SYNCHROINFO JOURNAL

www.media-publisher.eu/synchroinfo-journal

OBJECT PROGRAMMING IN MATLAB RADIO ENGINEERING APPLICATIONS

Yuri S. Shinakov [1], Marcelo Sampaio de Alencar [2],

[1] Institute of Radio and Information Systems (IRIS), Vienna, Austria; Shinakov@media-publisher.eu

[2] Institute for Advanced Studies in Communications, Campina Grande, Brazil; https://orcid.org/0000-0002-2849-1644

ABSTRACT

System objects integrate well with the MATLAB programming language, both when writing simple functions that work interactively in the command window, and when creating large applications. In contrast to MATLAB functions, system objects automatically manage state information, data indexing, and buffering, which is especially useful for iterative calculations or streaming data processing. Such an organization of calculations makes it possible to efficiently process long data arrays. System objects support fixed-point arithmetic and C-code generation from MATLAB and SIMULINK. With system objects, you can generate code for a computer or for an external device. System objects can be used in SIMULINK models in the form of MATLAB function blocks. The issues of the methodology for using MATLAB program modules (files), which are usually called system objects, are discussed. Each such object is a simulation computer model of a real radio engineering device or radio engineering system. Examples of such system objects are given and the effectiveness of their use in the educational process at a technical university is illustrated.

KEYWORDS: *Object-oriented software modules, Dynamic systems, Iterative computations, Streaming data processing, Radio signals resolution.*

DOI: 10.36724/2664-066X-2023-9-1-2-8

Received: November 12, 2022
Accepted: December 20, 2022

Citation: Yuri S. Shinakov, Marcelo Sampaio de Alencar, "Object programming in MATLAB radio engineering applications," *Synchroinfo Journal* **2023**, vol. 9, no. 1, pp. 2-8.

1. Introduction

System objects in MATLAB are a set of object-oriented software modules that provide the implementation of algorithms for the functioning of the corresponding objects [1]. They extend the capabilities of MATLAB, allowing you to simulate dynamic systems represented by algorithms for the functioning of these systems in time [2-4]. System objects integrate well with the MATLAB programming language, both when writing simple functions that work interactively in the command window, and when creating large applications.

2. Using an existing system object

Consider the use of an already existing system object using the example of calculating the discrete Fourier transform (DFT) [5] from a real array of input data:

```
% Create an array of input data
Fs = 1000; % sampling rate
T = 1/Fs; % Sampling interval
L = 1024; % Length of array of signal samples
t = (0:L-1)*T; % Sample moment vector
% The array of input data is the sum of two sinusoids
X = 0.7*sin(2*pi*50*t.') + sin(2*pi*120*t.');
```

To use a system object that exists in MATLAB, you must first describe it by typing the appropriate line in the command window. For example, to implement the DFT algorithm, we type:

```
>> H = dsp.FFT % Define an FFT object whose parameter values % are taken by default.
```

The notation used here is:
H is the assigned name of this object, used in the current session of work;
dsp is the name of the MATLAB package in which this system object is stored;
FFT is the name of the system object.
Now, to perform the DFT on the array X existing in the MATLAB workspace, we type the command in the command window:

```
>> Y = step(H, X); % Process input X by system object H
```

The result of the DFT execution is placed in the Y array. Here step is the name of the X data processing method by the H system object.
Typically, the default values of system object parameters must be changed before use. In this case, the description of this object should be entered as a string:

```
>> H = dsp.FFT('PropertyName', PropertyValue, ...)
```

Here PropertyName is the name of the parameter of the system object, PropertyValue is the value of this parameter; the required pairs are listed in random order.
For each system object that exists in MATLAB, a general list of all parameters can be called up by typing the help command in the command window. For example, for dsp.FFT:

```
>> help dsp.FFT

FFT properties:
FFTImplementation – choice of FFT implementation
BitReversedOutput – order of output elements relative to input
Normalize – permission to divide butterfly outputs by 2
FFTLengthSource – FFT length source
FFTLength – FFT length as an integer
```


Synchroinfo Journal **2023**, *vol. 9*, no. 1

Each of the system object parameters has a list of allowed values. For example, for dsp.FFT in the above list, the parameters are represented by hyperlinks that, by a simple click, you can get:

FFTImplementation

Can take one of the following values: [{'Auto'} | 'Radix-2' | 'FFTW'].

The default value is "Auto".

If 'Radix-2' is selected, the FFT length must be a power of 2.

BitReversedOutput

Can take one of the following values: ['true' | 'false'].

The default value is 'false', which means the frequency indexes are in linear order.

Setting it to 'true' causes the frequency indices to be output in reverse order; the value of FFTLength must be a power of 2. This parameter is applicable if FFTImplementation is set to 'Auto' or 'Radix-2'.

normalize

Can take one of the following values: ['true' | 'false'].

