='', LineWidth=1.7,Color='red');
            plot(signals.msine_time', signals.msine',st);
      end

      %Plot discrete msine
      if otras.fig1_prbs_mostrar(2)==1
            %st=plotset(Marker='o',MarkerFaceColor='red',MarkerEdgeColor='red');
            st=plotset(Marker='o', MarkerEdgeColor='blue');
            plot(signals.msine_dis_time', signals.msine_dis',st);
      end




      plotoption nolabel;


%-------------------------------------------------
function drawACF(par,signals,otras)
%drawACF

      if par.signal_type==1
            drawACF_PRBS(par,signals,otras);
      else
```

```
            drawACF_msine(par,signals,otras);
      end

%-------------------------------------------------
function drawACF_PRBS(par,signals,otras)
%drawACF_PRBS

      if otras.fig2_ACF_mostrar(3)==1  %Only plot positive lags
            ind=ceil(length(signals.prbs_ACF(:,1)-1)/2);
      else
            ind=1;
      end


      scale(otras.fig2_escala);
      sc=scale;



      %Dibujar ejes:
      st=plotset(Color='black',LineStyle=':');
      plot([sc(1),sc(2)],[0,0],st); %Eje X
      plot([0,0],[sc(3),sc(4)],st); %Eje Y

      %Plot vertical lines at x={Tsw,2Tsw,3Tsw,....}
      if otras.fig2_ACF_mostrar(4)==1
            N=(signals.prbs_T*par.prbs_Nc)/par.prbs_Tsw;
            st=plotset(LineStyle=':',Color='black');
            for h=1:N
                  line([1,0],h*par.prbs_Tsw,st);
            end
            if otras.fig2_ACF_mostrar(3)==0
                  for h=-N:-1
                        line([1,0],h*par.prbs_Tsw,st);
                  end
            end
      end

      %Plot continuous ACF
      if otras.fig2_ACF_mostrar(1)==1
            st=plotset(LineStyle='', LineWidth=1.7,Color='red');
            plot(signals.prbs_ACF(ind:end,1)', signals.prbs_ACF(ind:end,2)', st);
      end

      %Plot discrete ACF
      if otras.fig2_ACF_mostrar(2)==1
            st=plotset(Stems=true, LineWidth=1.5, Color='blue');
            plot(signals.prbs_ACF(:,1)', signals.prbs_ACF(:,2)', st);
      end


      label('Lags','');

      plotoption nolabel;

%-------------------------------------------------
function drawACF_msine(par,signals,otras)
%drawACF_msine

      if otras.fig2_ACF_mostrar(3)==1  %Only plot positive lags
            ind=ceil(length(signals.msine_ACF_con(:,1)-1)/2);

      else
            ind=1;
      end
```

```
	scale(otras.fig2_escala);
	sc=scale;



	%Dibujar ejes:
	st=plotset(Color='black',LineStyle=':');
	plot([sc(1),sc(2)],[0,0],st); %Eje X
	plot([0,0],[sc(3),sc(4)],st); %Eje Y



	%Plot continuous ACF
	if otras.fig2_ACF_mostrar(1)==1
		st=plotset(LineStyle='', LineWidth=1.7,Color='red');
		%st=plotset(Stems=true, LineWidth=1.5, Color='red');
		plot(signals.msine_ACF_con(ind:end,1)',
signals.msine_ACF_con(ind:end,2)', st);
	end


	%Plot discrete ACF
	if otras.fig2_ACF_mostrar(2)==1
		st=plotset(Stems=true, LineWidth=1.5, Color='blue');
		plot(signals.msine_ACF_dis(:,1)', signals.msine_ACF_dis(:,2)', st);
	end



	label('Lags','');

	plotoption nolabel;
%-------------------------------------------------
function drawSpectrum(par,signals,otras)
%drawSpectrum

	if par.signal_type==1
		drawSpectrum_PRBS(par,signals,otras);
	else
		drawSpectrum_msine(par,signals,otras);
	end


%-------------------------------------------------
function drawSpectrum_PRBS(par,signals,otras)
%drawSpectrum_PRBS

	if otras.fig3_Xunits==1
		label('Frequency (Hz)','');
		Xc=signals.prbs_spectrum_cont(:,2);
		Xd=signals.prbs_spectrum_dis(:,2);

	X_Cr=[signals.prbs_Nyquist_freq(2),signals.prbs_spectrum_dis(end,2),signals.prbs
_spectrum_dis(end,2),signals.prbs_Nyquist_freq(2),signals.prbs_Nyquist_freq(2)];

		X_Cl=[-signals.prbs_spectrum_dis(end,2),-signals.prbs_Nyquist_freq(2),-
signals.prbs_Nyquist_freq(2),-signals.prbs_spectrum_dis(end,2),-
signals.prbs_spectrum_dis(end,2)];
		NF=signals.prbs_Nyquist_freq(2);

	X_EFBr=[signals.prbs_effqband(3),signals.prbs_effqband(4),signals.prbs_effqband(
4),signals.prbs_effqband(3),signals.prbs_effqband(3)];
```

```
            X_EFBl=[-signals.prbs_effqband(4),-signals.prbs_effqband(3),-
signals.prbs_effqband(3),-signals.prbs_effqband(4),-signals.prbs_effqband(4)];



      elseif otras.fig3_Xunits==2
            label('Frequency (rad/s)','');
            Xc=2*pi*signals.prbs_spectrum_cont(:,2);
            Xd=2*pi*signals.prbs_spectrum_dis(:,2);

      X_Cr=2*pi*[signals.prbs_Nyquist_freq(2),signals.prbs_spectrum_dis(end,2),signals
.prbs_spectrum_dis(end,2),signals.prbs_Nyquist_freq(2),signals.prbs_Nyquist_freq(2)];

            X_Cl=2*pi*[-signals.prbs_spectrum_dis(end,2),-
signals.prbs_Nyquist_freq(2),-signals.prbs_Nyquist_freq(2),-
signals.prbs_spectrum_dis(end,2),-signals.prbs_spectrum_dis(end,2)];
            NF=2*pi*signals.prbs_Nyquist_freq(2);

      X_EFBr=2*pi*[signals.prbs_effqband(3),signals.prbs_effqband(4),signals.prbs_effq
band(4),signals.prbs_effqband(3),signals.prbs_effqband(3)];
            X_EFBl=2*pi*[-signals.prbs_effqband(4),-signals.prbs_effqband(3),-
signals.prbs_effqband(3),-signals.prbs_effqband(4),-signals.prbs_effqband(4)];



      else
            label('Harmonic number','');
            Xc=signals.prbs_spectrum_cont(:,1);
            Xd=signals.prbs_spectrum_dis(:,1);

      X_Cr=[signals.prbs_Nyquist_freq(1),signals.prbs_spectrum_dis(end,1),signals.prbs
_spectrum_dis(end,1),signals.prbs_Nyquist_freq(1),signals.prbs_Nyquist_freq(1)];

            X_Cl=[-signals.prbs_spectrum_dis(end,1),-signals.prbs_Nyquist_freq(1),-
signals.prbs_Nyquist_freq(1),-signals.prbs_spectrum_dis(end,1),-
signals.prbs_spectrum_dis(end,1)];
            NF=signals.prbs_Nyquist_freq(1);

      X_EFBr=[signals.prbs_effqband(1),signals.prbs_effqband(2),signals.prbs_effqband(
2),signals.prbs_effqband(1),signals.prbs_effqband(1)];
            X_EFBl=[-signals.prbs_effqband(2),-signals.prbs_effqband(1),-
signals.prbs_effqband(1),-signals.prbs_effqband(2),-signals.prbs_effqband(2)];


      end

      if otras.fig3_Xscale==1 && otras.fig3_Yscale==1
            tipo='linlin';
      elseif otras.fig3_Xscale==2 && otras.fig3_Yscale==1
            tipo='loglin';
      elseif otras.fig3_Xscale==1 && otras.fig3_Yscale==2
            tipo='linlog';
      else
            tipo='loglog';
      end



      scale(tipo,otras.fig3_escala);
      sc=scale;



      %Dibujar en verde claro la banda de frecuencia efectiva
```

```
	if otras.fig3_spectrum_mostrar(4)==1
		%Single-side spectrum
		Y_C=[sc(3),sc(3),sc(4),sc(4),sc(3)];
		st=plotset(Fill=true,FillColor='#CDFAE4');
		plot(X_EFBr,Y_C,st)

		%Dibujar frecuencia de Nyquist o frecuencia de plegamiento
		%st=plotset(LineStyle=':',Color='black');
		%line([1,0],NF,st);

			if otras.fig3_spectrum_mostrar(3)==1 %Double-side spectrum

				st=plotset(Fill=true,FillColor='#CDFAE4');
				plot(X_EFBl,Y_C,st)

				%Dibujar frecuencia de Nyquist o frecuencia de plegamiento
				%st=plotset(LineStyle=':',Color='black');
				%line([1,0],-NF,st);

			end


	end




	%Dibujar en amarillo claro la zona donde los armónicos del espectro de la señal
discreta no aportan información
	if otras.fig3_spectrum_mostrar(5)==1
		%Single-side spectrum
		Y_C=[sc(3),sc(3),sc(4),sc(4),sc(3)];
		st=plotset(Fill=true,FillColor='#FEFECD');
		plot(X_Cr,Y_C,st)

		%Dibujar frecuencia de Nyquist o frecuencia de plegamiento
		st=plotset(LineStyle=':',Color='black');
		line([1,0],NF,st);

			if otras.fig3_spectrum_mostrar(3)==1 %Double-side spectrum

				st=plotset(Fill=true,FillColor='#FEFECD');
				plot(X_Cl,Y_C,st)

				%Dibujar frecuencia de Nyquist o frecuencia de plegamiento
				st=plotset(LineStyle=':',Color='black');
				line([1,0],-NF,st);

			end


	end

	%estilo_stem=plotset(Stems=true,Color='blue',MarkerFaceColor='blue');
	if otras.fig3_spectrum_mostrar(1)==1
		if otras.fig3_Yscale==2
			st=plotset(Marker='o', MarkerEdgeColor='red');
		else
			st=plotset(Stems=true,LineWidth=1.5, Color='red');
		end
		plot(Xc', signals.prbs_spectrum_cont(:,3)',st);
	end
```

```
	%estilo_stem=plotset(Stems=true,Color='blue',MarkerFaceColor='blue');
	if otras.fig3_spectrum_mostrar(2)==1
		if otras.fig3_Yscale==2
			st=plotset(Marker='o', MarkerEdgeColor='blue');
		else
			st=plotset(Stems=true,LineWidth=1.5, Color='blue');
		end


		plot(Xd', signals.prbs_spectrum_dis(:,3)',st);
	end


	if otras.fig3_legend==2
		if otras.fig3_spectrum_mostrar(2)==1
			estilo_legend= {Color='Red',Marker='o'; Color='Blue',Marker='o'};
			legend('Continuous\nDiscrete',estilo_legend);
		else
			estilo_legend= {Color='Red',Marker='o'};
			legend('Continuous',estilo_legend);
		end
	end



	plotoption nolabel;

%-------------------------------------------------
function drawSpectrum_msine(par,signals,otras)
%drawSpectrum_msine

	if otras.fig3_Xunits==1
		label('Frequency (Hz)','');
		Xc=signals.msine_spectrum_cont(:,2);
		Xd=signals.msine_spectrum_dis(:,2);

	X_Cr=[signals.msine_Nyquist_freq(2),signals.msine_spectrum_dis(end,2),signals.ms
ine_spectrum_dis(end,2),signals.msine_Nyquist_freq(2),signals.msine_Nyquist_freq(2)];

		X_Cl=[-signals.msine_spectrum_dis(end,2),-signals.msine_Nyquist_freq(2),-
signals.msine_Nyquist_freq(2),-signals.msine_spectrum_dis(end,2),-
signals.msine_spectrum_dis(end,2)];
		NF=signals.msine_Nyquist_freq(2);

	X_EFBr=[signals.msine_effqband(3),signals.msine_effqband(4),signals.msine_effqba
nd(4),signals.msine_effqband(3),signals.msine_effqband(3)];
		X_EFBl=[-signals.msine_effqband(4),-signals.msine_effqband(3),-
signals.msine_effqband(3),-signals.msine_effqband(4),-signals.msine_effqband(4)];



	elseif otras.fig3_Xunits==2
		label('Frequency (rad/s)','');
		Xc=2*pi*signals.msine_spectrum_cont(:,2);
		Xd=2*pi*signals.msine_spectrum_dis(:,2);

	X_Cr=2*pi*[signals.msine_Nyquist_freq(2),signals.msine_spectrum_dis(end,2),signa
ls.msine_spectrum_dis(end,2),signals.msine_Nyquist_freq(2),signals.msine_Nyquist_freq(
2)];
		X_Cl=2*pi*[-signals.msine_spectrum_dis(end,2),-
signals.msine_Nyquist_freq(2),-signals.msine_Nyquist_freq(2),-
signals.msine_spectrum_dis(end,2),-signals.msine_spectrum_dis(end,2)];
		NF=2*pi*signals.msine_Nyquist_freq(2);
```

```
      X_EFBr=2*pi*[signals.msine_effqband(3),signals.msine_effqband(4),signals.msine_e
ffqband(4),signals.msine_effqband(3),signals.msine_effqband(3)];
            X_EFBl=2*pi*[-signals.msine_effqband(4),-signals.msine_effqband(3),-
signals.msine_effqband(3),-signals.msine_effqband(4),-signals.msine_effqband(4)];



      else
            label('Harmonic number','');
            Xc=signals.msine_spectrum_cont(:,1);
            Xd=signals.msine_spectrum_dis(:,1);

      X_Cr=[signals.msine_Nyquist_freq(1),signals.msine_spectrum_dis(end,1),signals.ms
ine_spectrum_dis(end,1),signals.msine_Nyquist_freq(1),signals.msine_Nyquist_freq(1)];

            X_Cl=[-signals.msine_spectrum_dis(end,1),-signals.msine_Nyquist_freq(1),-
signals.msine_Nyquist_freq(1),-signals.msine_spectrum_dis(end,1),-
signals.msine_spectrum_dis(end,1)];
            NF=signals.msine_Nyquist_freq(1);

      X_EFBr=[signals.msine_effqband(1),signals.msine_effqband(2),signals.msine_effqba
nd(2),signals.msine_effqband(1),signals.msine_effqband(1)];
            X_EFBl=[-signals.msine_effqband(2),-signals.msine_effqband(1),-
signals.msine_effqband(1),-signals.msine_effqband(2),-signals.msine_effqband(2)];


      end

      if otras.fig3_Xscale==1 && otras.fig3_Yscale==1
            tipo='linlin';
      elseif otras.fig3_Xscale==2 && otras.fig3_Yscale==1
            tipo='loglin';
      elseif otras.fig3_Xscale==1 && otras.fig3_Yscale==2
            tipo='linlog';
      else
            tipo='loglog';
      end




      scale(tipo,otras.fig3_escala);
      sc=scale;


      %Dibujar en verde claro la banda de frecuencia efectiva
      if otras.fig3_spectrum_mostrar(4)==1
            %Single-side spectrum
            Y_C=[sc(3),sc(3),sc(4),sc(4),sc(3)];
            st=plotset(Fill=true,FillColor='#CDFAE4');
            plot(X_EFBr,Y_C,st)

            %Dibujar frecuencia de Nyquist o frecuencia de plegamiento
            %st=plotset(LineStyle=':',Color='black');
            %line([1,0],NF,st);

                  if otras.fig3_spectrum_mostrar(3)==1 %Double-side spectrum

                        st=plotset(Fill=true,FillColor='#CDFAE4');
                        plot(X_EFBl,Y_C,st)

                        %Dibujar frecuencia de Nyquist o frecuencia de plegamiento
```

```
                        %st=plotset(LineStyle=':',Color='black');
                        %line([1,0],-NF,st);

                end


        end



        %Dibujar en amarillo claro la zona donde los armónicos del espectro de la señal
discreta no aportan información
        if otras.fig3_spectrum_mostrar(5)==1
                %Single-side spectrum
                Y_C=[sc(3),sc(3),sc(4),sc(4),sc(3)];
                st=plotset(Fill=true,FillColor='#FEFECD');
                plot(X_Cr,Y_C,st)

                %Dibujar frecuencia de Nyquist o frecuencia de plegamiento
                st=plotset(LineStyle=':',Color='black');
                line([1,0],NF,st);

                        if otras.fig3_spectrum_mostrar(3)==1 %Double-side spectrum

                                st=plotset(Fill=true,FillColor='#FEFECD');
                                plot(X_Cl,Y_C,st)

                                %Dibujar frecuencia de Nyquist o frecuencia de plegamiento
                                st=plotset(LineStyle=':',Color='black');
                                line([1,0],-NF,st);

                        end


        end

        %estilo_stem=plotset(Stems=true,Color='blue',MarkerFaceColor='blue');
        if otras.fig3_spectrum_mostrar(1)==1
                if otras.fig3_Yscale==2
                        st=plotset(Marker='o', MarkerEdgeColor='red');
                else
                        st=plotset(Stems=true,LineWidth=1.5, Color='red');
                end
                plot(Xc', signals.msine_spectrum_cont(:,3)',st);
        end