The default value is 'false', which means there is no normalization.

When set to 'true', the output of each FFT butterfly is divided by 2.

FFTLengthSource

Can take one of the following values: [{'Auto'} | 'Property'].

When set to 'Auto', the length of the FFT is equal to the number of rows in the input array.

FFTLength

This setting is applicable if FFTLengthSource is set to 'Property'. Its default value is 64, or a number in the form of a power of 2 for fixed-point input, or 'true' for BitReversedOutput, or 'Radix-2' for FFTImplementation.

You can call the values of only one required parameter in the command window. For example, for the Normalize parameter:

>> help dsp.FFT.Normalize

Takes one of the following values: ['true' | 'false'].

The default value is 'false', which means there is no normalization.

When set to 'true', the output of each FFT butterfly is divided by 2.

The step name defines the main operation mode of the system object, which implements the key input data processing algorithm available for this system object.

3. Benefits of using system objects

System objects use only two commands when programming data processing - a description when creating an object and a step command to perform processing with this object. This separation of description and execution allows the creation of many of the same type, reusable variants of an object, each with a different purpose. Using this approach eliminates repeated checking and validation of input, makes it easy to use loops in programming, and improves overall quality. In contrast, a MATLAB function validates the input values each time the function is called.

System objects are especially useful for processing streaming data, where a continuous stream is broken into segments and each segment is processed in turn in the same way. There is no need to keep a large amount of data in memory. Using streaming data allows you to use simplified programs that effectively use loops.

4. Application example: study of signal resolution

Laboratory work on the discipline "Statistical theory of radio engineering systems". Three types of radio signal are considered; 1) a radio pulse with a rectangular envelope; 2) a radio pulse with linear frequency modulation (LFM, chirp [3]) and a rectangular envelope; 3) a ck of coherent chirp pulses with a rectangular envelope.

In this lab, students are asked to experimentally investigate the resolution of some radio signals used in radar. To do this, it is first necessary to obtain realizations of these signals (more precisely, their complex envelopes), and then calculate and construct surfaces of uncertainty functions from these realizations. Based on the resulting graphic images of these two-dimensional functions, students must evaluate the resolution of each signal. Thus, the study should be carried out in three stages: 1) simulation of the signal

implementation, 2) calculation of the uncertainty function, 3) construction of an image of the surface of this function. Based on the graphic representation of the uncertainty function, students must evaluate the numerical value of the resolution, one of the parameters of the studied signals that is important for radar.

In the information base of the MATLAB system, there is a library named "phased", which contains system objects for modeling various signals. This library has system objects "RectangularWaveform" and "LinearFMWaveform". The first of them generates samples of the complex envelope of the signal, which is a single rectangular pulse or a pack of such pulses; the second one simulates readings of the complex envelope of a chirp signal with a rectangular envelope and a pack of such signals. For each of these system objects, various parameters are defined, the values of which can be changed. Examples of some of these options are shown below.

4.1. Single radio pulse with a rectangular envelope

The parameters of this system object are:

```
fs = 1e6; % Sample rate [Hz]
prf = 1e4; % Pulse repetition rate [Hz]
pw = 10e-6; % Pulse duration [s]
np = 5; % Number of pulses in a burst
```

To call the required system object and assign numerical values to its parameters, just enter the command:

```
hrect = phased.RectangularWaveform('SampleRate',fs,'PRF',prf,'PulseWidth',pw);
```

To create samples of this signal on one repetition period, it is enough to enter the command

```
x = step(hrect);
```

The column vector x of samples of the studied signal is further used to calculate the two-dimensional function of the uncertainty of this signal on the plane delay – Doppler frequency and plot its surface.

```
ambgfun(x, hrect.SampleRate, hrect.PRF); % Uncertainty function (Fig. 1).
```

To obtain numerical values of the resolution parameters, it is recommended to plot sections of the uncertainty function along the corresponding axes.

```
ambgfun(x,hrect.SampleRate,hrect.PRF,'Cut','Doppler'); % Cross section Doppler
ambgfun(x,hrect.SampleRate,hrect.PRF,'Cut','Delay'); % cross section delay
```

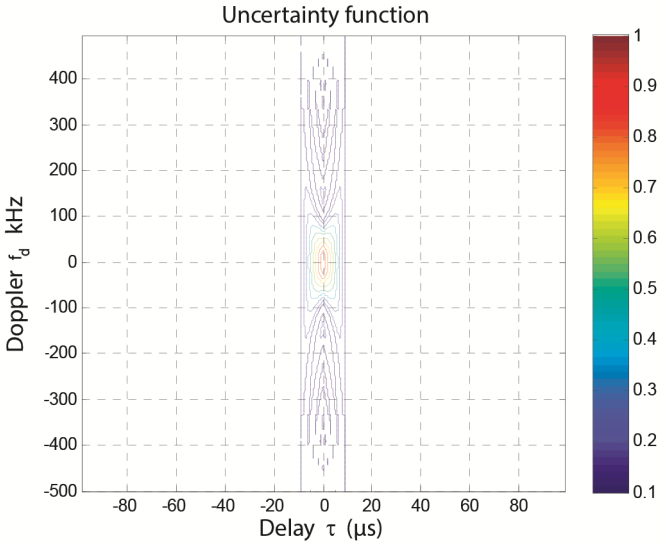

Fig. 1: Single radio pulse Uncertainty function

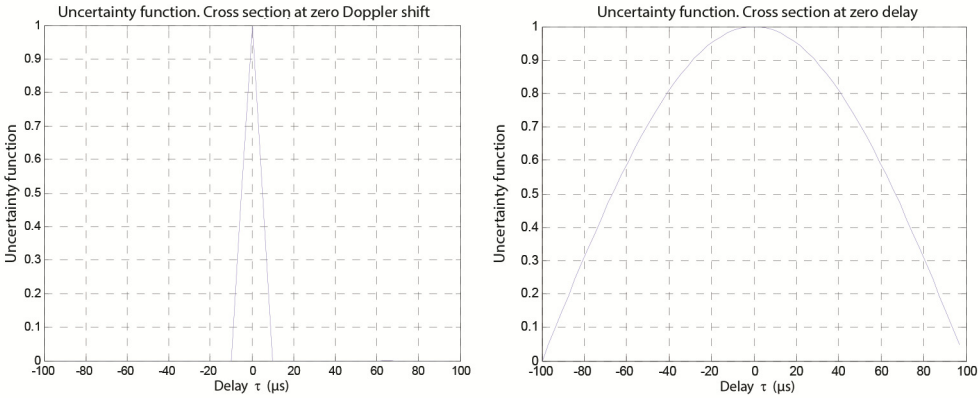

Fig. 2: Sections of the Single Radio Pulse Uncertainty Function

The result of the study: the resolution of this signal in terms of delay time is 10 μs, and in terms of Doppler frequency - 100 kHz.

4.2. Single pulse with linear frequency modulation and rectangular envelope

As in paragraph 4.1, the experimental study is reduced to entering the following MATLAB commands:

```
hlfm = phased.LinearFMWaveform('SampleRate',fs,'SweepBandwidth',1/pw,...
'PRF',prf,'PulseWidth',5*pw)
x = step(hlfm);
ambgfun(x,hlfm.SampleRate,hlfm.PRF,'Cut','Doppler');% Doppler Cross Section
ambgfun(x,hlfm.SampleRate,hlfm.PRF,'Cut','Delay'); % Section Delay
```

a. Virtualiza on deployment planning, design, management

Administration of the infrastructure should be more rigorously organized than that of a physical one because it is far more important. In fact, the virtual infrastructure should be seen as a full machine in and of itself, housed in a single physical location with strict security constraints. Planning contributes to assuring security and adherence to all essential organizational policies. To maximize security while minimizing costs, organizations should address security from the outset of the systems development life cycle. Organizations must follow basic ICT security recommendations and practice such as keeping software up to date using host-based firewalls, antivirus, and IDS, to mention a few.

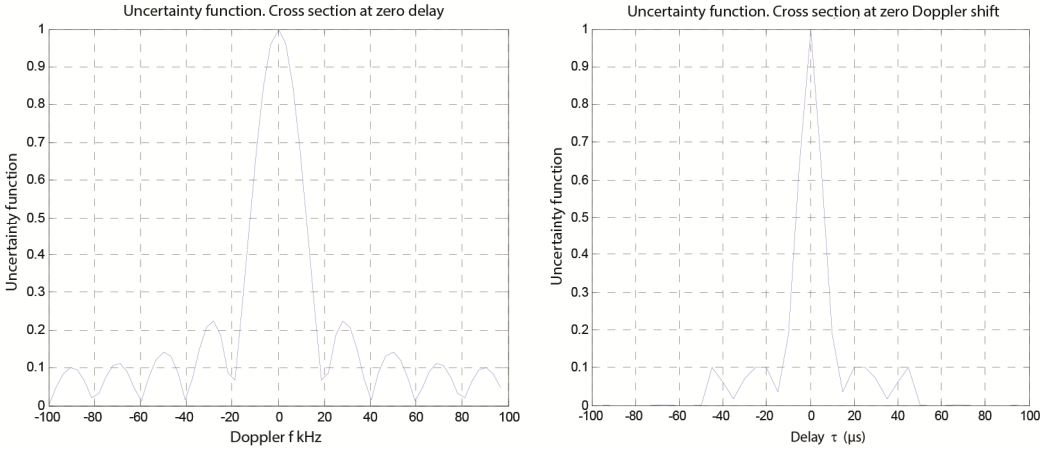

Fig. 3: Chirp-pulse uncertainty function cross sections

Note that the frequency change band "SweepBandwidth" of this signal is taken equal to the width of the spectrum of a single rectangular pulse, and the duration of the signal is increased by 5 times. The resolution of the chirp signal with these parameters turned out to be approximately 10 μs in delay and 20 kHz in Doppler frequency.