        %estilo_stem=plotset(Stems=true,Color='blue',MarkerFaceColor='blue');
        if otras.fig3_spectrum_mostrar(2)==1
                if otras.fig3_Yscale==2
                        st=plotset(Marker='o', MarkerEdgeColor='blue');
                else
                        st=plotset(Stems=true,LineWidth=1.5, Color='blue');
                end


                plot(Xd', signals.msine_spectrum_dis(:,3)',st);
        end


        if otras.fig3_legend==2
                if otras.fig3_spectrum_mostrar(2)==1
```

```
                estilo_legend= {Color='Red',Marker='o'; Color='Blue',Marker='o'};
                legend('Continuous\nDiscrete',estilo_legend);
        else
                estilo_legend= {Color='Red',Marker='o'};
                legend('Continuous',estilo_legend);
        end
    end




    plotoption nolabel;

%-------------------------------------------------
function drawDFT(par,signals,otras)
%drawDFT

    %estilo_stem=plotset(Stems=true,Color='blue',MarkerFaceColor='blue');
    estilo_stem=plotset(Stems=true,Color='blue');

    plot(signals.prbs_DFT(:,1)', signals.prbs_DFT(:,3)',estilo_stem);


    xmin=0;
    xmax=max(signals.prbs_DFT(:,1));
    ymin=min(signals.prbs_DFT(:,3));
    ymax=max(signals.prbs_DFT(:,3));

    rango=ymax-ymin;
    ymin=ymin-0.15*rango;
    ymax=ymax+0.15*rango;
    scale([xmin,xmax,ymin,ymax])

    plotoption nolabel;

%-------------------------------------------------
function drawTitle(par)
    scale([0 50 0 50]);
    % This function shows the title of the application
    %fonttitles =  Gestion_fuentes (FUENTE_TITULOS);
      fonttitles  =  fontset('Font',  face_font,  'Bold',  true,  'Size',  size_font+4,
'Color', [0 0 0]);
    if par.signal_type==1
            text(25,25,'Pseudo-random binary signal (PRBS)',fonttitles);
      else
            text(25,25,'Multisine signal',fonttitles);
      end

    plotoption noframe;
    plotoption nolabel;
      plotoption nomargin;


%-------------------------------------------------
function drawTitleSIG
%drawTitleSIG

      scale([0 50 0 50]);
    fonttitles = fontset('Font', face_font, 'Bold', true, 'Size', size_font, 'Color',
[0 0 0]);


    text(25,25,'Signal',fonttitles);
```

```
      plotoption noframe;
      plotoption nolabel;
      plotoption nomargin;
%-----------------------------------------------
function drawTitleACF
%drawTitleACF

      scale([0 50 0 50]);
    fonttitles = fontset('Font', face_font, 'Bold', true, 'Size', size_font, 'Color',
[0 0 0]);


      text(25,25,'Autocorrelation function',fonttitles);


      plotoption noframe;
      plotoption nolabel;
      plotoption nomargin;
%-----------------------------------------------
function drawTitlePS
%drawTitlePS

      scale([0 50 0 50]);
    fonttitles = fontset('Font', face_font, 'Bold', true, 'Size', size_font, 'Color',
[0 0 0]);


      text(25,25,'Power spectrum',fonttitles);


      plotoption noframe;
      plotoption nolabel;
      plotoption nomargin;
%-----------------------------------------------
function drawTitleDFT
%drawTitleDFT

      scale([0 50 0 50]);
    fonttitles = fontset('Font', face_font, 'Bold', true, 'Size', size_font, 'Color',
[0 0 0]);


      text(25,25,'DFT magnitude',fonttitles);


      plotoption noframe;
      plotoption nolabel;
      plotoption nomargin;

%-------------------------------------------------------------------------
% Virtual Auxiliar Window Functions
%-------------------------------------------------------------------------
function (otras)=pulsar_icono_settings(id,height,width,modo,otras)
%pulsar_icono_settings


      if otras.open_aux_window==0

            if modo=='1' && id==100 %Signal
                  otras.open_aux_window=1;
                  otras.tipo_aux_window=modo;
                  initGraphics(width,height,otras);

            elseif modo=='2' && id==100 %ACF
```

```
                otras.open_aux_window=1;
                otras.tipo_aux_window=modo;
                initGraphics(width,height,otras);

        elseif modo=='3' && id==100  %Power spectrum

                otras.open_aux_window=1;
                otras.tipo_aux_window=modo;
                initGraphics(width,height,otras);

        elseif modo=='4' && id==100  %DFT magnitude

                otras.open_aux_window=1;
                otras.tipo_aux_window=modo;
                initGraphics(width,height,otras);

        end

    else
        otras.open_aux_window=0;
        initGraphics(width,height,otras);
    end

%-------------------------------------------------------------------------------
function draw_auxiliar_window(par,signals,otras)
%draw_auxiliar_window

    if  otras.tipo_aux_window=='1' %Signal
        draw_settings_fig1(otras);

    elseif otras.tipo_aux_window=='2' %ACF Autocorrelation function
        draw_settings_fig2(otras);

    elseif otras.tipo_aux_window=='3' %Power spectrum
        draw_settings_fig3(otras);

    elseif otras.tipo_aux_window=='4'      %DFT magnitude
        draw_settings_fig4(otras);

    elseif otras.tipo_aux_window=='5'      %Sampling time
        draw_settings_fig5(otras,par,signals);

    end



%-------------------------------------------------------------------------------
function
(otras,par,signals)=drag_auxiliar_window(id,ix,nb,x1,y1,height,width,otras,par,signals
)
%drag_auxiliar_window


    if  otras.tipo_aux_window=='1'
        (otras)=drag_settings_fig1(id,ix,nb,x1,height,width,otras);

    elseif otras.tipo_aux_window=='2'
        (otras)=drag_settings_fig2(id,ix,nb,x1,height,width,otras);

    elseif otras.tipo_aux_window=='3'
        (otras)=drag_settings_fig3(id,ix,nb,x1,height,width,otras);

    elseif otras.tipo_aux_window=='4'
        (otras)=drag_settings_fig4(id,ix,nb,x1,height,width,otras);
```

```
       elseif otras.tipo_aux_window=='5'

       (otras,par,signals)=drag_settings_fig5(id,ix,nb,x1,height,width,otras,par,signal
s);



       end


%-------------------------------------------------------------------------------
function
(otras,par,signals)=mouseup_auxiliar_window(id,ix,nb,x1,y1,height,width,otras,par,sign
als)
%mouseup_auxiliar_window


       if  otras.tipo_aux_window=='1'

       (otras)=mouseup_settings_fig1(id,ix,nb,x1,height,width,otras,par,signals);

       elseif otras.tipo_aux_window=='2'

       (otras)=mouseup_settings_fig2(id,ix,nb,x1,height,width,otras,par,signals);


       elseif otras.tipo_aux_window=='3'

       (otras,par,signals)=mouseup_settings_fig3(id,ix,nb,x1,height,width,otras,par,sig
nals);

       elseif otras.tipo_aux_window=='4'

       (otras)=mouseup_settings_fig4(id,ix,nb,x1,height,width,otras,par,signals);

       elseif otras.tipo_aux_window=='5'

       (otras,par,signals)=mouseup_settings_fig5(id,ix,nb,x1,height,width,otras,par,sig
nals);

       end


%---------------------------------------------------------------------
function draw_settings_fig1(otras)
%draw_settings_fig1

       par1='{t_{\\min }}'; %tmin
       par2='{t_{\\max }}'; %tmax
       par3='{y_{\\min }}'; %ymin
       par4='{y_{\\max }}'; %ymax


       campo1=otras.fig1_escala(1);
       campo2=otras.fig1_escala(2);
       campo3=otras.fig1_escala(3);
       campo4=otras.fig1_escala(4);

       P=5;
       campo1_min=campo1-P;
       campo1_max=campo1+P;
       campo2_min=campo2-P;
       campo2_max=campo2+P;
```

```
campo3_min=campo3-P;
campo3_max=campo3+P;
campo4_min=campo4-P;
campo4_max=campo4+P;



                plotoption noframe;
                   plotoption nolabel;
                   plotoption nomargin;
                   plotoption math;
                subplotstyle('plotbg', FillColor=color_fondo_vauxiliar);
                   subplotstyle('controlbg', FillColor=color_fondo_vauxiliar);
                   subplotstyle('plotmargin',FixedControlVPos=true);


                   %Dimensiones de la zona donde aparecerá la información
                   xmin=0;xmax=20;
                   ymin=0;ymax=20;
                   scale([0 20 0 20]);


                   %Título de la ventana
          estilo_texto1                                                    =
fontset(Font=face_font,Bold=true,Size=size_font,Color='black');
          text(10,20,'\nSignal - Figure settings',estilo_texto1)
                text('\n\n');


                   % Radio buttons de selección de escala fija o variable
                   settabs([8,12,10]);
                   button('             Scale:\tLocked\tVariable',otras.fig1_tipo(2),
'radiobutton', '', 1);


                   %Campos xmin, xmax, ymin, ymax
                   sep2campos='   value:\t00000000000\tvalueXX:\t00000000000';

                   seplabel1='$\\quad';
                   seplabel22='$\t$\\quad\\,\\,';

                   settabs(sep2campos);
                   labels = [seplabel1,par1,seplabel22,par2,'$\t'];
                   textfield(labels, '%.1g%.1g', real([campo1, campo2]), '', 2)
                   slider('            \t            ',            [campo1;campo2],
[campo1_min,campo1_max;campo2_min,campo2_max], '', 'gg', 2);


                   settabs(sep2campos);
                   labels = [seplabel1,par3,seplabel22,par4,'$\t'];
                   textfield(labels, '%.1g%.1g', real([campo3, campo4]), '', 3)
                   slider('            \t            ',            [campo3;campo4],
[campo3_min,campo3_max;campo4_min,campo4_max], '', 'gg', 3);

                   text('\n')

                   % Checkmarks de opciones
                   settabs([2,30]);
                   button('\t   Plot   continuous   PRBS',otras.fig1_prbs_mostrar(1),
'ckeckmark', '', 4);

                   button('\t    Plot    discrete    PRBS',otras.fig1_prbs_mostrar(2),
'ckeckmark', '', 5);
```

```
                    if otras.tipo_signal==1
                          button('\t            Plot            clock            period
grid',otras.fig1_prbs_mostrar(3), 'ckeckmark', '', 6);
                    end

                    button('\t     Distinguish     cycles',otras.fig1_prbs_mostrar(4),
'ckeckmark', '', 7);

                    %AQUI
                    if otras.tipo_signal==1
                          button('\t       Drag       amplitude       and       clock
period',otras.fig1_prbs_mostrar(5), 'ckeckmark', '', 8);
                    else
                          button('\t    Drag    amplitude',otras.fig1_prbs_mostrar(5),
'ckeckmark', '', 8);
                    end









                    text('\n')

                    %Botones Default, Cancel y OK
                    settabs([10,10,10]);
                    pushbutton('\t Cancel \t OK \t', 'rg', 9);

                    text('\n')

%-----------------------------------------------------------------------
function (otras)=drag_settings_fig1(id,ix,nb,x1,height,width,otras)


              switch id

              case 2
                    if otras.fig1_tipo(2)==1 %Si es escala fija
                          if nb==1 %xmin
                                if x1 >= 0 && x1 < otras.fig1_escala(2)
                                      otras.fig1_escala(1)=x1;

                                end

                          elseif nb==2 %xmax
                                if x1 > otras.fig1_escala(1)
                                      otras.fig1_escala(2)=x1;

                                end
                          end
                    elseif       otras.fig1_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                          dialog('Warning:  The axis limits can not be configured if
the scale type is variable');
                    end
              case 3
                    if otras.fig1_tipo(2)==1 %Si es escala fija
                          if nb==1 %ymin
                                if x1 < otras.fig1_escala(4)
                                      otras.fig1_escala(3)=x1;
                                end
                          elseif nb==2 %ymax
                                if x1 > otras.fig1_escala(3)
                                      otras.fig1_escala(4)=x1;
```

```
                                end
                        end
                elseif      otras.fig1_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');

                end


        end



%--------------------------------------------------------------------
function (otras)=mouseup_settings_fig1(id,ix,nb,x1,height,width,otras,par,signals);
%mouseup_settings_fig1


        switch id
        case 1 % Radio buttons para selección de la escala
                otras.fig1_tipo(2)=x1;

                if x1==2 %Variable
                        otras=calcular_escalas_variables(par,signals,otras);
                end

        case 2
                if otras.fig1_tipo(2)==1 %Si es escala fija
                        if nb==1 %xmin
                                if x1 >= 0 && x1 < otras.fig1_escala(2)
                                        otras.fig1_escala(1)=x1;
                                end

                        elseif nb==2 %xmax
                                if x1 > otras.fig1_escala(1)
                                        otras.fig1_escala(2)=x1;

                                end
                        end
                elseif      otras.fig1_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');
                end
        case 3
                if otras.fig1_tipo(2)==1 %Si es escala fija
                        if nb==1 %ymin
                                if x1 < otras.fig1_escala(4)
                                        otras.fig1_escala(3)=x1;
                                end
                        elseif nb==2 %ymax
                                if x1 > otras.fig1_escala(3)
                                        otras.fig1_escala(4)=x1;
                                end
                        end
                elseif      otras.fig1_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');

                end


        case 4 %Checkmark Plot Continuous PRBS
```

```
                otras.fig1_prbs_mostrar(1)=x1;

            case 5 %Checkmark Plot Discrete PRBS
                otras.fig1_prbs_mostrar(2)=x1;

            case 6 %Checkmark Plot clock period grid
                otras.fig1_prbs_mostrar(3)=x1;

            case 7 %Checkmark Distinguish cycles
                otras.fig1_prbs_mostrar(4)=x1;

            case 8 %Checkmark Drag amplitude and clock period
                otras.fig1_prbs_mostrar(5)=x1;



            case 9 %Botones
                if (ix==1) %cancel
                    otras.fig1_tipo(2)=otras.fig1_tipo(1);
                    otras.fig1_escala=otras.fig1_escala_old;
                    otras.open_aux_window=0;
                    initGraphics(width,height,otras);
                elseif (ix==2) %ok
                    otras.open_aux_window=0;
                    otras.fig1_tipo(1)=otras.fig1_tipo(2);
                    initGraphics(width,height,otras);
                end
            end




%-------------------------------------------------------------------------
function draw_settings_fig2(otras)
%draw_settings_fig2

        par1='{x_{\\min}}'; %tmin
        par2='{x_{\\max}}'; %tmax
        par3='{y_{\\min }}'; %ymin
        par4='{y_{\\max }}'; %ymax


        campo1=otras.fig2_escala(1);
        campo2=otras.fig2_escala(2);
        campo3=otras.fig2_escala(3);
        campo4=otras.fig2_escala(4);

        P=5;
        campo1_min=campo1-P;
        campo1_max=campo1+P;
        campo2_min=campo2-P;
        campo2_max=campo2+P;
        campo3_min=campo3-P;
        campo3_max=campo3+P;
        campo4_min=campo4-P;
        campo4_max=campo4+P;



                plotoption noframe;
                   plotoption nolabel;
                   plotoption nomargin;
```

```
                plotoption math;
              subplotstyle('plotbg', FillColor=color_fondo_vauxiliar);
                subplotstyle('controlbg', FillColor=color_fondo_vauxiliar);
                subplotstyle('plotmargin',FixedControlVPos=true);



                %Dimensiones de la zona donde aparecerá la información
                xmin=0;xmax=20;
                ymin=0;ymax=20;
                scale([0 20 0 20]);


        %Título de la ventana
        estilo_texto1                                                                    =
fontset(Font=face_font,Bold=true,Size=size_font,Color='black');
        text(10,20,'\nAutocorrelation - Figure settings',estilo_texto1)
               text('\n\n');


                % Radio buttons de selección de escala fija o variable

                settabs([8,12,10]);
                button('              Scale:\tLocked\tVariable',otras.fig2_tipo(2),
'radiobutton', '', 1);
                text('\n');


                %Campos xmin, xmax, ymin, ymax
                sep2campos='   value:\t00000000000\tvalueXX:\t00000000000';

                seplabel1='$\\quad';
                seplabel22='$\t$\\quad\\,\\,';


                settabs(sep2campos);
                labels = [seplabel1,par1,seplabel22,par2,'$\t'];
                textfield(labels, '%.1g%.1g', real([campo1, campo2]), '', 2)
                slider('            \t             ',             [campo1;campo2],
[campo1_min,campo1_max;campo2_min,campo2_max], '', 'gg', 2);



                settabs(sep2campos);
                labels = [seplabel1,par3,seplabel22,par4,'$\t'];
                textfield(labels, '%.1g%.1g', real([campo3, campo4]), '', 3)
                slider('            \t             ',             [campo3;campo4],
[campo3_min,campo3_max;campo4_min,campo4_max], '', 'gg', 3);


                text('\n')