4.3. Packet of coherent chirp pulses with rectangular envelope

If this item is executed in the same session as item 2, then there is no need to call the “phased.LinearFMWaveform” system object again. It is enough to change the numerical values of the required parameters of the already used system object "hlfm". To do this, you can declare a new version of this object:

```
>>release(hlfm);
```

and change the values of the required parameters:

```
>>hlfm.NumPulses = 5;
```

Further, the study is performed in the same sequence as in the previous paragraphs:

```
x = step(hlfm);
ambgfun(x,hlfm.SampleRate,hlfm.PRF).
```

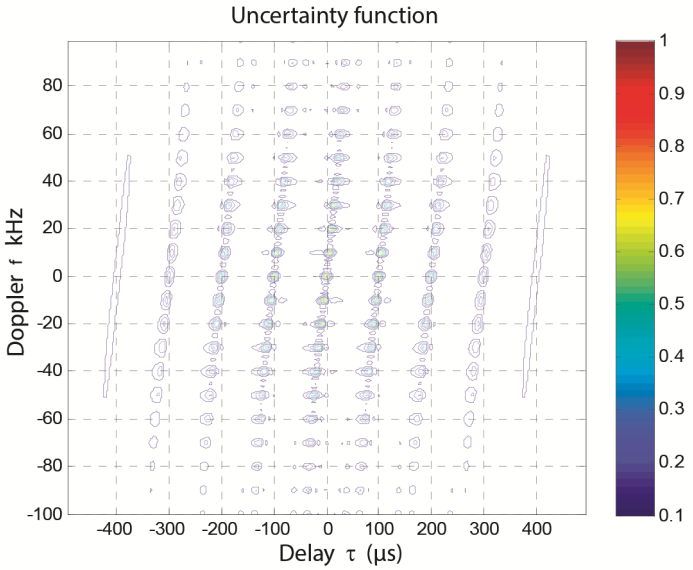

Fig. 4: Chirp Burst Uncertainty Function

5. Conclusion

System objects use only two commands when programming data processing - a description when creating an object and a step command to perform processing with this object. This separation of description and execution allows the creation of many of the same type, reusable variants of an object, each with a different purpose. Using this approach eliminates repeated control and validation of input, makes it easy to use loops in programming and improves the overall quality of calculations. Recall that MATLAB functions, on the other hand, must validate the input parameter values each time that function is called.

System objects are especially useful for processing streaming data, where a continuous stream is broken into segments and processed interactively. There is no need to keep a large amount of data in memory. Using streaming data allows you to use simplified programs that effectively use loops. The MATLAB information base contains all the necessary information, sufficient for any user to independently create software system objects of the real physical objects he needs.

It is useful to note one more general fact illustrated by this work: the MATLAB system allows the researcher to obtain results that cannot be obtained by other means with a small investment of time.

REFERENCES

[1] MATLAB Release 2011b/Help/Product Help/ Phased Array System Toolbox/System Objects/Waveforms.

[2] C. Voloşencu, editor. MATLAB Applications in Engineering [Internet]. IntechOpen; 2022. Available from: http://dx.doi.org/10.5772/intechopen.91588

[3] N. Dinesh Kumar, "Radiation Power Pattern Distortion Analysis Using MATLAB for MST Radar System," MATLAB Applications in Engineering. IntechOpen, Feb. 02, 2022. doi: 10.5772/intechopen.97637.

[4] R. Ren, XE Sun, L. Hu, "A new method for hosting and sharing MATLAB Web App.," Sci Rep. 2022 Dec 14, no. 12(1), p. 21645. doi: 10.1038/s41598-022-26165-3. PMID: 36517636; PMCID: PMC9750978.

[5] L. Borovyk, L. Traskovetska, O. Valchuk, I. Basaraba, and I. Gashchuk, "Application of the MatLab Opportunities During the Study of the Fourier Series by Future Border Guard Officers," *Revista Romaneasca Pentru Educatie Multidimensionala*, no. 14(4), pp. 372-393, 2022. https://doi.org/10.18662/rrem/14.4/646.