                % Checkmarks de opciones
                settabs([2,30]);
                button('\t         Plot         continuous         autocorrelation
function',otras.fig2_ACF_mostrar(1), 'ckeckmark', '', 4);

                button('\t          Plot          discrete          autocorrelation
function',otras.fig2_ACF_mostrar(2), 'ckeckmark', '', 5);

                button('\t  Only  plot  positive  lags',otras.fig2_ACF_mostrar(3),
'ckeckmark', '', 6);

                if otras.tipo_signal==1
```

```
                        button('\t Plot clock period grid',otras.fig2_ACF_mostrar(4),
'ckeckmark', '', 7);
                  end









                  text('\n')

                  %Botones Default, Cancel y OK
                  settabs([10,10,10]);
                  pushbutton('\t Cancel \t OK \t', 'rg', 8);

            text('\n')


%-----------------------------------------------------------------------
function (otras)=drag_settings_fig2(id,ix,nb,x1,height,width,otras)
%drag_settings_fig2

            switch id

            case 2
                  if otras.fig2_tipo(2)==1 %Si es escala fija
                        if             otras.fig2_ACF_mostrar(1)==0             &&
otras.fig2_ACF_mostrar(2)==1 %If it is only plotted the discrete ACF
                              if x1-floor(x1) < 0.5
                                    x1=floor(x1);
                              else
                                    x1=ceil(x1);
                              end
                        end

                        if nb==1 %xmin
                              if x1 < otras.fig2_escala(2)
                                    if otras.fig2_ACF_mostrar(3)==1 && x1<0
                                          x1=0;
                                    end
                                    otras.fig2_escala(1)=x1;

                              end

                        elseif nb==2 %xmax
                              if x1 > otras.fig2_escala(1)
                                    otras.fig2_escala(2)=x1;
                              end
                        end
                  elseif       otras.fig2_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');
                  end
            case 3
                  if otras.fig2_tipo(2)==1 %Si es escala fija
                        if nb==1 %ymin
                              if x1 < otras.fig2_escala(4)
                                    otras.fig2_escala(3)=x1;
                              end
                        elseif nb==2 %ymax
                              if x1 > otras.fig2_escala(3)
                                    otras.fig2_escala(4)=x1;
```

```
                                end
                        end
                elseif      otras.fig2_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');

                end


        end

%---------------------------------------------------------------------
function (otras)=mouseup_settings_fig2(id,ix,nb,x1,height,width,otras,par,signals)
%mouseup_settings_fig2

        switch id
        case 1 % Radio buttons para selección de la escala
                otras.fig2_tipo(2)=x1;

                if x1==2 %Variable
                        otras=calcular_escalas_variables(par,signals,otras);
                end

        case 2
                if otras.fig2_tipo(2)==1 %Si es escala fija
                        if              otras.fig2_ACF_mostrar(1)==0              &&
otras.fig2_ACF_mostrar(2)==1 %If it is only plotted the discrete ACF
                                if x1-floor(x1) < 0.5
                                        x1=floor(x1);
                                else
                                        x1=ceil(x1);
                                end
                        end

                        if nb==1 %xmin
                                if x1 < otras.fig2_escala(2)
                                        if otras.fig2_ACF_mostrar(3)==1 && x1<0
                                                x1=0;
                                        end

                                        otras.fig2_escala(1)=x1;

                                end
                        elseif nb==2 %xmax
                                if x1 > otras.fig2_escala(1)
                                        otras.fig2_escala(2)=x1;
                                end
                        end
                elseif      otras.fig2_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');
                end
        case 3
                if otras.fig2_tipo(2)==1 %Si es escala fija
                        if nb==1 %ymin
                                if x1 < otras.fig2_escala(4)
                                        otras.fig2_escala(3)=x1;
                                end
                        elseif nb==2 %ymax
                                if x1 > otras.fig2_escala(3)
                                        otras.fig2_escala(4)=x1;
                                end
                        end
```

```
                elseif      otras.fig2_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                    dialog('Warning:  The axis limits can not be configured if
the scale type is variable');

                end

            case 4 %Checkmark Plot Continuous ACF
                otras.fig2_ACF_mostrar(1)=x1;

            case 5 %Checkmark Plot Discrete ACF
                otras.fig2_ACF_mostrar(2)=x1;

            case 6 %Checkmark Only plot positive lags
                otras.fig2_ACF_mostrar(3)=x1;
                otras.fig2_escala(1)=0;
                otras=calcular_escalas_variables(par,signals,otras);

            case 7 %Checkmark Plot clock period grid
                otras.fig2_ACF_mostrar(4)=x1;



            case 8 %Botones
                if (ix==1) %cancel
                    otras.fig2_tipo(2)=otras.fig2_tipo(1);
                    otras.fig2_escala=otras.fig2_escala_old;
                    otras.open_aux_window=0;
                    initGraphics(width,height,otras);
                elseif (ix==2) %ok
                    otras.open_aux_window=0;
                    otras.fig2_tipo(1)=otras.fig2_tipo(2);
                    initGraphics(width,height,otras);
                end
            end


%-------------------------------------------------------------------------
function draw_settings_fig3(otras)
%draw_settings_fig3

    par1='{x_{\\min}}'; %tmin
    par2='{x_{\\max}}'; %tmax
    par3='{y_{\\min }}'; %ymin
    par4='{y_{\\max }}'; %ymax


    campo1=otras.fig3_escala(1);
    campo2=otras.fig3_escala(2);
    campo3=otras.fig3_escala(3);
    campo4=otras.fig3_escala(4);

    P=5;
    campo1_min=campo1-P;
    campo1_max=campo1+P;
    campo2_min=campo2-P;
    campo2_max=campo2+P;
    campo3_min=campo3-P;
    campo3_max=campo3+P;
    campo4_min=campo4-P;
    campo4_max=campo4+P;
```

```
                plotoption noframe;
                   plotoption nolabel;
                   plotoption nomargin;
                   plotoption math;

                subplotstyle('plotbg', FillColor=color_fondo_vauxiliar);
                   subplotstyle('controlbg', FillColor=color_fondo_vauxiliar);
                   subplotstyle('plotmargin',FixedControlVPos=true);



                   %Dimensiones de la zona donde aparecerá la información
                   xmin=0;xmax=20;
                   ymin=0;ymax=20;
                   scale([0 20 0 20]);


         %Título de la ventana
          estilo_texto1                                                                    =
fontset(Font=face_font,Bold=true,Size=size_font,Color='black');
          text(10,20,'\nPower spectrum - Figure settings',estilo_texto1)
                text('\n\n');

                   % Radio buttons de selección del tamaño de la figura
                   settabs([13,11,10]);
                   button('     Size:\tSmall\tMedium',otras.fig3_size,  'radiobutton',
'', 13);

                   % Radio buttons para mostrar leyenda
                   settabs([13,11,10]);
                   button('    Legend:\tNo\tYes',otras.fig3_legend, 'radiobutton', '',
14);


                text('\n');
                   % Radio buttons de selección de escala fija o variable

                   settabs([13,11,10]);
                   button('             Scale:\tLocked\tVariable',otras.fig3_tipo(2),
'radiobutton', '', 1);

                   % Radio buttons de selección de las unidades del eje X

                   settabs([13,11,13]);
                   button('                  X    axis    units:\tHz\trad/s\tHarmonic
number',otras.fig3_Xunits, 'radiobutton', '', 10);


                   % Radio buttons de selección del tipo de escala del eje X

                   settabs([13,11,10]);
                   button('    X axis scale:\tLinear\tLogarithmic',otras.fig3_Xscale,
'radiobutton', '', 11);

                % Radio buttons de selección del tipo de escala del eje Y

                   settabs([13,11,10]);
                   button('    Y axis scale:\tLinear\tLogarithmic',otras.fig3_Yscale,
'radiobutton', '', 12);


                   %Campos xmin, xmax, ymin, ymax
                   sep2campos='   value:\t00000000000\tvalueXX:\t00000000000';
```

```
                    seplabel1='$\\quad';
                    seplabel22='$\t$\\quad\\,\\,';


                    settabs(sep2campos);
                    labels = [seplabel1,par1,seplabel22,par2,'$\t'];
                    textfield(labels, '%.1g%.1g', real([campo1, campo2]), '', 2)
                    slider('            \t            ',            [campo1;campo2],
[campo1_min,campo1_max;campo2_min,campo2_max], '', 'gg', 2);


                    settabs(sep2campos);
                    labels = [seplabel1,par3,seplabel22,par4,'$\t'];
                    textfield(labels, '%.1g%.1g', real([campo3, campo4]), '', 3)
                    slider('            \t            ',            [campo3;campo4],
[campo3_min,campo3_max;campo4_min,campo4_max], '', 'gg', 3);


            text('\n');


                    % Checkmarks de opciones
                    settabs([2,30]);

                    button('\t    Plot    power    spectrum    of    the    continuous
signal',otras.fig3_spectrum_mostrar(1), 'ckeckmark', '', 4);

                    button('\t    Plot    power    spectrum    of    the    discrete
signal',otras.fig3_spectrum_mostrar(2), 'ckeckmark', '', 5);

                    button('\t           Plot           double-sided           power
spectrum',otras.fig3_spectrum_mostrar(3), 'ckeckmark', '', 6);

                    button('\t          Highlight          effective          frequency
band',otras.fig3_spectrum_mostrar(4), 'ckeckmark', '', 7);


                    button('\t    Drag    effective    frequency    band    and    maximum
power',otras.fig3_spectrum_mostrar(6), 'ckeckmark', '', 7);


                    if otras.fig3_spectrum_mostrar(2)==1
                          button('\t         Highlight         contribution         to
aliasing',otras.fig3_spectrum_mostrar(5), 'ckeckmark', '', 8);
                    end









                    text('\n')

                    %Botones Default, Cancel y OK
                    settabs([20,10,10]);
                    pushbutton('\t Cancel \t OK \t', 'rg', 9);

                    text('\n')

%-----------------------------------------------------------------------
function (otras)=drag_settings_fig3(id,ix,nb,x1,height,width,otras)
```

```
%drag_settings_fig3

            switch id

            case 2
                  if otras.fig3_tipo(2)==1 %Si es escala fija

                              if x1-floor(x1) < 0.5
                                    x1=floor(x1);
                              else
                                    x1=ceil(x1);
                              end



                        if nb==1 %xmin
                              if x1 < otras.fig3_escala(2)
                                    otras.fig3_escala(1)=x1;
                              end

                        elseif nb==2 %xmax
                              if x1 > otras.fig3_escala(1)
                                    otras.fig3_escala(2)=x1;
                              end
                        end
                  elseif      otras.fig3_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');
                  end
            case 3
                  if otras.fig3_tipo(2)==1 %Si es escala fija
                        if nb==1 %ymin
                              if x1 < otras.fig3_escala(4)
                                    otras.fig3_escala(3)=x1;
                              end
                        elseif nb==2 %ymax
                              if x1 > otras.fig3_escala(3)
                                    otras.fig3_escala(4)=x1;
                              end
                        end
                  elseif      otras.fig3_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');

                  end


            end



%--------------------------------------------------------------------
function
(otras,par,signals)=mouseup_settings_fig3(id,ix,nb,x1,height,width,otras,par,signals)
%mouseup_settings_fig3

            switch id
            case 1 % Radio buttons para selección de la escala
                  otras.fig3_tipo(2)=x1;


                  if x1==2 %Variable
                        otras=calcular_escalas_variables(par,signals,otras);
```

```
                end


        case 2
                if otras.fig3_tipo(2)==1 %Si es escala fija

                        if x1-floor(x1) < 0.5
                                x1=floor(x1);
                        else
                                x1=ceil(x1);
                        end


                        if nb==1 %xmin
                                if x1 < otras.fig3_escala(2)
                                        otras.fig3_escala(1)=x1;
                                end

                        elseif nb==2 %xmax
                                if x1 > otras.fig3_escala(1)
                                        otras.fig3_escala(2)=x1;
                                end
                        end
                elseif          otras.fig3_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');
                end
        case 3
                if otras.fig3_tipo(2)==1 %Si es escala fija
                        if nb==1 %ymin
                                if x1 < otras.fig3_escala(4)
                                        otras.fig3_escala(3)=x1;
                                end
                        elseif nb==2 %ymax
                                if x1 > otras.fig3_escala(3)
                                        otras.fig3_escala(4)=x1;
                                end
                        end
                elseif          otras.fig3_tipo(2)==2 && (nb==1 || nb==2)%Si es escala
variable
                        dialog('Warning:  The axis limits can not be configured if
the scale type is variable');

                end

        case 4 %Checkmark Plot power spectrum of continuous signal
                otras.fig3_spectrum_mostrar(1)=x1;


        case 5 %Checkmark Plot power spectrum of discrete signal
                otras.fig3_spectrum_mostrar(2)=x1;

                if x1==0
                        otras.fig3_spectrum_mostrar(5)=0;
                end


        case 6 %Checkmark Plot Two-sided spectrum
                otras.fig3_spectrum_mostrar(3)=x1;
                par.negative_harmonics=x1;

                if x1==1
                        otras.fig3_Xscale=1; %La escala en el eje X debe ser lineal
```

```
                        initGraphics(width,height,otras);    %Esta    sentencia    es
necesaria para que tenga efecto el tipo de escala seleccionado
                    end

                    (signals,otras)=compute_signals(par,otras);
                    otras=calcular_escalas_variables(par,signals,otras);

              case 7 %Checkmark Highlight effective frequency band
                    otras.fig3_spectrum_mostrar(4)=x1;

              case 8 %Checkmark Highlight contribution to aliasing
                    otras.fig3_spectrum_mostrar(5)=x1;



              case 9 %Botones
                    if (ix==1) %cancel
                          otras.fig3_tipo(2)=otras.fig3_tipo(1);
                          otras.fig3_escala=otras.fig3_escala_old;
                          otras.open_aux_window=0;
                          initGraphics(width,height,otras);
                    elseif (ix==2) %ok
                          otras.open_aux_window=0;
                          otras.fig3_tipo(1)=otras.fig3_tipo(2);
                          initGraphics(width,height,otras);
                    end


              case 10 % Radio buttons para selección unidades eje X
                    otras.fig3_Xunits=x1;


                    otras=calcular_escalas_variables(par,signals,otras);


              case 11 % Radio buttons para selección tipo escala eje X
                    otras.fig3_Xscale=x1;


                    if otras.fig3_Xscale==2 %Si se tiene escala logaritmica el espectro
debe ser de un solo lado.
                          par.negative_harmonics=0;
                          otras.fig3_spectrum_mostrar(3)=0;
                    end

                    otras=calcular_escalas_variables(par,signals,otras);

                    initGraphics(width,height,otras); %Esta senetencia es necesaria para
que tenga efecto el tipo de escala seleccionado


              case 12 % Radio buttons para selección tipo escala eje Y
                    otras.fig3_Yscale=x1;

                    otras=calcular_escalas_variables(par,signals,otras);

                    initGraphics(width,height,otras); %Esta senetencia es necesaria para
que tenga efecto el tipo de escala seleccionado

              case 13 % Radio buttons para selección del tamaño de la ventana
                    otras.fig3_size=x1;

                    initGraphics(width,height,otras); %Esta sentencia es necesaria para
que tenga efecto el tipo de escala seleccionado
```

```
            case 14 % Radio buttons para mostrar leyenda
                  otras.fig3_legend=x1;


            end


%-------------------------------------------------------------------------
function draw_settings_fig5(otras,par,signals)
%draw_settings_fig2


                plotoption noframe;
                   plotoption nolabel;
                   plotoption nomargin;
                   plotoption math;

                subplotstyle('plotbg', FillColor=color_fondo_vauxiliar);
                   subplotstyle('controlbg', FillColor=color_fondo_vauxiliar);
                   subplotstyle('plotmargin',FixedControlVPos=true);



                   %Dimensiones de la zona donde aparecerá la información
                   xmin=0;xmax=20;
                   ymin=0;ymax=20;
                   scale([0 20 0 20]);


            %Título de la ventana
      estilo_texto1 = fontset(Font=face_font,Bold=true,Size=size_font,Color='black');
      text(10,20,'\nSampling time, Ts',estilo_texto1)


      if otras.tipo_signal==1 %PRBS
            text('\n\n');
            % Ts
            D=floor(par.prbs_Tsw/par.prbs_Ts);

            ind=obtener_ind_D(D);
            if ind==1
                  ind_min=1;
            else
                  ind_min=ind-1;
            end
            if ind==26
                  ind_max=26;
            else
                  ind_max=ind+1;
            end

            settabs([5,25]);
            textfield('     D ', '%d', D, 'b', id=1);
            slider('\t',ind, [ind_min,ind_max],'', 'b', id=3);

            cadena=sprintf('\n     • Ts = Tck/D = %.2g', par.prbs_Tsw/D);

            if par.time_units==1
                  cadena=[cadena,' min     '];
            elseif par.time_units==0
                  cadena=[cadena,' s     '];
            elseif par.time_units==-1
                  cadena=[cadena,' ds     '];
            elseif par.time_units==-2
                  cadena=[cadena,' cs     '];
```

```
            elseif par.time_units==-3
                  cadena=[cadena,' ms     '];
            end

            text(cadena)
            Ts=convert2seconds(par.prbs_Ts,par.time_units);
            cadena=sprintf('     • Nyquist frequency = 1/(2Ts) = %.2g Hz', 1/(2*Ts));

            text(cadena)

            cadena=sprintf('                    •    Signal    samples    =    %d',
floor((signals.prbs_T*par.prbs_Nc)/(par.prbs_Tsw/D)));

            text(cadena);

            text('\n\n')


            %Botones Default, Cancel y OK
            settabs([10,10,10]);
            pushbutton('\t Cancel \t OK \t', 'rg', 8);

            text('\n')


      else %msine
            text('\n');

            % Ts
            Ts=par.msine_T/par.msine_N;


            cadena=sprintf('\n     • Ts = T/N = %.3g', Ts);

            if par.time_units==1
                  cadena=[cadena,' min     '];
            elseif par.time_units==0
                  cadena=[cadena,' s     '];
            elseif par.time_units==-1
                  cadena=[cadena,' ds     '];
            elseif par.time_units==-2
                  cadena=[cadena,' cs     '];
            elseif par.time_units==-3
                  cadena=[cadena,' ms     '];
            end

            text(cadena)
            Ts=convert2seconds(Ts,par.time_units);
            cadena=sprintf('     • Nyquist frequency = 1/(2Ts) = %.2g Hz', 1/(2*Ts));

            text(cadena)

            cadena=sprintf('     • Signal samples = %d', par.msine_N*par.msine_Nc);


            text(cadena);

            text('\n')


            %Botones Default, Cancel y OK
            settabs([15,10]);
            pushbutton('\t OK \t', 'g', 8);

            text('\n')
```

```
	end


%---------------------------------------------------------------------
function ind=obtener_ind_D(D)
%obtener_ind_D
	val_D=[1,2,5,8,10,15,16,20,25,30,32,35,40,45,50,55,60,64,65,70,75,80,85,90,95,10
0];
	L=length(val_D);

	for h=1:L
		if D==val_D(h)
			ind=h;
		end
	end

%---------------------------------------------------------------------
function D=obtener_valor_D(ind)
%obtener_valor_D

	val_D=[1,2,5,8,10,15,16,20,25,30,32,35,40,45,50,55,60,64,65,70,75,80,85,90,95,10
0];

	D=val_D(ind);



%---------------------------------------------------------------------
function
(otras,par,signals)=drag_settings_fig5(id,ix,nb,x1,height,width,otras,par,signals)
%drag_settings_fig5

		switch id

		case 1
			%Comprobar si D es un múltiplo de 5 o una potencia de 2, en el rango
[1,100]
			a=rem(x1,5);
			b=log2(x1);

			if  (x1>=1 && x1 <101) && (a==0 || (abs(b-round(b))<10^-10))

				par.prbs_Ts=par.prbs_Tsw/x1;
				(signals,otras)=compute_signals(par,otras);
				otras=calcular_escalas_variables(par,signals,otras);
			else
				dialog('WARNING: D has to be an integer, power of 2 or multiple
of 5, in the range [1,100]: 1, 2, 4, 5, 8, 10, 15, 16, 20,...');
			end

		case 3

			D=floor(par.prbs_Tsw/par.prbs_Ts);
			ind=obtener_ind_D(D);


			if abs((x1-floor(x1))) >= 0.5
				ind=ceil(x1);
			else
				ind=floor(x1);
			end

			if ind<1
```

```
                        ind=1;
                end

                if ind>26
                        ind=26;
                end



                D=obtener_valor_D(ind);
                par.prbs_Ts=par.prbs_Tsw/D;
                (signals,otras)=compute_signals(par,otras);
                otras=calcular_escalas_variables(par,signals,otras);









        end

%-----------------------------------------------------------------------
function
(otras,par,signals)=mouseup_settings_fig5(id,ix,nb,x1,height,width,otras,par,signals)
%mouseup_settings_fig2

        switch id

        case 3
                D=floor(par.prbs_Tsw/par.prbs_Ts);
                ind=obtener_ind_D(D);


                if abs((x1-floor(x1))) >= 0.5
                        ind=ceil(x1);
                else
                        ind=floor(x1);
                end

                if ind<1
                        ind=1;
                end

                if ind>26
                        ind=26;
                end


                D=obtener_valor_D(ind);
                par.prbs_Ts=par.prbs_Tsw/D;
                (signals,otras)=compute_signals(par,otras);
                otras=calcular_escalas_variables(par,signals,otras);



        case 8 %Botones
                if otras.tipo_signal==1
                        if (ix==1) %cancel
                                par.prbs_Ts=par.prbs_Ts_old;
                                otras.open_aux_window=0;
                                initGraphics(width,height,otras);
                        elseif (ix==2) %ok
```

```
                                otras.open_aux_window=0;
                                par.prbs_Ts_old=par.prbs_Ts;
                                initGraphics(width,height,otras);
                        end
                else
                        if (ix==1) %ok
                                otras.open_aux_window=0;
                                par.prbs_Ts_old=par.prbs_Ts;
                                initGraphics(width,height,otras);
                        end
                end

                (signals,otras)=compute_signals(par,otras);
                otras=calcular_escalas_variables(par,signals,otras);
        end


%-------------------------------------------------------------------------
% Window layout
%-------------------------------------------------
function initGraphics(width,height,otras)
%initGraphics


        figurestyle('plotmargin',  CenteredLabelHeight=1.5,  Left=4,  Right=1,  Top=0.2,
Bottom=1);
        figurestyle('labelfont', Size=size_font);
        figurestyle('legendfont', Size=size_font);
        %figurestyle('titlefont', Size=size_font+4);
        figurestyle('tickfont', Size=size_font, Font=face_font);
        figurestyle('controlfont', Size=size_font, Font=face_font);
        %figurestyle('winbg', FillColor=otras.color_fondo_ventana);
        %figurestyle('plotbg', FillColor='white');
        %figurestyle('controlbg', FillColor=otras.color_fondo_ventana);

        if otras.tipo_signal==1
                fig1='Title';
cor_fig1=[3,48,1,5];
        else
                fig1='Title';
cor_fig1=[4,24,1,5];
        end
        if otras.tipo_signal==1
                fig2='Parameters';
        cor_fig2=[4,55,7,40];
                fig3='Information';
        cor_fig3=[4,66,40,58];
        else
                fig2='Parameters';
        cor_fig2=[4,55,7,53];
                fig3='Information';
        cor_fig3=[4,66,53,71];
        end
        fig4='Shift register';
        cor_fig4=[0,0,0,0];

        fig5='Title Figure Signal';                                     cor_fig5=[78,84,1,3];
        fig6='Settings Figure Signal';
        cor_fig6=[156,158,1,3];
        fig7='Signal';
        cor_fig7=[73,159,3,29];

        fig8='Title Figure ACF';
        cor_fig8=[77,99,29,31];
```

```
       fig9='Settings Figure ACF';
       cor_fig9=[156,158,29,31];
       fig10='Autocorrelation Function';
       cor_fig10=[73,159,31,57];

       if otras.fig3_size==1
              fig11='Title Figure Power Spectrum';
       cor_fig11=[77,93,58,60];
              fig12='Settings Figure Power Spectrum';
       cor_fig12=[156,158,58,60];
              fig13='Power Spectrum';
       cor_fig13=[73,159,59,85];
       else
              fig11='Title Figure Power Spectrum';
       cor_fig11=[77,93,29,31];
              fig12='Settings Figure Power Spectrum';
       cor_fig12=[156,158,29,31];
              fig13='Power Spectrum';
       cor_fig13=[73,159,31,85];
       end


       %fig11='Title Figure DFT Magnitude';                cor_fig11=[77,93,58,60];
       %fig12='Settings Figure DFT Magnitude';
       cor_fig12=[156,158,58,60];
       %fig13='DFT Magnitude';
       cor_fig13=[73,159,59,85];


       if otras.fig3_size==1
              _plots
=[fig1,'\n',fig2,'\n',fig3,'\n',fig4,'\n',fig5,'\n',fig6,'\n',fig7,'\n',fig8,'\n',fig9
,'\n',fig10,'\n',fig11,'\n',fig12,'\n',fig13];

       coord0=[cor_fig1;cor_fig2;cor_fig3;cor_fig4;cor_fig5;cor_fig6;cor_fig7;cor_fig8;
cor_fig9;cor_fig10;cor_fig11;cor_fig12;cor_fig13];
       else
              _plots
=[fig1,'\n',fig2,'\n',fig3,'\n',fig4,'\n',fig5,'\n',fig6,'\n',fig7,'\n',fig11,'\n',fig
12,'\n',fig13];

       coord0=[cor_fig1;cor_fig2;cor_fig3;cor_fig4;cor_fig5;cor_fig6;cor_fig7;cor_fig11
;cor_fig12;cor_fig13];
       end

       %Virtual auxiliar window
       if otras.open_aux_window==1
              figAW='Virtual auxiliar window';

              if ismac()==1
                     cor_figAW=[31,74,9,53];
              else
                     cor_figAW=[9,47,9,52];
                     if otras.tipo_aux_window=='3'
                            cor_figAW=[2,58,9,69];
                     end
              end
              _plots =[_plots,'\n',figAW];
              coord0=[coord0;cor_figAW];
       end

    %Resize the window
       if (width>=width0 && height>=height0)
           if width>1920
                     width=1920;
```

```
          end
        if height>1080
                height=1080;
          end
          Kx=width/width0;
          Ky=height/height0;
   else
          Kx=1;
          Ky=1;
   end

   coord1=[Kx*coord0(:,1:2),Ky*coord0(:,3:4)];

   %Restricciones en las posiciones de las figuras
   %---------------

   %fig2='Parameters';
   %Mantener anchura y altura.
h=2;
   w0=abs(coord0(h,2)-coord0(h,1)); %SA


   coord1(h,2)=coord1(h,1)+w0; %SA
   coord1(h,3)=coord0(h,3); %SA
   coord1(h,4)=coord0(h,4); %SA

   %fig1='Title'; %SA
   %Mantener anchura y altura.
   %Alineada a la izquierda y sobre la fig2='Parameters';            %SA

h=1; k=2; %SA
   w0=abs(coord0(h,2)-coord0(h,1)); %SA
   h0=abs(coord0(h,4)-coord0(h,3)); %SA

   if ismac()==1
          desp=0;
   else
          desp=0;
   end
   coord1(h,1)=coord1(k,1)-desp; %SA
   coord1(h,2)=coord1(h,1)+w0; %SA

   coord1(h,4)=coord1(k,3);
   coord1(h,3)=coord1(h,4)-h0;

   %fig3='Information'; %SA
   %Mantener anchura y altura.
   %Alineada a la izquierda y debajo de la fig2='Parameters';        %SA

h=3; k=2; %SA
   w0=abs(coord0(h,2)-coord0(h,1)); %SA
   h0=abs(coord0(h,4)-coord0(h,3)); %SA

   if ismac()==1
          desp=0;
   else
          desp=0;
   end
   coord1(h,1)=coord1(k,1)-desp; %SA
   coord1(h,2)=coord1(h,1)+w0; %SA
   coord1(h,3)=coord1(k,4);
   coord1(h,4)=coord1(h,3)+h0;


   %fig7='Signal';
```

```
      %Alineada a la izquierda respecto a la fig3='Information';
    h=7; k=3; %SA
      if ismac()==1
            desp=0;
      else
            desp=2;
      end
      coord1(h,1)=coord1(k,2)+desp; %SA


      %fig5='Title Figure Signal';
      %Mantener anchura y altura
      %Alineada a la izquierda respecto a la fig7='Signal';

    h=5; k=7; %SA
      w0=abs(coord0(h,2)-coord0(h,1)); %SA
      h0=abs(coord0(h,4)-coord0(h,3)); %SA

      if ismac()==1
            desp=0;
      else
            desp=4.3;
      end
      coord1(h,1)=coord1(k,1)+desp; %SA
      coord1(h,2)=coord1(h,1)+w0; %SA
      coord1(h,4)=coord1(k,3); %SA
      coord1(h,3)=coord1(h,4)-h0; %SA

      %fig6='Settings Figure Signal';
      %Mantener anchura y altura
      %Alineada a la derecha respecto a la fig7='Signal';

    h=6; k=7; %SA
      w0=abs(coord0(h,2)-coord0(h,1)); %SA
      h0=abs(coord0(h,4)-coord0(h,3)); %SA

      if ismac()==1
            desp=0;
      else
            desp=1;
      end
      coord1(h,2)=coord1(k,2)-desp; %SA
      coord1(h,1)=coord1(h,2)-w0; %SA
      coord1(h,4)=coord1(k,3); %SA
      coord1(h,3)=coord1(h,4)-h0; %SA

if otras.fig3_size==1

      %fig10='ACF';
      %Alineada a la izquierda respecto a la fig3='Information';
    h=10; k=3; %SA
      if ismac()==1
            desp=0;
      else
            desp=2;
      end
      coord1(h,1)=coord1(k,2)+desp; %SA



      %fig8='Title Figure ACF';
      %Mantener anchura y altura
      %Alineada a la izquierda respecto a la fig10='Autocorrelation Function';

    h=8; k=10; %SA
```

```
      w0=abs(coord0(h,2)-coord0(h,1)); %SA
      h0=abs(coord0(h,4)-coord0(h,3)); %SA

      if ismac()==1
            desp=0;
      else
            desp=4.4;
      end
      coord1(h,1)=coord1(k,1)+desp; %SA
      coord1(h,2)=coord1(h,1)+w0; %SA
      coord1(h,4)=coord1(k,3); %SA
      coord1(h,3)=coord1(h,4)-h0; %SA

      %fig9='Settings Figure ACF';
      %Mantener anchura y altura
      %Alineada a la derecha respecto a la fig10='Autocorrelation Function';

   h=9; k=10; %SA
      w0=abs(coord0(h,2)-coord0(h,1)); %SA
      h0=abs(coord0(h,4)-coord0(h,3)); %SA

      if ismac()==1
            desp=0;
      else
            desp=1;
      end
      coord1(h,2)=coord1(k,2)-desp; %SA
      coord1(h,1)=coord1(h,2)-w0; %SA
      coord1(h,4)=coord1(k,3); %SA
      coord1(h,3)=coord1(h,4)-h0; %SA

end


      %fig13='Power spectrum';
      %Alineada a la izquierda respecto a la fig3='Information';
   h=13; k=3; %SA
      if otras.fig3_size==2
          h=h-3;
      end
      if ismac()==1
            desp=0;
      else
            desp=2;
      end
      coord1(h,1)=coord1(k,2)+desp; %SA


      %fig11='Title Figure Power Spectrum'; (o DFT Magnitude)
      %Mantener anchura y altura
      %Alineada a la izquierda respecto a la fig13='Power Spectrum'; (o DFT Magnitude)

   h=11; k=13; %SA
      if otras.fig3_size==2
            h=h-3;k=k-3;
      end
      w0=abs(coord0(h,2)-coord0(h,1)); %SA
      h0=abs(coord0(h,4)-coord0(h,3)); %SA

      if otras.fig3_type==1
            if ismac()==1
                  desp=0;
            else
                  desp=3.9;
            end
```

```
      else
            if ismac()==1
                  desp=0;
            else
                  desp=3.4;
            end
      end
      coord1(h,1)=coord1(k,1)+desp; %SA
      coord1(h,2)=coord1(h,1)+w0; %SA
      coord1(h,4)=coord1(k,3); %SA
      coord1(h,3)=coord1(h,4)-h0; %SA

      %fig12='Settings Figure Power Spectrum'; (o DFT Magnitude)
      %Mantener anchura y altura
      %Alineada a la derecha respecto a la fig13='Power Spectrum'; (o DFT Magnitude)

   h=12; k=13; %SA
      if otras.fig3_size==2
            h=h-3;k=k-3;
      end

      w0=abs(coord0(h,2)-coord0(h,1)); %SA
      h0=abs(coord0(h,4)-coord0(h,3)); %SA

      if ismac()==1
            desp=0;
      else
            desp=1;
      end
      coord1(h,2)=coord1(k,2)-desp; %SA
      coord1(h,1)=coord1(h,2)-w0; %SA
      coord1(h,4)=coord1(k,3); %SA
      coord1(h,3)=coord1(h,4)-h0; %SA

      if otras.open_aux_window==1
            %figAW='Virtual auxiliar window';
            %Mantener anchura y altura
            %Misma altura que la figura fig2='Parameters';
             h=14;k=2;
            if otras.fig3_size==2
                  h=h-3;
            end

            w0=abs(coord0(h,2)-coord0(h,1)); %SA
            h0=abs(coord0(h,4)-coord0(h,3)); %SA

            coord1(h,2)=coord1(h,1)+w0; %SA
            coord1(h,3)=coord1(k,3);
            coord1(h,4)=coord1(h,3)+h0; %SA
      end









      _plotpos = coord1;
   subplots(_plots);
   subplotpos(_plotpos);


@}
```

```
%====================================
```